

Interpretable Convolutional Neural Networks for Microscopic Wood Identification

Inauguraldissertation

zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaft
der Universität Mannheim

vorgelegt von

Lars Nieradzik
aus Bremen

Mannheim, 2025

Dekan: Professor Dr. Claus Hertling, Universität Mannheim

Referent: Professor Dr. Janis Keuper, Universität Mannheim

Korreferent: Professor Dr. Claudia Redenbach, Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau

ABSTRACT

Global deforestation threatens biodiversity and accelerates climate change, emphasizing the need for transparent supply chains. The EU regulation on deforestation-free supply chains (EUDR) has increased demand for accurate wood species verification, particularly in paper products. However, verifying wood species in paper is challenging due to pulp mixing and DNA degradation during pulp processing. Microscopic wood anatomy remains the primary and traditional approach for species identification. This work explores the use of Convolutional Neural Networks (CNNs) for automating wood species identification in microscopic images, while advancing our understanding of neural network interpretability. We develop a two-stage system combining vessel detection and classification, alongside novel tools for analyzing how neural networks process visual information. Through collaboration with wood anatomy experts, we evaluate both the system's performance and its decision-making process, providing insights into the relationship between machine learning approaches and domain expertise. The developed methods contribute both to practical wood identification in fiber material and to our broader understanding of neural network behavior.

ZUSAMMENFASSUNG

Die globale Entwaldung bedroht die Artenvielfalt und beschleunigt den Klimawandel, was die Notwendigkeit transparenter Lieferketten betont. Die EU-Verordnung über entwaldungsfreie Lieferketten (EUDR) hat die Nachfrage nach präziser Holzartenbestimmung erhöht, unter anderem bei Papierprodukten. Die Verifizierung von Holzarten in Papier stellt jedoch aufgrund von DNA-Degradierung und Fasermischungen eine besondere Herausforderung dar. Da traditionelle Methoden hier an ihre Grenzen stoßen, bleibt die mikroskopische Holz Anatomie der primäre Ansatz. Diese Arbeit untersucht den Einsatz von Convolutional Neural Networks (CNNs) zur Automatisierung der Holzartenbestimmung in mikroskopischen Aufnahmen und erforscht gleichzeitig die Interpretierbarkeit neuronaler Netzwerke. Wir entwickeln ein zweistufiges System, das Gefäßerkennung und -klassifizierung kombiniert sowie neue Methoden zur Analyse der visuellen Informationsverarbeitung in neuronalen Netzwerken. In Zusammenarbeit mit Expertinnen und Experten der Holz Anatomie evaluieren wir sowohl die Leistung des Systems als auch dessen Entscheidungsprozesse und gewinnen dabei Einblicke in das Zusammenspiel zwischen maschinellen Lernansätzen und Fachwissen. Die entwickelten Methoden leisten damit sowohl einen Beitrag zur praktischen Holzartenbestimmung in Faserstoffen als auch zu unserem grundlegenden Verständnis des Verhaltens neuronaler Netzwerke.

ACKNOWLEDGEMENTS

This thesis marks the culmination of a long and rewarding journey, and I am deeply grateful to all those who accompanied me along the way.

First and foremost, I would like to express my sincere gratitude to my supervisors, Professor Dr. Janis Keuper and Dr. Henrike Stephani, for their unwavering support and guidance. Our weekly meetings over more than three years were invaluable to my development as a researcher. Their thoughtful feedback, critical insights, and constant encouragement shaped the direction of this thesis and helped clarify my ideas. I am truly grateful for their mentorship and dedication.

I would also like to thank Professor Dr. Claudia Redenbach for kindly agreeing to serve as the second supervisor of this thesis.

This work was carried out through a close collaboration between the Fraunhofer Institute for Industrial Mathematics (ITWM) and the Thünen Institute of Wood Research. I am grateful to ITWM for providing the research environment that made this work possible.

My deepest thanks go to Stephanie Helmling, Andrea Olbrich, and Jödis Sieburg-Rockel at the Thünen Institute, whose expertise in wood anatomy and dedication were essential for the success of this research. I would also like to acknowledge Markus Rauhut and Stephanie Wrage, who actively participated in the research discussions. Additionally, I thank Dr. Thomas Stephani for his contributions through experiments involving training networks on hardwood vessels, as well as his active participation in discussions, feedback, and suggestions.

I am grateful to all my colleagues at Fraunhofer ITWM who created a collegial and stimulating work environment. I am especially thankful to those who shared this journey with me through work discussions, coffee breaks, and shared activities outside the office. Among them: Ahmed Alshembari, Lovro Bosnar, Juraj Fulir, Alexander Geng, Damjan Hatic, Alex Keilmann, Runzhou Mao, Aiswarya Nair, and Jelena Zaninovic.

A special thanks to everyone who took the time to read and provide feedback on my papers and drafts of this thesis. Your insights and suggestions significantly improved the final work.

Finally, I am immensely grateful to my family and friends for their continuous support, patience, and understanding throughout this journey. Their encouragement sustained me through the challenges and celebrated with me in the successes.

Thank you all.

CONTENTS

1	Introduction	3
1.1	Background and Motivation	3
1.2	Contributions	4
1.3	Publications	5
1.4	Statement on the Use of Generative AI	5
1.5	Structure of the Thesis	6
2	Theoretical Background	7
2.1	Foundations of Neural Networks	7
2.1.1	Linear Models	7
2.1.2	Feed-Forward Neural Networks	8
2.1.3	Normalization and Regularization	10
2.2	Convolutional Neural Networks	13
2.2.1	From Neural Networks to CNNs	13
2.2.2	Basic Operations and Layers	14
2.2.3	Receptive Fields	15
2.2.4	Architectures	16
2.2.5	Data Augmentation	17
2.3	Properties of Neural Networks	18
2.3.1	Distributed Information Processing	18
2.3.2	Hierarchical Feature Processing	18
2.3.3	Information Compression	19
2.3.4	Adversarial Vulnerability	19
2.4	Interpretability	21
2.4.1	Foundations of Model Interpretability	21
2.4.2	Deep Learning and Interpretability Challenges	22
2.4.3	Post-hoc Interpretation Methods	22
2.4.4	Balancing Accuracy and Interpretability	23
3	Microscopic Wood Identification	25
3.1	Dataset	25
3.1.1	Sample Preparation	25
3.1.2	Microscopy and Imaging	26
3.1.3	Annotation Pipeline	27
3.1.4	Dataset	28
3.1.5	Training and validation split	30
3.2	Vessel detection	31
3.2.1	Image size	31
3.2.2	Choice of object detection algorithm	32
3.2.3	YOLO	32
3.2.4	WoodYOLO	37
3.3	Vessel Classification	45
3.3.1	Image Preprocessing	46
3.3.2	Choice of architectures	47
3.3.3	Metrics	48
3.4	Evaluation	48
3.4.1	Baseline	49
3.4.2	WoodYOLO	53
4	Interpretability	57
4.1	Attribution Methods	57
4.1.1	Fundamentals and Terminology	57

4.1.2	Basic Gradient Methods	58
4.1.3	Path Integration Methods	58
4.1.4	Class Activation Methods	59
4.1.5	Ensemble and Perturbation Methods	60
4.1.6	Limitations	60
4.2	Consistency of Attribution Maps	61
4.3	Evaluating Attribution Methods	62
4.3.1	Qualitative Evaluation	62
4.3.2	Quantitative Evaluation	62
4.4	Improved Masking for Evaluating Attribution Methods	64
4.4.1	Foundations in Human Vision Research	64
4.4.2	Symbol Grounding Problem	65
4.4.3	Properties	69
4.4.4	Masking Operators	70
4.4.5	Adversarial Perturbation Mask	71
4.5	Evaluation of Attribution Map Metrics	72
4.5.1	Statistical Monotonicity	72
4.5.2	Completeness	75
4.5.3	Significance	75
4.5.4	Effect of perturbation strength	76
4.5.5	Attribution maps	77
4.6	Visual Explanations for Wood Species Classification	78
5	Understanding Neural Decisions: A Path-Based Framework	81
5.1	Decision Paths	81
5.1.1	Motivation	81
5.1.2	Mathematical Definition	82
5.1.3	Computation	83
5.1.4	Path Distribution	84
5.1.5	Relationship to Deletion and Insertion Metrics	85
5.2	Top-GAP: Constraining Neural Decisions	86
5.2.1	Motivation	86
5.2.2	Method	87
5.3	Evaluation	89
5.3.1	Analysis of Decision Paths in Neural Networks	89
5.3.2	Attribution maps and Frequency maps	92
5.3.3	Accuracy and Robustness	94
5.3.4	Spatial Focus and Receptive Field	96
5.3.5	Spatial Focus and Frequency maps	98
5.3.6	Ablation study	99
5.4	Visual Explanations for Wood Species Classification	100
6	Discussion and Conclusion	105
6.1	Wood Identification	105
6.1.1	Summary	105
6.1.2	Discussion	106
6.1.3	Limitations and Future Directions	107
6.2	Neural Network Interpretability	108
6.2.1	Summary	108
6.2.2	Discussion	110
6.2.3	Limitations	111
6.2.4	Future Directions	112
6.3	Broader Impact	114
	Bibliography	115
A	Appendix	131

A.1	Evaluation of attribution maps	131
A.2	Top-GAP	132
A.3	Path framework	134

INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

Global deforestation poses significant threats to biodiversity and accelerates climate change. In response, regulatory measures such as the EU regulation on deforestation-free supply chains (EUDR) [Par23] emphasize the need for transparent supply chains, increasing the demand for accurate verification of wood species and origins in traded products. For paper products, this verification is particularly challenging due to the destruction of DNA during production and the mixing of different pulps [TK15; Sch+20]. While methods such as genetic analysis, stable isotope analysis, and near-infrared spectroscopy (NIRS) exist, they are fundamentally limited in these contexts due to DNA degradation and pulp mixing during production. Wood anatomy remains the only established method for species identification [Hel+18; Ilv95].

Wood anatomists build their expertise through examining reference collections of known wood species. To prepare these references, solid wood samples undergo maceration – a chemical process to separate the wood tissue into individual cells [Fra45] – and are stained with specific dyes to enhance visibility under the microscope [RC19]. For analyzing paper products, the process is simpler: the paper sample is dissolved in water to separate the fibers. The cells are stained and then prepared on at least two microscopic slides. Wood anatomists then systematically scan multiple microscope slides to locate vessel elements – specialized water-conducting cells – which provide the most reliable diagnostic features. While other cell types (fibers, tracheids, parenchyma cells) are also present in the paper pulp, vessel elements are prioritized as they retain distinctive morphological features through the paper-making process. The mixing of different wood species in paper pulp means that experts must examine multiple vessels to determine the species composition with confidence. The identification itself requires extensive training, as experts must recognize characteristic features across multiple potential species. This meticulous manual process creates a significant bottleneck in meeting the growing demand for wood identification services, with a single analysis sometimes requiring half an hour of microscope work [Ilv95; Hel+18].

In parallel, advancements in machine learning, particularly convolutional neural networks (CNNs), have revolutionized image analysis tasks across various fields. While AI-based approaches for macroscopic wood identification have achieved notable success [Sil+22; Wie20; Rav+20; He+24], applications for microscopic wood analysis – especially for fibrous materials like paper – remain underdeveloped. The inherent complexity of neural networks, often referred to as their “black-box” nature, presents both challenges and opportunities in developing practical systems that can be trusted and validated by domain experts.

Wood identification provides an interesting case study for neural network interpretability. The field of wood anatomy has well-established standards, such as the 163 structural features defined by the International Association of Wood Anatomists (IAWA) [Com89], demonstrating the complexity and expertise required in this domain. While our work focuses on the computer science aspects rather than detailed anatomical analysis, the presence of domain experts

in our research team allows us to evaluate whether neural networks develop meaningful and reliable features for identification. By examining attribution maps and other interpretability techniques, we can verify that networks focus on anatomically relevant structures rather than artifacts or background patterns, while remaining open to the possibility that they might discover novel patterns or approaches to the identification task that complement traditional expertise.

This thesis seeks to address two interconnected challenges: developing an automated system for microscopic wood identification using deep learning, and advancing our understanding of how neural networks process visual information. Through this dual focus, we aim to contribute both practical tools for wood species verification and insights into neural network interpretation, while maintaining close collaboration with domain experts to ensure the biological relevance of our findings.

1.2 CONTRIBUTIONS

This thesis makes several contributions to the development of deep learning methods for microscopic wood identification and to the broader understanding of neural network interpretability. The main contributions are:

1. **Deep Learning Pipeline for Wood Identification:** We developed a two-stage deep learning system for detecting and classifying vessel elements in microscopy images of paper products. This pipeline significantly accelerates species verification. In collaboration with wood anatomists, we demonstrated that the model's attention aligns with known anatomical structures and occasionally highlights underutilized but biologically meaningful features, such as parenchyma cells.
2. **WoodYOLO:** We introduced WoodYOLO, a domain-specific object detector tailored for identifying vessel elements in fibrous materials. It achieves a 6% improvement in F2 score over YOLOv7, while using lower-resolution inputs and reduced computational resources. This demonstrates the benefits of specialized architectures over general-purpose models in domain-constrained tasks.
3. **Large-Scale Annotated Vessel Dataset:** We created a dataset of 1,614 microscopy images with over 118,000 annotated vessel elements across 13 hardwood genera. Developed in partnership with the Thünen Institute, this dataset addresses a critical resource gap and supports future research in wood anatomy and machine learning.
4. **Evaluation Framework for Attribution Methods:** We proposed a new evaluation framework for attribution methods based on adversarial masking. Using this approach, we identified SmoothGrad as the most reliable method across diverse model architectures and datasets.
5. **Decision Paths Analysis:** We developed a framework to analyze how neural networks combine visual features for classification. Our analysis shows that higher accuracy is associated with more redundant decision paths and smaller critical regions, offering novel insights into how robust predictions emerge from distributed feature representations.
6. **Top-GAP Architecture:** We designed Top-GAP, a novel architecture that constrains spatial processing by limiting receptive field size. This

improves robustness and interpretability, while also revealing a trade-off between focusing model attention and preserving flexibility through alternative feature paths.

Together, these contributions support the development of AI-assisted wood identification for regulatory purposes, while advancing interpretability tools that are applicable to convolutional neural networks in other scientific domains.

1.3 PUBLICATIONS

This thesis is based on the following peer-reviewed publications:

- **Automating Wood Species Detection and Classification in Microscopic Images of Fibrous Materials with Deep Learning**, published in "Microscopy and Microanalysis" [Nie+24a].
This work forms the foundation of the detection system presented in chapter 3.
- **WoodYOLO: A Novel Object Detector for Wood Species Detection in Microscopic Images**, published in "Forests" [Nie+24c].
This paper details the specialized detection architecture discussed in chapter 3.
- **Challenging the Black Box: A Comprehensive Evaluation of Attribution Maps of CNN Applications in Agriculture and Forestry**, published in "Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications" [Nie+24b].
This work provides the comparative analysis of interpretability methods presented in chapter 4.
- **Reliable Evaluation of Attribution Maps in CNNs: A Perturbation-Based Approach**, published in "International Journal of Computer Vision" [NSK24a].
This paper introduces the evaluation metrics discussed in chapter 4.
- **Top-GAP: Integrating Size Priors in CNNs for more Interpretability, Robustness, and Bias Mitigation**, presented at the "eXCV Workshop (ECCV 2024)" [NSK24b].
This work forms the basis of the architectural innovations presented in chapter 5.

Parts of the text and figures in this thesis are adapted from these publications. The author of this thesis is the first author of all these papers, with contributions from co-authors as detailed in the respective publications.

1.4 STATEMENT ON THE USE OF GENERATIVE AI

In the development of this thesis, various artificial intelligence (AI) and automated tools were used as supplementary resources. These included language models (Claude, ChatGPT, DeepSeek) and writing assistance tools (LanguageTool, Grammarly).

These AI tools were employed primarily for code debugging and optimization, brainstorming technical approaches, and enhancing writing clarity. They

were also used as discussion partners to identify potential logical gaps, which were then independently researched and verified through academic literature and peer-reviewed sources. All suggestions generated by these systems were treated as starting points for further development. Importantly, no text, code, or ideas were simply generated and copy-pasted. All content was independently developed, analyzed, and written by the author after careful consideration and research.

1.5 STRUCTURE OF THE THESIS

This thesis presents a comprehensive investigation of CNNs for wood identification, progressing from theoretical foundations to practical implementation and evaluation. The structure is organized as follows:

Chapter 2 provides the theoretical foundations necessary for understanding neural networks and their interpretability. Beginning with artificial neural networks, it introduces fundamental concepts from linear models and progresses to feed-forward neural networks and their convolutional variants, which form the basis of our wood identification system. The chapter then examines core properties of neural networks, including distributed information processing, hierarchical feature processing, information compression and adversarial vulnerability. It concludes with a discussion of interpretability concepts, establishing the theoretical groundwork for understanding model decisions.

