Multifaceted Analysis of Deep Convolutional Neural Networks and Novel Fourier Modules

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Julia Grabinski

aus Hagen

Mannheim, 2025

Dekan: Prof. Dr. Claus Hertling, Universität Mannheim
Referent: Prof. Dr.-Ing. Margret Keuper, Universität Mannheim
Korreferent: Prof. Dr. Simone Schaub-Meyer, Technische Universität Darmstadt
Korreferent: Prof. Dr. Ivo Ihrke, Universität Siegen

Tag der mündlichen Prüfung: 14.07.2025

Erklärung zum Einsatz von Generativen Textmodellen

Erklärung zum Einsatz von Generativen Textmodellen In der Erstellung dieser Arbeit wurden die generative Textmodelle (auch bekannt als large language models (LLMs)) OpenAI ChatGPT, Google Gemini, LanguageTool, sowie Grammarly eingesetzt, um die schriftliche Präsentation dieser Thesis zu verbessern. In diesem Zusammenhang wurden einzelne und bereits formulierte Sätze und Textpassagen sprachlich und grammatikalisch überarbeitet, umformuliert, strukturiert und/oder von diesen Modellen zusammengefasst. Die erstellten Texte wurden zudem manuell geprüft und häufig weiter überarbeitet. Die Modelle wurden nicht dazu eingesetzt, neue Inhalte zu generieren. Insbesondere wurden alle in dieser Thesis eingeführten Methoden, Experimente und Resultate eigenständig – beziehungsweise mit oder von den jeweils gekennzeichneten Autoren – erarbeitet.

Abstract

The increasing reliance on neural networks in everyday applications underscores a critical challenge: ensuring their robustness and reliability beyond idealized conditions. Evaluating vision classification models solely through clean accuracy and spatial perspectives is insufficient. Thus, we employ a multifaceted analysis of robust models and leverage Fourier theory to enhance robustness, efficiency, and our fundamental understanding of convolutional neural networks.

This thesis explores the interplay between adversarial robustness, confidence calibration, efficiency and sampling artifacts through the lens of Fourier Theory in convolutional neural networks.

We first demonstrate that adversarially robust models exhibit significantly lower overconfidence compared to their non-robust counterparts. We demonstrate that even subtle modifications to fundamental network components can significantly improve confidence calibration, highlighting the power of architectural design. Building on this, we investigate aliasing effects in robust models, revealing that they downsample more effectively than standard models, leading to reduced aliasing. To quantify this phenomenon, we introduce a novel aliasing measure and show its connection to catastrophic overfitting in FGSM adversarial training, inspiring an early stopping criterion based on our aliasing measure.

Leveraging these discoveries, we present *Frequency Low Cut (FLC) Pooling* and *Aliasing and Sinc Artifact-free Pooling (ASAP)*, novel Fourier-domain downsampling methods designed to be inherently aliasing-free. These techniques contribute to enhanced native robustness and improved adversarial training stability, effectively addressing catastrophic overfitting in FGSM adversarial training.

Building upon our Fourier-domain investigations, we present *Neural Implicit Frequency Filters (NIFFs)*, enabling efficient large convolutions. By leveraging neural implicit functions for weight learning and efficient Fourier-domain convolution, NIFFs provide a feasible and fair comparison to large spatial convolutions. Using NIFFs, we analyse learned kernel size preferences, revealing insights that facilitate more efficient network design. We demonstrate that optimal feature extraction often requires kernels larger than the typical 3×3, with 9×9 kernels being predominately learned by the network, especially on ImageNet-1k.

Our multifaceted analysis and findings contribute to a deeper understanding of adversarial robustness, confidence calibration, sampling artifacts and the role of Fourier theory in convolutional neural network design, paving the way for more robust and efficient deep learning models.

Zusammenfassung

Die zunehmende Abhängigkeit von neuronalen Netzen in alltäglichen Anwendungen unterstreicht eine kritische Herausforderung: die Gewährleistung ihrer Robustheit und Zuverlässigkeit über idealisierte Bedingungen hinaus. Die Bewertung von Bildklassifikationsmodellen allein durch die Klassifikationsgenauigkeit und räumliche Perspektiven ist unzureichend. Daher führen wir eine facettenreiche Analyse robuster Modelle durch und nutzen Konzepte der Fourier-Theorie, um Robustheit, Effizienz und unser grundlegendes Verständnis von Convolutional Neural Networks (CNNs) zu verbessern.

Diese Dissertation untersucht das Zusammenspiel zwischen adversärer Robustheit, Kalibrierung, Effizienz und Sampling-Artefakten mithilfe von Konzepten der Fourier-Theorie in Convolutional Neural Networks.

Wir zeigen zunächst, dass adversär robuste Modelle eine signifikant geringere Überkonfidenz aufweisen als ihre nicht-robusten Gegenstücke. Wir demonstrieren, dass selbst kleine Modifikationen an grundlegenden Netzwerkkomponenten die Kalibrierung erheblich verbessern können, was die Wichtigkeit eines durchdachten Architekturdesignes Neuronaler Netze unterstreicht. Aufbauend darauf untersuchen wir Aliasing-Effekte in robusten Modellen und stellen fest, dass robuste Modelle effektiver downsamplen als Standardmodelle, was zu weniger Aliasing Artefakten führt. Um dieses Phänomen zu quantifizieren, führen wir eine neuartige Aliasing Messung ein. Wir zeigen, dass ein Anstieg des Aliasing mit Catastrophic Overfitting im FGSM-adversären Training in Verbindung steht. Daher nutzen wir unsere Aliasing-Messung, um das adversäre Training anzuhalten, bevor Catastrophic Overfitting auftritt.

Aufbauend auf diesen Entdeckungen präsentieren wir *Frequency Low Cut (FLC) Pooling* und *Aliasing and Sinc Artifact-free Pooling (ASAP)*, neuartige Fourier-Domain-Downsampling-Methoden, die aliasing-frei downsamplet. Unsere neuen Methoden tragen zu einer verbesserten nativen Robustheit und einer erhöhten Stabilität des adversären Trainings bei und adressieren effektiv Catastrophic Overfitting im FGSM-adversären Training.

Wir setzen unsere erfolgreiche Integration der Fourier-Theorie fort und präsentieren *Neural Implicit Frequency Filters (NIFFs)*, die effiziente, große Faltungen ermöglichen. Durch die Nutzung neuronaler impliziter Funktionen um die Gewichte zu Lernen und der Effizienz der Fourier-Domain-Faltungen bieten NIFFs einen praktikablen und fairen Vergleich zu großen räumlichen Faltungen. Mit unseren NIFFs analysieren wir gelernte Kernelgrößenpräferenzen und gewinnen Erkenntnisse, die ein effizienteres Netzwerkdesign ermöglichen. Wir zeigen, dass optimale Feature-Extraktion oft größere Kernel als die typischen 3×3 erfordert, wobei 9×9 Kernel auf Datensätzen wie ImageNet-1k ausreichende Ergebnisse liefern.

Unsere facettenreiche Analyse und unsere neuen Methoden tragen zu einem tieferen Verständnis von adversärer Robustheit, Kalibrierung, Sampling-Artefakten und der Rolle der Fourier Theorie im Convolutional-Neural-Network-Design bei und ebnen den Weg für robustere und effizientere Deep-Learning-Modelle.

Acknowledgements

I'm deeply grateful to all the brilliant and kind people I had the pleasure of meeting during my PhD journey, with special thanks to those who supported me both professionally and personally.

First and foremost, I would like to thank my two advisors, Margret and Janis Keuper, for their extensive and diverse guidance, patience, and support throughout my research. I was fortunate to benefit from both of your perspectives, which complemented each other perfectly, allowing me to gain invaluable knowledge throughout my journey. Thank you, Margret, for encouraging me to begin my PhD journey and for your constant availability and support. You motivated me to keep going and pushed me to reach my full potential. Your guidance helped me find my way and place within the academic world. Without you, I would not be submitting this thesis today. Thank you so much. Thank you, Janis, for your unwavering support and calming presence, which helped keep me grounded and more at ease throughout my journey. Your clear perspective, along with your valuable and honest feedback, guided me throughout my PhD. I'm truly grateful to both of you for mentoring me and preparing me so well for the academic world.

I was fortunate to conduct visiting research at Tampere University, Finland, in the computer vision department led by Esa Rathu during the final year of my PhD. I would like to thank Esa and his group for their incredible hospitality and for fostering a warm and welcoming atmosphere in the lab. This experience allowed me to explore entirely new topics and broaden my horizons, for which I am truly grateful. A special thanks to Ville and Nicklas for introducing me to Finnish culture and exposing me to fascinating and diverse research topics that, while different from my prior work, were truly intriguing.

Special thanks to my collaborators with whom I had the pleasure of working: Paul, Shashank, Steffen and Teja. Working together on various projects has been an invaluable learning experience, and I deeply appreciate the collective effort that led to our successful publications. Some collaborations have grown into cherished friendships, and each of you has left a lasting impact on my growth as a researcher.

I would like to extend my gratitude to all my colleagues, past and present, across the various departments I've had the pleasure of working with. Our engaging discussions, retreats, and seminars created a research environment that was both enjoyable and supportive. I am also deeply thankful for the conference opportunities that allowed me to meet so many wonderful and inspiring individuals.

Thank you to my friends outside academia for your understanding and keeping me grounded and smiling through the challenges of this journey. Additionally, I want to extend my gratitude to all my sports and dancing buddies, who have been essential in helping me maintain a healthy work-life balance.

Last but not most importantly I would like to thank Chris, my partner, best friend and greatest supporter during this whole journey. Your encouragement and ability to bring joy in both the good times and the challenging moments have been invaluable. I also extend my deepest thanks to my family for their unconditional love and support, which has been a constant source of strength.

Contents

List of Abbreviations

1D	**one-d**imensional
2D	**two-d**imensional
3D	**three-d**imensional
AA	**A**uto **A**ttack
Acc	**Acc**uracy
Acc@1	Top **1 Acc**uracy
Acc@5	Top **5 Acc**uracy
AM	**A**liasing **M**easure
APGD	**A**daptive **P**rojected **G**radient **D**escent
ASAP	**A**liasing and **S**inc **A**rtifact-free **P**ooling
AUC	**A**rea **U**nder the **C**urve
AT	**A**vdersarial **T**raining
BIM	**B**asic **I**terative **M**ethod
CIFAR	**C**anadian **I**nstitute **F**or **A**dvanced **R**esearch
CNN	**C**onvolutional **N**eural **N**etwork
CNNs	**C**onvolutional **N**eural **N**etworks
CW	**C**arlini and **W**agner Attack
DDN	**D**ecoupling **D**irection and **N**orm
DF	**D**eep**F**ool
DFT	**D**iscrete **F**ourier **T**ransform
ECE	**E**xpected **C**alibration **E**rror
FAB	**F**ast **A**daptive **B**oundary Attack
FCN	**F**ully **C**onnected **N**etwork
FFT	**F**ast **F**ourier **T**ransform
FGSM	**F**ast **G**radient **S**ign **M**ethod
FLC	**F**requency **L**ow **C**ut
FM	**F**eature **M**ap
FT	**F**ourier **T**ransform

i.e.	**i**d **e**st
MLP	**M**ulti **L**ayer **P**erceptron
MLPs	**M**ulti **L**ayer **P**erceptrons
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
NIFF	**N**eural **I**mplicit **F**requency **F**ilter
NIFFs	**N**eural **I**mplicit **F**requency **F**ilters
PCA	**P**rincipal **C**omponent **A**nalysis
PGD	**P**rojected **G**radient **D**escent
PRN	**P**reAct-**R**es**N**et
RA	**R**and**A**ug
ReLU	**Re**ctified **L**inear **U**nit
RN	**R**es**N**et
ROC	**R**eceiver **O**perating **C**haracteristic
SGD	**S**tochastic **G**radient **D**escent
SiLU	**Si**gmoid **L**inear **U**nit
SVHN	**S**treet **V**iew **H**ouse **N**umbers
VGG	**V**isual **G**eometry **G**roup
ViT	**Vi**sion **T**ransformer
vs.	versus
W.l.o.G.	**W**ithout loss **o**f **G**enerality
WRN	**W**ide-**R**es**N**et
w.r.t.	**w**ith **r**espect **t**o

Chapter 1

Introduction

Contents

In recent years, neural networks have become deeply embedded in our daily lives, powering vision-based applications across a vast spectrum, from facial recognition systems in mobile devices [Guo & Zhang (2019)] and real-time object detection in autonomous vehicles [Bochkovskiy et al. (2020)] to advanced medical imaging [Zhou et al. (2019)] and smart city surveillance [Chen et al. (2019); Reed et al. (2023); Gupta (2024)]. The profound integration of neural networks into so many facets of modern life raises a critical requirement: these networks must not only perform well under known conditions but must also be robust and reliable under distributions shift [Zhang (2019); Hendrycks & Dietterich (2019); Gavrikov et al. (2025)]. As their applications extend further into safety-critical domains, ensuring robustness and reliability becomes paramount; otherwise, we risk deploying systems that may fail in unexpected ways [Amodei et al. (2016)] or offer potential for attackers to fool the network [Goodfellow et al. (2015); Madry et al. (2018); Andriushchenko et al. (2020)]. To train more robust models, commonly used techniques include incorporating additional data [Gowal et al. (2021b); Rade & Moosavi-Dezfooli (2021); Rebuffi et al. (2021)] or employing more sophisticated training methods [Madry et al. (2018); Zhang et al. (2019b)]. Figure 1.1 highlights a significant advantage of robust models, *i.e.* adversarially trained models, observed in our work: they maintain accuracy under various image distortions, such as shifts, blurring, and aliasing. Specifically, while the non-robust model shows reduced confidence under shifts and blurring, it is completely misled by aliasing and adversarial attacks, resulting in high-confident incorrect predictions. In contrast, the robust model remains accurate and less overconfident across all tested distortions. We demonstrate in Chapter 4 that robust models are often less overconfident while non-robust models make high-confident incorrect predictions when attacked. Furthermore, Figure 1.1 reveals another interesting observation: adversarial attacks and aliasing produce visually similar artifacts that

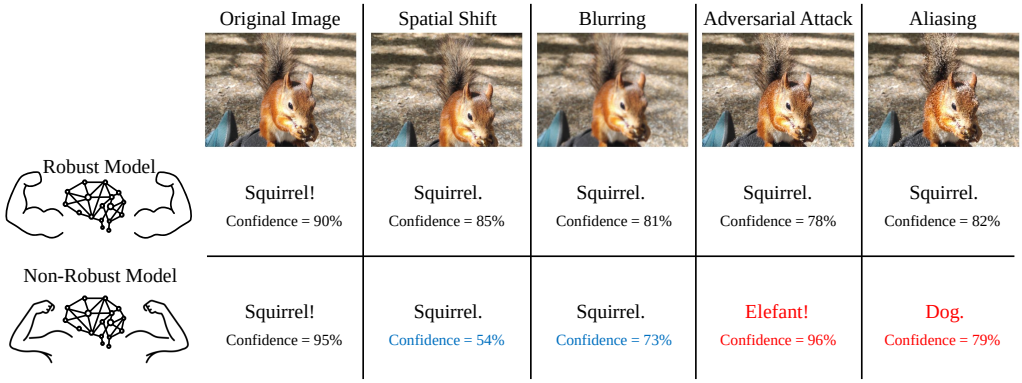


FIGURE 1.1: **Challenges for Robust and Non-Robust Deep Neural Networks.** While robust models correctly predict the class label, non-robust models get fooled easily and fail with high confidence (high confidence predictions are marked by "!" otherwise "."), overestimating their certainty in incorrect classifications.

both significantly degrade network performance. We find in Chapter 5 that adversarial training not only improves model accuracy and reliability, but also results in desirable properties such as reduced aliasing after downsampling. However, training robust models via *e.g.* adversarial training often leads to more complex training tasks, resulting in longer training times as shown in Table 6.7 in Chapter 6. Furthermore, as task complexity increases, larger model sizes are often preferred [Croce et al. (2021)] leading to increased computational costs and memory consumption.

The development of deep learning models, especially in the case of making them more robust via adversarial training or trough more training data, demands significant computational resources, with the training of large networks requiring extensive GPU hours and substantial energy consumption. This leads to higher operational costs and leading to significant environmental problems [Patterson et al. (2021); Dhar (2020)]. Consequently, there is a critical need for approaches that make neural networks architecture design [Tan & Le (2021)] and training processes [Menghani (2023)] more resource-efficient to reduce computational load without sacrificing performance.

Achieving efficiency begins with a fundamental examination of network architectures and their design. By analysing how different architectural choices affect computational demands and performance, we can identify ways to build neural networks that maximize resource use while maintaining high accuracy and reliability. Efficient network architectures should prioritize optimization techniques that reduce parameter counts, accelerate convergence, and ensure generalization across diverse datasets and tasks. These architectural considerations are essential for advancing the field toward more sustainable and adaptable neural networks that can efficiently meet the demands of real-world applications.

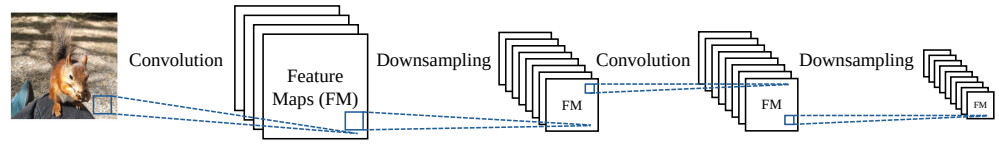
Achieving this balance requires a deep understanding of what a neural network is fundamentally doing. At its core, a neural network receives input signals and interprets them to produce meaningful outputs tailored to specific tasks, whether that means classifying an image [He et al. (2016a)], detecting objects [Bochkovskiy et al. (2020)] or reconstructing a scene [Mildenhall et al. (2021); Yu et al. (2021)]. An abstract representation of the feature extraction process in a classification CNN is depicted in Figure 1.2 (a). In this thesis, we focus on improving the two components, convolutions and downsampling, to create efficient and sustainable systems and optimise not only for performance but also for efficiency and resilience.

This thesis leverages mathematical principles like aliasing-free sampling, Fourier transformations, and filtering, allowing modern computer vision systems to sidestep common pitfalls like poor robustness and reliability. We reveal that robust models are less overconfident than their non-robust counterparts in Chapter 4 and learn to downsample with fewer artifacts in Chapter 5. By revisiting core concepts from signal processing, we systematically analyse and refine classification convolutional neural networks (CNNs), enhancing their efficiency, robustness, and capacity to process larger inputs while minimizing computational overhead. One key aspect of this thesis is the application of the Sampling Theorem [Shannon (1949)], through which we demonstrate the intrinsic connection between aliasing and adversarial vulnerabilities in CNNs in Chapter 5. By integrating classical signal processing techniques with advanced neural architectures, we develop more effective downsampling methods in Chapter 6 that enhance native adversarial robustness and improve adversarial training to be more effective and efficient. Therefore, we leverage the Fourier domain, guaranteeing aliasing-free downsampling as depicted in Figure 1.2 (b). Subsequently, in Chapter 7, by leveraging the Convolution Theorem [Bracewell & Kahn (1966); Forsyth & Ponce (2003)], we implement convolution in the frequency domain (Figure 1.2 (c)) to systematically study kernel sizes in CNNs and make large convolutions more computationally efficient. This approach not only strengthens the theoretical foundations of CNNs but also yields practical benefits by optimizing computational efficiency and reducing memory consumption. By bridging the gap between traditional signal processing and deep learning, this thesis provides a principled framework for designing more robust and resource-efficient neural networks.

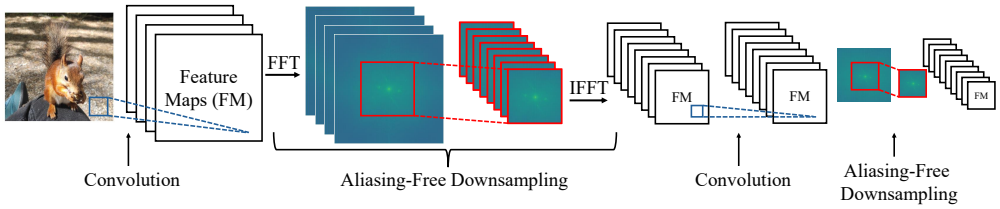
In summary, combining foundational mathematical principles, like Fourier Theory, with modern deep learning techniques provides a strong basis for developing robust, reliable, and efficient computer vision models. This combination supports the development of safer, more sustainable, and computationally efficient solutions, enabling their use in fields such as healthcare, astrophysics, and autonomous systems.

1.1 Contribution Overview

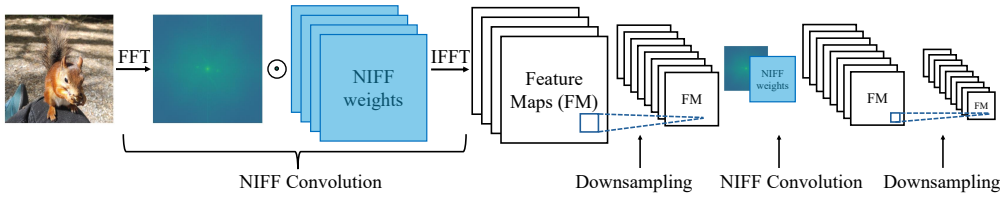
This thesis is organized into three parts, one preliminary part which includes foundations and related work in Chapters 2 and 3, respectively, followed by two main parts, Parts I and II. In the first chapter of the preliminary part, in Chapter 2, we introduce the foundational concepts relevant to this thesis. Following, in Chapter 3, we review related research important for the approaches presented in the subsequent chapters. In the first main part, Part I, we present two key studies investigating the relationship between model robustness and both model reliability and aliasing artifacts. First, in Chapter 4, we demonstrate that adversarially trained classification models exhibit reduced overconfidence. Second, in Chapter 5, we show that these models learn to downsample with fewer aliasing artifacts. In the second main part, Part II, we introduce novel Fourier modules, partially derived from the studies conducted in Part I. Chapter 6 aims at enhancing model robustness and improving adversarial training efficiency via aliasing-free downsampling in the Fourier domain. Additionally, we enable a comprehensive analysis of classification network design, specifically focusing on the preferred learned kernel size within a network in Chapter 7 by leveraging the Fourier domain again. Finally, in Chapter 8, we summarize the key contributions of this thesis, discuss its limitations and implications, and outline potential directions for future research.



(a) Feature Extraction CNN.



(b) Aliasing-Free Feature Extraction.



(c) NIFF Feature Extraction.

FIGURE 1.2: **Our Fourier Modules.** (a) Standard feature extraction process in a CNN. (b) Feature extraction with our aliasing-free downsampling presented in Chapter 6. (c) Feature extraction with our NIFF Convolutions presented in Chapter 7.2.1.

In the following, we give a brief overview on the main chapters of this thesis. In Part I, we focus on analysing adversarially robust models from different perspectives and find that adversarial robustness benefits more than robustness, namely reliability and sampling.

In Chapter 4, we show that robustness is not only important for its own sake but also comes with a beneficial side effect. We demonstrate that non-robust models tend to be highly overconfident, *i.e.*, prone to high-confidence incorrect predictions, which can lead to critical failures in real-world scenarios. In contrast, we find that adversarially robust models exhibit less overconfidence, making them more suitable for deployment not only due to their robustness but also because of their more reliable behaviour.

In Chapter 5, we continue analysing robust models and reveal that robust models downsample more effectively. We measure the amount of aliasing after downsampling and demonstrate that non-robust models induce higher levels of aliasing, while adversarially trained models intrinsically learn cleaner downsampling, *i.e.*, with less aliasing. Additionally, we show that catastrophic overfitting in fast gradient sign methods (FGSM) adversarial training (AT) is strongly correlated with increased aliasing after downsampling. As a result, we introduce a more efficient stopping criterion based on our aliasing measure, which is more efficient than using the more computationally expensive Projected Gradient Descent (PGD) adversaries.

In Part II, we introduce novel Fourier modules for both analysing and implementing models from an alternative perspective. Specifically, we propose downsampling and convolution in the Fourier domain.

Based on the findings of Chapter 5, we introduce aliasing-free downsampling in

the Fourier domain, called Frequency Low Cut Pooling, short FLC Pooling in the second part, in Chapter 6. FLC Pooling reduces the risk of catastrophic overfitting, thereby enhancing FGSM AT. Furthermore, we extend FLC Pooling to not only eliminate aliasing but also reduce sinc-interpolation artifacts, introducing aliasing- and sinc-artifact pooling (ASAP). As a result, our downsampling methods improve native robustness and enhance fast FGSM AT as well as PGD training. Additionally, our downsampling methods not only increase robustness but also provide higher reliability in terms of prediction confidence. We demonstrate that robust networks trained with our improved downsampling methods are less overconfident, showing reduced confidence in incorrect predictions compared to the baseline, which enhances their suitability for real-world applications.

Finally, in Chapter 7, we not only leverage the Fourier domain for aliasing-free downsampling but also introduce convolutions in the Fourier domain to learn spatially infinite convolutions. To achieve this efficiently, we use a neural implicit function to limit the number of learnable parameters, calling our method Neural Implicit Frequency Filters (NIFF). We use NIFF to study the practical receptive field that a network prefers to learn to make models more efficient and effective for a given network architecture and dataset. Further, NIFF scales effectively with increased input and convolution size due to the neural implicit function and the convolution applied in the frequency domain. The trick of calculating the convolution in the frequency domain is already applied in the 1D case for State-Space models [Poli et al. (2023)]. In the 2D image domain with images of size $N \times N$, this could offer even greater gains, as the runtime of convolutions with NIFF is $O(N^2 \log(N))$, compared to the common convolution's $O(N^2 M^2)$, where M is the kernel size of the convolution and commonly $N > M$ is given.
Following, we outline each topic covered in this thesis in more depth and acknowledge any relevant publications and collaborations with other researchers.

1.1.1 Robust Models are Less Over-Confident

In this work, we present an extensive study demonstrating that models trained to be robust against adversarial samples are also significantly less overconfident in their incorrect predictions. This behaviour is highly desirable for networks deployed in real-world applications. Consider, for instance, a network misclassifying a malignant tumor as benign with high confidence; such an error could have life-threatening consequences. Therefore, networks should not only be resilient to adversarial attacks, *i.e.* worst-case perturbations, but also avoid excessive confidence in their false predictions, given the potentially severe impacts. We show that networks exhibiting high robustness to adversarial attacks are less overconfident and can better distinguish correct from incorrect predictions based on their confidence scores. However, we also find that commonly used confidence measures, such as the expected calibration error, fail to accurately reflect calibration for networks with low accuracy.

Contribution: This work was created in collaboration with Julia Grabinski, Paul Gavrikov, Janis Keuper, and Margret Keuper. It was published in [Grabinski et al. (2022a)] at NeurIPS 2022. Margret Keuper proposed the idea of evaluating robust versus non-robust models in terms of their confidence, following initial results provided by Julia Grabinski. Paul Gavrikov trained the non-robust models, while Julia Grabinski evaluated all models regarding their confidence and calibration. Julia Grabinski also created all figures and wrote the main body of the paper, which was subsequently polished by all the authors.

1.1.2 Aliasing and Adversarially Robust Generalization of CNNs

In this work, we reveal that robust models inherently learn to downsample more smoothly, leading to less aliasing artifacts after downsampling. We show that the amount of aliasing artifacts after downsampling is much lower for robust models compared to their non-robust counterparts. To measure this, we introduce a novel aliasing measure which quantifies how much aliasing is introduced by the downsampling operation applied in the downsampling layers of a network. Furthermore, we provide evidence of a strong correlation between catastrophic overfitting in simple FGSM AT and an increase in aliasing after downsampling. By using our aliasing measure, we are then able to quantify an optimal early stopping point for FGSM AT.

Contribution: This work was created in collaboration with Julia Grabinski, Janis Keuper, and Margret Keuper. It was published in [Grabinski et al. (2022c)] as a full paper at ECML 2022 and in [Grabinski et al. (2022d)] as an abstract at the AAAI Workshop on Adversarial Machine Learning and Beyond 2022. After thorough discussions with Margret and Janis Keuper on methods for measuring aliasing, Julia Grabinski proposed and implemented the aliasing measure. She was fully responsible for the evaluation of all models and wrote the first draft of the paper, which was subsequently polished by all the authors.

1.1.3 Aliasing-Free Downsampling in the Frequency Domain

Inspired by our prior work, we introduce a completely aliasing-free downsampling operation called FrequencyLowCut Pooling, short FLC Pooling as well as Sinc interpolation and Aliasing-free Pooling, short ASAP. FLC Pooling and ASAP leverage the Fourier domain to guarantee aliasing-free downsampling by removing the high-frequency components which could lead to aliasing. In comparison to prior work, which could only reduce aliases but not guarantee aliasing-free downsampling, both methods enhance the native robustness of CNNs. Additionally, models employing our downsampling enhance AT and effectively reduce the risk of catastrophic overfitting, leading to higher robust and clean accuracy. We can also show that models employing our downsampling techniques show favourable confidence calibrations compared to standard downsampling when combined with AT.

Contribution: This work was created in collaboration with Julia Grabinski, Steffen Jung, Janis Keuper, and Margret Keuper. FLC Pooling was published in [Grabinski et al. (2022b)] at ECCV 2022 and the extension to ASAP [Grabinski et al. (2023)] is currently under review at IJCV. Building on initial results from previous work and discussions with Margret and Janis Keuper, Julia Grabinski proposed and implemented the concept of aliasing-free downsampling in the frequency domain. She trained all models except for the large model on ImageNet-1k with AT, which was handled by Steffen Jung. The extension to aliasing- and sinc-interpolation-free pooling using a Hamming window was also proposed by Julia Grabinski. For this study, she conducted all training and evaluations, except for the AT on ImageNet-1k, which was trained by Steffen Jung. Julia Grabinski drafted the initial version of the paper, which was subsequently refined and polished by all authors.

1.1.4 Neural Implicit Frequency Filters

In this work, we leverage the Fourier domain to perform convolutions in the frequency domain, establishing a framework to explore the benefits of increased spatial context and analyse the preferred kernel sizes a network might naturally learn. A thorough study of filter sizes requires that we decouple this variable from other network aspects, such as width or the number of learnable parameters. Furthermore, the computational cost of the convolution operation must remain manageable; simply increasing the convolution kernel size is not feasible. Thus, we propose to learn the frequency representations of filter weights as neural implicit functions, such that the better scalability of the convolution in the frequency domain can be leveraged. Additionally, due to the implementation of the proposed neural implicit function, even large and expressive spatial filters can be parametrised by only a few learnable weights. To ensure efficient implementation within existing frameworks, our module is plug-and-play, allowing it to replace conventional convolutions in a CNN seamlessly. Our analysis reveals that, although the networks could learn very large convolution kernels, the learned filters are well localized and relatively small in practice when transformed from the frequency to the spatial domain. Additionally, we can reduce the amount of transformations needed between spatial and Fourier domain by combining our NIFF with our proposed downsampling from Chapter 6.

Contribution: This work was created in collaboration with Julia Grabinski, Janis Keuper, and Margret Keuper. It was published in [Grabinski et al. (2024)] in the TMLR Journal 2024 with featured certification and is presented at ICLR 2025. The need for frequency convolutions was based on our prior work introducing downsampling in the frequency domain. The idea of using a neural implicit function was proposed by Margret and Janis Keuper. The implementation and evaluation were handled by Julia Grabinski. She drafted the initial version of the paper, and Margret Keuper helped in formalizing the convolution theorem. The paper was subsequently refined and polished by all authors.

1.2 Outline

Following, we give a brief overview of the outline of this thesis. First, we describe the preliminary Chapters 2 and 3 followed by the two main parts, which include Chapters 4 to 7. Lastly, we briefly discuss Chapter 8, which concludes this thesis and discusses future directions following from this thesis.

Chapter 2, Foundations This chapter lays the groundwork for this thesis by reviewing the foundational concepts it builds upon. First, we provide an overview of convolutional neural networks (CNNs), which is the primary architecture employed in this thesis, highlighting their components, which are later adapted, and the evaluation methods used throughout the thesis. A particular emphasis is placed on adversarial attacks and training, as they serve as key tools for exposing network-specific weaknesses. Subsequently, we introduce the datasets used in the various studies presented in this thesis. Finally, we revisit digital signal processing principles utilized to evaluate and enhance modern computer vision models in this thesis.
Chapter 3, Related Work This chapter reviews related work, emphasising the various properties analysed and improved in this thesis. Starting with robustness and its diverse perspectives, followed by reliability evaluations, which are later shown to be closely connected. This chapter also highlights frequency domain implementations in image classification and their relevance to the analyses and improvements

presented in this thesis. Lastly, it explores dynamic and steerable filters, which can be implemented in the frequency domain, as demonstrated in Chapter 7.
The research presented in this thesis is organized into two parts, as outlined in Figure 1.3. The first part (Chapters 4 and 5) comprises two key studies: one investigating the reliability of robust models and the other showcasing the signal-processing advantages achievable through adversarial robustness.

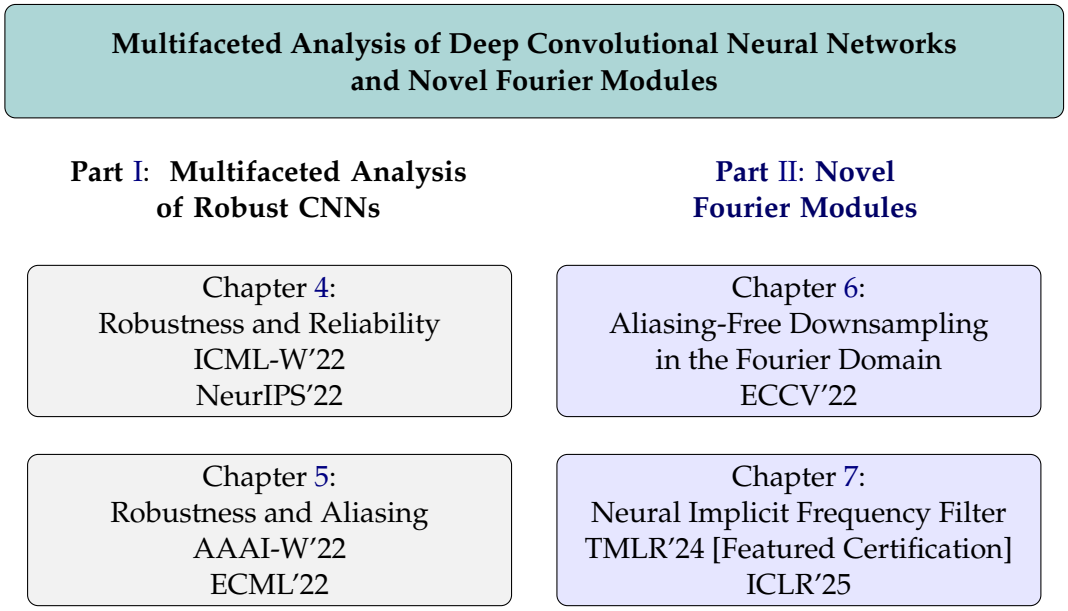


FIGURE 1.3: **Contribution Outline.**

The second part (Chapters 6 and 7) focuses on practical implementations in the Fourier domain derived from the correlations observed in the first part, resulting in enhanced robustness and efficiency.
Chapter 4, Robust Models are Less Over-Confident This chapter reveals that adversarially robust models, in addition to their robustness, possess a valuable property: They are significantly less overconfident compared to their non-robust counterparts. Furthermore, we show that fundamental building blocks play a critical role in achieving desirable confidence calibration.
Chapter 5, Aliasing and Adversarially Robust Generalization of CNNs This chapter demonstrates that robust models downsample more effectively than non-robust models, resulting in reduced aliasing after downsampling. To quantify this difference, we introduce a novel aliasing measure. Additionally, we show that catastrophic overfitting in FGSM AT aligns with an increase in aliasing, inspiring the development of a new early stopping criterion based on our metric.
Chapter 6, Aliasing-Free Downsampling in the Frequency Domain This chapter introduces aliasing-free downsampling methods inspired by the observations in Chapter 5. Our novel downsampling techniques improve native robustness and enhance AT, enabling FGSM AT to converge more effectively by reducing the risk of catastrophic overfitting.
Chapter 7, Neural Implicit Frequency Filters This chapter introduces Neural Implicit Frequency Filters (NIFFs), which facilitate fast and efficient large convolutions. Using NIFFs, we investigate optimal kernel sizes in various CNNs and find that the commonly used 3×3 kernels are insufficient. However, slightly larger kernels, such as 9×9, appear adequate for datasets like ImageNet-1k.

Chapter 8, Conclusion and Outlook This chapter concludes the thesis by summarizing the key insights and discussing the impact and limitations of our work, followed by inspiration for future research in three practical directions and a brief general overview.

Please Note: Throughout this thesis, we refer to the state-of-the-art models available at the time each individual paper was written. Similarly, the presented results reflect the status at that time. For completeness, we have, if necessary, add a paragraph discussing progress in subsequent work that demonstrates advancements in a similar direction.

1.3 Publications

The following peer-reviewed papers contribute to this thesis:

- Aliasing and Adversarial Robust Generalization of CNNs
  Julia Grabinski, Janis Keuper and Margret Keuper
  Journal Springer, Machine Learning, presented at ECML 2022
  [Grabinski et al. (2022c)]

- Aliasing Coincides with CNNs Vulnerability Towards Adversarial Attacks
  Julia Grabinski, Janis Keuper and Margret Keuper
  The AAAI-22 Workshop on Adversarial Machine Learning and Beyond 2022, (Short Paper)
  [Grabinski et al. (2022d)]

- Robust Models are Less Over-Confident
  Julia Grabinski, Paul Gavrikov, Janis Keuper and Margret Keuper
  Advances in Neural Information Processing Systems (NeurIPS) 2022
  [Grabinski et al. (2022a)]

- FrequencyLowCut Pooling–Plug & Play against Catastrophic Overfitting
  Julia Grabinski, Steffen Jung, Janis Keuper and Margret Keuper
  Proceedings of the European Conference on Computer Vision (ECCV) 2022
  [Grabinski et al. (2022b)]

- [Featured Certification] As large as it gets: Learning Infinitely Large Filters via Neural Implicit Functions in the Fourier Domain
  Julia Grabinski, Janis Keuper and Margret Keuper
  Transactions on Machine Learning Research (TMLR) 2024, presented at ICLR 2025
  [Grabinski et al. (2024)]

The following papers contribute to this thesis:

- Fix your Downsampling ASAP! Aliasing and Sinc Artifact-Free Pooling in the Fourier Domain
  Julia Grabinski, Janis Keuper and Margret Keuper
  In submission to International Journal of Computer Vision (IJCV)
  [Grabinski et al. (2023)]

Additional publications not being part of this thesis:

- On the Unreasonable Vulnerability of Transformers for Image Restoration - and an easy fix
  Shashank Agnihotri, Kanchana Vaishnavi Gandikota, **Julia Grabinski**, Paramanand Chandramouli and Margret Keuper
  Proceedings of the IEEE/CVF International Conference on Computer Vision, 4th Workshop on Adversarial Robustness In the Real World (AROW), ICCV 2023
  [Agnihotri et al. (2023)]

- Towards Class-wise Robustness Analysis
  Tejaswini Medi, **Julia Grabinski** and Margret Keuper
  [Medi et al. (2023)]

- Improving Feature Stability during Upsampling – Spectral Artifacts and the Importance of Spatial Context
  Shashank Agnihotri, **Julia Grabinski** and Margret Keuper
  Proceedings of the European Conference on Computer Vision (ECCV) 2024
  [Agnihotri et al. (2024b)]

- Beware of Aliases–Signal Preservation is Crucial for Robust Image Restoration
  Shashank Agnihotri, **Julia Grabinski**, Janis Keuper and Margret Keuper
  [Agnihotri et al. (2024a)]

1.4 Notation

The following notation is used throughout this thesis:

$\mathbb{R}$	the space of real numbers
$\mathbb{C}$	the space of complex numbers
$\mathcal{F}(.)$	Fourier transform
$\mathcal{F}^{-1}(.)$	inverse Fourier transform
j	imaginary unit
u	frequency
a	a constant
τ	a distance
T	a period
Ш_T	Dirac comb with period T
δ	Dirac delta function
H(n)	Hamming window
β	beta, Kaiser window sidelobe attenuation (relative)
$\circledast$	circular convolution
$*$	linear convolution
$\odot$	point-wise multiplication
x	input data sample (e.g., an image)
y	true label of input x
$\hat{y}$	predicted label of input x
$\hat{z}$	prediction confidence for predicted label $\hat{y}$
f_θ	neural network with parameters θ
$g(x)$	feature map within a neural network $f_\theta(x)$ w.r.t. to input x
$G(u)$	the Fourier transform of $g(x)$
F_Φ	neural implicit function parametrised by Φ
$J(\theta, x, y)$	loss function (e.g., cross-entropy)
$\mathbb{E}[X]$	expectation operator, expected value (or mean) of a random variable X
$\mathbb{P}$	probability of an event
x'	adversarially perturbed input
x'_n	adversarially perturbed input, after n iterations
α	alpha, initial perturbation budget for adversarial samples
Δ	delta, adversarial perturbation added to x
ϵ	epsilon, perturbation budget (attack strength)

Chapter 2

Foundations

Contents

In this chapter, we first introduce the network architectures utilized in this thesis, convolutional neural networks (CNNs) and their components. Subsequently, we outline the evaluation methods relevant to this work, with a particular focus on assessing robustness through adversarial attacks and explaining the fundamental principles of adversarial training. Additional details on related work in the field of adversarial attacks and training methods are provided in Section 3.1. Following, we introduce the datasets used for evaluation. The final part of this chapter addresses digital signal processing principles relevant to this work, as we aim to offer a novel perspective on the signal processing properties of modern CNNs. To this end, we first present Fourier theory, covering the Fourier Transform, the Fast Fourier Transform, as well as the Convolution and Sampling Theorem. Furthermore, we discuss various artifacts that may arise *e.g.* from improper sampling, such as aliasing and sinc-interpolation artifacts. Lastly, we explain principal component analysis (PCA), which is employed in Chapter 7 to efficiently visualize kernel weights.

2.1 Convolutional Neural Networks

CNNs have been the leading architecture for various vision tasks [LeCun et al. (1998); Krizhevsky et al. (2012); He et al. (2016a); Liu et al. (2022b)]. In this thesis, we focus on classification CNNs and thus highlight the key components in the following.

2.1.1 Components

The primary feature of CNNs, as indicated by their name, is the use of convolutions. Convolutions enable the incorporation of local neighbouring pixels, promoting locality, and are more efficient due to the shared parameters in the convolution kernel. Additionally, CNNs encompass other elements, such as downsampling, activation functions, and fully connected layers. The following sections present the key components relevant to this thesis.

Convolution. The convolution $*$ expresses how one function $f(x)$ is changed by another function $g(x)$. It is defined by the integral over the product of these two functions where one is shifted by τ:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(\tau) g(x - \tau) d\tau \tag{2.1}$$

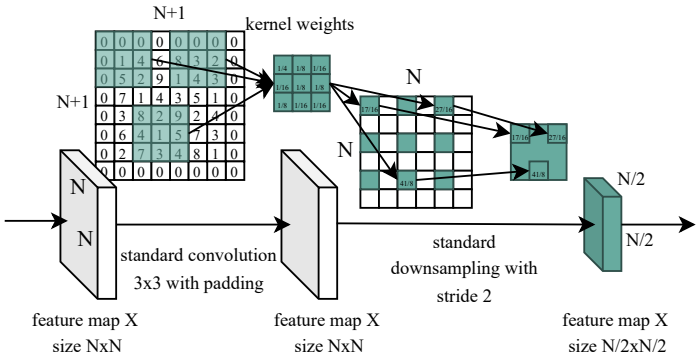
In CNNs, discrete 2D convolution are commonly applied where a filter or kernel g, which is a discrete matrix of weights of size $l \times k$, is convolved with a discrete input image or feature map f of size $n \times m$ to obtain the feature map $f' = f * g$:

$$f'(i,j) = \sum_{a=-\lfloor l/2 \rfloor}^{\lfloor l/2 \rfloor} \sum_{b=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} f(i-a, j-b) g(a,b) \tag{2.2}$$

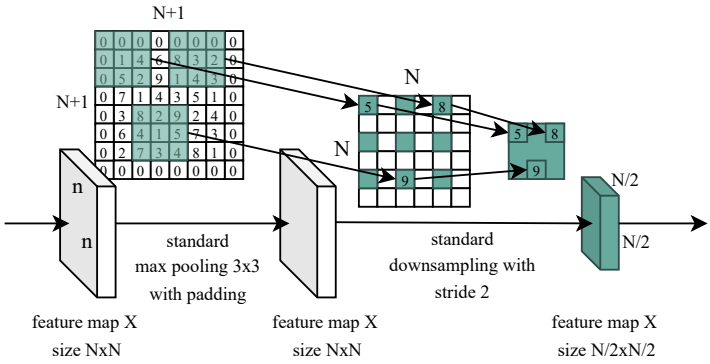
The filter g slides across the input f, performing element-wise multiplications and summing the results to produce a single value at coordinate (i,j) in the output feature map. Most CNN architectures primarily use square-shaped kernels g *i.e.* $l = k$ [He et al. (2016a); Simonyan & Zisserman (2015); Liu et al. (2022b)] or emulate them through separated convolutional filters, consisting of filter with shapes $l \times r$ followed by a filter of size $r \times k$ [Szegedy et al. (2017); Liu et al. (2023)]. However, it's not clear if this is the optimal configuration. With our method proposed in Chapter 7, we provide an analysis of kernels beyond square shapes in Section 7.3.2. Generally, convolutions allow the network to detect various features, such as edges, textures, and patterns within the input data. By leveraging spatial hierarchies and local connectivity, convolutions enable CNNs to effectively capture spatial dependencies and hierarchical features, making them highly effective for image classification and other computer vision tasks. However, common CNNs employ small kernels due to the computational costs of the convolution operation [Sandler et al. (2018)]. In contrast, we present in Chapter 7 an approach that enables large convolutions with low computational costs.

Downsampling. To abstract from the localized spatial information and learn higher-order relations of parts, objects and entire scenes, CNNs apply downsampling operations to implement a spatial pyramid representation over the network layers. Thus, the networks can learn local invariant priors and enhance efficiency due to compression.

Downsampling is typically performed via convolution with stride greater than one or by so-called pooling layers depicted in Figure 2.1. Convolutions with stride two are executed following Equation 2.2; however, the kernel g is shifted according to the stride. The most common pooling layers are AveragePooling and MaxPooling. AveragePooling computes the average in the local neighbourhood of the kernel, while MaxPooling selects the maximum value in each window. All of these standard downsampling operations are highly sensitive to small shifts or noise in the layer input as demonstrated in Chapter 5 and in prior work [Li et al. (2021); Chaman & Dokmanic (2021); Zhang (2019)].



(a) Convolution with Stride Two.



(b) MaxPooling.

FIGURE 2.1: **Standard downsampling operations used in CNNs.** (a) Downsampling via convolution with a stride of two. First, the feature map is padded, and the convolution is executed. The stride defines the step size of the kernel. Thus, for a stride of two, the kernel is moved two spatial units at a time. Effectively, this downsampling applies a standard convolution with a stride of one and discards every second point in the spatial dimension. (b) Downsampling via MaxPooling. The maximum value for each spatial window location is selected, and the striding is implemented accordingly.

Activation Function. For a network to learn more complex relations besides linear transformations, non-linearities or so-called activation functions are introduced. There is a huge variety of activation functions, however, the most commonly used one is the Rectified Linear Unit (ReLU). The ReLU function is defined as follows [Nair & Hinton (2010)]:

$$\text{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Other commonly used activation functions are the Sigmoid Linear Unit (SiLU) or hyperbolic tangent (Tanh). The SiLU function [Elfwing et al. (2018)] is formulated as:

$$\text{SiLU}(x) = x \cdot \sigma(x) \tag{2.4}$$

where $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

The Tanh function [Rumelhart et al. (1986)] is defined as:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.6}$$

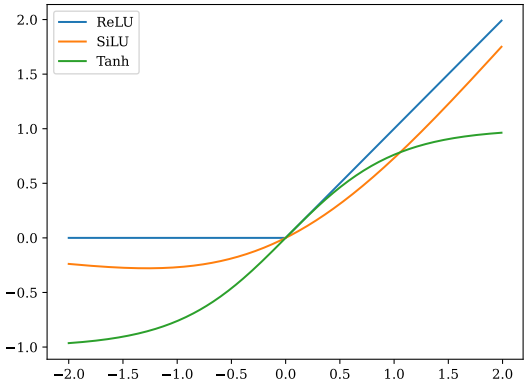


FIGURE 2.2: **Common Activation Functions.** Examples of three different activation functions used in CNNs: ReLU, SiLU and Tanh.

Fully Connected Layer. At the end of each classification CNN, a fully connected layer is placed to perform the final prediction. By connecting every neuron in one layer to every neuron in the next, fully connected layers allow the network to combine the spatially distributed features into a comprehensive understanding of the input, thereby enabling high-level reasoning and leading to accurate predictions.

2.1.2 Evaluation Methods

To evaluate classification models, several performance indicators can be used. The most common one is accuracy [Deng et al. (2009)].

Accuracy is a fundamental metric used to evaluate the performance of a classification model. It is defined as the ratio of the number of correct predictions to the total number of predictions made. Mathematically, accuracy is expressed as:

$$\text{accuracy} = \frac{\text{\# correct predictions}}{\text{\# total predictions}} \tag{2.7}$$

This metric provides a straightforward measure of how well a model's predictions align with the true labels. While accuracy is useful for assessing overall performance, it may not always provide a comprehensive estimate of the model's real-world effectiveness. Therefore, evaluating a model using additional metrics such as confidence for reliability or adversarial robustness for out-of-distribution performance is essential for the practical deployment of vision classification networks.

Confidence as an evaluation metric for classification tasks with n classes is defined in this thesis to the model's prediction value after Softmax. After the fully connected layer at the end of the network, all final predictions are fed through a Softmax layer, which is defined as follows:

$$\text{Softmax}(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.8}$$

Thus, these values are commonly interpreted as confidence for each class. The class with the highest confidence is selected as the final prediction.

Expected Calibration Error (ECE) is defined by [Naeini et al. (2015)] for a model f with $0 \leq p < \infty$ as

$$\text{ECE}_p = \mathbb{E}[|\hat{z} - \mathbb{E}[1_{\hat{y}=y}|\hat{z}]|^p]^{\frac{1}{p}} \tag{2.9}$$

where the model f predicts $\hat{y} = y$ with the confidence $\hat{z}$. This can be directly related to the overconfidence $o(f)$ and under-confidence $u(f)$ of a network as follows [Wenger et al. (2020)]:

$$|o(f)\mathbb{P}(\hat{y} \neq y) - u(f)\mathbb{P}(\hat{y} = y)| \leq \text{ECE}_p, \tag{2.10}$$

where [Mund et al. (2015)]

$$o(f) = \mathbb{E}[\hat{z}|\hat{y} \neq y] \quad u(f) = \mathbb{E}[1 - \hat{z}|\hat{y} = y], \tag{2.11}$$

I.e. the overconfidence measures the expectation of $\hat{z}$ on wrong predictions, under-confidence measures the expectation of $1 - \hat{z}$ on correct predictions and ideally both should be zero. The ECE provides an upper bound for the difference between the probability of the prediction being wrong weighted by the networks overconfidence and the probability of the prediction being correctly weighted by the networks under-confidence and converges to this value for the parameter $p \to 0$ (see Equation 2.9).

Yet, it should be noted that the ECE metric is based on the assumption that networks make correct as well as incorrect predictions. A model that always makes incorrect predictions and is less confident in its few correct decisions than it is in its many erroneous decisions can end up with a comparably low ECE. Therefore, ECE values for models with an accuracy below 50% are hard to interpret.

Robustness can be evaluated using several metrics. For this thesis, the primary metrics include robustness against common corruptions, pixel shifts, and adversarial attacks. Robustness against common corruptions can be assessed using an ensemble of sixteen different image distortions [Hendrycks & Dietterich (2019)]. Pixel shift robustness measures the model's ability to handle spatial shifts [Zhang (2019)], while adversarial robustness evaluates the model's resistance to adversarial attacks. Each of these metrics is quantified by calculating the accuracy (see Equation 2.7) when the model is exposed to corrupted or perturbed data.

2.2 Adversarial Attacks

As previously introduced, adversarial attacks are one way to evaluate a model's robustness to artificially crafted image perturbations. These perturbations are either crafted given full knowledge about the model's architecture and weights, so-called white-box attacks [Goodfellow et al. (2015); Madry et al. (2018)], or without concrete knowledge of the network's weights and limited knowledge of the model's architecture, so-called black-box attacks [Andriushchenko et al. (2020)]. Hence, in a black-box attack scenario, the attacker either needs to query the original model several times with induced perturbations to estimate the success of the attack [Andriushchenko et al. (2020)] or uses a surrogate model [Lord et al. (2022)].

For this thesis, the white-box attacks are more relevant as they can be used to examine model-specific vulnerabilities and test for different components that might not align with human vision [Madry et al. (2018)].

One of the earliest methods to attack a model in a white-box attack setting is the Fast Gradient Sign Method (FGSM) [Goodfellow et al. (2015)]. FGSM is a single-step adversarial attack which produces the perturbed image x' by adding a perturbation with strength ϵ in the direction of the gradient of the network $sign(\nabla_x \mathrm{J}(\theta, x, y))$ onto the original image x with class label y.

$$x' = x + \epsilon \cdot sign(\nabla_x \mathrm{J}(\theta, x, y)) \tag{2.12}$$

This attack operates under the L_∞ norm, meaning that the perturbation is constrained such that no individual pixel is altered by more than ϵ, ensuring that the change remains small but still effective at misleading the model.

Thereby, the image is in a single-step perturbed in the most harmful way, pushing the sample away from the true class label and possibly changing the network's prediction. This single-step attack can be enhanced to be more harmful by adding several steps to the generation of x'_n called the Basic Iterative Method (BIM) [Kurakin et al. (2017)]. The starting point of this method is the original image, thus $x'_0 = x$. For the nth step to obtain the perturbed image x'_n, similarly to Equation 2.12 the perturbation is calculated but with noise level α.

$$x'_n = x'_{n-1} + \alpha \cdot sign(\nabla_x \mathrm{J}(\theta, x'_{n-1}, y)) \tag{2.13}$$

Afterwards, x'_n is clipped to stay within the bounds of ϵ and within the maximum and minimum pixel intensities where the scalar ϵ is multiplied with the one-vector $\mathbb{1}$ to match the dimensions of x.

$$x'_n = min(255, x + \mathbb{1}\epsilon, max(0, x - \mathbb{1}\epsilon, x'_n)) \tag{2.14}$$

Following up, Madry et al. (2018) suggested starting from random noise instead of zeros for the adversarial noise, as done when x'_0 is set to the original images x. This method is called Project Gradient Descent (PGD).

Similar to FGSM, these iterative attacks operate under the L_∞ norm. However, such attacks can also be adapted to L_2 norm, resulting in more spread-out, smooth perturbations. In contrast, L_∞ perturbations introduce small, localized changes across all pixels, making them harder to detect visually. For this thesis, we mainly focus on L_∞ perturbations.

Multi-step attacks increase the effectiveness of the attack, however, they increase computational costs depending on the number of steps as later shown in Table 6.7.

In Section 3.1, we introduce additional related work, discussing various adversarial attacks and their distinct properties.

2.2.1 Adversarial Training

Adversarial Training (AT) induces adversarial samples in the network's training process. To adversarially train a network h_θ with a given input set S of image and label pairs (x, y), the following min-max optimization problem must be solved [Goodfellow et al. (2015)]:

$$\min_{\theta} \frac{1}{|S|} \sum_{(x,y)\in S} \max_{||\Delta|| \leq \epsilon} \mathrm{J}(\theta, (x+\Delta), y) \tag{2.15}$$

where Δ is the added perturbation. This AT technique assumes that the attacker has full knowledge of the model's architecture and weights. The order of the min-max optimization ensures that the training is as effective as possible, as the attack, which seeks to maximize the loss, influences the adjustment of the network's weights, θ, during the overall minimization process. This basic form of adversarial training can be modified to incorporate more complex adversaries, additional loss functions, or supplementary data. In Section 3.1, we provide a more detailed discussion of existing adversarial training methods in related work.

2.3 Datasets

The focus of this thesis lies in classification tasks, which can be evaluated on different datasets. The datasets presented are categorized into two categories: high- and low-resolution.

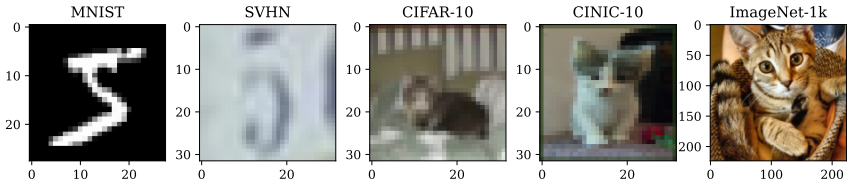


FIGURE 2.3: **Datasets used in this Thesis.** Example images from the different datasets used in this thesis, arranged from left (easiest problem to solve) to right (hardest problem to solve). MNIST images have a resolution of 28×28 pixels, while SVHN, CIFAR-10, and CINIC-10 images are 32×32 pixels. ImageNet images include the standard preprocessing size of 224×224 pixels.

2.3.1 Low-Resolution Datasets

MNIST [LeCun & Cortes (2010)] is a dataset containing handwritten digits from zero to nine, which are in grayscale and of size 28 by 28 pixels. The dataset contains 60.000 training samples and 10.000 samples.

Street View House Numbers (SVHN) [Netzer et al. (2011)] contains images from house digits from Google Street View images. Similar to MNIST, the numbers represented are from zero to nine, hence, a ten-class problem. In contrast to MNIST, the images are real-world images and in RGB colours encoded. All images are resized

to 32 by 32 pixels. The dataset contains 73.257 digits for training, 26.032 digits for testing, plus 531.131 additional samples, to use as extra training data.

CIFAR-10 [Krizhevsky (2009)] contains ten classes showing animals and means of transportation; each image is coloured and of size 32 by 32 pixels. The dataset contains 50.000 training images and 10.000 testing images, with each class being equally represented.

CIFAR-100 [Krizhevsky (2009)] shares similar characteristics with CIFAR-10, but instead of 10 classes, it contains 100. Each class has 500 training images and 100 testing images.

CINIC-10 [Darlow et al. (2018)] is similar to CIFAR-10 with the same classes. However, it has 4.5 times more images than CIFAR-10, 270.000, as it should bridge CIFAR-10 and ImageNet in the sense of available image data. Each subset, training, validation and test contains 90.000 images.

2.3.2 High-Resolution Datasets

ImageNet-1k [Deng et al. (2009)] represents 1000 object classes. The images are dominantly real-world images of varying sizes and different qualities. The standard preprocessing [Szegedy et al. (2015)] of ImageNet-1k includes resizing of 256 by 256 pixels and centered cropping to 224 by 224 pixels. The dataset contains 1.281.167 training images, 50.000 validation images and 100.000 test images.

ImageNet-100[Deng et al. (2009)] is a subset of ImageNet-1k and contains only 100 selected classes instead of 1000.

2.4 Digital Signal Processing Fundamentals

Signal processing is a field of engineering and applied mathematics that focuses on analysing, modifying, and extracting useful information from signals. A signal is any time-varying or spatially varying quantity that conveys information such as audio, images or sensor data.

In the context of machine learning, signal processing plays a crucial role in transforming raw data into meaningful features that improve model performance. Techniques like filtering via convolution, Fourier transforms and wavelet analysis help reduce noise, enhance patterns, and compress data, making it easier for machine learning algorithms to detect trends, classify information, and make predictions [Li et al. (2020); Rao et al. (2021)]. In this thesis, we want to emphasize not only the use of signal processing techniques to enhance neural network predictions but also reveal that operations within our networks violate basic signal processing fundamentals. Thus, we suggest improving at this end to make our models more robust and reliable for use in real-world applications.

In the following, we revisit some of the digital signal processing fundamentals used in this thesis to analyse and improve our current deep convolutional neural networks.

2.4.1 Fourier Transform

The Fourier transform (FT), $\mathcal{F}$, transforms a signal from the spatial or temporal domain to the frequency domain. The spatial function $f(x)$ to be transformed must follow the Dirichlet conditions. It must be absolutely integrable over the entire real line and sufficiently smooth *i.e.* shouldn't have too many oscillations or abrupt changes and a finite number of discontinuities, maxima and minima within a given interval such that the integral converges. If the spatial signal is periodic, it can be represented as a sum of sine and cosine functions of different frequencies u in the Fourier domain. Otherwise, if the function $f(x)$ is non-periodic in the spatial domain, it's FT $\mathcal{F}(f(x))$ denoted as $F(u)$ is defined as follows [Bracewell & Kahn (1966)]:

$$\mathcal{F}\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x) e^{-j2\pi ux} dx \tag{2.16}$$

The inverse FT $F(u) \xrightarrow{\mathcal{F}^{-1}} f(x)$ is given by:

$$\mathcal{F}^{-1}\{F(u)\} = f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi xu} du \tag{2.17}$$

However, in computer vision, most analogue continuous signals are discretized to store and work with them in an efficient manner. Hence, the signals used in this thesis are predominantly discrete, for which the discrete Fourier transformation (DFT) needs to be applied.

The DFT $F(k)$ with discrete frequencies k for a periodic, one dimensional signal $f(n)$ with N samples is defined as:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N}, \tag{2.18}$$

The inverse transformation is given by:

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{j2\pi kn/N}, \tag{2.19}$$

The DFT assumes periodicity, thus, it is assumed that the signal is repeated periodically to perform the DFT [Arfken (1985)].

The signals used in computer vision are mainly images, which are two-dimensional or scene volumes or videos, which can be three- or higher-dimensional. A n-dimensional discrete Fourier transform is a cascade of one-dimensional Fourier transforms [Oppenheim (1999); Bracewell & Kahn (1966)].

For the transformation of two-dimensional signals *e.g.* for images, the 2D DFT $F(k,l)$ of a input signal $f(n,m)$ of size $N \times M$ is defined as follows:

$$F(k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(n,m) e^{-j2\pi(\frac{kn}{N} + \frac{lm}{M})}, \tag{2.20}$$

The 2D inverse DFT is given by:

$$f(n,m) = \frac{1}{NM} \sum_{l=0}^{M-1} \sum_{k=0}^{N-1} F(k,l) e^{j2\pi(\frac{kn}{N} + \frac{lm}{M})}, \tag{2.21}$$

2.4.2 Fast Fourier Transform

The direct computation of the DFT requires $O(N^2)$ operations. To address this inefficiency, Cooley & Tukey (1965) introduced the Fast Fourier Transform (FFT), which leverages a divide-and-conquer strategy to reduce the computational complexity to $O(NlogN)$.

The FFT exploits the inherent symmetry that arises from the periodic nature of the transformed signal. To provide an intuitive understanding of this symmetry, consider the signal shifted by N as follows:

$$\begin{aligned} F(k+N) &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi(k+N)n/N}, \\ &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi n} e^{-j2\pi kn/N}, \\ &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N}, \end{aligned} \tag{2.22}$$

as $e^{j2\pi n} = 1$ for any integer n. Thus

$$F(k+N) = F(k) \tag{2.23}$$

and also

$$F(k+iN) = F(k) \tag{2.24}$$

hold for any integer i.

Given this symmetry, [Cooley & Tukey (1965)] partition the DFT into smaller components, enabling its computation through a divide-and-conquer approach. Following, the DFT is reorganized into two parts:

$$\begin{aligned} F(k) &= \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N} \\ &= \sum_{m=0}^{N/2-1} f(2m) e^{-j2\pi k2m/N} \\ &+ \sum_{m=0}^{N/2-1} f(2m+1) e^{-j2\pi k(2m+1)/N} \\ &= \sum_{m=0}^{N/2-1} f(2m) e^{-j2\pi km/(N/2)} \\ &+ e^{-j2\pi k/N} \sum_{m=0}^{N/2-1} f(2m+1) e^{-j2\pi km/(N/2)} \end{aligned} \tag{2.25}$$

Each part represents the even-numbered and odd-numbered values, respectively. However, the runtime is still the same as each term consists of $O(\frac{N}{2}N)$ computations, in total still $O(N^2)$.

Luckily, this division into two parts can be continued in each part again. Hence, the range of k is $0 \leq k \leq N$ while m is now in the range of $0 \leq m \leq M$ where $M = N/2$. Thus, solving the problem only takes half of the computations as before, $O(N^2)$ becomes $O(M^2)$ where M is half the size of N. As long as M is even-valued,

the problem can be divided again into smaller parts. Applying this divide-and-conquer strategy in a recursive implementation takes only $O(NlogN)$.