Chapter 3 presents our comprehensive approach to microscopic wood identification. It begins by detailing our dataset creation process, including sample preparation, microscopy techniques, and annotation procedures. The chapter then describes the development of our vessel detection system, introducing our novel WoodYOLO architecture and its advantages over traditional approaches. The final sections cover vessel classification, including our pre-processing strategies, architectural choices, and thorough evaluation of both baseline and improved systems.

Chapter 4 examines the interpretability challenges of neural networks in the context of wood anatomy classification. It provides a systematic review of attribution methods and develops an evaluation framework to assess their reliability. The chapter explores how neural networks process anatomical features differently from human experts, revealing both the strengths and limitations of current interpretation approaches.

Chapter 5 develops two complementary tools for understanding neural network decisions: decision paths, which reveal how different combinations of regions maintain classification, and Top-GAP, an architecture for constraining these combinations. These tools provide insights into the distributed nature of neural network processing and suggest that robust visual recognition inherently requires complex feature combinations that resist reduction to simple, interpretable rules.

The thesis concludes in chapter 6 by synthesizing our findings in both wood identification and neural network interpretability, discussing the practical implications and theoretical insights gained from our investigation.

THEORETICAL BACKGROUND

This chapter presents the theoretical foundations necessary for understanding neural networks and their interpretability – two core aspects of this thesis. We first introduce artificial neural networks and their convolutional variants, which form the basis of our wood identification system. We then examine fundamental properties of these networks, which become essential for our later discussions on interpretability. The chapter concludes with an overview of interpretability concepts, providing the theoretical groundwork for our novel approaches to understanding network decisions.

2.1 FOUNDATIONS OF NEURAL NETWORKS

2.1.1 Linear Models

Modern neural networks build upon the principles of linear models [HTF09]. Consider a simple *linear regression* problem with data pairs $\{(x_1, y_1), \dots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^k$ represents input features, and $y_i \in \mathbb{R}$ is the corresponding output.

The standard linear regression model is expressed as:

$$\hat{y} = xw + b,$$

where $w \in \mathbb{R}^k$ is the weight vector, and $b \in \mathbb{R}$ is a scalar bias term. We denote the model parameters collectively as $\theta = \{w, b\}$. These parameters are estimated by minimizing a loss function that measures the discrepancy between predicted and actual outputs.

For regression tasks, the *mean squared error* (MSE) is commonly used as the loss function:

$$L(\hat{y}, y) = (\hat{y} - y)^2.$$

In practice, we optimize over mini-batches of data. Let B denote a set of indices randomly sampled from the full dataset indices $\{1, \dots, n\}$, where each mini-batch contains a small subset of training examples. The average loss over a mini-batch is computed as:

$$L_B(\theta) = \frac{1}{|B|} \sum_{i \in B} L(\hat{y}_i, y_i)$$

The optimization process begins with *stochastic gradient descent* (SGD), which updates the parameters iteratively:

$$\theta_{t+1} := \theta_t - \eta \nabla_{\theta} L_B(\theta_t)$$

where η is the learning rate controlling the step size, t denotes the current iteration, and $\nabla_{\theta} L_B(\theta_t)$ is the gradient of the mini-batch loss with respect to all parameters.

While SGD forms the foundation of neural network optimization, adaptive learning rate methods have become the standard due to their improved convergence properties [SSH21].

The *Adam* (Adaptive Moment Estimation) optimizer [KB17] extends the basic update rule θ_{t+1} by incorporating two key concepts: momentum and adaptive learning rates.

The momentum aspect is captured by the first moment estimate m_t , which computes an exponential moving average of the gradients. The adaptive learning rate component is handled by the second moment estimate v_t , which tracks the exponential moving average of squared gradients:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} L_B(\theta_t) \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} L_B(\theta_t))^2 \end{aligned}$$

Here, β_1 (typically 0.9) controls how much past gradients influence the momentum, while β_2 (typically 0.999) determines how much past squared gradients affect the adaptive learning rates. The momentum term helps overcome local minima and accelerates convergence, while the adaptive learning rates allow different parameters to be updated at different scales based on their gradient history.

Since both moving averages are initialized as zero vectors, they are biased towards zero during the early steps of training. Adam corrects this initialization bias by scaling the moving averages. After applying this bias correction, the final parameter update rule becomes:

$$\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{v_t} + \epsilon} \quad (1)$$

where ϵ (typically 10^{-8}) ensures numerical stability. This update rule automatically adjusts the effective learning rate for each parameter: parameters with larger historical gradients receive smaller updates, while parameters with smaller or infrequent gradients receive larger updates.

For linear models, these adaptive optimization methods are unnecessary since efficient closed-form solutions exist through ordinary least squares. However, Adam's ability to handle non-convex optimization landscapes and automatically adjust learning rates makes it essential for training neural networks, which we will explore in detail in the following sections.

2.1.2 Feed-Forward Neural Networks

Linear models, while powerful in their simplicity, are limited in their ability to capture complex patterns in data. Neural networks overcome this limitation by composing multiple linear transformations with nonlinear functions. Through this composition, neural networks can approximate any continuous function to arbitrary precision [HSW89], a property known as *universal approximation*. The nonlinear transformations between layers are crucial: without them, the entire network would reduce to a single linear transformation, regardless of its depth.

A *feed-forward neural network* (NN) [LBH15] starts with an input vector $x \in \mathbb{R}^Q$, which undergoes an affine transformation followed by a nonlinearity:

$$h_1 = g(xW_1 + b_1),$$

where $W_1 \in \mathbb{R}^{Q \times K_1}$ is the weight matrix, $b_1 \in \mathbb{R}^{K_1}$ is the bias vector, and $g(\cdot)$ is a nonlinear activation function.

While theoretically any nonlinear function could serve as an activation function, certain choices have proven more effective in practice. The Rectified

Linear Unit (ReLU), defined as $g(x) = \max(x, 0)$, has become the most widely used due to its computational efficiency and effectiveness in deep networks [RZL17].

The Gaussian Error Linear Unit (GELU) provides a smooth approximation of ReLU [HG23] and is popular in transformer architectures [Vas+23]. It is defined as $g(x) = x\Phi(x)$, where $\Phi(x)$ is the standard normal cumulative distribution function. Another common choice is the Leaky ReLU [MHN13], defined as $g(x) = \max(x, \alpha x)$ where α is typically 0.01. This modification of ReLU allows small negative gradients, preventing neurons from becoming permanently inactive.

The output of this transformation, h_1 , is referred to as a *hidden layer*. For a network with L layers, subsequent transformations follow a similar pattern:

$$h_l = g(h_{l-1}W_l + b_l), \quad l = 2, \dots, L, \quad (2)$$

where $W_l \in \mathbb{R}^{K_{l-1} \times K_l}$ is the weight matrix and $b_l \in \mathbb{R}^{K_l}$ is the bias vector for layer l . The dimensions K_i of each hidden layer are hyperparameters that control the network's capacity and computational requirements.

The complete network can thus be viewed as a composition of functions $f(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$, where each f_i represents the combination of an affine transformation and nonlinear activation at layer i . This composition structure becomes particularly important when computing gradients during training through the chain rule.

The network concludes with an output layer that maps the final hidden representation to the desired output space:

$$\hat{y} = h_L W_{L+1} + b_{L+1},$$

where $W_{L+1} \in \mathbb{R}^{K_L \times D}$ and $b_{L+1} \in \mathbb{R}^D$ transform the last hidden layer into the prediction $\hat{y} \in \mathbb{R}^D$. The output layer typically omits the nonlinear activation function for regression tasks, allowing the network to produce unrestricted values.

For classification tasks, we apply additional transformations to obtain properly normalized probability distributions. In binary classification ($D = 1$), the *sigmoid* function maps the scalar output to a probability in the range $(0, 1)$:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (3)$$

Here, \hat{y} represents the probability of the positive class, while $1 - \hat{y}$ gives the probability of the negative class.

For multi-class problems ($D = K > 2$), the *softmax* function generalizes this concept by mapping the logit vector to a probability distribution over all K classes:

$$\hat{y}_i = \text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}, \quad \text{for } i = 1, \dots, K$$

The resulting vector $\hat{y} \in [0, 1]^K$ satisfies $\sum_{i=1}^K \hat{y}_i = 1$, ensuring a valid probability distribution. In practice, to prevent numerical overflow, the softmax computation is typically normalized by subtracting the maximum logit, a technique known as the log-sum-exp trick [BHH20]:

$$\text{softmax}(z)_i = \frac{e^{z_i - \max_j z_j}}{\sum_{j=1}^K e^{z_j - \max_j z_j}}$$

This normalization is particularly important when dealing with large logits, as direct computation of e^{z_i} could exceed floating-point limits.

These output transformations are paired with appropriate loss functions for training. For binary classification, the binary cross-entropy loss is defined as:

$$L(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad (4)$$

where $y \in \{0, 1\}$ is the true label and \hat{y} is the predicted probability.

For multi-class problems, the categorical cross-entropy loss extends this concept:

$$L(\hat{y}, y) = - \sum_{i=1}^K y_i \log(\hat{y}_i)$$

where y is a one-hot encoded vector representing the true class.

While cross-entropy serves as a fundamental loss function for classification tasks, deep learning applications often require specialized loss functions tailored to their specific objectives. For instance, in object detection tasks, the loss must account for both the localization accuracy of predicted bounding boxes and the confidence of object presence. This leads to complex loss formulations that typically combine multiple components. A key example is the use of Intersection over Union (IoU) based losses, which measure the spatial overlap between predicted and ground truth bounding boxes. We will explore these object detection losses in detail in chapter 3.

2.1.3 Normalization and Regularization

While feed-forward networks provide a powerful framework for learning, their training process faces fundamental challenges related to gradient propagation. Consider a deep network with L layers computing a composition of functions as introduced earlier: $f(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$ (see equation (2)). During backpropagation, we compute gradients through repeated application of the chain rule:

$$\frac{\partial L}{\partial f_1} = \frac{\partial L}{\partial f_L} \cdot \frac{\partial f_L}{\partial f_{L-1}} \cdot \dots \cdot \frac{\partial f_2}{\partial f_1} \quad (5)$$

If each partial derivative term is less than 1, their product decreases exponentially with depth, causing gradients to vanish [BSF94; Hoc91; PMB13]. Conversely, if terms are greater than 1, the product grows exponentially, leading to exploding gradients. For example, with sigmoid activations $\sigma(x)$, the derivative $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ is bounded by 0.25, meaning that gradients through a 10-layer network are scaled by at least $0.25^{10} \approx 10^{-6}$.

Early attempts to address these issues focused on careful weight initialization and activation function selection [LJH15; GB10; He+15b]. However, a more effective solution emerged through **normalization techniques**, which actively standardize intermediate representations during training. These techniques not only help maintain activations in reasonable ranges but, more importantly, fundamentally alter the optimization landscape by controlling the magnitude of these partial derivatives.

Batch Normalization (BatchNorm) [IS15] has become a cornerstone of modern neural networks. Initially, BatchNorm was thought to improve training by reducing internal covariate shift, which refers to the change in the distribution

of network activations due to the updating of preceding layers. However, recent work [San+19; Bjo+18] has revealed that BatchNorm’s primary benefit comes from smoothing the optimization landscape, making gradients more predictive and stable during training.

For a layer with activations $x \in \mathbb{R}^{B \times D}$, where B is the batch size and D is the feature dimension, BatchNorm performs the following steps:

$$\mu_d = \frac{1}{B} \sum_{b=1}^B x_{bd} \quad (\text{compute mean for each feature})$$

$$\sigma_d^2 = \frac{1}{B} \sum_{b=1}^B (x_{bd} - \mu_d)^2 \quad (\text{compute variance})$$

$$\hat{x}_{bd} = \gamma_d \frac{x_{bd} - \mu_d}{\sqrt{\sigma_d^2 + \epsilon}} + \beta_d \quad (\text{normalize and rescale})$$

where γ_d, β_d are learnable scale and shift parameters that allow the network to recover the original representation if needed, and ϵ is a small constant (typically 10^{-5}) added for numerical stability when dividing.

During inference, we cannot use batch statistics because this would make predictions inconsistent. Since neural networks often need to process individual inputs in practice, we instead use running averages of the means μ_{running} and variances $\sigma_{\text{running}}^2$ that were computed and updated throughout training:

$$\hat{x}_{bd} = \gamma_d \frac{x_{bd} - \mu_{\text{running}}}{\sqrt{\sigma_{\text{running}}^2 + \epsilon}} + \beta_d$$

BatchNorm’s effectiveness depends critically on batch size, as small batches can lead to noisy statistics and poor performance [WH18]. Therefore, it is recommended to use sufficiently large batch sizes when memory permits.

Layer Normalization (LayerNorm) [BKH16] addresses this limitation by computing statistics across the feature dimension instead of the batch dimension:

$$\mu_b = \frac{1}{D} \sum_{d=1}^D x_{bd}$$

$$\sigma_b^2 = \frac{1}{D} \sum_{d=1}^D (x_{bd} - \mu_b)^2$$

$$\hat{x}_{bd} = \gamma \frac{x_{bd} - \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} + \beta$$

where, similar to BatchNorm, γ and β are learnable scale and shift parameters that allow the network to recover the original representation if needed.

An important advantage of LayerNorm is its behavior during inference. Since statistics are computed independently for each input sample across its features, the normalization process remains identical between training and inference, eliminating the need for maintaining running averages. This property, combined with its more memory-efficient training due to not storing running statistics, has made LayerNorm particularly attractive for modern architectures.

Initially developed for recurrent neural networks, LayerNorm has been widely used in modern architectures. Transformer models [Vas+23] use LayerNorm, and recent convolutional architectures like ConvNeXt [Liu+22] have

also adopted it, showing competitive or superior performance compared to BatchNorm-based models.

While normalization techniques help stabilize training, deep neural networks also require explicit **regularization** to prevent overfitting. Two fundamental approaches have emerged as particularly effective: *dropout* and *weight decay*.

Dropout [Sri+14] is a regularization technique that prevents overfitting by randomly removing activations during training. Without dropout, the network might learn to rely too heavily on specific combinations of activations in its hidden layers, leading to patterns that work well for training data but do not generalize well to new examples. Dropout addresses this by randomly “dropping out” (setting to zero) a fraction of the hidden layer activations during each training step:

$$\hat{h} = \begin{cases} \frac{h}{1-p} & \text{with probability } 1-p \\ 0 & \text{with probability } p \end{cases} \quad (6)$$

where h represents any element of a hidden layer activation h_i (as defined in the feed-forward network section), and p is typically set between 0.1 and 0.5. When an activation is kept (with probability $1-p$), its value is scaled up by $\frac{1}{1-p}$ to maintain the expected magnitude of inputs to the subsequent layer. This scaling ensures that the expected sum of the inputs to the next layer remains approximately the same whether we are training (with dropout) or performing inference (without dropout).

An important insight is that dropout can be interpreted as training an ensemble of networks sharing parameters [GG16]. Each application of dropout effectively samples a different subnetwork from an exponential number of possible network configurations, as different combinations of activations are randomly zeroed out during training.

Weight decay, also known as L2 regularization in the statistical learning literature, adds a penalty term to the loss function that encourages smaller weights:

$$L_{\text{total}} = L_{\text{task}} + \frac{\lambda}{2} \sum_i w_i^2 \quad (7)$$

where λ controls the strength of regularization and w_i are the model parameters. The factor of $\frac{1}{2}$ is included for mathematical convenience: when we take the derivative of the regularization term with respect to w_i , the 2 from the power cancels with the $\frac{1}{2}$, giving us simply λw_i in the gradient. This makes the weight decay interpretation more direct, as each weight is directly scaled by λ in the update step.

While L1 regularization ($\sum_i |w_i|$) is another possible choice that promotes sparsity, L2 regularization is more commonly used in deep learning as it provides smoother gradients and better optimization properties.

In the context of neural networks, weight decay is typically implemented as part of the optimizer rather than the loss function.

This is particularly relevant when using adaptive optimizers like Adam (discussed earlier in equation (1)). The original Adam implementation applies the weight decay term after computing the adaptive learning rates, which can

lead to suboptimal regularization. To address this, AdamW [LH19] decouples weight decay from the gradient updates:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} L_{\text{task}}(\theta_t) \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} L_{\text{task}}(\theta_t))^2 \\ \hat{\theta}_{t+1} &= \theta_t - \eta \left(\frac{m_t}{\sqrt{v_t} + \epsilon} + \lambda \theta_t \right) \end{aligned}$$

where the weight decay term $\lambda \theta_t$ is applied independently of the adaptive learning rate computation.