2.4.3 Convolution Theorem

In Chapter 7, we apply convolutions not only within the common framework of CNNs but also in the frequency domain. To establish this connection, we introduce the convolution theorem, which explains the relationship between convolution in the spatial domain and convolution in the frequency domain. The convolution theorem [Bracewell & Kahn (1966); Forsyth & Ponce (2003)] states that a convolution, denoted by $*$, between a signal $g(x)$ and filter $k(x)$ in the spatial domain can be equivalently represented by a point-wise multiplication, denoted by $\odot$, of these two signals in the frequency domain, by computing their FT, denoted by the function $\mathcal{F}(.)$:

$$\mathcal{F}(g * k) = \mathcal{F}(g) \odot \mathcal{F}(k) \tag{2.26}$$

with

$$\mathcal{F}(g)(u) = G(u) = \int_{-\infty}^{\infty} g(x) e^{-j2\pi ux} dx \tag{2.27}$$

This holds because the Fourier transformation as a system has specific properties when the signal is shifted. According to the shifting property of Fourier transform, if a function $g(x)$ is shifted by a in the spatial domain expressed by $g(x-a)$, this results in a linear phase shift in the Fourier domain:

$$\mathcal{F}(g(x-a)) = e^{-j2\pi ua} G(u) \tag{2.28}$$

By leveraging this shift property of the FT, the convolution theorem can be established. The continuous convolution is defined as follows:

$$(g * k)(y) = \int_{-\infty}^{\infty} g(x) k(y-x) dx \tag{2.29}$$

The Fourier transformation of $(g * k)(y)$ is defined by:

$$\mathcal{F}(g * k)(u) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} g(x) k(y-x) dx \right] e^{-j2\pi uy} dy \tag{2.30}$$

By reversing the order, the integration becomes:

$$\int_{-\infty}^{\infty} g(x) \left[\int_{-\infty}^{\infty} k(y-x) e^{-j2\pi uy} dy \right] dx \tag{2.31}$$

where $g(x)$ can be factored out of the inner integral. Given the shift property, the inner integral can be expressed as:

$$\int_{-\infty}^{\infty} k(y-x) e^{-j2\pi uy} dy = \mathcal{F}(k(y-x)) = e^{-j2\pi ux} K(u) \tag{2.32}$$

Substituting this result, we obtain:

$$\begin{aligned}
&\int_{-\infty}^{\infty} g(x) \left[\int_{-\infty}^{\infty} k(y-x) e^{-j2\pi uy} dy \right] dx \\
&= \int_{-\infty}^{\infty} g(x) e^{-j2\pi ux} K(u) dx \\
&= \left[\int_{-\infty}^{\infty} g(x) e^{-j2\pi ux} dx \right] K(u) \\
&= G(u)K(u) = \mathcal{F}(g)(u)\mathcal{F}(k)(u),
\end{aligned} \tag{2.33}$$

which demonstrates that, for all spatial frequencies u, the Fourier transform of the convolution is given by:

$$\mathcal{F}(g * k)(u) = \mathcal{F}(g)(u) \odot \mathcal{F}(k)(u). \tag{2.34}$$

For further details on this topic, we refer the reader to [Bracewell & Kahn (1966); Forsyth & Ponce (2003)].

Circular Convolutions vs. Linear Convolution. Applying a point-wise multiplication to a discrete signal (*i.e.* the signal is sampled) in the frequency domain translates to a circular convolution, denoted by $\circledast$, in the spatial domain due to the sampling theorem introduced in Section 2.4.4. Yet, CNNs apply linear convolution, denoted by $*$, and not circular convolutions. A circular convolution can approximate the linear convolution by zero-padding the input g with size M and the kernel k with size N to length $L \leq M + N - 1$ [Winograd (1978)]. We use this property in Chapter 7 to establish a fair comparison of standard CNN kernels and our proposed kernels.

2.4.4 Sampling Theorem

Sampling in the spatial domain is a pointwise multiplication with Dirac comb, which is a periodic function [Shannon (1949)]:

$$\text{Ш}_T(t) := \sum_{i=-\infty}^{\infty} \delta(t - iT) \tag{2.35}$$

with period T and a integer factor of i. The equivalent for this function in the Fourier domain is similarly a Dirac comb but with period $1/T$.

$$\mathcal{F}(\text{Ш}_T(u)) := \sum_{i=-\infty}^{\infty} \delta(u - i\frac{1}{T}) \tag{2.36}$$

The Dirac impulse comb behaves such that as T increases in the spatial domain, the distance between each Dirac impulse in the Dirac comb in the frequency domain decreases. Following the convolution theorem introduced in Section 2.4.3, sampling in the spatial domain corresponds to a convolution with the transformed impulse comb in the frequency domain. This leads to replica of the original signal, which overlap if T is too large. This is referred to as *aliasing*.

2.4.5 Aliasing

Aliasing is a distortion that occurs when a signal is sampled at a rate that is too low to accurately capture its frequency content *i.e.* the sampling rate f_s which is inverse proportional to the sampling period $T_s = 1/f_s$ is too low. Specifically, if the sampling rate f_s is less than twice the highest frequency component present in the signal u_{max}

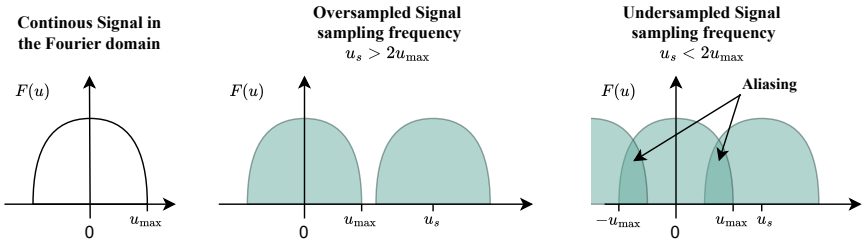


FIGURE 2.4: **Aliasing Theory** Left: The frequency spectrum of a continuos 1D signal with maximum frequency, $u_{\max}$. After sampling, replicas of the signal appear at distances u_s which is inverse proportional to the sampling rate f_s. Center: The spectrum after sampling with a sufficiently high sampling rate and sampling frequency. Right: The spectrum after under-sampling with aliases due to overlapping replicas.

known as the Nyquist rate $2u_{max}$ then higher frequencies in the original signal will appear as lower frequencies in the sampled signal [Shannon (1949)]. Thus multiple high-frequency sinusoids can produce the exact same sampled values as a lower-frequency sinusoid, making them indistinguishable after sampling. Following the sampled signal X_s is obtained by convolving the original signal $X(u)$ with $S(u) = \mathcal{F}(\text{Ш}_T)(u)$:

$$X_s(u) = X(u) * S(u) = \sum_{i=-\infty}^{\infty} X(u - if_s) \tag{2.37}$$

Demonstrating that in the spectrum of the original signal $X(u)$ is replicated at integer multiples of the sampling frequency f_s. Aliasing thus occurs when these replicated spectral copies overlap *i.e.* $f_s \leq 2u_{max}$ causing higher frequencies to be "folded back" and superimposed onto lower frequencies as demonstrated in Figure 2.4.

In the spatial domain, these aliasing artifacts become visible as grid artifacts, as shown in Figure 6.1 in Chapter 6. We relate aliasing after downsampling and robustness of CNNs in Chapter 5 and propose aliasing-free downsampling in Chapter 6.

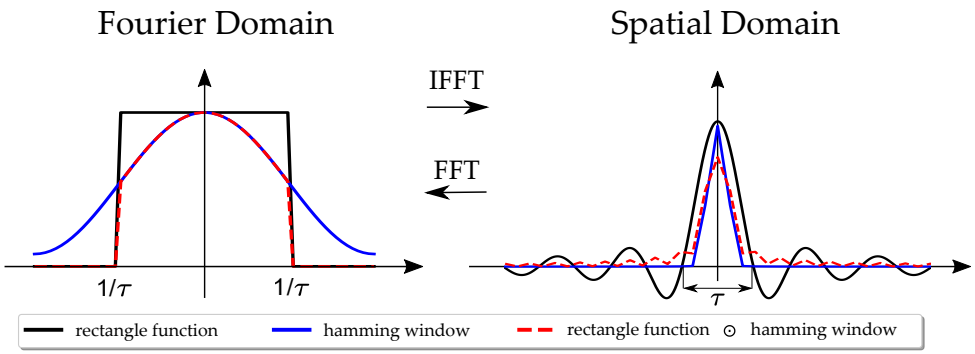


FIGURE 2.5: **Sinc interpolation Artifacts Theory.** Transformation of the rectangle function (black), the Hamming window (blue), and the point-wise multiplication of these two (red, dashed) from the Fourier domain (left) to the spatial domain (right). The rectangle function transforms into an infinite sinc function with infinite oscillations. In contrast, the side lobes of the Hamming window become nearly zero. Thus applying a Hamming window on the rectangle function in the Fourier domain results in a suppressed version of the sinc function in the spatial domain, with reduced oscillating side lobes.

2.4.6 Sinc Interpolation Artifacts

While aliasing is the most commonly referred artifact in computer vision [Vasconcelos et al. (2021); Karras et al. (2021)], other kinds of spectral leakage artifacts also exist, such as sinc interpolation artifacts. They appear as soon as a rectangle function is applied in the Fourier domain. The rectangle function $\prod$ in the Fourier domain is defined as follows:

$$rect(u,\tau) = \prod\left(\frac{u}{2/\tau}\right) = \begin{cases} 1 & \text{for } |u| \leq 1/\tau \\ 0 & \text{otherwise} \end{cases} \tag{2.38}$$

with a length of $2/\tau$ as shown in Figure 2.5 (left) within the interval $[-1/\tau, 1/\tau]$. The equivalent of this rectangle function is a sinc function in the spatial domain, as shown in Figure 2.5 (right). This rectangle function in the Fourier domain corresponds to the following sinc function $f(x,\tau)$ in the spatial domain, which depends on τ:

$$f(x,\tau) = \mathcal{F}^{-1}\left[\prod\left(\frac{u}{2/\tau}\right)\right] = \frac{2}{\tau} sinc\left(\frac{2\pi x}{\tau}\right) \tag{2.39}$$

Subsequently, the rectangle-function $\prod\left(\frac{u}{2/\tau}\right)$ can be introduced within the integration limits of $[-1/\tau, 1/\tau]$:

$$\begin{aligned} f(x,\tau) &= \mathcal{F}^{-1}\left[\prod\left(\frac{u}{2/\tau}\right)\right] \\ &= \int_{-1/\tau}^{1/\tau} 1 \cdot e^{j2\pi ux} du \\ &= \frac{1}{j2\pi x}\left[e^{j2\pi ux}\right]_{-1/\tau}^{1/\tau} \\ &= \frac{1}{j2\pi x}(e^{j2\pi\frac{x}{\tau}} - e^{-j2\pi\frac{x}{\tau}}) \\ &= \frac{1}{\pi x}\frac{e^{j2\pi\frac{x}{\tau}} - e^{-j2\pi\frac{x}{\tau}}}{2j} \\ &= \frac{1}{\pi x} sin\left(\frac{2\pi x}{\tau}\right) \\ &= \frac{2}{\tau}\frac{sin(\frac{2\pi x}{\tau})}{\frac{2\pi x}{\tau}} \\ &= \frac{2}{\tau} sinc\left(\frac{2\pi x}{\tau}\right) \end{aligned} \tag{2.40}$$

Hence, the sinc interpolation artifacts are proportional to the chosen size of τ. We revisit in Section 6.2.3 that these sinc interpolation artifacts can arise from the application of our proposed *FreqeuncyLowCut Pooling* and propose to apply a Hamming window for mitigation.

2.4.7 Principal Component Analysis

The Principal Component Analysis (PCA) [Jolliffe (1986)] is a widely used dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional representation while preserving as much variance as possible. One key aspect of the PCA is that it transforms a dataset with many possibly correlated variables into a new set of linearly uncorrelated variables, the principal components, which are orthogonal to each other. The first principal component captures the largest amount of variance in the data, thereby representing the most significant trend or underlying pattern. To accurately calculate PCA, the data x with data points i and features j is first standardized.

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{2.41}$$

This step is crucial as PCA identifies directions of maximum variance, and unstandardized features with larger scales would disproportionately influence the components, potentially confusing true underlying relationships. After standardization the covariance matrix Σ_{jk} is calculated.

$$\Sigma_{jk} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{ij} - \mu_j)(x_{ik} - \mu_k) \tag{2.42}$$

Further, the eigenvectors v and eigenvalues λ of the covariance matrix of the data are computed, identifying the directions (principal components) along which the data exhibits the greatest variance.

$$\Sigma v = \lambda v \tag{2.43}$$

The eigenvectors represent the directions in which the data varies the most, while the explained variance, given by the corresponding eigenvalues, quantifies the proportion of the total variance captured by each eigenvector. Following, the principal components are ordered by the magnitude of their eigenvalues. A scree plot can then be employed to determine the optimal number, k, of components to retain. Lastly, the standardized data is projected onto the new, lower-dimensional subspace by multiplying it with the matrix formed by the top-k selected principal components. For more details on PCA, we refer to [Dunteman (1989)]. By projecting the data onto these principal components, PCA reduces redundancy and noise, making it particularly useful for aggregation, compression and feature extraction. We utilize the compression property of PCA in Section 7.3 to visualize the structure of the learned kernel weights in an aggregated manner.

Chapter 3

Related Work

Contents

In this section, we discuss related work relevant to this thesis. First, we explore the field of robustness, emphasising its significance and methods for enhancing robustness in computer vision networks. We outline different settings with which robustness can be measured, starting with image corruptions, which might occur due to bad weather conditions or wrong camera settings, followed by downsampling attacks. A downsampling attack embeds an image within a higher-resolution image, relying on the fact that preprocessing pipelines typically rescale input images. Subsequently, we discuss adversarial attacks and adversarial training particularly relevant for our analysis presented in Chapters 4 and 5, which both reveal interesting properties of adversarially trained networks. Further, in Chapter 6, we introduce two novel downsampling approaches that enhance robustness. Next, we provide an overview of confidence calibration, as we demonstrate in Chapter 4 a close connection between robustness and calibration. Furthermore, we delve into the use of frequency representations in modern computer vision, particularly in the context of aliasing and other spectral artifacts, which are closely tied to the inherent robustness of networks as we show in Chapters 5 and 6. Additionally, we discuss the training of convolutional neural networks (CNNs) in the frequency domain, which has shown potential for developing more efficient networks. Finally, we examine dynamic and steerable filters, achievable through convolutions in the frequency domain, as presented in Chapter 7.

3.1 Robustness

Robustness involves several perspectives. Generally speaking, a network should be robust against changes in the input that would not cause a human to change their prediction [Madry et al. (2018)]. For example, small pixel changes in an input image should not affect the model's prediction [Zhang (2019)]. Similarly, a sticker covering only a small part of the input should not be able to deceive the network [Brown et al. (2018)]. However, state-of-the-art vision models remain vulnerable to small changes in the input, leading to unreliable and untrustworthy outcomes, which makes them unsuitable for real-world tasks. In the following, different approaches are described to test and enhance a network's robustness.

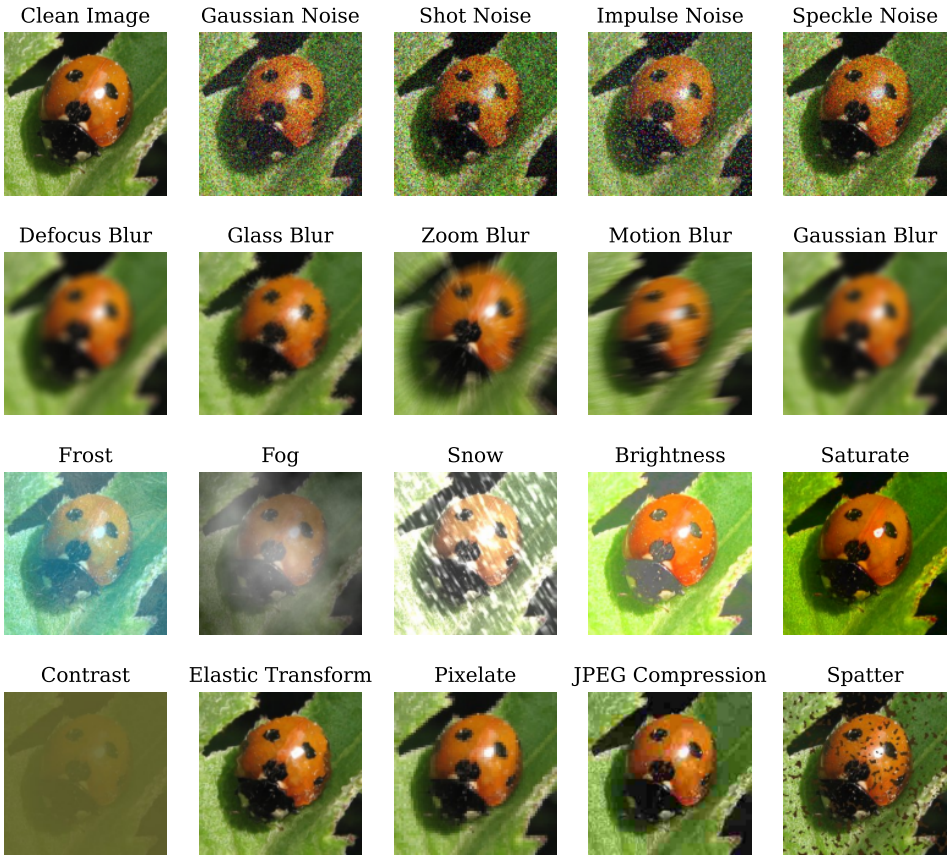


FIGURE 3.1: **Example of Common Corruption.** Example of the different Common Corruptions types [Hendrycks & Dietterich (2019)] with severity four applied on one image of a ladybug from the validation set of ImageNet-1k [Deng et al. (2009)].

3.1.1 Common Corruptions

One aspect of assessing the robustness of CNNs involves evaluating their resilience against common corruptions caused by factors such as diverse weather conditions, varying lighting conditions, or subpar camera quality. To measure this kind of robustness, the widely recognized ImageNet-C dataset is utilized [Hendrycks & Dietterich (2019)]. This dataset aims to simulate real-world scenarios through synthetic means a few examples of these corruptions are depicted in Figure 3.1

Approaches that improve upon this form of robustness often employ data augmentation techniques [Hendrycks et al. (2020); Cubuk et al. (2019)], include shape

biasing [Geirhos et al. (2019)], or combine adversarial training (AT) with augmentations [Kireev et al. (2022)]. Another approach by Vasconcelos et al. (2021) uses non trainable low-pass filters to reduce aliasing in the network.

With a similar viewpoint regarding aliasing, we demonstrate in Chapter 6 that instead of suppressing aliasing, guaranteeing aliasing-free downsampling results in higher performance and improved robustness against common corruptions compared to Vasconcelos et al. (2021).

3.1.2 Downsampling Attacks

Xiao et al. (2017) demonstrated the effectiveness of downsampling attacks, which modify images such that their original size exceeds the network's input limits, necessitating downsampling during preprocessing. This process allows an entirely different image to be embedded within the larger one, becoming apparent only after downsampling and determining the predicted class label. Subsequently, Lohn (2020) and Kim et al. (2021a) revisited this issue, proposing various defense mechanisms against such attacks, as they remain feasible due to vulnerabilities in downscaling within common Python libraries like TensorFlow and OpenCV [Lohn (2020)]. These attacks underscore the critical role of downsampling in computer vision, emphasising the need for careful handling. In Chapter 5, we demonstrate the close connection between improper downsampling and robustness. Furthermore, in Chapter 6, we propose an improved downsampling method that ensures aliasing-free processing, thereby enhancing the robustness of models.

3.1.3 Adversarial Attacks

In contrast to common corruptions and downsampling attacks, which involve fixed alterations applied to the dataset, adversarial attacks exploit network-specific vulnerabilities. These attacks are crafted to deceive a network and can operate under varying conditions. Designed "to measure progress of machine learning algorithms towards human-level abilities" [Madry et al. (2018)], adversarial attacks are often imperceptible to humans but significantly alter the model's predictions. Adversarial attacks can be categorized along two dimensions: (1) white-box vs. black-box, and (2) targeted vs. untargeted. In a white-box attack scenario [Goodfellow et al. (2015); Madry et al. (2018); Croce & Hein (2021); Kurakin et al. (2017)], the attacker has full access to the network's architecture and weights. Conversely, in a black-box attack [Andriushchenko et al. (2020)], the attacker only has access to the network's outputs, making the attack more challenging. Targeted attacks aim to manipulate the model's prediction towards a specific class label, whereas untargeted attacks aim to misclassify the input into any class other than the true class. In this thesis, we primarily focus on untargeted, white-box attacks due to their relevance in revealing network-specific vulnerabilities.

One well-known white-box attack is the Fast Gradient Sign Method [Goodfellow et al. (2015)], FGSM, mathematically described in Section 2.2, is an efficient single-step attack. Thus, FGSM is fast to compute, yet not as effective as other methods. More effective attacks use multiple optimization steps, *e.g.* as in the white-box Projected Gradient Descent (PGD) [Madry et al. (2018)], Basic Iterative Method (BIM) [Kurakin et al. (2018)], DeepFool (DF) [Moosavi-Dezfooli et al. (2016)], Carlini and Wagner (CW) [Carlini & Wagner (2017)] or Decoupling Direction and Norm (DDN) [Rony et al. (2019)].

On the other hand, black-box attacks do not have access to the model's weights and thus not to the gradients of the models. Hence, they are often developed on surrogate models [Chen et al. (2017); Ilyas et al. (2018); Tu et al. (2019)] to reduce interaction with the attacked model in order to prevent threat detection. In general, though, these attacks are less powerful due to their limited access to the target networks.

Measuring Robustness Against Adversarial Attacks. While each attack can be used to evaluate the robustness of a model by exposing it to specific attacks and measuring its accuracy on prediction tasks, several hyperparameters must be carefully selected. These include the choice of attack, the attack's strength, determined by the attack budget ϵ, the attack norm, and other settings *e.g.* the number of attack iterations for more specific attacks like PGD [Madry et al. (2018)] or BIM [Kurakin et al. (2018)]. We discuss these settings more in-depth in Section 2.2. To establish a standardized evaluation framework, Croce & Hein (2020a) introduced AutoAttack, an ensemble of multiple attacks. AutoAttack includes two versions of adaptive PGD (APGD) [Croce & Hein (2021)], one targeted and one untargeted, the fast adaptive boundary attack (FAB) [Croce & Hein (2020b)], and Squares [Andriushchenko et al. (2020)]. Due to its strong performance and comprehensive approach, AutoAttack is widely used to benchmark adversarial robustness and is featured in RobustBench [Croce et al. (2021)]. We use this benchmark in Chapters 4, 5 and 6 to ensure comparability with state-of-the-art.

Establish Robustness Against Adversarial Attacks. To ensure adversarial robustness, two main approaches can be employed. The first approach focuses on designing networks that exhibit higher native robustness through architectural modifications as proposed in Chapter 6 for downsampling, in [Dai et al. (2022)] for activation functions or in [Rodríguez-Muñoz & Torralba (2022)] for complete layers. The second approach involves adversarial training, which is discussed in the following.

3.1.4 Adversarial Training

AT [Goodfellow et al. (2015); Rony et al. (2019); Wong et al. (2020)] exposes the network to adversarial examples during training. The adversarial samples are incorporated by introducing an additional loss term during network training [Engstrom et al. (2019); Zhang et al. (2019b)]. For the mathematical formulation, we refer to Section 2.2. Further, additional training data [Rebuffi et al. (2021); Gowal et al. (2021a); Carmon et al. (2019a); Sehwag et al. (2022); Gowal et al. (2021b); Wang et al. (2020b)] can be utilized, particularly the *ddpm* dataset [Gowal et al. (2021b); Rade & Moosavi-Dezfooli (2021); Rebuffi et al. (2021)], which consists of one million extra samples for CIFAR-10 and is generated using the model proposed by Ho et al. (2020). Data augmentation has also proven to enhance adversarial robustness [Gowal et al. (2021a)], and combining it with weight averaging further improves performance [Rebuffi et al. (2021)]. Some more advanced techniques involve adding specifically generated images to the training dataset [Gowal et al. (2021b)]. **?** observed that the performance depends on the models' capacity. High-capacity models are able to fit the (adversarial) training data better, leading to increased robust accuracy. Later research investigated the influence on increased model width and depth [Gowal et al. (2021a); Xie & Yuille (2020)], and quality of convolution filters [Gavrikov & Keuper (2022)]. Consequently, the best-performing entries on RobustBench [Croce et al. (2021)] often use Wide-ResNet-70-16s or even larger architectures. Besides this

trend, concurrent works also started to additionally modify specific building blocks of CNNs [Dai et al. (2022); Rodríguez-Muñoz & Torralba (2022)]. We show in Section 6.3.3 that weaknesses in simple AT, like FGSM, can be overcome by improving the network's downsampling operation.

While AT is quite effective, it comes with a major drawback: the vast increase in computational resources required for training. Generating adversaries during training alone can increase training time by a factor of seven to fifteen [Madry et al. (2018); Wang et al. (2020b); Wu et al. (2020); Zhang et al. (2019b)] due to the additional forward and backward computations needed for each attack step. Adding additional data or generating new images [Gowal et al. (2021b)] further amplifies the computational burden. Further, the slower convergence due to the harder learning problem typically increases the training time [Kurakin et al. (2018); Madry et al. (2018); Wang et al. (2020b); Zhang et al. (2019b)].

Therefore, utilizing an efficient adversary-generating method without the need for additional data is preferable. FGSM [Goodfellow et al. (2015)] introduced formally in Section 2.2, is a single-step attack and thus is more efficient than more complex, multi-step methods like PGD [Madry et al. (2018)]. In common settings, this iterative process of PGD takes nearly nine times (Table 6.7) longer than simple FGSM training. However, FGSM training is susceptible to catastrophic overfitting [Wong et al. (2020)].

Catastrophic Overfitting in FGSM AT. AT with single-step FGSM is a simple approach to achieve basic adversarial robustness [Chen et al. (2021a); Rice et al. (2020)] (described in Section 2.2 in detail). Unfortunately, the robustness of this approach against stronger attacks like PGD is starting to drop after a certain number of training epochs. Wong et al. (2020) called this phenomenon *catastrophic overfitting*. They concluded that single-step adversarial attacks tend to overfit to the chosen adversarial perturbation magnitude (given by ϵ) but fail to be robust against multi-step attacks like PGD. Thus, Rice et al. (2020) introduced early stopping as a countermeasure. After each training epoch, the model is evaluated on a small portion of the dataset with a multi-step attack, which again increases the computation time. As soon as the accuracy drops compared with a hand-selected threshold, the model training is stopped. Further, Chen et al. (2021a) suggested preventing overfitting by forcing the network to learn smoothly during AT. Therefore, they perform stochastic weight averaging as well as smoothing of the logits. Kim et al. (2021b) and Stutz et al. (2021) showed that the observed overfitting is related to the flatness of the loss landscape. They introduced a method to compute the *optimal* perturbation length ϵ' for each image and do single-step FGSM training with this optimal perturbation length to prevent catastrophic overfitting. Andriushchenko & Flammarion (2020) showed that catastrophic overfitting not only occurs in deep neural networks but can also be present in single-layer convolutional neural networks. They propose a new kind of regularization, called GradAlign, to improve FGSM perturbations and flatten the loss landscape to prevent catastrophic overfitting. We demonstrate in Section 5.3.4 that catastrophic overfitting can be efficiently mitigated by introducing a novel early stopping criteria based on our proposed aliasing measure. Further, we replace standard downsampling with our aliasing-free downsampling to inherently boost the network's robustness and reduce the risk of overfitting in Chapter 6.

Adversarial Attack Detection. A practical defense, besides AT, can also be established by the detection and rejection of malicious input. Most detection methods are

based on input sample statistics [Hendrycks & Gimpel (2017); Li & Li (2017); Harder et al. (2021); Feinman et al. (2017); Grosse et al. (2017)], while others attempt to detect adversarial samples via inference on surrogate models, yet these models themselves might be vulnerable to attacks [Cohen et al. (2020); Metzen et al. (2017)]. While all of these approaches perform additional operations on top of the model's prediction, we show that simply taking the model's output after Softmax as prediction confidence can be used as a heuristic to reject erroneous samples in Chapter 4.

Adversarial Training and Calibration. Only a few but notable prior works, such as [Lakshminarayanan et al. (2017); Qin et al. (2021)] have investigated adversarial training with respect to model calibration. Without providing a systematic overview, Lakshminarayanan et al. (2017) showed that AT can help to smoothen the prediction distributions of CNN models. Qin et al. (2021) investigated adversarial data points generated using [Carlini & Wagner (2017)] with respect to non-robust models and find that easily attackable data points are badly calibrated while adversarial models have better calibration properties. In contrast, we analyse the robustness and calibration of pairs of robust and non-robust versions of the same models in Section 4.2 rather than investigating individual data points. Tomani & Buettner (2021) introduced an adversarial calibration loss to reduce the calibration error. Further, Stutz et al. (2020) proposed confidence-calibrated adversarial training to force adversarial samples to show uniform confidence, while clean samples should be one-hot encoded. Complementary to RobustBench [Croce et al. (2021)], Chapter 4 provides an analysis of the predictive confidences of adversarially trained, robust models. Further, we released the conventionally trained counterparts of the models from RobustBench [Croce et al. (2021)] in `https://github.com/GeJulia/robustness_confidences_evaluation` to facilitate future research on the analysis of the impact of training schemes versus architectural choices. Importantly, this proposed large-scale study allows a differentiated view of the relationship between robustness and model calibration, as discussed in Section 4.2. In particular, the adversarially trained models are not always better calibrated than vanilla models, particularly on clean data, while they are consistently less overconfident.

3.2 Confidence Calibration

For many models that perform well w.r.t standard benchmarks, it has been argued that the model accuracy may be an insufficient metric [Amodei et al. (2016); DeGroot & Fienberg (1983); Corbière et al. (2019); Varshney & Alemzadeh (2017)], in particular when real-world applications with potentially open-world scenarios are considered. In these settings, reliability must be established, which can be quantified by the prediction confidence [Ovadia et al. (2019)]. Ideally, a reliable model would provide high-confidence predictions on correct classifications and low-confidence predictions on false ones [Corbière et al. (2019); Nguyen et al. (2015)]. However, most networks are not able to instantly provide a sufficient calibration. Most common CNNs are overconfident [Lakshminarayanan et al. (2017); Guo et al. (2017); Nguyen et al. (2015)]. Moreover, the most dominantly used activation in modern CNNs [He et al. (2016a); Szegedy et al. (2015); Simonyan & Zisserman (2015); Huang et al. (2017a)] remains the ReLU function, while it has been pointed out by Hein et al. (2019) that the ReLU operation causes a general increase in the models' prediction confidences, regardless of the prediction validity. This is also the case for the vast

majority of the adversarially trained models considered in Section 4.2.2, except for the model by Dai et al. (2022) to which particular attention is devoted.

Other methods to improve calibration are based on additional loss functions [Lakshminarayanan et al. (2017); Gurau et al. (2018); Moon et al. (2020); Li & Hoiem (2020); Hein et al. (2019)], on adaptions of the training input by label smoothing [Szegedy et al. (2016); Reed et al. (2014); Müller et al. (2019); Qin et al. (2021)] or on data augmentation [Zhang et al. (2018); DeVries & Taylor (2017); Lakshminarayanan et al. (2017); Thulasidasan et al. (2019)]. Further, [Ovadia et al. (2019)] presents a benchmark on classification models regarding model accuracy and confidence under dataset shift.

3.3 Frequency Domain for Image Classification

The frequency domain has been utilized for various purposes in modern image classification [Yin et al. (2019); Saikia et al. (2021); Pratt et al. (2017)]. It provides deeper insights into the data, enabling the detection of noise applied to the data, such as that introduced by adversarial attacks [Harder et al. (2021); Hossain et al. (2023); Bernhard et al. (2021); Wang et al. (2020a)]. This makes it an effective tool for identifying attacks and even enhancing robustness, as presented in Chapter 6. Additionally, the convolution theorem, facilitated in Chapter 7, allows for more efficient and faster convolutions in the frequency domain, which have been employed to reduce inference time and memory consumption [Ayat et al. (2019); Guan et al. (2019); Mathieu et al. (2013); Pratt et al. (2017); Vasilache et al. (2014); Wang et al. (2016)]. In the following, we discuss different ways the frequency domain has been leveraged in related work and shortly outline how we leveraged the frequency domain in this thesis.

3.3.1 Frequency Analysis for Robustness and Attack Detection

Yin et al. (2019) showed that conventional CNNs are sensitive to changes in the high-frequency components of an image, like Gaussian noise, while most CNNs are robust against changes in the low-frequency components, *i.e.* in the coarse structures. In contrast, when models are trained using additional data augmentation techniques, they are less sensitive to high-frequency changes but sacrifice their robustness in the low-frequency domain [Yin et al. (2019)]. Recently, Hossain et al. (2023) analysed the frequency spectrum of adversarial examples and observed that adversarial perturbations are not exclusively affecting high-frequency components, which was assumed before [Wang et al. (2020a)]. Hossain et al. (2023) observed that the perturbations spectrum highly depends on the dataset used, *i.e.* CIFAR-10 or ImageNet. Also Bernhard et al. (2021) highlighted the fact that adversarial attacks do not exclusively hurt the high-frequency components by incorporating a frequency constraint, which needs to be adapted to the frequency features of the data. Harder et al. (2021) used the spectrum of the adversarial examples to detect them, and Lorenz et al. (2021) trained a classifier to detect adversarial examples and defend CNNs. These works indicate that there is a severe domain shift between the frequency distribution of genuine images and adversarial attacks.

In Chapter 5, we analyse robust models and their downsampling operation in the Fourier domain and reveal that robust models suffer much less from aliasing artifacts than non-robust models.

3.3.2 Aliasing in CNNs

The issue of aliasing effects in CNN-based neural networks has been extensively explored in the literature from various perspectives. A brief mathematical overview is given in Section 2.4.5. Azulay & Weiss (2019) discussed why CNNs struggle to learn invariance to small image transformations (such as shifts) from training data and argued that aliasing during downsampling is responsible for this behaviour. Since then, anti-aliasing filters have become increasingly important in the Deep Learning community. Zhang (2019) proposed to apply predefined fixed blurring kernels before downsampling to reduce aliasing. Building on this work, shift invariance is further improved in [Zou et al. (2023)] by utilizing learned blurring filters instead of predefined kernels. In [Li et al. (2021)], the pooling operations leverage the low-frequency components of wavelets to mitigate aliasing and enhance robustness against common image corruptions. Depth-adaptive blurring filters before pooling are proposed in [Hossain et al. (2023)], along with an anti-aliasing activation function. This activation function is inspired by C-ReLU [Shang et al. (2016)] but uses a smooth roll-off phase instead of a sharp cutoff at the threshold t. The importance of anti-aliasing is also recognized in the field of image generation. The use of blurring filters to eliminate aliases during image generation in generative adversarial networks (GANs) is suggested in [Karras et al. (2021)], while [Durall et al. (2020)] and [Jung & Keuper (2021)] incorporate additional loss terms in the frequency domain to address aliasing. As mentioned previously, we empirically demonstrate that adversarially robust models exhibit lower levels of aliasing in their downsampling layers in Chapter 5. Motivated by these findings, we propose two aliasing-free downsampling methods in the Fourier domain for increased native robustness and to prevent catastrophic overfitting in Chapter 6. In contrast to previous work, which focused on fixing shift-invariance and model robustness by incorporating anti-aliasing techniques, we are the first to focus on the analysis of aliasing and obtain a distinct aliasing measurement in Chapter 5 which we reformulate in Chapter 6 to guarantee aliasing-free downsampling.

3.3.3 Spectral Leakage Artifacts in CNNs

In contrast to the well-known issue of aliasing, spectral leakage artifacts have received less attention in the context of CNNs. A common example is the induction of sinc interpolation artifacts, mathematically described in Section 2.4.6, which often arise when applying finite windows to periodic signals in the frequency domain. These artifacts manifest as ringing effects in the spatial domain, as described in [Gonzales & Wintz (1987)], and are associated with the Gibbs phenomenon [Hamming & Stearns (1979)].

To mitigate these spectral leakage artifacts, various window functions can be employed, as discussed in [Prabhu (2014)]. In digital image processing, the use of 2D window functions is prevalent, particularly in biomedical image processing, where these functions play a crucial role in spectral analysis [Jähne (2005); Semmlow & Griffel (2021)].

More recently, spectral leakage artifacts in CNNs have been studied in [Tomen & van Gemert (2021)], showing that small spatial kernels can contribute to such artifacts. As a solution, they propose learning larger spatial kernels while applying a Hamming window to the convolution weights.

In Section 6.2.3, we show that spectral leakage artifacts also play a crucial role during downsampling in the frequency domain. By reducing sinc-interpolation artifacts, we can enhance the network's robustness against adversarial attacks and common corruptions.

3.3.4 Training CNNs in the Frequency Domain

In the following, we discuss two directions which leverage the frequency domain for CNN training and inference. First, we explore its role in enhancing robustness and generalization. Second, we examine resource-efficient training and inference techniques that utilize the convolution theorem.

Related work has shown that a model's robustness can be enhanced by training separate low- and high-frequency experts [Saikia et al. (2021)]. They suggest training two different models: one that relies solely on low-frequency components for prediction and another that uses only high-frequency components. The joint prediction from both models achieves much higher robustness than standard models. This approach bypasses the time and computational resources required for AT, though it still necessitates training both experts separately. In the field of domain adaptation and generalization, Yang & Soatto (2020) demonstrated that the Fourier phase and amplitude of an image can be adapted for data augmentation, leading to a model with improved generalization in the context of domain adaptation. Similarly, Xu et al. (2021) used the Fourier phase and shuffled the amplitude of images to train for better domain generalization. In Section 6.3 we can show that downsampling in the frequency domain can enhance generalized robustness against a variety of challenges, like adversarial attacks, robustness against common corruptions and pixel shifts.

Another line of work concentrates on resource-efficient and fast convolutions by leveraging the convolution theorem, which is described in detail in Section 2.4.3. Most works implementing convolutions in the frequency domain focus on the efficiency of the time and memory consumption of this operation [Ayat et al. (2019); Guan et al. (2019); Mathieu et al. (2013); Pratt et al. (2017); Vasilache et al. (2014); Wang et al. (2016)] since the equivalent of convolution in the spatial domain is a point-wise multiplication in the frequency domain. However, most of these approaches still learn the convolution filters in the conventional setting in the spatial domain and transform feature maps and kernels into the frequency domain to make use of the faster point-wise multiplication at inference time [Ayat et al. (2019); Mathieu et al. (2013); Wang et al. (2016)]. This is mostly due to the fact that one would practically need to learn filters as large as feature maps when directly learning the frequency representation. Those few but notable works that propose to learn convolutional filters in CNNs purely in the frequency domain, until now, could only achieve desirable evaluation performances on MNIST and reach low accuracies on more complex datasets like CIFAR or ImageNet [Pan et al. (2022); Pratt et al. (2017); Watanabe & Wolf (2021)] if they could afford to evaluate on such benchmarks at all. In contrast, Rao et al. (2021) demonstrated that Vision Transformer can greatly benefit from convolutions in the frequency domain, increasing performance but also the number of learnable parameters. Table 3.1 gives a brief overview of some of these approaches, executing the convolution or even more components in the frequency domain.

Further approaches apply model compression by zeroing out small coefficients of the feature map in the frequency domain [Guan et al. (2019)], enhancing robustness by frequency regularization [Lukasik et al. (2023)] or enriching conventional

convolutions with the frequency perspective to get more global information [Chi et al. (2020)].

In contrast to all these approaches, Chapter 7 introduces neural implicit frequency filters (NIFFs), which efficiently learn and execute convolutions in the frequency domain. While NIFF offers great potential for efficient large convolutions, we focus on the analysis of which filter size CNNs practically make use of if they have the opportunity to learn infinitely large kernels in this thesis. Our NIFFs are the first approach to fully learn the convolutions of a CNN in the frequency domain while maintaining state-of-the-art performance on common image classification benchmarks.

TABLE 3.1: **Vision Models operating in the Fourier Domain.** Overview of different approaches operating in the frequency domain compared to our NIFF introduced in Chapter 7. The approaches in the first two rows [Mathieu et al. (2013); Wang et al. (2016)] focus on the time improvement of the point-wise multiplication compared to standard convolution. The third and fourth row present approaches that operate (almost) fully in the frequency domain [Pratt et al. (2017); Pan et al. (2022)]. Tomen & van Gemert (2021) apply a global filter layer via point-wise multiplication in the frequency domain at the stem of a transformer model. In comparison, our NIFF, a plug-and-play operation, can replace any kind of convolution operation to learn and infer fully in the frequency domain. Note that these methods have been proposed and optimized for different purposes, and the respective numbers on CIFAR-10 reported are not comparable.

Method	Architecture	Operations in the Fourier Domain	reported CIFAR-10 Acc@1
Fast FFT 2013	not reported	Executing of the Convolution	no acc. reported
CS-unit 2016	not reported	Convolution + Downsampling	no acc. reported
FCNN 2017	FCNN	Full Network	$\sim$ 23%
CEMNet 2022	CEMNet	Full Network except MaxPooling	59.33%-78.37%
GFN 2021	Swin-Transformer	First Patchifying operation	98.60%
NIFF (ours)	any CNN	Convolution	90.63%-94.03%

3.4 Dynamic and Steerable Filters

Our NIFFs proposed in Chapter 7 implemented inherently so-called dynamic filters, as they can learn filters of varying size and enable an effective analysis of the individual filter size needed for the task and network at hand. Dynamic filtering and steerable filters [Freeman et al. (1991)] allow for adaptive filtering based on the content or orientation of the input data by *e.g.* deformable convolutions [Dai et al. (2017)]. Unlike traditional convolutions, where the kernel is fixed and applied uniformly across the input, deformable convolutions allow the filter to adapt its shape and size based on the input data. This is done via a learnable offset that controls the sampling locations of the convolutional operation within the input feature map. Further, Pintea et al. (2021) proposed to use flexible kernel sizes at each stage by learning the σ values for a combination of Gaussian derivative filters, thus inherently controlling the size of the filter. Sosnovik et al. (2020) combined the concepts of steerable filters and group equivariance to achieve scale-equivariant representations, allowing it to effectively handle objects at different scales without the need for explicit scale normalization techniques. Similarly, Worrall & Welling (2019) aimed to

achieve equivariance over scale. They integrate scale-space operators, such as Gaussian blurring and Laplacian of Gaussian operators, into the network layer to be more robust to variations in object size or resolution.

3.4.1 Large Kernel Sizes

While the first CNNs used relatively large kernels w.r.t the image recognition tasks *e.g*. 5×5 for MNIST in [LeCun et al. (1998)] or 11×11 for ImageNet [Krizhevsky et al. (2012)]. This trend shifted with the introduction of VGG [Simonyan & Zisserman (2015)], which uses only small 3×3 kernels, making computations more efficient. Recently, the development of Vision Transformers (ViTs) has shifted the focus back to utilizing a larger spatial context for vision tasks. ViTs have led to significant improvements in image classification [Dosovitskiy et al. (2021); Vaswani et al. (2017)] and related tasks. These models rely on larger image patches and self-attention mechanisms, enabling the encoding of the entire spatial context. Subsequently, Liu et al. (2021) demonstrated that incorporating convolutions with receptive fields of 7×7 to 12×12 can enhance network performance, while follow-up works further increased the window sizes [Dong et al. (2022)]. The general observation remains that image-processing models benefit from encoding a larger spatial context, particularly in deeper network layers. In this regard, Liu et al. (2022b) proposed a CNN architecture employing 7×7 depth-wise convolutions, outperforming transformer models and demonstrating that convolutional approaches remain highly relevant for image recognition tasks. In the context of CNNs, Guo et al. (2022) and Peng et al. (2017) further increased the receptive fields of the convolution and Ding et al. (2022) and Liu et al. (2023) achieved improved results with 31×31 and even 61×61 sized convolution filters, respectively. To facilitate scaling, Ding et al. (2022) used depth-wise convolutions instead of full convolutions and thereby increased the model size by only 10-16%. Liu et al. (2023) decomposed the 61×61 convolution into two parallel and rectangular convolutions to reduce the computation load and parameters. Further, we showed in [Agnihotri et al. (2024b)] that increased spatial context can improve the upsampling in decoder networks.

In sequence modelling, so-called *long convolutions* enable models to encode larger parts of a sequence at a low cost. Several works [Poli et al. (2023); Gu et al. (2022)] make use of the convolution theorem to model global large 1D convolutions. Rao et al. (2021) proposed Global Filter Networks based on Swin-Transformers. They encode intermediate layers as convolutions in the frequency domain followed by a feed-forward neural network and thus facilitate efficient convolutions with infinite extent. They achieve very good results on ImageNet-1k. However, they face the computational overhead of learning the filter weights directly, resulting in an increase in the number of parameters. In Chapter 7, we compute convolutions in the frequency domain to benefit from the better scalability of the convolution operation. Yet, our NIFF module uses neural implicit functions to preserve the original number of model parameters. *E.g*. the model proposed by Rao et al. (2021) is over six times larger than ours.

3.4.2 Neural Implicit Representations

As previously outlined, large kernels, which offer a broader context and thus greater flexibility, suffer from low computational efficiency. Neural implicit representation can be used to enhance the learning of large kernels.

Generally speaking, neural implicit representations map a point, encoded by coordinates, to a continuous output domain using a Multi-Layer Perceptron (MLP). This approach represents an object as a function rather than as fixed discrete values, which is more common in traditional methods (*e.g.*, discrete grids of pixels for images, discrete samples of amplitudes for audio signals, and voxels, meshes, or point clouds for 3D shapes) [Chibane et al. (2020); Sitzmann et al. (2020)]. Continuous representations through neural implicit methods offer a fixed number of parameters and are independent of spatial or frequency resolutions. Moreover, neural implicit representation can also be used to learn to generate high-resolution data [Chen et al. (2021b)], or to learn 3D representations from 2D data [Sitzmann et al. (2019)].

HyperNetworks. One specific form of neural implicit representations are HyperNetworks [Ha et al. (2017)], which are essentially neural networks designed for specific subtasks within a larger neural network. Ha et al. (2017) utilized a predefined SIREN [Sitzmann et al. (2020)] architecture, consisting of stacked 1x1 convolutions and periodic activations, to predict spatial kernel weights with a predefined size. While Ma et al. (2022) and Ma et al. (2023) introduced hyperconvolutions for biomedical image segmentation. Similarly, Romero et al. (2022b) and Romero et al. (2022a) introduced continuous kernels by learning spatial kernel weights that match the size of the feature maps, using a HyperNetwork composed of stacked 1x1 convolutions. These kernels are then masked and cropped to obtain the actual filter size. All of their operations are applied in the spatial domain. To enhance the learning of appropriate filter structures, they replace the periodic activation function of the SIREN with Multiplicative Anisotropic Gabor Networks. Additionally, Romero et al. (2022a) learned a Gaussian filter on top to determine the spatial kernel size. In contrast, our NIFFs, introduced in Chapter 7, directly learn the Fourier representation of convolution filters, *i.e.* a representation in a basis of sine and cosine waves, using stacked 1×1 convolutions with conventional activations. Thus, we can efficiently execute the convolution in the Fourier domain with the objective of investigating the effective spatial extent of the learned convolution filters

Part I

Multifaceted Analysis of Robust CNNs

Chapter 4

Robust Models are Less Over-Confident

Contents

In this chapter, we analyse the correlation between adversarial robustness of models and their inherent confidence calibration. Despite the success of CNNs in many academic benchmarks for computer vision tasks [He et al. (2016a); Liu et al. (2022b)], their application in the real-world is still facing fundamental challenges [Hendrycks & Dietterich (2019); Guo et al. (2017); Goodfellow et al. (2015)]. One of these open problems is the inherent lack of robustness, unveiled by the striking effectiveness of adversarial attacks [Goodfellow et al. (2015); Madry et al. (2018)]. Current attack methods are able to manipulate the network's prediction by adding specific but small amounts of noise to the input. In turn, adversarial training (AT) aims to achieve robustness against such attacks and, ideally a better model generalization ability by including adversarial samples in the training set [Goodfellow et al. (2015); Rony et al. (2019); Wong et al. (2020)]. Following, we empirically analyse a variety of adversarially trained models, following denoted as robust models, that achieve high robust accuracies when facing state-of-the-art attacks and show that AT has an interesting side-effect: it leads to models that are significantly less overconfident with their decisions compared to models trained without AT, in the following denoted as non-robust models. Further, our analysis of robust models reveals that not only AT but also the model's building blocks (like activation [Dai et al. (2022)] functions and pooling, explained in detail in Chapter 6) have a strong influence on the models' prediction confidences.
The trained models and our evaluation can be found at `https://github.com/GeJulia/robustness_confidences_evaluation`.

This chapter is based on Grabinski et al. (2022a) . Julia Grabinski, as the first author, conducted all experiments besides training the non-robust models. These

models were trained and made available on GitHub by Paul Gavrikov. Julia Grabinski was the main writer.

4.1 Introduction

CNNs have been shown to successfully solve problems across various tasks and domains. However, distribution shifts in the input data can have a severe impact on the prediction performance. In real-world applications, these shifts may be caused by a multitude of reasons, including corruption due to weather conditions, camera settings, noise, and maliciously crafted perturbations to the input data intended to fool the network (adversarial attacks). In recent years, a vast line of research (*e.g.* [Hendrycks & Dietterich (2019); Goodfellow et al. (2015); Madry et al. (2018)]) has been devoted to solving robustness issues, highlighting a multitude of causes for the limited generalization ability of networks and potential solutions to facilitate the training of better models.

A second, yet equally important issue that hampers the deployment of deep learning-based models in practical applications is the lack of calibration concerning prediction confidences. In fact, most models are overly confident in their predictions, even if they are wrong [Lakshminarayanan et al. (2017); Guo et al. (2017); Nguyen et al. (2015)]. Specifically, most conventionally trained models are unaware of their own lack of expertise, *i.e.* they are trained to make confident predictions in any scenario, even if the test data is sampled from a previously unseen domain. Adversarial examples leverage this weakness [Goodfellow et al. (2015)], as they are fooling the network with high-confident wrong predictions [Lee et al. (2018)]. In turn, AT has shown to improve the prediction accuracy under adversarial attacks [Goodfellow et al. (2015); Zhang et al. (2019b); Rony et al. (2019); Engstrom et al. (2019)]. However, few works have been investigating the links between calibration and robustness [Lakshminarayanan et al. (2017); Qin et al. (2021)], thus, we provide a systematic synopsis of adversarial robustness and prediction confidence in the following.

In this chapter, we provide an extensive empirical analysis of diverse adversarially robust models with regard to their prediction confidences. Therefore, we evaluate more than 70 adversarially robust models and their conventionally trained counterparts, which show low robustness when exposed to adversarial examples. By measuring their output distributions on benign and adversarial examples for correct and erroneous predictions, we show that adversarially trained models have benefits beyond adversarial robustness; they are less overconfident.

To cope with the lack of calibration in conventionally trained models, Corbière et al. (2019) proposed to rather use the true class probability than the standard confidence obtained after the Softmax layer, such as to circumvent the overlapping confidence values for wrong and correct predictions. However, we observe that these overlaps are an indicator of insufficiently calibrated models and can be mitigated by the improvement of CNNs building blocks, namely downsampling and activation functions, that have been proposed in the context of adversarial robustness [Grabinski et al. (2022b); Dai et al. (2022)]. We evaluate our proposed downsampling methods from Chapter 6 in more depth in Section 6.3.4.

In the following, we focus on analysing the relationship between robust models and model confidence. Our experiments for 71 robust and non-robust model pairs on the datasets CIFAR-10 [Krizhevsky (2009)], CIFAR-100 and ImageNet-1k [Deng

et al. (2009)] confirm that non-robust models are overconfident with their false predictions. This highlights the challenges for usage in real-world applications. In contrast, we show that robust models are generally less confident in their predictions, and, especially CNNs, which include improved building blocks (downsampling and activation) turn out to be better calibrated, manifesting low confidence in wrong predictions and high confidence in their correct predictions. Further, we demonstrate that the prediction confidence of robust models can be used as an indicator for erroneous decisions. However, despite their robustness, we find that adversarially trained networks still overfit to training adversaries, exhibiting similar performance to non-robust models against unseen attacks. Our contributions can be summarized as follows:

- We provide an extensive analysis of the prediction confidence of 71 adversarially trained models (**robust models**), and their conventionally trained counterparts (**non-robust models**). We observe that most non-robust models are exceedingly overconfident, while robust models exhibit less confidence and are better calibrated for slight domain shifts.
- We observe that specific layers that are considered to improve model robustness also impact the models' confidences. In detail, improved downsampling layers and activation functions can lead to an even better calibration of the learned model.
- We investigate the detection of erroneous decisions by using the prediction confidence. We observe that robust models are able to detect wrong predictions based on their confidence. However, when faced with unseen adversaries or corruptions, they exhibit in some cases similar performance as non-robust models.

Our analysis provides a first synopsis of adversarial robustness and model calibration and aims to foster research that addresses both challenges jointly rather than considering them as two separate research fields. To further promote this research, we released our modelzoo at `https://github.com/GeJulia/robustness_confidences_evaluation`.

4.2 Experiments

In the following, we first describe our experimental setting in which we then conduct an extensive analysis on the two CIFAR datasets, CIFAR-10 and CIFAR-100, with respect to robust and non-robust model[1] confidence on clean and perturbed samples as well as their ECE. Further, we observe by computing the ROC curves of these models that robust models are best suited to distinguish between correct and incorrect predictions based on their confidence. In addition, we point out that the improvement of pooling operations or activation functions within the network can enhance the models' calibration further. Finally, we also investigate ImageNet-1k as a high-resolution dataset and observe that the model with the highest capacity and AT can achieve the best performance results and calibration.

[1]The classification into robust and non-robust models is based on the models' robustness against adversarial attacks. We consider a model to be robust when it achieves considerably high accuracy under AutoAttack [Croce & Hein (2020a)].

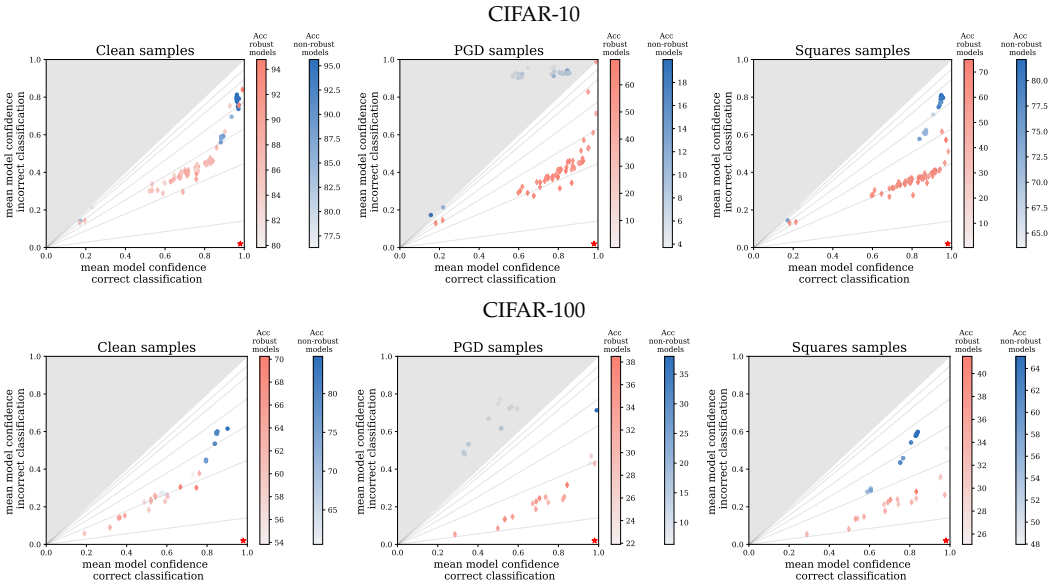


FIGURE 4.1: **Confidence Overview for Robust- and Non-Robust Models.** Mean model confidences for their correct (x-axis) and incorrect (y-axis) predictions over the full CIFAR-10 dataset (top) and CIFAR-100 dataset (bottom), for clean (left) and perturbed data with the attacks PGD (middle) and Squares (right). Each point represents a model. Circular points (light blue colourmap) represent non-robust models, while diamond-shaped points (light red colourmap) represent robust models. The colour of each point indicates the model's accuracy, with darker colours signifying higher accuracy (better performance) on the given data samples (clean or perturbed). The star in the bottom right corner indicates the optimal model calibration and the grey area marks the region where the confidence distribution of the network is worse than random, *i.e.*, where the model is more confident in incorrect predictions than incorrect ones.

4.2.1 Experimental Setup

We have collected 71 checkpoints of robust models [Rebuffi et al. (2021); Huang et al. (2021); Zhang et al. (2020, 2019a); Hendrycks et al. (2019); Zhang et al. (2021); Chen & Lee (2021); Andriushchenko & Flammarion (2020); Cui et al. (2021); Rice et al. (2020); Dai et al. (2022); Gowal et al. (2021a); Sitawarin et al. (2021); Chen et al. (2022); Zhang et al. (2019b); Wu et al. (2020); Wong et al. (2020); Huang et al. (2020); Carmon et al. (2019b); Pang et al. (2020); Gowal et al. (2021b); Sehwag et al. (2020); Sridhar et al. (2022); Chen et al. (2020); Sehwag et al. (2022); Addepalli et al. (2021); Ding et al. (2020); Rade & Moosavi-Dezfooli (2021); Wang et al. (2020b); Engstrom et al. (2019)] listed on the ℓ_∞-RobustBench leaderboard [Croce et al. (2021)]. Additionally, we compare each appearing architecture to a second model *trained without AT or any specific robustness regularization and without any external data* (even if the robust counterpart relied on it).

Non-robust Model Training: For training, *CIFAR-10/-100* data was zero-padded by 4 pixels along each dimension and then transformed using 32×32 pixels random crops and random horizontal flips. Channel-wise normalization was replicated as reported by the original dataset authors. Training hyperparameters have been set to an initial learning rate of 1e-2, a weight decay of 1e-2, a batch size of 256 and a nesterov momentum of 0.9. We scheduled the SGD optimizer to decrease the learning rate every 30 epochs by a factor of $\gamma = 0.1$ and trained for a total of 125 epochs. The loss is determined using Categorical Cross Entropy and we used the model obtained at the epoch with the highest validation accuracy. Training was executed on

a *A+ Server* SYS-2123GQ-NART-2U machine with four *NVIDIA* A100-SXM4-40GB GPUs for approximately 17 GPU hours. Training ImageNet-1k architectures with our hyperparameters resulted in a rather poor performance and we therefore rely on the baseline model without AT provided by *timm* [Wightman (2019)].

Then we collect the predictions alongside their respective confidences of robust and non-robust models on clean validation samples, as well as on samples attacked by a white-box attack (PGD) [Madry et al. (2018)], and a black-box attack (Squares) [Andriushchenko et al. (2020)] as well as common corruptions [Hendrycks & Dietterich (2019)] for CIFAR-10. PGD effectively exposes network-specific vulnerabilities due to its optimization for individual architectures and weights. In our comparison, some robust models might have been trained using PGD or a similar attack. Consequently, PGD could be considered an in-domain attack for those models. In contrast, the *Squares* attack alters the data at random with an allowed budget until the label flips without access to the model architecture and weights. Such samples are rather to be considered out-of-domain samples even for adversarially trained models and provide a proxy for a model's generalization ability. Thus, *Squares* can be seen as an unseen attack for all models, while PGD might not be unseen for some adversarially trained, robust models. Similarly, common corruptions are out-of-domain samples, yet not optimized for the respective network at all.

4.2.2 CIFAR Models

CIFAR-10 Evaluation. As described in Section 2.3, CIFAR-10 [Krizhevsky (2009)] is a simple ten-class dataset consisting of 50.000 training and 10.000 validation images with a resolution of 32×32 pixels. Due to its low resolution and the resulting reduced training costs compared to datasets like ImageNet-1k, CIFAR-10 is a practical choice for AT experiments, which explains its prevalence on RobustBench [Croce et al. (2021)].

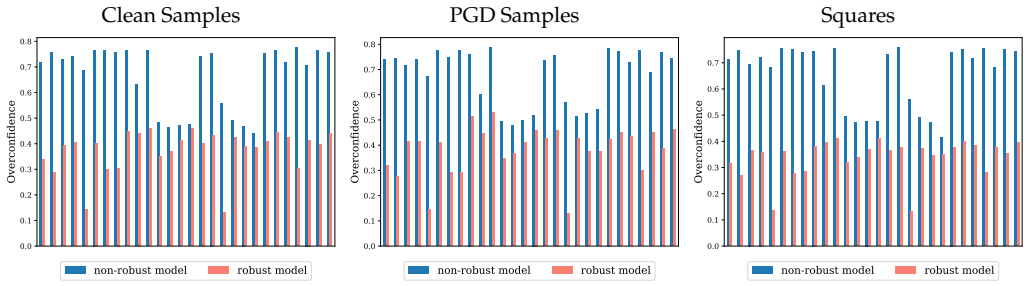


FIGURE 4.2: **Robust Models are Less Over-Confident on CIFAR-10.** Overconfidence (lower is better) bar plots of the robust models and their respective non-robust counterparts trained on CIFAR-10. Non-robust models are highly overconfident, in contrast, their robust counterparts are less overconfident.

Figure 4.1 shows an overview of all robust and non-robust models trained on CIFAR-10 in terms of their accuracy as well as their confidence in their correct and incorrect predictions. Along the isolines, the ratio between confidence in correct and incorrect predictions is constant. The gray area indicates scenarios where models are even more confident in their incorrect predictions than in their correct predictions. Concentrating on the models' confidence, we can observe that robust models (marked by a diamond) are, in general, less confident in their predictions, while non-robust models (marked by a circle) exhibit high confidence in all their predictions, both correct and incorrect. This indicates that non-robust models are not only

more susceptible to (adversarial) distribution shifts but are also highly overconfident in their false predictions. Practically, such behaviour can lead to catastrophic consequences in safety-related, real-world applications. Robust models tend to have lower average confidence and a favourable confidence trade-off even on clean data (Figure 4.1, top left). Considering adversarial samples like PGD (Figure 4.1, top middle), the non-robust models even fall into the gray area of the plot where more confident decisions are likely incorrect. As expected, adversarially trained models not only make fewer mistakes in this case but are also better adjusted in terms of their confidence. Black-box attacks (Figure 4.1,top right) provide non-targeted out-of-domain samples. Adversarially trained models are overall better calibrated to this case, *i.e.* their mean confidences are hardly affected, whereas non-robust models' confidences fluctuate heavily.

TABLE 4.1: **Robust Models have a Lower ECE on CIFAR-10.** Mean ECE (lower is better) and standard deviation over all non-robust model versus all their robust counterparts trained on CIFAR-10. Robust model exhibit a significantly lower ECE.

Robustness \ Samples	Clean ↓	PGD ↓	Squares ↓
non-robust models	0.6736 ± 0.1208	0.6809 ± 0.1061	0.6635 ±0.1156
robust models	0.1894 ±0.1531	0.2688 ± 0.1733	0.2126 ± 0.1431

Four models stand out in Figure 4.1 (top left): two robust and two non-robust models which are much less confident in their true and false predictions than others. These less confident models are indeed trained from two different model architectures, with and without AT. The model by Pang et al. (2020) uses a hypersphere embedding, which normalizes the features in the intermediate layers and weights in the Softmax layer, leading to less confident predictions. The other model proposed by Chen et al. (2020) employs an ensemble of three different pre-trained models (ResNet-50) to boost robustness. These architectural changes have a significant impact on the absolute model confidence, yet do not necessarily lead to a better calibration. These models are underconfident in their correct predictions and tend to be comparably confident in wrong predictions.

Table 4.1 reports the mean ECE over all robust models and their non-robust counterparts and Figure A.1 in the Appendix A.1 presents the ECE of each model and it's non-robust counterpart, showing similar results. Robust models on CIFAR-10 exhibit a significantly lower ECE, suggesting that they are better calibrated. The models' full empirical confidence distributions are given in Figure A.3 in the Appendix A.3 for transparency. Figure 4.2 further visualizes the significant decrease in overconfidence of robust models w.r.t. their non-robust counterparts.

CIFAR-10-C Evaluation. Additionally, to the previously studied attacks, we evaluate the confidence of robust versus non-robust CIFAR-10 models on the out-of-distribution dataset CIFAR-10-C with severity level 4 (the results for other severity levels follow the same trajectory and are therefore omitted). We benchmark models robust to adversarial attacks and their non-robust counterparts and evaluate the prediction confidence.