2.2 CONVOLUTIONAL NEURAL NETWORKS

While feed-forward networks with their layer-wise transformations provide a foundation for many machine learning tasks, the structure of image data presents unique challenges. Consider processing a 224×224 RGB image with a standard feed-forward network: each element in the first layer would need to connect to every pixel in the input image, requiring over 150,000 weights per output dimension ($224 \times 224 \times 3$ input values). This dense connectivity makes traditional feed-forward architectures computationally intractable for image processing tasks. *Convolutional neural networks (CNNs)* [LeC+89; KSH12] address this challenge by replacing these dense connections with localized operations that exploit the spatial structure of the data.

2.2.1 From Neural Networks to CNNs

The development of CNNs was influenced by studies of the visual cortex [HW62], which revealed that visual processing begins with neurons that respond to small, specific regions of the visual field. Early computational models like the Neocognitron [Fuk80] implemented this principle using cascaded layers of local feature detectors. Each layer applied fixed Gaussian filters to detect patterns at different scales, creating a hierarchy of increasingly complex feature detectors. These models demonstrated the effectiveness of local processing but were limited by their hand-designed filters.

Modern CNNs build on this foundation by making the local filters learnable parameters. Consider an input image tensor $x \in \mathbb{R}^{H \times W \times C}$, where H and W represent spatial dimensions (height and width) and C represents channels (e.g., $C = 3$ for RGB images). Instead of the matrix multiplication xW used in feed-forward networks, CNNs define a small window of weights that slides across the spatial dimensions. This window computes the same local transformation at each position, replacing the dense connections of standard feed-forward layers with a sparse, repeating pattern.

For example, processing a 224×224 RGB image with a 3×3 sliding window requires only 27 weights ($3 \times 3 \times 3$) that are reused across all spatial positions, compared to the millions of weights needed in a fully-connected layer. Through training, these local transformations learn to detect relevant patterns across the entire image.

Mathematically, CNNs maintain the same structure as feed-forward networks (refer to equation (2)):

$$f(x) = f_L \circ f_{L-1} \circ \dots \circ f_1(x)$$

However, each function f_i now represents a transformation using these sliding windows instead of dense connections. The specific implementation of these operations through convolutions will be detailed in the following sections.

2.2.2 Basic Operations and Layers

The fundamental building block of CNNs is the *convolutional layer*, which implements the local transformations introduced in the previous section. For an input tensor $x \in \mathbb{R}^{H \times W \times C}$, the layer applies a set of learnable kernels $W_m \in \mathbb{R}^{k \times k \times C}$, where $m = 1, \dots, M$ indexes the output channels and k is the kernel size (typically 3 or 7). Using square bracket notation for indexing, the layer computes for each spatial position $[i, j]$ and output channel m :

$$h_m[i, j] = g \left(\sum_{c=1}^C \sum_{p,q=1}^k x[i+p, j+q, c] W_m[p, q, c] + b_m \right) \quad (8)$$

where $b_m \in \mathbb{R}$ is a learnable bias term and $g(\cdot)$ is a nonlinear activation function [How+17].

The spatial dimensions of the output feature maps depend on three hyperparameters: the number of kernels M , the stride s , and the padding scheme. The stride controls how the kernel moves across the input: a stride of s means the kernel center moves s positions at a time, producing output feature maps with dimensions reduced by a factor of s . To maintain spatial dimensions when $s = 1$, we typically pad the input tensor with zeros around its borders before applying the convolution. This "same" padding ensures that each pixel in the input contributes equally to the output, preventing information loss at the boundaries.

CNNs commonly employ pooling operations to explicitly reduce spatial dimensions and introduce invariance to small translations [SMB10]. For an input feature map $h \in \mathbb{R}^{H \times W \times C}$, *max pooling* outputs the maximum value within each local neighborhood:

$$\text{MaxPool}(h)[i, j, c] = \max_{(p,q) \in N(i,j)} h[p, q, c]$$

where $N(i, j)$ defines a local $p \times p$ neighborhood around position (i, j) . Typically, $p = 2$ is used, halving the spatial dimensions. *Average pooling* provides an alternative by computing the mean over each neighborhood:

$$\text{AvgPool}(h)[i, j, c] = \frac{1}{|N(i, j)|} \sum_{(p,q) \in N(i,j)} h[p, q, c]$$

A special case, *Global Average Pooling* (GAP) [LCY14], averages over the entire spatial domain:

$$\text{GAP}(h)_c = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W h[i, j, c]$$

This operation collapses the spatial dimensions into a single value per channel, producing a vector $\in \mathbb{R}^C$. Modern architectures often use GAP as the final dimensionality reduction layer before classification, replacing the traditional flatten and dense layers while maintaining spatial equivariance

throughout the network. For instance, in a CNN processing a 224×224 RGB image, the final convolutional layer might output a $7 \times 7 \times 512$ feature map. GAP reduces this to a 512-dimensional feature vector that can be directly connected to classification layers.

2.2.3 Receptive Fields

The hierarchical nature of convolutional neural networks raises a fundamental question in architecture design: how does spatial information propagate through the network? This question is particularly important for tasks like object detection, where the network must capture features at scales large enough to represent entire objects, such as cars. The concept of *receptive fields* provides a theoretical framework for analyzing this information flow by precisely defining how each computation in the network relates to the original input space.

For a single convolutional layer with kernel size k , each output value integrates information from exactly $k \times k$ input positions. As we compose multiple layers, these regions of influence expand systematically. Consider a network with L layers, where layer i has kernel size k_i and stride s_i . The receptive field size r_i at layer i follows a recursive pattern [Luo+16]:

$$r_i = r_{i-1} + (k_i - 1) \prod_{j=1}^{i-1} s_j,$$

initialized with $r_1 = k_1$. This formulation reveals how architectural choices affect spatial information propagation. Networks using convolutions with stride 1 ($s_i = 1$) exhibit linear growth in receptive field size, with each layer expanding the field by $k_i - 1$ pixels. In contrast, incorporating strides greater than 1 leads to multiplicative growth, enabling deeper layers to efficiently capture long-range spatial dependencies.

Consider a network with two layers, where each layer uses a 3×3 kernel ($k_i = 3$) and stride 2 ($s_i = 2$). The receptive field size at the second layer is calculated as:

$$r_2 = r_1 + (k_2 - 1)s_1 = 3 + (3 - 1)(2) = 7 \text{ pixels.}$$

This means neurons in the second layer can theoretically capture context spanning a 7×7 region of the input image. However, this would be insufficient for detecting cars or wood vessels, motivating the need for more layers.

Theoretical analysis of receptive fields differs from their empirical behavior. Research has revealed that neurons primarily respond to a subset of their theoretical input region, a phenomenon known as the *effective receptive field* [Luo+16]. The influence of input pixels follows a Gaussian-like distribution, with central pixels contributing more strongly to the output than peripheral ones.

The disparity between theoretical and effective receptive fields has important implications for architecture design. While deeper networks theoretically enable broader spatial context integration, the effective receptive field often grows more slowly than calculations suggest. This insight has motivated several architectural innovations.

2.2.4 Architectures

Early CNN architectures like VGG [SZ15] demonstrated the effectiveness of simple, repeated blocks of convolution and max pooling operations. These networks used a straightforward design principle: stacking multiple 3×3 convolution layers interspersed with max pooling operations. This approach aligns logically with the observation that too few 3×3 kernels alone result in limited receptive field sizes. By increasing depth while keeping other architectural choices fixed, VGG substantially improved performance and became a foundational design for subsequent architectures.

However, increasing the depth of neural networks introduces significant training challenges, as previously discussed (section 2.1.3). In deep networks, gradients diminish as they are propagated through many layers, which impedes the effective updating of weights in earlier layers (see especially equation (5)).

ResNet [He+15a] addressed this issue by introducing residual connections. These connections allow the input of a layer to bypass the convolutional transformations and be added directly to the layer’s output:

$$h_l = g(F_l(h_{l-1})) + h_{l-1},$$

where F_l represents the convolutional transformation (see equation (8)) at layer l , and g is the activation function.

The addition of these shortcut connections enables the training of networks with many more layers. ResNet can scale to hundreds of layers without being affected by the vanishing gradient problem.

Furthermore, Veit et al. [VWB16] showed that networks with residual connections can be interpreted as ensembles of multiple potential paths of varying depths. Each path represents a unique subset of layers, with some paths bypassing intermediate transformations entirely. This interpretation helps explain the robust performance of ResNet and its ability to generalize. The residual connections, in essence, create a more flexible learning structure, which allows the network to adjust to different depths dynamically during training.

DenseNet [HLW16] further extended the concept of connectivity. Unlike ResNet, which only adds the input of a layer to its output through residual connections, DenseNet concatenates the outputs from all preceding layers:

$$h_l = g(F_l([h_1, h_2, \dots, h_{l-1}])),$$

where $[\cdot]$ denotes channel-wise concatenation. This dense connectivity encourages feature reuse and results in more compact models. However, the concatenation operation does come with some computational overhead, as it increases the dimensionality of the feature maps at each layer. Despite this, we also obtain reduced parameter counts.

Channel attention mechanisms were introduced through *squeeze-and-excitation* (SE) blocks [Hu+19], which enable networks to model interdependencies between channels. EfficientNet [TL19] leveraged these SE blocks alongside a systematic approach to model scaling, introducing compound scaling coefficients that jointly optimize network depth, width, and input resolution. Their approach demonstrated that carefully balancing these dimensions is crucial for efficient network design.

More recent architectures like *ConvNeXt* [Liu+22] have modernized CNN design by incorporating insights from transformer architectures [Dos+21].

They employ larger kernel sizes (7×7) in earlier layers, replace ReLU with GELU activation functions [HG23], and modify the position of layer normalization. A significant innovation was their adoption of LayerScale [Tou+21], which introduces learnable scaling factors initialized near zero for the residual path in deeper layers. This initialization effectively makes the network behave like a shallow architecture initially, with deeper layers gradually becoming active during training as the scaling factors increase.

2.2.5 Data Augmentation

Training deep neural networks requires substantial amounts of labeled data to achieve good generalization. Data augmentation addresses this challenge by systematically expanding the training dataset through controlled transformations, particularly important when working with specialized image domains like microscopy where labeled data may be limited.

The core idea behind data augmentation is that the network should recognize an object regardless of certain transformations applied to its image. For example, if we have an image of a vessel element, the network should still identify it correctly even if the image is slightly rotated or its brightness is adjusted. Mathematically, given an input image x and a transformation T , we want the network f to produce similar predictions for both the original and transformed images: $f(T(x)) \approx f(x)$. This property helps the network learn features that are robust to common variations in the input.

Consider our training data consisting of image-label pairs $\{(x_i, y_i)\}_{i=1}^n$. Data augmentation expands this dataset by applying transformations T to create additional valid training examples: $\{(T(x_i), y_i)\}_{i=1}^n$. These transformations typically fall into two main categories: *geometric* and *photometric*.

Geometric transformations modify the spatial arrangement of pixels while preserving semantic content. Common operations include horizontal and vertical flips, random scaling, rotations, and translations [SK19]. The choice and magnitude of these transformations should reflect the expected variation in the target domain. For instance, in microscopy images of wood vessels, small rotations are valid augmentations as vessel orientation can vary naturally, but extreme distortions might create unrealistic patterns.

Photometric transformations modify pixel intensities while maintaining the image's semantic content. These include adjustments to brightness, contrast, and the addition of random noise. In specialized domains like microscopy, these transformations must be carefully calibrated to preserve important visual features.

More sophisticated augmentation strategies have emerged in recent years. *Mosaic augmentation* [BWL20] combines multiple training images into a single training instance by dividing the input space into four regions and filling each with a different scaled image. Traditional augmentations like rotation maintain all spatial relationships within an image. In contrast, mosaic augmentation places objects from different images next to each other at the boundaries where the four images meet. For example, a car from one image might appear next to a tree from another image. This combination didn't exist in either original image. This approach is particularly effective for object detection tasks as it creates training instances with object combinations not found in natural images, while ensuring each individual object remains realistic and recognizable.

TrivialAugment [MH21] provides a simplified approach to augmentation by randomly sampling a single transformation and applying it with random

magnitude. The method relies on two random choices: selecting which transformation to apply from a predefined set, and determining how strongly to apply it. This straightforward approach eliminates the need for parameter tuning while maintaining competitive performance on natural image datasets. However, its application to specialized domains like microscopy requires careful consideration, as the range of valid transformations and their magnitudes may differ significantly from natural images. For instance, certain color transformations or extreme geometric distortions that work well for natural scenes might alter biologically significant features in microscopy images.

2.3 PROPERTIES OF NEURAL NETWORKS

Neural networks, particularly Convolutional Neural Networks (CNNs), exhibit several fundamental properties that arise from their architecture and training process. This section examines these properties and their supporting empirical evidence, drawing parallels with human visual processing where relevant.

2.3.1 *Distributed Information Processing*

The most fundamental property of neural networks is their *distributed processing* of information across multiple units. This property has direct parallels in biological vision, where information is similarly distributed across populations of neurons rather than encoded in individual cells [DZR12]. In both artificial and biological systems, this distributed representation enables robust processing of visual information.

Research has shown that these activation patterns often encode semantic relationships, where similar inputs tend to produce similar patterns of activation [HMR86; RM86]. This organization of semantic relationships in the activation space supports the network's ability to generalize across related inputs and maintain stable outputs despite variations in the input.

In CNNs, distributed processing manifests in the network's response to partial or corrupted inputs. Studies have shown that CNNs can maintain reasonable classification accuracy even when significant portions of images are occluded or distorted [Zho+15b]. This robustness mirrors human visual perception, where we can recognize objects under varying conditions of occlusion, illumination, and viewpoint.

The distributed nature of processing extends to the network's internal architecture. Recent research on the lottery ticket hypothesis [FC19] has revealed that networks contain multiple sparse subnetworks capable of solving the same task. This architectural redundancy suggests that distributed processing occurs not just at the level of individual layers but across the entire network structure. Training techniques like dropout [Sri+14] (see equation (6)) explicitly leverage these forms of redundancy to improve generalization, demonstrating how architectural properties can be exploited to enhance network performance.

2.3.2 *Hierarchical Feature Processing*

A second key property is the network's hierarchical organization of visual processing, which shows remarkable similarities to the primate visual system. Just as the ventral visual stream processes information through a hierarchy of cor-

tical areas [DZR12], CNNs transform visual information through successive layers of increasing abstraction.

Zeiler and Fergus [ZF13] provided compelling evidence for this hierarchical processing by visualizing activations at different network depths. Their study revealed that early layers respond to simple edges and textures, while deeper layers detect progressively more complex patterns. This organization parallels the hierarchical processing observed in primate visual cortex, where early areas respond to simple features while higher areas encode more complex object properties.

The hierarchical property enables efficient transfer learning in both artificial and biological systems. In CNNs, features learned in early layers transfer well across tasks [Yos+14], similar to how early visual processing in biological systems provides general feature detection that supports multiple higher-level visual functions.

2.3.3 Information Compression

The third fundamental property involves how networks transform and compress information through their layers. The Information Bottleneck theory [TZ15] provides a theoretical framework for understanding this behavior. According to this perspective, each layer transforms its input to preserve task-relevant information while discarding irrelevant variations.

This compression property manifests in the network's ability to achieve remarkable parameter efficiency. Multiple studies have demonstrated that networks can capture complex patterns with relatively few parameters compared to the theoretical capacity needed to memorize their training data [Zha+17]. This efficiency suggests that networks discover useful abstractions.

Empirical studies have supported this view by showing that representations become increasingly specialized for the target task in deeper layers [AB18]. This compression property helps explain why networks can achieve good generalization despite having enough parameters to memorize their training data. Rather than storing exact input-output mappings, networks appear to learn compressed representations that capture essential patterns.

These three properties – distributed processing, hierarchical organization, and information compression – are deeply interconnected and mutually reinforcing. The distributed nature of processing supports robust hierarchical feature extraction, while information compression enables efficient learning and generalization. The parameter efficiency observed in neural networks emerges from the successful interaction of these properties, allowing networks to learn compact yet powerful representations of complex data distributions.

2.3.4 Adversarial Vulnerability

The final fundamental property of neural networks is their vulnerability to *adversarial attacks*. Small, carefully crafted perturbations to inputs can cause dramatic changes in network outputs while remaining imperceptible to human observers. These adversarial examples, first identified by [Sze+14], reveal critical insights into how neural networks process information and highlight important security considerations for deployment in real-world applications.

Consider a trained neural network f and an input x correctly classified as class y . An adversarial example can be constructed by solving the optimization problem:

$$\begin{aligned}
& \text{minimize}_{\delta} \quad \|\delta\|_p \\
& \text{subject to} \quad f(x + \delta) \neq y \\
& \quad \quad \quad x + \delta \in [0, 1]^n \\
& \quad \quad \quad \|\delta\|_p \leq \varepsilon
\end{aligned}$$

where δ represents the adversarial perturbation, ε bounds its magnitude, and $\|\cdot\|_p$ typically denotes either the L_∞ or L_2 norm. The choice of norm significantly impacts the nature of the resulting perturbations. The L_∞ norm, defined as $\|x\|_\infty = \max_i |x_i|$, measures the maximum change to any input dimension. In image domains, this effectively limits the maximum change to any pixel. When $\varepsilon = 8/255$, for instance, each pixel can change by at most 8 pixels. The L_2 norm, defined as $\|x\|_2 = \sqrt{\sum_i x_i^2}$, measures the Euclidean distance between the original and perturbed inputs. This allows larger changes to some pixels while maintaining a bound on the total perturbation magnitude.

The Fast Gradient Sign Method (FGSM) [GSS15] provides an efficient approach for generating adversarial examples:

$$x' = x + \varepsilon \cdot \text{sgn} \left(\frac{\partial L(f(x), y)}{\partial x} \right),$$

where L represents the loss function and $\text{sgn}(\cdot)$ is the sign function. While computationally efficient, FGSM generates relatively weak attacks due to its single-step nature.

Projected Gradient Descent (PGD) [Mad+19] strengthens this approach by iteratively applying gradient steps followed by projection onto the allowed perturbation set. Starting from the input x , for iteration steps $t = 0, 1, \dots, T$, PGD computes:

$$x_{t+1} = \text{Proj}_{\|\cdot\|_p \leq \varepsilon} \left(x_t + \alpha \cdot \text{sgn} \left(\frac{\partial L(f(x_t), y)}{\partial x} \right) \right),$$

where α determines the step size and Proj denotes projection onto the ε -ball in the chosen norm. PGD typically finds more effective adversarial examples than FGSM. Recent work has introduced more sophisticated evaluation frameworks like AutoAttack [CH20], which combines multiple attack strategies including adaptive PGD variants and black-box attacks.

A crucial practical consideration in adversarial attacks is the effect of image quantization and compression. When adversarial examples are saved as JPEG images, the perturbations may be partially removed due to quantization and compression artifacts [DGR16]. This can significantly impact attack success rates in real-world scenarios.

Several theoretical frameworks have emerged to explain adversarial vulnerability. One perspective argues that these examples exploit statistical irregularities in the training distribution [Ily+19]. Under this view, networks learn both robust features that align with human perception and non-robust features that correlate with class labels but are easily manipulated. This interpretation connects adversarial vulnerability to the information compression property discussed earlier: networks may preserve these non-robust features during training because they provide useful signal for the classification task, even though they lead to brittleness under adversarial attack.

Adversarial training provides a direct approach to improving model robustness. During each training step, the model is trained on both clean examples and their adversarial counterparts generated using methods like PGD. While effective, this approach substantially increases computational cost and can reduce accuracy on clean inputs, illustrating the inherent trade-offs in building robust systems.

Understanding these vulnerabilities provides crucial insights into how neural networks process information and make decisions. The ability to generate minimal perturbations that dramatically affect network outputs reveals fundamental aspects of network behavior, from decision boundaries to feature sensitivity. These insights become particularly valuable when examining interpretability and architectural constraints in subsequent chapters. For instance, analyzing how networks respond to small input changes helps identify which features truly drive predictions and where architectural modifications might improve robustness. We will revisit these concepts of minimal perturbations in section 4.4.5 when developing improved evaluation metrics for attribution methods.

2.4 INTERPRETABILITY

The increasing complexity of deep learning models has made interpretability a crucial aspect of modern machine learning. While these models achieve remarkable performance across various tasks, their decision-making processes often remain opaque. This section examines the theoretical foundations of interpretability and its relationship with model complexity.

2.4.1 Foundations of Model Interpretability

Model interpretability can be characterized through two fundamental properties [Lip17]: transparency and post-hoc interpretability. Transparency refers to the inherent understandability of a model's components and operations, while post-hoc interpretability describes how well model decisions can be explained after they are made.

A model's transparency can be assessed at multiple interconnected levels: algorithmic transparency, decomposability, and simulatability. Algorithmic transparency reflects our theoretical understanding of the learning process itself. For instance, in convex optimization problems, we can prove convergence properties and uniqueness of solutions. Decomposability measures how well we can break down a model into interpretable components. Simulatability captures whether a human can mentally simulate the model's operation, typically only possible for very simple models.

Consider a linear model with input $x \in \mathbb{R}^k$ and parameters $\theta = \{w \in \mathbb{R}^k, b \in \mathbb{R}\}$:

$$\hat{y} = xw + b$$

This model exhibits high transparency at all levels. At the component level, each weight w_i represents the change in output for a unit change in the corresponding input feature:

$$\frac{\partial \hat{y}}{\partial x_i} = w_i$$

The linear relationship between inputs and outputs makes the model simulatable, allowing humans to mentally verify its predictions.

2.4.2 *Deep Learning and Interpretability Challenges*

The relationship between model complexity and interpretability presents subtle trade-offs that challenge common assumptions. While simpler models like linear classifiers are often considered more interpretable than deep neural networks, this view fails to account for the complexity of feature engineering and the nature of visual recognition tasks [Rud19].

For image classification tasks, linear models require extensive feature engineering to capture useful patterns. Consider the classical Viola-Jones face detector [VJ01], which uses thousands of hand-engineered Haar-like features. While individual engineered features may have clear mathematical definitions, the transformation from raw pixels to these features introduces its own form of complexity. The feature engineering process often involves complex, hand-designed transformations that make it difficult to relate the model's decisions back to the input space, thus compromising both decomposability and the ability to provide meaningful post-hoc interpretability.

Deep neural networks present distinct challenges when evaluated against Lipton's interpretability framework [Lip17]. Algorithmic transparency presents a mixed picture for deep networks. While we understand the individual mathematical operations (matrix multiplications, nonlinear activations), the non-convex nature of the optimization landscape severely limits our theoretical guarantees. Unlike convex problems where we can prove global convergence, deep network optimization lacks general convergence guarantees to global optima. Theoretical results exist only for highly restrictive cases: in the infinite-width limit, networks converge to an explicit limiting kernel and exhibit predictable training dynamics [JGH20].

Decomposability becomes problematic as we cannot easily assign interpretable meanings to individual neurons or layers, particularly in intermediate representations.

Most critically, simulatability breaks down entirely. While individual operations remain simple, their composition creates decision boundaries of such complexity that humans cannot mentally simulate the model's behavior. This loss of simulatability requires post-hoc interpretability methods to understand model decisions.

2.4.3 *Post-hoc Interpretation Methods*

Post-hoc interpretation methods provide tools for understanding model decisions after they are made. These methods are particularly crucial for deep learning models, where direct transparency is limited by model complexity. Among these methods, feature attribution techniques form a major category of post-hoc interpretation. They help identify which parts of an input most influenced the model's decision.

Various approaches have been developed to compute these attributions, ranging from gradient-based methods to class activation maps. In chapter 4, we will examine these methods in detail, exploring their mathematical foundations, implementation challenges, and practical limitations, particularly regarding their ability to capture feature interactions and the challenge of evaluating attribution quality.

2.4.4 *Balancing Accuracy and Interpretability*

The relationship between model accuracy and interpretability has long been debated. The conventional perspective suggests an inherent trade-off: more accurate models tend to be less interpretable, forcing practitioners to choose between performance and interpretability [Rud19]. However, this perspective oversimplifies a complex relationship that depends heavily on the specific task, data characteristics, and how interpretability is defined and measured.

Recent theoretical work has begun to challenge the universality of the accuracy-interpretability trade-off [DBR20]. In certain domains, interpretable models can achieve competitive or superior performance compared to black-box alternatives.

One promising avenue for potentially reconciling accuracy and interpretability lies in regularization techniques, which can influence model structure in ways that may enhance both generalization and interpretability. Consider how different regularization approaches modify the standard loss function. Sparsity regularization through L_1 penalties modifies the objective as:

$$L_{\text{total}} = L_{\text{task}} + \lambda \sum_{l,i,j} |W_l[i,j]|$$

where the sum is taken over all weight matrices W_l and their elements, encouraging many weights to approach zero and potentially creating networks with fewer active connections that may be more transparent in their decision processes.

Weight decay through L_2 regularization, as discussed earlier (see equation (7)), takes the form:

$$L_{\text{total}} = L_{\text{task}} + \lambda \sum_{l,i,j} W_l[i,j]^2$$

This promotes smoother decision boundaries by preventing weights from growing too large, which may lead to more stable and consistent explanations.

Activation regularization presents another approach by applying penalties directly to layer activations:

$$L_{\text{total}} = L_{\text{task}} + \lambda \sum_{l,i} |h_l[i]|$$

where $h_l[i]$ represents the i -th element of the activation vector h_l in layer l . This promotes sparse representations where only subsets of neurons activate for given inputs. Since many post-hoc interpretation methods directly rely on these activations, enforcing sparsity helps suppress uninformative or redundant activity. This idea will be explored further in chapter 5.

However, it is crucial to recognize that regularization does not improve interpretability on all levels. Higher sparsity and smoother decision boundaries can make networks more stable when inputs change, reducing erratic behavior. Additionally, post-hoc interpretation methods may produce clearer results when feature maps contain more zeros, as sparse activations reduce visual noise. Nevertheless, fundamental limitations persist. Deep networks remain non-convex optimization problems without theoretical guarantees. More critically, they are never mentally simulatable by humans.

These theoretical foundations of interpretability provide the groundwork for our subsequent investigations. In chapter 4, we will analyze specific attribution techniques and their evaluation, while chapter 5 explores architectural approaches for enhancing interpretability.

The chapter is based on the works detailed in papers [Nie+24a; Nie+24c], produced during this PhD research. The main contributions are:

1. the development of a novel deep learning pipeline for the identification of wood species in microscopy images of paper products;
2. the construction of a specialized dataset in collaboration with the Thünen Institute of Wood Research; and
3. the design of a new object detection algorithm tailored to wood microscopy images, building on and extending YOLOv7.

The microscopic wood identification process described here is structured as a two-stage approach: vessel detection followed by classification. While modern object detectors are capable of performing detection and classification in a single step [Wan+24], the two-stage approach offers several significant advantages.

First, it mirrors the established practices of human experts in microscopic wood analysis [Hel+18], making it a familiar and reliable framework. Building on this proven process allows for the development of new methods in future research. Second, this approach addresses the challenges of working with high-resolution microscopy images, which often require substantial GPU resources. While detection can be performed effectively on downsampled images to locate vessel elements, classification depends on the fine details of vessel morphology and therefore must be carried out on full-resolution images. By separating these tasks, we optimize GPU usage without compromising the accuracy of classification.

In the following sections, we first provide an overview of the dataset used in this study. Subsequently, we detail the vessel detection process, followed by a comprehensive explanation of the classification methodology. The chapter ends with an evaluation of our proposed wood identification pipeline.

3.1 DATASET

The dataset was constructed in collaboration with the Thünen Institute of Wood Research. Initially, no images or annotations were available, requiring the creation of the dataset from scratch. The sample preparation and image acquisition were conducted by experts at the Thünen Institute of Wood Research. These preparatory steps were not part of the author's work.

3.1.1 *Sample Preparation*

The generation of the dataset began with the careful preparation of biological samples, a process requiring specialized expertise. Hardwood samples were chosen for their distinctive anatomical features, particularly vessel elements, which serve as water-conducting cells. Compared to other wood cell types, such as fibers, tracheids, and parenchyma cells, vessel elements possess the most morphologically distinct characteristics, making them ideal for classification [Ilv95; Hel+18].

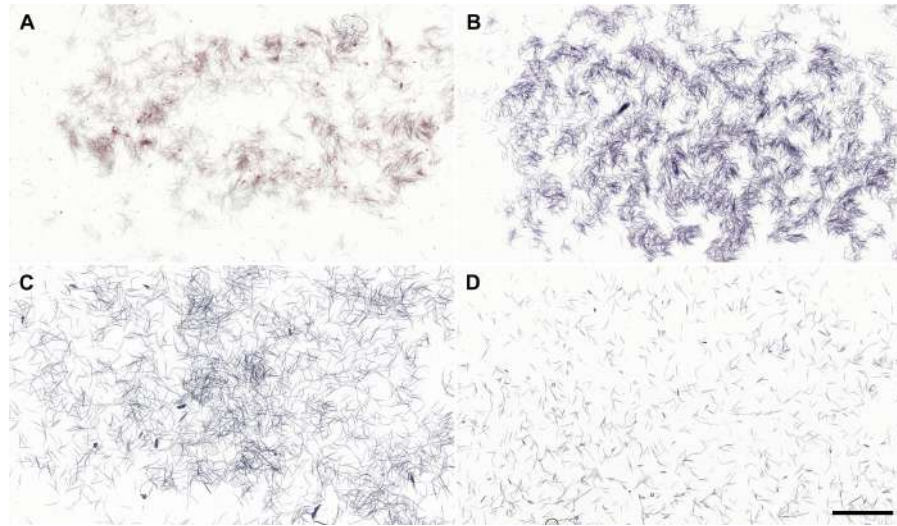


Figure 1: Overview images of differently stained macerated samples of A *Acacia*, B *Populus*, C *Hevea*, D *Salix*. Scale bar=5mm. Source: [Nie+24a].

Samples were selected from commonly processed timbers cultivated in plantations for pulp, paper, and fiberboard production. Both morphologically similar and highly distinct species were included to evaluate the versatility of the classification methods. Reference material was drawn from vouchered specimens in the Thünen Institute’s wood collection and other documented sources.

The cell compounds of wooden tissue were dissolved into individual cells using the maceration method described by [Fra45], a straightforward and reproducible technique. The process was carried out following established protocols [Hel+16; Hel+18].

To improve visibility, staining techniques were employed. Two stains were used: Alexander Herzberg solution and nigrosin (1 wt%). Alexander Herzberg solution, compliant with the TAPPI standard T 401 om-03, is widely used in paper testing but is non-permanent, requiring immediate examination. Nigrosin, by contrast, is a permanent stain that enhances the contrast of key morphological features such as pits [Hub+19]. These stains were critical in ensuring the visibility of the fine structures in the nearly transparent cellulose material. Preparations were performed by multiple technicians to introduce variability in cell density, reflecting the real-world differences observed in laboratory workflows.

3.1.2 Microscopy and Imaging

After biological preparation, the samples were digitized using the Axioscan 7 microscope slide scanner (Zeiss, Germany). The scanner, equipped with an Objective N-Achroplan 5x/0.15, provided high-resolution imaging over an area of approximately 8 cm² per slide. To capture depth variations and enhance structural detail, five focal levels were recorded per sample. These focal levels effectively resulted in five RGB images per sample, with each level highlighting different aspects of the structure. The multilevel imaging approach was crucial for capturing fine structural differences, such as intact vessel elements and fragmented pieces caused by the preparation process.

The images were recorded in RGB format at resolutions up to $32,000 \times 54,000$ pixels, with a pixel scale of $0.69 \times 0.69 \times 16.33 \mu\text{m}^3$. Each image file using the CZI format is approximately 1.3 GB in size. Since these large file sizes cannot fit into GPU memory, specialized techniques had to be developed to enable training of the neural networks. Figure 1 illustrates the diversity in staining contrast across multiple species.

Imaging was performed using ZEN Slidescan 3.5 software, which enabled consistent and high-throughput digitization of the dataset.

3.1.3 Annotation Pipeline

The annotation of vessel elements using bounding boxes was performed using ZEN blue 3.4 software. This software facilitated precise marking of vessel element locations by wood anatomists and other trained annotators. To reduce the manual effort required for annotation, we employed an active learning approach [GIG17]. Active learning is a machine learning strategy designed to minimize labeling effort by iteratively improving the training dataset in a targeted manner.

In this approach, a model (in this case, an object detection model) is first trained on a small, annotated dataset known as the initial training set. An acquisition function – often leveraging the model’s uncertainty – then selects the most informative data points from a pool of unlabelled samples. These selected samples are passed to an oracle (in this case, human experts) for labeling. The newly labeled data points are added to the training set, and the model is retrained on this expanded dataset. This iterative process of selection, labeling, and retraining continues, with the training set growing in size and quality over time. The advantage of active labeling is that the labeling effort is reduced by focusing annotation on the most informative samples.