First, we compare the model's overconfidence with respect to each corruption type. In accordance with our findings on adversarial perturbations, robust models are much less overconfident than their non-robust counterparts. Figure 4.3 depicts

the overconfidence of each model pair for each corruption type. We can clearly see that robust models are generally much less overconfident.

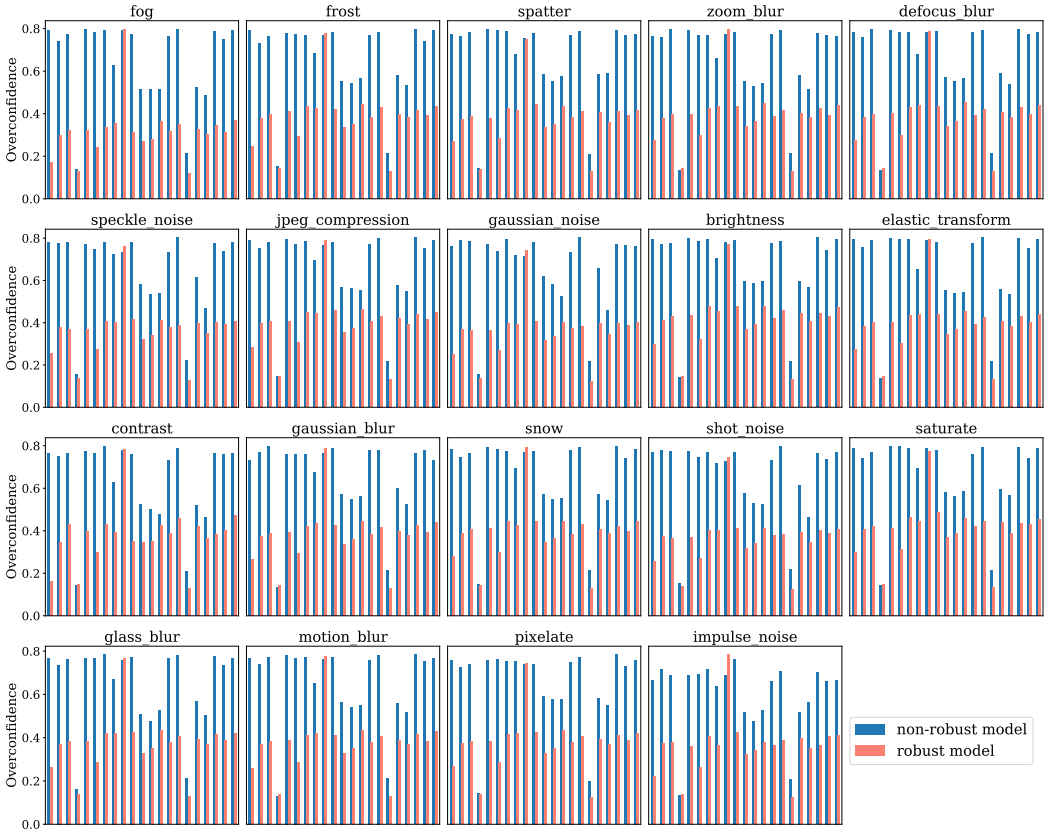


FIGURE 4.3: **Robust Models are Less Over-Confident on CIFAR-10-C.** Overconfidence bar plots of the robust CIFAR-10 model and the respective non-robust counterpart evaluated on CIFAR-10-C. Robust models are less overconfident for most models compared to their non-robust counterparts.

CIFAR-100 Evaluation. Similar resolution as CIFAR-10, CIFAR-100 includes 100 classes and can be seen as a more challenging classification task. This is reflected in the reduced model accuracy on the clean and adversarial samples (Figure 4.1 , bottom). On CIFAR-100, robust models are also less overconfident. They are slightly closer to the optimal calibration point in the lower right corner, even on clean data and perform significantly better on PGD samples where the confidences of non-robust models are again reversed (middle). The Squares attack illustrates the stable behaviour of robust models' confidences. Similar to CIFAR-10, the models' full empirical confidence distributions are given in the Appendix A.3 in Figure A.4 for transparency.

We also report the ECE values for CIFAR-100 in Table 4.2 and Figure 4.5 and overconfidence in Figure A.2 in the Appendix A.2. Please note that the accuracy of the CIFAR-100 models is not very high (ranging between 56.87% and 70.25% even for clean samples), resulting in an unreliable calibration metric. Especially under PGD attacks, non-robust networks make mostly incorrect predictions such that the ECE collapses to being the expected confidence value of incorrect predictions (see eq. [2.9]), regardless of the confidence of the few correct predictions. In this case, ECE is not meaningful. We discuss this in more depth in Section 4.3.

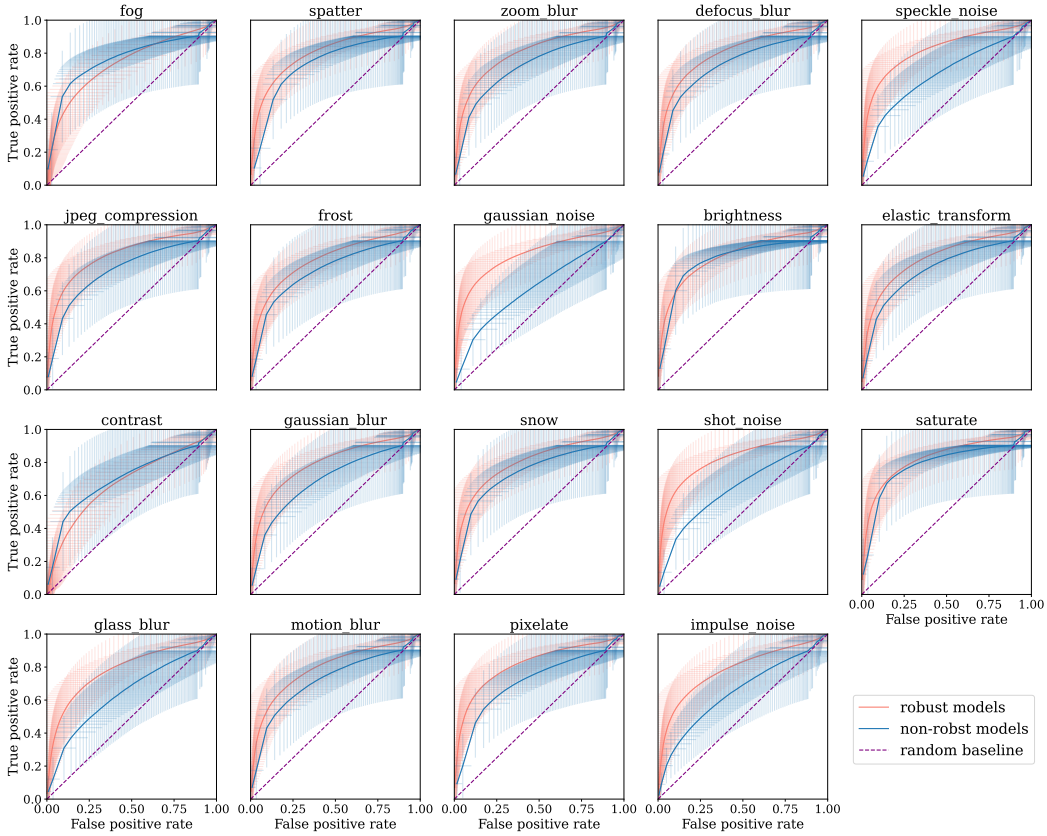


FIGURE 4.4: **Robust Models can Distinguish between Correct and Incorrect Predictions on CIFAR-10-C.** Mean ROC curves for each robust and non-robust CIFAR-10 model pair evaluated on CIFAR-10-C. Robust models can better distinguish between correct and incorrect predictions for all corruptions, but fog, brightness, contrast and saturate.

Another interesting observation is that non-robust models can achieve higher accuracy on the clean data and, quite surprisingly, on the applied black-box attacks (Figure 4.1, right). This indicates that most robust models overfit white-box attacks used during training and do not generalise well to other attacks. While making more mistakes, robust models still have a favourable distribution of confidence over non-robust models in this case.

Model Confidences Can Predict Erroneous Decisions. Next, we evaluate the prediction confidences in terms of their ability to predict whether a network prediction is correct or incorrect. We visualize the ROC curves for all models and compare the averages of robust and non-robust models in Figure 4.6 (top row for CIFAR-10, bottom row for CIFAR-100). While robust and non-robust models perform on average very similarly on clean data, robust model confidences can reliably predict erroneous classification results on adversarial examples where non-robust models fail. Also, for out-of-domain samples from the black-box attack Squares (middle right) and common corruptions (right), robust models can reliably assess their prediction quality and can better predict whether their classification result is correct. In Figure 4.4 we present the mean ROC curves for each corruption. The ROC curve analysis reveals that robust models demonstrate improved ability to separate correct from incorrect predictions using confidence values when compared to non-robust models. However, robust models are inferior regarding their calibration on corruptions

TABLE 4.2: **ECE Evaluation on CIFAR-100.** Mean ECE (lower is better) and standard deviation for all non-robust models versus their robust counterparts trained on CIFAR-100. The robust models exhibit a slightly lower ECE on clean samples and a significantly lower ECE on Square samples. However, on PGD samples, the robust models actually exhibit a higher ECE compared to their non-robust counterparts.

Robustness \ Samples	Clean ↓	PGD ↓	Squares ↓
non-robust models	0.3077 ± 0.1257	0.2159 ± 0.0738	0.2780 ± 0.1348
robust models	0.2962 ± 0.1722	0.2307 ± 0.1494	0.2076 ± 0.1247

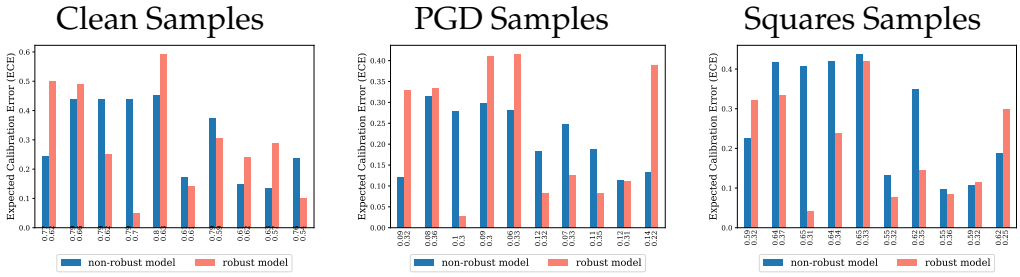


FIGURE 4.5: **ECE Bar Plots on CIFAR-100.** ECE (lower is better) bar plots of robust models and their non-robust counterparts trained on CIFAR-100. The models' accuracy is marked for the different samples below each bar. The ECE score of the robust models for the CIFAR-100 is much worse than the one for CIFAR-10. Interestingly the robust models seem to be as bad as their non-robust counterparts or even worse on clean and PGD samples.

changing the image's colour palette, like fog, brightness, contrast, and saturation. We report the precision recall curves for CIFAR-10 and CIFAR-100 in Figures A.5 and A.6 in the Appendix A.4, which present similar results. The non-robust models perform slightly better on clean samples while being completely fooled by PGD samples and less well on Square samples compared to their robust counterparts.

Robust Model Confidences Can Detect Adversarial Samples. Further, we evaluate the adversarial detection rate of the robust models based on their ROC curves (averaged over all robust models) in Figure 4.7, comparing the confidence of correct predictions on clean samples and incorrect predictions caused by adversarial attacks. We observe different behaviours for gradient-based, white-box attacks and black-box attacks. While non-robust models fail completely against gradient-based attacks, they are almost as good as robust models for the detection of black-box attacks. Similarly, when taking the left two plots from Figure 4.6 into account, one might get the impression that non-robust models perform similar or even better on detecting erroneous samples compared to robust ones. Thus, we hypothesize that robust models indeed overfit the adversaries seen during training, as those are mostly gradient-based adversaries. Therefore, we assume that adversarially trained models are not better calibrated in general; however, when strictly looking at overconfidence, robust models are consistently less overconfident and, therefore, better applicable for safety-critical applications. We report similar results for the precision recall curves in Figure A.7 in Appendix A.4.

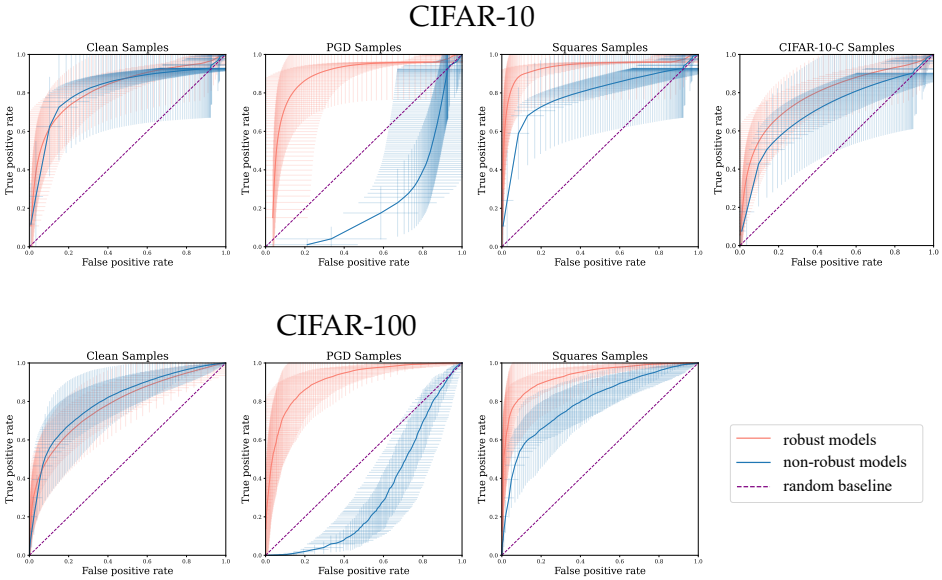


FIGURE 4.6: **Robust Models can Better Distinguish between Correct and Incorrect Predictions on CIFAR-10 and CIFAR-100 under attack.** Average ROC curve for all robust and non-robust models trained on CIFAR-10 (top) and CIFAR-100 (bottom). Standard deviation is marked by the error bars. The dashed line would mark a model which has the same confidence for each prediction. We observe that the model's confidence can be an indicator of the correctness of the prediction. On PGD samples the non-robust models fail while the robust models can distinguish correct from incorrect predictions based on the prediction confidence.

Improved CNN Building Blocks. Moreover, we see indications that exchanging simple building blocks like the activation function [Dai et al. (2022)] or the downsampling method, explained in depth in Section 6.2, alters the properties of robust models concerning their confidence calibration. We investigate these models' prediction confidence and observe that their correct prediction confidence is high while the confidence in the erroneous predictions remains low. Our findings should nurture future research on jointly considering model calibration and robustness.

Downsampling Technique. Most common CNNs apply downsampling to compress feature maps with the intent to increase spatial invariance and overall higher sparsity. However, we show in Chapter 5 that aliasing during the downsampling operation highly correlates with the lack of adversarial robustness and provide aliasing-free downsampling in Chapter 6, which enables improved downsampling of the feature maps. Figure 4.8, left, compares the confidence distribution of three different networks. The top row shows a PreAct-ResNet-18 (PRN-18) baseline without AT, the second row shows our approach applied to the same architecture, and the third row shows a robust model trained by Rebuffi et al. (2021) without improved downsampling. The baseline model is highly susceptible to adversarial attacks, especially under white-box attacks, while the two robust counterparts remain low-confident in false predictions and show higher confidence in correct predictions. However, while the model by Rebuffi et al. (2021) has a high variance amongst the predicted confidences, our approach significantly improves this by disentangling the confidences. Our model provides low variance and high confidence in correct predictions and reduced confidence in false predictions across all evaluated samples.

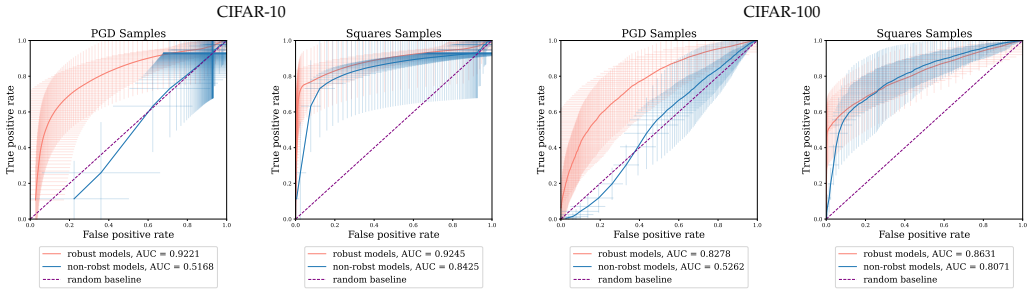


FIGURE 4.7: **Robust Models can Use Prediction Confidence as Attack Detection on CIFAR-10 and CIFAR-100.** Average ROC curve for all robust and non-robust models, comparing confidence on clean, correctly classified samples to perturbed, wrongly classified samples on CIFAR-10 and CIFAR-100. The confidence of robust models can be used as thresholds for detecting white-box adversarial attacks (PGD). For black-box adversarial attacks (Squares), both robust and non-robust models can partially detect erroneous samples.

Activation Function. Next, we analyse the influence of activation functions. Only one RobustBench model utilizes an activation other than ReLU. Dai et al. (2022) introduced learnable activation functions with the intent to improve robustness. Figure 4.8, right, shows at the top row a Wide-ResNet-28-10 (WRN-28-10) baseline model without AT, the model by Dai et al. (2022) in the middle and a model with the same architecture adversarially trained by Carmon et al. (2019a) without improved activation function.
Although this is an arguably sparse basis for a thorough investigation, we observe that the model by Dai et al. (2022) can retain high confidence in correct predictions for both clean and perturbed samples. Furthermore, the model is much less confident in its wrong predictions for the clean as well as the adversarial samples. Similar to the used pooling variation, also the activation function seems to influence the model's calibration.

To show the impact of improved downsampling and activation functions, we provide the ROC curves and AUC values of the models with and without those improved building blocks (similar to Figure 4.8). Figure 4.9 presents the ROC curves and AUC values for each approach on the improved building blocks as well as on comparable robust models with the same architecture. The improved building blocks result in slightly better calibration.

Next, we compare the confidence impact of improved downsampling operations and activations on out-of-distribution data like CIFAR-10-C. We summarize our findings by the mean over all corruptions. Figure 4.10 shows that robust models are, on average, better calibrated than non-robust models. The impact of improved downsampling or activation functions is marginal.

Summary of Low-Resolution Datasets. On CIFAR-10 and CIFAR-100 non-robust models can achieve higher standard accuracy and at least match or even exceed the performance of robust models under black-box attacks like Squares. Only under the white-box attack PGD, the robust models show higher accuracy. However, non-robust models are highly overconfident in all their predictions and are hence limited in their applicability for real-world tasks. In contrast, the correctness of a robust model's prediction can be estimated by the prediction confidence and can additionally serve as a defence against adversarial attacks. Further, we observe that the confidence of non-robust models decreases with increasing task complexity. In

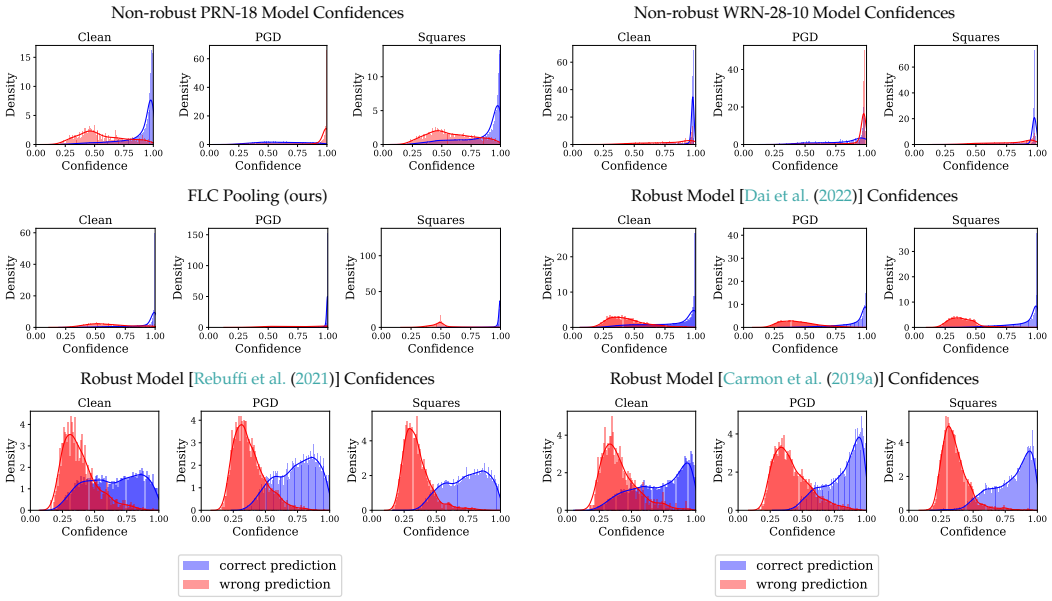


FIGURE 4.8: **Adapting CNN Building Blocks for Optimized Confidence Distributions.** Left: Confidence distribution on three different PRN-18. The first row shows a model without adversarial training and standard pooling, the second row the model which uses FLC Pooling which we introduce in Chapter 6 instead of standard pooling and the third row shows the model by Rebuffi et al. (2021) adversarially trained and with standard pooling. Right: Confidence distribution on three different WRN-28-10. The first row shows a model without adversarial training and standard activation (ReLU), the second row the model by Dai et al. (2022) which uses learnable activation functions instead of fixed ones and the third row shows the model by Carmon et al. (2019a) adversarially trained and with the standard activation (ReLU)

contrast, robust models are less affected by the increased task complexity and exhibit similar confidence characteristics on both datasets.

4.2.3 ImageNet-1k Models

We rely on the models provided by RobustBench [Croce et al. (2021)] for our ImageNet-1k evaluation. We report the clean and robust accuracy against PGD and Squares with an ϵ of $4/255$ in Table 4.3. The non-robust model, trained without AT, achieves the highest performance on clean samples but collapses under white- and black-box attacks. Further, the models trained with multistep adversaries by Engstrom et al. (2019) and Salman et al. (2020) achieve higher robust and clean accuracy than the model trained by Wong et al. (2020) which is trained with single-step adversaries. Moreover, the largest model, a WRN-50-2, yields the best robust performance. Still, the amount of robust networks on ImageNet-1k is quite small, thus we can not make any generalized assumptions. In terms of overconfidence and ECE, a clear trend is recognizable in Figure 4.11; The non-robust models exhibit a much higher ECE and are also highly over-confident on all samples. Figure 4.12 depicts the precision-recall curve for our evaluated models. Under evaluation with clean samples, the non-robust model without AT performs best. Under both attacks (PGD and Squares), the largest model, Wide-ResNet-50-2 (WRN-50-2) by Salman et al. (2020) performs best and the worst performer is the smallest model, a ResNet-18 (RN-18). This may suggest that bigger models can not only achieve a better trade-off in clean and robust

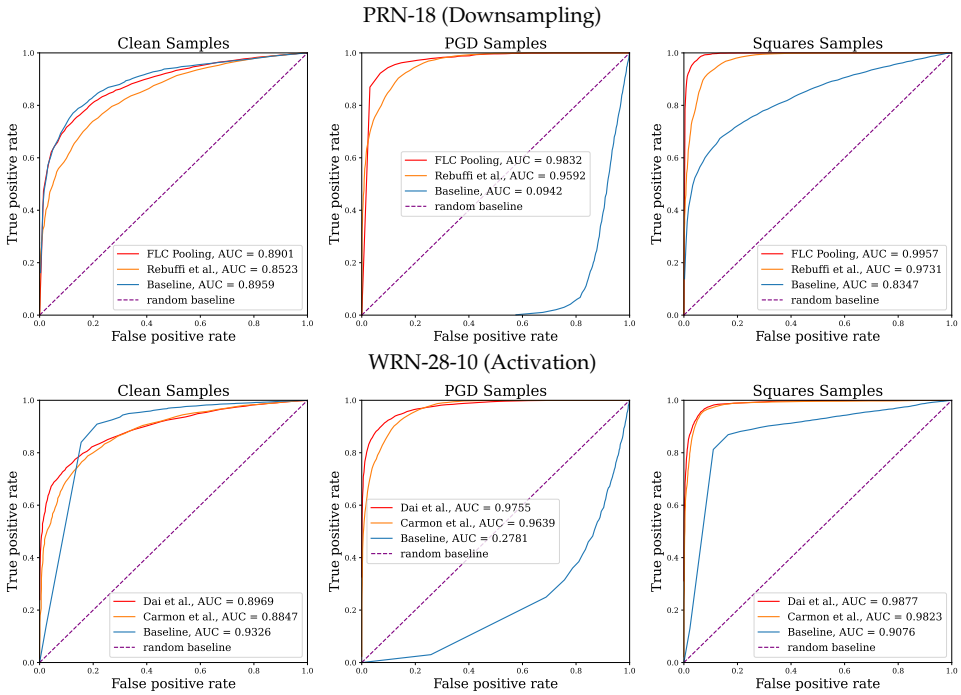


FIGURE 4.9: **Adapting CNN Building Blocks for Optimized ROC curves on CIFAR-10.** ROC curves for robust models with and without improved building block, like downsampling (top) and activation function (bottom).

accuracy but are also more successful in disentangling confidences between correct and incorrect predictions.

TABLE 4.3: **Performance Evaluation for Robust Models on ImageNet-1k.** Clean and robust accuracy against PGD and Squares (higher is better) of ResNet-50 trained on ImageNet-1k with AT over 10000 samples.

Method	Architecture	Acc@1 ↑	PGD ↑	Squares ↑
Baseline	RN50	**76.13**	0.00	11.48
Engstrom et al. (2019)	RN50	62.41	35.47	54.93
Wong et al. (2020)	RN50	53.83	29.43	42.26
Salman et al. (2020)	RN50	63.87	42.23	56.58
Salman et al. (2020)	WRN50-2	68.41	**44.75**	**61.29**
Salman et al. (2020)	RN18	52.50	31.92	43.81

4.3 Discussion

Our experiments confirm that non-robust models are highly overconfident, especially under gradient-based, white-box attacks. However, when confronted with clean samples, common corruptions or unseen black-box attacks like Squares, non-robust and robust models are equally able to detect wrongly classified samples based

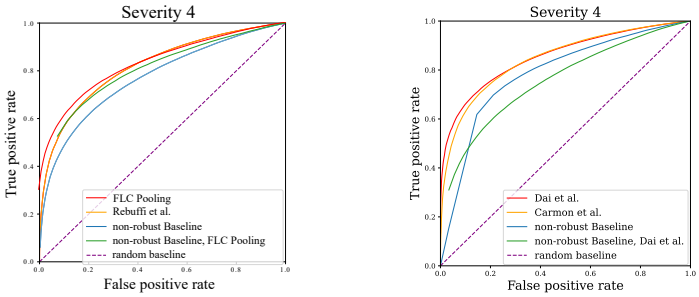


FIGURE 4.10: **Adapting CNN Building Blocks for Optimized ROC curves on CIFAR-10-C.** ROC curve for improved downsampling (left) and activation function (right) on the mean CIFAR-10-C corruptions with severity 4. Robust models are superior to the normal models, and, the impact of activation and pooling is marginal.

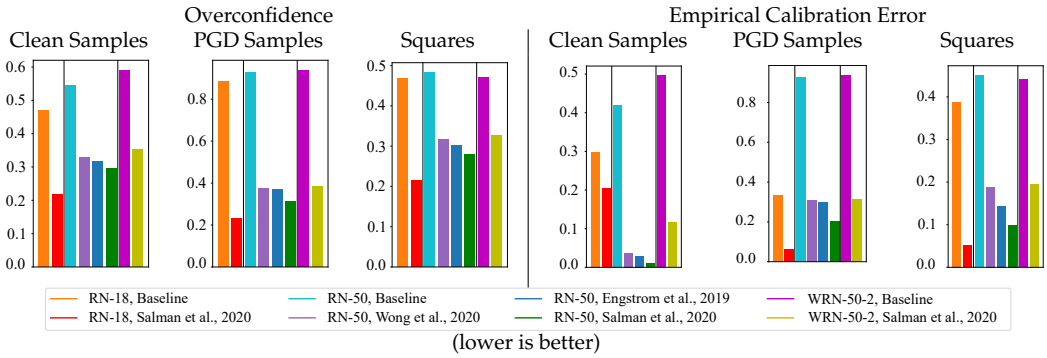


FIGURE 4.11: **Robust Models are Less Over-Confident on ImageNet-1k.** Overconfidence (left) and ECE (right) (lower is better) bar plots of the models trained on ImageNet provided by RobustBench [Croce et al. (2021)] and their respective non-robust counterparts. The non-robust baselines exhibit the highest overconfidence and ECE. In contrast, the robust models are better calibrated.

on their prediction confidence. Indicating that adversarially trained networks overfit the kind of adversaries seen during training.

Moreover, our results indicate that the selection of the activation functions as well as the downsampling, are important factors for the models' performance and confidence. Our method, which improves the downsampling, as well as the method by Dai et al. (2022), which improves the activation function, exhibit the best calibration for the network's prediction; High confidence on correct predictions and low confidence on the incorrect ones.

A further point to discuss is that the ECE is not a well-suited measure as it aggregates the information too strongly. Thus, the network could be highly confident for wrong samples and low-confident in true samples, yet averaged, it might be on par with the overall accuracy for the reported class.

4.3.1 Limitations

Our evaluation is based on the models provided on RobustBench [Croce et al. (2021)]. Thus, the amount of networks on more complex datasets, like ImageNet-1k, is rather small and therefore, the evaluation not universally applicable. While the number of

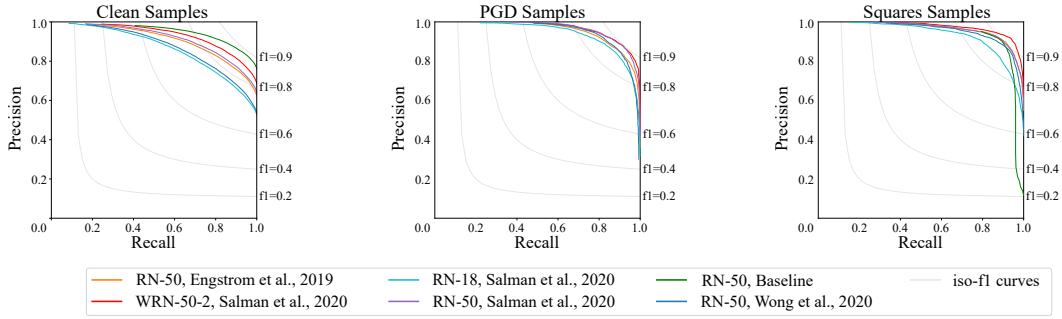


FIGURE 4.12: **Robust Models can Disentangle Confidences between Correct and Incorrect Predictions on ImageNet-1k.** Precision Recall curves for the classification of correct versus erroneous predictions based on the confidence on ImageNet-1k, evaluated over 10,000 samples. Robust and non-robust models are taken from RobustBench [Croce et al. (2021)]. For clean samples (left) the non robust baseline performs best, while its confidences are less reliable under attack (middle and right). The robust WRN-50-2 by Salman et al. (2020) performs best on the PGD and Squares samples.

models for CIFAR is large, the proposed database can only be understood as a starting point for future research. Further, the analysis of neural network building blocks *e.g.* our analysis on the use of improved building blocks, is limited to single-model evaluations, potentially leading to inconclusive results. A deeper investigation of downsampling blocks done in Chapter 6 reveals significant variability in confidence distribution improvements across different random seeds, highlighting their potential unreliability. This observation emphasizes the need for further exploration of the impact of training hyperparameters on such essential properties. Additionally, we rely simply on the confidence obtained after the Softmax layer, while there are many other metrics for uncertainty measurement. *E.g.* Guo et al. (2017) introduced temperature scaling which is a variant of Platt Scaling [Platt (1999)] to calibrate neural networks predictions. However, including temperature scaling assumes that the model is refined or trained with this additional scaling factor. As we wanted to study the networks as they are, the Softmax is the most straight-forward method.

4.4 Conclusion

We provide an extensive study on the confidence of robust models and observe an overall trend: Robust models tend to be less overconfident than non-robust models. Thus, while achieving a higher robust accuracy, AT generates models that are less overconfident. Further, the prediction confidence of robust models can actually be used to reject wrongly classified samples on clean data and even adversarial examples.

Moreover, we observe indications that modifying basic building blocks, such as the activation function [Dai et al. (2022)] or the downsampling method, affects the confidence calibration properties of robust models. Specifically, in our investigated examples, these models exhibit increased prediction confidence for correct predictions while maintaining low confidence for erroneous predictions. Thus, when optimizing the architectures (which we do in the following *e.g.* Chapter 6) and training schemes of deep neural networks, we should address model robustness and calibration jointly rather than treating them as separate aspects. This approach is essential for developing reliable and robust networks suitable for real-world applications.

Chapter 5

Aliasing and Adversarially Robust Generalization of CNNs

Contents

In this chapter, we analyse adversarially trained, robust models in the context of a specific network operation, the downsampling layer, and provide evidence that robust models have learned to downsample more accurately and suffer significantly less from downsampling artifacts, aka. aliasing, than non-robust models. We propose a novel aliasing measure with which we can quantify the amount of aliasing introduced after a downsampling operation. In the case of catastrophic overfitting, we observe a strong increase in aliasing and propose a novel early stopping approach based on our proposed aliasing measure.

This chapter is based on Grabinski et al. (2022c) and Grabinski et al. (2022d) . Julia Grabinski, as the first author, conducted all experiments and was the main writer.

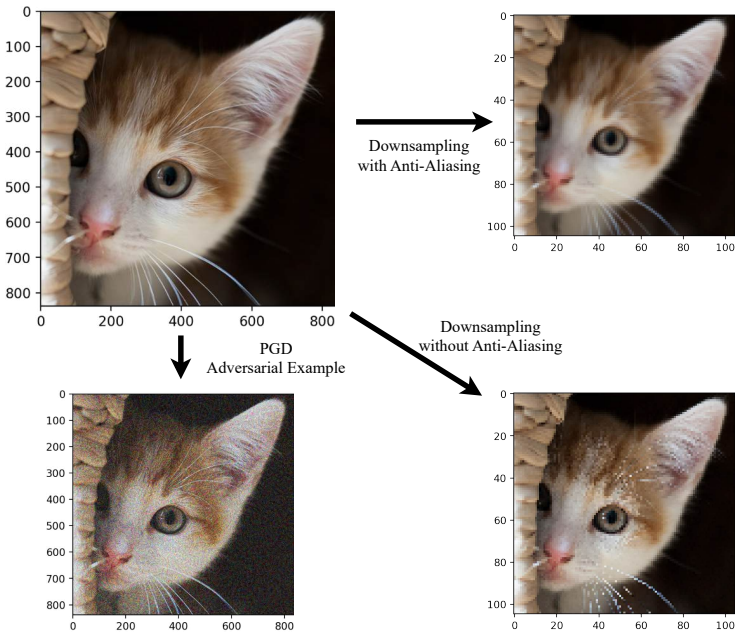


FIGURE 5.1: **Similarities between Aliasing and Adversarial Attack.** Illustration of downsampling with (top right) and without an anti-aliasing filter (bottom right), as well as an adversarial example (bottom left). The top left image shows the original. In the top right, the image is correctly downsampled using an anti-aliasing filter. In the bottom right, no filter is applied, leading to aliasing. The adversarial example (bottom left) exhibits visually similar artifacts. In this chapter, we investigate the role of aliasing in adversarial robustness.

5.1 Introduction

CNNs provide highly accurate predictions in a wide range of applications. Yet, to allow for practical applicability, CNN models should not be fooled by small image perturbations, as they are realized by adversarial attacks [Goodfellow et al. (2015); Moosavi-Dezfooli et al. (2016); Rony et al. (2019)]. Such attacks are optimized to find image perturbations such that the network makes incorrect predictions while human observers are not distracted by these perturbations and can still easily recognize the correct class label. Susceptibility to such perturbations is prohibitive for the applicability of CNN models in real-world scenarios, as it indicates limited reliability and generalization of the model.

A range of sophisticated methods have been proposed to improve adversarial robustness [Goodfellow et al. (2015); Rony et al. (2019); Madry et al. (2018)]. However, a clear trade-off exists: while methods like [Madry et al. (2018); Wu et al. (2020); Zhang et al. (2019b)] offer strong defenses against diverse attacks, they are computationally demanding. In contrast, resource-efficient methods, such as [Goodfellow et al. (2015)], demonstrate good performance against simple adversaries but suffer from catastrophic overfitting, resulting in a loss of generalization against stronger attacks like PGD [Madry et al. (2018)].

In fact, when considering the architecture design of commonly employed CNN models, we might have found one factor why these models perform well in the standard setting but can be fooled easily.

Concretely, most architectures sub-sample feature maps without ensuring to sample above the Nyquist rate [Shannon (1949)], such that, after each downsampling operation, spectra of sub-sampled feature maps may overlap with their replica. This

is called *aliasing* and implies that the network should be genuinely unable to fully restore an image from its feature maps. Following this line of thought, recently, several publications suggest improving CNNs by including anti-aliasing techniques during downsampling in CNNs [Zhang (2019); Zou et al. (2023); Li et al. (2021); Hossain et al. (2023)]. They aim to make the models more robust against image translations, such that the class prediction does not suffer from small vertical or horizontal shifts of the content.

In contrast, we investigate the relationship between adversarial robustness and aliasing. Thus, we introduce a novel aliasing measure and compare several recently proposed adversarially robust models to conventionally trained models in terms of aliasing. We inspect intermediate feature maps before and after the downsampling operation at inference. Our first observation is that these models indeed fail to subsample according to the Nyquist Shannon Theorem [Shannon (1949)]: we observe severe aliasing. Further, our experiments reveal that adversarially trained networks exhibit less aliasing than standard trained networks, indicating that AT encourages CNNs to learn how to properly downsample data without severe artifacts. Next, we visualize the frequency spectra of adversarial attacks on baseline models as well as on adversarially trained ones. Our experiments show that attacks behave in a less characteristic spectrum when attacked models are adversarially robust. This indicates that adversarial attacks might employ network aliasing as a backdoor, such that high-frequency changes can flip the network decision, while attacks on adversarially robust networks have to hamper with the low-frequency components of the image, *i.e.* the coarse details. Additionally, we ablate on the importance of spatial encoding for robustness and compare the attack structures of adversaries created for CNNs versus fully connected networks (FCNs), which do not inherently encode the spatial relations. Finally, we investigate the behaviour during FGSM AT and observe a strong correlation between catastrophic overfitting during training and the amount of aliasing in the network's downsampling operations. Specifically, the amount of aliasing increases significantly as the model overfits to the FGSM perturbations. Based on these findings, we propose a new early stopping criterion based on our measurement of aliasing to prevent catastrophic overfitting during AT.

In summary, our contributions are:

- We introduce a novel measure for aliasing and show that common CNN downsampling layers fail to sub-sample their feature maps in a Nyquist-Shannon conform way.
- We analyse various adversarially trained models, that are robust against a strong ensemble of adversarial attacks, AutoAttack [Croce & Hein (2020a)], and show that they exhibit significantly less aliasing than standard models.
- We ablate on the importance of spatial encoding for aliasing and adversarial robustness and show that attacks on FCN are more centered towards the objects and thus stronger visible for the human eye.
- We present strong evidence that catastrophic overfitting coincides with an increased amount of aliasing for several network architectures.
- We introduce a new early stopping criterion for FGSM AT based on our aliasing measure.

5.2 Method

CNNs usually have a pyramidal structure in which the data is progressively sub-sampled in order to aggregate spatial information while the number of channels increases. During sub-sampling, no explicit precautions are taken to avoid aliases, which arise from under-sampling. Specifically, when sub-sampling with stride 2, any frequency larger than $N/2$, where N is the size of the original data, will cause pathological overlaps in the frequency spectra (Figure 2.4 in Section 2.4.5). Those overlaps in the frequency spectra cause ambiguities such that high-frequency components appear as low-frequency components. Hence, local image perturbations can become indistinguishable from global manipulations.

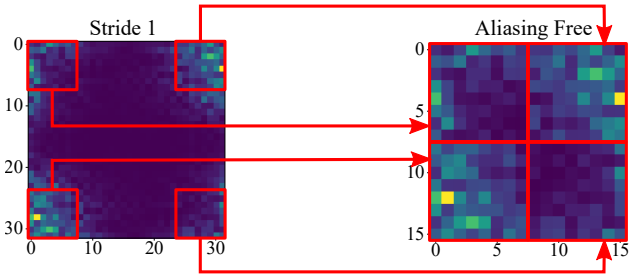


FIGURE 5.2: **Aliasing Measure, Computation of the Aliasing-Free Feature Map.** Step-by-step computation of the aliasing-free version of a feature map. The left image depicts the magnitude of the Fourier representation (not center-shifted) of a feature map, with the zero-frequency component in the upper-left corner (*i.e.*, high-frequency components are in the center). Aliasing-free downsampling suppresses high-frequency components prior to sampling. This can be efficiently implemented in the Fourier domain by cropping and reassembling the low-frequency regions of the Fourier representation, *i.e.*, its four corners. Aliasing corresponds to folding the deleted high-frequency components into the reconstructed representation

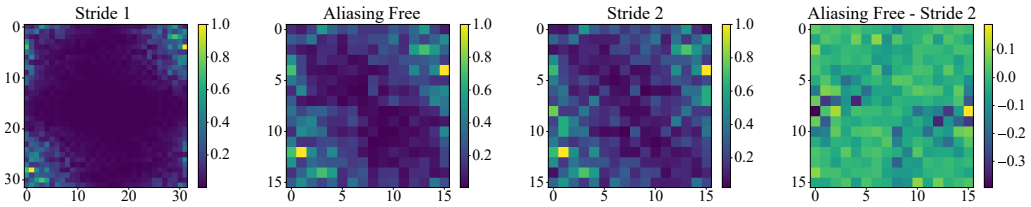


FIGURE 5.3: **Aliasing Measure, Overview.** Left: The FFT-transformed feature map (not center-shifted) at the original resolution. Middle left: Downsampling of this feature map as described in Figure 5.2. Middle right: FFT-transformed, standard downsampled feature map (not center-shifted) with a stride of two. Right: Difference between the aliasing-free and standard downsampled feature maps in the Fourier domain.

5.2.1 Aliasing Measure

To measure the possible amount of aliasing appearing after downsampling, we compare each downsampled feature map in the Fourier domain with its aliasing-free counterpart. To this end, we consider a feature map $g(x)$ of size $2N \times 2N$ before downsampling. We compute an "aliasing-free" downsampling by extracting the N

lowest frequencies along both axes in Fourier space. W.l.o.G. , we consider specifically downsampling operations by strided convolutions since these are predominantly used in adversarially robust models [Zagoruyko & Komodakis (2016)].

In each strided convolution, the input feature map $g(x)$ is convolved with the learned weights w and downsampled by a stride of two, thus potentially introducing frequency replica (*i.e.* aliases) in the downsampled signal $\hat{g}_{s2}$.

$$\hat{g}_{s2} = g(x) * k(w, 2) \tag{5.1}$$

Afterwards the 2D FFT of the new feature maps $\hat{g}_{s2}$ is computed, which we denote G_{s2}.

$$G_{s2}(k,l) = \frac{1}{N^2} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \hat{g}_{s2}(m,n) e^{-2\pi j(\frac{k}{N}m + \frac{l}{N}n)}, \tag{5.2}$$

To measure the amount of aliasing, we explicitly construct feature map frequency representations without such aliases. Therefore, the original feature map $g(x)$ is convolved with the learned weights w of the strided convolution without applying the stride $k(w, 1)$ to obtain $\hat{g}_{s1}$.

$$\hat{g}_{s1} = g(x) * k(w, 1) \tag{5.3}$$

for $k, l = 0, \ldots, N-1$. For the non-downsampled feature maps $\hat{g}_{s1}$, we proceed similarly and compute for $k, l = 0, \ldots, 2 \cdot N - 1$

$$G_{s1}(k,l) = \frac{1}{4N^2} \sum_{m=0}^{2N-1} \sum_{n=0}^{2N-1} \hat{g}_{s1}(m,n) e^{-2\pi j(\frac{k}{2N}m + \frac{l}{2N}n)}. \tag{5.4}$$

The aliasing-free version G_{s1} can be obtained by setting all frequencies above the Nyquist rate to zero before downsampling,

$$G_{s1}^{\uparrow}(k,l) = 0 \tag{5.5}$$

for $k \in [N/2, 3N/2]$ and for $l \in [N/2, 3N/2]$. Then the downsampled version in the frequency domain corresponds to extracting the four corners of $G_{s1}^{\uparrow}$ and reassembling them as shown in Figure 5.2,

$$\begin{aligned}
G_{s1}(k,l) &= G_{s1}^{\uparrow}(k,l) && \text{for} \quad k,l = 0, \ldots, N/2 \\
G_{s1}(k,l) &= G_{s1}^{\uparrow}(k+N,l) && \text{for} \quad k = N/2, \ldots, N \\
& && \text{and} \quad l = 0, \ldots, N/2 \\
G_{s1}(k,l) &= G_{s1}^{\uparrow}(k,l+N) && \text{for} \quad k = 0, \ldots, N/2 \\
& && \text{and} \quad l = N/2, \ldots, N \\
G_{s1}(k,l) &= G_{s1}^{\uparrow}(k+N,l+N) && \text{for} \quad k,l = N/2, \ldots, N
\end{aligned} \tag{5.6}$$

This way, we guarantee that there are no overlaps, *i.e.* aliases, in the frequency spectra. Figure 5.2 illustrates the computing process of the aliasing-free downsampling in the Fourier domain. The aliasing-free feature map can be compared to the actual feature map in the Fourier domain to measure the degree of aliasing. The full procedure is presented in Figure 5.3, where we start on the left with the original feature map. Then, we obtain the two downsampled versions (with and without aliases) and compute their L_1 difference.

The overall aliasing measure, AM, for a downsampling operation is calculated by

the L_1 distance between downsampled and aliasing-free feature maps in the Fourier domain, averaged over all K feature maps,

$$\mathrm{AM} = \frac{1}{K} \sum_{k=0}^{K} \|G_{s1,k} - G_{s2,k}\|. \tag{5.7}$$

The proposed AM is zero when aliasing is absent in all downsampled feature maps, indicating sampling above the Nyquist rate. An AM value exceeding zero signifies the presence of aliasing, suggesting theoretical vulnerability to adversarial attacks due to the model's inability to reliably differentiate between fine details and coarse input structures.

5.3 Experiments

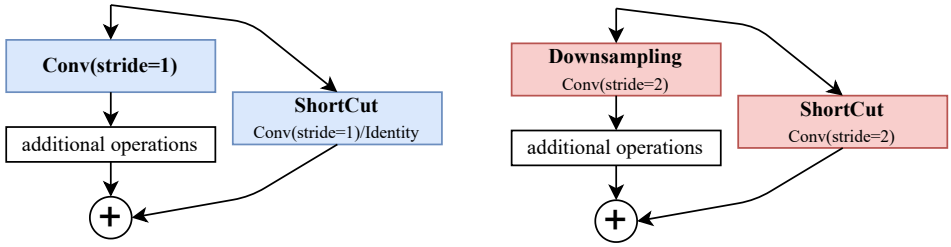


FIGURE 5.4: **Abstract Illustration of a Building Block in ResNet architectures.** Abstract illustration of a building block in PRN-18 and WRN-28-10. The first operation in a block is a convolution, executed with a stride of either one or two. For a stride of one (left), the shortcut simply passes the identity of the feature maps forward. If the first convolution is executed with a stride of two, the shortcut must also have a stride of two (right) to ensure that both representations can be added at the end of the building block.

5.3.1 Aliasing in Existing Models

We conduct an extensive analysis of already existing adversarially robust models trained on CIFAR-10 [Krizhevsky (2009)] with two different architectures, namely WideResNet-28-10 (WRN-28-10) [Zagoruyko & Komodakis (2016)] and Preact ResNet-18 (PRN-18) [He et al. (2016b)]. These architectures are widely supported by numerous AT approaches on RobustBench [Croce et al. (2021)] while remaining compact enough to allow efficient execution and training. As a baseline, we trained a plain WRN-28-10 and PRN-18, both with similar training schemes on an NVIDIA Tesla V100. Each model is trained with 200 epochs, a batch size of 512, CrossEntropy loss and SGD with an adaptive learning rate starting at 0.1 and reducing it at 100 and 150 epochs by a factor of 10, a momentum of 0.9 and a weight-decay of 5e-4. All adversarially trained networks are pre-trained models provided by RobustBench [Croce et al. (2021)].

Both architectures have similar building blocks and the key operations, including downsampling, are shown abstractly in Figure 5.4. Each block starts with a convolution with stride two, followed by additional operations like ReLU and convolutions with stride one. The characteristic skip connection of ResNet architectures

also needs to be implemented with stride two if downsampling is applied in the corresponding block. Consequently, we need to analyse all downsampling units and skip connections before they are summed up to form the output feature map.

The WRN-28-10 networks have four operations in which downsampling is performed. These operations are located in the second and third blocks of the network. In comparison, the PRN-18 networks have six downsampling operations, located in the second, third and fourth layers of the network.

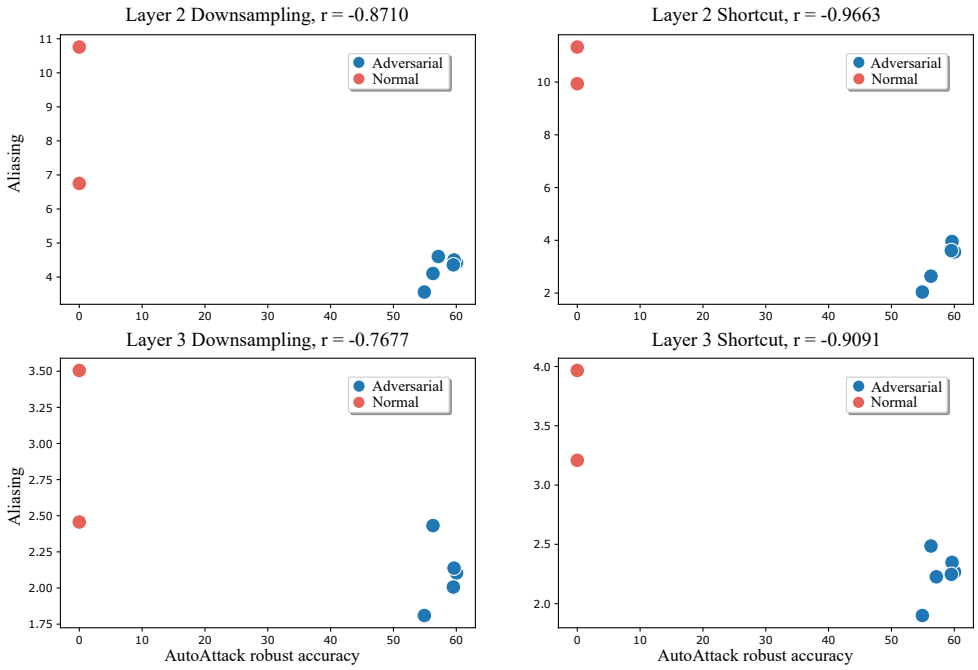


FIGURE 5.5: **Adversarial Robustness vs. Aliasing on WRN-28-10 models.** Adversarial robustness versus aliasing and the according correlation r, evaluated on different pre-trained WRN-28-10 models from RobustBench [Croce et al. (2021)] as well as two baseline models without AT, one from RobustBench and one trained by us. All blue dots represent adversarially trained networks and the light red ones represent standard trained models.

WideResNet 28-10. In the following, we compare differently trained WRN-28-10 networks in terms of their robust accuracy against AutoAttack [Croce & Hein (2020a)] and the amount of aliasing after downsampling.

Figure 5.5 indicates significant differences between adversarially trained and standard trained networks. First, the networks trained without AT are not able to reach any robust accuracy, meaning their accuracy under adversarial attacks is equal to zero. Second, and this is most interesting for our investigation, standard trained networks exhibit much more aliasing during their downsampling layers than adversarially trained networks. Through all layers and operations in which downsampling is applied, the adversarially trained networks (blue dots Figure 5.5) have much higher robust accuracy and much less aliasing compared to the standard trained networks (red dots Figure 5.5). We indicate the Pearson correlation r between aliasing and robust accuracy above each scatter plot in Figure 5.5, indicating a significant negative correlation. Additionally, we can observe that the amount of aliasing in the second layer is much higher than in the third layer. This can be explained by the different feature map sizes in the two layers as we calculate the absolute L_1 norm.

When comparing the conventionally trained networks against each other, it can be observed that the specific training scheme used for training the network can have an influence on the amount of aliasing of the network. Concretely, the standard baseline model provided by RobustBench [Croce et al. (2021)] exhibits less aliasing than the one trained by us. Unfortunately, there is no further information about the exact training schedule from RobustBench, such that we can not make any assumptions on the interplay between training hyperparameters and aliasing.

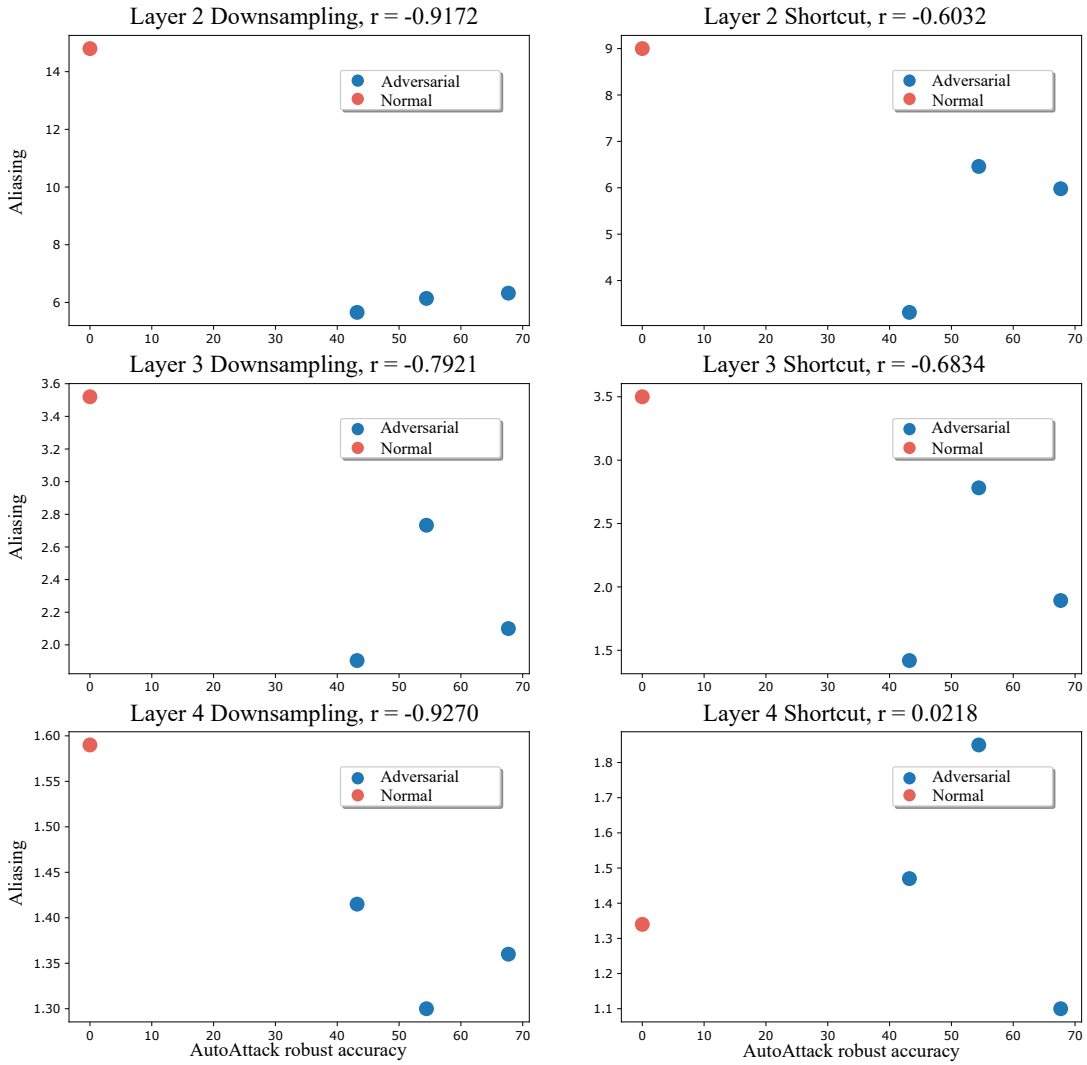


FIGURE 5.6: **Adversarial Robustness vs. Aliasing on PRN-18 models.** Adversarial robustness versus aliasing and the according correlation r, exemplary evaluated on different pretrained PRN-18 models. The blue dots represent adversarial trained networks and the light red dot represent a standard trained model without AT.

Preact ResNet-18. Consistent with our approach for WRN-28-10, we performed identical measurements and employed the same training protocol for PRN-18 models while accounting for an extra layer and two additional downsampling steps.

The overall results, presented in Figure 5.6, are similar to the ones for the WRN-28-10 networks; most adversarially trained networks exhibit significantly less aliasing and higher robustness than conventionally trained ones. Yet, the additional downsampling layer allows one further observation. While the absolute aliasing

measure is overall lower, the robust networks reduce the amount of aliasing predominantly in the earlier layers, the second and third layers. The amount of aliasing in the fourth layer of adversarially robust models is not significantly different from the amount of aliasing in conventionally trained models in the same layer. This phenomenon might be explained by the sparsity of the deeper layers. While the earlier feature maps represent the spatial properties of input images, deeper layers rather encode semantic properties that are sparsely encoded and, therefore, might be harder to affect. Yet, this aspect needs further investigation for a better understanding, which goes beyond the scope of this work.

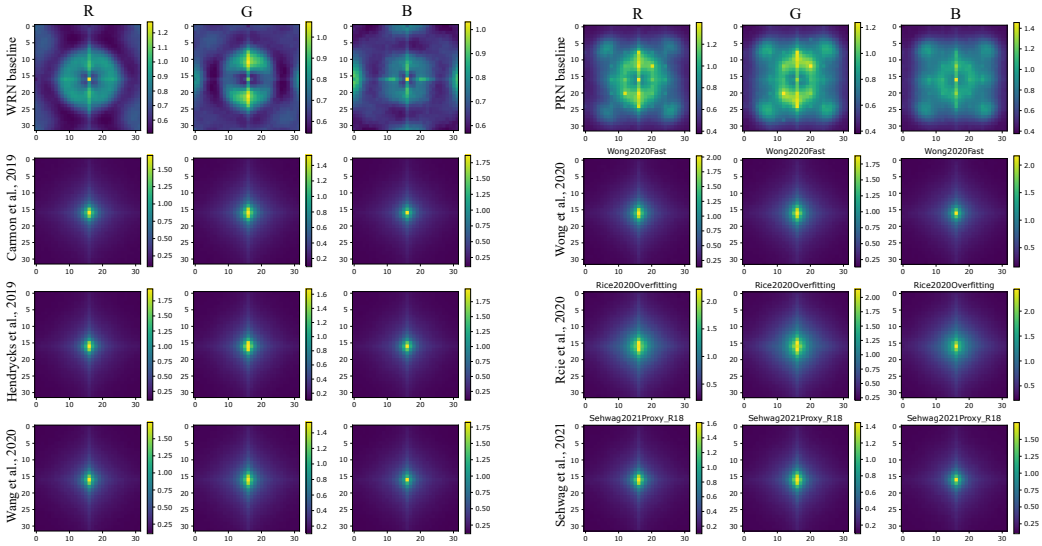
Pooling Variation. In addition to our observation of robust and non-robust networks on CIFAR-10, we conducted an ablation on MNIST to inspect the influence of different pooling methods. Therefore, we trained six small CNNs which all have the same architecture and only differ in the downsampling operation. Either we downsample by using Max- or AvgeragePooling, or we use convolution with stride two, as it was done for the models provided by RobustBench [Croce et al. (2021)]. We train three different seeds and report the mean and standard deviation over these runs. Each network was trained for 10 epochs with the Adam optimizer, a cycling learning rate and a maximal learning rate of $5e-3$. We employ a CrossEntropy loss and batch size is chosen to be 100. For the AT, we used FGSM adversaries with $\epsilon = 0.3$ and $\alpha = 3.75$. The training was executed on one NVIDIA Titan V100, each training run took around 2 minutes.

TABLE 5.1: **Performance Evaluation of different Pooling variants.** Mean clean and robust accuracy against AutoAttack (AA) [Croce & Hein (2020a)] with $\epsilon = 0.3$ for different pooling variations in the same network architecture on MNIST trained with three different seeds. As well as the amount of aliasing encountered in the downsampling layers.

Pooling	Training	Acc@1 ↑	AA ↑	Aliasing ↓
Convolution	Clean	99.30 ± 0.01	0.00 ± 0.00	8.91 ± 2.17
MaxPooling	Clean	99.44 ± 0.03	0.00 ± 0.00	29.07 ± 2.10
AveragePooling	Clean	99.21 ± 0.04	0.00 ± 0.00	18.50 ± 0.95
Convolution	FGSM	98.78 ± 0.33	54.97 ± 38.98	9.15 ± 5.42
MaxPooling	FGSM	98.20 ± 0.03	25.05 ± 10.21	11.49 ± 0.98
AveragePooling	FGSM	98.68 ± 0.21	55.86 ± 35.55	16.75 ± 2.11

Models using MaxPooling result in the most pronounced aliasing when trained without AT and demonstrate the poorest performance against adversarial attacks, regardless of AT. In contrast, downsampling via convolution and AveragePooling effectively suppress aliasing and increases robustness against attacks when adversarially trained. AveragePooling can be interpreted as blurring before downsampling, which is often applied in signal processing to suppress high-frequency components and thus reduces aliasing. Overall, the amount of aliasing is reduced due to FGSM AT and the learned convolutions with stride two exhibit the lowest aliasing compared to the static downsampling via Average- or MaxPooling. AveragePooling exhibits the highest aliasing when combined with AT. The high standard deviation in adversarial robustness suggested that some of the trained models experienced catastrophic overfitting.

Spectrum of Adversarial Perturbations. Further, we analyse the spectrum of the perturbations created by adversarial examples, *i.e.* perturbations created by AutoAttack [Croce & Hein (2020a)].



(A) WRN (B) PRN

FIGURE 5.7: **Center-shifted Spectrum of the AutoAttack Perturbations.** Center-shifted spectrum of the RGB perturbations created over CIFAR-10 samples by AutoAttack (Croce & Hein, 2020a) on the baseline model (top row) and three different robust models from RobustBench [Croce et al. (2021)]. While the robust models differ from the non-robust baseline, there is no significant difference among the robust models.

Firstly, we compare the spectrum of the perturbations created by the AutoAttack standard attack on our baseline model as well as on the robust models which we already evaluated in section 5.3.1. We compute the perturbations as differences between adversarial and clean images. Afterwards, we transform each perturbation into the Fourier space and take each of the three channels, RGB. The results are presented in Figure 5.7.

The frequency distribution of adversarial attacks, like AutoAttack, is variable, aligning with the findings of [Maiya et al. (2021)]. We observe variations between the non-robust baseline and the robust models. However, the spectral differences among the robust models are not significant.

5.3.2 CNN vs. FCN

We also conducted additional experiments on MNIST to investigate the difference between fully connected networks (FCNs) and CNNs. While CNNs work with the structural characteristics (neighbourhood) of the data, the FCN only takes the single pixels as stand-alone into account. Thus, we might not encounter aliasing issues in FCNs. Both FCN and CNN classifiers compress spatial resolution during the mapping from input data to class labels, not solely in the final layer. While this compression, inherent in mapping images to semantic labels, obscures intuitive understanding of spatial information in dense architectures, convolutional networks with controlled spatial compression (*i.e.* sampling) allow for systematic aliasing measurement. This is challenging in dense networks. To further understand the behavioural differences between convolutions and dense networks (FCNs), we performed an ablation study on MNIST.

We trained a FCN without convolutional layers with the same amount of layers, three, and approximately the same number of parameters, 40000, clean and with FGSM AT. The cleanly trained network can achieve a clean accuracy of 97.68% and no robust accuracy 0%. These results are similar to the CNN performance as reported in Table 5.1.