Algorithm 3.1 Annotation Pipeline for Vessel Elements

Require: High-resolution microscopic images $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$

Ensure: Annotated dataset \mathcal{A} with verified bounding boxes

- 1: **Step 1: Strategic Initial Annotation**
 - 2: Select diverse subset $\mathcal{I}_{\text{manual}} \subset \mathcal{I}$
 - 3: Annotate $\mathcal{I}_{\text{manual}}$ manually to create $\mathcal{A}_{\text{manual}}$
 - 4: **Step 2: Active Learning Loop**
 - 5: Initialize $\mathcal{A}_{\text{verified}} = \mathcal{A}_{\text{manual}}$
 - 6: Train initial detector D_0 on $\mathcal{A}_{\text{verified}}$
 - 7: **while** not convergence AND budget allows **do**
 - 8: Select batch $\mathcal{I}_{\text{batch}} \subset \mathcal{I} \setminus \mathcal{I}_{\text{verified}}$
 - 9: Predict boxes $\mathcal{B}_{\text{pred}}$ on $\mathcal{I}_{\text{batch}}$ using current detector
 - 10: **for** each image $I \in \mathcal{I}_{\text{batch}}$ **do**
 - 11: Expert reviews predictions: $\mathcal{B}_{\text{corrected}}$
 - 12: Add $(I, \mathcal{B}_{\text{corrected}})$ to $\mathcal{A}_{\text{verified}}$
 - 13: **end for**
 - 14: Train updated detector on $\mathcal{A}_{\text{verified}}$
 - 15: Evaluate detector performance on validation set
 - 16: **if** performance plateaus OR budget exhausted **then**
 - 17: break
 - 18: **end if**
 - 19: **end while**
 - 20: **return** Final verified dataset \mathcal{A}
-

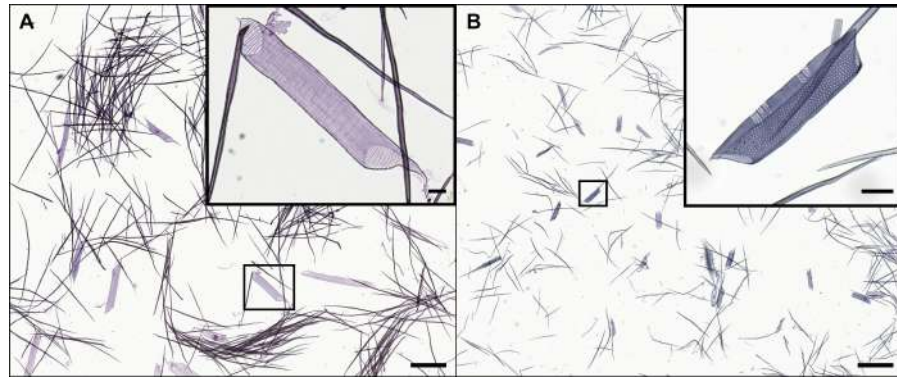


Figure 2: Image cutouts of vessel elements from overviews of A *Schima* and B *Populus*. Scale bars: overview=1mm, cutout=100 μ m. Source: [Nie+24a].

In our case, the initial dataset samples were chosen such that there is an approximately equal representation of images across different stainings and classes. Instead of using an automated acquisition function to select subsequent samples, we relied on human experts to guide the selection process. For example, experts identified errors like water bubbles in the images and prioritized adding similar difficult samples to the next annotation batch. This human-guided process helped the model improve where it struggled most. For generating the bounding boxes, we used the YOLOv7 algorithm [WBL22]. This is explained in more detail in section 3.2.

The algorithm 3.1 based on Nieradzic et al. [Nie+24a] shows our annotation pipeline.

After just a few iterations of the active learning loop, only small improvements in our metrics were observed. At this point, the experts had annotated and corrected 321 images, covering a total of 27,842 individual vessels. This illustrates the discrepancy between the relatively small number of images but the large number of annotated vessels. Figure 2 shows two examples of vessels in different images. After the first publication, the annotation process was continued, resulting in an expanded dataset.

3.1.4 Dataset

Our final dataset comprises approximately 1,614 images, primarily representing 13 genera of wood species. Among these, nine genera were of particular importance: *Acacia*, *Betula*, *Eucalyptus*, *Fagus*, *Hevea*, *Liquidambar*, *Populus*, *Salix*, and *Schima*. These genera were prioritized based on their relevance to the research and the availability of samples.

An overview of the dataset distribution is presented in figure 3. A substantial portion of the dataset remains unannotated, presenting an opportunity for exploring semi-supervised learning techniques such as pseudo-labeling [Cas+20; Ara+20] to leverage the unannotated images for improved model performance.

Each image contains multiple bounding boxes marking individual vessel elements. With the extended dataset, a total of 118,287 annotated bounding boxes have been created across the 1,614 images.

While the dataset is relatively large, all images were digitized using a single microscope (Zeiss Axioscan 7). Variability within the dataset arises from factors such as differences in staining during sample preparation and bright-

ness adjustments introduced during image digitization. This uniform imaging setup ensures consistency but may limit the generalizability of models to data collected under different imaging conditions.

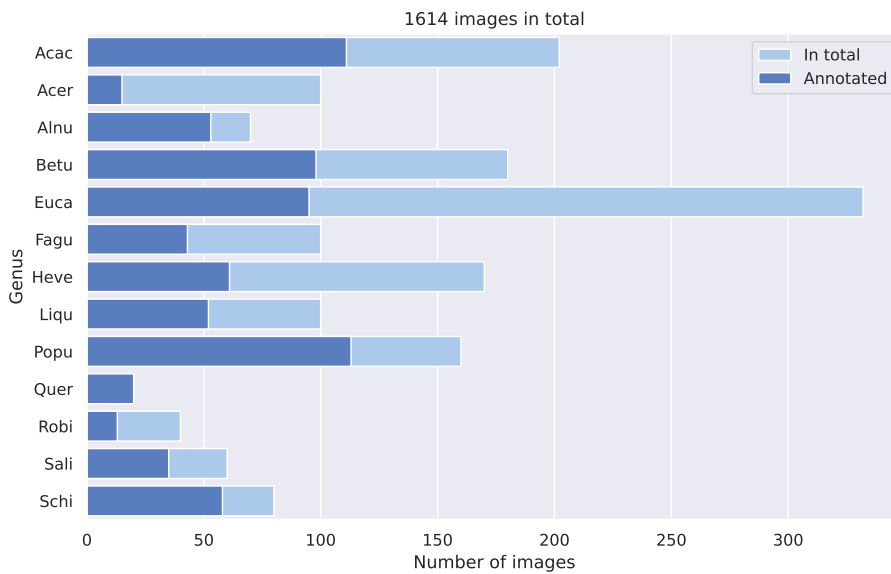


Figure 3: Visualization of the distribution of images across different classes in the dataset. This chart illustrates the number of annotated vessel elements for each image type. The genera were abbreviated with the first four letters.

The dataset also presents a challenge in the variation of bounding box distributions across genera. For example, images of *Hevea* typically contain only a few bounding boxes, while other genera, such as *Eucalyptus*, may feature hundreds of annotated vessels in a single image. Addressing this disparity requires careful handling during training to ensure balanced performance across all classes.

As previously mentioned, each image in the dataset consists of five focal levels. These levels highlight different morphological features of the vessel elements. Figure 4 illustrates how the focal plane influences the visibility of specific structures.



Figure 4: Comparison of focal levels for a single vessel element from the genus *Eucalyptus*. Depending on the focal level, distinct features such as vessel-ray pits or inter-vessel pits come into focus. Scale bar = 100 μ m.

Unlike classification, which relies both on internal (e.g. pits) and external structures, detection focuses mainly on the overall shape of the vessel to distinguish it from other objects, such as fibers. As a result, object detection prioritizes the vessel's external geometry over finer internal details visible in different focal planes, as we will demonstrate later.

3.1.5 Training and validation split

Before training neural networks on the dataset, it is crucial to carefully define the data-splitting strategy for training and validation. Proper splitting prevents the model from learning irrelevant patterns or “wrong” features, a phenomenon known as data leakage [KN22]. Data leakage occurs when the model uses information not genuinely predictive of the target variable. For instance, if images of certain classes are consistently darker, the network might exploit brightness differences instead of learning meaningful features, leading to biased and unreliable predictions.

The maceration process described in section 3.1.1 separates wood tissue into individual cells to enable microscopic examination. While anatomical features remain consistent at the genus level, there are two main sources of variation in the samples. First, as wood is a natural product, there is inherent biological variation within species. However, experience shows that diagnostic features remain reliable for genus-level identification. Second, variations can arise from differences in individual preparation techniques. To mitigate the risk of the network learning these preparation-specific artifacts, we ensure that each genus in the dataset is represented by at least two independent macerates. Images from these macerates are then split into separate subsets for training and validation. This approach reduces the likelihood of overfitting to preparation-specific characteristics and promotes robust generalization to unseen data.

To achieve statistically reliable results, we employ cross-validation [HTF09], a widely used technique for evaluating machine learning models. Cross-validation involves dividing the dataset into multiple subsets (or “folds”), training the model on a subset of folds, and validating it on the remaining ones. This process is repeated so that each fold serves as the validation set once, with the final performance estimate obtained by averaging results across iterations.

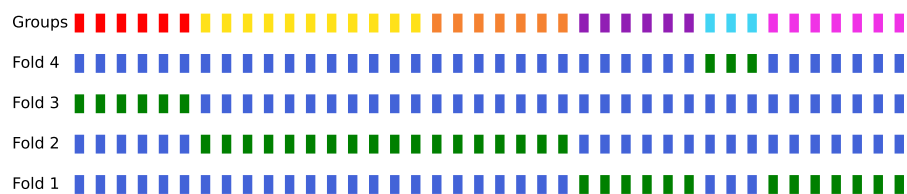


Figure 5: This plot shows a 4-fold cross-validation with 6 groups. For simplicity, we do not consider different classes. Green markers indicate test data, while blue markers represent training data. Each group appears in exactly one validation fold, ensuring no group is repeated in the validation set across folds.

For our dataset, we adopt group-stratified cross-validation. An example is shown in figure 5. In this method, the data is stratified by class to maintain consistent class distributions in both training and validation sets. Additionally, we treat each macerate as a distinct group, ensuring all samples from a single macerate are assigned exclusively to either the training or validation set. This grouping effectively eliminates data leakage by preventing the model from learning preparation-specific features, thereby enhancing its ability to generalize.

3.2 VESSEL DETECTION

3.2.1 Image size

The detection step focuses on locating vessel elements within the microscopy images. Vessel elements contain genus-specific morphological features, critical for classification, which require high-resolution images to discern. However, detection can still be effectively carried out on lower-resolution images, as previously mentioned.

There are two main strategies to manage high-resolution images:

- **Resizing:** This approach reduces the image size to fit within the GPU memory, enabling faster inference. However, this comes at the cost of some loss in resolution, which could impact the accuracy of vessel detection.
- **Tiling the images** [ÜÖÇ19]: This method splits the image into smaller sections, which allows us to work within the GPU memory limits. However, tiling results in slower inference times, and additional care is needed to process vessel elements located at the boundaries between tiles.

In practice, these strategies can be combined with other optimization techniques. One of these methods is Automatic Mixed Precision (AMP) [Mic+18], which uses 16-bit precision (fp16) for certain operations instead of the standard 32-bit precision. This reduces memory usage and speeds up computation without sacrificing accuracy.

Another helpful technique is gradient accumulation. When the image or batch size is too large to fit into memory, gradient accumulation allows us to simulate a larger batch size by accumulating gradients over several smaller mini-batches before updating the model’s weights. This technique enables us to work with larger batch sizes.

Finally, if memory issues persist, other adjustments can be made, such as reducing the batch size or modifying the model architecture (e.g., reducing the number of filters in convolutional layers).

In our experiments, we examined the size distribution of vessel elements based on the bounding box dimensions (maximum of width and height). Table 1 summarizes the mean size (\pm standard deviation) for various genera. The average vessel element size, with the associated standard deviation, was sufficient for detection even at a reduced image resolution (10% of the original). For instance, the genus *Hevea* had a mean vessel size of approximately 1391 ± 422 pixels, with other genera like *Acer* and *Quercus* having mean sizes of around 487 ± 111 pixels and 701 ± 156 pixels, respectively. These sizes are still large enough for detection at lower resolutions.

To put this into perspective, the original image size is around $32,000 \times 54,000$ pixels. At 10% resolution, this corresponds to images that are approximately 3200×5400

Genus	Mean \pm Std (px)
Acac	468.92 ± 164.94
Acer	487.43 ± 111.39
Alnu	861.56 ± 281.81
Betu	975.67 ± 296.58
Euca	758.78 ± 244.24
Fagu	741.08 ± 164.35
Heve	1391.43 ± 422.39
Liqu	1471.97 ± 469.98
Popu	813.28 ± 234.92
Quer	701.20 ± 156.86
Robi	382.49 ± 172.01
Sali	646.28 ± 191.09
Schi	1574.52 ± 534.37

Table 1: Vessel element size statistics (bounding box dimensions) for various genera.

pixels. This reduced resolution maintains sufficient detail for vessel element detection, as the smallest vessels remain identifiable.

As a result, resizing the images, combined with the use of Automatic Mixed Precision (AMP) and a reduction in batch size, proved to be an effective strategy for training on consumer-grade hardware. We will also show empirically later in figure 11 that resolutions that are too high are not necessary.

3.2.2 *Choice of object detection algorithm*

The selection of an appropriate object detection algorithm is crucial for our study. In the following, we will detail some possible choices.

YOLO (You Only Look Once) [Red+16] is known for its speed and efficiency, thanks to its single-stage detection pipeline. Unlike two-stage detectors like RCNN [Gir+14], Faster RCNN [Ren+16], and Mask RCNN [He+18], which generate region proposals and then classify them, YOLO completes the detection process in one pass. This makes it much faster, which is crucial for high-throughput applications. Although early YOLO versions were less accurate, especially for small or overlapping objects, newer versions have significantly improved, achieving accuracy that is comparable to or even better than two-stage models in many cases.

Since its introduction in 2016, YOLO has been widely used for object detection tasks. Recent versions like YOLOv7 [WBL22], YOLOv9 [WYL24], and YOLOv10 [Wan+24] show results similar to or better than two-stage detectors on standard benchmarks. For example, YOLOv7 achieves 56.8% mAP on COCO, which outperforms a Mask RCNN configuration with InternImage [Wan+23] (55.3% mAP). Similarly, YOLOv9 and YOLOv10 score 55.6% mAP and 54.4% mAP, respectively. These results show that modern single-stage detectors can offer similar or better performance than two-stage models, while maintaining their speed advantage.

Other single-stage detectors, like SSD (Single Shot MultiBox Detector) [Liu+16] and RetinaNet [Lin+18], use similar approaches to YOLO but have become less popular. SSD improved detection of objects at various sizes with multi-scale feature maps, while RetinaNet addressed class imbalance with Focal Loss. However, both models have been largely surpassed by YOLO, which now integrates some of these ideas and provides better overall performance.

Transformer-based models, like DETR (DEtection TRansformer) [Zha+22; Zha+24], offer a different approach. DETR uses attention mechanisms to model global relationships in images and eliminates the need for anchor boxes. While effective in some cases, DETR has practical limitations. It requires more time to train [Gao+21] and has higher computational demands, especially when handling high-resolution images typical in microscopy. Additionally, DETR's inference speed is slower than YOLO's, making it less suitable for applications that need both efficiency and scalability.

Due to its widespread use and strengths in handling large-resolution microscopy images, we selected YOLOv7 for our baseline study.

3.2.3 *YOLO*

In the following, we will explain how the YOLOv7 algorithm works and how we adapted and applied it to our dataset for object detection. We base our description of YOLO not only on the paper but also on the actual code, which may account for slight differences between the two. This approach ensures that our explanation more accurately reflects the underlying implementation.

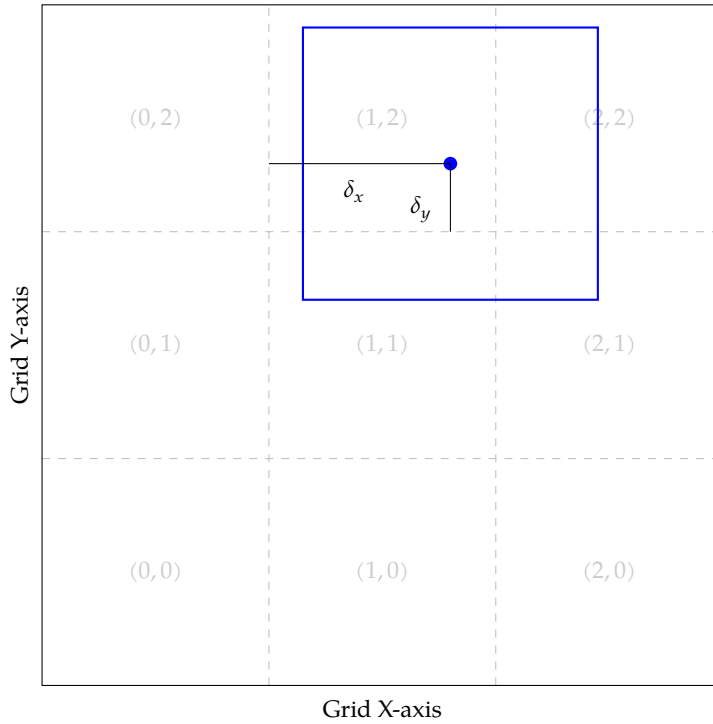


Figure 6: Illustration of YOLO’s coordinate decomposition for object detection. The blue box represents the ground truth bounding box. Here, $\delta_x = 0.8$ and $\delta_y = 0.3$, leading to a predicted object center at $(x, y) = (1.8, 2.3)$.