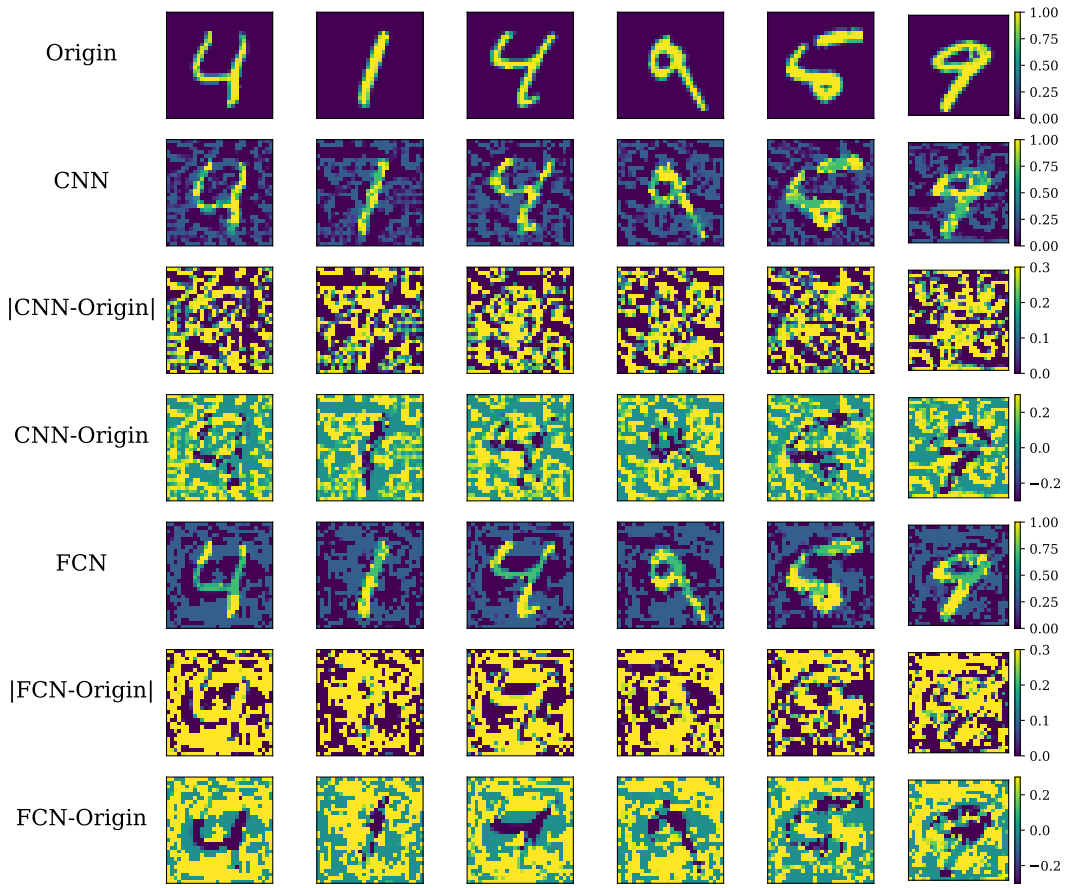


FIGURE 5.8: **Adversarial Sample Generated for a CNN vs. FCN.** Comparison of adversarial examples generated for a CNN and an FCN. The top row shows the original image without perturbations. The second, third, and fourth rows show the adversarial examples generated for the CNN, as well as the absolute and normalized differences between the original image and the adversaries, including the pure perturbations. The bottom three rows show the adversarial examples generated for the FCN, along with the absolute and normalized differences between the original image and the adversaries generated for the FCN.

Attack Structures. Further, we visualize the adversarial examples created on the CNN compared to the FCN presented in Figures 5.8 and 5.9. In Figure 5.8 we randomly picked six samples from MNIST to investigate the difference of the perturbations on FCN vs. CNN. While CNN perturbations exhibit fine-structured artifacts, FCN adversaries are characterized by block-like patterns. Figure 5.9 illustrates the mean perturbation across MNIST classes. For FCNs, these mean perturbations are concentrated on the target objects. In contrast, CNN mean perturbations are distributed throughout the image and appear more noisy.

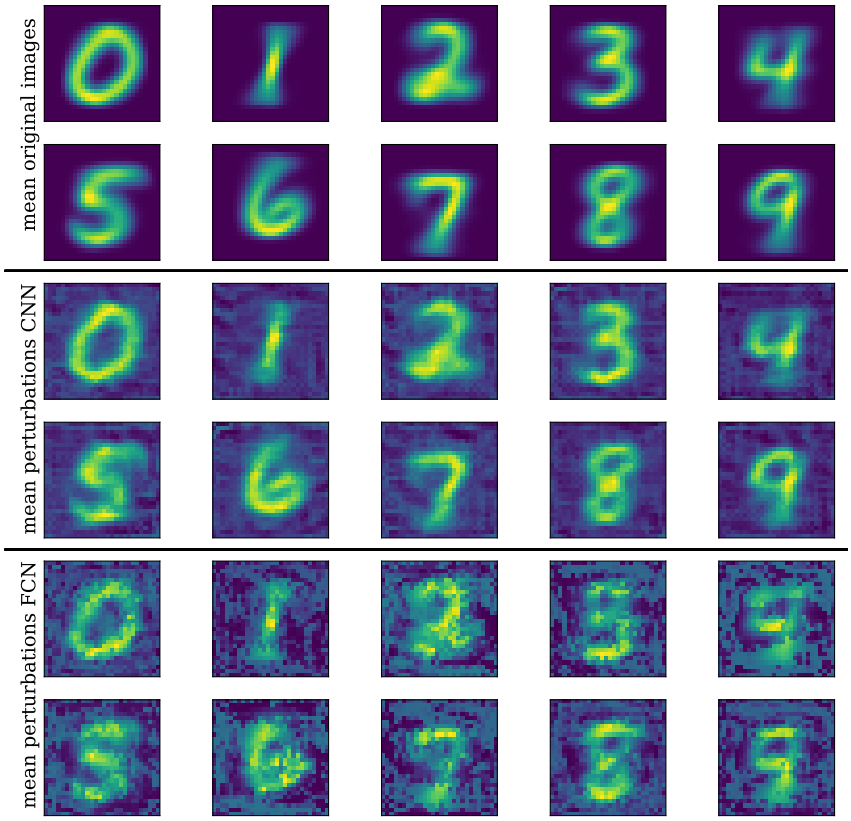


FIGURE 5.9: **Mean Adversarial Attack for a CNN vs. FCN.** Comparison of the mean image over each MNIST class original and perturbed with FGSM between FCN and CNN. The top two rows show the clean images, the middle two rows are the mean adversarial images on the CNN and the last two rows are the mean adversarial images on the FCN. While the MNIST numbers are still visible for the mean original images and the perturbations on the CNN, the perturbations on the FCN are much more related to the numbers that should be recognized in the MNIST task making the perturbations more visible to the human eye.

Intrinsic Robustness. Finally we measure the robustness of CNN and FCN with respect to varying epsilons. We compare the standard trained FCN vs. CNN. Figure 5.10 presents the results, revealing that CNNs exhibit greater robustness at lower epsilon values, while FCNs become more robust beyond an epsilon of 0.07. Both architectures are effectively compromised when epsilon exceeds 0.1. We acknowledge that network robustness is also influenced by factors such as model parameters and activation functions, which were held constant within our experiments.

5.3.3 Aliasing During Adversarial Training

Next, we consider the amount of aliasing during training of AT and conventional training. We adversarially trained five PRN-18 models with different training schemes and one PRN-18 without AT as baseline, using the same training parameters as described in Section 5.3.1. The AT schemes provided by Wong et al. (2020) and Rice et al. (2020) are used while we adapt the proposed method by Wong et al. (2020) to run with and without early stopping. During each training run, we computed the amount of aliasing in each downsampling and shortcut layer for each epoch from 100 randomly picked CIFAR-10 training samples.

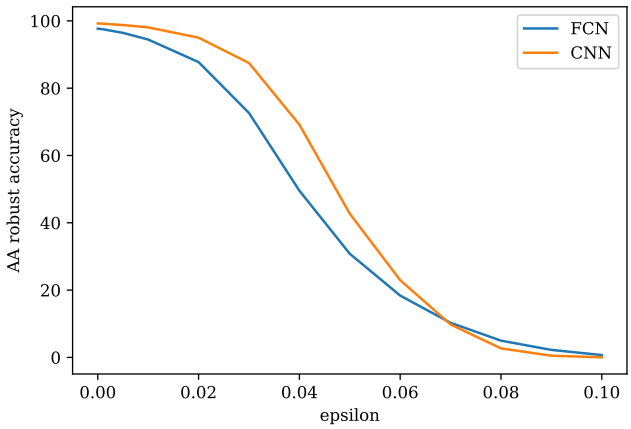


FIGURE 5.10: **Robustness Evaluation CNN vs. FCN.** Robust accuracy on AutoAttack [Croce & Hein (2020a)] for FCN and CNN across varying epsilon values. At small epsilon values, the CNN outperforms the FCN. However, as epsilon increases, both models fail, with the CNN being completely fooled at a lower epsilon threshold.

Figure 5.11 depicts the aliasing levels across all layers, with each plot representing the mean aliasing between downsampling and shortcut layers for various AT schemes and the baseline. We observe that adversarially trained networks consistently exhibit reduced aliasing in the second and third layers throughout training. In the fourth layer, aliasing is substantially lower for all methods, including the baseline. Table 5.2 presents the final robust accuracy for all adversarially trained models. All AT models demonstrate lower aliasing and higher robust accuracy compared to the baseline.

TABLE 5.2: **Performance Evaluation for different AT schemes.** Clean and robust accuracy against PGD (higher is better) and the amount of aliasing in the second layer (lower is better) for the baseline and adversarially trained networks, using the training scheme provided by Wong et al. (2020) and Rice et al. (2020), as well as FGSM with the training schedule from Wong et al. (2020), including early stopping criteria based on our aliasing measure.

Method	Acc@1 ↑	PGD ↑	Aliasing ↓
Baseline	**93.29**	0.00	12.12
FGSM (Wong et al., 2020)	90.85	7.05	9.31
early-stopping FGSM (Wong et al., 2020)	80.16	39.76	6.14
Free (Wong et al., 2020)	83.86	48.10	5.62
PGD (Wong et al., 2020)	85.06	**56.37**	6.30
Robust Overfitting (Rice et al., 2020)	84.58	46.70	3.99
Aliasing FGSM (ours)	82.91	52.43	5.78

Further, we can observe that early stopping for FGSM AT plays a crucial role for the robust accuracy as well as for the amount of aliasing. Table 5.2 shows that FGSM without early stopping performs nearly as poorly as training without any adversaries. Figure 5.11 indicates that the model trained with FGSM without early

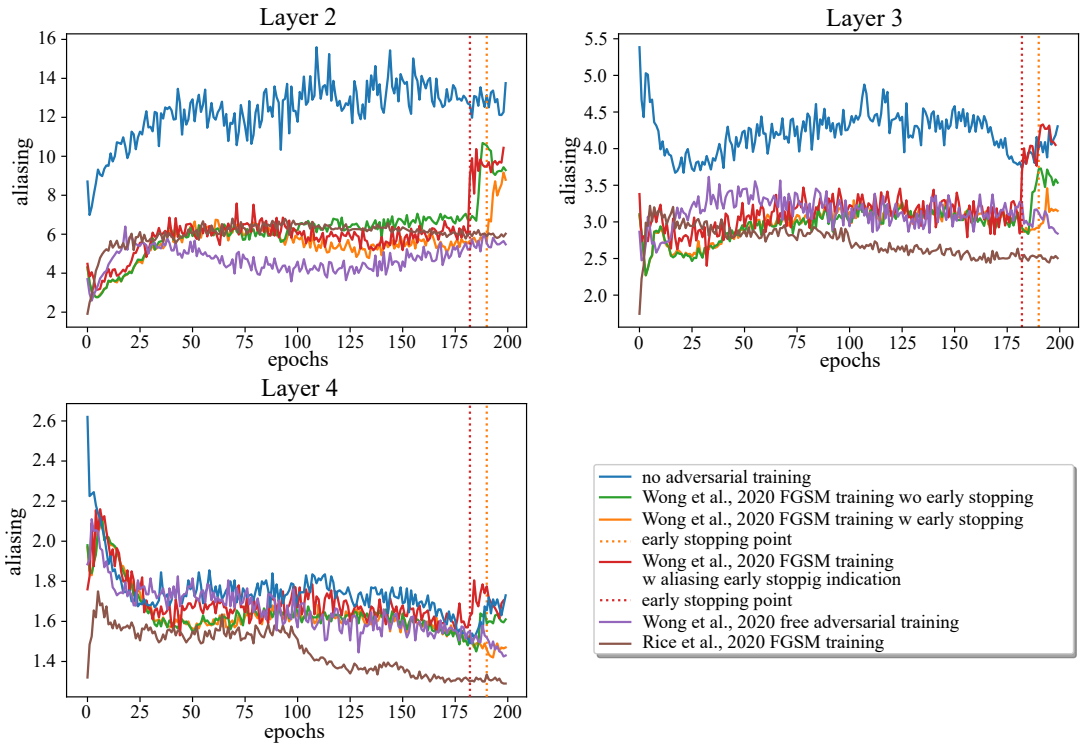


FIGURE 5.11: **Aliasing and Early Stopping in FGSM AT.** Amount of aliasing in all layers of PRN-18 during training over 100 random images of CIFAR-10 training set for the baseline and AT training provided by Wong et al. (2020) and Rice et al. (2020). The amount of aliasing is quantified as the mean value between the aliasing measure of the downsampling and the shortcut layer. All networks are trained for 200 epochs, except FGSM training including early stopping. There, the training is stopped earlier based on the evaluation of the model on PGD or our aliasing measure. Still, we record the epochs after early stopping and mark the point of early stopping by the dashed lines, to demonstrate the relationship between aliasing and early stopping.

stopping has a high increase in aliasing at the end of the training. The models which include early stopping stop before and thus exhibit no increased aliasing. The FGSM AT without early stopping continues and the amount of aliasing increases while the robust accuracy drops. This indicates that aliasing and adversarially robust generalization are highly related.

The decline in PGD robustness for FGSM AT without early stopping is known as catastrophic overfitting. To mitigate this, models are evaluated with PGD attacks after each epoch, comparing robust accuracy to the previous epoch. This approach, while less computationally intensive than full PGD training, yields comparable robustness. However, early stopping can lead to an underconverged network, which we discuss in Section 5.4.4 in greater detail.

5.3.4 Catastrophic Overfitting

Following, we explore the relation of catastrophic overfitting and aliasing further and show that the amount of aliasing increases when catastrophic overfitting takes place.

Aliasing vs. PGD accuracy. To highlight the relationship between aliasing and PGD attackability, we examined their correlation during training. Figures 5.12 and

5.13 depict aliasing levels and PGD accuracy across epochs for WRN-28-10 and PRN-18, respectively, demonstrating that catastrophic overfitting, marked by a sharp drop in PGD accuracy, coincides with a significant and sustained increase in aliasing.

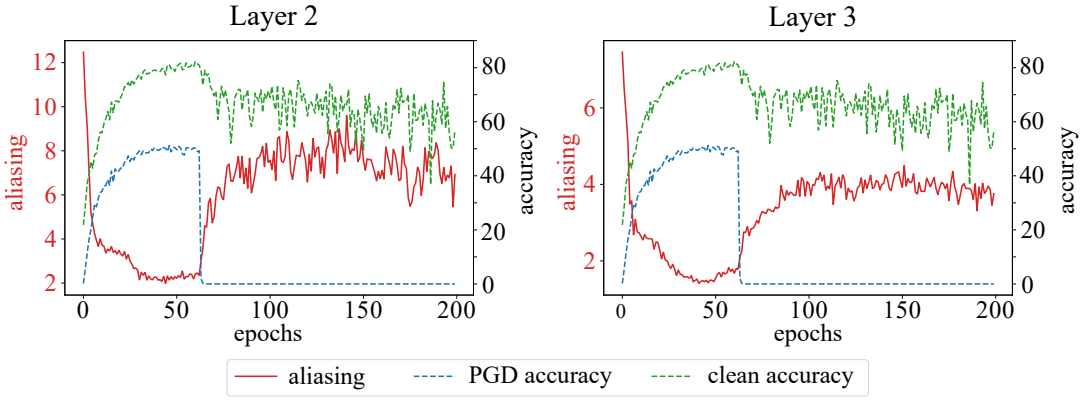


FIGURE 5.12: **Aliasing and Catastrophic Overfitting in FGSM AT on a WRN-28-10.** Aliasing, clean accuracy and PGD accuracy during training of a WRN-28-10 with FGSM AT and cycling learning rate. The model starts to exhibit robust overfitting in epoch 70, *i.e.* the PGD accuracy drops to zero and the amount of aliasing increases significantly.

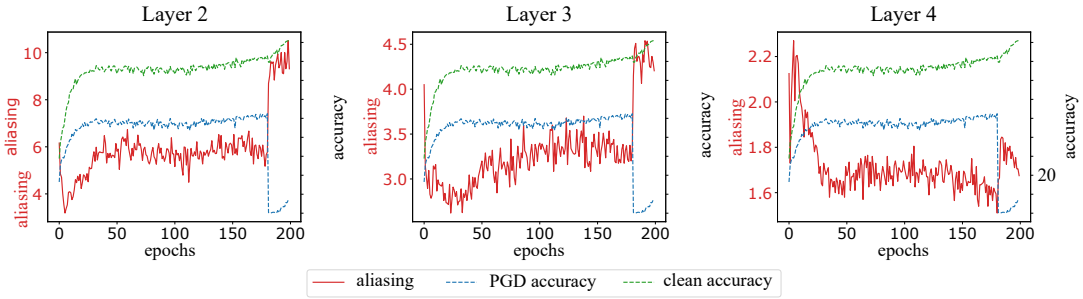


FIGURE 5.13: **Aliasing and Catastrophic Overfitting in FGSM AT on a PRN-18.** Aliasing, clean accuracy and PGD accuracy during training of a PRN-18 with FGSM AT and cycling learning rate. The model starts to exhibit catastrophic overfitting in epoch 180, *i.e.* the PGD accuracy drops to zero and the amount of aliasing increases significantly.

While the WRN-28-10 suffers from catastrophic overfitting already at epoch 60 the PRN-18 needs to be trained at least 180 epochs to exhibit catastrophic overfitting and an increased amount of aliasing. While the clean accuracy for the WRN-28-10 is highest right before the increase in aliasing, *i.e.* the catastrophic overfitting, and stays further below, the PRN-18 clean accuracy increases right after the increase in aliasing.

Spectrum of Adversarial Perturbations during AT. Next, we visualize the spectrum of attacks on our robust models before and after catastrophic overfitting. The results for the attacks are depicted in Figure 5.15. While the perturbations of the robust models before catastrophic overfitting exhibit spectral characteristics similar to the robust models from RobustBench [Croce et al. (2021)], the perturbations of the models after catastrophic overfitting shift more toward the higher frequency spectrum. In contrast, they do not fall into the middle-frequency spectrum, similar to the baseline model in Figure 5.7.

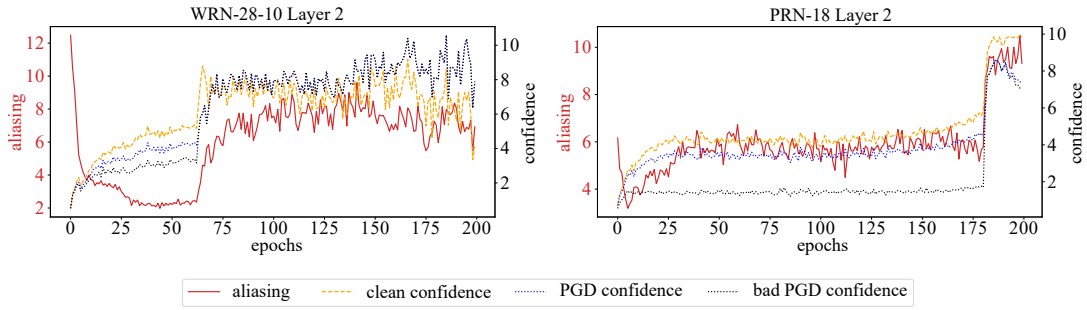


FIGURE 5.14: **Confidence and Aliasing during FGSM AT.** Aliasing of the second layer (red) and confidence of a WRN-28-10 (left) and PRN-18 (right) model trained with FGSM AT. In red the amount of aliasing, the dotted lines represent the confidences overall on the clean (yellow) and adversarial data (blue) as well as the confidence on the false predictions caused by the adversaries (black).

Effect on Network Confidences. To investigate further into the co-occurrence of aliasing and catastrophic overfitting, we take a look into the predicted confidence of the network as previously done in Chapter 4. Therefore, we calculate the overall confidence of the network predictions on the clean and PGD perturbed data as well as the confidence on the false predictions caused by PGD, which we call *bad pgd confidence*.

Figure 5.14 presents the networks' confidences on the clean and PGD data as well as the confidence on the wrong predictions caused by PGD and the amount of aliasing. We can observe for both WRN-28-10 and PRN-18, the model's confidence on the clean and adversarial test data increases significantly when aliasing increases, *i.e.* catastrophic overfitting takes place. Interestingly, the false confidence on PGD perturbations is relatively low before the increase in aliasing but gets highest after the increase. We assume that the network is not only not robust but much more confident with its false predictions after catastrophic overfitting, which is well aligned with our finding in Chapter 4 where we found that non-robust models are much more overconfident than robust models.

5.3.5 Aliasing Early Stopping

In Section 5.3.4, we showed that catastrophic overfitting during FGSM AT coincides with a sudden increase in aliasing in the models' feature maps. Now, we investigate whether we can exactly determine this overfitting point using our proposed aliasing measure. Catastrophic overfitting mainly occurs for FGSM AT consequently, we perform our early stopping criteria with aliasing.

Analogous to [Wong et al. (2020)], who set a threshold for the robust accuracy gap to PGD [Madry et al. (2018)], we define a threshold for the aliasing gap between successive epochs.

In this experiment, we only employ our aliasing measure computed from the feature maps in the second layer as we could observe the strongest peak of aliasing in the second layer. Further, when focusing only on one layer, we can additionally save computing time and be more efficient. One PRN-18 layer with downsampling includes two downsampling operations, so we build the mean between their aliasing measure as done before. However, the aliasing also depends on the images in each specific batch, *i.e.* aliasing is low for feature maps computed on very smooth input images while it is high for textured input data. To reduce noise, we apply a median

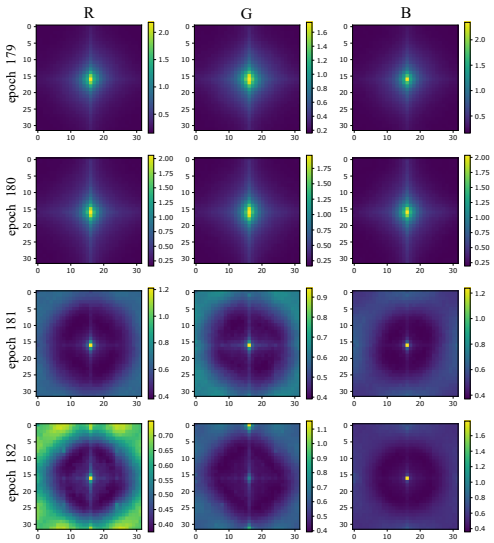


FIGURE 5.15: **Center-shifted Spectrum of AutoAttack Perturbations after Catastrophic Overfitting.** Center-shifted spectrum of the RGB perturbations created by AutoAttack at different epochs of a model training, before (two top rows, epoch 179 and 180) and after robust overfitting (bottom rows, epoch 181 and 182).

filter on the aliasing measure. This median filtered version is represented by the blue line in Figure 5.16.

On the median filtered aliasing curves, we simply compare each new aliasing measure to the median and predict a high loss in PGD accuracy when the aliasing measure increases. Figure 5.17 evaluates, for five different FGSM AT runs, the early stopping points computed by Algorithm 1 for varying thresholds t. We report the distance (in epochs) of our predicted stopping point to the best early stopping point predicted using PGD. Across all training runs, relative thresholds ranging from 0.3 to 0.35 consistently identified the PGD stopping point. Therefore, we decide for a threshold of 0.33, meaning training halts upon an increase in aliasing exceeding 33% (see Algorithm 1).

Our AM can determine the early stopping point in FGSM AT "on the fly" without explicit robustness tests. The computation of the early stopping point with our aliasing measure takes only around 903.44 milliseconds per epoch, while PGD takes around 1315.89 milliseconds per epoch on a NVIDIA Tesla V100. The results of FGSM AT are presented in Table 5.2. When compared with FGSM AT and FGSM AT with early stopping based on PGD provided by Wong et al. (2020) our aliasing early stopping is able to find the best trade-off between clean accuracy and robust accuracy while keeping the amount of aliasing low.

5.4 Discussion

Our experiments reveal that common CNNs fail to sub-sample their feature maps in a Nyquist-Shannon conform way and consequently introduce aliasing artifacts. Further, we can give strong evidence that aliasing and adversarial robustness are highly related. All evaluated robust models exhibit significantly less aliasing than standard trained models.

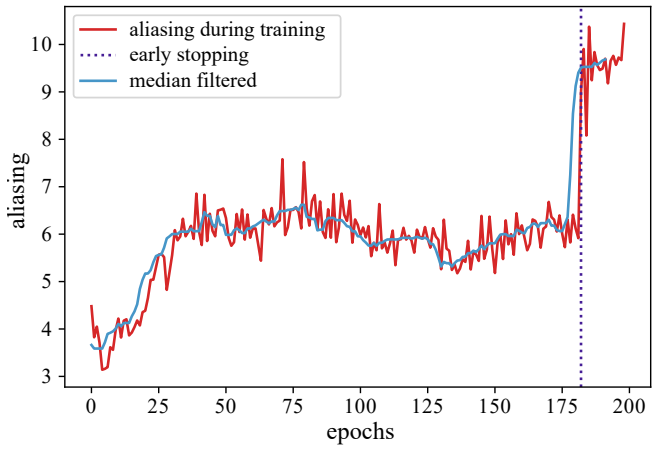


FIGURE 5.16: **Aliasing Measure Evaluation during FGSM AT.** Aliasing measure in the second layer during training of a PRN-18 with FGSM AT. In epoch 182, the PGD robustness as well as the proposed aliasing measure predict the best early stopping point.

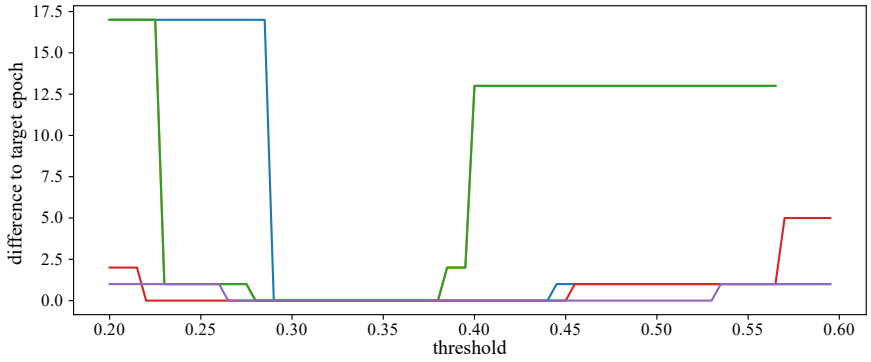


FIGURE 5.17: **Threshold Estimation for our Aliasing Measure.** Evaluation of the threshold for our aliasing measure over five independent training runs. Each line represents one aliasing one run. The difference of epochs is the L_1 difference between the early stopping epoch and the epoch calculated by the aliasing measure with the corresponding threshold.

We also gave an example use case for this finding, *i.e.* we showed that our aliasing measure could replace the explicit evaluation of network robustness as an early stopping criterion in FGSM. We discuss both aspects in the following.

5.4.1 Spectrum of Adversarial Perturbations

We could show that adversarial perturbations of robust models dominantly lie in the low-frequency spectrum, while the perturbations of non-robust models can lie in the low as well as in the middle or high frequencies. For models that exhibit robust overfitting, the generated perturbations dominantly lie in the high-frequency spectrum. These findings align with [Maiya et al. (2021)], suggesting that the frequency spectrum in which perturbations can occur depends not only on the dataset but also on the specific model architecture and training scheme. Furthermore, we assume that aliasing is one of the key vulnerability exploited by adversarial attacks.

Algorithm 1 Aliasing Early Stopping

Require: bat
Ensure: the model weights m_{best}, stopping point p_{stop}
$t \leftarrow 0.33$
$AM_{hist} \leftarrow []$, history of the aliasing measure
$AM_d, AM_s \leftarrow$ ALIASING MEASURE$(2, m, X)$ ▷ Aliasing Measure quantifies the aliasing for the downsampling layer AM_d and the shortcut AM_s
$AM \leftarrow (AM_d + AM_s)/2$
$AM_{hist} \leftarrow AM_{hist}$.append$(AM)$
for e in N **do**
  do network training
  $AM_d, AM_s \leftarrow$ ALIASING MEASURE$(2, m, X)$
  $AM \leftarrow (AM_d + AM_s)/2$
  $AM_{hist} \leftarrow AM_{hist}$.append$(AM)$
  if $AM > MEDIAN(AM_{hist}) * (1 + t)$ **then**
    $p_{stop} \leftarrow e$
    return m_{best}, p_{stop}
  else
    AM_{hist}.append(AM)
    $m_{best} = m$
  end if
end for

5.4.2 Aliasing in Pre-Trained Models

After the application of downsampling operations in standard CNNs all feature maps suffer from aliasing artifacts occurring due to insufficient sub-sampling.

As already observed in Section 5.3.1, low aliasing is especially important in the earlier layers. This can likely be explained by the fact that information is spatially more and more compressed as it is propagated to deeper layers. Therefore, deep layers require sparsity in the feature maps to be expressive. Thus, we hypothesize that deeper layers are less vulnerable to aliasing, whereas earlier layers are more susceptible. Therefore, the difference in aliasing between robust and non-robust models is most pronounced in the early layers.

Summarizing, adversarially trained networks exhibit significantly less aliasing in their feature maps than standard trained networks with the same architecture. As shown in Section 5.3.1, this is valid for different model architectures and training schemes. It raises the question of whether models with a low amount of aliasing are necessarily more robust. We show in the following, in Chapter 6, that aliasing-free downsampling indeed improves native robustness and enhances AT.

5.4.3 Aliasing and Catastrophic Overfitting

We provide strong evidence that catastrophic overfitting in FGSM AT is negatively correlated with the amount of aliasing after downsampling. Whenever a model experiences catastrophic overfitting during FGSM AT, the amount of aliasing increases significantly, *i.e.* the increase in aliasing marks the point at which the model loses its robust generalization ability.

Aliasing as Early Stopping Indication. We could show that our aliasing measure can be an indication for the early stopping point, which is needed to prevent catastrophic overfitting on single-step AT schemes like FGSM. Thereby, we choose the relative increase in aliasing, t, at which to stop to be 33%. This threshold is chosen by comparing different trained networks and their aliasing measure during network training. With this setting, we aim for transferability of the approach to different network architectures and datasets.

Yet, the same issue exists for previous approaches where some threshold had to be determined [Wong et al. (2020)]. In contrast to the explicit robustness evaluation in [Wong et al. (2020)], the aliasing measure indicator does not depend on specific, externally computed perturbations but can be evaluated during each training iteration on the training batch.

5.4.4 Limitations

However, early stopping inevitably interrupts the training process, resulting in under-converged networks. While our early stopping criterion provides a straightforward and effective method to halt training before catastrophic overfitting occurs, it may also lead to networks that are not fully converged, potentially reducing their performance.

Furthermore, our aliasing measure essentially applies a rectangular function to the frequency representation, which may introduce additional spectral artifacts, such as sinc-interpolation artifacts, which are discussed in greater detail in Section 6.2.3. Consequently, the aliasing-free ground truth may inherently favour sinc-like structures. Additionally, the feature map that is downsampled using a stride of two and subsequently transformed into the frequency domain may suffer from spectral leakage artifacts. This happens due to the absence of padding, which leads to an insufficiently refined frequency representation, causing energy to disperse across multiple frequency bins.

5.5 Conclusion

In conclusion, we demonstrate a strong correlation between aliasing and adversarial robustness in CNNs. Specifically, increased aliasing coincides with decreased PGD robustness during catastrophic overfitting in FGSM AT. Further, we are able to tackle the problem of catastrophic overfitting via early stopping based on our aliasing measure. We hypothesize that aliasing is one of the main underlying factors that lead to the vulnerability of CNNs. Recent methods to increase model robustness rather heal the symptoms of the underlying problem than investigate its origins. To overcome this challenge we might need to start thinking about CNNs in a more signal processing manner and account for basic principles from this field, like the Nyquist-Shannon theorem, which gives us clear instructions on how to prevent aliasing. In the following, Chapter 6, we show how to inherently implement this principle in current downsampling methods. Thereby enhancing the models' native robustness and AT. Besides downsampling, padding can also lead to unwanted aliasing effects. Still, it is not straightforward to incorporate this knowledge into the architecture and structure of common CNN designs as we have many components to account for. We aim to offer a novel yet fundamentally sound perspective on CNNs, enhancing their performance and reliability for real-world applications.

Part II

Novel Fourier Modules

Chapter 6

Aliasing-Free Downsampling in the Frequency Domain

Contents

In this chapter, we introduce aliasing-free downsampling. Since from an image and signal processing point of view, the huge success of CNNs is counter-intuitive, as the inherent spatial pyramid design of most CNNs is apparently violating basic signal processing laws, *i.e.* the *Sampling Theorem* in their downsampling operations, leading to aliasing. This issue was broadly neglected until model robustness started to receive more attention. We showed previously in Chapter 5 that there is a strong correlation between the vulnerability of CNNs and aliasing artifacts induced by bandlimit-violating downsampling. Thus, we propose to downsample in the frequency domain to ensure aliasing-free downsampling, denoted by *Frequency Low Cut Pooling (FLC Pooling)*, which we further extend to *Aliasing and Sinc Artifact-free Pooling (ASAP)*. ASAP is aliasing-free and removes further artifacts from sinc-interpolation. Our experimental evaluation on ImageNet-1k, ImageNet-C and CIFAR datasets on various CNN architectures shows that networks using FLC Pooling and ASAP as downsampling methods achieve higher robustness against common corruptions and adversarial attacks natively, *i.e.*, without explicitly training for robustness, while maintaining a clean accuracy similar to the respective baseline

models. Further, FLC Pooling and ASAP effectively reduced the risk of catastrophic overfitting in FGSM AT, leading to improved performance.

This chapter is based on Grabinski et al. (2022b, 2023) . Julia Grabinski, as the first author, conducted all experiments besides the AT of the networks on ImageNet-1k, Steffen Jung conducted those. Julia Grabinski was the main writer.

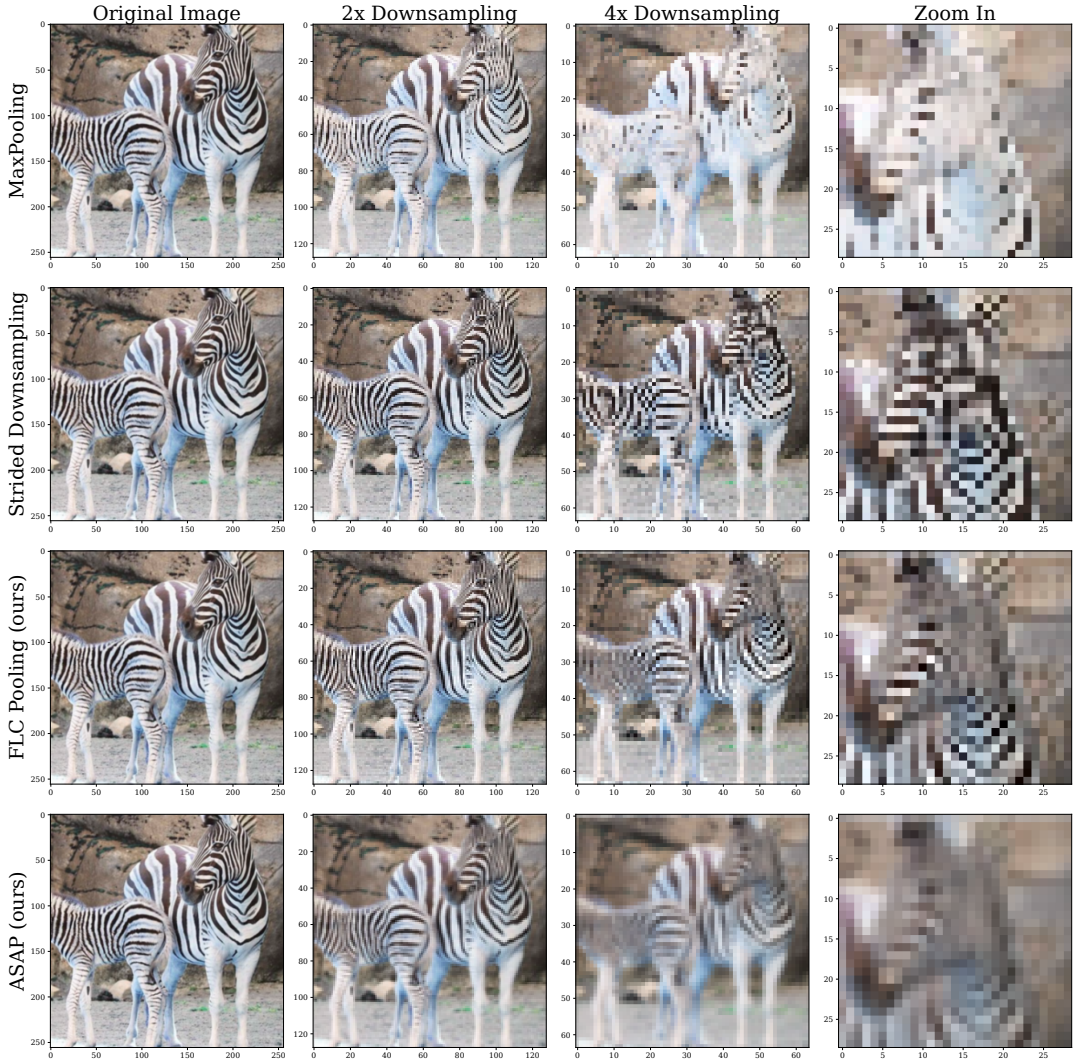


FIGURE 6.1: **Impact of Downsampling Methods: Visual Quality of different Downsampling Techniques.** A qualitative comparison of different downsampling techniques. The first and second rows show the commonly used MaxPooling and strided downsampling. In the third and fourth rows, we apply our FLC Pooling and ASAP, respectively. While MaxPooling does not preserve the image structure well, FLC Pooling and ASAP much better retain structural and spatial information. Strided downsampling also preserves the zebra's structure, yet suffers from severe aliasing artifacts, visible as grid-like patterns on the zebra's fur. These artifacts are removed with our FLC Pooling and ASAP. While, FLC Pooling exposes sinc artifacts, visible for example around the zebras head after the first two downsampling stages, such artifacts are removed in ASAP.

6.1 Introduction

Most CNN architectures use a combination of small convolutional kernels and downsampling to increase the network's receptive field while keeping computational costs low. However, standard downsampling methods such as MaxPooling, AveragePooling, or convolution with a stride of two suffer from a significant drawback: their susceptibility to aliasing [Zhang (2019); Zou et al. (2023)]. We demonstrated in Chapter 5 that aliasing correlates with the network's vulnerability to distribution shifts [Zhang (2019); Zou et al. (2023)] and adversarial attacks [Li et al. (2021); Hossain et al. (2023)]. Based on these results, we investigate the manner in which 2D signals, in case of CNNs' input images and feature maps, are downsampled and how this results in undesired artifacts. So far, prior research mainly focused on reducing aliasing artifacts [Hossain et al. (2023); Zhang (2019); Zou et al. (2023)], proposing the use of blur kernels for mitigation [Zhang (2019); Zou et al. (2023)]. However, these approaches are neither capable of completely removing aliasing nor addressing other types of spectral leakage artifacts associated with downsampling in CNNs.

We propose downsampling in the frequency domain to achieve aliasing-free downsampling, via **F**requency **L**ow **C**ut (FLC) Pooling. Networks using FLC Pooling achieve higher robustness against common corruptions [Hendrycks & Dietterich (2019)], adversarial attacks [Goodfellow et al. (2015); Madry et al. (2018)], and enhance AT [Goodfellow et al. (2015)] by mitigating catastrophic overfitting. Building on these insights, we analyse the properties of CNN feature maps after frequency-domain downsampling and identify limitations in FLC Pooling. Specifically, while FLC Pooling eliminates aliasing, it remains susceptible to other spectral distortions, such as sinc-interpolation artifacts, which manifest as rippling patterns in the spatial domain. For instance, in Figure 6.1, the third row and third column show repeated structures near the zebra's head caused by these artifacts. To address this, we propose an improved method: **A**liasing and **S**inc **A**rtifact-free **P**ooling (ASAP). By extending FLC Pooling with Hamming windowing in the frequency domain, ASAP further reduces spectral artifacts, enhancing both inherent robustness and robustness after AT. In contrast to our downsampling, MaxPooling distorts the zebra's image (Figure 6.1, first row), and strided downsampling generates substantial aliasing (Figure 6.1, second row). These results highlight the importance of revisiting and improving CNN downsampling methods.

Our contributions are as follows:

- We investigate current downsampling methods, such as MaxPooling as well as strided convolutions, and demonstrate why they should be replaced with downsampling methods that conform to signal processing principles like the Nyquist Shannon Theorem.
- We introduce **F**requency **L**ow **C**ut Pooling, short FLC Pooling, for guaranteed aliasing-free downsampling without additional hyperparameters.
- We show that even aliasing-free downsampling can be susceptible to further corruptions in the frequency domain, specifically sinc-interpolation artifacts.
- Consequently, we propose **A**liasing and **S**inc **A**rtifacts-free **P**ooling, short ASAP. Thus, we achieve higher robustness with and without AT compared to state-of-the-art models incorporating standard downsampling methods. Like FLC Pooling, ASAP is hyperparameter-free.

- To validate the robustness of FLC Pooling and ASAP, we empirically evaluate against adversarial attacks [Goodfellow et al. (2015); Madry et al. (2018); Croce & Hein (2021)], common corruptions included in ImageNet-C [Hendrycks & Dietterich (2019)] and spatial shifts [Zhang (2019)].
- Furthermore, we combine FLC Pooling and ASAP with FGSM [Goodfellow et al. (2015)] and PGD [Madry et al. (2018)] AT. Our results demonstrate that the resulting models achieve favourable performance in both clean and robust accuracy while effectively avoiding catastrophic overfitting during FGSM AT.

6.2 Method

First, we elaborate shortly on aliasing in CNN's downsampling. Afterwards, we present our aliasing-free downsampling method, FLC Pooling, which completely removes aliasing artifacts during downsampling. Additionally, we further extend FLC Pooling by introducing Aliasing and Sinc Artifact-free Pooling, short ASAP, which addresses not only the well-known issue of aliasing but also tackles sinc-interpolation artifacts.

6.2.1 Aliasing in CNNs Downsampling

Common CNNs sub-sample their intermediate feature maps to aggregate spatial information and increase the network's invariance. However, current sub-sampling methods do not incorporate aliasing prevention. Specifically, sub-sampling with insufficient sampling rates causes pathological overlaps in the frequency spectra (Figure 2.4). These overlaps occur when the sampling rate falls below twice the signal's bandwidth [Shannon (1949)], leading to ambiguities: high-frequency components cannot be clearly distinguished from low-frequency components. As a result, CNNs might misconceive local uncorrelated image perturbations as global manipulations. In Chapter 5, we show that aliasing in CNNs strongly coincides with the robustness of the model. From this observation, we hypothesize that models over-relying on high-frequency components are inherently less robust. Empirical evidence in Figure 6.9 validates this hypothesis within adversarial training (AT), showing that catastrophic overfitting is accompanied by increased aliasing during FGSM AT. Based on this observation, we expect that networks employing aliasing-free sampling methods, such as FLC Pooling and ASAP, exhibit more stable behaviour in FGSM AT settings.

6.2.2 FrequencyLowCut Pooling

Previous approaches have introduced downsampling methods that reduce aliasing by applying blurring techniques before downsampling [Zhang (2019); Zou et al. (2023); Hossain et al. (2023)]. They do so by classical blurring operators in the spatial domain. While those methods reduce aliasing, they can not entirely remove it due to sampling theoretic considerations in theory and limited filter sizes in practice (see [Gonzalez & Woods (2006)] for details).

In contrast, we perfectly remove aliases in CNNs' downsampling operations without adding additional hyperparameters. Therefore, we directly address the downsampling operation in the frequency domain, where we can sample according to the Nyquist rate, *i.e.* remove all frequencies above $\frac{\text{samplingrate}}{2}$ and thus discard

any potential aliases. Similar to our proposed aliasing measure in Chapter 5, we exclusively select the low-frequency components below the Nyquist frequency. However, we employ a simple yet effective method, fftshift, to facilitate the extraction of low-frequency components. Our proposed aliasing-free downsampling operation is illustrated in Figure 6.2.

We first perform a Discrete Fourier Transform (DFT) on the feature maps g. Feature maps with height M and width N that are to be downsampled are represented as follows:

$$G(k,l) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m,n) e^{-2\pi j \left(\frac{k}{M}m + \frac{l}{N}n\right)}. \tag{6.1}$$

In the resulting frequency space representation $G(k,l)$, all coefficients of frequencies k,l, with $|k|$ or $|l| > \frac{\text{samplingrate}}{2}$ have to be set to 0 before downsampling. CNNs commonly downsample with a factor of two, such that the resulting sampling rate is $\frac{1}{2}$. Aliasing-free downsampling thus corresponds to removing coefficients where $|k|, |l| > \frac{1}{4}$. The remaining coefficients denoted by G_d are then transformed back to the spatial domain via inverse DFT:

$$g_d(\hat{m},\hat{n}) = \frac{1}{\hat{K}\hat{L}} \sum_{k=0}^{\hat{K}-1} \sum_{l=0}^{\hat{L}-1} G_d(k,l) e^{2\pi j \left(\frac{\hat{m}}{\hat{K}}k + \frac{\hat{n}}{\hat{L}}l\right)}. \tag{6.2}$$

Implementation. Practically, the DFT(g) returns an array G of complex numbers with size $K \times L = M \times N$, where the frequency $k,l = 0$ is stored in the upper left corner and the highest frequency is in the center. We thus shift the low-frequency components into the center of the array via FFT-shift to get G_s and crop the frequencies below the Nyquist frequency, u_{max}, as $\mathrm{G}_{sd} = \mathrm{G}_s[K' : 3K', L' : 3L']$ for $K' = \frac{K}{4}$ and $L' = \frac{L}{4}$, for all samples in a batch and all channels in the feature map. After the inverse FFT-shift, we obtain array G_d with size $[\hat{K},\hat{L}] = [\frac{K}{2}, \frac{L}{2}]$, containing exactly all frequencies below the Nyquist frequency u_{max}, which we can transform back to the spatial domain via inverse DFT for the spatial indices $\hat{m} = 0 \dots \frac{M}{2}$ and $\hat{n} = 0 \dots \frac{N}{2}$.

We thus receive the aliasing-free downsampled feature map g_d with size $[\frac{M}{2}, \frac{N}{2}]$. Due to the process of cutting out the low-frequency components, we call this approach **F**requency **L**ow **C**ut Pooling, short FLC Pooling.

Figure 6.2 presents this procedure in detail. Particularly, FLC Pooling begins with an FFT to obtain the spectral representation. The next step is the low-pass cut, which corresponds to the third step in Figure 6.2, followed by the IFFT. In the spatial domain, the low pass cut operation would amount to convolving the feature map with an infinitely large (non-bandlimited) $\mathrm{sinc}(m) = \frac{\sin(m)}{m}$ filter, which can not be implemented in practice. In the following, we discuss the practical implications of this in detail and propose an extension of our FLC Pooling in Section 6.2.3.

Why Do We Need FLC Pooling? In the following, we briefly discuss why spatial blurring [Zhang (2019); Zou et al. (2023); Hossain et al. (2023)] can not guarantee the prevention of aliasing in the feature maps, even if large convolutional kernels would be applied, and why, in contrast, the proposed FLC Pooling can guarantee to prevent aliasing. Zhang (2019); Zou et al. (2023) proposed to apply approximated Gaussian filter kernels to the feature map. This operation is motivated by the fact that an actual Gaussian in the spatial domain corresponds to a Gaussian in the frequency domain. As the standard deviation of the Gaussian in the spatial domain increases, the standard deviation of its frequency representation decreases. Yet,

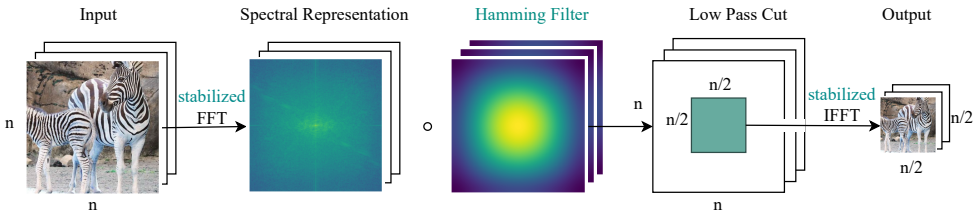


FIGURE 6.2: **FLC Pooling and ASAP, Method Overview.** Overview of our aliasing-free FLC Pooling and it's extension to aliasing- and sinc-artifact-free downsampling, short ASAP. First, we transform the input into the frequency domain and center-shift using our stabilized FFT and shift the low-frequency components to the center. For ASAP we afterwards apply a Hamming filter to the frequency representation to prevent sinc interpolation artifacts, this step is skipped for FLC Pooling. Next, we apply the low pass cut to downsample aliasing-free. Finally, we transform the feature map back into the spatial domain using our stabilized IFFT.

the Gaussian distribution has infinite support, regardless of its standard deviation, *i.e.* the function never actually drops to zero. The convolution in the spatial domain corresponds to the point-wise multiplication in the frequency domain. Therefore, even after convolving a signal with a perfect Gaussian filter with a large standard deviation (and infinite support), all frequency components that were $\neq 0$ prior to the convolution remain non-zero, albeit reduced in magnitude. Specifically, convolution with a Gaussian filter, even under theoretically ideal conditions, can mitigate apparent aliasing but cannot entirely eliminate it. In practice, these ideal settings are not given: Prior works such as [Zhang (2019); Zou et al. (2023)] have to employ approximated Gaussian filters with finite support (usually not larger than 7×7). In the spatial domain, the ideal low pass filter operation employed in our FLC Pooling corresponds to a convolution of the feature maps with the Fourier transform of the rectangular function $rect(u, \tau)$ (by the Convolution Theorem, *e.g.* [Gonzalez & Woods (2006)]) defined here between frequencies $-1/\tau$ and $1/\tau$.

$$rect(u, \tau) = \prod \left(\frac{u}{2/\tau} \right) = \begin{cases} 1 & \text{for } |u| \leq 1/\tau \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

with $1/\tau$ defined at Nyquist frequency u_{max} or practically speaking for our implementation in FLC Pooling at M/2 and N/2.

This trivially guarantees all frequencies above below M/2 and N/2 to be zero. The Fourier transform of this rectangle function is given by (details to the derivation are given in Section 2.4.6):

$$f(x, \tau) = F^{-1} \left[\prod \left(\frac{u}{2/\tau} \right) \right] = \frac{2}{\tau} sinc \left(\frac{2\pi x}{\tau} \right), \tag{6.4}$$

However, while the ideal low pass filter in the Fourier domain has finite support $\prod \left(\frac{u}{2/\tau} \right)$, its equivalent in the spatial domain has infinite support, *i.e.* $\frac{2}{\tau} sinc \left(\frac{2\pi x}{\tau} \right)$ will never approach zero. Hence, we need an infinitely large convolution kernel to apply perfect low-pass filtering in the spatial domain. This is obviously not possible in practice and prior work did use Gaussian kernels not larger than 7×7.

Sinc Interpolation Artifacts. Following this line of argumentation, we now discuss the occurrence of sinc interpolation artifacts from the application of a rectangular impulse in the frequency domain more in depth. In Figure 6.6 we visualize aliasing in the spatial domain. The leftmost image represents the original image before downsampling, the other images depict the downsampled versions (here by a factor of eight for better visibility). The middle image (strided downsampling), downscaled without any anti-aliasing, exhibits prominent grid artifacts, which are particularly noticeable in the zebra's fur. In contrast, the image downsampled using FLC Pooling shows an aliasing-free downscaled version (the fourth image from the left in Figure 6.6).

However, upon closer examination of the quality and stability of the FLC pooled image, we observe that it is still susceptible to sinc-interpolation artifacts, or *ringing artifacts*, predominantly visible around the zebra's head in Figures 6.1 and 6.6. In the following, we briefly discuss these artifacts and present a simple approach to mitigate them.

As previously discussed, we apply a rectangle function as point-wise multiplication in the Fourier domain, which is equivalent to circular convolution, *i.e.* a convolution with circular boundary conditions, with a sinc-function in the spatial domain. This leads to (i) sinc-interpolation artifacts, also referred to as *ringing artifacts* in the spatial domain, and (ii) to wrap-around effects of the circular sinc-convolution at the boundaries of the feature map, where, for example, signal from the left boundary of a feature map is convolved into the signal on the right.

6.2.3 Sinc Interpolation Artifact-Free Pooling

To achieve completely aliasing-free downsampling, it is necessary to apply the rectangle function in the frequency domain. To mitigate sinc artifacts (*i.e.* ringing artifacts), it is common to apply a filter that smoothens the sharp edges of the rectangle function. Therefore, we propose the use of a Hamming window, $H(n)$, which is defined for 1D signals as follows:

$$H(n) = a - (1 - a) \cdot cos\left(\frac{2\pi n}{N}\right), \quad 0 \leq n \leq N \tag{6.5}$$

The 2D Hamming filter is defined as the outer product of two 1D Hamming filters, where $a = 25/46$ and N represents the number of samples in the signal. However, unlike in [Tomen & van Gemert (2021)], we do not utilize the Hamming filter as a window function in the spatial domain. Instead, we directly apply the Hamming filter in the frequency domain as a point-wise multiplication.

The spatial representation of the Hamming window from the frequency domain is shown in Figure 2.5 (right, blue line) in Chapter 2. The side slopes of the transformed Hamming window become near zero, effectively reducing interpolation artifacts. Further, we show in Figure 2.5 the combination of the rectangle function and the Hamming window (red dashed line) in the Fourier domain (left) and the spatial domain (right). The Hamming window suppresses the oscillations of the sinc function, while remaining completely aliasing-free. Further, the possible boundary artifacts from the circular convolution with the sinc function are also suppressed as the side slope of the signal becomes near zero.

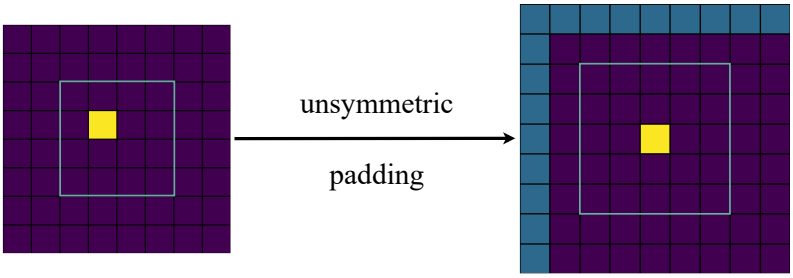


FIGURE 6.3: **Effect of Unsymmetrical Padding for Low-Frequency Component Cuts.** We use unsymmetrical padding to achieve an uneven number of input samples. Thus, the frequency representation is symmetric and the DC component is centered. During the cut of the low-frequency components also the cut remains symmetric while this is not the case for the unpadded signal.

Stabilized Fast Fourier Transform. To further enhance our approach, we found that it is necessary to stabilize the Fourier Transform. Common FFT implementations [Cooley & Tukey (1965)] leverage the separability of the Fourier Transform. Typically, the n-dimensional FT are cascaded 1D Fourier transforms. Hence, the 2D FFT is computed first in one direction (vertically) and then in the other direction (horizontally). Due to numerical instabilities, this process can introduce shifts in the image after multiple transform applications (see Figure 6.7, second and third column).

We suppose that the numerical issues of the usual row-first FFT originate from the fact that our signal is usually even-sized (even number of pixels in width and height), leading to non-centered frequency representations (see Figure 6.3, left). The resulting asymmetric cut-out of the low-frequency components can be avoided by padding. Specifically, asymmetric padding can help to center the frequency representation as demonstrated in Figure 6.3, right.

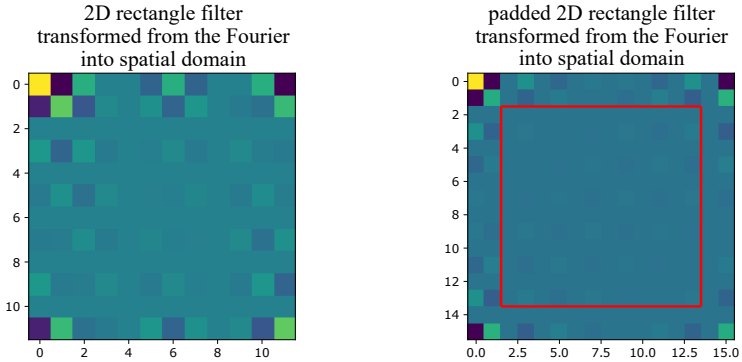


FIGURE 6.4: **Effect of Padding to Prevent Sinc Interpolation Artifacts.** Left: Rectangle filter transformed from the Fourier domain to the spatial domain without padding. At the edges we can see severe sinc-interpolation artifacts. Right: Padded rectangle filter transformed from the Fourier domain to the spatial domain. After the transformation the padding is removed by a centered crop (red line). Thus, some of the dominant sinc-interpolation artifacts are removed.

In theory, padding can also serve as a mitigating factor against sinc interpolation artifacts. The rationale behind this is twofold. First, the padding compensates for possible ringing artifacts that fold in, as the padding is removed after the operation, as shown in Figure 6.4. Second, through the padding and transformation into the frequency domain, we artificially increased the resolution of the signal. Thus,

when cutting at the Nyquist frequency of the higher resolution signal, we artificially increase the value for $1/\tau$, resulting in a more narrow sinc in the spatial domain. In our analysis, we provide two options for padding. Small padding, denoted by ASAP_{sp}, only adds one line of zeros to the bottom and left. Large padding, denoted by ASAP_{lp}, pads by $\frac{\text{inputsize}}{2} - 1$ to the top and left. Both lead to centered representations (see Figure 6.7). Larger padding could provide better results, while smaller padding is more efficient. Yet, since the FFT algorithm is optimized for data in the array size of powers of two, even small padding can increase the compute costs when the size before padding is exactly a power of two.

Therefore, we evaluate a very cost-efficient heuristic to avoid the numerical artifacts introduced by row-first 2D FFTs: we transpose the feature map before every other FFT, such that row-first and column-first FFT are applied in an alternating manner. We denote this stabilization method as $\text{ASAP}_{\text{stbl}}$. We discuss the computation cost in Section 6.3.4.

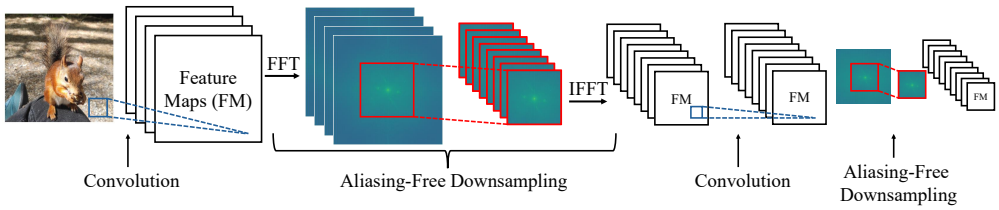


FIGURE 6.5: **FLC Pooling and ASAP, integration in CNN.** Abstract integration of our aliasing-free downsampling for feature extraction in a CNN, similar to Figure 1.2 (b) from Chapter 1.

6.2.4 Integration into CNNs

The integration of our FLC Pooling as ASAP and straight-forward, whenever the networks includes an operation with stride two, it is replaced with our downsampling. If the original downsampling operation was a convolution, we add a convolution after our downsampling approach to account for the flexibility of the operation. When the original downsampling operation was Max- or AveragePooling, it is simply replaced. Therefore, the abstract workflow of a CNN incorporating our downsampling is illustrated in both Figure 6.5 and Figure 1.2 (b).

6.3 Experiments

For evaluation, we first visualize aliasing artifacts following standard downsampling, as well as sinc-interpolation artifacts resulting from FLC Pooling. Second, we compare the performance of state-of-the-art models trained using different downsampling techniques against both our FLC Pooling and ASAP. This evaluation shows that FLC Pooling and ASAP exhibit superior native robustness against spatial shifts, common corruptions and adversarial attacks. Third, we evaluate our FLC Pooling and ASAP networks in combination with AT and demonstrate their ability to prevent catastrophic overfitting during FGSM AT, resulting in higher robust accuracy while maintaining high clean accuracy. Last, we conduct an ablation study.

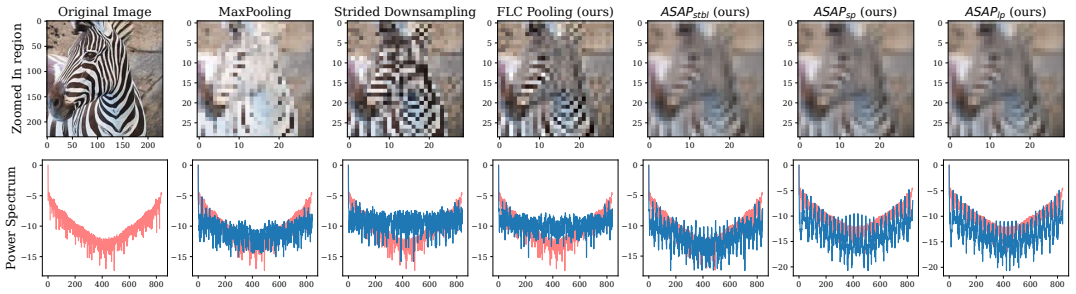


FIGURE 6.6: **Image Quality and Power Spectrum Differences for different Downsampling Techniques.** Qualitative comparison of different downsampling methods regarding their image quality and power spectrum differences. The first row shows the original images and the downsampled versions (downsampled by factor eight) with different pooling methods. After MaxPooling the zebra's fur structures are much less recognizable. When simple downsampling via stride is applied, grid structures appear, and we observe aliasing artifacts. FLC Pooling removes these aliasing artifacts. However, ringing artifacts surrounding the zebra's head become visible. Only our ASAP is able to downsample the image without artifacts. The second row presents the power spectrum (in log scale for the y-axis, x-axis presents the frequency bands) of the images. The first column represents the original power spectrum of the image. Underlying each power spectrum of the downsampled versions we plotted the spectrum of the original image in red. Our ASAP variants are the only methods to achieve a similar power spectrum to the original image.

6.3.1 Artifact Representation

The subsequent analysis focuses on the artifacts produced by improper downsampling. We begin with qualitative observations and then proceed to a quantitative assessment of different downsampling techniques.

Qualitative Analysis in the Spatial Domain. We first visually inspect downsampling artifacts for several downsampling stages (factor 2, 4, and 8) in a toy example in Figure 6.7. MaxPooling (first column) has the expected effect of disintegrating the spatial structure of the sample. In the bottom row of column two, the sinc-interpolation artifacts for FLC Pooling, discussed in Section 6.2.3, become visible: they appear as ringing artifacts. The same effect can be observed in Figures 6.1 and 6.6. To mitigate these artifacts, we apply a Hamming filter, as described in Section 6.2.3, suppressing these artifacts (four last columns of Figure 6.7). A remaining, potentially undesired effect is the slight shift of the signal to the lower right, which is removed by the stabilization in all ASAP variants.

Qualitative Analysis in the Frequency Domain. Figure 6.6 depicts the 1D power spectrum after downsampling for a more realistic example. The first column depicts the original image with the full power spectrum. Each column presents a different downsampling technique, with the qualitative result after downsampling in the first row and its power spectrum in the second row. The power spectrum for each downsampling method in Figure 6.6 demonstrates that the strided convolution alters the frequency spectrum and is not able to represent the high-frequency components accurately compared to the original. While the power spectrum of the MaxPooled images appears more similar to the original power spectrum than the strided convolution, it still missed to represent the low-frequency components correctly. FLC Pooling introduces slightly more varied frequency components, which can be attributed to the additional ringing artifacts resulting from sinc-interpolation.

FIGURE 6.7: **Qualitative Image Comparison of Different Downsampling Techniques.** We qualitatively compare the influence of downsampling using different methods. The first column shows the commonly used MaxPooling. In the second column, we use FLC Pooling, and in the third column, we show FLC with an additional Hamming window applied. In the last three columns, we present our ASAP variants, which include the Hamming window and different stabilization techniques.