Traditional object detection methods like R-CNN first identify regions that might contain objects and then classify each region separately. In contrast, YOLO predicts object locations across the entire image in a single pass. YOLO achieves this by directly utilizing the feature maps produced by convolutional layers.

The network processes an input image of size $W \times H \times 3$ through multiple layers that progressively reduce the spatial dimensions. This reduction, or downsampling, is typically achieved through either convolutional layers or max pooling layers with a stride of 2, where each such layer halves the spatial dimensions of its input. This is discussed in more detail in section 2.2.2. Common architectures accumulate five such downsampling operations through the network, leading to a total stride of 32 (2^5). For example, with an input image of 224×224 , this results in a feature map of size 7×7 ($224/32 = 7$).

Each position in this feature map corresponds to a region in the original image (in our example with a 224×224 input and 7×7 feature map, each cell represents roughly a 32×32 pixel region) and is responsible for detecting objects whose centers fall within that region. When an object is detected, its position is represented through two components, as shown in figure 6. First, the grid cell indices (i, j) indicate which feature map cell contains the object’s center. Second, the offsets (δ_x, δ_y) specify the precise location within that cell, where δ_x and δ_y are normalized values between 0 (inclusive) and 1 (exclusive).

More formally, this coordinate decomposition can be understood through the floor function notation $\lfloor \cdot \rfloor$. For any position (x, y) relative to the grid structure, we can decompose it into integer and fractional components:

$$x = \lfloor x \rfloor + \delta_x, \quad y = \lfloor y \rfloor + \delta_y, \quad (9)$$

where $\lfloor x \rfloor$ and $\lfloor y \rfloor$ correspond to the grid cell indices i and j respectively, while δ_x and δ_y represent the offsets within the cell, with $0 \leq \delta_x, \delta_y < 1$.

The mapping between image coordinates and feature map coordinates is determined by the network's stride. If S_f represents the stride of 32, then a position (x, y) in the original image corresponds to:

$$x = S_f(\lfloor x \rfloor + \delta_x) = S_f(i + \delta_x), \quad y = S_f(\lfloor y \rfloor + \delta_y) = S_f(j + \delta_y).$$

For each feature map position, the network predicts not only the object's location but also its dimensions (width and height normalized to the image size), a confidence score indicating the likelihood of an object being present, and probabilities for each possible object class. Instead of a single bounding box per grid cell, the architecture allows for multiple predictions – the number of bounding boxes per cell, denoted as B , is a parameter that varies across different YOLO implementations.

As a result, the output of the network is a tensor of size $S \times S \times (B \times 5 + C)$, where B is the number of bounding boxes per grid cell, 5 refers to the four bounding box coordinates and the confidence score, and C is the number of classes. In the previous example, $S = 7$, $B = 1$ and $C = 0$.

Loss function

The training process involves minimizing a custom loss function that addresses both localization and classification errors. The total loss is expressed as:

$$L = L_{\text{coord}} + L_{\text{conf}} + L_{\text{class}},$$

where L_{coord} penalizes inaccuracies in bounding box localization, L_{conf} measures the quality of confidence predictions, and L_{class} accounts for classification errors.

The coordinate loss L_{coord} is designed to encourage the predicted bounding box center and dimensions to closely match the ground truth. It is defined as [Red+16]:

$$L_{\text{coord}}(b, \hat{b}) = (x - \hat{x})^2 + (y - \hat{y})^2 + (\sqrt{w} - \sqrt{\hat{w}})^2 + (\sqrt{h} - \sqrt{\hat{h}})^2,$$

where x, y, w, h is a ground truth bounding box b , and $\hat{x}, \hat{y}, \hat{w}, \hat{h}$ is the corresponding predicted box \hat{b} . To better handle scale variations in bounding box sizes, the square root is applied to the width and height terms.

More recent YOLO versions, including YOLOv7, have adopted Intersection over Union (IoU)-based losses for localization. The definition is as follows:

$$L_{\text{coord}}(b, \hat{b}) = 1 - \text{IoU}(b, \hat{b}),$$

where

$$\text{IoU}(b, \hat{b}) = \frac{\text{Area of Intersection}}{\text{Area of Union}}.$$

To further improve localization, variants such as Generalized IoU (GIoU) [Rez+19], Complete IoU (CIoU) [Zhe+21], and Distance IoU (DIoU) [Zhe+19]

have been proposed. For instance, CIoU incorporates aspect ratio and center distance into the loss, providing better alignment between predicted and ground truth boxes. We will discuss this later in more detail. YOLOv7 uses CIoU by default.

The confidence loss L_{conf} optimizes the quality of confidence score predictions. This score represents the likelihood that a bounding box contains an object. Typically, the confidence loss is formulated using Binary Cross-Entropy (BCE) applied to the Intersection over Union (IoU) between the predicted and ground truth bounding boxes:

$$L_{\text{conf}}(b, \hat{b}) = \text{BCE}(\hat{o}, \text{IoU}(b, \hat{b})),$$

where \hat{o} denotes the predicted confidence. The BCE loss is defined as:

$$\text{BCE}(\hat{p}, p) = -(p \cdot \log(\hat{p}) + (1 - p) \cdot \log(1 - \hat{p})),$$

with $\hat{p} \in [0, 1]$ representing the predicted probability and $p \in \{0, 1\}$ indicating the ground truth label. Intuitively, the predicted confidence $\hat{p} = \hat{o}$ reflects how strongly the network believes that the bounding box is valid. When no object is present at a specific location, $p = \text{IoU}(b, \hat{b})$ is set to 0 (i.e. there is no overlap with some ground truth box). Refer also to equation (4).

Finally, the classification loss L_{class} typically uses cross-entropy to ensure accurate class predictions. However, in our case, classification is not performed during detection. Instead, it is handled in a subsequent step, where a separate classifier processes the cropped bounding box patches. We can, therefore, set $L_{\text{class}} = 0$ and reduce the network outputs to $S \times S \times (B \times 5)$.

In the original YOLO model [Red+16], only the final feature map was used for bounding box regression. In contrast, modern YOLO variants like YOLOv7 [WBL22] leverage multiple layers of the feature pyramid, enabling the network to handle objects of varying scales more effectively.

By incorporating predictions from higher-resolution feature maps, modern variants can better localize small objects, which may be difficult to detect using only coarse, low-resolution features. At the same time, predictions from deeper, lower-resolution layers of the pyramid capture context and are well-suited for detecting larger objects.

This strategy of incorporating multiple feature maps affects the total loss function L , which we have previously defined. In models that use a feature pyramid, the loss is summed across all layers:

$$L_{\text{all}} = L_1 + L_2 + \dots + L_n,$$

where n is the number of feature maps used, typically $n = 3$ or $n = 4$. YOLOv7, for example, offers variants with both 3 and 4 feature maps.

Architecture

Table 2 provides a comparison of various YOLOv7 models, evaluated on the COCO dataset [Lin+15a].

To understand the YOLOv7 architecture, it is important to first mention that object detectors usually consist of three main components: the backbone, the neck and the head. The backbone usually consists of a pre-trained classification network, such as ResNet-18, which extracts hierarchical features from the input

Model Name	#Params	#Feature Maps	Test Size	AP ₅₀
YOLOv7-tiny	6.2M	3	416	52.8%
YOLOv7	36.9M	3	640	69.7%
YOLOv7-X	71.3M	3	640	71.2%
YOLOv7-W6	116.2M	4	1280	72.6%
YOLOv7-E6	97.2M	4	1280	73.5%
YOLOv7-D6	154.7M	4	1280	73.8%
YOLOv7-E6E	151.7M	4	1280	74.4%

Table 2: Comparison of YOLOv7 Model Variants. The table presents key characteristics of each model: the number of parameters (#Params), the number of feature maps used, the input image size (test size), and the average precision at an IoU threshold of 50% (AP₅₀) for COCO.

image to form a feature pyramid. These features, which span multiple levels of resolution, are then processed by the neck, which summarizes and refines the feature maps. Finally, the head creates the actual predictions, including bounding box coordinates and class labels.

The backbone in YOLOv7 is based on CSPDarknet [Wan+19a], an enhanced version of the Darknet architecture [RF18]. Darknet is similar to ResNet through its use of residual connections. The main difference between the two architectures is that Darknet employs both 1×1 and 3×3 kernels, whereas ResNet primarily relies on 3×3 kernels. Additionally, Darknet features optimized filter sizes and number of layers specifically tailored for object detection.

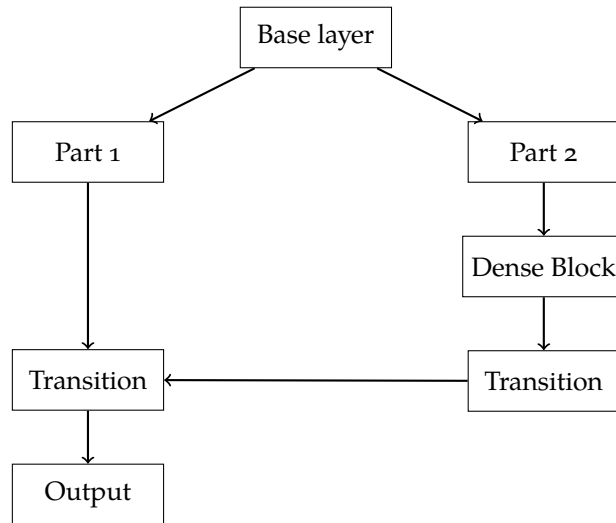


Figure 7: Architecture of CSPDenseNet showing the cross stage partial connections with dense block integration. The design splits the feature maps to create a more efficient gradient flow path. Adapted from Wang et al. [Wan+19a].

CSPDarknet introduces Cross-Stage Partial (CSP) connections to the standard Darknet architecture, which enhance gradient flow during training while reducing computational overhead. This module is notably implemented in YOLOv4 as CSPDarknet53 [BWL20] and in YOLOX/YOLOv5 as a variant of that design [Ge+21b]. The CSP approach is to split feature maps at the beginning of each stage into two paths: one traverses the dense block (containing internal residual connections based on concatenation), while the other

bypasses it entirely. These paths are subsequently concatenated at the end. Therefore, it is a hybrid architecture that combines the strengths of both ResNet and DenseNet. Figure 7 visualizes this approach.

YOLOv7 extends the CSP concept through its implementation of Extended Efficient Layer Aggregation Networks (E-ELAN), which is a modification of the standard ELAN architecture introduced by Wang, Liao, and Yeh [WLY22]. E-ELAN primarily differs from CSP by incorporating additional layer stacking, which increases the network’s depth. In YOLOv7, variants with four feature maps utilize E-ELAN, while those with three feature maps employ the simpler ELAN, with less layers.

To reduce computational costs in the backbone, the YOLOv7-tiny variant employs max pooling for spatial downsampling, while other variants rely on strided convolutions. Additional differences include variations in filter sizes and the number of convolutional layers.

The neck of YOLOv7 integrates a Path Aggregation Network (PANet) [Liu+18], which enhances feature fusion across different levels of the backbone. By aggregating features across multiple layers, PANet improves the network’s ability to capture both fine-grained details and broader contextual information. Unlike the standard Feature Pyramid Network (FPN) [Lin+17], which includes only a top-down path, PANet introduces an additional bottom-up path for more effective feature aggregation. To aggregate features from different layers, the top-down path uses downsampling through strided convolutions or max pooling, while the bottom-up path employs upsampling. In YOLOv7, nearest neighbor interpolation is used in the bottom-up path, and strided convolutions are used in the top-down path.

Finally, the head of YOLOv7 generates predictions for object locations and classes. YOLOv7 uses a coupled head, similar to YOLOv3 through YOLOv5. This means there is a single convolutional pathway that produces the $5 + C$ outputs. In contrast, YOLOX introduced a decoupled head [Ge+21b]. It splits the $5 + C$ outputs per bounding box into two separate convolutional paths: one for bounding box coordinates (5 outputs) and another for classification (C classes). However, since we use a separate classifier for vessel detection, in our case a decoupled head is unnecessary, as only a single branch is needed.

While the architectural principles behind the different YOLOv7 variants are fundamentally the same, there are also many small differences apart from number of layers and filter sizes. Notably, models trained on images of size 1280×1280 incorporate an auxiliary head with deep supervision, adding an extra loss term to guide training and improve performance. Since YOLOv7 builds on earlier versions like Scaled-YOLOv4 [WBL21], the code base also contains other undocumented changes.

3.2.4 *WoodYOLO*

Although YOLOv7 worked well as a baseline for vessel detection, it was not fully optimized for our specific needs. The complexity of YOLOv7’s code-base made it difficult to identify which parts of the architecture were most effective for our application. As a result, we decided to systematically rewrite YOLOv7, evaluating each component to determine which changes truly improved performance. The outcome of this process is WoodYOLO – a cleaner, more efficient version of YOLOv7, incorporating only the modifications that made a significant impact on performance.

WoodYOLO retains the basic output structure of YOLOv7, with dimensions $S \times S \times (B \times 5 + C)$. However, we set $B = 1$ and $C = 0$, meaning each grid

cell predicts a single bounding box without class labels. This simplification is possible because the classification task is handled separately by a dedicated classifier, removing the need to include it in the detection model. The reduction in bounding boxes is motivated by several key considerations.

In YOLOv7, the B bounding boxes correspond to anchor boxes – predefined bounding box shapes optimized for a specific dataset using methods like clustering. These anchor boxes simplify bounding box prediction by requiring the model to predict only offsets relative to the predefined widths and heights. However, determining these optimal anchor boxes requires dataset-specific clustering, which introduces additional complexity. By limiting the model to a single bounding box per grid cell, WoodYOLO eliminates the need for this step. Instead, the width and height of the bounding box are directly predicted by the network.

Additionally, research such as YOLOX [Ge+21b] has shown that removing anchor boxes can improve detection performance. This finding suggests that multiple anchor boxes per grid cell may not be necessary. Anchor boxes were initially essential when the original YOLO architecture used only a single feature map (e.g., 15×15) for detection. Modern versions of YOLO, however, leverage feature pyramids, which combine predictions from multiple feature maps at different scales.

For instance, when processing an image of size 2048×2048 , the network typically uses three feature maps of sizes 256×256 , 128×128 , and 64×64 . The total number of bounding boxes across these maps is:

$$\text{Total bounding boxes} = 256^2 + 128^2 + 64^2 = 65536 + 16384 + 4096 = 86016.$$

In YOLOv7, this number is further multiplied by B , typically $B = 3$ or $B = 5$, resulting in a large number of predictions. In contrast, the original YOLO [Red+16] used only a single 7×7 feature map with $B = 2$, yielding a total of 98 theoretical boxes. While this small number of predictions was sufficient for the simpler tasks of early object detection models, it would be inadequate for most modern applications.

Finally, reducing the number of bounding boxes also simplifies the loss function. Anchor-based methods require matching ground truth boxes to the best-fitting anchors, which significantly increases computational overhead during training. WoodYOLO avoids this bottleneck, leading to faster training and a more efficient overall pipeline.

Network outputs

The removal of anchor boxes in WoodYOLO necessitated adjustments to how we define and interpret the network’s outputs, particularly for bounding box dimensions within grid cells. Apart from generalizing YOLOv7’s approach for normalized coordinates (\hat{x}, \hat{y}) and confidence scores (\hat{o}) , we also introduce important modifications to accommodate our anchor-free design.

Each feature map f produces five outputs per grid cell, corresponding to the bounding box’s center coordinates, dimensions, and confidence score. For a single grid cell at location (i, j) within a feature map f of dimension $g_w \times g_h \times 5$, the outputs are defined as:

$$\begin{aligned}
\hat{x} &= S_f(i + \alpha\sigma(f_{i,j,1}) + \beta), \\
\hat{y} &= S_f(j + \alpha\sigma(f_{i,j,2}) + \beta), \\
\hat{w} &= \sigma(f_{i,j,3})^\gamma \cdot g_w \cdot m_w, \\
\hat{h} &= \sigma(f_{i,j,4})^\gamma \cdot g_h \cdot m_h, \\
\hat{o} &= \sigma(f_{i,j,5}),
\end{aligned}$$

where $f_{i,j,k}$ is the k th raw output of the neural network at grid cell (i, j) and $\sigma(\cdot) \in [0, 1]$ is the sigmoid activation function (see equation (3)). S_f represents the stride of the feature map f . Here, $\sigma(f_{i,j,1})$ and $\sigma(f_{i,j,2})$ correspond to δ_x and δ_y in equation (9).