In contrast, all our ASAP variants obtain a power spectrum similar to the one of the original image.

Quantitative Analysis. To quantify several different downsampling methods, we evaluate the aliasing measure proposed in Section 5.2.1 as well as the difference in power spectrum for different downsampling methods in Table 6.1. Methods that apply blurring before downsampling, including prior work [Zhang (2019); Zou et al. (2023)], FLC Pooling, and all variants of our proposed ASAP, best preserve the power spectrum of the original image. Surprisingly, while the qualitative analysis of the power spectrum after MaxPooling in Figure 6.6 appeared satisfactory, the power spectrum seems to be altered in an undesirable way, leading to a high difference in power spectrum when evaluated over several images. With respect to the amount of aliasing introduced by the downsampling method, our FLC Pooling and all variants of ASAP are the only entirely aliasing-free approaches. ASAP$_{lp}$ offers the most favourable trade-off, being aliasing-free and preserving the power spectrum well.

Padding. As previously discussed, we introduce three variants of our ASAP to enhance its stability. Two of these variants incorporate padding, which increases the size of the feature maps. This padding can potentially reduce artifacts but also increases the computational burden for the FFT and IFFT. Figure 6.7 (columns four to six) demonstrates the qualitative visual effect of using large or small asymmetric padding before transforming into the frequency domain. In both cases, the representations are correctly centered; however, large padding is more effective at preventing ringing artifacts. On the downside, large padding significantly increases the computational cost of the FFT. Depending on the padding size, this results in an 8% increase

TABLE 6.1: **Aliasing Measure and Power Spectrum Difference for various Downsampling Techniques.** The mean and standard deviation of the aliasing measure proposed in Chapter 5 and power spectrum difference, measured via KL divergence, after downsampling with conventional methods and our downsampling methods over 1000 images from CIFAR-10. FLC Pooling, as well as ASAP, do not suffer from aliasing, hence aliasing is zero.

Name	Aliasing ↓	Power Spectrum difference ↓
Max Pooling	0.26 ± 0.30	0.0113 ± 0.1019
Strided Downsampling	0.17 ± 0.17	0.0025 ± 0.0070
BlurPooling [Zhang (2019)]	0.17 ± 0.17	**9e-06 ± 0.0008**
ABlurPooling [Zou et al. (2023)]	0.22 ± 0.23	0.0007 ± 0.0115
Wavelet Pooling [Li et al. (2020)]	0.84 ± 0.30	0.0025 ± 0.0070
FLC Pooling (ours)	**0**	0.0036 ± 0.0107
$\text{ASAP}_{\text{stbl}}$ (ours)	**0**	0.0011 ± 0.0049
ASAP_{sp} (ours)	**0**	0.0006 ± 0.0075
ASAP_{lp} (ours)	**0**	0.0004 ± 0.0071

TABLE 6.2: **Time Evaluation of Additional Padding within our ASAP.** We evaluate the time for a single execution on 32×32 CIFAR-10 input images across 100 independent runs over the validation set on an NVIDIA A100. Small padding adds only one line of zeros at the bottom and left. Large padding adds padding equal to half of the input size, with one line less on the top and right. A sequence includes three stacked operations, such that the input is downsampled by a factor of eight, while a single execution only downsamples by a factor of two.

Padding Size	Sec Per Single Execution ↓	Sec Per Sequence ↓
No Padding	0.0522 ± 0.0181	0.0738 ± 0.0195
Small Padding	0.0623 ± 0.0211	0.0794 ± 0.0185
Large Padding	0.2903 ± 0.0254	0.3782 ± 0.0282

in computational cost for small padding and a $5.1\times$ increase for large padding on an NVIDIA A100, as reported in Table 6.2. For a single execution on the largest feature map size of 32×32, the computational time increases by 19% for small padding and $5.6\times$ for large padding.

TABLE 6.3: **Native Robustness of Aliasing-Free Downsampling on ImageNet-1k.** Clean and robust accuracy (in percent, higher is better) for several common models trained without adversarial training on ImageNet-1k. Attacks are performed with an epsilon of $\epsilon = 1/255$ to reveal the difference in native robustness. While standard robustness evaluations are typically conducted with $\epsilon = 4/255$ this attack strength would cause all networks to fail completely. Our FLC Pooling and ASAP strengthen the native robustness against adversarial attacks. In addition, ASAP_{sp} outperforms all methods on common corruptions.

Arch	Method	Acc@1	Acc@5	APGD	FGSM	Corr@1	Corr@5
ResNet-18	Baseline	69.56	89.09	0.01	21.20	34.37	54.66
	BlurPooling 2019	71.38	90.12	0.07	21.78	35.97	56.48
	Wavelet Pooling 2020	71.29	90.12	0.01	23.10	37.53	58.34
	FLC Pooling (ours)	69.16	88.91	**0.32**	35.21	40.19	61.82
	$\text{ASAP}_{\text{stbl}}$ (ours)	69.53	89.11	0.31	36.44	40.33	61.99
	ASAP_{lp} (ours)	71.18	89.93	0.31	**40.33**	43.14	64.63
	ASAP_{sp} (ours)	**71.54**	**90.24**	0.28	39.57	**43.44**	**64.87**
ResNet-50	Baseline	75.85	92.88	0.07	36.00	40.80	61.18
	BlurPooling 2019	77.19	93.38	0.24	39.44	43.03	63.60
	Wavelet Pooling 2020	76.56	92.95	0.13	38.45	42.13	62.79
	low-pass 2021	77.50	-	-	-	30.00	-
	low-pass 2021 + RA	78.40	-	-	-	34.50	-
	low-pass 2021 + Swish + RA	**78.80**	-	-	-	35.10	-
	FLC Pooling (ours)	77.13	93.44	0.53	**59.05**	50.51	71.59
	$\text{ASAP}_{\text{stbl}}$ (ours)	77.12	93.45	0.67	57.72	50.76	71.87
	ASAP_{lp} (ours)	78.11	94.00	0.67	58.37	53.34	74.00
	ASAP_{sp} (ours)	78.54	**94.10**	**0.94**	56.84	**54.20**	**74.83**
ResNet-101	Baseline	77.25	93.54	0.04	38.66	46.09	67.01
	BlurPooling 2019	78.15	94.03	0.25	42.76	46.92	67.70
	Wavelet Pooling 2020	78.06	93.96	0.04	46.11	48.49	69.16
	FLC Pooling (ours)	78.16	93.95	1.26	58.49	52.91	73.72
	$\text{ASAP}_{\text{stbl}}$ (ours)	78.11	94.12	1.32	**59.56**	53.13	73.99
	ASAP_{lp} (ours)	79.07	94.35	1.57	58.99	55.82	76.19
	ASAP_{sp} (ours)	**79.34**	**94.63**	**1.78**	58.06	**56.14**	**76.51**
WideResNet-50-2	Baseline	78.29	94.03	0.28	37.74	45.23	65.75
	BlurPooling 2019	78.60	94.18	0.60	39.39	46.22	66.58
	FLC Pooling (ours)	79.67	94.74	**0.64**	**54.00**	48.48	69.22
	$\text{ASAP}_{\text{stbl}}$ (ours)	79.68	94.71	0.61	53.43	48.99	69.63
	ASAP_{lp} (ours)	80.01	**94.98**	0.45	49.88	50.17	70.40
	ASAP_{sp} (ours)	**80.31**	94.96	0.51	48.52	**50.93**	**70.98**
MobileNet-v2	Baseline	71.36	90.12	0.00	14.29	34.13	54.43
	BlurPooling 2019	**72.47**	**90.69**	0.00	13.37	34.33	54.50
	Wavelet Pooling 2020	71.94	90.46	0.00	13.04	34.28	54.61
	FLC Pooling (ours)	66.81	87.72	0.26	25.21	34.70	55.64
	$\text{ASAP}_{\text{stbl}}$ (ours)	66.83	87.71	0.29	25.58	35.10	56.18
	ASAP_{lp} (ours)	68.70	88.80	**0.42**	**28.37**	37.15	58.48
	ASAP_{sp} (ours)	69.14	88.87	0.41	28.06	**38.34**	**59.98**

6.3.2 Native Robustness

To evaluate the native robustness, *i.e.* the robustness of models trained without AT, of our proposed FLC Pooling and ASAP, we conduct experiments using two different datasets, ImageNet-1k [Deng et al. (2009)] and CIFAR-10 [Krizhevsky (2009)].

High-Resolution Data. We use ImageNet-1k [Deng et al. (2009)] and trained one network per ASAP method (stabilized FFT, large and small padding). The baseline networks utilized the pre-trained weights provided by PyTorch. The weights for BlurPooling are provided by Zhang (2019) and for Wavelet Pooling by Li et al. (2020). Zou et al. (2023) only provided weights for ResNet-101. For our FLC Pooling and ASAP, we follow the training procedures suggested by the original authors of each network.

Table 6.3 presents the performance of each network on clean, perturbed, and corrupted versions of the ImageNet-1k dataset. All models benefit from the use of our ASAP method for robustness against common corruptions. Consistently, our ASAP_{sp} outperforms all other methods. Interestingly, all of our ASAP variants outperform the baseline and all other state-of-the-art methods, like BlurPooling [Zhang (2019)], adaptive BlurPooling (ABlurPooling) [Zou et al. (2023)] or WaveletPooling [Li et al. (2020)] on the corrupted data. For all ResNet-like networks, the clean performance of the network is improved with our ASAP_{sp} and ASAP_{lp}. However, for MobileNet-v2 [Sandler et al. (2018)], our downsampling methods cannot beat the baseline in terms of clean accuracy. We hypothesize that this behaviour is due to the highly optimized training schedule used to train a MobileNet-v2. Thus, including a new kind of downsampling might require additional finetuning of the training hyperparameters. Analysing the adversarial robustness of our FLC Pooling and ASAP networks, we observe a trend towards higher robustness against FGSM and APGD for all downsampling methods, including the removal of high-frequency information in the frequency domain. Hence, networks with our ASAP and our FLC Pooling can maintain high accuracy under FGSM attack. The stronger APGD attack is able to fool the baseline in almost all cases completely. Other methods against aliasing are similarly weak in preventing the model from being fooled. In contrary, networks using our ASAP and FLC Pooling cannot be fooled on all samples by the attack. Comparing the robustness against common corruptions ASAP_{sp} outperforms all other methods for all architecture, even the ResNet-50 models provided by Vasconcelos et al. (2021) which are specialized to perform well on common corruptions by including a low-pass filter as well as RandAugment [Cubuk et al. (2020)] and exchanges activation function using Swish [Ramachandran et al. (2018)].

Low-Resolution Data. We train ResNet-18 [He et al. (2016a)] and Wide-ResNet-50-2 [Zagoruyko & Komodakis (2016)] models on CIFAR-10 [Krizhevsky (2009)] with five different random seeds per network architecture. We compare the standard baseline network, BlurPooling [Zhang (2019)], ABlurPooling [Zou et al. (2023)] and WaveletPooling [Li et al. (2020)]. All networks are trained with the same set of hyperparameters: 150 epochs, a batch size of 256, a cosine learning rate schedule with a maximum learning rate of 0.2 and a minimum of 0.0, a momentum of 0.9, and a weight decay of 0.002. We utilize label smoothing with a factor of 0.1, and Stochastic Gradient Descent (SGD) for optimization.

TABLE 6.4: **Native Robustness of Aliasing-Free Downsampling on CIFAR-10.** Mean accuracy (in percentage) and standard deviation on clean samples, perturbed samples with FGSM [Goodfellow et al. (2015)] and PGD [Madry et al. (2018)] as well as corrupted samples [Hendrycks & Dietterich (2019)] for four different architectures (five different random seeds) trained without adversarial training on CIFAR-10. Attacks are done with an epsilon of $\epsilon = 1/255$ and corruption performance is reported as mean over all severities. For CIFAR-10 our ASAP and FLC Pooling outperform the baseline and show overall a high robustness against adversarial attacks and common corruptions.

Arch	Method	Acc@1 ↑	FGSM ↑	PGD ↑	Corruptions ↑
ResNet-9	Baseline 2023	94.29	59.58	53.04	-
	DCT Conv WD 2023	93.18	59.25	56.08	-
	DCT Conv SD 2023	93.09	59.87	56.89	-
	FLC Pooling (ours)	94.53 ± 0.11	**69.05 ± 0.22**	**66.60 ± 0.58**	**75.29 ± 0.75**
	$ASAP_{stbl}$ (ours)	**94.56 ± 0.16**	68.96 ± 0.61	65.81 ± 0.89	74.79 ± 0.64
	$ASAP_{lp}$ (ours)	94.52 ± 0.15	68.56 ± 0.52	66.06 ± 1.04	75.04 ± 0.73
	$ASAP_{sp}$ (ours)	94.55 ± 0.08	68.52 ± 0.53	65.28 ± 1.03	74.61 ± 0.73
ResNet-18	Baseline	93.03 ± 0.13	78.62 ± 0.28	72.49 ± 0.67	76.93 ± 0.45
	BlurPooling 2019	**93.25 ± 0.17**	79.24 ± 0.32	75.23 ± 0.55	77.70 ± 0.54
	ABlurPooling 2023	92.77 ± 0.15	**79.65 ± 0.52**	**76.94 ± 0.79**	76.59 ± 0.33
	WaveletPooling 2020	93.00 ± 0.06	79.15 ± 0.15	72.71 ± 0.55	78.40 ± 0.23
	DCT Conv WD 2023	88.80 ± 0.16	60.53 ± 1.06	58.65 ± 1.21	73.88 ± 0.41
	DCT Conv SD 2023	89.93 ± 0.10	61.37 ± 0.51	59.40 ± 0.74	75.84 ± 0.35
	FLC Pooling (ours)	93.12 ± 0.19	78.92 ± 0.26	74.17 ± 0.60	78.59 ± 0.29
	$ASAP_{stbl}$ (ours)	93.12 ± 0.25	79.08 ± 0.43	75.06 ± 0.76	**78.68 ± 0.19**
	$ASAP_{lp}$ (ours)	93.24 ± 0.15	79.17 ± 0.23	74.94 ± 0.56	78.65 ± 0.33
	$ASAP_{sp}$ (ours)	93.00 ± 0.12	79.12 ± 0.49	74.69 ± 1.36	78.42 ± 0.20
WideResNet-50-2	Baseline	94.33 ± 0.13	77.92 ± 0.64	69.36 ± 1.20	77.08 ± 0.38
	BlurPooling 2019	94.42 ± 0.12	76.21 ± 0.30	68.66 ± 0.49	77.59 ± 0.47
	ABlurPooling 2023	93.66 ± 0.18	78.26 ± 1.50	71.76 ± 1.91	76.74 ± 1.10
	WaveletPooling 2020	94.44 ± 0.13	78.12 ± 1.11	69.26 ± 1.02	**79.95 ± 0.42**
	FLC Pooling (ours)	94.33 ± 0.20	75.41 ± 0.41	66.30 ± 0.87	79.33 ± 0.43
	$ASAP_{stbl}$(ours)	**94.51 ± 0.17**	77.22 ± 0.89	71.24 ± 1.86	79.90 ± 0.37
	$ASAP_{lp}$ (ours)	94.20 ± 0.18	**78.63 ± 0.57**	**72.36 ± 0.67**	79.72 ± 0.41
	$ASAP_{sp}$(ours)	94.16 ± 0.21	77.30 ± 0.24	70.74 ± 1.14	79.61 ± 0.82
AlexNet	Baseline	**89.45 ± 0.23**	69.28 ± 0.22	69.97 ± 0.28	73.74 ± 0.13
	FLC Pooling (ours)	87.80 ± 0.10	70.40 ± 0.36	71.49 ± 0.36	**74.33 ± 0.50**
	$ASAP_{stbl}$ (ours)	87.59 ± 0.23	70.50 ± 0.26	71.52 ± 0.22	73.84 ± 0.27
	$ASAP_{lp}$ (ours)	87.90 ± 0.09	**70.87 ± 0.26**	**71.93 ± 0.20**	74.02 ± 0.42
	$ASAP_{sp}$ (ours)	87.68 ± 0.16	70.71 ± 0.30	71.82 ± 0.26	73.94 ± 0.21

Table 6.4 presents the results of the low-resolution (32×32 pixel) dataset, CIFAR-10. FLC Pooling and ASAP consistently outperform the baseline in terms of native robustness while maintaining similar clean performance. On ResNet-18, ABlurPooling [Zou et al. (2023)] achieves the highest native robustness against adversarial attacks, yet with a slight decrease in clean performance and robustness against common corruptions.

Further, we compare our method to [Rodríguez-Muñoz & Torralba (2022)], which uses anti-aliasing mechanisms for downsampling and activation function to gain high native robustness *i.e.* without AT. Figure 6.8 demonstrates that our improved downsampling techniques can consistently provide higher native robustness compared to [Rodríguez-Muñoz & Torralba (2022)].

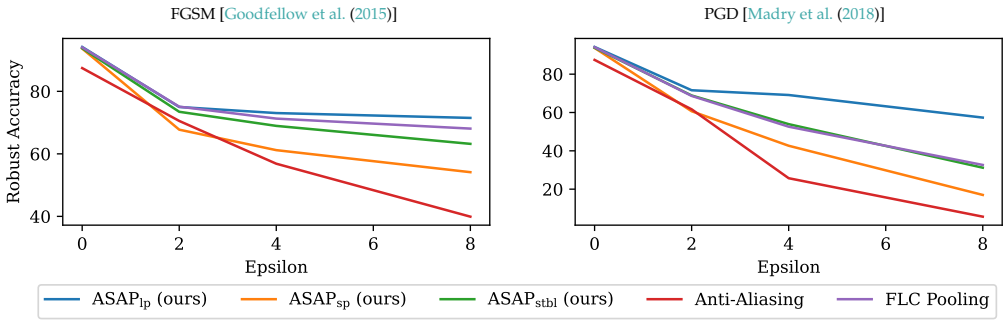


FIGURE 6.8: **ASAP exhibits High Robustness under Different Attack Budgets.** Native robustness for different attack epsilon budgets of ResNet-50 on CIFAR-10. We compare models using our FLC Pooling and ASAP to the approach by Rodríguez-Muñoz & Torralba (2022) (Anti-Aliasing). Our downsampling variants consistently exhibit higher native robust accuracy on adversarial attacks than the model by Rodríguez-Muñoz & Torralba (2022).

TABLE 6.5: **Consistency under Spatial Shifts for Different Downsampling Techniques.** Mean consistency and standard deviation under spatial shifts of different downsampling methods. We trained five different random seeds on a PRN-18 model without AT on CIFAR-10. ASAP$_{lp}$ performs best under spatial shifts and FLC Pooling second best.

Model	Acc@1 ↑	Consistency under shift ↑
Baseline	93.03 ± 0.13	88.02 ± 6.51
BlurPooling 2019	**93.25 ± 0.17**	91.38 ± 3.25
ABlurPooling 2023	92.77 ± 0.15	88.94 ± 3.95
FLC Pooling (ours)	93.12 ± 0.19	91.77 ± 3.18
ASAP$_{stbl}$ (ours)	93.12 ± 0.25	88.80 ± 4.23
ASAP$_{sp}$ (ours)	93.24 ± 0.15	88.19 ± 2.74
ASAP$_{lp}$ (ours)	93.00 ± 0.12	**91.85 ± 2.76**

Shift-Invariance. Anti-aliasing in CNNs has previously been discussed in the context of shift-invariance [Zhang (2019)]. Following our evaluation of the model against adversarial attacks and common corruptions, we further analyse the performance of FLC Pooling and ASAP under pixel shifts. Our model is compared with the baseline as well as the shift-invariant models proposed by Zhang (2019) and Zou et al.

(2023) in Table 6.5. Both FLC Pooling and ASAP_{lp} demonstrate the highest consistency under pixel shifts, outperforming sophisticated methods such as BlurPooling [Zhang (2019)] and ABlurPooling [Zou et al. (2023)], which are specifically designed to enhance shift-invariance.

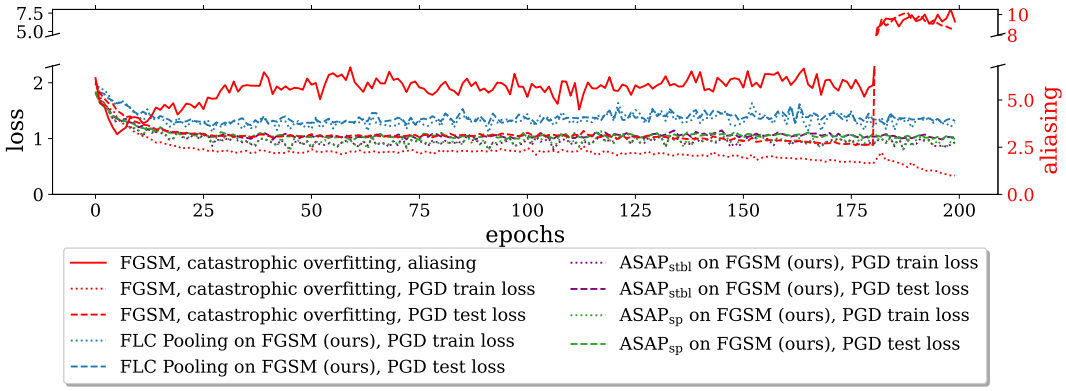


FIGURE 6.9: **Catastrophic Overfitting in FGSM AT and Aliasing.** Example of FGSM AT facing catastrophic overfitting and its relationship to aliasing. FGSM training is prone to catastrophic overfitting (solid lines) and experiences a significant increase in aliasing (red solid line) as soon as catastrophic overfitting occurs. Specifically, the error on stronger adversaries, like PGD, increases (blue solid line), while the error on the simpler adversary, FGSM, (yellow solid line) remains low. Our methods, FLC Pooling and ASAP, are able to train with fast FGSM AT while preventing catastrophic overfitting (dashed and dotted lines).

6.3.3 Adversarial Training and Catastrophic Overfitting

In the following, we demonstrate that our FLC Pooling and ASAP can enhance AT and help in mitigating catastrophic overfitting in FGSM AT. Catastrophic overfitting occurs when models adversarially trained with FGSM [Goodfellow et al. (2015)] overfit to the FGSM attack itself [Kim et al. (2021b); Wong et al. (2020)], leading to a significant drop in robustness against stronger attacks like PGD [Madry et al. (2018)]. This phenomenon typically arises after several training epochs. A common technique to address catastrophic overfitting is early stopping, which typically relies on monitoring PGD accuracy during training and halting once it declines [Wong et al. (2020)]. However, early stopping interrupts the training process and may leave networks under-converged. In Chapter 5, we observed that catastrophic overfitting in FGSM AT often correlates with increased aliasing in the model's downsampling layers. We proposed a stopping criterion based on our aliasing measure as an alternative to early stopping based on PGD. Nevertheless, interruptions due to early stopping remains a limitation. Our FLC Pooling and ASAP, both aliasing-free techniques, can reduce the risk of catastrophic overfitting in FGSM AT. These methods enable the use of cost-effective FGSM AT while achieving results comparable to more computationally expensive methods like PGD.

Qualitative Analysis. In Figure 6.9, we evaluate the training loss during FGSM AT on CIFAR-10 with different downsampling techniques. The baseline model clearly exhibits catastrophic overfitting, showing a huge increase in PGD test loss and, additionally, a huge increase in aliasing at the same time. In comparison our FLC Pooling, $\text{ASAP}_{\text{stbl}}$ and ASAP_{sp} which are aliasing-free, do no experience this increase in PGD test loss, indicating that they do not experience catastrophic overfitting. As

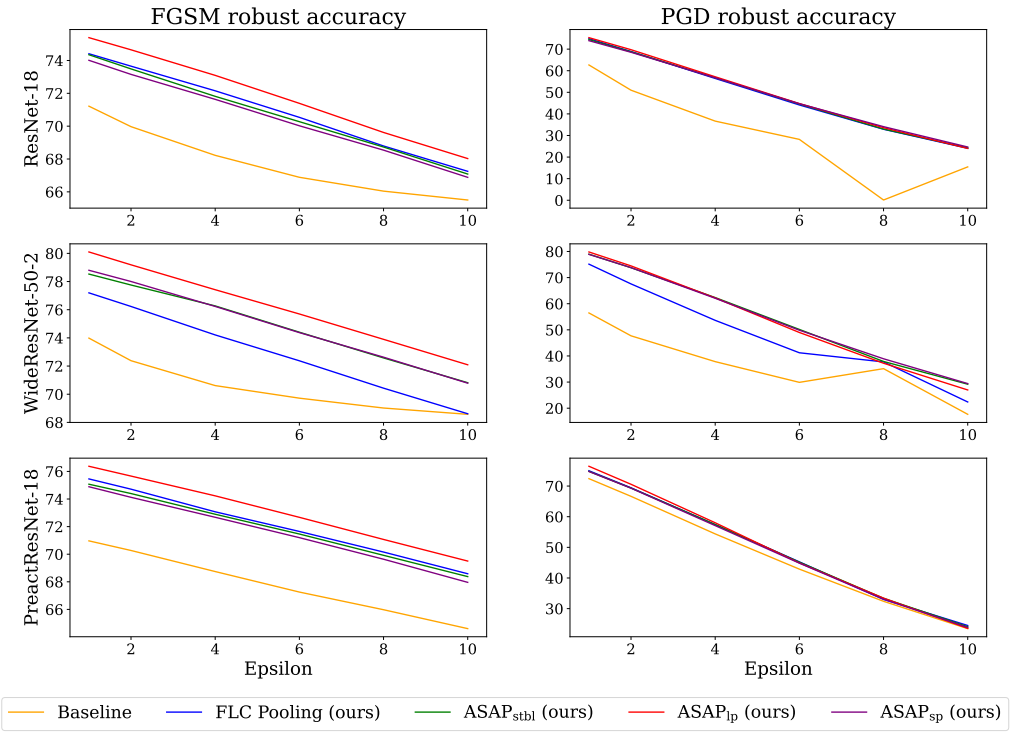


FIGURE 6.10: **Robustness of our Downsampling Approaches with AT.** Evaluation (accuracy in percent) of networks trained with FGSM AT [Goodfellow et al. (2015)] on CIFAR-10 evaluated on FGSM [Goodfellow et al. (2015)] (left) and PGD [Madry et al. (2018)] (right) adversaries with different budgets of ϵ. Our three ASAP variants consistently exhibit higher robust accuracy on all architectures, adversarial attacks and across ϵ values than the baseline.

a result, networks incorporating our downsampling methods can be trained to full convergence without requiring early stopping. To validate these empirical findings, we strengthen our hypothesis by AT different networks on CIFAR-10 with three different random seeds in the following paragraph.

Adversarial Training on Low-Resolution Data. For CIFAR-10, we trained each model architecture with FGSM AT using three different random seeds. All hyperparameters were kept consistent across architectures and downsampling methods. Each network underwent 300 training epochs with a batch size of 512 and a cycling learning rate schedule ranging from a maximum of 0.2 to a minimum of 0.0. The momentum was set to 0.9, and weight decay was set to 0.0005. We employed CrossEntropy loss and utilized SGD as optimizer. The budget for the adversaries during training was set to $\epsilon = 8/255$.

The results in Table 6.6 indicate that ASAP and FLC Pooling achieve higher robustness against adversarial attacks than the baseline. Particularly when confronted with more complex adversaries like PGD [Madry et al. (2018)], our ASAP$_{lp}$ consistently outperforms the baseline and FLC Pooling. The high variance in performance on PGD samples for the ResNet-18 and Wide-ResNet-50-2 baselines indicate that some of the trained models lose all their robustness against PGD during FGSM AT due to catastrophic overfitting. In contrast, our FLC Pooling and ASAP do not experience this issue and maintain high robustness against strong and simple adversaries in all models. For PreAct-ResNet-18, which is commonly used for AT [Gowal et al. (2021b); Rade & Moosavi-Dezfooli (2021); Rebuffi et al. (2021)], none of the networks

TABLE 6.6: **Robustness of various Architectures with our Downsampling Approaches with AT.** Accuracy (in percent) for several common models trained with FGSM AT [Goodfellow et al. (2015)] on CIFAR-10. We report adversarial robustness against FGSM [Goodfellow et al. (2015)] and PGD [Madry et al. (2018)] with an ϵ of 8/255. We clearly see that our ASAP, which neither suffers from aliasing nor sinc artifacts, is also more robust in combination with adversarial training.

	Method	Acc@1 ↑	FGSM ↑	PGD ↑
ResNet-18	Baseline	78.85 ± 1.74	34.49 ± 2.68	21.14 ± 14.88
	FLC Pooling	79.77 ± 0.49	34.37 ± 1.07	32.23 ± 0.68
	$\text{ASAP}_{\text{stbl}}$	79.59 ± 0.64	35.13 ± 0.75	32.65 ± 0.59
	ASAP_{lp}	**80.63 ± 0.14**	**37.04 ± 0.65**	33.43 ± 0.13
	ASAP_{sp}	79.19 ± 0.32	35.44 ± 0.75	**33.68 ± 0.31**
Wide-ResNet -50-2	Baseline	79.42 ± 0.34	39.18 ± 7.15	23.36 ± 16.35
	FLC Pooling	82.94 ± 0.89	39.23 ± 0.32	29.69 ± 11.81
	$\text{ASAP}_{\text{stbl}}$	83.63 ± 0.14	**39.67 ± 0.28**	37.62 ± 0.24
	ASAP_{lp}	**84.60 ± 0.13**	39.56 ± 0.88	36.99 ± 0.19
	ASAP_{sp}	83.26 ± 0.24	39.16 ± 0.37	**38.86 ± 0.17**
PreAct- ResNet-18	Baseline	77.92 ± 0.19	31.74 ± 0.56	32.52 ± 0.35
	FLC Pooling	79.99 ± 0.09	36.39 ± 0.74	33.15 ± 0.19
	$\text{ASAP}_{\text{stbl}}$	79.91 ± 0.17	36.25 ± 0.20	33.20 ± 0.14
	ASAP_{lp}	**81.29 ± 0.20**	**38.02 ± 0.85**	**33.48 ± 0.05**
	ASAP_{sp}	79.77 ± 0.20	35.88 ± 0.49	33.35 ± 0.19

experiences catastrophic overfitting. This is one reason why this network architecture is frequently used for AT. Still, our FLC Pooling and ASAP outperform the baseline on clean and perturbed samples. Furthermore, all ASAP variants exhibit superior robustness against FGSM attacks and higher clean accuracy compared to the baseline (up to 5% improvement against FGSM attacks and up to 4% improvement on clean data). ASAP_{lp} improves the clean as well as the robust performance for the smaller models like ResNet-18 and PreAct-ResNet-18. The larger Wide-ResNet-50-2 only benefits from the large padding for clean accuracy.

CIFAR Training Efficiency. The incorporation of our FLC Pooling and ASAP into FGSM AT increases the computational costs due to repeated FFT and IFFT as well as potential padding. However, due to the efficiency of FGSM AT it is still much more efficient than comparable AT methods as presented in Table 6.7. Our analysis demonstrates that incorporating additional FFT operations increases training time by only a factor of 1.3, whereas more sophisticated AT, like PGD [Madry et al. (2018)] or TRADES [Zhang et al. (2019b)] increase the training time by a factor of at least 9 or even 17, respectively. Our proposed ASAP increases the training time dependent on the variant used. Simple stabilization by the transpose operation and small padding increase the AT time per epoch by a factor of 1.2. While large padding increases the training time by a factor of 2.8, it is still faster than other sophisticated AT methods. Further, we want to point out that incorporating additional data like *ddpm* [Ho et al. (2020)], which is a widely used source for AT [Gowal et al. (2021b); Rade & Moosavi-Dezfooli (2021); Rebuffi et al. (2021)] increases the training time by a factor of 20.

TABLE 6.7: **Runtime of different AT schemes.** Runtime of different AT schemes in seconds per epoch over 200 epochs with a batch size of 512, using PRN-18 on the original CIFAR-10 dataset without additional data. Experiments were conducted on a single Nvidia Tesla V100. We evaluate clean and robust accuracy (higher is better) on APGD [Croce & Hein (2021)] using our trained models. Note that the models reported by the original authors may yield varying results due to differences in hyperparameter selection. The top row shows the baseline without AT.

Method	Average #seconds per epoch ↓	Acc (%) Acc@1 ↑	APGD ↑
Baseline	6.6 ± 0.01	93.06	0.00
FGSM & early stopping 2020	12.6 ± 0.01	82.88	11.82
FGSM & FLC Pooling (ours)	14.7 ± 0.01	80.94	31.16
FGSM & $\text{ASAP}_{\text{stbl}}$ (ours)	15.6 ± 0.08	80.47	31.75
FGSM & ASAP_{sp}(ours)	17.1 ± 0.16	80.47	31.40
FGSM & ASAP_{lp} (ours)	36.4 ± 0.01	81.12	31.39
PGD 2018	115.4 ± 0.2	83.11	41.12
Robustness lib 2019	117 ± 19.0	76.37	33.09
AWP 2020	179.4 ± 0.4	82.61	**53.53**
MART 2020b	180.4 ± 0.8	55.49	10.03
TRADES 2019b	219.4 ± 0.5	81.49	49.65

Ablating the Attack Strength. Additionally, we assessed the behaviour of our FLC Pooling and ASAP under different budget settings of ϵ for FGSM and PGD. Figure 6.10 displays the mean robust accuracy trend across each architecture, varying the budget of ϵ. It is evident that our ASAP consistently outperforms the baseline. Moreover, ASAP_{lp} demonstrates superior performance over all epsilon strengths under FGSM attack. In comparison, under PGD attack all our ASAP variants perform equally well.

TABLE 6.8: **Robustness Evaluation of our Downsampling on ImageNet-1k with AT.** Comparison of ResNet-50 models trained on ImageNet-1k. The first row presents the baseline without AT. The second, third and fourth row are trained with FGSM AT and the last three row with PGD AT. The clean and robust accuracy (against AutoAttack [Croce & Hein (2020a)]) are present in percentage on the ImageNet-1k validation set. We compare against models reported on RobustBench [Croce et al. (2021)].

	Method	Acc@1 ↑	AA L_{inf} $\epsilon = \frac{4}{255}$ ↑
	Non-adversarial training [Croce et al. (2021)]	76.52	0.00
FGSM	FLC Pooling (ours)	63.52	27.29
FGSM	ASAP_{sp} (ours)	64.51	30.93
FGSM	Wong *et al.*, 2020 [Wong et al. (2020)]	55.62	26.24
PGD	Robustness lib, 2019 [Engstrom et al. (2019)]	62.56	29.22
PGD	Salman *et al.*, 2020 [Salman et al. (2020)]	64.02	34.96
PGD	ASAP_{sp} (ours)	64.54	31.02

Adversarial Training on High-Resolution Data. We also train our FLC Pooling and ASAP_{sp} with FGSM and PGD AT on ImageNet-1k. We trained on a ResNet-50

and compared models reported on RobustBench [Croce et al. (2021)] with the same architecture. Table 6.8 demonstrates that our FGSM trained model outperforms the baseline model trained with FGSM [Wong et al. (2020)] in robust and clean accuracy, and even the model trained by Engstrom et al. (2019) which takes significantly longer than our method as shown in Table 6.7. The model by Salman et al. (2020) achieves higher robustness, while being slightly worse on clean samples than our FGSM trained ASAP_{sp} model. Further, Table 6.8 also indicates that PGD training can benefit from proper downsampling. For this evaluation, we train a ASAP_{sp} ResNet-50 with the training schedule by [Engstrom et al. (2019)] and achieve higher robustness and clean accuracy than their baseline. We also achieve higher clean performance than [Salman et al. (2020)] while not relying on extra data.

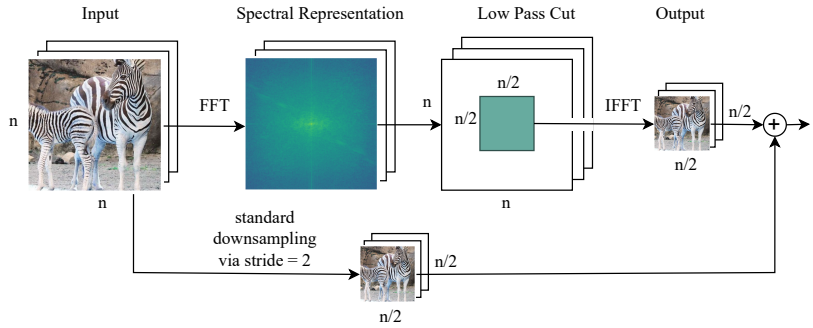
ImageNet Training Efficiency. When evaluating practical training times (in minutes) on ImageNet-1k per epoch, we can not see a measurable difference in the costs between a ResNet-50 with FLC Pooling or strided convolution.

We varied the number of workers for dataloaders with clean training on 4 A-100 GPUs and measured $\approx$ 43m for 12 workers, $\approx$ 22m for 48 workers and $\approx$ 18m for 72 workers for both. FGSM-based AT with the pipeline by [Goodfellow et al. (2015)] takes 1:07 hours for both FLC Pooling and strided convolutions per epoch. We conclude that training with FLC Pooling in terms of practical runtime is scalable (runtime increase in ms-s range) and training times are likely governed by other factors.

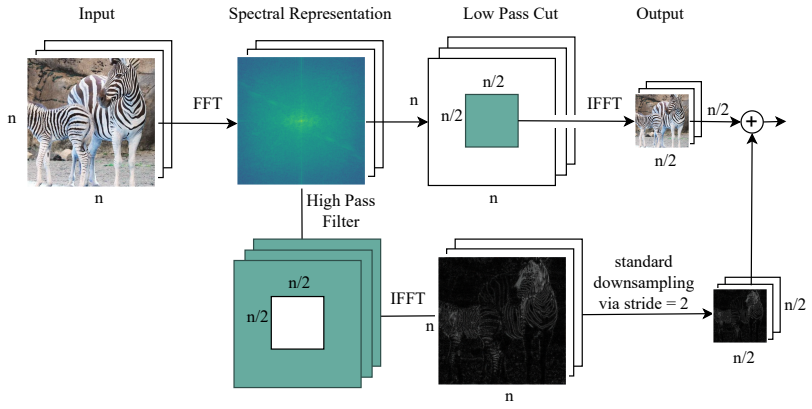
The training time of our model should be comparable to the one from Wong *et al*. [Wong et al. (2020)] while other reported methods have a significantly longer training time. Yet, the clean accuracy of the proposed model using FLC Pooling improves about 8% over the one reached by [Wong et al. (2020)], with a 1% improvement in robust accuracy. For instance, the training procedure proposed in [Engstrom et al. (2019)] increases the training time by factor four on CIFAR-10 compared to our model (see Table 6.7). This model achieves overall comparable results to ours. The model by Salman et al. (2020) is trained with the training schedule proposed by **?** and uses a multi-step adversarial attack for training. Since there is no release of the training script of this model on ImageNet, we can only roughly estimate their training times. Since they adopt the training schedule from **?**, we assume a similar training time increase of a factor of four, which is similar to the multi-step times reported for PGD in Table 6.7.

6.3.4 Ablation Studies

In the following, we conduct a series of ablation studies. Specifically, we investigate the effect of including additional frequency components in FLC Pooling. Moreover, we ablate the specific window functions to be used and the combination of the different ASAP variants. Further, we ablate the frequency spectrum of adversarial attacks of our different downsampling variants similar to our evaluations in Section 5.3.1 Figures 5.7 and 5.15. Lastly, we revisited the findings from Section 4.2.2, where we discovered that basic building blocks influence a model's confidence distribution, resulting in a desirable pattern: high confidence in correct predictions and low confidence in incorrect ones. To further investigate, we conducted an ablation study analysing three different random seeds for each model. Our results revealed that, in addition to the building blocks, hyperparameters as simple as the random seed also play a role in shaping the confidence distribution.



(a) FLC Pooling Plus Downsampling via Stride Two.



(b) FLC Pooling Plus High-Pass Filtered Downsampling.

FIGURE 6.11: **FLC Pooling Plus.** Adding additional frequency components to the low-cut frequencies. FLC Pooling Plus either includes the original downsampled signal with strided convolution applied (a), or the high-frequency components filtered by a high-pass filter in the Fourier domain and downsampled in the spatial domain by an identity convolution of stride two (b).

Ablation on Including Additional Frequency Components in FLC Pooling. We aim to determine whether the aggressive FLC Pooling leads to significant loss of information and whether additional high-frequency information is needed to compensate for the discarded details. Hence, in addition to the low-frequency components, we tested different settings in which a second path is established to incorporate high-frequency components or the original information. Figure 6.11 illustrates the procedure for including a second path. One approach involves standard downsampling and adding the result to the FLC pooled feature map. The other approach applies a high-pass filter to the feature map, followed by downsampling. Afterwards, the FLC pooled feature maps and the high-pass filtered, downsampled ones are added together. Table 6.9 demonstrates that both FLC Pooling Plus variants achieve minor improvements in clean accuracy but also minor degradation in robust accuracy. Therefore, we decided to retain only the low-frequency information preserved by FLC Pooling, as incorporating additional components did not improve robustness and led to increased training time per epoch as well as a slight increase in model size.

Ablation on Window Function. For our work, we mainly focused on the Hamming window, but there are several widely known window functions that could

TABLE 6.9: **Performance Evaluation of FLC Pooling Plus.** Performance evaluation of FLC Pooling with an additional path for standard downsampling or a high-pass pooled version. We train with FGSM AT [Goodfellow et al. (2015)] and evaluate clean accuracy, PGD [Madry et al. (2018)] robust accuracy, training time, and model size. The results demonstrate that incorporating additional information does not improve performance. Given the extra computational cost required for the high-frequency or original components, we chose to discard them entirely and focus solely on low-frequency cutting.

Method	Acc@1 ↑	PGD L_{inf} $\epsilon = \frac{8}{255}$ ↑	Seconds per epoch (avg) ↓	Model size (MB) ↓
FLC pooling	84.81	38.41	34.6 ± 0.1	42.648
FLC pooling + HighPass pooling	85.38	38.02	45.2 ± 0.4	42.652
FLC pooling + Original pooling	85.37	38.30	35.4 ± 0.1	42.652

be used to reduce sinc artifacts. Thus, we ablated four additional choices for the

TABLE 6.10: **Using Different Window Functions in our ASAP.** Ablation study on using different window functions in the frequency domain to reduce spectral artifacts. We report the mean clean and robust accuracy along with their standard deviations on CIFAR-10 for five different window functions, evaluated over five random seeds. The best result is marked in **bold**, and the second best is indicated with underlining. Using no window function, such as in our FLC Pooling, or a simple Gaussian window, which requires an additional hyperparameter, performs poorly. All window functions that are specializations of the Kaiser window perform reasonably well. The Hamming window, used in our ASAP method, consistently ranks among the top two performing methods.

Window	Hyperparameter	Acc@1 ↑	FGSM ↑	PGD ↑	Corruptions ↑
None		93.12 ± 0.19	78.92 ± 0.26	74.17 ± 0.60	78.59 ± 0.29
Hamming		$\underline{93.12 \pm 0.25}$	$\mathbf{79.08 \pm 0.43}$	$\underline{75.06 \pm 0.76}$	$\underline{78.68 \pm 0.19}$
Gaussian	$\sigma = (k-1)/6$	92.34 ± 0.15	77.85 ± 0.30	69.47 ± 0.40	78.53 ± 0.14
Hanning		93.13 ± 0.19	$\underline{79.23 \pm 0.27}$	$\mathbf{75.32 \pm 0.92}$	78.56 ± 0.26
Kaiser	$\beta = 7$	$\mathbf{93.21 \pm 0.17}$	78.87 ± 0.33	74.57 ± 1.25	78.66 ± 0.21
Blackman		93.00 ± 0.17	78.82 ± 0.43	74.78 ± 1.16	$\mathbf{78.77 \pm 0.21}$

window function in our ASAP method. Here, we additionally evaluate a standard Gaussian kernel, a Blackman window, a Hanning window, and a Kaiser window with $\beta = 7$. Similar to the Kaiser kernel, we needed to choose an additional hyperparameter σ for the Gaussian kernel. We set sigma σ in relation to the kernel size k such that $\sigma = (k-1)/6$ as the length of 99 percentile of the Gaussian pdf is 6σ. Table 6.10 presents the performance of five different random seeds trained on CIFAR-10 with the mentioned different window functions. One can note that the models using a Gaussian window do not support the robustness of the network well, while all models based on a Kaiser window [1] perform similarly well on clean, perturbed and corrupted data. When considering the top two performing methods, the Hamming window used for our ASAP performs consistently well and is, thus, a good choice.

Ablation on ASAP Variants. We combine our method to stabilize the FFT by transposing, $\text{ASAP}_{\text{stbl}}$ with padding. Table 6.11 ablates on this combination. We evaluate

[1] Hamming, Hanning and Blackman window are all specializations of a Kaiser window with fixed $\beta = 6.0$, $\beta = 5.0$ and $\beta = 8.6$ respectively.

TABLE 6.11: **Study on Stabilization in our ASAP.** We report accuracy (in percent) on clean samples, perturbed samples using FGSM [Goodfellow et al. (2015)] and PGD [Madry et al. (2018)], as well as corrupted samples [Hendrycks & Dietterich (2019)], for different configurations of ASAP on ResNet-18 [He et al. (2016a)] trained without adversarial training on CIFAR-10. The attacks are performed with $\epsilon = 1/255$, and corruption performance is reported as the mean over all severities.

Architecture	Acc@1 ↑	FGSM ↑	PGD ↑	Corruptions ↑
Baseline	93.03 ± 0.13	78.62 ± 0.28	72.49 ± 0.67	76.93 ± 0.45
$\text{ASAP}_{\text{stbl}}$	93.12 ± 0.25	79.08 ± 0.43	75.06 ± 0.76	**78.68 ± 0.19**
ASAP_{lp}	**93.24 ± 0.15**	**79.17 ± 0.23**	74.94 ± 0.56	78.65 ± 0.33
ASAP_{sp}	93.00 ± 0.12	79.12 ± 0.49	74.69 ± 1.36	78.42 ± 0.20
$\text{ASAP}_{\text{lp+stbl}}$	93.22 ± 0.07	78.37 ± 0.77	**75.65 ± 1.43**	78.38 ± 0.38
$\text{ASAP}_{\text{sp+stbl}}$	92.88 ± 0.11	77.29 ± 1.02	74.82 ± 1.90	78.77 ± 0.17

over five different random seeds. The combination of both approaches, stabilization through transposing the signal and padding, yields no further systematic benefit, indicating that both approaches address the same issue during pooling in the frequency domain.

Ablation on Attack Spectrum. We investigate if there is a difference in perturbations created by APGD [Croce & Hein (2020a)] depending on the models' downsampling. Figure 6.12 presents the perturbations created by APGD on conventionally trained models (top) and adversarially trained models (bottom). The perturbations on conventionally trained models target all frequency bands, as shown in the spectrum difference. While the attack mostly targets low-frequency bands for the adversarially trained models, there is no clear difference between models, including conventional downsampling or our downsampling methods.

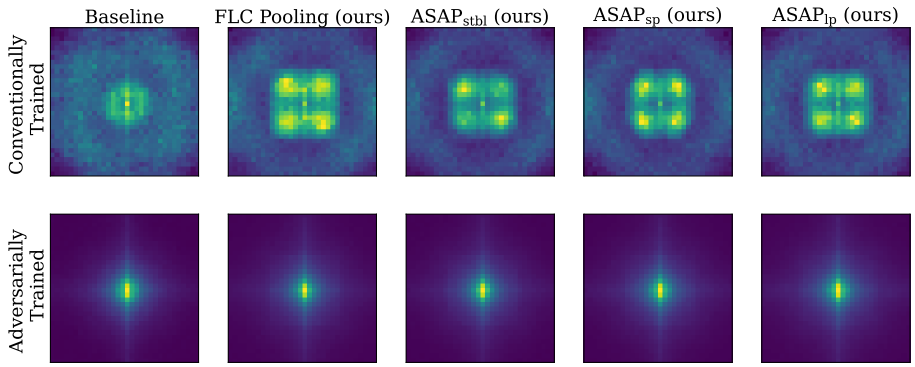


FIGURE 6.12: **Center-shifted Attack Spectrum Difference for our Downsampling approaches.** Average difference in the center-shifted spectrum over 1000 CIFAR-10 images between the clean image and the attacked image with APGD [Croce & Hein (2021)]. For conventionally trained networks (top row), the spectrum of the perturbation differs depending on the downsampling method. However, for adversarially trained networks (bottom row), there is no clear difference.

Ablation on Confidence Distributions. In Section 4.2.2, we discovered that simple building blocks like activation functions and downsampling strategies can positively impact a model's confidence distribution. While previous evaluations of a single model with FLC Pooling demonstrated a desirable confidence distribution, we now quantitatively assess whether this holds across all our downsampling approaches. Figure 6.13 presents an ablation study over three random seeds for FLC Pooling and $\text{ASAP}_{\text{stbl}}$ models trained with FGSM AT, compared to standard downsampling trained under the same conditions. Additional visualizations for all ASAP variants are presented in Figure B.1 in Appendix B.1, showing similar trends as depicted in the bottom row of Figure 6.13. On clean samples, all models display similar confidence distributions. However, under adversarial attacks, the baseline model tends to produce incorrect predictions with very high confidence. In contrast, FLC Pooling and ASAP approaches significantly reduce the number of highly confident incorrect predictions. Nevertheless, the desirable confidence distribution observed for a single model in Figure 4.8 from Section 4.2.2 does not fully generalize across different seeds. This highlights the importance of future research to identify specific hyperparameters that influence confidence distributions and enable consistent improvements.

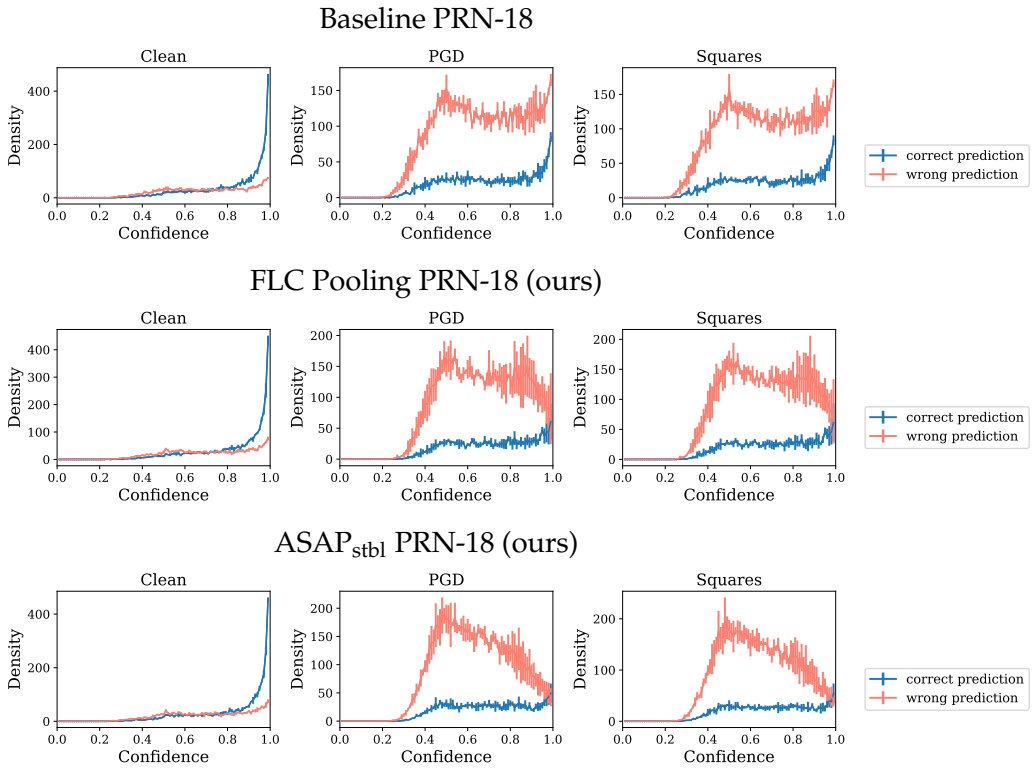


FIGURE 6.13: **Confidence Distribution of our $\text{ASAP}_{\text{stbl}}$ with FGSM AT.** Mean confidence distribution and standard deviation for PRN-18 baseline, FLC Pooling and $\text{ASAP}_{\text{stbl}}$ models trained with FGSM AT on three different seeds. While both downsampling methods perform equally well on clean samples, our models with FLC Pooling and $\text{ASAP}_{\text{stbl}}$ exhibits fewer highly confident incorrect predictions on PGD and Square attacks. However, when examining multiple random seeds, the results deviate from the observations for a single model shown in Figure 4.8 in Chapter 4, underscoring the influence of training hyperparameters.

6.4 Discussion

Our FLC Pooling and ASAP achieve higher native robustness against common corruptions and adversarial attacks while reducing the risk of catastrophic overfitting during FGSM AT. In the following, we discuss efficiency implications and outline the primary limitations

6.4.1 Efficiency

For our aliasing-free downsampling, additional operations are required to transform data between the spatial and frequency domains, resulting in increased computational effort. This computational cost becomes especially pronounced when using large additional padding, with the transformation cost increasing by a factor of 5.6 in execution time. Nevertheless, we demonstrated that small padding can yield equal or even better robustness improvements while only increasing computational effort by a factor of 1.2 compared to no padding (Table 6.2). In contrast, additional data augmentation for robustness against common corruptions increases the number of samples to be learned, and AT requires multiple forward and backward passes for each batch. As reported in Table 6.7, more sophisticated AT methods can increase training time by factors of up to 17. Additionally, incorporating external data sources, such as *ddpm* [Ho et al. (2020)], a widely used resource for AT [Gowal et al. (2021b); Rade & Moosavi-Dezfooli (2021); Rebuffi et al. (2021)], further increases training time by a factor of 20. In summary, improving downsampling to be aliasing-free and thus also integrating FFT operations into the network achieves high robustness while maintaining relatively low training time compared to other sophisticated training methods to increase robustness.

6.4.2 Limitations

FLC Pooling and ASAP consistently reduce highly confident incorrect predictions under adversarial attacks compared to the baseline, however, the desirable confidence distribution observed for a single model does not fully generalize across seeds. This variability underscores the need for future research to identify specific hyperparameters that reliably influence confidence distributions and ensure consistent improvements.

Our focus was limited to classification tasks, which appear less reliant on high-frequency information, as significant removal or inclusion as done in our ablation in Section 6.3.4 did not impact performance. However, we find in our subsequent work [Agnihotri et al. (2024a)], not included in this thesis, that for tasks that rely on fine details, like image reconstruction, the high-frequency information needs to be preserved for satisfying results.

6.5 Conclusion

We introduce two downsampling methods: aliasing-free and hyperparameter-free pooling in the frequency domain, called FrequencyLowCut Pooling (FLC Pooling) and Aliasing and Sinc Artifact-free Pooling (ASAP). FLC Pooling ensures aliasing-free downsampling, while ASAP further addresses artifacts from sinc interpolation. Our qualitative analysis confirms that FLC Pooling eliminates aliasing, and ASAP avoids both aliasing and sinc interpolation artifacts. To enhance FFT transformation stability, we address shift issues caused by repeated FFT executions. Quantitative

evaluations reveal that FLC Pooling and ASAP provide superior native robustness against spatial shifts, common corruptions and adversarial attacks while maintaining high accuracy on clean data. When integrated with FGSM AT, these methods mitigate catastrophic overfitting, enabling efficient and fast AT without the need for early stopping, which often results in under-converged networks. These findings highlight the critical role of analysing signal-processing properties in neural network components to improve native robustness.

Chapter 7

Neural Implicit Frequency Filters

Contents

In this chapter, we introduce Neural Implicit Frequency Filters (NIFF), a powerful tool for analysing the effective filter size of CNNs. Recent work in image classification models has trended towards increasing spatial context, whether through larger convolution kernels or self-attention. However, these approaches often scale poorly, with gains in accuracy coming at high computational costs. To facilitate a meaningful study of the effective filter size needed, several challenges need to be addressed: (i) we need an effective means to train models with large filters (potentially as large as the input data) without increasing the number of learnable parameters, (ii) the employed convolution operation should be a plug-and-play module that can replace conventional convolutions and allow for an efficient implementation in current frameworks, (iii) the study of filter sizes has to be decoupled from other aspects such as the network width or the number of learnable parameters, and (iv) the cost of the convolution operation itself has to remain manageable *i.e.* we can not naïvely increase the size of the convolution kernel. To address these challenges, we propose to learn the *frequency representations* of filter weights as neural implicit functions, such that the better scalability of the convolution in the frequency domain can be leveraged. Additionally, due to the implementation of the proposed neural implicit function, even large and expressive spatial filters can be parametrised by only a few

learnable weights. Notably, our analysis reveals that while these networks could learn large convolution kernels, the resulting filters remain well-localized and relatively compact in the spatial domain. We anticipate that our insights into optimised filter sizes help create more efficient and effective models in the future. Our code is available at `https://github.com/GeJulia/NIFF`.

This chapter is based on Grabinski et al. (2024). Julia Grabinski, as the first author, conducted all experiments and was the main writer.

7.1 Introduction

Recent progress in image classification, such as [Liu et al. (2022b)], builds upon observations on the behaviour of vision transformers [Dosovitskiy et al. (2021); Khan et al. (2022); Touvron et al. (2021); Vaswani et al. (2017)], which rely on the learned self-attention between large image patches and therefore allow information from large spatial contexts to be encoded. In particular, there has been a recent trend toward increasing the spatial context during encoding in CNNs, leading to improved performance accordingly, as for example in [Guo et al. (2022); Liu et al. (2023); Ding et al. (2022); Peng et al. (2017)]. Yet, model parameters and training times scale poorly with the filter size, such that the increased model accuracy often comes at significant costs if no additional model-specific tricks are applied. At the same time, it remains unclear whether there is an optimal filter size and which size of filters would be learned, could the models learn arbitrary sizes.

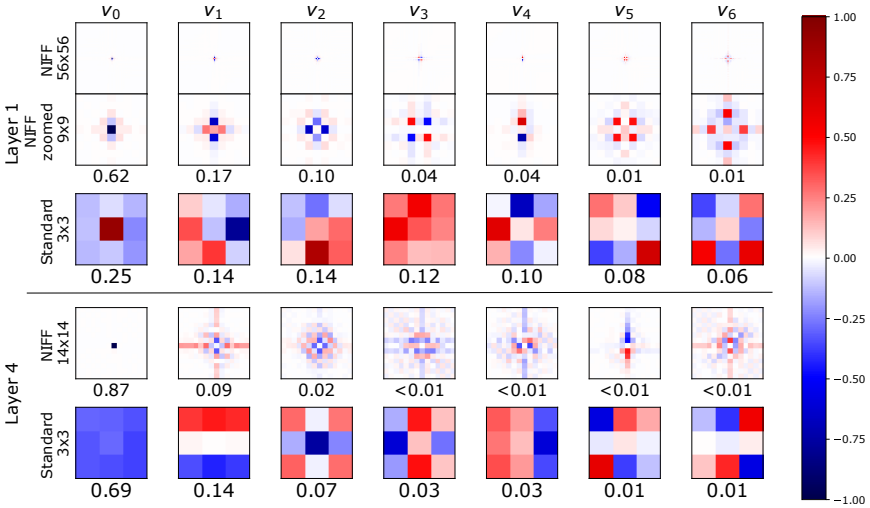


FIGURE 7.1: **ImageNet-1k Models Learn Larger Kernels than 3×3.** PCA components of learned kernel weights in the first layer of a ResNet-50 trained on ImageNet-1k: the first row shows the learned NIFF kernels transformed to the spatial domain. The second row visualizes a zoomed-in version with explained variance for each component denoted below. The third row shows the PCA and explained variance for a standard CNN with the standard kernel size of 3×3. The fourth row demonstrates that NIFF learns large, highly structured spatial filters in the fourth layer of the same network, while the baseline model is limited to small filters. This PCA analysis is one of the tools we apply to address the central question of this chapter: *How large do CNN kernels really need to be?* It reveals that networks capable of learning filter kernels as large as their feature maps still tend to learn well-localized, small kernels. However, these kernels are larger than the commonly used 3×3 kernels.

With our NIFF, we aim to provide such a study and the corresponding tool that can modularly replace the convolution operation in any CNN architecture, allowing for the efficient training of arbitrarily large convolution kernels. However, efficiently training models on standard datasets such as ImageNet [Deng et al. (2009)] with large filters (potentially as large as the input data) is non-trivial. Not only should the number of parameters remain in a range comparable to the one of the baseline models, but also, the cost of the actual convolution operation has to remain manageable. Neural implicit functions, such as previously used in [Romero et al. (2022a); Sitzmann et al. (2020)], can limit the number of learnable parameters while learning large convolution filters. Yet, their evaluation is limited to low-resolution data because of the poor scalability of the convolution operation itself, *i.e.* increasing the size of the learned convolution filters directly or implicitly is not a scalable solution. Therefore, we propose to learn filters in the Fourier domain via neural implicit functions. This has several advantages: First, the convolution operation can be executed in the Fourier domain, where it scales significantly better with the filter size. Second, due to the implementation of the neural implicit function, the number of learnable model parameters remains similar to that of the baseline model. Third, the learned filters are directly expressed in the Fourier domain *i.e.* as oriented sine and cosine waves. Thus, highly structured periodic spatial convolution kernels can be learned using small MLPs with only a few parameters. We propose **N**eural **I**mplicit **F**requency **F**ilter (NIFF),a plug-and-play convolution module, designed to replace standard CNN convolutions and facilitate efficient integration into current frameworks. The resulting neural implicit frequency CNNs are the first models to achieve results on par with the state-of-the-art on standard high-resolution image classification benchmarks while executing convolutions solely in the frequency domain. Thus, NIFFs allow us to provide an extensive analysis of the practically learned filter sizes while decoupling the filter size from other aspects, such as the network width or the number of learnable parameters. Interestingly, our analysis reveals that although the proposed networks could learn very large convolution kernels, the learned filters practically correspond to well-localized and relatively small convolution kernels when transformed from the frequency to the spatial domain.

Our contributions can be summarized as follows:

- We present a novel approach which enables efficient learning of spatially infinitely large convolutional filters. We introduce MLP-parametrised Neural Implicit Frequency Filters (NIFFs), which learn filter representations directly in the frequency domain and can be plugged into any CNN architecture.
- Empirically, we show that NIFFs facilitate a model performance on par with the baseline without any hyperparameter tuning. Hence, our proposed frequency filters allow, for the first time, to efficiently analyse filter sizes and encoded context sizes via filters that potentially have an infinite extent in the spatial domain.
- Furthermore, we analyse the spatial representations of the resulting large filters learned by various CNN architectures and show very interesting results in terms of practically employed spatial extent.
- Finally, we integrate our NIFF convolution with the FLC Pooling from Chapter 6, further advancing the transformation of CNNs into the Fourier domain.

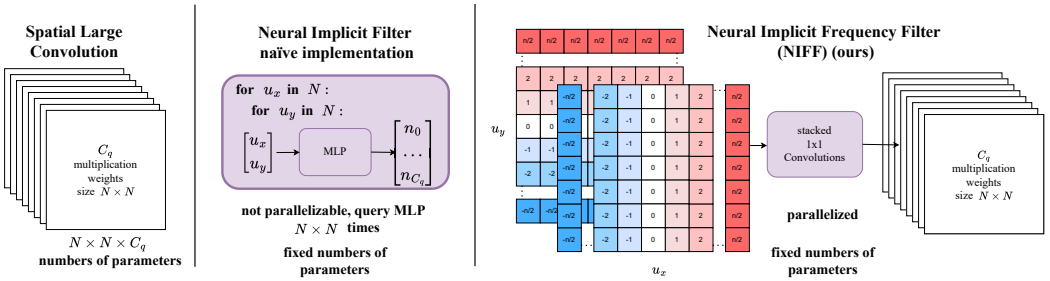


FIGURE 7.2: **Method Comparison: Standard Large Convolution vs. our NIFF.** While learning large kernels increases the number of learnable parameters quadratically (here N×N (left)), neural implicit functions use a fixed amount of learnable parameters (middle). When using a simple MLP, the input is a 2D vector containing the u_x and u_y coordinates of the desired filter position. Our NIFF (right) efficiently implements the MLP using several 1×1 convolutions, starting with an input channel size of two, which encodes the u_x and u_y directions. This eliminates the need to iterate over each coordinate separately. Subsequently, we include hidden layers and activation functions to facilitate learning. In the final layer, the number of output channels is set to match the desired number of element-wise multiplication weights.

7.2 Method

For the development of our NIFF, we use several concepts from different backgrounds. We leverage neural implicit functions in a novel setting: We learn neural implicit representations of the spatial frequencies of large convolution kernels. This allows us to employ the Convolution theorem from signal processing [Forsyth & Ponce (2003)] and conduct convolutions with large kernels efficiently in the frequency domain via point-wise multiplications.

Properties of Convolutions in the Frequency Domain. According to the Convolution theorem [Forsyth & Ponce (2003)], a circular convolution, denoted by $\circledast$, between a signal g and filter k in the spatial domain can be equivalently represented by a point-wise multiplication, denoted by $\odot$, of these two signals in the frequency domain, for example by computing their Fast Fourier Transform (FFT), denoted by the transform $\mathcal{F}(.)$ and then, after point-wise multiplication, their inverse FFT $\mathcal{F}^{-1}(.)$:

$$g \circledast k = \mathcal{F}^{-1}(\mathcal{F}(g) \odot \mathcal{F}(k)) \tag{7.1}$$

While this equivalence has been less relevant for the relatively small convolution kernels employed in traditional CNNs (typically 3×3 [He et al. (2016a); Simonyan & Zisserman (2015)] or at most 7×7 [Liu et al. (2022b)]), it becomes highly relevant as the filter size increases to a maximum: The convolution operation in the spatial domain is in $O(M^2N^2)$ for a discrete 2D signal g with $N \times N$ samples and filters k of size $M \times M$, *i.e.* $O(N^4)$ when discrete filters k have the same size as the signal g. In contrast, the computation is in $O(N^2\log(N))$ when executed using FFT [Cooley & Tukey (1965)] and point-wise multiplication according to Equation (7.1). The proof for the FFT according to [Cooley & Tukey (1965)] is given in Section 2.4.2.

Thus, for efficient filter learning, we assume that our input signal, *i.e.* our input image or feature map, is given in the frequency domain. There, we can directly learn the element-wise multiplication weights m corresponding to $\mathcal{F}(k)$ in Equation (7.1) and thereby predict infinitely large spatial kernels. These element-wise multiplication weights, m, perform a circular convolution, which can be interpreted as an

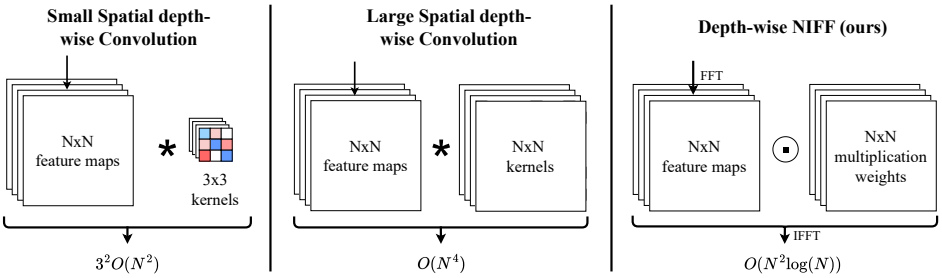


FIGURE 7.3: **Concept of our NIFF Convolutions.** Concept of our NIFF convolutions and the computational complexity for each operation, illustrated using the example of a depth-wise convolution. Left: In the standard depth-wise convolution, the number of kernels matches the number of feature maps. Each kernel is convolved with its corresponding feature map. Middle: A large convolution where the kernels are as large as the feature maps. Right: The NIFF convolution, which applies a simple pointwise multiplication between the FFT-transformed feature maps and the learned kernel weights via our NIFF. The updated feature maps are then transformed back into the spatial domain using the IFFT.

infinite convolution due to the periodic nature of the frequency domain representation. Practically, this means that representing a signal (*e.g.*, an image or feature map) in the frequency domain and transforming it back to the spatial domain implicitly assumes the signal is periodic and infinitely repeating. For many images, such boundary conditions can make sense since the image horizon line is usually horizontally aligned.

Practically, the kernels applied in the frequency domain are bandlimited to the highest spatial frequency that can be represented in the feature map. However, since higher frequencies can not be encoded in the discrete input signal by definition, this is no practical limitation. In the frequency domain, we can thus efficiently apply convolutions with filters of standard sizes, such as 224×244 (for ImageNet) or 32×32 (for CIFAR-10), using learned spatial frequencies solely limited by the resolution of the input feature map.

7.2.1 Neural Implicit Frequency Filters

Images are typically stored as pixel-wise discrete values. Similarly, the filter weights of CNNs are usually learned and stored as discrete values, *i.e.* a 3×3 filter has 9 parameters to be learned and stored, a 7×7 filter as in ConvNeXt has 49 parameters and a filter as large as the feature map would require *e.g.* 224×224 (50176) parameters for ImageNet-sized network input. In this case, it is not affordable to directly learn these filter weights, neither in the spatial nor in the frequency domain. To address this issue, we propose to parametrise filters by neural implicit functions instead of learning the kernels directly. This is particularly beneficial since we can directly learn the neural implicit filter representations in the frequency domain. Figure 7.2 depicts the benefit of neural implicit functions for large kernel sizes.

Thus, formally, we learn a function F parametrised by Φ that takes as input the spatial frequency (u_x, u_y) whose filter value it predicts,

$$F_\Phi : \mathbb{R}^2 \mapsto \mathbb{C}^C, m(u_x, u_y) := F_\Phi(u_x, u_y) \tag{7.2}$$

where C is the number of filter channels and the complex-valued $m(u_x, u_y)$ in dimension c is the c-th filter value in the frequency domain to be multiplied with $\mathcal{F}(g)(u_x, u_y)$ for feature map g. Specifically, with the implicit function F_Φ, we parametrise

the weights with which the feature maps are multiplied in the frequency domain based on equation 7.1 by point-wise multiplication. The number of MLP output channels is equivalent to the number of channels C for the convolution, and its hidden dimensions determine the expressivity of each learned filter, which we term *Neural Implicit Frequency Filter (NIFF)*. In practice, the complex and real valued parts of NIFFs are independently parametrised.

Efficient Parametrisation of NIFFs. Neural implicit functions allow parametrising large filters with only a few MLP model weights, however, their direct implementation would be highly inefficient. Therefore, we resume to a trick that allows efficient training and inference using standard neural network building blocks. Specifically, we arrange the input to the MLP, *i.e.* the discrete spatial frequencies (u_x, u_y) for which we need to retrieve filter values, in 2D arrays that formally resemble feature maps in CNNs but are fixed for all layers. Thus, the MLP takes one input matrix encoding the x coordinates and one encoding the y coordinates, as shown in Figure 7.3. Then, the MLP can be equivalently and efficiently computed using stacked 1×1 convolutions, where the first 1×1 convolution has input depth two for the two coordinates, and the output layer 1×1 convolution has C output dimensions.

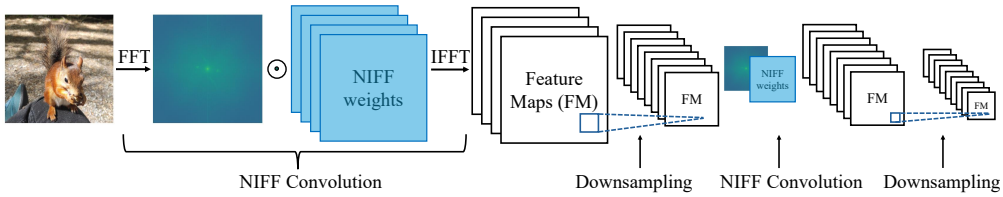


FIGURE 7.4: **NIFF, integration in CNN.** Abstract integration of our NIFF convolution for feature extraction in a CNN, similar to Figure 1.2 (c) from Chapter 1.

7.2.2 Common CNN Building Blocks using NIFF

As shown in Figure 7.4, our NIFF replaces the convolution operation within the CNN feature extraction. Since CNNs utilize various convolution types, we will discuss each variant and its corresponding NIFF replacement in the following. Well-established models like ResNet [He et al. (2016a)] use full convolutions, while more recent architectures employ depth-wise and 1×1 convolutions separately [Liu et al. (2022b)]. Our neural implicit frequency filters can be implemented for all these cases. However, operations that include downsampling by a stride of two are kept as original spatial convolution. We ablate in Section 7.3.6 on the combination of our NIFF with FLC Pooling, introduced in Section 6.2.2, to perform downsampling with our NIFF in the frequency domain. In the following, we describe how commonly used convolution types are implemented using NIFF.

Depth-Wise Convolution. The NIFF module for the depth-wise convolution is as follows: First, we transform the feature maps into the frequency domain via FFT. The learned filters are then applied through element-wise multiplications with the feature maps. Subsequently, the feature maps are transformed back to the spatial domain via inverse FFT if the subsequent operation occurs in the spatial domain. The entire process is visualized in Figure 7.3, right. As discussed above, this process allows to train models with large *circular* convolutions in a scalable way. An ablation on circular versus linear convolutions in the frequency domain is given in Table 7.3.

Full Convolution. To perform a full 2D convolution (*2DConv*) using NIFF with C_p input channels and C_q output channels, the convolved feature maps are summed according to

$$\text{2DConv}(g_{C_p}, k_{C_p,C_q}) = \sum_{c}^{C_p} g_c \circledast k_{c,C_q} = g_{C_q}. \tag{7.3}$$

Conveniently, a summation in the spatial domain is equivalent to a summation in the frequency domain and can be performed right away.

$$g + k = \mathcal{F}^{-1}(\mathcal{F}(g) + \mathcal{F}(k)) \tag{7.4}$$

The full convolution in the frequency domain can be implemented by first predicting the frequency representation of k_{C_p,C_q} directly using NIFF, *i.e.* for each output channel. Then, all input channels are element-wise multiplied with the filter weights and summed up in 2DConv_{NIFF}:

$$\sum_{c}^{C_p} g_c \circledast k_{c,C_q} = \mathcal{F}^{-1}\left(\sum_{c}^{C_p} \mathcal{F}(g_c) \odot \mathcal{F}(k_{c,C_q})\right) \tag{7.5}$$

The NIFF needs to output $C_p \times C_q$ multiplication weights instead of only C_q at its last layer, leading to a significant increase of learnable parameters for our NIFF. Thus, for efficiency reasons, we decompose the full convolution into a depth-wise convolution followed by a 1×1 convolution in practice. The transformation into the frequency domain is applied before the depth-wise convolution, where the backward transformation into the spatial domain is applied after the 1×1 convolution. While not equivalent, the amount of learnable parameters decreases significantly with this approach, and the resulting models achieve similar or better performance in practice. An ablation on full convolutions versus depth-wise separable convolutions is provided in Tables 7.1 and 7.2.

1×1 **Convolution.** To perform a 1×1 convolution, we transform the input into the frequency domain via FFT. Afterwards, we apply a linear layer with channel input neurons and desired output dimension output neurons on the channel dimension. Finally, we transform back into the spatial domain via inverse FFT. While spatial 1×1 convolutions only combine spatial information in one location, our 1×1 in the frequency space is able to combine and learn important information globally.

Other operations, such as downsampling, normalization, and non-linear activation, are applied in the spatial domain so that the resulting models are as close as possible to their baselines while enabling infinite-sized convolutions. Yet, we ablate in Section 7.3.6 on the effect of transforming more modules in the Fourier domain.

7.3 Experiments

7.3.1 Training Details

ImageNet. The training parameters and data preprocessing are kept the same for ImageNet-1k and ImageNet-100. Each network architecture is trained with the general training pipeline and data preprocessing provided by Liu et al. (2022b). The training parameters for each individual network are taken from the original papers provided by the authors ResNet [He et al. (2016a)], DenseNet-121 [Huang et al. (2017b)] ConvNeXt-tiny [Liu et al. (2022b)] and MobileNet-v2 [Sandler et al. (2018)].

CIFAR-10. For CIFAR-10, we used the same training parameter for all networks. We trained each network for 150 epochs with a batch size of 256 and a cosine learning rate schedule with a learning rate of 0.02. We set the momentum to 0.9 and weight decay to 0.002. The loss is calculated via LabelSmoothing loss with label smoothing of 0.1 and as an optimizer, we use SGD.

For data preprocessing, we used zero padding by four and cropping back to 32×32 and horizontal flip, as well as normalizing with mean and standard deviation.

Computing Infrastructure. For training, we use NVIDIA Titan V and NVIDIA A100 GPUs. For the training on low-resolution data (CIFAR-10), we used one NVIDIA Titan V. Depending on the model architecture and the convolution used (baseline, NIFF or large convolution) the training took between 15 minutes and 90 minutes. For the training on high-resolution data (ImageNet-100 and 1k), we used four NVIDIA A100 in parallel. The training time depends on the used model architecture and varies if we use the full ImageNet-1k dataset or only ImageNet-100. The training time for ImageNet-1k varies between one day and one hour and ten days and nine hours for ImageNet-100 between 93 minutes and one day eight hours depending on the model architecture and the number of epochs for training.

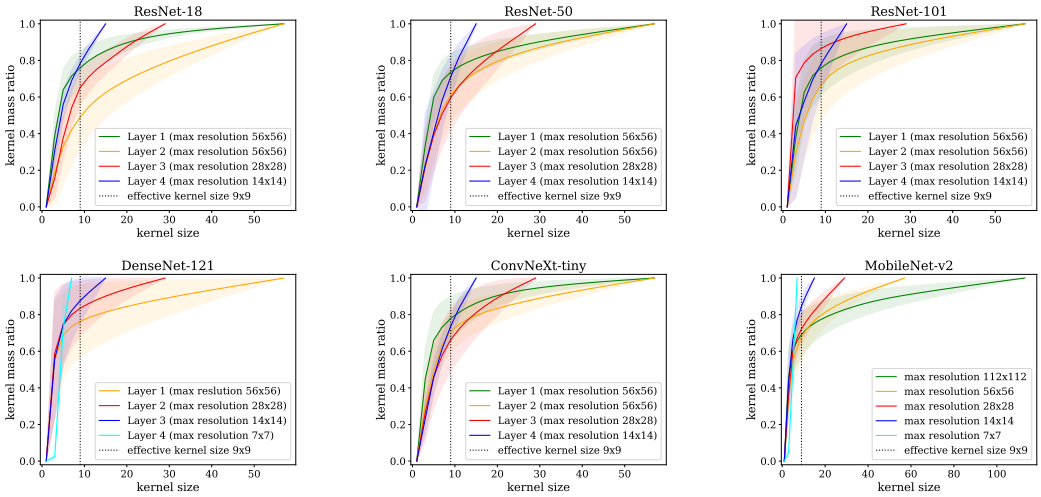


FIGURE 7.5: **Kernel Mass Ratio on ImageNet-1k.** Effective kernel size evaluation on ImageNet-1k with out KMR. We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. For ResNet-18, ResNet-50, ResNet-101, DenseNet-121 and ConvNeXt-tiny each layer encodes one resolution. Thus, these network's layers could be summarised (Layer 1 encoding 56×56, Layer 2 encoding 28×28, Layer 3 encoding 14×14 and Layer 4 encoding 7×7). However, for MobileNet-v2 the resolution is downsampled within a layer.

7.3.2 How Large Do Spatial Kernels Really Need To Be?

To answer this question, we quantitatively analyse how large the spatial kernels really tend to be by introducing a new metric, called kernel mass ratio, short KMR. To do so, we transform the learned filters into the spatial domain and plot the relative density of each spatial kernel k, *i.e.* the ratio of the kernel mass that is contained within centered, smaller sized, squared kernels. The full kernel has the same width

and height as the feature map (FM) it is convolved to.

$$\text{KMR}(\text{width}, \text{height}) = \frac{\sum_{w=1}^{\text{width}} \sum_{h=1}^{\text{height}} k\left(\text{c} - \lfloor \frac{\text{width}}{2} \rfloor + w, \text{c} - \lfloor \frac{\text{height}}{2} \rfloor + h\right)}{\sum_{w=1}^{\text{FMwidth}} \sum_{h=1}^{\text{FMheight}} (k(w,h))}, \tag{7.6}$$

where c is the center of the full kernel k in the spatial domain.

In Figure 7.5, we report the kernel mass ratio for different networks trained on ImageNet-1k. We observe that all networks mostly contain well-localized kernels that are significantly smaller than possible. Yet, the first layer in DenseNet-121 and the second and third layer in ResNet-18 also contain larger kernels that make use of the possible kernel size up to 56×56 and 28×28, respectively. Surprisingly, the smaller ResNet-18 model learns larger kernels than the ResNet-50 or ResNet-101 in the second layer. For MobileNet-v2, the spatial kernels are predominantly well-localized and small. However, at the maximal resolution of 112×112 some kernels are quite large, at least 56×56 (15%), indicating that MobileNet-v2, which uses small 3×3 kernels, could benefit from larger kernels. These findings are in line with Romero et al. (2022a), who investigated the spatial kernel size of CNNs in the spatial domain. Similar results on ImageNet-100 are reported in the Appendix C.1 in Figure C.1.

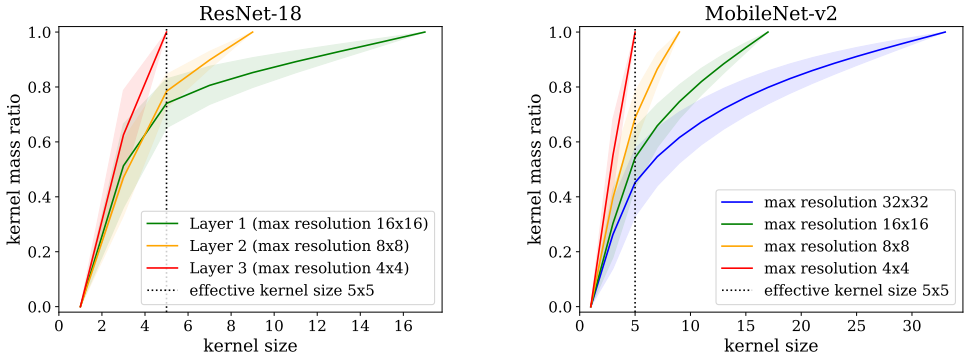


FIGURE 7.6: **Kernel Mass Ratio on CIFAR-10.** Effective kernel size evaluation on CIFAR-10 with out KMR. We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. For ResNet-18, each layer encodes one resolution. Thus, the layers could be summarised (Layer 1 encoding 16×16, Layer 2 8×8 and Layer 3 4×4). For MobileNet-v2 the resolution is downsampled within a layer.

In Figure 7.6, we evaluate the spatial KMR for ResNet-18 and MobileNet-v2 trained on CIFAR-10; an overview with all models is shown in Figure C.2 in Appendix C.1 and demonstrates similar results. All models tend to learn in all layers well-localized, small kernels, similar to networks trained on high-resolution data. For CIFAR-10, the learned kernels barely exceed the size of 5×5 for ResNet-18. However, the first block in MobileNet-v2 seems to make use of larger kernels, as over 50% of the kernels are larger than 5×5.

Non-Square Kernels. Following [Romero et al. (2022a)], we analyse our kernels beyond square shapes, which are typically applied in CNNs, but also rectangular shape kernels. Thus, we fit a 2D Gaussian to the kernels learned using our NIFF and compare the variance given by σ_x and σ_y in the x- and y- direction. In detail, we evaluate the ratio between σ_x and σ_y of the Gaussian. To aggregate over all kernels within one layer, we plot the mean and standard deviation of these ratios in Figure 7.7 for ImageNet-1k and ResNet-50, MobileNet-v2 and ConvNeXt-tiny. We report all models on ImageNet-1k and ImageNet-100 in the Appendix C.2 Figures C.3 and C.4, respectively. The mean over all kernels for all models within a layer is near to a ratio of one, indicating that most kernels exhibit square shapes. For some layers (the last layer of ResNet-50 and ResNet-101 on ImageNet-1k and ResNet-18 and ResNet-50 on ImageNet-100), the variance is quite high, indicating that σ_x and σ_y differ and non-square, rectangle kernels are learned. Note that the learned kernels by Romero et al. (2022a) are parametrised by a Siren [Sitzmann et al. (2020)], leading to more wave-like, smooth kernels. In contrast, we learn the kernels in the frequency domain which could be wave-like, but are mostly not wave-like as shown in Figures C.24 and C.23 which depict a random selection of the learned spatial kernels in the Appendix C.3. Therefore, the measured standard deviations in x and y direction should not be understood as a kernel mask, as argued in [Romero et al. (2022a)]. They merely indicate the rough spatial distribution of filter weights.

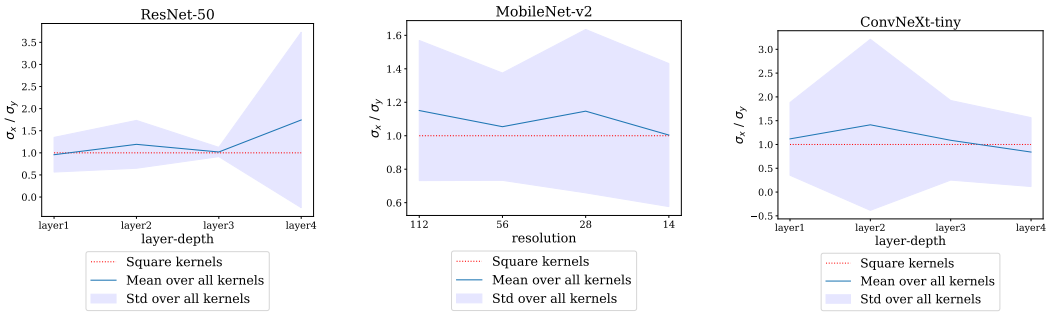


FIGURE 7.7: **Analysis of Non-Square Kernel Shapes for our NIFF on ImageNet-1k.** We analyse NIFF ResNet-50, MobileNet-v and ConvNeXt-tiny of non-square kernel shapes on ImageNet-1k. We compare the variance σ_x and σ_y in x- and y-direction of a Gaussian fitted onto our learned spatial weights. The red dashed line indicates square-shaped kernels as the variance σ_x and σ_y are equal.

7.3.3 Quantitative Results

We evaluate a variety of classification CNNs with our proposed NIFF. Overall, we achieve accuracies on par with the respective baselines. For high-resolution data, our NIFF CNNs perform slightly better than the baseline models, while the large kernels are apparently less beneficial for low-resolution data, especially in deeper layers where the feature map sizes are very small. Further, we evaluate the performance of NIFF when combined with other modules applied in the frequency domain.

ImageNet. For high-resolution datasets such as ImageNet-1k and ImageNet-100, NIFF CNNs demonstrate performance comparable to baseline models. These results are presented in Table 7.1. For models which originally employ full 2D convolutions, we also ablate on the effect of replacing these with depth-wise separated

TABLE 7.1: **NIFF performance evaluation on ImageNet-1k and ImageNet-100.** Evaluation of top 1 and 5 test accuracy on ImageNet-1k and ImageNet-100 for different network architectures. We used the standard training parameter for each architecture and the advanced data augmentation from Liu et al. (2022b) for all architectures. Additionally, we add the numbers reported by Pytorch for the *timm* baseline models on ImageNet-1k for each network. Our ablation on linear convolutions is given in Table 7.3.

Model		Method	ImageNet-1k			ImageNet-100		
			# Params ↓	Acc@1 ↑	Acc@5↑	# Params ↓	Acc@1 ↑	Acc@5 ↑
ResNet	18	Pytorch	11.689.512	69.76	89.08	11.227.812	-	-
		Baseline	11.689.512	**72.38**	**90.70**	11.227.812	**87.52**	**97.50**
		NIFF 2D conv (ours)	21.127.320	71.00	89.95	20.665.620	86.42	97.08
		separated conv	3.327.400	69.12	88.84	2.865.700	86.52	97.20
		NIFF (ours)	3.660.360	70.75	90.01	3.198.660	86.52	97.14
ResNet	50	Pytorch	25.557.032	76.13	92.86	23.712.932	-	-
		Baseline	25.557.032	79.13	94.43	23.712.932	89.88	98.22
		NIFF 2D conv (ours)	33.778.328	78.65	94.25	31.934.228	**90.08**	98.06
		separated conv	18.275.688	77.76	93.72	16.431.588	89.78	98.18
		NIFF (ours)	18.605.000	**79.65**	**94.80**	16.760.900	89.98	**98.44**
ResNet	101	Pytorch	44.549.160	77.37	93.55	42705.060	-	-
		Baseline	44.549.160	**80.63**	95.11	42.705.060	**90.54**	98.14
		separated conv	28.394.088	79.53	94.64	26.549.988	90.20	98.36
		NIFF (ours)	29.187.432	80.26	**95.23**	27.343.332	**90.54**	**98.38**
DenseNet	121	Pytorch	7.978.856	**76.65**	92.17	7.978.856	-	-
		Baseline	7.978.856	75.11	**92.50**	7.056.356	90.06	98.20
		NIFF 2D conv (ours)	10.119.752	73.96	91.82	9.197.252	89.66	98.32
		separated conv	6.145.128	69.80	89.02	5.222.628	89.94	98.08
		NIFF (ours)	6.159.512	74.58	92.33	5.237.012	**90.24**	98.18
Conv	NeXt-t	Pytorch	28.589.128	**82.52**	**96.15**	28.589.128	-	-
		Baseline	28.589.128	-	-	27.897.028	91.70	98.32
		NIFF (ours)	28.890.664	81.83	95.80	28.231.684	**92.00**	**98.42**
Mobile	Net-v2	Pytorch	3.504.872	**71.88**	90.29	3.504.872	-	-
		Baseline	3.504.872	70.03	89.54	2.351.972	84.06	96.52
		NIFF (ours)	3.512.560	71.53	**90.50**	2.359.660	**85.46**	**96.70**

convolutions in spatial domain with the original filter size. In line with the finding *e.g.* in [Liu et al. (2022b)], the differences are rather small, while the amount of parameters decreases significantly as depth-wise separable convolutions are applied. According to our results, this is equally true for both, convolutions executed in the spatial and in the frequency domain. For comparability, we simply used the baseline hyperparameters reported for each of these models in the respective papers, *i.e.* we achieve results on par with the respective baselines without hyperparameter optimization. For different ResNet architectures, we even achieve improvements on both high-resolution datasets.

However, for small networks like MobileNet-v2, we observe that both the baseline model and the NIFF version have comparably low accuracy. For MobileNet-v2, the training pipeline is usually highly optimized for best performance. The data augmentation scheme from [Liu et al. (2022b)], that we employ for all trainings to achieve comparable results, does not seem to have a beneficial effect here.

Similar to ImageNet-1k, ImageNet-100 exhibits a trend where NIFF provides substantial gains for larger models but less so for lightweight models.

CIFAR-10. Although NIFF CNNs can perform on par with the respective baseline on high-resolution datasets, their performance is limited on low-resolution datasets. Table 7.2 presents the results on CIFAR-10 with different architectures. Unfortunately, our NIFF CNNs lose around 1 to 3 % points compared to the baseline models.

TABLE 7.2: **NIFF performance evaluation on CIFAR-10.** Performance evaluation of different networks trained on CIFAR-10, including the number of learnable hyperparameters for each network. To ensure comparability across all models and architectural variations, we applied the same training schedule for each. Results show that NIFF CNNs achieve slightly better performance when using a ConvNeXt backbone [Liu et al. (2022b)]. However, for other architectures, their performance is slightly lower.

Model	Method	# Params ↓	Acc@1 ↑
ConvNeXt-tiny	Baseline	6.376.466	90.37
	NIFF (ours)	6.305.746	**91.48**
ResNet-18	Baseline	11.173.962	**92.74**
	NIFF 2D conv (ours)	20.613.546	92.66
	separated conv	2.810.341	90.18
	NIFF (ours)	1.932.432	90.63
ResNet-50	Baseline	23.520.842	**93.75**
	NIFF 2D conv (ours)	31.743.914	93.39
	separated conv	16.237.989	92.13
	NIFF (ours)	15.491.600	93.11
DenseNet-121	Baseline	6.956.426	**93.93**
	NIFF 2D conv (ours)	9.099.098	92.47
	separated conv	5.121.189	92.00
	NIFF (ours)	5.555.856	92.49
MobileNet-v2	Baseline	2.236.682	**94.51**
	NIFF (ours)	2.593.760	94.03
MobileNet-v3	Baseline	1.528.106	86.28
	NIFF (ours)	1.526.466	**86.60**

This can be addressed to our previous observation: Networks trained on CIFAR-10 exploit a limited portion of NIFF's available kernel size, effectively using kernels equivalent to the baseline model's 3×3 kernels.

This slight drop in performance might be due to the low spatial resolution of images and feature maps. Particularly in deeper layers, the potential benefit from large convolutions is therefore limited. Further, we want to emphasize that NIFF is not conceived to improve over baseline methods (we are, of course, glad to observe this tendency for high-resolution data) but to facilitate the evaluation of effective kernel sizes in CNNs.

7.3.4 Filter Analysis

In this section, we visualize the spatial kernels learned by our NIFF. We do so by transforming the learned multiplication weights via IFFT into the spatial domain. Afterwards, we apply Principle Component Analysis (PCA) per layer to evaluate the predominant structure of the learned spatial kernels. For each layer, we separately plot the six most important eigenvectors and zoom in to visualize the 9×9 center. The full visualizations without center cropping are present in Figure C.13 in the Appendix C.3. Additionally, an overview of a random selection of original spatial kernels is given in Figures C.24 and C.23 in the Appendix C.3.

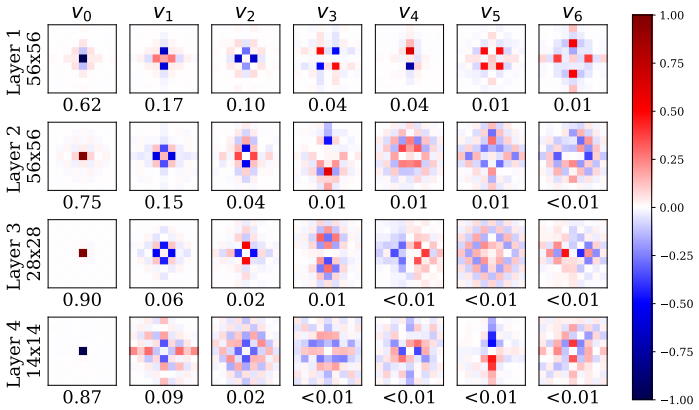


FIGURE 7.8: **Spatial Filter Visualizations for our NIFF ResNet-50 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-1k zoomed to 9×9 (the full size filters are presented in Figure C.13 in the Appendix C.3). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

High-Resolution Data. Our results indicate that the networks, although they could learn infinitely large kernels, learn well-localized, quite small (3×3 up to 9×9) kernels, especially in early layers. The spatial kernel size is different for low- and high-resolution data. Figure 7.8 visualizes the eigenvectors of the spatial kernel

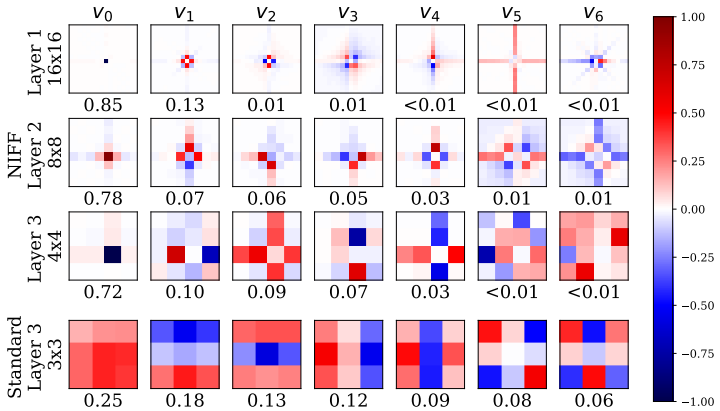


FIGURE 7.9: **Spatial Filter Visualizations for our NIFF ResNet-18 on CIFAR-10.** PCA basis and explained variance for each basis vector (below) of all spatial filters learned by NIFF for each layer as well as the learned filters for the third layer of a standard ResNet-18 trained on CIFAR-10. On the left, the layer and its filter size are given. Most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

weights for each layer in our NIFF CNN for ResNet-50 trained on ImageNet-1k. All layers dominantly learn filters with well-localized, small kernel sizes. Especially in the later layers, most variance is explained by simple 1×1 kernels, whereas the first-layer kernels have the most variance explained by simple 5×5 or 7×7 kernels. Hence, the networks learn larger kernels than the standard 3×3 kernels used in a ResNet. However, the full potential of possible sizes of 56×56 and more is not used. Similar results are shown in Figure 7.1 as well as for more architectures on ImageNet-1k in the Appendix C.3 (Figures C.5, C.6 and C.7) and ImageNet-100 (Figures C.8, C.9, C.10 and C.11); all networks learn well-localized, small kernels, even

though they could learn much larger ones.
Some exceptions can be observed for the 4th layer in Figures 7.1 and 7.8, where the eigenvector with the second-highest explained variance still uses a larger extent of the kernel. In combination with the downsampling applied prior to the fourth layer, this indicates that the network tries to keep a large spatial context.

Low-Resolution Data. The same effect can be observed for networks trained on CIFAR-10. Figure 7.9 depicts the PCAs of the spatial kernel weights for each layer in our NIFF CNN for a ResNet-18 trained on CIFAR-10. There, the dominant basis vectors for the first layer learned by our NIFF kernels do not exceed a size of 3×3 in the spatial domain. However, in the second layer, a few kernels (19%) are larger than 3×3, mostly 7×7. In the last layer, the kernels again do not exceed the standard size of 3×3. Similar results on MobileNet-v2 are presented in Figure C.12 in the Appendix C.3.

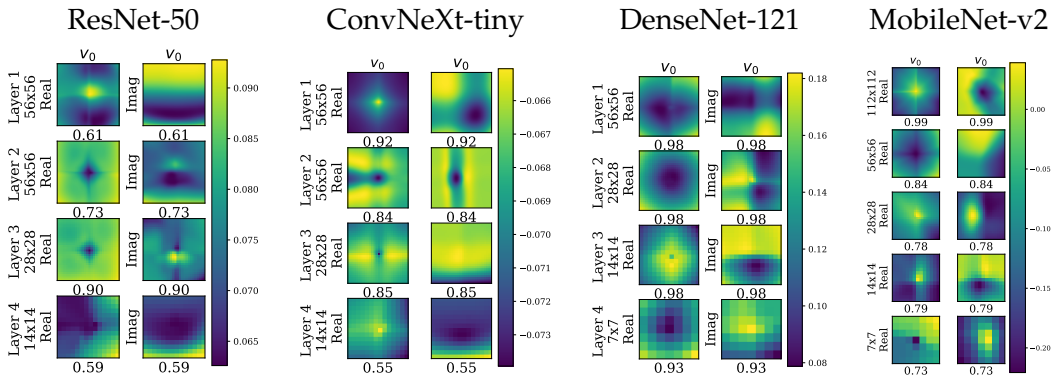


FIGURE 7.10: **Center-shifted Frequency Filter Visualizations for our NIFF models on ImageNet-1k.** First PCA basis and explained variance (below) of the element-wise multiplication weights for the real and imaginary part in the center-shifted frequency domain for each layer of a ResNet-50, ConvNeXt-tiny, DenseNet-121 and MobileNet-v2 trained on ImageNet-1k. For each network the maximal filter size is denoted left for the corresponding layer. The left image represents the real part, the right the imaginary part.

NIFF Multiplication Weights. Moreover, we analyse the learned element-wise multiplication weights for the real and imaginary parts of different models trained on ImageNet-1k in the frequency domain. We present the most important information contained in the first principle component for each network in Figure 7.10. In Appendix C.3, we include Figures C.25, C.26, C.28 and C.27 which show the full PCA per layer for the learned element-wise multiplication weights for ResNet-50, DenseNet-121, ConvNeXt-tiny and MobileNet-V2 respectively. For ResNet-50 and ConvNeXt-tiny, it seems as if the networks focus in the first layer on the low- and middle-frequency spectra and in the later layers more on the high-frequency spectra. The multiplication weights learned for MobileNet-V2 focus in the first layer on low-frequency information in the second layer on high-frequency information and in the third layer again on low-frequency information. The DenseNet-121 learns high-frequency information prior in the first two layers and low-frequency information predominately in the later, third layer. Hence, a general claim for different models and their learned multiplication weights in the frequency domain can not be derived from our empirical analysis. Still, for all networks, the imaginary part

seems to be less important for these networks and thus, the learned structures are less complex. This might be owed to the fact that with increased sparsity through the activation function in the network, the network favours cosine structures (structures with a peak in the center) over sine structures.

7.3.5 Circular vs. Linear Convolution

TABLE 7.3: **NIFF Performance Approximating Linear Convolutions on ImageNet-100.** Evaluation of top 1 and top 5 accuracies on ImageNet-100 for networks learning filters with our NIFF, but afterwards, these filters are padded in the spatial domain and transformed back into the frequency domain to mimic linear convolutions. All ResNet architectures and DenseNet-121 were trained with separate depth-wise and 1x1 convolutions for efficiency reasons. All models using finite linear convolutions performed significantly worse than the baseline and our NIFF, which applies circular convolutions. This observation is consistent with low-resolution data like CIFAR-10.

Model	Name	Acc@1 ↑	Acc@5 ↑
ConvNeXt-tiny	Baseline	91.70	98.32
	NIFF (ours)	**92.00**	**98.42**
	NIFF linear	83.00	95.36
ResNet-18	separated conv	86.52	97.20
	NIFF (ours)	86.52	97.14
	NIFF linear	81.90	95.42
ResNet-50	separated conv	89.78	98.18
	NIFF (ours)	89.98	98.44
	NIFF linear	86.76	97.16
ResNet-101	separated conv	90.20	98.36
	NIFF (ours)	90.54	**98.38**
	NIFF linear	86.70	97.06
DesNet-121	separated conv	89.94	98.08
	NIFF (ours)	**90.24**	98.18
	NIFF linear	81.40	95.52
MobileNet-v2	Baseline	84.06	96.52
	NIFF (ours)	**85.46**	**96.70**
	NIFF linear	73.90	93.16

Our NIFF, as proposed, performs a circular convolution, which allows us to directly apply the Convolution theorem and execute the spatial convolution as multiplication in the frequency domain. However, standard convolutions in CNNs are finite linear convolutions. A circular convolution can mimic a linear convolution when zero-padding a signal with length M and a kernel with length K to length $L \leq M + K - 1$ [Winograd (1978)]. Thus, to ablate on circular versus finite linear convolutions, the input feature maps with size $N \times N$ are zero-padded to $2N \times 2N$. For both the linear and the circular case, NIFF learns filters with the original size $N \times N$ of the feature map. To mimic linear filters, the learned filters by our NIFF are transformed into the spatial domain and zero-padded similarly to the input feature maps to $2N \times 2N$. Afterwards, they are transformed back into the frequency domain

TABLE 7.4: **NIFF Performance Approximating Linear Convolutions on CIFAR-10.** Evaluation of top 1 on CIFAR-10 for networks learning filters with our NIFF but afterwards those are padded in the spatial domain and transformed back into the frequency domain to mimic linear, non-circular convolutions. All models using finite linear convolutions perform significantly worse than the baseline and our NIFF. This observation is consistent with high-resolution data like ImageNet-100.

Model	Method	Acc@1 ↑
ConvNeXt-tiny	Baseline	90.37
	NIFF (ours)	**91.48**
	NIFF linear	84.30
ResNet-18	Baseline	**92.74**
	NIFF full (ours)	92.66
	NIFF full linear	85.10
	separated conv	90.18
	NIFF (ours)	90.63
	NIFF linear	83.11
ResNet-50	Baseline	**93.75**
	NIFF full (ours)	93.39
	NIFF full linear	88.22
	separated conv	92.13
	NIFF (ours)	93.11
	NIFF linear	87.54
DenseNet-121	Baseline	**93.93**
	NIFF full (ours)	92.47
	NIFF full linear	85.73
	separated conv	92.00
	NIFF (ours)	92.49
	NIFF linear	79.95
MobileNet-v2	Baseline	**94.51**
	NIFF (ours)	94.03
	NIFF linear	93.21

and the point-wise multiplication is executed. Note that this is not efficient and just serves the academic purpose of verifying whether any accuracy is lost when replacing linear convolutions with circular ones in our approach. However, the resulting networks experience a performance drop compared to the baseline and our NIFF, as shown in Tables 7.3 and 7.4. We hypothesize that this drop in performance results from the enforcement of really large kernels. The additional padding mimics linear finite convolutions that are as big as the feature maps. Related work has shown that larger context can improve model performance [Ding et al. (2022); Liu et al. (2023)]. Still, there is a limit to which extent this holds as with large kernels artifacts may arise [Tomen & van Gemert (2021)]. Thus, enforcing kernels as large as the feature maps seems to be not beneficial, as shown by our quantitative results. Another explanation for the drop in accuracy could be the introduction of sinc interpolation artifacts into the padded and transformed feature maps and kernels. The padding

is formally a point-wise multiplication with a rectangular function in the spatial domain. Thus, sinc-interpolation artifacts in the frequency domain can arise. Figure 7.12 demonstrates that the learned spatial kernels are larger than the learned kernel when we apply a circular convolution with our NIFF. While the first and second layers still learn relatively small filters compared to the actual size they could learn, the third and fourth layers make use of the larger kernels. Since these results come with a significant drop in accuracy, we should, however, be careful when interpreting them. We provide the PCA analysis without zoom in the Appendix C.3 in Figure C.22.

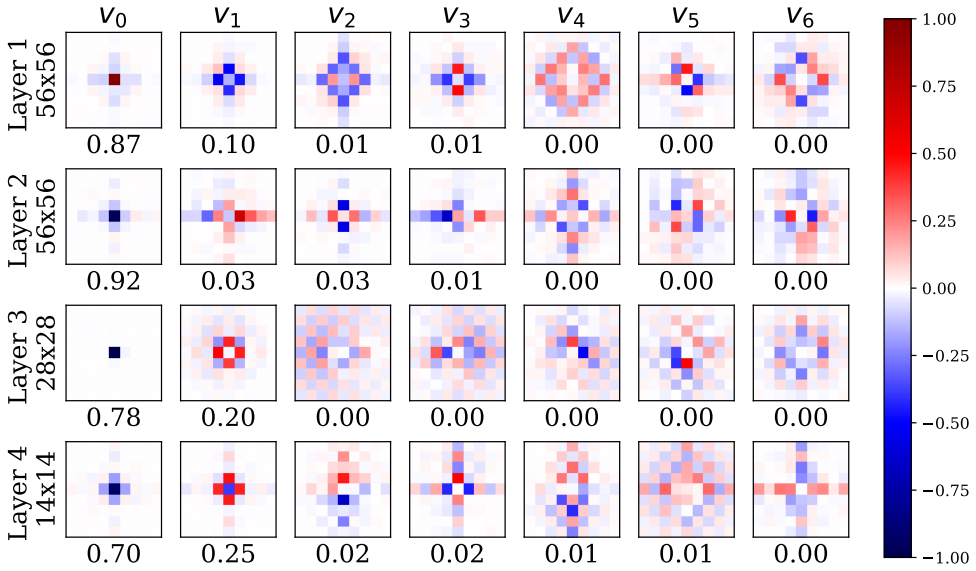


FIGURE 7.11: **Spatial Filter Visualizations for our NIFF ResNet-18 with Additional Padding on ImageNet-100.** PCA analysis of the learned kernels in the spatial domain of a ResNet-18 with additional zero padding before our NIFF trained on ImageNet-100. We plot for each kernel the zoomed-in (9×9) version for better visibility the original PCA analysis is presented in Figure C.21 in the Appendix C.3. Still, most kernels exhibit well-localized, small spatial kernels.

Ablation on Padding. We further evaluate our NIFF when the feature maps are padded with different kinds of padding methods. Due to the padding of the feature maps and the cropping after the application of our NIFF, possible artifacts can be mitigated. The padding is applied around the feature maps before transforming them into the frequency domain. The padding size is as large as the original feature map. After the application of our NIFF, the feature maps are transformed back into the spatial domain and cropped to their original size. The resulting networks experience similar performance as our baseline NIFF as presented in Table 7.5. Figures 7.11 and C.21 show the learned spatial kernels when only the feature maps are padded. The learned spatial kernels are still relatively small and well-localized.

7.3.6 Ablation on More Modules

We demonstrate that our NIFF can be combined with other Fourier modules to achieve networks that operate mostly in the Fourier domain. Hence, the number of

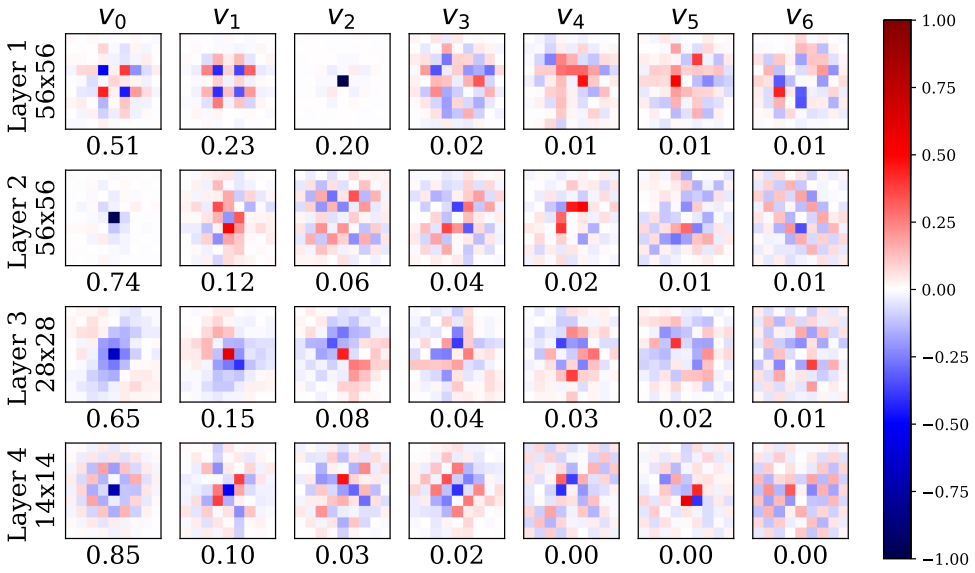


FIGURE 7.12: **Spatial Filter Visualizations for our NIFF ResNet-18 with Approximating Linear Convolutions on ImageNet-100.** Actual kernels in the spatial domain of a ResNet-18 which mimics linear convolutions with our NIFF trained on ImageNet-100. We plot for each kernel the zoomed-in (13×13) version below for better visibility. Still, most kernels exhibit well-localized, small spatial kernels. However, they are slightly larger than the kernels learned without padding and cropping.

transformations can be reduced. Table 7.6 demonstrates that adding the downsampling layer, our FLC Pooling from Section 6.2.2 or the last AveragePooling and the fully connected layer also yields good results. Also, combining all of them, NIFF, FLC Pooling and the Average Pooling plus Fully connected layer performs quite well. However, incorporating the ComplexBatchNorm [Trabelsi et al. (2018)] leads to a drop in accuracy by roughly 15%. We also tried to incorporate the non-linearity into the frequency domain, but we were not able to achieve much better results than by removing it fully.

7.4 Discussion

We now discuss key architectural design choices for NIFF. Further, we compare NIFF's performance with both standard and large filter sizes, analysing required FLOPs and runtime for NIFF versus standard convolutions across various kernel sizes. Finally, we outline limitations.

7.4.1 NIFF's Architecture

In the following, we discuss the architecture used for our NIFFs for each backbone network architecture. Note that the size of the NIFF is adjusted to the size of the baseline network as well as the complexity of the classification task.

High-Resolution Data For the networks trained on ImageNet-100 and ImageNet-1k, the size of the neural implicit function to predict the NIFF is kept the same for each architecture, respectively, while the size of the neural implicit function is adjusted to the network architecture to achieve approximately the same number of

TABLE 7.5: **Performance of Different Padding Methods for our NIFF.** Evaluation of top 1 and top 5 accuracies on ImageNet-100 for our NIFF ResNet-18 with different kinds of padding.

Name	Acc@1 ↑	Acc@5 ↑
Baseline	87.52	97.50
NIFF (ours)	86.52	97.14
NIFF zero padding	87.00	97.54
NIFF reflect padding	86.64	97.24
NIFF circular padding	87.06	97.34

TABLE 7.6: **Towards CNNs in the Fourier Domain.** Comparison on CIFAR-10 of NIFF MobileNet-v2 [Sandler et al. (2018)] incorporating more modules, like our FLC Pooling introduced in Chapter 6 and other methods like Complex BatchNorm, short Complex BN [Trabelsi et al. (2018)] and the AveragePooling which is equivalent to selecting the DC-component in the frequency domain and operating the fully connected layer, FC, into the frequency domain.

NIFF (ours)	FLC Pooling (ours)	Complex BN 2018	AvgPooling + FC	Acc@1 ↑
✓	✗	✗	✗	**94.03**
✓	✓	✗	✗	93.61
✓	✗	✓	✗	78.60
✓	✗	✗	✓	93.82
✓	✓	✓	✗	79.83
✓	✓	✗	✓	93.27
✓	✗	✓	✓	73.90
✓	✓	✓	✓	55.81

trainable parameters. Hence, the lightweight MobileNet-v2 model [Sandler et al. (2018)] and the small DenseNet-121 [Huang et al. (2017b)] incorporate a smaller lightweight neural implicit function to predict the NIFF, while larger models like ResNet [He et al. (2016a)] or ConvNeXt-tiny [Liu et al. (2022b)] incorporate a larger neural implicit function. For simplicity, we define two NIFF architectures. One for the large models and one for the smaller, lightweight models.

For the smaller, lightweight models, the neural implicit function consists of three stacked 1×1 convolutions with one SiLU activation after the first one and one after the second one. The dimensions for the three 1×1 convolutions are as follows. We start with two channels and expand to eight channels. From these eight channels, the second 1×1 convolution suppresses the channels down to four. Afterwards, the last 1×1 convolution maps these four channels to the desired number of point-wise multiplication weights.

For the larger models, we used four layers within the neural implicit function for NIFF. The structure is similar to the small NIFF between each 1×1 convolution a SiLU activation function is applied. The dimensions for the four layers are as follows. First, from two to 16 channels, secondly from 16 to 128 channels and afterwards suppressed down from 128 to 32 channels. The last 1×1 convolution maps these 32 channels to the desired number of point-wise multiplication weights.

We show that the smaller NIFF size for the lightweight models does not influence

the resulting performance. Thus, we train a lightweight MobileNet-v2 with larger NIFFs (similar size as the larger models). The results are presented in Table 7.7. The network does not benefit from the larger NIFF size. Hence, we assume that keeping the smaller NIFFs for the smaller, lightweight models can achieve a good trade-off between the number of learnable parameters and performance.

TABLE 7.7: **NIFF Size Evaluation.** Evaluation of top 1 and top 5 accuracies on ImageNet-100 for different NIFF sizes for the lightweight MobileNet-v2 [Sandler et al. (2018)].

Name	Acc@1 ↑	Acc@5 ↑
Baseline	**84.94**	96.28
small NIFF	83.72	**96.40**
big NIFF	83.82	96.32

Low-Resolution Data All networks trained on CIFAR-10 incorporate the same NIFF architecture. The NIFF consists of two stacked 1×1 convolutions with a ReLU activation function in between. The 1×1 convolution receives as input two channels, which encode the x and y coordinates as described in Figure 7.2. The 1×1 convolution expands these two channels to 32 channels. From these 32 channels, the next 1×1 convolution maps the 32 channels to the desired number of point-wise multiplication weights.

Ablation on Separated Convolution Further, we ablate our design choices to use separated depth-wise and 1×1 convolutions instead of full convolutions for efficiency. Hence, we train all standard networks incorporating full convolutions to use separated convolutions (separated in depth-wise and 1×1 convolution) as well as our NIFF as full convolution. Tables 7.1 and 7.2 demonstrate that using separated convolutions in the spatial domain performs slightly worse than the baseline but also reduces the amount of learnable parameters similarly to our NIFF. Using full convolutions in our NIFF leads to an increase in accuracy but also an increased amount of learnable parameters. Hence, we can see a clear trade-off between the number of learnable parameters and accuracy.

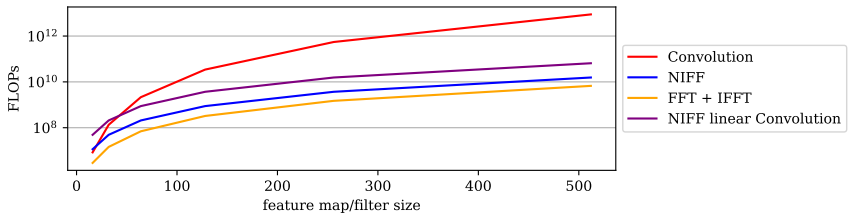


FIGURE 7.13: **FLOPs for Different Kinds of Convolutions.** FLOPs in Log-scale for computing a simple FFT and IFFT, a standard depth-wise convolution and our NIFF (including FFT and IFFT) and a linear convolution executed without NIFF for convolutions with kernels as big as the feature maps for the example of 64 channels.

7.4.2 Efficiency

Since our NIFF implementation is conceived for analysis purposes, our models are not optimized for runtime. In particular, we compute repeated FFTs in *PyTorch* to

allow the computation of the remaining network components in the spatial domain so that the models are equivalent to large kernel models computed in the spatial domain. Yet, Figure 7.13 demonstrates that with large kernel sizes, our NIFF approach with repeated FFTs is much more efficient in terms of FLOPs compared to the spatial convolution.

Our approach is slower than the current implementation with spatial convolutions due to the repetitive use of FFT and IFFT. However, when comparing the number of FLOPs needed to compute convolutions with kernel sizes as big as the feature maps to our NIFF approach, NIFF requires significantly fewer FLOPs, especially with increased feature map size. Figure 7.13 demonstrates that most of the FLOPs for our NIFF result from the additional FFT and IFFT operation. Still, we require much fewer FLOPs than large spatial convolutions. Moreover, when combined with further modules in the Fourier domain like our FLC Pooling from Chapter 6, the number of transformations needed is reduced.

Moreover, we evaluate the runtime per epoch for each model on CIFAR-10 (Table 7.8) and ImageNet-100 (Table 7.9) and compare it to the standard spatial 3×3 convolution, which has a much smaller spatial context than our NIFF as well as spatial convolutions which are as large as the feature maps. This would be comparable to our NIFF. Obviously, small spatial kernels (3×3) are much faster than larger kernels like NIFF or large spatial kernels. However, NIFF is much faster than the large spatial kernels during training. Especially on high-resolution datasets like ImageNet-100, our NIFF is over four times faster on ResNet-50 and over three times faster on ConvNeXt-tiny compared to the large convolution in the spatial domain.

In general, we want to emphasize that our NIFF models still learn infinite large kernels while all kernels in the spatial domain are limited to the set kernel size. Performing a 2D spatial convolution on an $N \times N$ image, g, with filters of the same size requires $O(N^4)$ operations. However, employing FFT and pointwise multiplication, as done by our NIFF, (following Equation 7.1) reduces the complexity to $O(N^2 \log(N))$.

TABLE 7.8: **NIFF Efficiency Evaluation on CIFAR-10.** Average training (left) and inference time (right) per epoch in seconds and standard deviation on one NVIDIA Titan V of NIFF compared to standard spatial convolutions 3×3 or 7×7 and maximal larger spatial convolutions on CIFAR-10.

Architecture	Training Time ↓			Inference Time ↓		
Filter Size	Baseline 3x3/7x7	Spatial Conv feature map sized	NIFF (ours)	Baseline 3x3/7x7	Spatial Conv feature map sized	NIFF (ours)
ConvNeXt-tiny	68.31 ± 0.62	108.97 ± 0.25	96.48 ± 1.97	2.35 ± 0.05	4.06 ± 0.02	6.32 ± 0.14
ResNet-18	8.57 ± 0.20	22.75 ± 0.28	17.87 ± 0.54	2.69 ± 0.10	3.29 ± 0.08	3.14 ± 0.54
ResNet-50	7.98 ± 0.10	37.05 ± 0.48	27.36 ±0.16	3.09 ± 0.13	5.05 ± 0.45	5.26 ± 0.06
DenseNet-121	26.36 ± 1.32	67.11 ± 0.25	84.51 ± 3.71	3.40 ± 0.05	5.27 ± 0.02	6.50 ± 0.14
MobileNet-v2	22.50 ± 0.26	143.47 ± 0.20	83.41 ± 4.78	2.93 ± 0.06	8.68 ± 0.43	7.02 ± 0.04

7.4.3 Limitations

While NIFF maintains performance and generates reasonable filter weights, it approximates, rather than precisely replicates, linear convolutions. We detail the differences between NIFF and linear convolutions and propose an approximation in

TABLE 7.9: **NIFF Efficiency Evaluation on ImageNet-100.** Average training (left) and inference (right) time per epoch in seconds and standard deviation on four NVIDIA A100 of NIFF compared to standard spatial convolutions (3×3 or 7×7) and maximal larger spatial convolutions on ImageNet-100.

Architecture	Training Time ↓			Inference Time ↓		
Filter Size	Baseline 3x3/7x7	Spatial Conv feature map sized	NIFF (ours)	Baseline 3x3/7x7	Spatial Conv feature map sized	NIFF (ours)
ConvNeXt-tiny	92.19 ± 4.21	487.89 ± 3.54	149.70 ± 1.32	4.86 ± 0.13	17.07 ± 0.19	6.06 ± 0.08
ResNet-18	31.99 ± 0.71	608.13 ± 22.53	89.00 ± 0.60	4.53 ± 0.52	24.57 ± 0.21	4.52 ± 0.18
ResNet-50	85.51 ± 0.22	951.43 ± 6.09	204.82 ± 0.33	5.43 ± 0.30	42.33 ± 0.21	10.35 ± 0.14
ResNet-101	152.39 ± 5.07	1392.21 ± 47.91	349.83 ± 2.44	7.35 ± 0.17	54.85 ± 0.25	16.82 ± 0.10
DenseNet-121	128.95 ± 1.64	10188.15 ± 28.17	408.08 ± 2.20	5.12 ± 0.07	104.36 ± 0.40	12.04 ± 0.18
MobileNet-v2	32.64 ± 0.25	856.84 ± 4.35	100.75 ± 0.15	4.27 ± 0.13	28.97 ± 0.04	6.61 0.16

Section 7.3.5. Nevertheless, potential factors such as ringing artifacts from the periodicity assumption (briefly discussed in Section 7.2) may affect comparability. Furthermore, the periodicity assumption and global filter application inherent in NIFF raise concerns regarding vulnerability to image corruptions and attacks. We discussed our NIFF only in the context of filter analysis, yet its true advantages may become more evident when applied to real high-resolution images. This is particularly relevant since computation times for NIFF are significantly higher than the standard convolutions with the currently employed kernel sizes. Furthermore, NIFF has the potential to enhance applications where data is inherently acquired in the frequency domain, such as seismic imaging or medical imaging. However, exploring these applications is beyond the scope of this thesis and should be considered for future research. We further discuss this direction in Section 8.2.

7.5 Conclusion

Using the proposed NIFF, we analysed the effective kernel sizes learned by state-of-the-art CNNs. We found that most models do not utilize the full potential for larger receptive fields. Specifically, on low-resolution datasets like CIFAR-10, networks rarely learn kernels larger than 5×5. On high-resolution datasets such as ImageNet, while models use kernels larger than 3×3, the majority remain within 9×9. Consequently, networks with small kernels, such as ResNet [He et al. (2016a)], can benefit from NIFF, whereas networks like ConvNeXt [Liu et al. (2022b)] are already near optimal for ImageNet. However, networks do learn a small number of large kernels, suggesting no globally optimal fixed kernel size exists. This necessitates a reevaluation of the current practice of using uniform kernel sizes.

In summary, we introduce NIFF CNNs, enabling the learning of convolution filters in the frequency domain that translate to infinitely large spatial kernels. NIFF efficiently replaces spatial convolutions with frequency-domain element-wise multiplication. Analysis of the resulting spatial kernels reveals they are localized and generally small (9×9). NIFF achieves comparable or superior performance to spatial convolutions on high-resolution datasets, leveraging large spatial context.

We anticipate that NIFF will significantly impact high-resolution real-world applications. Furthermore, its frequency-specific implementation offers advantages for frequency-domain imaging modalities, such as MRI and seismic scans.

Chapter 8

Conclusion and Outlook

Contents

The growing use of neural networks in daily life, especially in safety-focused applications, makes robustness and reliability key. While augmentation and adversarial training can improve robustness, a deeper understanding of model failures and their mitigation is necessary for practical application.

Thus, we take a step further in this direction by thoroughly analysing the properties of adversarially robust models in Part I of this thesis. With this analysis, we want to highlight the importance of evaluation beyond accuracy for classification models. In Part II, we take training CNNs beyond the spatial perspective, leveraging Fourier Theory, such as the Sampling and Convolution Theorems, to improve the robustness and efficiency of modern CNNs. Particularly, we develop aliasing-free downsampling to improve inherent robustness and make adversarial training more efficient and effective. Additionally, we establish a comprehensive and fair framework for systematically examining the preferred kernel size for a given network and task in an efficient manner. In the following, we outline the key insights and limitations of our work, as well as potential directions for future research.

8.1 Key Insights and Conclusion

Summarizing this thesis combines and highlights the need of a multifaceted analysis of classifier models beyond accuracy and provides inspiration for switching perspectives to *e.g.* Fourier representations for efficient and robust model development. Part I focuses on the multifaceted analysis of robust models, investigating their reliability and downsampling quality. For evaluating the downsampling quality, we introduce a novel aliasing measure. Part II leverages Fourier theory, proposing novel Fourier modules, for enhanced model robustness and efficiency. We continue assessing our

novel approaches from various perspectives *e.g.* different robustness measures, reliability and artifact measures for Chapter 6. Further, we introduce novel quantitative and qualitative measures for studying the learned kernel weights in Chapter 7.

In the following, we revisit the contributions of each chapter in more detail, highlighting the key findings and placing them within the broader context of this thesis. Subsequently, we discuss future research directions from this work in Section 8.2.

In Part I, which includes Chapters 4 and 5, we conduct two relevant studies of robust models and show that robust models are less overconfident and learn to downsample with less aliasing, respectively. This suggests that robustness enhances network properties beyond its primary purpose, including reliability and proper sampling. Our research underscores the importance of comprehensive model evaluation, extending beyond accuracy, for secure real-world applications.

Chapter 4 outlines the relationship between adversarial robustness and the reliability of various classification CNN models. We conduct an extensive study that evaluates the model's confidence of adversarial robust models and their non-robust counterparts. We find that robust models are less overconfident, and their prediction confidence can be used to reject erroneous samples. Further, we observe that adapting basic building blocks within the network, like activation function or downsampling, can lead to more desired confidence distributions *i.e.* high confidence in correct predictions and low confidence in incorrect ones. We argue that relying solely on accuracy for model evaluation is inadequate for real-world applications. To ensure safe and reliable deployment, models must perform robustly under both known and unknown conditions. While accuracy can give a sense of the known conditions, adversarial attacks can reveal worst-case behaviour in unknown conditions. Our findings suggest that non-robust models often exhibit high levels of overconfidence compared to adversarially robust models, raising significant safety concerns. Additionally, commonly used metrics like the expected calibration error may become unreliable when a network's accuracy is too low. To address these limitations, we recommend evaluating models using an ensemble of metrics to obtain a comprehensive understanding of their performance.

Chapter 5 further investigates into the properties of robust models. By proposing a new aliasing measure that quantifies downsampling-induced artifacts, we can analyze the behaviour of robust and non-robust models. We show that AT enables models to learn more effective downsampling strategies, reducing aliasing. In our study, we find a significant negative correlation between robustness and aliasing in almost all cases. Further, we analyse FGSM AT and the phenomena of catastrophic overfitting and find that catastrophic overfitting coincides with a steep increase in aliasing in the epoch of catastrophic overfitting. Hence, we propose to use our aliasing measure as an alternative early stopping criterion for FGSM AT and show that the resulting models achieve results on par.

These two chapters underscore the importance of comprehensive model evaluation, beyond accuracy, and the application of classical principles, like the sampling theorem, to develop reliable, robust, and efficient models.

In Part II, we continue broadening our perspective and leverage the Fourier theory to enhance CNNs for improved downsampling and more efficient large convolutions in Chapters 6 and 7, respectively.

Chapter 6 introduces aliasing-free downsampling based on the revealed correlation between aliasing and robustness in Chapter 5. Our proposed FLC Pooling and ASAP guarantee to be aliasing-free and ASAP further reduces sinc-interpolation artifacts. Both methods increase the native robustness of the models, leading to better

performance on common corruptions under pixel shift or adversarial attacks. Moreover, our FLC Pooling and ASAP enhance adversarial training, leading to higher adversarial robustness and less overconfident models under attack. In the case of FGSM AT, which is significantly more efficient than more sophisticated AT methods, our downsampling approach reduces the risk of catastrophic overfitting. This enables faster AT and improved convergence compared to FGSM AT with early stopping. Our results demonstrate that leveraging perspectives such as the Fourier theory leads to more robust and efficient models. This highlights the need to broaden the perspectives we adopt, similar to our recommendation of using multiple evaluation metrics for a comprehensive performance assessment in Part I.