The scalars α and β control the flexibility of the grid boundaries, while γ is an object size bias. The parameters m_w and m_h define the maximum allowed size of the bounding boxes relative to the image dimensions.

Since we have an estimate of the typical size of the objects (vessels), we introduced the parameters $m_w, m_h \in [0, 1]$ to limit the size of the predictions and prevent them from being unreasonably large.

As shown in table 1, the average maximum width or height of a vessel in our dataset is approximately 1600 pixels. Given an image size of 32000×54000 , this means a vessel would occupy at most 10% of the image dimensions. Thus, setting $m_w = m_h = 0.1$ serves as an appropriate initial choice for these hyperparameters.

YOLO9000 [RF16] and YOLOv3 [RF18] used coordinate transformations for x and y as $i + \sigma(f_{i,j,1})$ and $j + \sigma(f_{i,j,2})$, respectively. However, this approach constrained objects to always fall within the grid cells. To address this, YOLOv4 [BWL20] introduced a multiplicative factor, allowing the predicted center point to extend beyond the boundaries of its grid cell, thus improving localization flexibility. Later, YOLOv7 added a shifting factor to adjust the starting point of the coordinate range, further enhancing flexibility.

Here, we use a generalized form of this transformation, where the user can set two hyperparameters, α and β . This transformation maps the offsets from the range $[0, 1]$ to a more flexible range $[\beta, \alpha + \beta]$.

For YOLOv7, the parameters are set to $\alpha = 2$ and $\beta = -0.5$. This means that the center of coordinates can extend half a cell to the left (-0.5) and one and a half cells to the right ($+1.5$). Therefore, out of bound solutions are also possible. If we were to set $\alpha = 2$ and $\beta = 0$, the center can shift by up to two grid cells to the right, but this approach would introduce even more asymmetry and directional bias. Mathematically, the theoretical width maximum for α and β is $\alpha = g_w - 1$ and $\beta = -i$. This would allow the predicted bounding box center to be in the entire grid. However, such large values can destabilize the training process by making the model overly flexible, leading to poor convergence.

In our dataset, some vessels have bounding boxes that are very close to each other. By using an extended range for the coordinate transformations, the model can theoretically have two grid cells adjacent to each other both predicting the same bounding box, differing only in the predicted size of the bounding box. This helps the model effectively handle objects that are tightly clustered together or close to grid boundaries. This is especially useful for smaller feature maps.

For the width and height predictions (\hat{w}, \hat{h}) , we set $\gamma = 2$ as is also done in YOLOv7. This approach helps to bias the network towards smaller predictions,

which is particularly useful when dealing with small object sizes. For example, if the ground truth value is 0.4, but the network predicts 0.45, squaring the output results in $0.45^2 = 0.2025$, which is a significant deviation from the true value. However, if the ground truth is 0.9, and the network predicts 0.95, squaring the output gives $0.95^2 = 0.9025$, which is much closer to the true value. In the case of small values, even a small error (such as a 0.05 difference) leads to a substantial deviation when squared, as seen from 0.4^2 vs 0.45^2 . This approach is similar to the one used in the original YOLO [Red+16], where the loss function $L_{\text{coord}}(b, \hat{b})$ incorporates the square root of the width and height for bounding box predictions.

Loss function

In our model, we adopt the standard loss functions for coordinate prediction, L_{coord} , and confidence prediction, L_{conf} , as used in YOLOv7. However, unlike YOLOv7, which relies exclusively on the fixed cIoU (complete IoU) loss for bounding box regression, we introduce flexibility by allowing the user to choose from different IoU loss variants. This flexibility allows the loss function to be tailored to specific datasets, as the choice of IoU loss can influence performance by incorporating dataset-specific characteristics. Table 3 provides a comparison of different IoU loss variants and their respective use cases.

Loss Name	Use Case	Implementation
IoU	Best for clearly visible objects that typically have good overlap with ground truth.	IoU computes the ratio of intersection to union of predicted and ground truth boxes.
GIoU [Rez+19]	Handles cases where predicted and ground truth boxes have zero or minimal overlap.	Extends IoU by considering the smallest enclosing box, enabling gradients for non-overlapping cases.
DIoU [Zhe+19]	Crowded scenes with multiple overlapping objects.	Enhances IoU by adding a penalty for the distance between predicted and ground truth box centers.
CIoU [Zhe+21]	Ensures better alignment of object centers and aspect ratios.	Combines IoU with penalties for center distance and aspect ratio differences.

Table 3: Comparison of IoU Loss Variants. The table outlines specific use cases and implementation details for different IoU loss types.

In some cases, the complete IoU may not be the optimal choice, particularly when the aspect ratio between the predicted and ground truth bounding boxes is not as important or is relatively consistent. For our application, where the goal is to identify as many objects as possible, a slight mismatch in bounding box size is not critical. In such cases, alternative variants like GIoU or DIoU may lead to better results by not focusing as much on achieving a perfect aspect ratio.

An important aspect of the loss computation involves determining which predicted boxes are used for IoU calculations. YOLOv7 follows a two-step approach: first, it selects the anchor corresponding to the ground truth box’s position in the grid, defined as $(i, j) = (\lfloor x \rfloor, \lfloor y \rfloor)$. Second, it applies multi-positives (also known as center sampling) [Ge+21b; Tia+19] to include additional neighboring boxes in the assignment process. While this strategy can be effective, YOLOv7 fixes the selection process, offering no user configurability.

In contrast, our WoodYOLO model introduces a novel configurability, allowing users to define the number of neighboring boxes to include. This user-defined flexibility supports three configurations.

Configura- tion	Grid	Description									
0 Neighbors	<table border="1"> <tr><td>·</td><td>·</td><td>·</td></tr> <tr><td>·</td><td>■</td><td>·</td></tr> <tr><td>·</td><td>·</td><td>·</td></tr> </table>	·	·	·	·	■	·	·	·	·	Only the ground truth box (■) is selected for loss computation.
·	·	·									
·	■	·									
·	·	·									
2 Neighbors	<table border="1"> <tr><td>·</td><td>□</td><td>·</td></tr> <tr><td>·</td><td>■</td><td>□</td></tr> <tr><td>·</td><td>·</td><td>·</td></tr> </table>	·	□	·	·	■	□	·	·	·	The ground truth (■) plus two adjacent boxes (□) in vertical and horizontal directions.
·	□	·									
·	■	□									
·	·	·									
4 Neighbors	<table border="1"> <tr><td>·</td><td>□</td><td>·</td></tr> <tr><td>□</td><td>■</td><td>□</td></tr> <tr><td>·</td><td>□</td><td>·</td></tr> </table>	·	□	·	□	■	□	·	□	·	The ground truth (■) plus all four adjacent boxes (□) in cardinal directions.
·	□	·									
□	■	□									
·	□	·									

Table 4: Neighbor selection configurations for loss computation. The black square (■) represents the ground truth box position, white squares (□) show selected neighboring boxes, and dots (·) indicate unused grid cells.

In table 4, "■" represents the original ground truth bounding box, while "□" indicates the neighboring boxes that are incorporated into the loss calculation. Cells marked with "·" are not involved in the computation. In the 0 neighbors configuration, the loss L_{coord} is computed exclusively for the original ground truth box "■", without considering any neighboring boxes ("□"). For the 2 neighbors configuration, the closest neighboring boxes along the horizontal and vertical directions are included in the calculation. In the example provided, this corresponds to the boxes immediately to the right and above the ground truth box. Finally, in the 4 neighbors configuration, the bounding boxes from all non-diagonal directions are considered alongside the ground truth box.

Object detection involves a one-to-many relationship, where a single ground-truth box can correspond to multiple valid predicted boxes; this strategy aims to replicate that relationship within the loss function. It is important to note that this process does not alter the number of bounding boxes generated by the model, which remains defined by the feature map dimensions. Instead, it only influences the loss function during training, leaving the prediction process unchanged.

After selecting candidate bounding boxes, filtering can be applied through label assignment strategies to again reduce the number of valid boxes per object. We experimented with state-of-the-art techniques such as Optimal Transport Assignment (OTA) [Ge+21a], SimOTA [Ge+21b], and Task Alignment Learning (TAL) [Fen+21]. These methods aim to optimize the matching between ground truth and predicted boxes. However, in our experiments, they did not yield significant improvements for our specific application and incurred a noticeable decrease in training speed due to the computational cost of the filtering step. Moreover, TAL is incompatible with our framework because it relies on classification predictions, which are absent in our pipeline. Consequently, we omit a formal description of these methods.

Architecture

To adapt and optimize YOLOv7 for our vessel detection task, we adopted the common approach that redefines the architecture into three distinct components: the backbone, neck, and head. This modularity allows us to experiment with alternative designs for each component and tailor them to our specific requirements, as opposed to relying solely on the monolithic YOLOv7 struc-

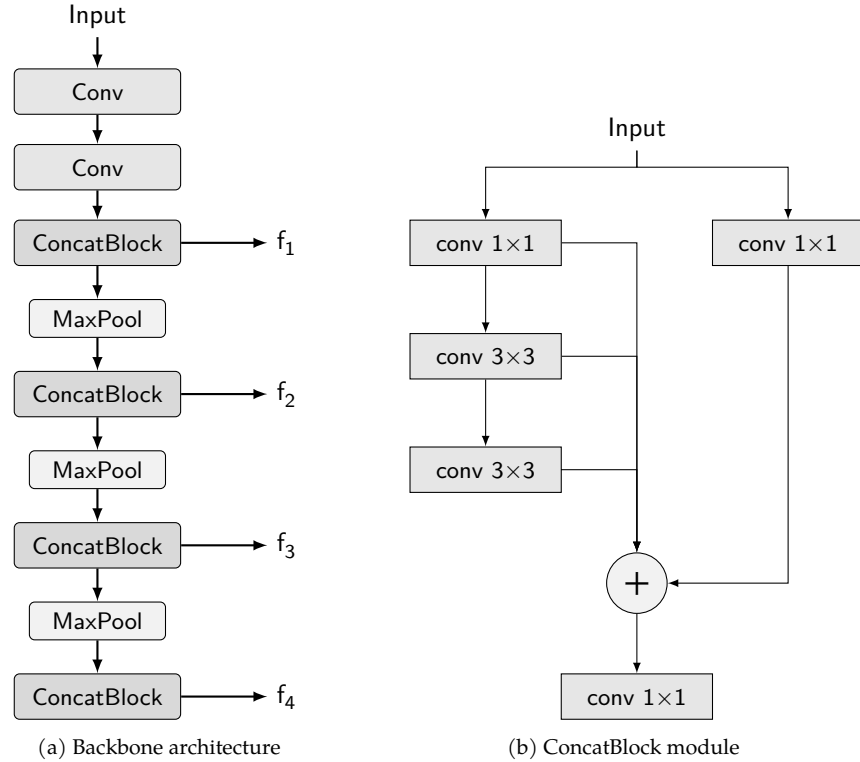


Figure 8: Backbone architecture. (a) The backbone architecture uses stacked ConcatBlocks and max pooling operations to generate multi-scale features. (b) The ConcatBlock structure combines features through multiple parallel pathways with dense connections to the concatenation node.

ture. This flexibility also enables precise evaluation of the contributions of individual components to overall performance.

We began with the YOLOv7-tiny backbone due to its simplicity and computational efficiency, especially given our need to process high-resolution images within GPU memory constraints. However, several modifications were made to better suit our dataset and goals.

The LeakyReLU activation used in YOLOv7-tiny was replaced with ReLU, as it showed better results and is also used more in general in datasets like ImageNet. Additionally, in the original YOLOv7-tiny architecture, the neck and head components were closely integrated. We have explicitly separated these components to enable independent tuning and evaluation. Furthermore, we reformulated YOLOv7’s ELAN blocks as, what we call, ConcatBlocks. Although the module remains functionally equivalent to ELAN, this reformulation provides a clearer and more explicit representation of its implementation within the architecture. Refer to figure 8b for the detailed structure of the ConcatBlock.

Each ConcatBlock uses 1×1 convolutions to adjust the dimensions of feature maps, followed by concatenation to maintain information flow. When compared to figure 7, this approach aligns with the cross-stage partial (CSP) connection concept.

By stacking multiple ConcatBlocks and incorporating max pooling, we can construct the backbone. This backbone is versatile and can also be applied to other domains, such as ImageNet classification. The structure is illustrated

in figure 8a. Each convolution is combined with a batch normalization and a ReLU activation.

The backbone generates four feature maps (f_1, f_2, f_3, f_4), of which we utilize only the latter three. While f_1 could theoretically improve detection of extremely small objects, our experiments showed no significant performance benefit from its inclusion. The combination of f_2, f_3 , and f_4 provides sufficient coverage for all relevant bounding box scales in our application.

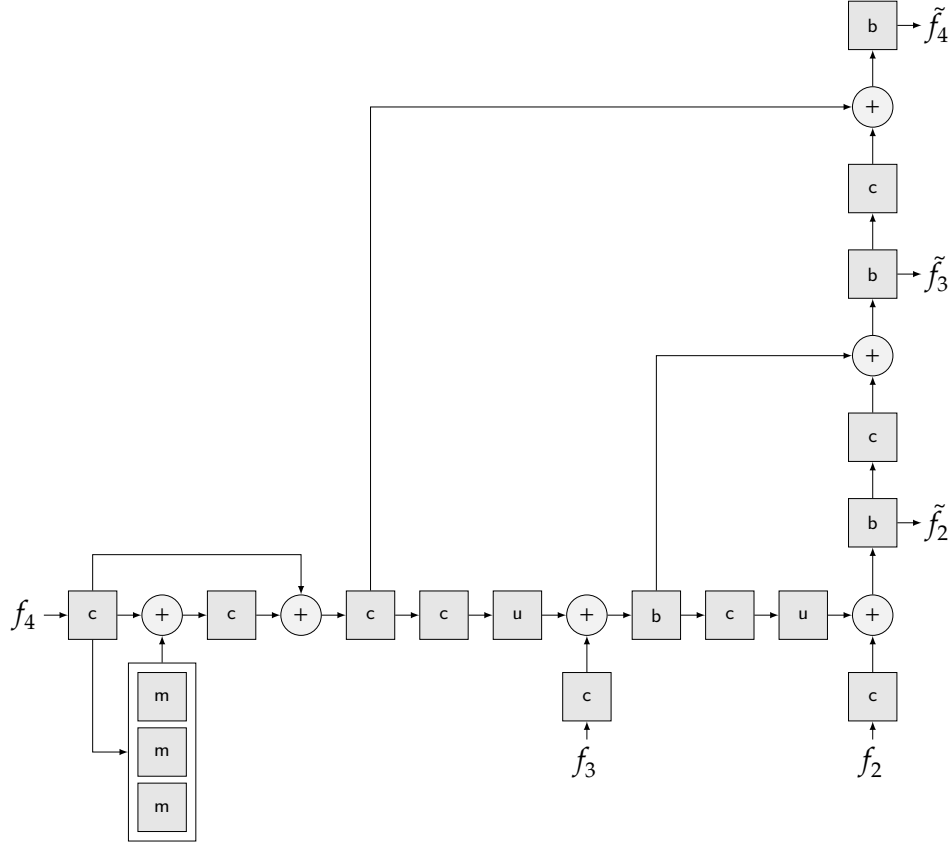


Figure 9: YOLOv7-tiny's neck architecture based on PANet. The three inputs f_2, f_3, f_4 from the backbone are taken as input, and then refined to $\tilde{f}_2, \tilde{f}_3, \tilde{f}_4$. "c" = Convolution with BN and ReLU, "+" = Concatenation, "m" = MaxPooling, "u" = Upsampling, "b" = ConcatBlock. Adapted from [Nie+24c].

The neck architecture adopts a bidirectional path aggregation strategy, characterized by a unique L-shaped structure, as depicted in figure 9. It processes three input feature maps (f_2, f_3 , and f_4) and produces refined outputs (\tilde{f}_2, \tilde{f}_3 , and \tilde{f}_4), with spatial resolutions decreasing from f_2 to f_4 . This refinement leverages two interconnected paths: a horizontal path for upsampling (moving from f_4 to f_2) and a vertical path for downsampling (moving from \tilde{f}_2 to \tilde{f}_4).