Chapter 7 introduces Neural Implicit Frequency Filters (NIFFs), a powerful tool which also leverages Fourier theory, for analysing the optimal filter sizes in CNNs. Amid the trend of increasingly large models and filters, we pose the question: *"How large do CNN kernels really need to be?"* NIFFs enable us to study the preferred filter size learned decoupled from the number of parameters in an efficient manner by leveraging the Convolution Theorem and Neural Implicit Functions in the Fourier domain. Our analysis reveals that networks primarily learn small, localized filters, typically 9×9 for high-resolution datasets like ImageNet and 3×3 for low-resolution datasets like CIFAR. However, in rare cases, networks prefer much larger filters, sometimes as large as the feature map itself. This suggests a need to reconsider the static implementation of CNNs. Additionally, NIFFs facilitate efficient large-scale convolutions, a promising avenue for handling growing data scales and resolutions. The frequency-specific nature of NIFFs also makes them well-suited for imaging modalities inherently recorded in the frequency domain, such as MRI or seismic scans, offering exciting potential for future applications.

Furthermore, we integrate our NIFF with the aliasing-free pooling method introduced in Chapter 6, reducing the number of FFT transformations required while achieving results comparable to the baseline. This marks a step forward in transitioning classical CNNs into the Fourier domain.

8.1.1 Impact on the Community

Based on our key findings, we outline the impact we hope this thesis will have on our research community in the following.

First and most importantly, we want to raise awareness for proper analysis settings and shift the focus of our evaluation metrics from the simple evaluation of accuracy to more diverse and very important properties like reliability and robustness, making models more applicable and safe for the real world. Furthermore, we demonstrate that fundamental principles from digital signal processing are essential for achieving desirable properties in neural networks. Therefore, analysing and thoroughly understanding these models and their components remains a key challenge. By leveraging well-established theorems and classical signal processing methods, we enhance network robustness and gain deeper insights into kernel design. We hope to inspire future research to further explore these rich theoretical foundations.

Another key aspect we aim to raise awareness of is the impact of architectural choices on a network's performance and biases. While the computer vision community has become increasingly aware of issues such as aliasing [Karras et al. (2021); Hossain et al. (2023); Zhang (2019); Zou et al. (2023); Vasconcelos et al. (2020)], many practitioners applying computer vision models remain unaware of these challenges. For instance, the widely used U-Net [Ronneberger et al. (2015)], which is still a standard in medical image segmentation, introduces several artifacts due to its reliance

on MaxPooling for downsampling. We examine this issue in more detail in Section 8.2, highlighting its implications for model performance and potential mitigation strategies.

Furthermore, our analysis of CNN kernel design indicates that for current low-resolution classification datasets, small kernels are a suitable choice. However, as input image resolutions increase, we expect networks to favour larger kernel sizes. This trend is supported by Transformer models, which emphasize the importance of larger receptive fields achieved through larger convolutions such as the 11×11 kernels used in Swin Transformer [Liu et al. (2021, 2022a)] and the role of cross-patch correspondences in capturing a more global representation of the input for image classification Additionally, Rao et al. (2021) demonstrated that incorporating frequency representations within Transformer architectures can lead to improved performance. With the general framework established by our NIFF, we aim to provide a broader understanding of the necessary model capacity to enhance efficiency. Moreover, our approach could facilitate the use of larger convolutions, which are significantly more computationally efficient when performed in the Fourier domain rather than in the spatial domain.

Another key takeaway we highlight is the value of reassessing one's own methods. During subsequent experiments with our FLC Pooling, we discovered that it introduced sinc-interpolation artifacts due to the rectangular function applied. Consequently, we refined it with a Hamming window, proposing our ASAP method. However, both approaches are based on the assumption that blurring is acceptable for the task at hand. Our subsequent work in [Agnihotri et al. (2024a)], not presented in this thesis, demonstrates that for tasks such as image deblurring or denoising, the high-frequency information removed by FLC Pooling and ASAP is essential and should be preserved. This setting is discussed in our ablation study in Section 6.3.4. Although high-frequency information was not essential for classification in our ablation, it is critical for image deblurring, which relies on restoring edge details as present in our follow-up work not included in this thesis [Agnihotri et al. (2024a)].

8.1.2 Limitations

We have discussed the specific limitations of each approach presented in this thesis within their respective chapters. Below, we summarize the key limitations from our analysis in Part I, which includes Chapters 4 and 5. Finally, we highlight the main limitation in Part II, which includes Chapters 6 and 7 and briefly discuss how we aim to address some of these limitations in future work.

In Part I, the primary limitation of both analyses is their reliance on the provided models, allowing us to evaluate only one model per set of training hyperparameters. Consequently, variations resulting from different training strategies or even simple variance by selecting a different random seed cannot be accounted for. Notably, our experiments in Section 6.3.4, which assess model confidence calibration under attack, confirm the findings from Section 4.2.2 regarding improved downsampling and confidence calibration. However, the observed benefits are less pronounced than those found for a single seed evaluated in Section 4.2.2, highlighting the limited expressiveness of our evaluation. Further, we limit our scope to confidences based on the values after Softmax, which, as we discuss in Section 4.3.1, is not optimal. Yet, as already discussed, using the Softmax output is the most straightforward method without the need for training or fine-tuning the model, which was essential for our analysis. Additionally, our proposed aliasing measure introduced in Chapter 5 might similarly as our FLC Pooling from Chapter 6 suffer from sinc-interpolation

artifacts due to the rectangle cutout of the low-frequency components, limiting its reliability to serve as the optimal downsampling groundtruth. Given these limitations, we emphasize the importance of diverse evaluations, highlighting that a comprehensive analysis may require multiple perspectives to draw accurate conclusions. One example of this kind of comprehensive analysis is given in Chapter 7 where we first evaluate the kernel mass ratio (Section 7.3.2) which gives an overview of all kernels and the following PCA analysis (Section 7.3.4) dives deeper in the predominate structure of the kernels from different perspectives.

Part II presents novel Fourier modules derived from Fourier theory to enhance CNNs. Specifically, we propose downsampling and convolutions in the Fourier domain to improve robustness and efficiency. Although our models achieve comparable or superior performance and are theoretically grounded in the Sampling and Convolution theorems, they still rely on approximations. *E.g.* our FLC Pooling, suffers from sinc-interpolation artifacts and our ASAP is highly smooth which is not an issue for classification tasks looked at in this thesis but is indeed an issue if you want to take it beyond classification to *e.g.* pixel-wise prediction tasks like segmentation or image deblurring. We propose a solution for the loss of high-frequency information in [Agnihotri et al. (2024a)] countering this limitation. However, the discrete transformation between the spatial and Fourier domains inherently introduces approximation errors due to the quantization of both frequency components and pixel values.

Next, we outline the limitation we seek to address in future research for Chapters 6 and 7. While all our approaches work well on datasets like CIFAR or ImageNet, testing on real-world data is not in the scope of this thesis. These datasets contain real images but are cleaner and quite low-resolution compared to real-world data. Thus, we suggest, as one further direction, to use the tools developed in this thesis for real-world applications. There are many use cases where proper sampling is of high importance, such as in medical imaging, seismic imaging, or astrophysics. Further, high-resolution data is available but still hard to handle, which could be addressed *e.g.* with our NIFF.

8.2 Future Directions

In the following, we discuss potential future directions resulting from this thesis. The first three parts of this Section focus on extending and integrating our methods proposed in Part II. Followed by Section 8.2.4, which highlights broader directions that can be pursued based on the ideas and approaches developed in this thesis.

8.2.1 Exploring the Multifaceted Role of Aliasing

In Chapter 5, we establish a strong link between robustness and aliasing caused by incorrect sampling. Subsequently, we introduce aliasing-free sampling in Chapter 6, which leads to increased robustness of the networks with and without AT. In [Agnihotri et al. (2024a)], we show that aliasing is not only an issue for decoder networks but also plays a crucial role in encoder networks. Specifically, encoder-decoder networks benefit from symmetric sampling, meaning that both the encoder and decoder should employ similar sampling approaches. We propose an aliasing-free path for both to enhance the network robustness against adversarial attacks on inputs in image deblurring tasks. For image generation, Karras et al. (2021) showed that aliasing is one of the main drivers for unrealistic image generation, as details appear glued to image coordinates instead of being aligned with the surface of the object. They

propose an ensemble of architecture improvements, incorporating a low-pass filter after sampling and another post-activation, effectively suppressing aliasing. Their approach underscores that aliasing artifacts can originate from sources beyond mere sampling *e.g.* activation functions. Building upon our findings in Chapter 4, which demonstrated that smoother activation functions improve network calibration, and the observation in [Hein et al. (2019)] that ReLU networks are highly overconfident, further investigation into the interplay of activation functions, aliasing, and reliability is a valuable avenue for future research.

Moreover, we believe that there are many other applications and areas where aliasing plays a crucial role. One specific area is medical imaging, where careful precautions are taken to reduce aliasing during image acquisition. However, when using and interpreting the data, the most commonly used network architecture is still the classical U-Net [Ronneberger et al. (2015)], which employs in the standard setting MaxPooling as the downsampling method. As we demonstrated in Section 6.3.1 (Figure 6.6 and Table 6.1), MaxPooling is highly susceptible to aliasing, significantly distorting the information in the image. Thus, depending on the task to solve, we recommend replacing standard downsampling with our aliasing-free downsampling for classification tasks or for reconstruction tasks, adapting the method presented in [Agnihotri et al. (2024a)], which provides an aliasing-free path while preserving high-frequency information necessary for detailed reconstructions. Our method proposed in [Agnihotri et al. (2024a)] offers an aliasing-free path that significantly enhances the network's robustness, making it more suitable for real-world tasks.

Our goal is to highlight the existence and urgency of this problem. Standard downsampling methods as well as activating functions can compromise neural network security, resulting in high-confidence but incorrect predictions and increased attackability.

8.2.2 High-Resolution with NIFF and FLC Pooling

The data employed for academic testing of new models is still very low-resolution. CIFAR with 32×32 pixels is so small that recognizing specific classes like different cat breeds would not be possible (*e.g.* Figure 2.3). In contrast, ImageNet images with 224×224 pixels have a higher resolution and also include different cat breeds, like Persian or Egyptian cats. However, also ImageNet is still far below the actual images we humans perceive every day (*e.g.* 11% of mobile phones in December 2024 use 360×800 pixels [StatCounter (2024)] and our flat screens use up to 7680×4320 pixels for 8K screens [Statista (2021)] or 1920×1080 for standard full HD screens [Video Electronics Standards Association (VESA) (2013)]). For a long time, we were restricted to small resolutions due to computational constraints and limited data availability. However, with the increase in data size and increased computational resources, it is now possible to make predictions on much higher-resolution data. Reed et al. (2023) presented an academic approach for analysing high-resolution images to enable effective geospatial representation learning, called Scale-Aware Masked Autoencoder (Scale-MAE). This method was applied in real-world scenarios to better organize first-aid efforts during events such as the 2023 earthquake in Turkey [Gupta (2024)], highlighting the relevance and need for high-resolution image processing to have a real impact.

Scale-MAE decodes masked images, splitting them during the decoding process into low- and high-frequency components using a so-called Laplacian Block to produce images at different scales. However, all these operations are performed in the

spatial domain, which does not entirely separate low- from high-frequency information. This limitation could be addressed by our FLC Pooling implementation, which preserves high-frequency information instead of removing it. Consequently, we would achieve a complete separation of high-frequency information from low-frequency components. Additionally, with our NIFF, we can utilize larger convolutions, eliminating the need for compute- and memory-consuming transformer architectures, which are often preferred due to the global view they provide compared to conventional CNN architectures. These changes could enhance both model performance and efficiency, with the added benefit of creating more sustainable models. NIFF could also be applied in other domains with high-resolution data, such as astrophysics, seismic, or medical imaging, by leveraging global information and making computations more efficient at scale.

8.2.3 NIFF Beyond 2D

While our NIFF framework demonstrates excellent scalability in 2D, we believe there is significant potential for extending its application to higher-dimensional convolutions. The first step would be the extension to 3D convolutions, which could find applications in Video Processing [Tran et al. (2015)], Action Recognition [Carreira & Zisserman (2017)] or Medical Imaging [Çiçek et al. (2016); Kamnitsas et al. (2017)], 3D convolutions are still sparsely used and convolutions beyond 3D are not common at all due to the high computational costs associated with increased dimensions [Rahim et al. (2021)]. However, with NIFF, these limitations can be mitigated, as it leverages the convolution theorem to maintain computational efficiency, even in higher dimensions. Furthermore, the efficient learning capabilities of NIFF, facilitated by neural implicit filters, can be adapted to accommodate the training of higher-dimensional weight tensors. This makes NIFF particularly well-suited for applications in medical imaging, such as MRI, where data dimensions can range from one to six [Sosnovik et al. (2009)], enabling the model to leverage the diversity and depth of multi-dimensional data. Moreover, NIFF naturally exploits frequency-specific features, which can be particularly relevant in medical imaging, where such features often carry critical information.

8.2.4 A Comprehensive Overview

In this thesis, we demonstrate the importance of implementing models grounded in theory and the need of adapting different perspectives to gain a comprehensive model understanding. Therefore, reintroducing fundamental signal processing techniques to enhance robustness, reliability, and efficiency represents a step in the right direction toward developing sustainable and secure models for the future.

In the following, we list several areas that can benefit from our proposed approaches:

1. **Increased Robustness.** We demonstrate in Chapter 6 that proper downsampling, grounded in Fourier theory, leads to improved robustness. Our improvement stems from the proper evaluation and understanding of the properties of robust model properties through a signal processing lens. Incorporating this fundamental knowledge enables better handling of noise, distortions, and irregularities inherent in real-world data, thereby improving model robustness and generalizability.

2. **Efficient Feature Extraction.** Signal processing methods are highly effective in extracting relevant features from signals structured in time or space, which can reduce the complexity of neural networks. With our NIFF, we demonstrate that it is possible to learn spatially infinite large convolutions while maintaining a comparable number of parameters to the baseline. Additionally, by leveraging the Convolution Theorem, we ensure that the computations remain manageable. The following outlines additional benefits and related concepts:

   - **Dimensionality Reduction.** Techniques like Fourier transforms, wavelet analysis, or spectrograms can transform raw signals into representations that are easier to analyse, reducing the dimensionality of the input data while preserving crucial information. Guan et al. (2019) made use of this property and zeroed out small coefficients of the feature map in the frequency domain. This approach could be further explored in combination with our NIFF, for instance, by applying additional sparsity regularization on the learned weights.
   - **Domain-Specific Features.** In audio (*e.g.*, speech recognition) and image processing (*e.g.*, edge detection), features that are derived through signal processing (*e.g.*, Mel-frequency cepstral coefficients (MFCCs) or Gabor filters) can lead to better model performance, as they capture key signal characteristics more efficiently than raw data. Alekseev & Bobe (2019) indeed showed that constraining the first layer of a CNN to fit a learned Gabor function improves convergence and reduces training complexity. We show in subsequent work not included in this thesis that we can constrain our NIFF to learn Wiener filters, which enhances model robustness.
   - **Multi-Scale Analysis.** Wavelet transforms, our FLC Pooling, ASAP or other signal processing tools enable multi-scale analysis, capturing information at different resolutions. Thus benefiting both robustness and the acquisition of global insights by mitigating sampling artifacts. As discussed in Section 8.2.2, incorporating our NIFF and an adapted version of our FLC Pooling could enhance Scale-MAE [Reed et al. (2023)] by capturing more global information, reducing artifacts during the split into high- and low-frequency images, and increasing efficiency.

3. **Reduced Model Complexity.** As shown with our NIFF, signal processing-based feature extraction often helps reduce the size and complexity of neural networks. By preprocessing data and extracting meaningful features, models can become more compact. Further executing large convolutions in the frequency domain reduces the computational costs significantly, as shown in Table 7.13.

4. **Prevents Overfitting.** By focusing on signal-relevant features, models avoid learning spurious correlations and are less prone to overfitting, as shown with our FLC Pooling and ASAP in Section 6.3.3, which both reduce the risk of catastrophic overfitting in FGSM AT.

5. **Improved Interpretability.** The model's behaviour can partially be traced back to physical properties of the signal, such as frequency, phase [Meyer et al. (2018)], or energy, which might be easier to understand and use in a meaningful way. We leveraged this property in the analysis of the filters learned by our NIFF, observing that the network learns less complex structures for the

imaginary part in Section 7.3.4. This suggests that the network prefers learning cosine structures (with a peak in the center) over sine structures.

6. **Explainability and Verification.** Signal processing-based models may offer greater explainability as their foundations are based on well-understood mathematical operations. This is increasingly important in fields where safety and trust are crucial (*e.g.*, healthcare, finance, and autonomous systems). Furthermore, models that leverage signal processing principles may be easier to verify and validate against real-world phenomena, ensuring more reliable performance. Chapter 5 demonstrates a strong link between robustness and aliasing after downsampling, emphasizing the need to mitigate these issues for enhanced model robustness. This was further validated in Chapter 6, where aliasing-free pooling improved model robustness. Building on this, we believe further advancements for network modules and learning strategies are possible. For instance, Karras et al. (2021) improved image realism by reducing aliasing through enhanced activation functions, and Jung & Keuper (2021) achieved better image generalization with an added frequency loss.

7. **Incorporating Domain Knowledge.** Domain-specific knowledge across fields like telecommunications, audio processing, and medical imaging can be integrated more easily if models follow digital signal processing pipelines. Leveraging this knowledge can ensure that models respect physical and engineering constraints inherent to signals (*e.g.*, bandwidth limitations, sampling theorem) [Ladwig et al. (2024); Harder et al. (2022)].

8. **Data Augmentation and Preprocessing.** The preprocessing process of the input data needs to be designed carefully to avoid *e.g.* downscaling attacks as proposed by Xiao et al. (2017). Signal processing methods can be used to preprocess or augment data to evaluate modern models against diverse image corruptions. Müller et al. (2023) proposed a benchmark for investigating robustness to realistic, practically relevant optical blur effects based on optical aberrations, implemented with specific kernels in the Fourier domain. We expect that using signal-processing-conform downsampling methods, such as our FLC Pooling and ASAP, could better withstand certain corruptions compared to standard downsampling, similar to the results presented in Section 6.3 on common corruptions.

Bibliography

Addepalli, S., Jain, S., Sriramanan, G., Khare, S., and Radhakrishnan, V. B. Towards achieving adversarial robustness beyond perceptual limits. In *International Conference on Machine Learning Workshop on Adversarial Machine Learning*, 2021.

Agnihotri, S., Gandikota, K. V., Grabinski, J., Chandramouli, P., and Keuper, M. On the unreasonable vulnerability of transformers for image restoration - and an easy fix. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 3707–3717, 2023.

Agnihotri, S., Grabinski, J., Keuper, J., and Keuper, M. Beware of aliases–signal preservation is crucial for robust image restoration. *arXiv preprint arXiv:2406.07435*, 2024a.

Agnihotri, S., Grabinski, J., and Keuper, M. Improving feature stability during upsampling – spectral artifacts and the importance of spatial context. In *European Conference on Computer Vision*. Springer, 2024b.

Alekseev, A. and Bobe, A. Gabornet: Gabor filters with learnable parameters in deep convolutional neural network. In *International Conference on Engineering and Telecommunication*, pp. 1–4, 2019.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Andriushchenko, M. and Flammarion, N. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020.

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

Arfken, G. Discrete orthogonality–discrete fourier transform. *Mathematical Methods for Physicists*, 3:787–792, 1985.

Ayat, S. O., Khalil-Hani, M., Ab Rahman, A. A.-H., and Abdellatef, H. Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. *Neurocomputing*, 364:152–167, 2019.

Azulay, A. and Weiss, Y. Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning Research*, 20(184):1–25, 2019.

Bernhard, R., Moëllic, P.-A., Mermillod, M., Bourrier, Y., Cohendet, R., Solinas, M., and Reyboz, M. Impact of spatial frequency based constraints on adversarial robustness. In *International Joint Conference on Neural Networks*, pp. 1–8. IEEE, 2021.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

Bracewell, R. and Kahn, P. B. The fourier transform and its applications. *American Journal of Physics*, 34(8):712–712, 1966.

Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2018.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57. IEEE, 2017.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019a.

Carmon, Y., Raghunathan, A., Schmidt, L., Duchi, J. C., and Liang, P. S. Unlabeled data improves adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019b.

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Chaman, A. and Dokmanic, I. Truly shift-invariant convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3773–3783, 2021.

Chen, E.-C. and Lee, C.-R. Ltd: Low temperature distillation for robust adversarial training. *arXiv preprint arXiv:2111.02331*, 2021.

Chen, J., Cheng, Y., Gan, Z., Gu, Q., and Liu, J. Efficient robust training via backward smoothing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6222–6230, 2022.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on Artificial Intelligence and Security*, pp. 15–26, 2017.

Chen, Q., Wang, W., Wu, F., De, S., Wang, R., Zhang, B., and Huang, X. A survey on an emerging area: Deep learning for smart city data. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(5):392–410, 2019.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., and Wang, Z. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 699–708, 2020.

Chen, T., Zhang, Z., Liu, S., Chang, S., and Wang, Z. Robust overfitting may be mitigated by properly learned smoothening. In *International Conference on Learning Representations*, 2021a.

Chen, Y., Liu, S., and Wang, X. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8628–8638, 2021b.

Chi, L., Jiang, B., and Mu, Y. Fast fourier convolution. In *Advances in Neural Information Processing Systems*, volume 33, pp. 4479–4488. Curran Associates, Inc., 2020.

Chibane, J., Mir, A., and Pons-Moll, G. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21638–21652. Curran Associates, Inc., 2020.

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention*, pp. 424–432. Springer, 2016.

Cohen, G., Sapiro, G., and Giryes, R. Detecting adversarial samples using influence functions and nearest neighbors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14453–14462, 2020.

Cooley, J. W. and Tukey, J. W. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

Corbière, C., Thome, N., Bar-Hen, A., Cord, M., and Pérez, P. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020a.

Croce, F. and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *International Conference on Machine Learning*, pp. 2196–2205. PMLR, 2020b.

Croce, F. and Hein, M. Mind the box: l_1-apgd for sparse adversarial attacks on image classifiers. In *International Conference on Machine Learning*, pp. 2201–2211. PMLR, 2021.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops*, pp. 702–703, 2020.

Cui, J., Liu, S., Wang, L., and Jia, J. Learnable boundary guided adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15721–15730, 2021.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 764–773, 2017.

Dai, S., Mahloujifar, S., and Mittal, P. Parameterizing activation functions for adversarial robustness. In *IEEE Security and Privacy Workshops*, pp. 80–87. IEEE, 2022.

Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

DeGroot, M. H. and Fienberg, S. E. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Dhar, P. The carbon impact of artificial intelligence. *Nat. Mach. Intell.*, 2(8):423–425, 2020.

Ding, G. W., Sharma, Y., Lui, K. Y. C., and Huang, R. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2020.

Ding, X., Zhang, X., Han, J., and Ding, G. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11963–11975, 2022.

Dong, X., Bao, J., Chen, D., Zhang, W., Yu, N., Yuan, L., Chen, D., and Guo, B. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12124–12134, 2022.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Dunteman, G. H. *Principal components analysis*, volume 69. Sage, 1989.

Durall, R., Keuper, M., and Keuper, J. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7890–7899, 2020.

Elfwing, S., Uchibe, E., and Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Engstrom, L., Ilyas, A., Salman, H., Santurkar, S., and Tsipras, D. Robustness (python library), 2019. URL https://github.com/MadryLab/robustness.

Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

Forsyth, D. and Ponce, J. *Computer Vision: A Modern Approach*. An Alan R. Apt book. Prentice Hall, 2003. ISBN 9780130851987.

Freeman, W. T., Adelson, E. H., et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

Gavrikov, P. and Keuper, J. Adversarial robustness through the lens of convolutional filters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 139–147, 2022.

Gavrikov, P., Lukasik, J., Jung, S., Geirhos, R., Mirza, M. J., Keuper, M., and Keuper, J. Can we talk models into seeing the world differently? In *International Conference on Learning Representations*, 2025.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., and Brendel, W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

Gonzales, R. C. and Wintz, P. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

Gonzalez, R. C. and Woods, R. E. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006. ISBN 013168728X.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2015.

Gowal, S., Qin, C., Uesato, J., Mann, T., and Kohli, P. Uncovering the limits of adversarial training against norm-bounded adversarial examples. *arXiv preprint arXiv:2010.03593*, 2021a.

Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. Improving robustness using generated data. *Advances in Neural Information Processing Systems*, 34, 2021b.

Grabinski, J., Gavrikov, P., Keuper, J., and Keuper, M. Robust models are less overconfident. In *Advances in Neural Information Processing Systems*, 2022a.

Grabinski, J., Jung, S., Keuper, J., and Keuper, M. Frequencylowcut pooling–plug & play against catastrophic overfitting. In *European Conference on Computer Vision*, 2022b.

Grabinski, J., Keuper, J., and Keuper, M. Aliasing and adversarial robust generalization of cnns. *Machine Learning*, pp. 1–27, 2022c.

Grabinski, J., Keuper, J., and Keuper, M. Aliasing coincides with CNNs vulnerability towards adversarial attacks. In *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022d.

Grabinski, J., Keuper, J., and Keuper, M. Fix your downsampling asap! aliasing and sinc artifact free pooling in the fourier domain. *arXiv preprint arXiv:2307.09804*, 2023.

Grabinski, J., Keuper, J., and Keuper, M. As large as it gets – studying infinitely large convolutions via neural implicit frequency filters. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. Featured Certification.

Grosse, K., Manoharan, P., Papernot, N., Backes, M., and McDaniel, P. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022.

Guan, B., Zhang, J., Sethares, W. A., Kijowski, R., and Liu, F. Specnet: spectral domain convolutional neural network. *arXiv preprint arXiv:1905.10915*, 2019.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2017.

Guo, G. and Zhang, N. A survey on deep learning based face recognition. *Computer vision and image understanding*, 189:102805, 2019.

Guo, M.-H., Lu, C.-Z., Hou, Q., Liu, Z.-N., Cheng, M.-M., and min Hu, S. Segnext: Rethinking convolutional attention design for semantic segmentation. In *Advances in Neural Information Processing Systems*, 2022.

Gupta, R. Understanding chaotic environments and the policies surrounding dual-use ai. Fall 2024 BAIR Vision Workshop, 2024.

Gurau, C., Bewley, A., and Posner, I. Dropout distillation for efficiently estimating model confidence. *arXiv preprint arXiv:1809.10562*, 2018.

Ha, D., Dai, A. M., and Le, Q. V. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Hamming, R. W. and Stearns, S. D. Digital filters. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):67–67, 1979.

Harder, P., Pfreundt, F.-J., Keuper, M., and Keuper, J. Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain. In *International Joint Conference on Neural Networks*, pp. 1–8. IEEE, 2021.

Harder, P., Yang, Q., Ramesh, V., Sattigeri, P., Hernandez-Garcia, A., Watson, C. D., Szwarcman, D., and Rolnick, D. Generating physically-consistent high-resolution climate data with hard-constrained neural networks. In *NeurIPS 2022 Workshop on Tackling Climate Change with Machine Learning*, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer International Publishing, 2016b.

Hein, M., Andriushchenko, M., and Bitterwolf, J. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41–50, 2019.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations*, 2019.

Hendrycks, D. and Gimpel, K. Early methods for detecting adversarial images. In *International Conference on Learning Representations Workshops*, 2017.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pp. 2712–2721. PMLR, 2019.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. AugMix: A simple data processing method to improve robustness and uncertainty. *International Conference on Learning Representations*, 2020.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Hossain, M. T., Teng, S. W., Lu, G., Rahman, M. A., and Sohel, F. Anti-aliasing deep image classifiers using novel depth adaptive blurring and activation function. *Neurocomputing*, 536:164–174, 2023.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017a.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017b.

Huang, H., Wang, Y., Erfani, S., Gu, Q., Bailey, J., and Ma, X. Exploring architectural ingredients of adversarially robust deep neural networks. *Advances in Neural Information Processing Systems*, 34:5545–5559, 2021.

Huang, L., Zhang, C., and Zhang, H. Self-adaptive training: beyond empirical risk minimization. *Advances in Neural Information Processing Systems*, 33:19365–19376, 2020.

Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pp. 2137–2146. PMLR, 2018.

Jähne, B. *Digital image processing*. Springer Science & Business Media, 2005.

Jolliffe, I. *Principal Component Analysis*. Springer Verlag, 1986.

Jung, S. and Keuper, M. Spectral distribution aware image generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 1734–1742, 2021.

Kamnitsas, K., Ledig, C., Newcombe, V. F., Simpson, J. P., Kane, A. D., Menon, D. K., Rueckert, D., and Glocker, B. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical image analysis*, 36:61–78, 2017.

Karras, T., Aittala, M., Laine, S., Härkönen, E., Hellsten, J., Lehtinen, J., and Aila, T. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., and Shah, M. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

Kim, B., Abuadbba, A., Gao, Y., Zheng, Y., Ahmed, M. E., Nepal, S., and Kim, H. Decamouflage: A framework to detect image-scaling attacks on cnn. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 63–74, 2021a. doi: 10.1109/DSN48987.2021.00023.

Kim, H., Lee, W., and Lee, J. Understanding catastrophic overfitting in single-step adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8119–8127, 2021b.

Kireev, K., Andriushchenko, M., and Flammarion, N. On the effectiveness of adversarial training against common corruptions. In *Uncertainty in Artificial Intelligence*, pp. 1012–1021. PMLR, 2022.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

Kurakin, A., Goodfellow, I. J., and Bengio, S. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

Ladwig, D., Spitznagel, M., Vaillant, J., Dorer, K., and Keuper, J. Ai-guided noise reduction for urban geothermal drilling. In *Proceedings of the Upper-Rhine Artificial Intelligence Symposium*, 2024.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in Neural Information Processing Systems*, 31, 2018.

Li, Q., Shen, L., Guo, S., and Lai, Z. Wavelet integrated cnns for noise-robust image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7245–7254, 2020.

Li, Q., Shen, L., Guo, S., and Lai, Z. Wavecnet: Wavelet integrated cnns to suppress aliasing effect for noise-robust image classification. *IEEE Transactions on Image Processing*, 30:7074–7089, 2021. doi: 10.1109/tip.2021.3101395.

Li, X. and Li, F. Adversarial examples detection in deep networks with convolutional filter statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5764–5772, 2017.

Li, Z. and Hoiem, D. Improving confidence estimates for unfamiliar examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2686–2695, 2020.

Liu, S., Chen, T., Chen, X., Chen, X., Xiao, Q., Wu, B., Kärkkäinen, T., Pechenizkiy, M., Mocanu, D. C., and Wang, Z. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In *International Conference on Learning Representations*, 2023.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012–10022, 2021.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., and Guo, B. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022a.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022b.

Lohn, A. J. Downscaling attack and defense: Turning what you see back into what you get. *arXiv preprint arXiv:2010.02456*, 2020.

Lord, N. A., Mueller, R., and Bertinetto, L. Attacking deep networks with surrogate-based adversarial black-box methods is easy. In *International Conference on Learning Representations*, 2022.

Lorenz, P., Harder, P., Straßel, D., Keuper, M., and Keuper, J. Detecting autoattack perturbations in the frequency domain. In *International Conference on Machine Learning Workshop on Adversarial Machine Learning*, 2021.

Lukasik, J., Gavrikov, P., Keuper, J., and Keuper, M. Improving native cnn robustness with filter frequency regularization. *Transactions on Machine Learning Research*, 2023.

Ma, T., Dalca, A. V., and Sabuncu, M. R. Hyper-convolution networks for biomedical image segmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1933–1942, 2022.

Ma, T., Wang, A. Q., Dalca, A. V., and Sabuncu, M. R. Hyper-convolutions via implicit kernels for medical image analysis. *Medical Image Analysis*, 86:102796, 2023. ISSN 1361-8415.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

Maiya, S. R., Ehrlich, M., Agarwal, V., Lim, S.-N., Goldstein, T., and Shrivastava, A. A frequency perspective of adversarial robustness. *arXiv preprint arXiv:2111.00861*, 2021.

Mathieu, M., Henaff, M., and LeCun, Y. Fast training of convolutional networks through ffts. In *International Conference on Learning Representations*, 2013.

Medi, T., Grabinski, J., and Keuper, M. Towards class-wise robustness analysis. In *The 2nd Workshop and Challenges for Out-of-Distribution Generalization in Computer Vision, ICCV Workshops*, 2023.

Menghani, G. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. In *International Conference on Learning Representations*, 2017.

Meyer, S., Djelouah, A., McWilliams, B., Sorkine-Hornung, A., Gross, M., and Schroers, C. Phasenet for video frame interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Moon, J., Kim, J., Shin, Y., and Hwang, S. Confidence-aware learning for deep neural networks. In *International Conference on Machine Learning*, pp. 7034–7044. PMLR, 2020.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Müller, P., Braun, A., and Keuper, M. Classification robustness to common optical aberrations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pp. 3632–3643, 2023.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? *Advances in Neural Information Processing Systems*, 32, 2019.

Mund, D., Triebel, R., and Cremers, D. Active online confidence boosting for efficient object classification. In *IEEE International Conference on Robotics and Automation*, pp. 1367–1373, 2015. doi: 10.1109/ICRA.2015.7139368.

Naeini, M. P., Cooper, G., and Hauskrecht, M. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 4. Granada, 2011.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 427–436, 2015.

Oppenheim, A. V. *Discrete-time signal processing*. Pearson Education India, 1999.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019.

Pan, H., Chen, Y., Niu, X., Zhou, W., and Li, D. Learning convolutional neural networks in the frequency domain. *arXiv preprint arXiv:2204.06718*, 2022.

Pang, T., Yang, X., Dong, Y., Xu, K., Zhu, J., and Su, H. Boosting adversarial training with hypersphere embedding. *Advances in Neural Information Processing Systems*, 33:7779–7792, 2020.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Peng, C., Zhang, X., Yu, G., Luo, G., and Sun, J. Large kernel matters – improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

Pintea, S. L., Tömen, N., Goes, S. F., Loog, M., and van Gemert, J. C. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, 30:8342–8353, 2021.

Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. 1999.

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Prabhu, K. M. *Window functions and their applications in signal processing*. Taylor & Francis, 2014.

Pratt, H., Williams, B., Coenen, F., and Zheng, Y. Fcnn: Fourier convolutional neural networks. In *Machine Learning and Knowledge Discovery in Databases*, pp. 786–798, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71249-9.

Qin, Y., Wang, X., Beutel, A., and Chi, E. Improving calibration through the relationship with adversarial robustness. *Advances in Neural Information Processing Systems*, 34:14358–14369, 2021.

Rade, R. and Moosavi-Dezfooli, S.-M. Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *International Conference on Machine Learning Workshop on Adversarial Machine Learning*, 2021.

Rahim, R., Shamsafar, F., and Zell, A. Separable convolutions for optimizing 3d stereo networks. In *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3208–3212. IEEE, 2021.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. In *International Conference on Learning Representations*, 2018.

Rao, Y., Zhao, W., Zhu, Z., Lu, J., and Zhou, J. Global filter networks for image classification. In *Advances in Neural Information Processing Systems*, 2021.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Data augmentation can improve robustness. In *Advances in Neural Information Processing Systems*, 2021.

Reed, C. J., Gupta, R., Li, S., Brockman, S., Funk, C., Clipp, B., Keutzer, K., Candido, S., Uyttendaele, M., and Darrell, T. Scale-mae: A scale-aware masked autoencoder for multiscale geospatial representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4088–4099, 2023.

Reed, S., Lee, H., Anguelov, D., Szegedy, C., Erhan, D., and Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020.

Rodríguez-Muñoz, A. and Torralba, A. Aliasing is a driver of adversarial attacks. *arXiv preprint arXiv:2212.11760*, 2022.

Romero, D. W., Bruintjes, R.-J., Tomczak, J. M., Bekkers, E. J., Hoogendoorn, M., and van Gemert, J. Flexconv: Continuous kernel convolutions with differentiable kernel sizes. In *International Conference on Learning Representations*, 2022a.

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. CKConv: Continuous kernel convolution for sequential data. In *International Conference on Learning Representations*, 2022b.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Rony, J., Hafemann, L. G., Oliveira, L. S., Ayed, I. B., Sabourin, R., and Granger, E. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4322–4330, 2019.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Saikia, T., Schmid, C., and Brox, T. Improving robustness against common corruptions with frequency biased models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10211–10220, 2021.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Sehwag, V., Wang, S., Mittal, P., and Jana, S. Hydra: Pruning adversarially robust neural networks. *Advances in Neural Information Processing Systems*, 33:19655–19666, 2020.

Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., and Mittal, P. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022.

Semmlow, J. L. and Griffel, B. *Biosignal and medical image processing*. CRC press, 2021.

Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learning*, pp. 2217–2225. PMLR, 2016.

Shannon, C. Communication in the presence of noise. *Proceedings of the IRE*, 37(1): 10–21, 1949. doi: 10.1109/JRPROC.1949.232969.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Sitawarin, C., Chakraborty, S., and Wagner, D. Sat: Improving adversarial training via curriculum-based loss smoothing. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pp. 25–36, 2021.

Sitzmann, V., Zollhöfer, M., and Wetzstein, G. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019.

Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7462–7473. Curran Associates, Inc., 2020.

Sosnovik, D. E., Wang, R., Dai, G., Reese, T. G., and Wedeen, V. J. Diffusion mr tractography of the heart. *Journal of Cardiovascular Magnetic Resonance*, 11(1):47, 2009.

Sosnovik, I., Szmaja, M., and Smeulders, A. Scale-equivariant steerable networks. In *International Conference on Learning Representations*, 2020.

Sridhar, K., Sokolsky, O., Lee, I., and Weimer, J. Improving neural network robustness via persistency of excitation. In *American Control Conference*, pp. 1521–1526. IEEE, 2022.

StatCounter. Mobile screen resolution stats worldwide, 2024. URL https://gs.statcounter.com/screen-resolution-stats/mobile/worldwide. Accessed: 2025-01-11.

Statista. Pixels per inch (ppi) of 4k and 8k tvs, by screen size, as of 2021, 2021. URL https://www.statista.com/statistics/1196348/pixels-per-inch-4k-8k-tv-by-screen-size/. Accessed: 2025-01-11.

Stutz, D., Hein, M., and Schiele, B. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *International Conference on Machine Learning*, pp. 9155–9166. PMLR, 2020.

Stutz, D., Hein, M., and Schiele, B. Relating adversarially robust generalization to flat minima. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7787–7797, 2021. doi: 10.1109/ICCV48922.2021.00771.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Tan, M. and Le, Q. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pp. 10096–10106. PMLR, 2021.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Tomani, C. and Buettner, F. Towards trustworthy predictions from deep neural networks with fast adversarial calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9886–9896, 2021.

Tomen, N. and van Gemert, J. C. Spectral leakage and rethinking the kernel size in cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5138–5147, 2021.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

Trabelsi, C., Bilaniuk, O., Zhang, Y., Serdyuk, D., Subramanian, S., Santos, J. F., Mehri, S., Rostamzadeh, N., Bengio, Y., and Pal, C. J. Deep complex networks. In *International Conference on Learning Representations*, 2018.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4489–4497, 2015.

Tu, C.-C., Ting, P., Chen, P.-Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.-J., and Cheng, S.-M. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 742–749, 2019.

Varshney, K. R. and Alemzadeh, H. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255, 2017.

Vasconcelos, C., Larochelle, H., Dumoulin, V., Roux, N. L., and Goroshin, R. An effective anti-aliasing approach for residual networks. *arXiv preprint arXiv:2011.10675*, 2020.

Vasconcelos, C., Larochelle, H., Dumoulin, V., Romijnders, R., Le Roux, N., and Goroshin, R. Impact of aliasing on generalization in deep convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10529–10538, 2021.

Vasilache, N., Johnson, J., Mathieu, M., Chintala, S., Piantino, S., and LeCun, Y. Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Video Electronics Standards Association (VESA). Coordinated video timings (cvt) standard. Patent, 2013. Available through VESA.

Wang, H., Wu, X., Huang, Z., and Xing, E. P. High-frequency component helps explain the generalization of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8684–8694, 2020a.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020b.

Wang, Z., Lan, Q., Huang, D., and Wen, M. Combining fft and spectral-pooling for efficient convolution neural network model. In *2nd International Conference on Artificial Intelligence and Industrial Engineering*, pp. 203–206. Atlantis Press, 2016.

Watanabe, T. and Wolf, D. F. Image classification in frequency domain with 2srelu: a second harmonics superposition activation function. *Applied Soft Computing*, 112: 107851, 2021.

Wenger, J., Kjellström, H., and Triebel, R. Non-parametric calibration for classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 178–190. PMLR, 2020.

Wightman, R. Pytorch image models, 2019.

Winograd, S. On computing the discrete fourier transform. *Mathematics of computation*, 32(141):175–199, 1978.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

Worrall, D. and Welling, M. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020.

Xiao, Q., Li, K., Zhang, D., and Jin, Y. Wolf in sheep's clothing - the downscaling attack against deep learning applications. *arXiv preprint arXiv:1712.07805*, 2017.

Xie, C. and Yuille, A. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2020.

Xu, Q., Zhang, R., Zhang, Y., Wang, Y., and Tian, Q. A fourier-based framework for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14383–14392, 2021.

Yang, Y. and Soatto, S. Fda: Fourier domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4085–4095, 2020.

Yin, D., Gontijo Lopes, R., Shlens, J., Cubuk, E. D., and Gilmer, J. A fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. *Advances in Neural Information Processing Systems*, 32, 2019a.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 2019b.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pp. 11278–11287. PMLR, 2020.

Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., and Kankanhalli, M. Geometry-aware instance-reweighted adversarial training. In *International Conference on Learning Representations*, 2021.

Zhang, R. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning*, 2019.

Zhou, Y., He, X., Huang, L., Liu, L., Zhu, F., Cui, S., and Shao, L. Collaborative learning of semi-supervised segmentation and classification for medical images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2079–2088, 2019.

Zou, X., Xiao, F., Yu, Z., Li, Y., and Lee, Y. J. Delving deeper into anti-aliasing in convnets. *International Journal of Computer Vision*, 131(1):67–81, 2023.

List of Figures

List of Tables

Appendix A

Supplementary for Chapter 4

Contents

In the following, we provide additional information and details that accompany chapter 4:

- Section A.1 - additional ECE bar plots presenting all individual robust models and their non-robust counterparts on CIFAR-10.
- Section A.2 - additional overconfidence bar plots presneting all individual robust models and their non-robust counterparts on CIFAR-100.
- Section A.3 - full empirical confidence distributions
- Section A.4 - additional precision recall curves for CIFAR-10 and CIFAR-100
- Section A.5 - additional ROC curves for ImageNet-1k
- Section A.6 - the overview of all trained models in our modelzoo

A.1 Additional ECE Bar Plots

In the following, we present the ECE bar plots of each model and its non-robust counterpart for CIFAR-10 in Figures A.1. Robust models exhibit a significantly lower ECE suggesting that they are better calibrated.

A.2 Additional Overconfidence Bar Plots

In the following, we present the overconfidence bar plots of each model and its non-robust counterpart for CIFAR-100 in Figures A.2. Most robust models exhibit significantly lower overconfidence, however not all models especially those under attack.

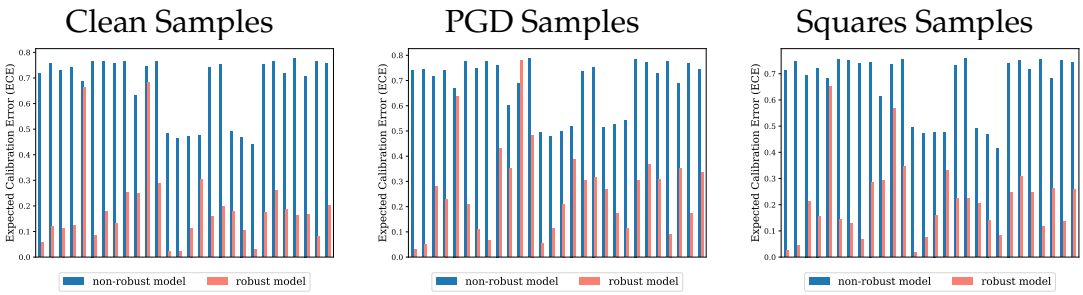


FIGURE A.1: **Robut Models have a Lower ECE on CIFAR-10** ECE (lower is better) bar plots of robust models and their non-robust counterparts trained on CIFAR-10. Non-robust models exhibit a much higher ECE than their robust counterparts.

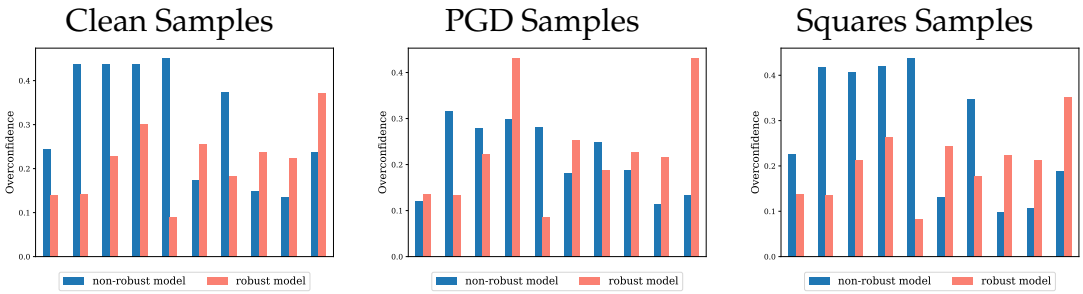


FIGURE A.2: **Overconfidence of Robust vs. Non-Robust Models on CIFAR-100** Overconfidence (lower is better) bar plots of the robust models and their respective non-robust counterparts trained on CIFAR-100. For CIFAR-100 most robust models are less overconfident on clean and Squares samples than their non-robust counterparts. However, on PGD samples more robust models are overconfident than their non-robust counterparts.

A.3 Empirical Confidence Distributions

In the following, we present the empirical confidence distribution of all robust models and their non-robust counterparts for CIFAR-10 and CIFAR-100 in Figures A.3 and A.4, respectively. Each row contains the robust and non-robust counterpart and their confidence distributions on the clean samples and the perturbed samples by PGD and Squares.

A.4 Additional Precision Recall Curves

In the following, we present the precision recall curves for CIFAR-10 and CIFAR-100 in Figures A.5 and A.6, respectively. We can observe similar results as presented in the ROC curves in Chapter 4, the non-robust models are slightly better on clean samples and fail completely under attack. For Squares samples, both the robust models perform slightly better.

When considering the precision recall curves between the confidence of clean correct samples and perturbed wrong samples (Figure A.7), we can observe that robust models are indeed able to detect adversarial samples based on thresholding the confidence of the model.

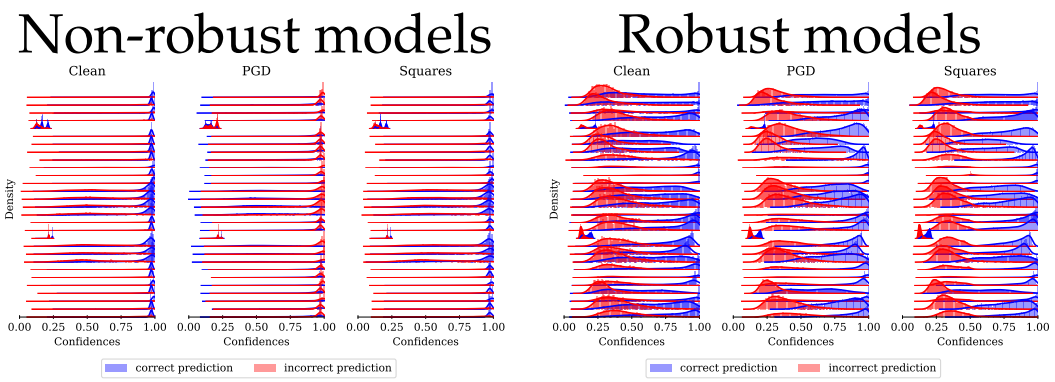


FIGURE A.3: **Robust Models are Better Calibrated on CIFAR-10.** Density plots for robust and non-robust models on CIFAR-10, showing the models' confidence in their correct and incorrect predictions. Each row contains the same model, both adversarially (right) and standard (left) trained. Non-robust models exhibit high confidence in all of their predictions, even though they may be incorrect. Especially in the case of PGD samples, these models are highly confident in their incorrect predictions. In contrast, the robust models are better calibrated: they are confident in their correct predictions and less confident in their incorrect ones.

A.5 ROC curves for ImageNet-1k

Figure A.8 presents the ROC curve on the clean as well as the perturbed samples for the robust models and the baseline on ImageNet-1k. The difference between the robust models is barely visible.

A.6 Model Overview

In the following we provide the overview over all models trained and provided at `https://github.com/GeJulia/robustness_confidences_evaluation` with their names, the respective paper, the architecture as well as their performance measured in clean and robust accuracy.

The robust checkpoints provided by *RobustBench* [Croce et al. (2021)] are licensed under the MIT Licence. The clean models for ImageNet are provided by *timm* [Wightman (2019)] under the Apache 2.0 licence.

Paper	Dataset	Architecture	Adv. Trained Clean Acc.	Adv. Trained Robust Acc.	Norm. Trained Clean Acc.	Norm. Trained Robust Acc.
2020	cifar10	PreActResNet-18	79.84	43.93	94.51	0.0
2019a	cifar10	WideResNet-28-10	89.69	59.53	95.10	0.0
2020	cifar10	WideResNet-28-10	88.98	57.14	95.10	0.0
2020b	cifar10	WideResNet-28-10	87.50	56.29	95.10	0.0
2019	cifar10	WideResNet-28-10	87.11	54.92	95.35	0.0
2020	cifar10	WideResNet-34-20	85.34	53.42	95.46	0.0
2019b	cifar10	WideResNet-34-10	84.92	53.08	95.26	0.0

Continued on next page

Paper	Dataset	Architecture	Adv. Trained Clean Acc.	Adv. Trained Robust Acc.	Norm. Trained Clean Acc.	Norm. Trained Robust Acc.
2019	cifar10	ResNet-50	87.03	49.25	94.90	0.0
2020	cifar10	ResNet-50	86.04	51.56	86.50	0.0
2020	cifar10	WideResNet-34-10	83.48	53.34	95.26	0.0
2020	cifar10	WideResNet-34-20	85.14	53.74	76.30	0.0
2020	cifar10	PreActResNet-18	83.34	43.21	94.25	0.0
2020	cifar10	WideResNet-28-4	84.36	41.44	94.33	0.0
2019a	cifar10	WideResNet-34-10	87.20	44.83	95.26	0.0
2020	cifar10	WideResNet-34-10	84.52	53.51	95.26	0.0
2020	cifar10	WideResNet-28-10	88.25	60.04	95.10	0.0
2020	cifar10	WideResNet-34-10	85.36	56.17	95.64	0.0
2021a	cifar10	WideResNet-70-16	85.29	57.20	87.91	0.0
2021a	cifar10	WideResNet-70-16	91.10	65.88	87.91	0.0
2021a	cifar10	WideResNet-34-20	85.64	56.86	88.33	0.0
2021a	cifar10	WideResNet-28-10	89.48	62.80	88.20	0.0
2022	cifar10	WideResNet-34-10	85.85	59.09	95.64	0.0
2022	cifar10	ResNet-18	84.38	54.43	94.87	0.0
2021	cifar10	WideResNet-34-10	86.84	50.72	95.26	0.0
2022	cifar10	WideResNet-34-10	85.32	51.12	95.35	0.0
2021	cifar10	WideResNet-34-20	88.70	53.57	95.44	0.0
2021	cifar10	WideResNet-34-10	88.22	52.86	95.26	0.0
2021	cifar10	WideResNet-28-10	89.36	59.64	95.10	0.0
2021	cifar10	WideResNet-28-10	87.33	60.75	88.20	0.0
2021	cifar10	WideResNet-106-16	88.50	64.64	86.92	0.0
2021	cifar10	WideResNet-70-16	88.54	64.25	87.91	0.0
2021	cifar10	WideResNet-70-16	92.23	66.58	87.91	0.0
2022	cifar10	WideResNet-28-10	89.46	59.66	95.10	0.0
2022	cifar10	WideResNet-34-15	86.53	60.41	95.50	0.0
2021	cifar10	PreActResNet-18	83.53	56.66	89.01	0.0
2021	cifar10	PreActResNet-18	89.02	57.67	89.01	0.0
2021	cifar10	PreActResNet-18	86.86	57.09	89.01	0.0
2021	cifar10	WideResNet-34-10	91.47	62.83	88.67	0.0
2021	cifar10	WideResNet-28-10	88.16	60.97	88.20	0.0
2021	cifar10	WideResNet-34-R	90.56	61.56	95.60	0.0
2021	cifar10	WideResNet-34-R	91.23	62.54	95.60	0.0
2021	cifar10	ResNet-18	80.24	51.06	94.87	0.0
2021	cifar10	WideResNet-34-10	85.32	58.04	95.26	0.0
2021b	cifar10	WideResNet-70-16	88.74	66.11	87.91	0.0
2022	cifar10	WideResNet-28-10-PSSiLU	87.02	61.55	85.53	0.0
2021b	cifar10	WideResNet-28-10	87.50	63.44	88.20	0.0
2021b	cifar10	PreActResNet-18	87.35	58.63	89.01	0.0
2021	cifar10	WideResNet-34-10	85.21	56.94	95.64	0.0
2021	cifar10	WideResNet-34-20	86.03	57.71	95.29	0.0

Continued on next page

Paper	Dataset	Architecture	Adv. Trained Clean Acc.	Adv. Trained Robust Acc.	Norm. Trained Clean Acc.	Norm. Trained Robust Acc.
2021a	cifar100	WideResNet-70-16	60.86	30.03	60.56	0.0
2021a	cifar100	WideResNet-70-16	69.15	36.88	60.56	0.0
2021	cifar100	WideResNet-34-20	62.55	30.20	80.46	0.0
2021	cifar100	WideResNet-34-10	70.25	27.16	79.11	0.0
2021	cifar100	WideResNet-34-10	60.64	29.33	79.11	0.0
2022	cifar100	WideResNet-34-10	62.15	26.94	78.75	0.0
2020	cifar100	WideResNet-34-10	60.38	28.86	78.79	0.0
2021	cifar100	WideResNet-34-10	62.82	24.57	79.11	0.0
2019	cifar100	WideResNet-28-10	59.23	28.42	79.16	0.0
2020	cifar100	PreActResNet-18	53.83	18.95	76.18	0.0
2021	cifar100	WideResNet-70-16	63.56	34.64	60.56	0.0
2021	cifar100	WideResNet-28-10	62.41	32.06	61.46	0.0
2021	cifar100	PreActResNet-18	56.87	28.50	63.45	0.0
2021	cifar100	PreActResNet-18	61.50	28.88	63.45	0.0
2021	cifar100	PreActResNet-18	62.02	27.14	76.66	0.0
2021	cifar100	WideResNet-34-10	65.73	30.35	79.11	0.0
2021	cifar100	WideResNet-34-10	64.07	30.59	79.11	0.0
2020	imagenet	ResNet-50	55.62	26.24	80.37	0.0
2019	imagenet	ResNet-50	62.56	29.22	80.37	0.0
2020	imagenet	ResNet-50	64.02	34.96	80.37	0.0
2020	imagenet	ResNet-18	52.92	25.32	69.74	0.0
2020	imagenet	WideResNet-50-2	68.46	38.14	81.45	0.0

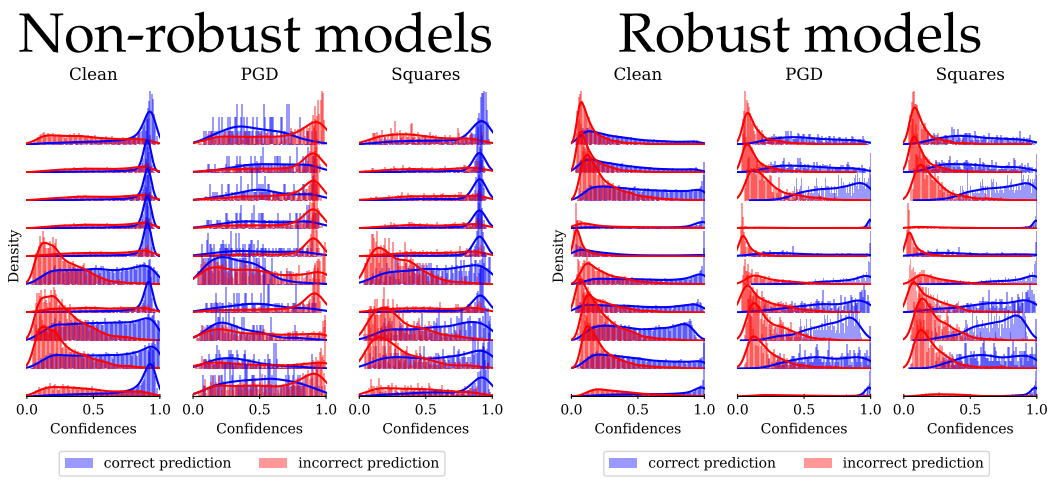


FIGURE A.4: **Robust Models are Better Calibrated on CIFAR-100.** Density plots for robust and non-robust models on CIFAR-100, showing the models' confidence in their correct and incorrect predictions. Each row contains the same model, both adversarially and standard trained. Non-robust models exhibit high confidence in all their predictions, even though they may be incorrect. Especially in the case of PGD samples, the models are highly confident in their incorrect predictions. In contrast, the robust models are better calibrated: they are mediocre confident in their correct predictions and fortunately much less confident in their incorrect ones.

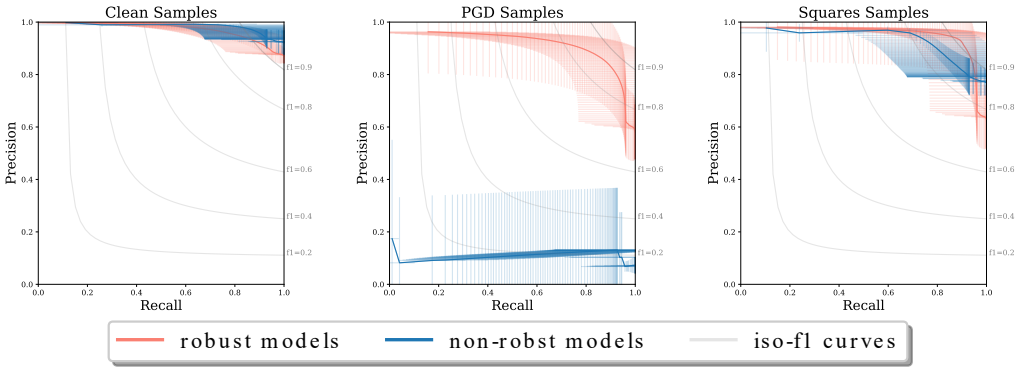


FIGURE A.5: **Robust Models can Disentangle Confidences between Correct and Incorrect Predictions on CIFAR-10.** Average precision-recall curve for all robust and non-robust models trained on CIFAR-10. Standard deviation is indicated by the error bars. For clean samples, the non-robust models can slightly better distinguish between correct and incorrect predictions based on the confidence of the predictions. The superiority of the robust models is evident in samples created by PGD, where the non-robust models are unable to distinguish. However, for samples created by Squares, the classification of correct and incorrect predictions based on confidence is almost equally feasible for robust and non-robust models.

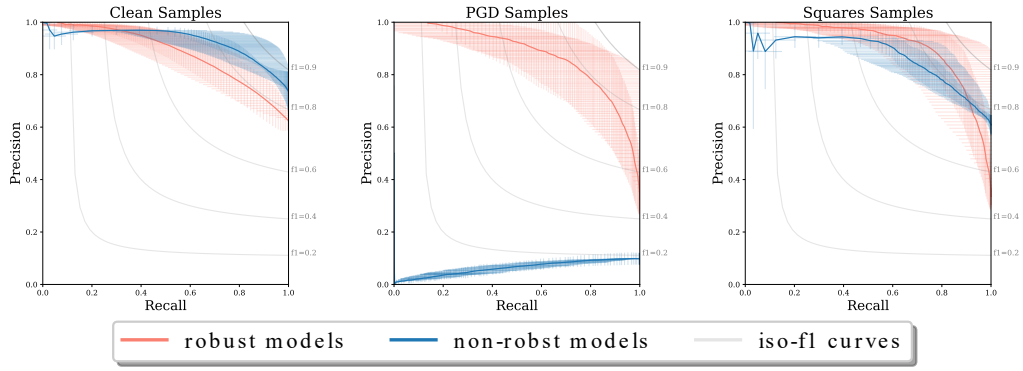


FIGURE A.6: **Robust Models can Disentangle Confidences between Correct and Incorrect Predictions on CIFAR-100.** Average precision-recall curve for all robust and non-robust models trained on CIFAR-100 for 1000 samples. Standard deviation is indicated by the error bars. For clean samples, the non-robust models can slightly better distinguish between correct and incorrect predictions based on the confidence of the predictions. The superiority of the robust models is clearly evident in samples created by PGD, where the non-robust models are unable to distinguish. However, for samples created by Squares, the classification of correct and incorrect predictions based on confidence is almost equally feasible for both robust and non-robust models.

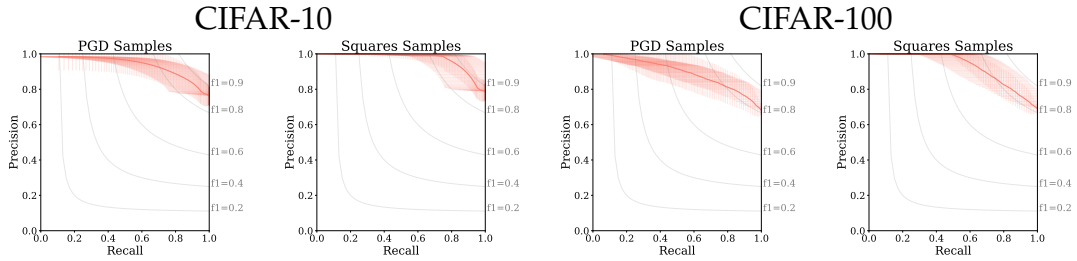


FIGURE A.7: **Robus Model Confidences as Detection for Adversarial Samples.** Precision-recall curve comparing the confidence of clean, correctly classified samples and perturbed, incorrectly classified samples on CIFAR-10 and CIFAR-100. The confidences of robust models can be used as thresholds for detecting adversarial attacks.

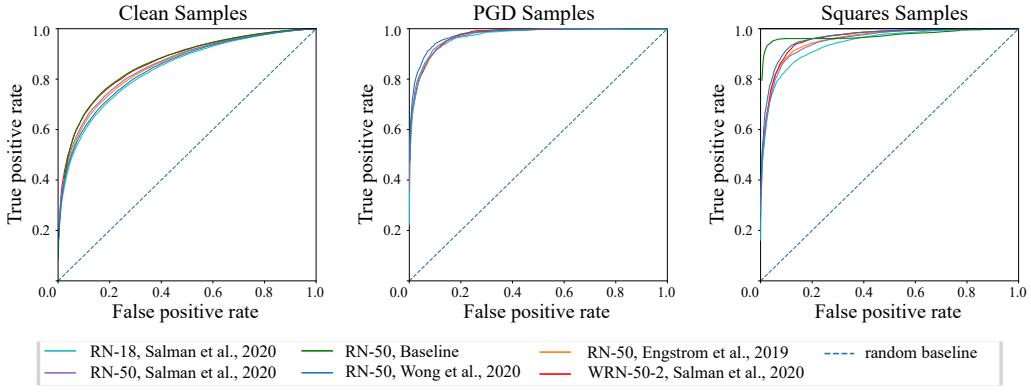


FIGURE A.8: **ROC curves for Robust vs. Non-Robust Models on ImageNet-1k.** ROC curves for the robust models and the non-robust baseline trained on ImageNet-1k provided on RobustBench [Croce et al. (2021)].

Appendix B

Supplementary for Chapter 6

Contents

In the following, we provide additional information and details that accompany Chapter 6:

- Section B.1 - additional confidence distributions for all our ASAP variant

B.1 Confidence Distributions

Following, we provide additional plots in Figure B.1 presenting the confidence distributions of all our proposed downsampling methods and the baseline with standard downsampling. While the baseline exhibits many very high overconfident predictions under attack our downsampling methods exhibit less high-confident false predictions.

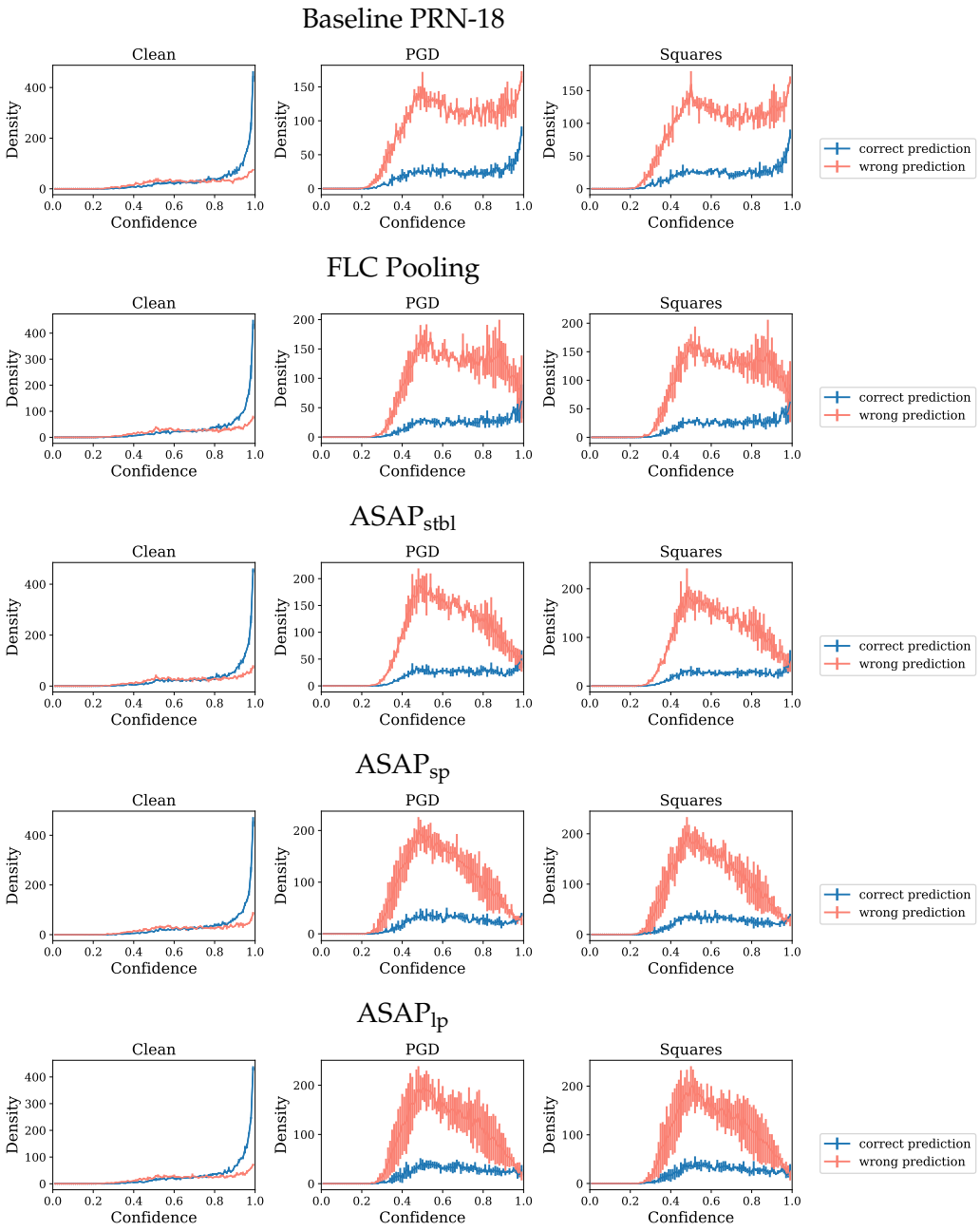


FIGURE B.1: **Aliasing-Free Downsampling Exhibits Desirable Confidence Distributions.** Mean confidence distribution and standard deviation for PRN-18 baseline, FLC Pooling, ASAP$_{stbl}$, ASAP$_{sp}$ and ASAP$_{lp}$ models trained with FGSM AT on three different seeds. While both downsampling methods perform equally well on clean samples, our models with FLC Pooling,ASAP$_{stbl}$, ASAP$_{sp}$ and ASAP$_{lp}$ exhibits fewer highly confident incorrect predictions on PGD and Square attacks. However, when examining multiple random seeds, the results deviate from the observations for a single model shown in Figure 4.8 in Chapter 4, underscoring the influence of training hyperparameters.

Appendix C

Supplementary for Chapter 7

Contents

In the following, we provide additional information and details that accompany chapter 7:

- Section C.1 - kernel mass evaluation for all networks on ImageNet-1k, ImageNet-100 and CIFAR-10
- Section C.2 - additional evaluation of non-square kernels
- Section C.3 - additional filter visualizations in the spatial domain
- Section C.4 - additional filter visualizations in the frequency domain

C.1 Additional Kernel Mass Evaluation

In the following, we present the kernel mass evaluations for all models on ImageNet-100 and additional models on CIFAR-10 in Figures C.1 and C.2, respectively. The findings are consistent across all models and datasets, as discussed in the Chapter 7. Models trained on large datasets like ImageNet-1k and ImageNet-100 accumulate the most kernel mass within 9×9 kernels, whereas for a smaller dataset like CIFAR-10, the most kernel mass is concentrated within 5×5 kernels. Hence, the standard kernel size of 3×3, as used in ResNet [He et al. (2016a)], DenseNet [Huang et al. (2017b)], and MobileNet [Sandler et al. (2018)], is insufficient. In contrast, ConvNeXt [Liu et al. (2022b)], which employs 7×7 kernels, represents a significant step in the right direction.

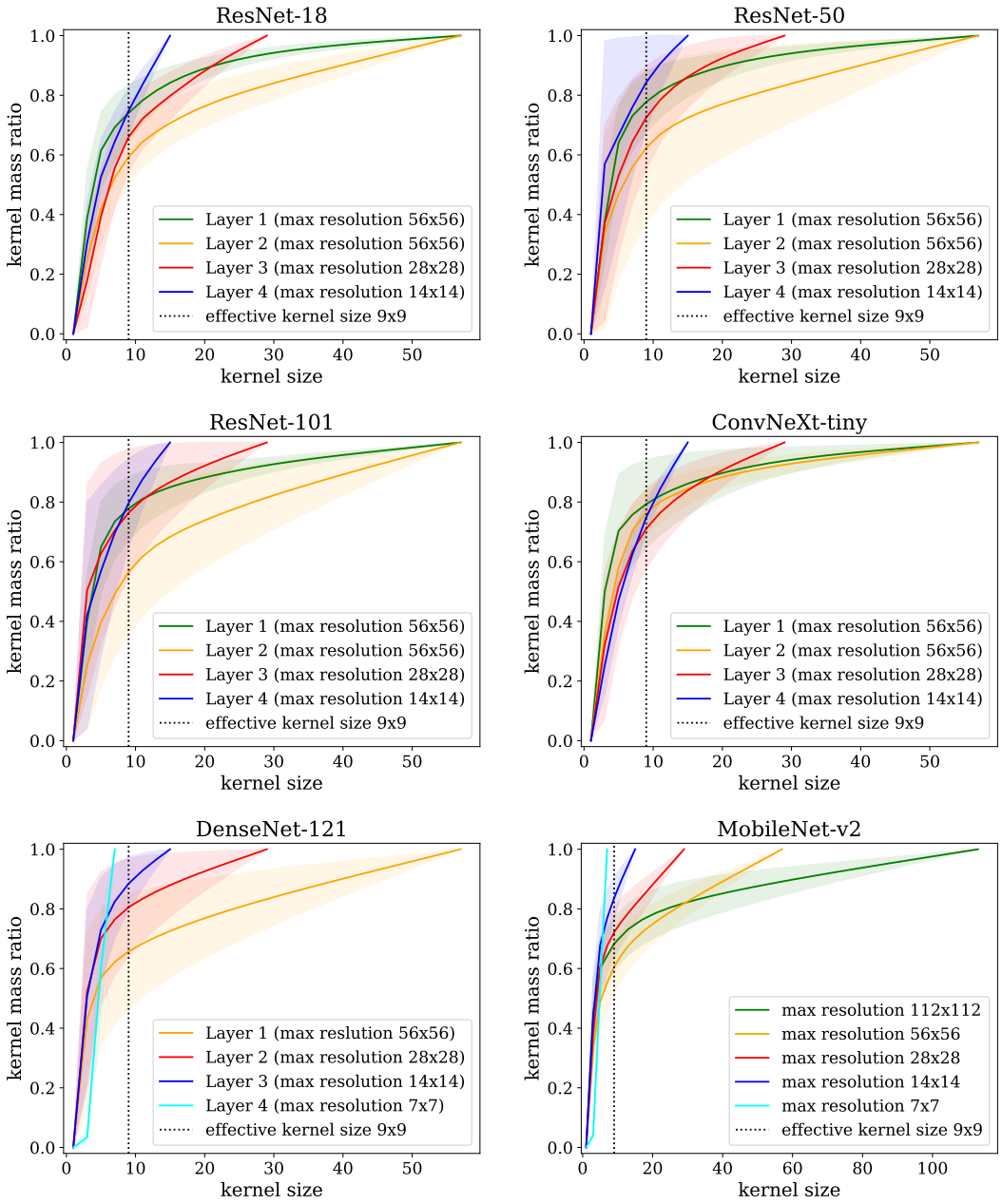


FIGURE C.1: **Effective Kernel Size Evaluation on ImageNet-100.** We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. For ResNet and ConvNeXt-tiny each layer encodes one resolution. Thus, the layers could be summarised (Layer 1 encoding 56×56, Layer 2 56×56, Layer 3 28×28 and Layer 4 14×14). For DenseNet-121 each layer can be summarised similarly, yet the after the first layer the feature maps are already downsampled resulting in the following: Layer 1 encoding 56×56, Layer 2 28×28, Layer 3 14×14 and Layer 4 7×7. However, for MobileNet-v2 the resolution is downsampled within a layer.

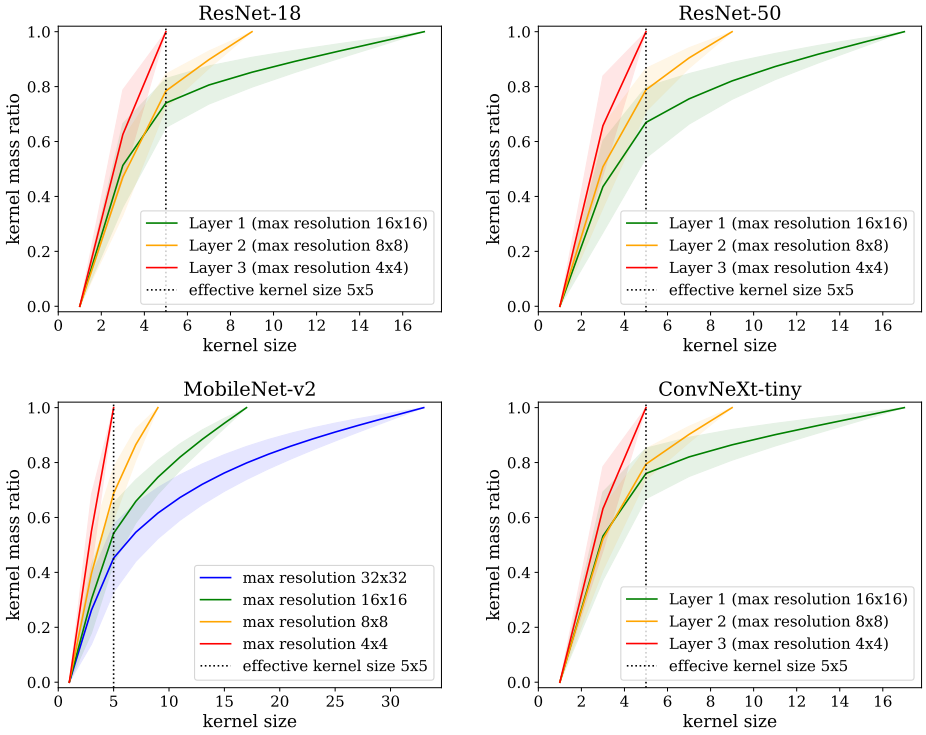


FIGURE C.2: **Effective Kernel Size Evaluation on CIFAR-10.** We plot the average ratio of the entire kernel mass contained within the limited spatial kernel size, where the x-axis denotes the width and height of the squared kernels. For ResNet models, each layer encodes one resolution. Thus, the layers could be summarised (Layer 1 encoding 16×16, Layer 2 8×8 and Layer 3 4×4). For ConvNeXt-tiny the first layer started with 32×32. However, for MobileNet-v2 the resolution is downsampled within a layer.

C.2 Additional Evaluation of Non-Square Kernels

In the following, we present the full results on evaluating the kernel shape of the learned kernels in the spatial domain. Figures C.3 and C.4 demonstrate that all models have a mean of roughly one indicating the same results presented in Section 7.3, the learned kernels are predominately square-shaped.

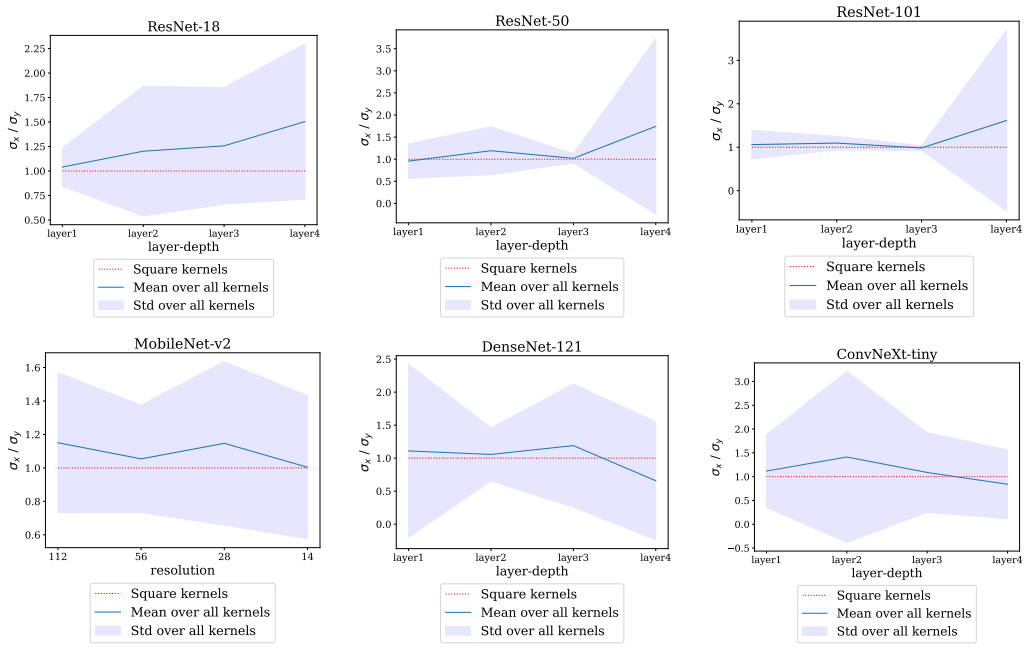


FIGURE C.3: **Analysis of Non-Square Kernel Shapes for our NIFF on ImageNet-1k.** Analysis of non-square kernel shapes on ImageNet-1k. We compare the variance σ_x and σ_y in x- and y-direction of a Gaussian fitted onto our learned spatial weights. The red dashed line indicates square-shaped kernels as the variance σ_x and σ_y are equal.

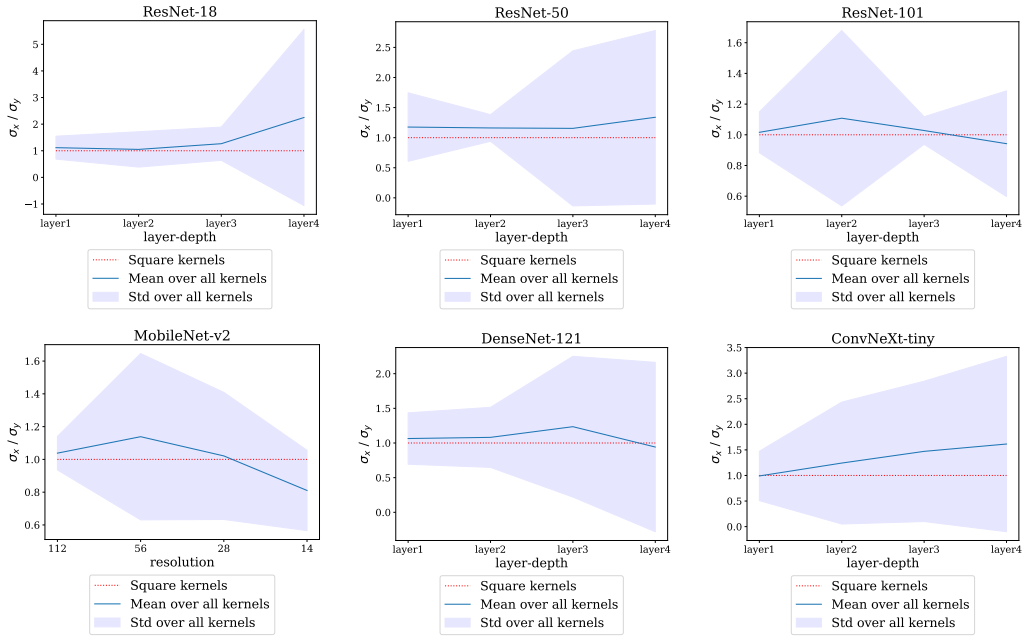


FIGURE C.4: **Analysis of Non-Square Kernel Shapes for our NIFF on ImageNet-100.** Analysis of non-square kernel shapes on ImageNet-100. We compare the variance σ_x and σ_y in x- and y-direction of a Gaussian fitted onto our learned spatial weights. The red dashed line indicates square-shaped kernels as the variance σ_x and σ_y are equal.

C.3 Additional Spatial Filter Visualization

In the following, we present the spatial filter visualizations. First, the zoomed-in PCA analysis for additional networks on ImageNet-1k, ImageNet-100, and CIFAR-10 is provided. Figures C.5, C.6 and C.7 present the PCA for ConvNeXt-tiny, DenseNet-121 and MobileNet-v2 trained on ImageNet-1k, respectively.

All spatial filter for the models trained on ImageNet-100 are presented in Figures C.8, C.9, C.10, and C.11 for ResNet-50, ConvNeXt-tiny, DenseNet-121, and MobileNet-v2, respectively.

For CIFAR-10, the additional MobileNet-v2 results are shown in Figure C.12. The findings are consistent with those reported in Chapter 7, showing that the highest variance for the kernels is expressed in small, well-localized kernels compared to the size they could have learned.

Second, we provide the original-size PCA analysis for all networks and high-resolution datasets which we zoomed in prior for better visibility to ensure transparency. The results for ImageNet-1k are presented in Figures C.13, C.14, C.15 and C.16 for ResNet-50, DenseNet-121, ConvNeXt-tiny and MobileNet-v2 respectively. The results for ImageNet-100 are presented in Figures C.17, C.18, C.19 and C.20 for ResNet-50, ConvNeXt-tiny, DenseNet-121, and MobileNet-v2, respectively.

Further, we provide the full visualization without zoom of the PCA analysis for the ResNet-18 trained on ImageNet-100 with additional zero padding before our NIFF in Figure C.21.

The full visualization without zoom of the PCA analysis for the ResNet-18 trained on ImageNet-100 with additional zero padding of our NIFF and the featuremaps to mimic a linear convolution in Figure C.22.

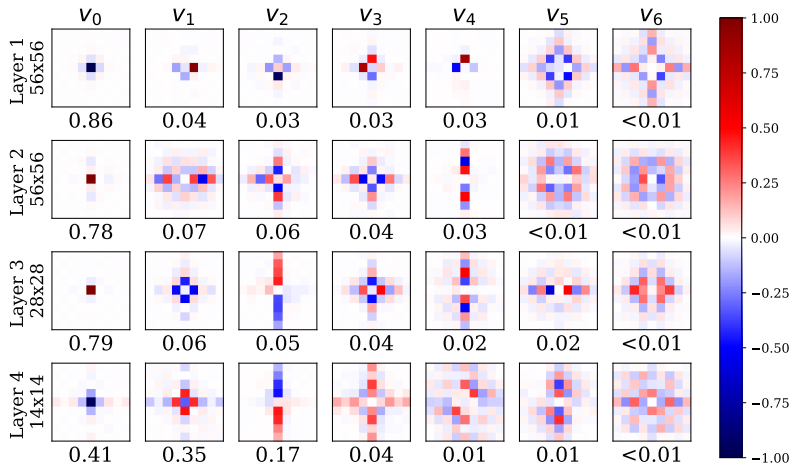


FIGURE C.5: **Spatial Filter Visualization for our NIFF ConvNeXt-tiny on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. ConvNeXt convolutions are standardly equipped with larger kernel sizes than usual (7×7). However, our analysis reveals that the network barely uses large filters if it gets the opportunity to learn large filters. The learned filters in the first and third layer mostly use small (3×3), well-localized filters.

Finally, we present a random selection of the spatial kernels in both their full size and zoomed-in views in Figure C.23 for ResNet-50 and Figure C.24 for ConvNeXt-tiny.

C.4 Additional Frequency Filter Visualization

In the following, we provide the visualizations of the full PCA analysis up to the fourth principle component of the multiplication weights of our ImageNet-1k NIFFs learned in the frequency domain. Figures C.25, C.26, C.28 and C.27 show the full PCA per layer for the learned element-wise multiplication weights for ResNet-50, DenseNet-121, ConvNeXt-tiny and MobileNet-V2 respectively.

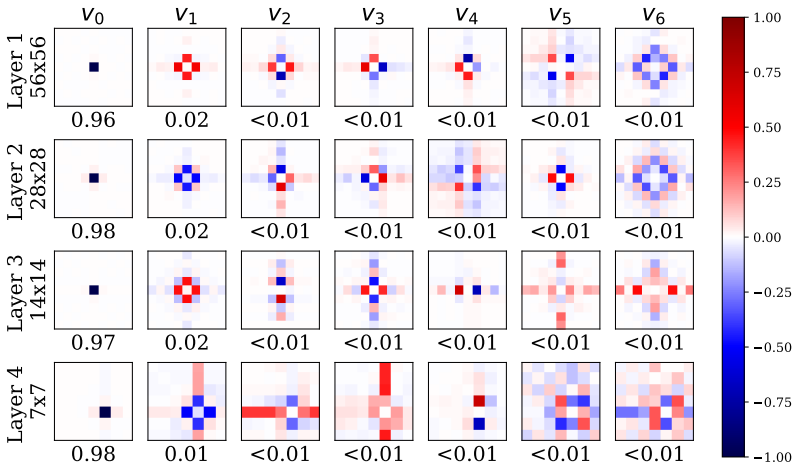


FIGURE C.6: **Spatial Filter Visualization for our NIFF DenseNet-121 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

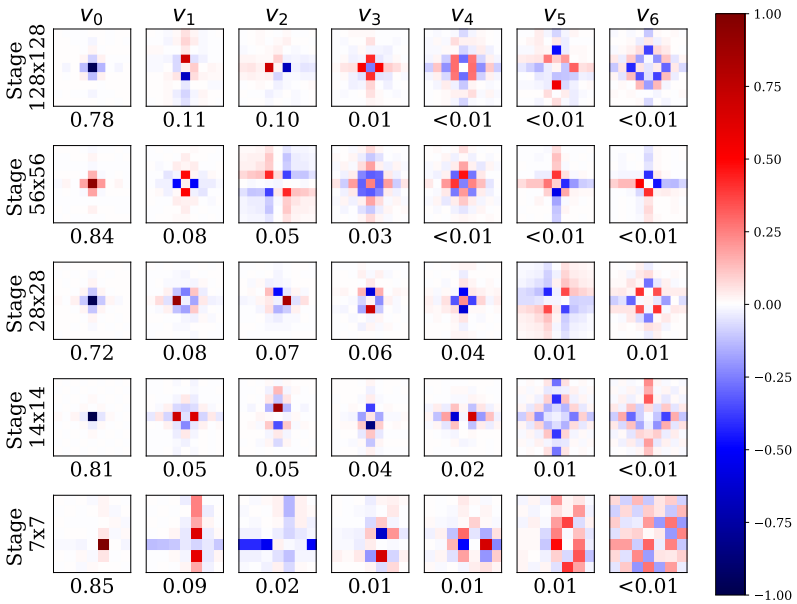


FIGURE C.7: **Spatial Filter Visualization for our NIFF MobileNet-v2 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-1k zoomed to 9×9. On the left, the maximal filter size for the corresponding stage is given. For MobileNet-v2 the feature maps are downsampled within a layer, thus the stages are combine by feature maps size rather than the layers. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

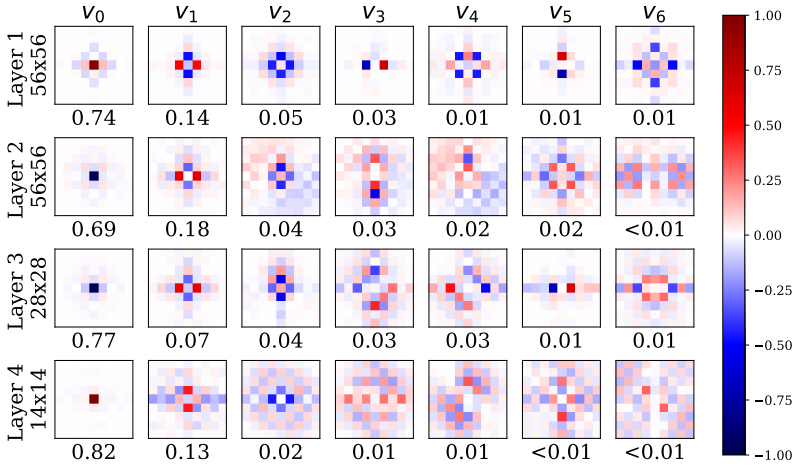


FIGURE C.8: **Spatial Filter Visualization for our NIFF ResNet-50 on ImageNet-100.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a really small kernel size although they could use a much bigger kernel.

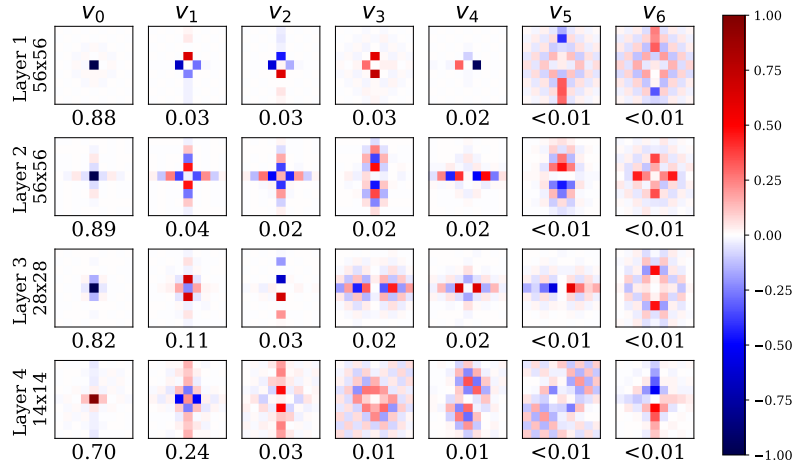


FIGURE C.9: **Spatial Filter Visualization for our NIFF ConvNeXt-tiny on ImageNet-100.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a really small kernel size although they could use a much bigger kernel.

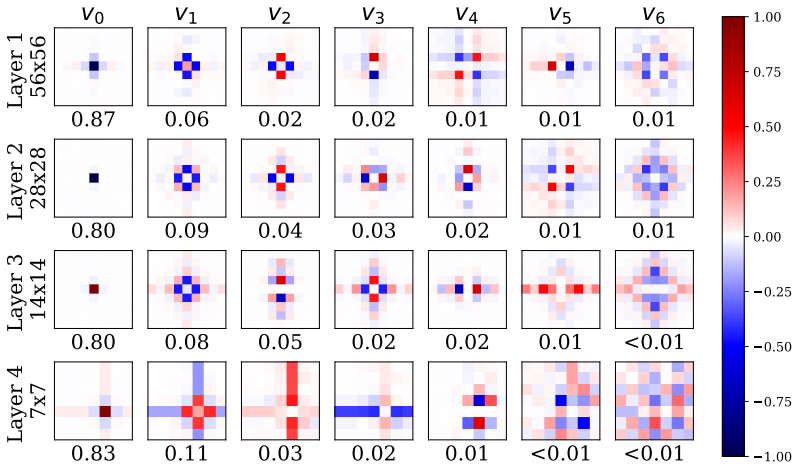


FIGURE C.10: **Spatial Filter Visualization for our NIFF DenseNet-121 on ImageNet-100.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

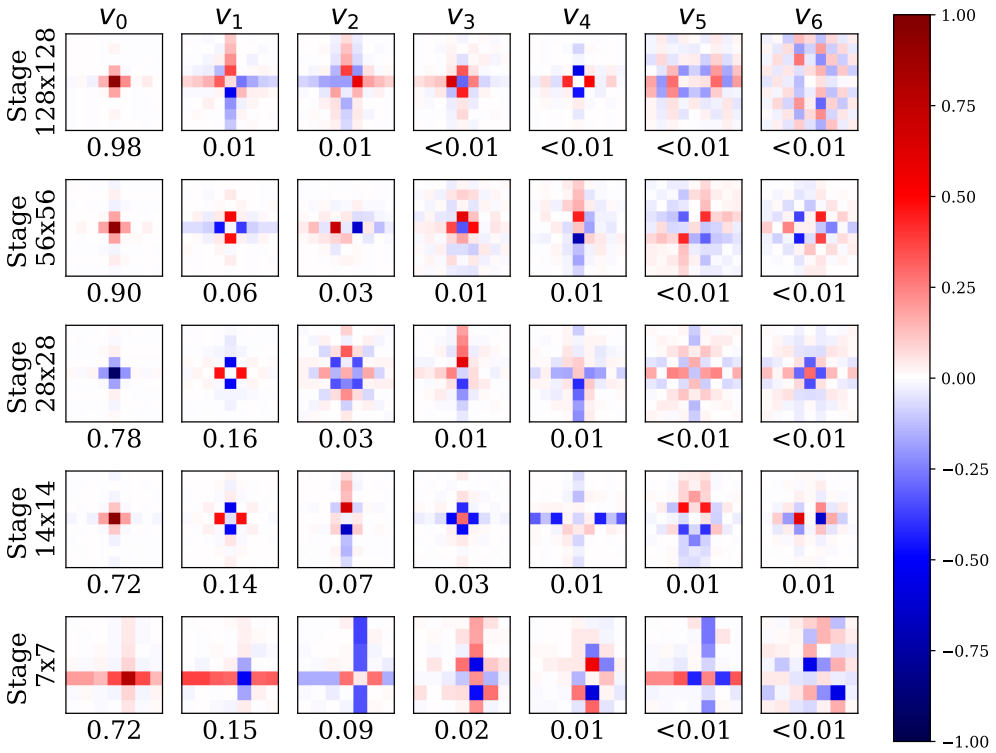


FIGURE C.11: **Spatial Filter Visualization for our NIFF MobileNet-v2 on ImageNet-100.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-100 zoomed to 9×9. On the left, the maximal filter size for the corresponding stage is given. For MobileNet-v2 the feature maps are downsampled within a layer, thus the stages are combine by feature maps size rather than the layers. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

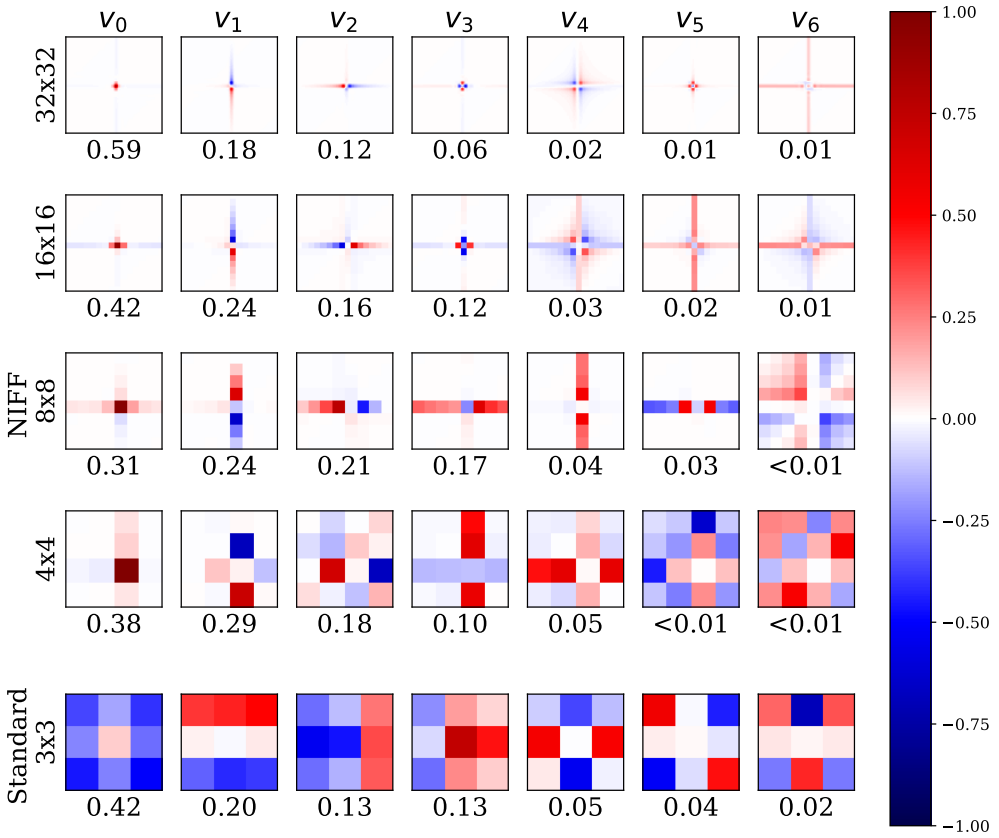


FIGURE C.12: **Spatial Filter Visualization for our NIFF MobileNet-v2 on CIFAR-10.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each resolution for the NIFF convolutions of a MobileNet-V2 trained on CIFAR-10 as well as the learned filters for the third layer of a standard MobileNet-V2 trained on CIFAR-10 (bottom row). On the right, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use much bigger kernels.

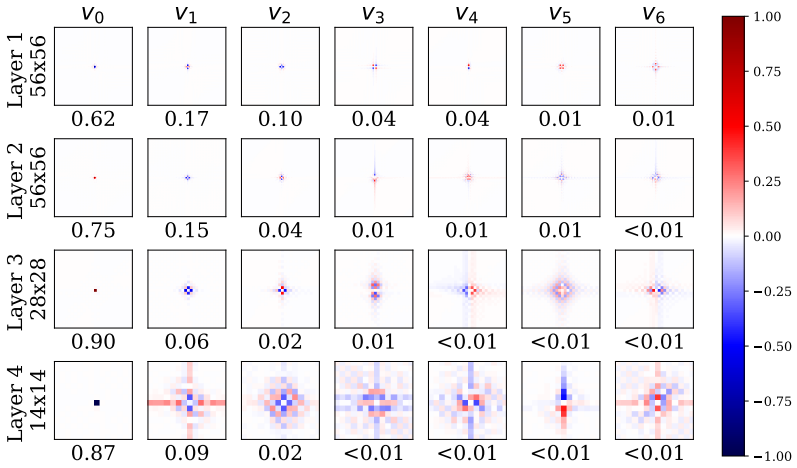


FIGURE C.13: **Spatial Filter Visualization for our NIFF ResNet-50 on ImageNet-1k without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-1k original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

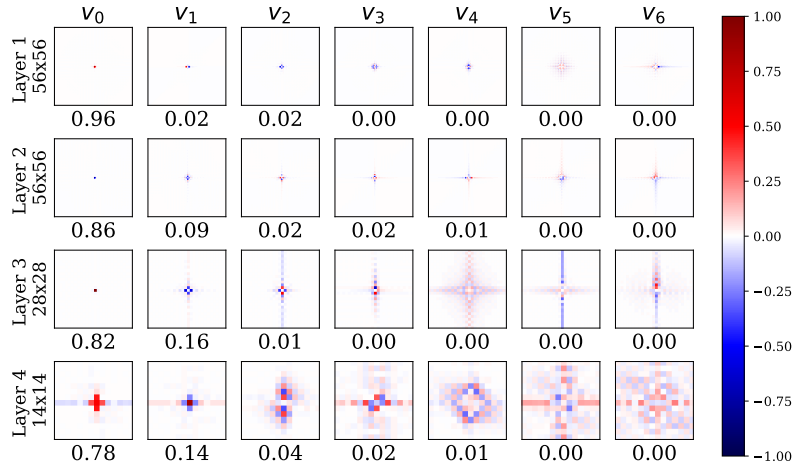


FIGURE C.14: **Spatial Filter Visualization for our NIFF ConvNeXt-tiny on ImageNet-1k without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-1k original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

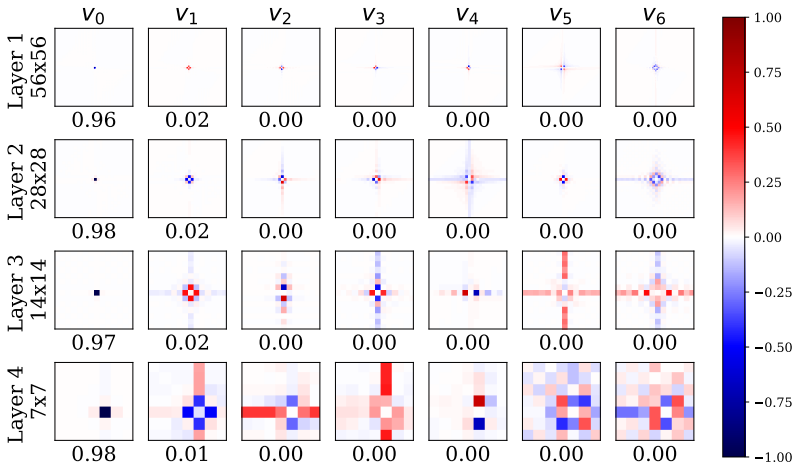


FIGURE C.15: **Spatial Filter Visualization for our NIFF DenseNet-121 on ImageNet-1k without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-1k original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

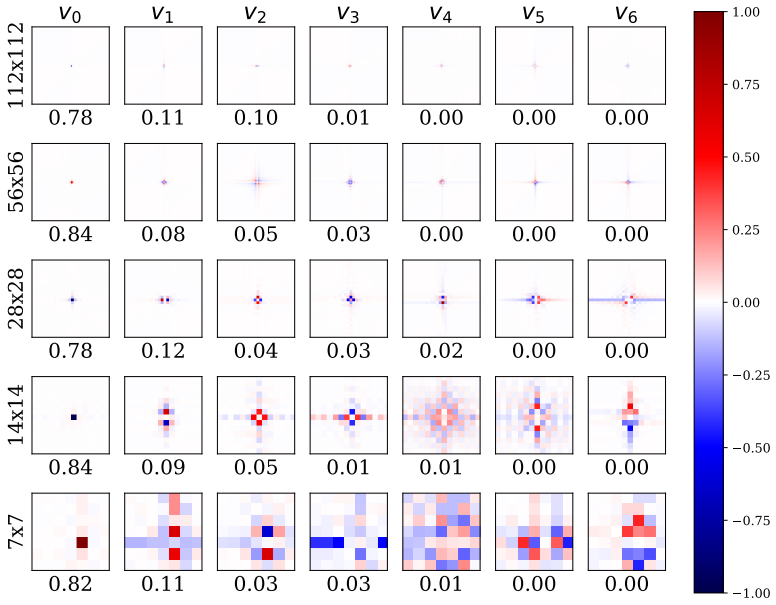


FIGURE C.16: **Spatial Filter Visualization for our NIFF MobileNet-v2 on ImageNet-1k without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-1k original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

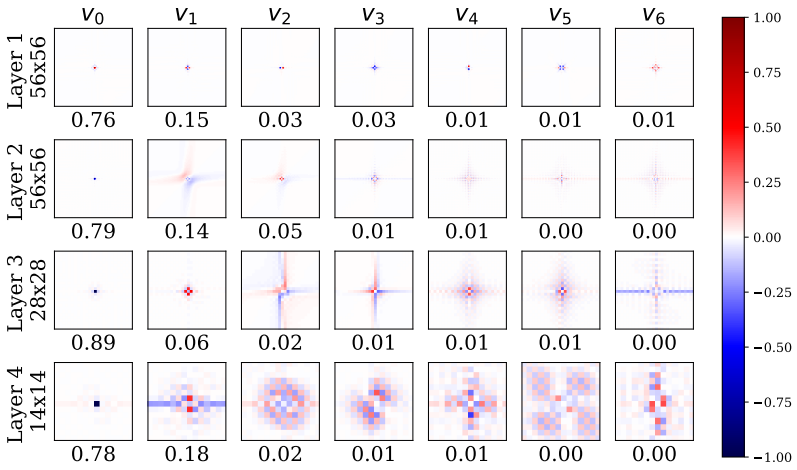


FIGURE C.17: **Spatial Filter Visualization for our NIFF ResNet-50 on ImageNet-100 without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ResNet-50 trained on ImageNet-100 original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

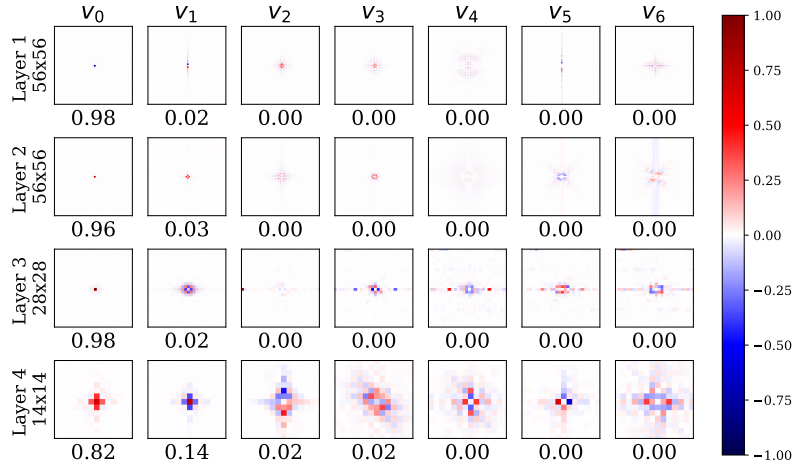


FIGURE C.18: **Spatial Filter Visualization for our NIFF ConvNeXt-tiny on ImageNet-100 without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a ConvNeXt-tiny trained on ImageNet-100 original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

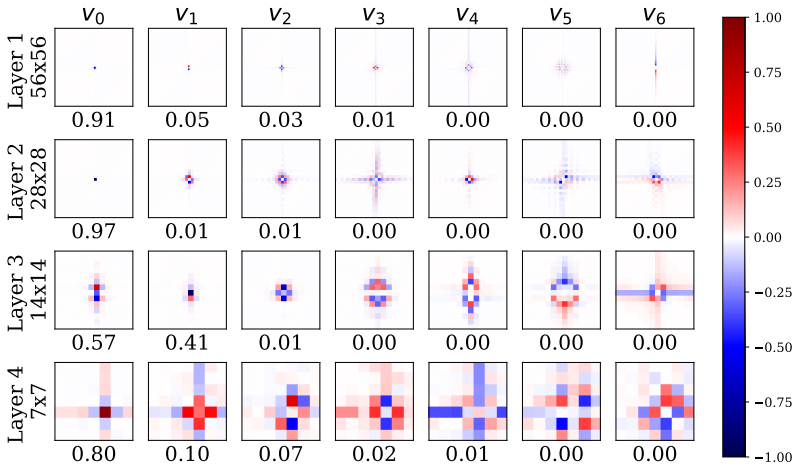


FIGURE C.19: **Spatial Filter Visualization for our NIFF DenseNet-121 on ImageNet-100 without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a DenseNet-121 trained on ImageNet-100 original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

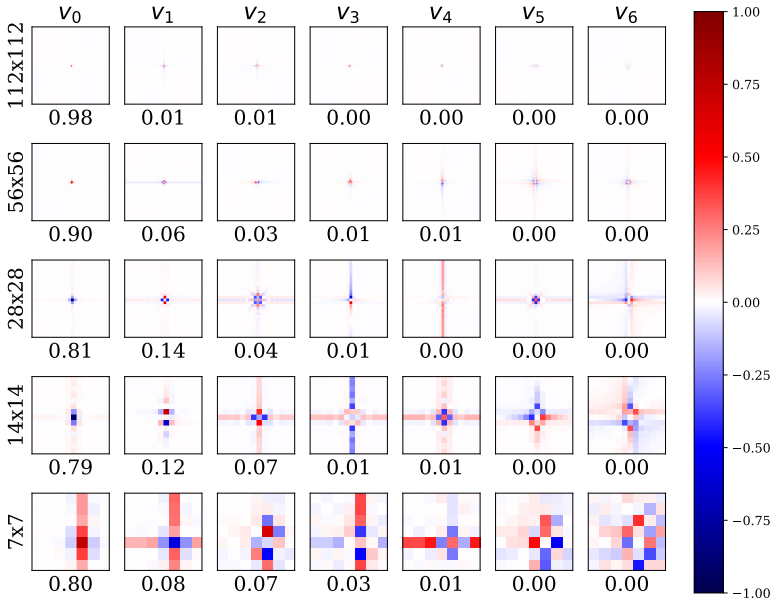


FIGURE C.20: **Spatial Filter Visualization for our NIFF MobileNet-v2 on ImageNet-100 without Zoom.** PCA basis and explained variance for each basis vector (below) of all spatial filters for each layer of a MobileNet-v2 trained on ImageNet-100 original size (not zoomed). On the left, the maximal filter size for the corresponding layer is given. We can see that most filters only use a well-localized, small kernel size although they could use a much bigger kernel.

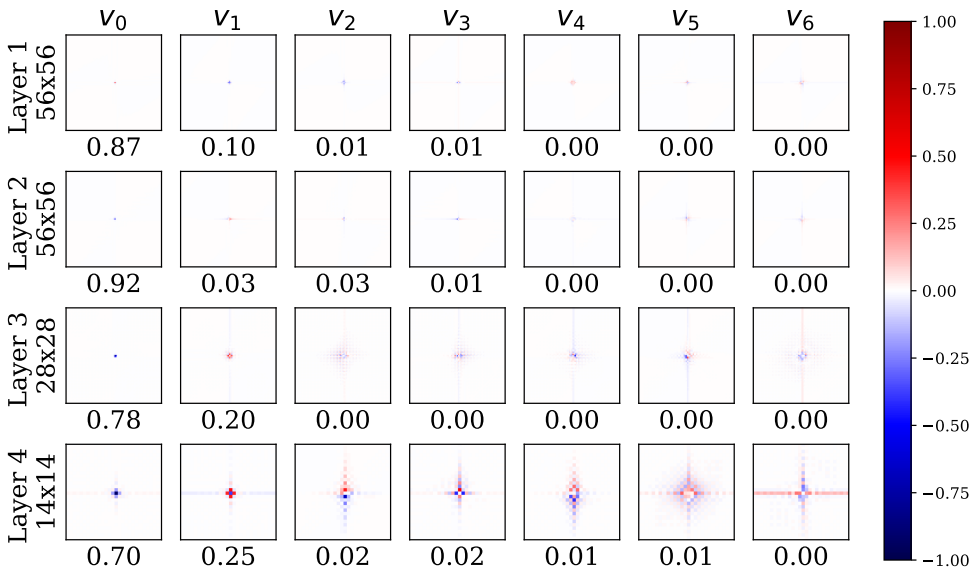


FIGURE C.21: **Spatial Filter Visualization for our NIFF ResNet-18 with additional Zero Padding on ImageNet-100 without Zoom.** Actual kernels in the spatial domain of a ResNet-18 with additional zero padding before our NIFF trained on ImageNet-100. Still, most kernels exhibit well-localized, small spatial kernels.

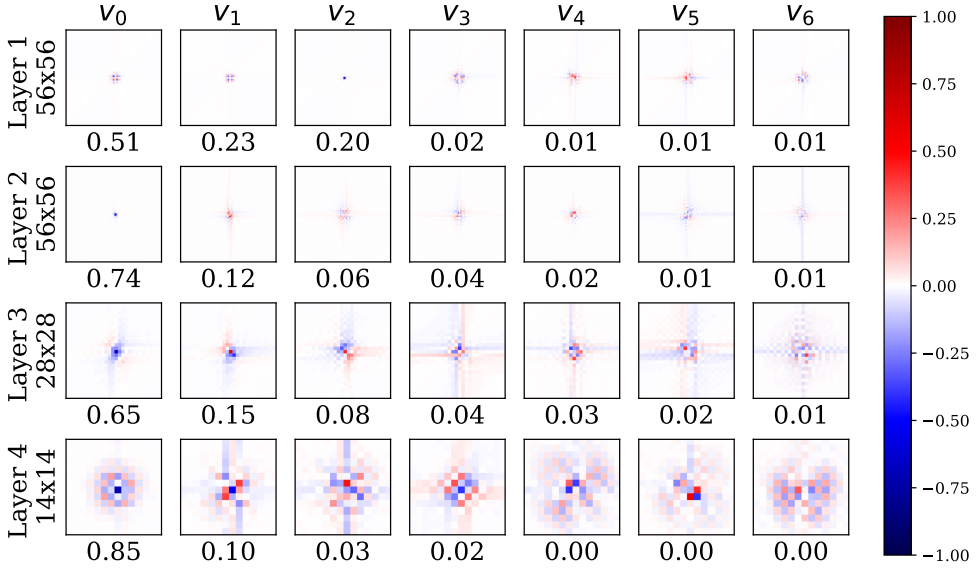


FIGURE C.22: **Spatial Filter Visualization for our NIFF ResNet-18 Mimic Linear Convolutions on ImageNet-100 without Zoom.** Actual kernels in the spatial domain of a ResNet-18 which mimics linear convolutions with our NIFF to mimic linear convolutions trained on ImageNet-100. Still, most kernels exhibit well-localized, small spatial kernels. However, they are slightly larger than the kernels learned without padding and cropping.

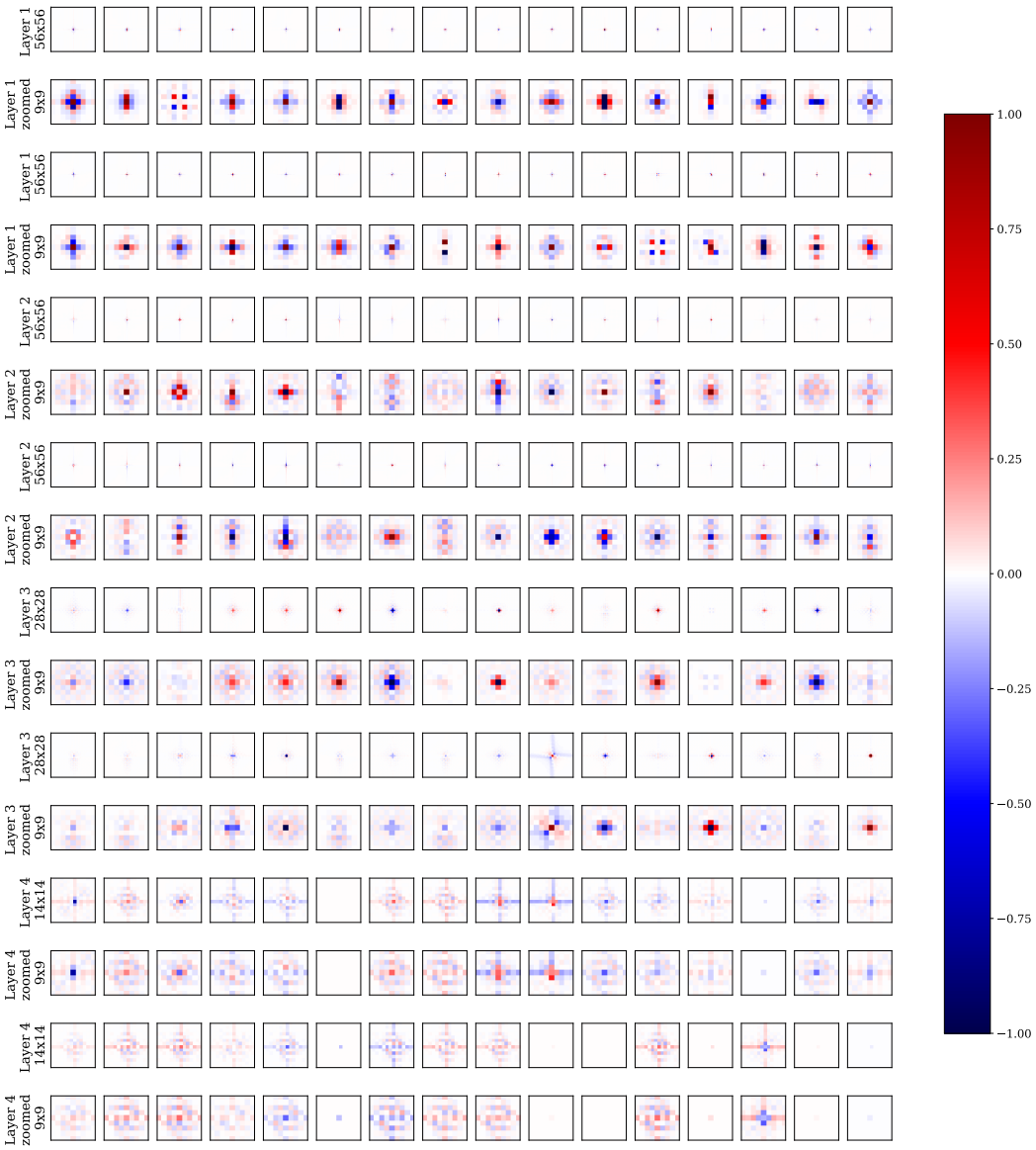


FIGURE C.23: **Overview of random Spatial Kernels of our NIFF ResNet-50 on ImageNet-1k.** Actual kernels in the spatial domain of a ResNet-50 including our NIFF trained on ImageNet-1k. We plot for each kernel the zoomed-in (9×9) version below for better visibility. Overall, most kernels exhibit well-localized, small spatial kernels.

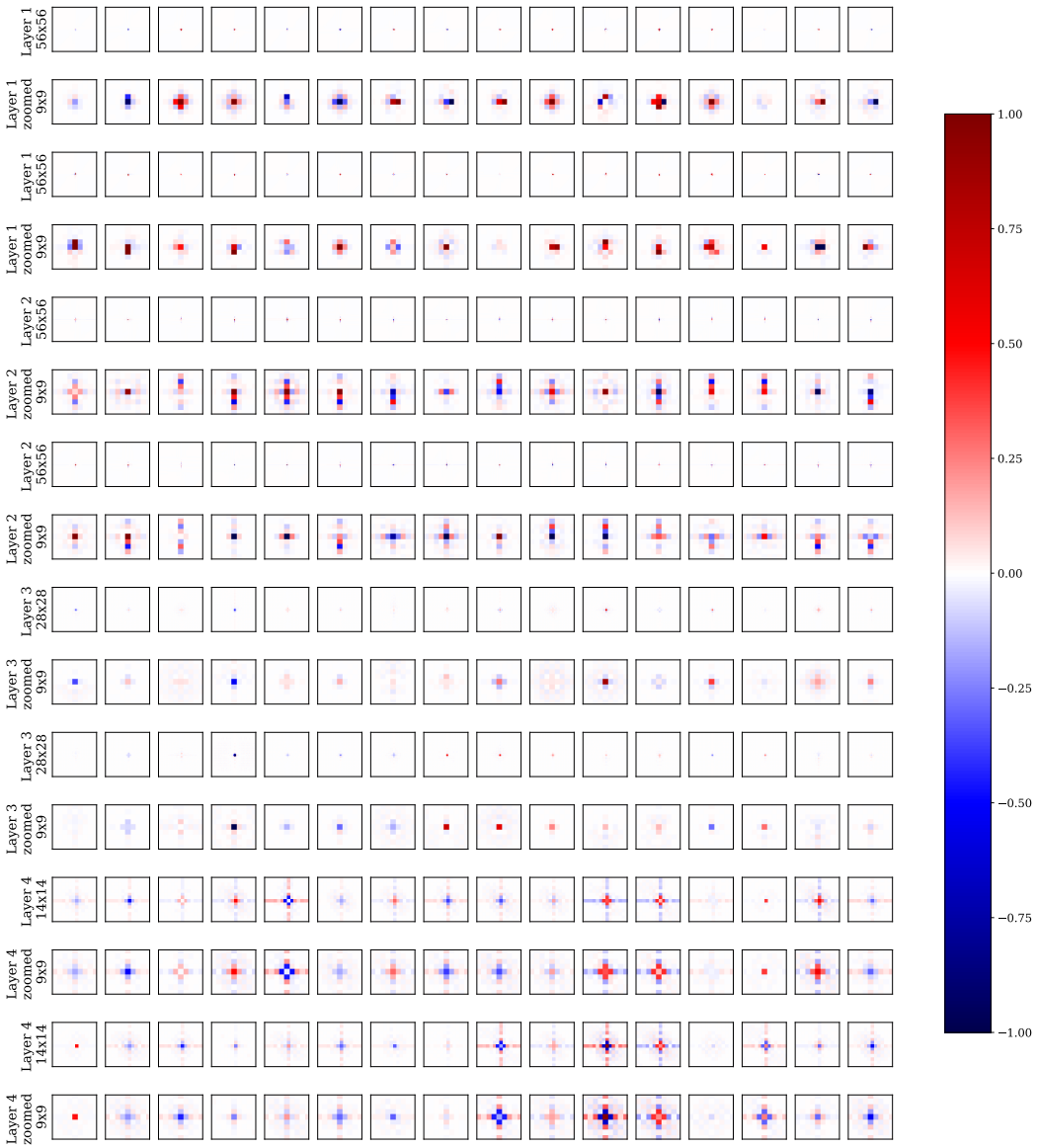


FIGURE C.24: **Overview of random Spatial Kernels of our NIFF ConvNeXt-tiny on ImageNet-1k.** Actual kernels in the spatial domain of a ConvNeXt-tiny including our NIFF trained on ImageNet-1k. We plot for each kernel the zoomed-in (9×9) version below for better visibility. Overall, most kernels exhibit well-localized, small spatial kernels.

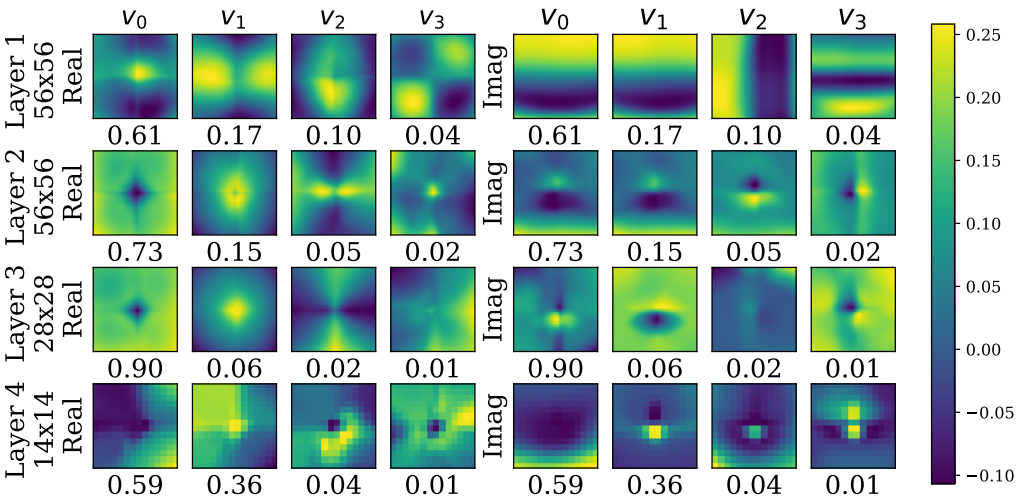


FIGURE C.25: **Frequency Filter Visualization for our NIFF ResNet-50 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a ResNet-50 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

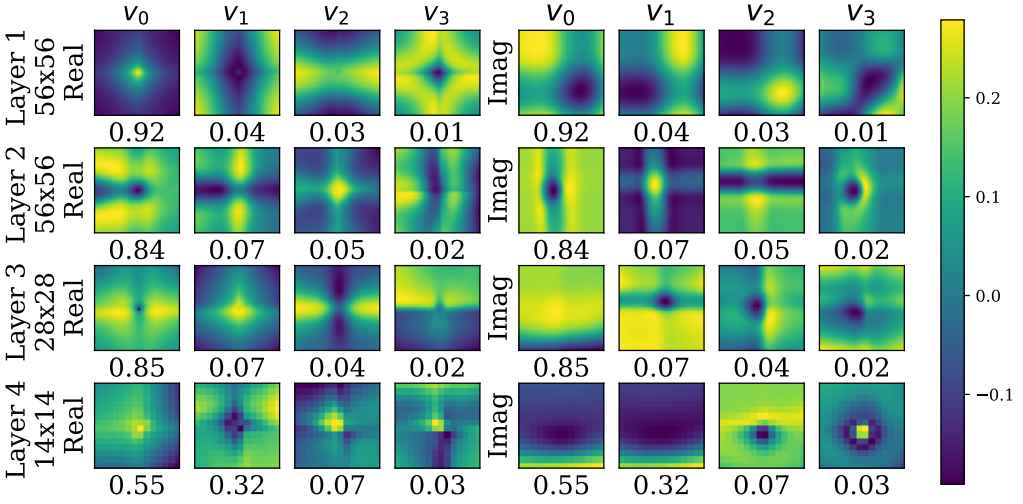


FIGURE C.26: **Frequency Filter Visualization for our NIFF ConvNeXt-tiny on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a ConvNeXt-tiny trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

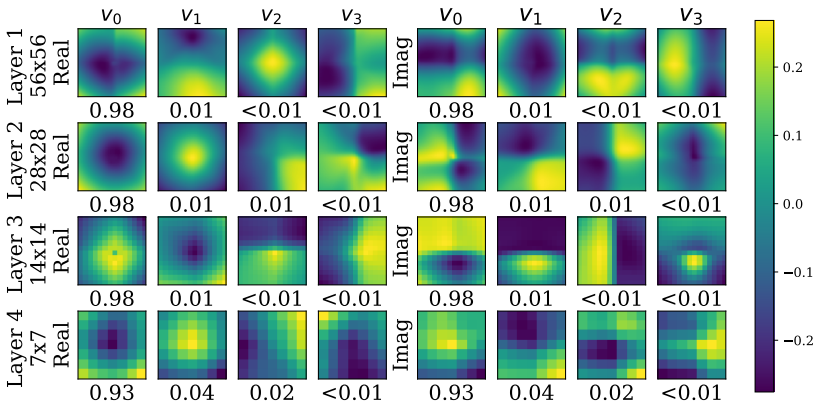


FIGURE C.27: **Frequency Filter Visualization for our NIFF DenseNet-121 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a DenseNet-121 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.

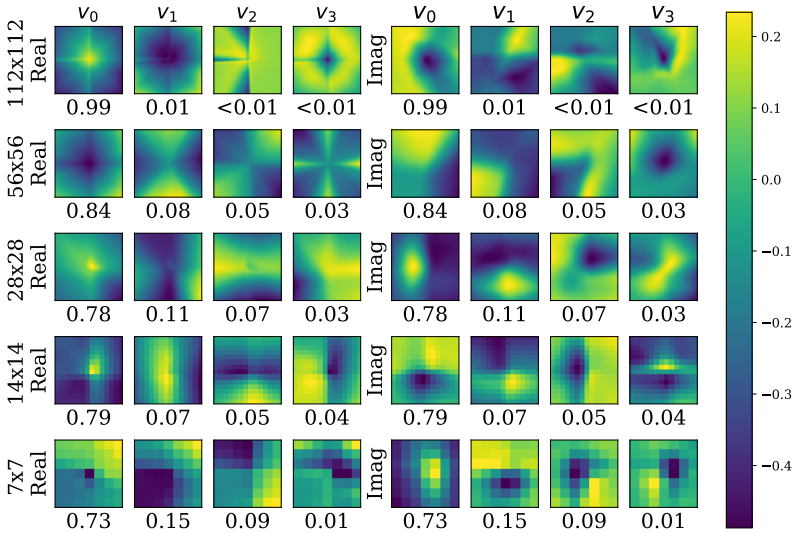


FIGURE C.28: **Frequency Filter Visualization for our NIFF MobileNet-v2 on ImageNet-1k.** PCA basis and explained variance for each basis vector (below) of all element-wise multiplication weights for the real and imaginary part in the frequency domain for each layer of a MobileNet-v2 trained on ImageNet-1k. On the left, the maximal filter size for the corresponding layer is given. Right the weights for the real values are given, and on the left are the imaginary values.