The horizontal path initiates the refinement process, starting with a feature enhancement block applied to f_4 , the smallest input feature map. This block consists of three parallel max pooling layers ("m" blocks) with different kernel sizes and carefully adjusted padding to maintain the same spatial dimensions. This design ensures effective feature extraction from f_4 , which, despite being the smallest feature map, is quite large. For instance, when processing an input image of size 2048×2048 , the feature maps are sized 256×256 (f_2), 128×128 (f_3), and 64×64 (f_4). The extracted features are then processed through a sequence of convolution blocks ("c" blocks) incorporating convolution, batch

normalization, and ReLU activation. These blocks, combined with nearest-neighbor upsampling layers (“u” blocks), progressively refine and integrate features across increasing spatial resolutions.

In contrast, the vertical path follows a downsampling approach to complement the horizontal path. This path employs “c” blocks with varying stride sizes to systematically reduce spatial dimensions, aggregating multi-scale features. Each output feature map is further refined by a ConcatBlock (“b” block), which enhances its feature representations by fusing information from multiple layers.

YOLOv7’s neck architecture draws inspiration from PANet, enhancing the original design by combining the FPN’s top-down path with an additional bottom-up path. The integration of these paths is facilitated through strategic concatenation operations (“+”), allowing feature fusion across scales. As a result, the refined feature maps has both detailed spatial information and high-level semantic context.

Finally, this information is used in the form of refined feature maps ($\tilde{f}_2, \tilde{f}_3, \tilde{f}_4$) in the subsequent detection head. It follows a simple design pattern and processes each refined feature map independently. For each input, a convolution block with batch normalization and ReLU activation is first applied to expand the channel dimensionality, followed by a 1×1 convolution that produces the 5 final detection outputs: $\hat{x}, \hat{y}, \hat{w}, \hat{h}, \hat{o}$. This is the standard structure of a coupled head.

Metric

The predominant metric for evaluating object detection performance is mean Average Precision (mAP), which provides a single score that captures both precision and recall across different confidence thresholds. The Average Precision (AP) for a single class is defined as the integral of the precision-recall curve [Eve+10]:

$$AP = \int_0^1 p(r) dr$$

where r represents recall and $p(r)$ is the precision at that recall level. In practice, this integral is approximated using eleven equally spaced recall points to create a discrete summation. The computation of these metrics relies on defining what constitutes a “correct” detection using the Intersection over Union (IoU) metric, which measures the overlap between predicted and ground truth bounding boxes. A detection is considered correct (a true positive) only if the IoU exceeds a predetermined threshold, traditionally set at 0.5.

While mAP has become the de facto standard in object detection evaluation, our initial experiments revealed several limitations for vessel detection in microscope images. In our application, minor variations in bounding box placement do not significantly impact the utility of the detection. The standard threshold of 0.5 proved too stringent, particularly given the errors in ground truth annotations.

Furthermore, mAP treats precision and recall with equal importance, which doesn’t align with our application requirements. In vessel detection, achieving high recall is more important, as missing vessels (false negatives) is more problematic than detecting spurious objects (false positives). This is particularly relevant since false positives can be filtered out in subsequent classification steps, while missed vessels cannot be recovered. We also observed cases where

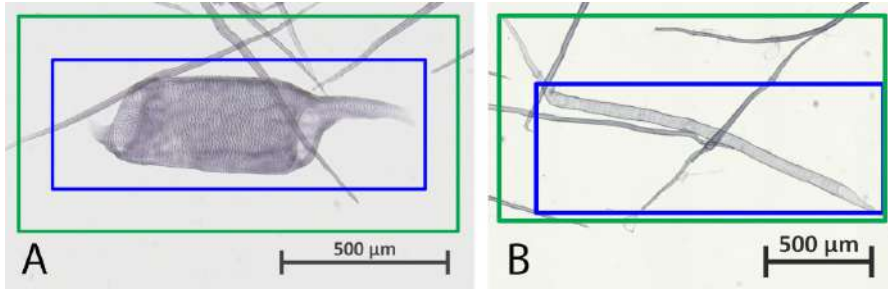


Figure 10: Comparison of predicted bounding boxes (blue) and ground truth boxes (green). A high IoU threshold can result in both predicted boxes being rated as errors. (A) The overlap is below 0.5. Due to incorrect annotations, the predicted bounding boxes are sometimes more accurate. (B) Imperfect prediction as the end of the object (vessel element) is not detected. From: [Nie+24c].

predicted bounding boxes were actually more accurate than the ground truth annotations, yet the rigid evaluation scheme of mAP would penalize such cases despite their practical utility. Figure 10 shows an example of two cases where an IOU threshold of 0.5 would lead to harsh evaluation results.

After using mAP in the baseline YOLOv7 model, we transitioned with our WoodYOLO model to the F2 score as our primary evaluation metric. The F2 score belongs to the family of F-beta measures, which provide a weighted harmonic mean of precision and recall. The general formula for F-beta score is:

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

where β represents the relative importance of recall to precision. For our specific case, we use $\beta = 2$ which means that recall is weighted more than precision. This decision aligns with our objective to prioritize the detection of vessels (minimizing missed detections) over avoiding false positives.

The precision and recall values are computed using:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where TP, FP, and FN represent true positives, false positives, and false negatives respectively, determined using an IoU threshold of 0.3.

This modified evaluation approach better serves our specific use case, where the primary goal is to identify all vessel elements while accepting that false positives can be handled through subsequent processing steps. The lower IoU threshold of 0.3 acknowledges that perfect alignment with ground truth boxes is not essential for our application, and partial overlap is often sufficient for practical purposes. The more lenient threshold also makes the metric more robust to variations in annotation style and quality, providing a more realistic assessment of model performance.

3.3 VESSEL CLASSIFICATION

After identifying the vessel elements in our images, the next step is to classify them into their respective genera. Compared to the complexities of object detection, this task is more straightforward since the input consists of an image patch and a classification label. As a result, specialized loss functions or models

are unnecessary, and we can rely on standard neural network architectures for training.

3.3.1 Image Preprocessing

A primary challenge in vessel classification is the significant variation in image sizes, as detailed in Table 1. Neural networks require inputs of uniform dimensions within a batch. Although it is technically possible to address size variability by using a batch size of 1, this approach is impractical due to the resulting slow training speeds and unstable gradient updates. With over 100,000 images in our dataset, efficient training necessitates larger batch sizes. Another reason is the use of batch normalization layers in most architectures (section 2.1.3).

To standardize image inputs for the network, we propose several preprocessing approaches:

- **Direct Resizing:** Resizing all images to a fixed target size without maintaining aspect ratios. While this method is computationally simple and widely used (e.g., in ImageNet), it risks introducing distortions that may compromise features critical for genus classification.
- **Resizing with Padding:** Images are resized to match a target dimension (e.g., 800 pixels on the longer side) while preserving their aspect ratios. The shorter side is then padded with zeros to create a square image. For images that are already smaller than the target dimension, no resizing is performed, just padding. This method avoids distortions and ensures that structural integrity is maintained.
- **Padding Only:** Images are padded to the size of the largest image in the batch. This approach retains original size-based features, which may provide additional classification cues, but significantly increases memory consumption due to the inclusion of empty pixels.

Another consideration is the color properties of the images. Vessel elements are stained with chemical solutions that introduce color variations. To address this, we also evaluated the impact of converting images to grayscale. Grayscale conversion reduces data dimensionality and eliminates potential biases related to staining colors, ensuring the network focuses on structural rather than chromatic features. Notably, color is not used by experts for genus classification.

Many images include multiple focal planes, providing a quasi-3D perspective of the vessel elements. We explored two strategies for integrating this information into the model:

- **Channel Stacking:** Treating focal planes as separate input channels, similar to RGB color channels, allows the network to process all planes simultaneously and leverage spatial relationships between them.
- **Independent Processing:** Treating each focal plane as an independent image and aggregating predictions across planes, for example, by averaging the classification probabilities. This approach is simpler and may reduce computational requirements.

Due to the limited number of 5 focal planes, employing a 3D convolutional neural network (CNN) is not advantageous. A kernel of size $3 \times 3 \times 3$ would rapidly traverse the depth axis, offering little benefit for feature extraction.

It is important to note that preprocessing has different implications for object detection and classification tasks. In object detection, the primary goal is to locate vessels, which can tolerate some distortions or variations in color without significantly impacting performance. For classification, however, preserving the fine structural details of the vessel elements is critical to distinguish between genera.

3.3.2 *Choice of architectures*

Recent research has highlighted important considerations that inform our architecture selection strategy. While ImageNet performance has traditionally guided architecture choices, multiple studies demonstrate that ImageNet accuracy does not reliably predict performance on real-world tasks [FKS23; Nay+22]. This finding is particularly relevant for our work, as we aim to solve a practical classification problem rather than compete on standard benchmarks.

The relationship between model architecture and real-world performance requires careful consideration. Vision Transformers (ViTs) [Dos+21] have emerged as a powerful tool for visual recognition, generating significant interest in the research community. However, recent comprehensive studies have revealed a more nuanced picture. [Bai+21] demonstrated that when compared under fair experimental conditions with similar model capacities, well-designed CNNs can match Transformers in adversarial robustness when properly implementing Transformer training techniques. [Gol+23] further supports this finding through extensive experiments across diverse computer vision tasks, showing that well-designed convolutional neural networks pre-trained through supervised learning often demonstrate superior performance. More recent work by [VSL24] provides additional insights, revealing that supervised ConvNeXt architectures not only compete well with CLIP-based models [Rad+21] in transferability but also demonstrate superior performance on ImageNet robustness benchmarks and better calibration compared to ViT counterparts.

Given these insights, we adopt an empirical approach to architecture selection that prioritizes proven architectures with different strengths. Our primary choice is ConvNeXt, which has demonstrated exceptional performance across various tasks [Gol+23] and shows superior performance on synthetic data while maintaining strong transferability comparable to CLIP models [VSL24]. We also include DenseNet-121 [HLW16], whose dense connectivity pattern through direct connections from any layer to all subsequent layers offers potential advantages in feature reuse and information flow. The classical ResNet-34 [He+15a] architecture is included as it represents a fundamental approach to deep learning through skip connections, which has influenced numerous subsequent designs. Finally, we evaluate EfficientNet [TL19], which showed strong performance in the Battle of the Backbones [Gol+23] study while maintaining computational efficiency through its compound scaling approach.

While the timm repository [Wig19] currently contains 1,446 architecture variants, exhaustively evaluating all options would not be practical or necessarily beneficial. This is particularly relevant given recent findings by [VSL24] showing that models with similar ImageNet accuracies can exhibit substantially different behaviors in terms of error types, calibration, transferability, and feature invariance. Furthermore, as demonstrated by [FKS23], interventions such as data augmentation can often yield more substantial improvements than marginal architectural refinements. Therefore, our selection strategy focuses on these representative architectures that span different design philosophies

and have demonstrated robust performance across various evaluation metrics beyond simple ImageNet accuracy.

3.3.3 Metrics

Selecting the right metric to evaluate classification performance is crucial, particularly in the context of imbalanced datasets like ours. While accuracy is a commonly used metric, it is insufficient for this task due to its bias toward classes with a larger number of samples. For example, in our dataset, the genus *Hevea* typically has only a few vessel elements per image, whereas *Populus* and *Fagus* often contain hundreds. A metric focused solely on accuracy would undervalue the performance on *Hevea*, as misclassifications in this minority class would have a negligible impact on the overall score.

To address this challenge, we employ macro-averaged F1 score (Macro F1) as our primary evaluation metric. Macro F1 averages the F1 scores across all classes, giving equal importance to each class regardless of its sample size. This metric is particularly effective for highlighting the classifier’s performance on rare or minority classes, making it well-suited for our imbalanced dataset. It is calculated as:

$$\text{Macro F1} = \frac{1}{n} \sum_{i=1}^n \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i},$$

where n is the number of classes, and Precision_i and Recall_i are calculated for each class i . Notably, this is the same metric we use for object detection, albeit with a key difference: in classification, we set $\beta = 1$ (equal weight to precision and recall), whereas in object detection, such as in our WoodYOLO model, we use $\beta = 2$ to prioritize recall. Another metric related to Macro F1 is Micro F1, which treats the entire dataset as a single pool, summing true positives, false positives, and false negatives across all classes before computing the F1 score. While this approach simplifies evaluation, it heavily favors majority classes, as shown by [OB21], leading to misleadingly high scores for imbalanced datasets. Therefore, we use Macro F1 for our application.

Additionally, we incorporate the confusion matrix to gain a more detailed understanding of model performance. The confusion matrix provides granular insights into the classifier’s behavior, showing not only the true positives (diagonal elements) but also the types of misclassifications (off-diagonal elements). This enables us to identify specific classes that are frequently confused, offering opportunities for targeted improvements in the model’s architecture or training process.

3.4 EVALUATION

After developing our system, which includes a detector to identify vessels and a classifier to determine their genera, we proceed to evaluate its performance. We begin by analyzing the baseline model, utilizing YOLOv7 and the selected architectures for classification. Next, we assess our newly developed WoodYOLO to quantify its improvements.

For evaluation, we use a subset of 321 images containing 27,842 vessels. Although smaller than the complete dataset presented in figure 3, this subset was available at the time of the initial study. Despite this limitation, the dataset remains robust, as it includes nearly 30,000 vessels, allowing for statistically significant results.

We employ a cross-validation strategy, as previously discussed. The reported values represent averaged results across folds. Additionally, we use an independent test dataset comprising unseen images. Cross-validation results guide parameter tuning, while the test dataset provides a final assessment of model performance.

3.4.1 Baseline

Detection

Our initial experiments with YOLOv7 models revealed that larger models did not necessarily yield better performance. While the larger model sizes ranged from 70.4 million parameters (W6) to 151.7 million parameters (E6E), increasing model capacity not only failed to improve detection performance but also introduced training instability and memory constraints on consumer-grade GPUs. Since the vessels in the microscope images can be identified primarily by their shape and other low-level features, large feature extractors are not necessary for effective detection. However, if our detector were to include a classification step, the requirements might differ.

For the remainder of the experiments, we used the YOLOv7-W6 model. This model utilizes four feature maps and was pretrained on a resolution of 1280. Compared to other models pretrained at the same resolution, it had lower GPU requirements and could be more easily scaled to higher resolutions.

Image resolution emerged as a more critical factor for detection performance, as illustrated in figure 11. We use the standard mAP metric for detection in our baseline model. Increasing the image resolution from 2560 to 6400 pixels resulted in a 7% improvement in mAP, primarily due to enhanced recall in identifying vessel elements.

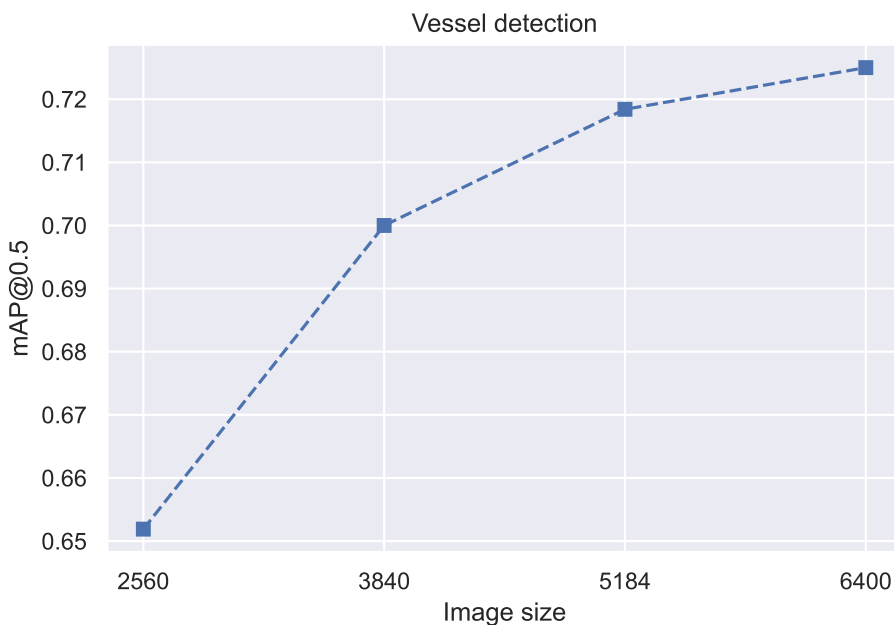


Figure 11: Impact of image size on mean average precision (mAP) for the validation dataset (cross validation). Each image is resized to a square, with the "image size" representing the height/width in pixels. For example, at a resolution of 2560×2560, the mAP reaches 0.65. Source: [Nie+24a].

Beyond a resolution of 5184 pixels, further increases yielded diminishing returns while significantly increasing computational costs. Both training and inference times slowed with larger images.

Other hyperparameters, such as learning rate, number of epochs, and gradient accumulation, resulted in only minor variations in mAP (less than 0.5%).

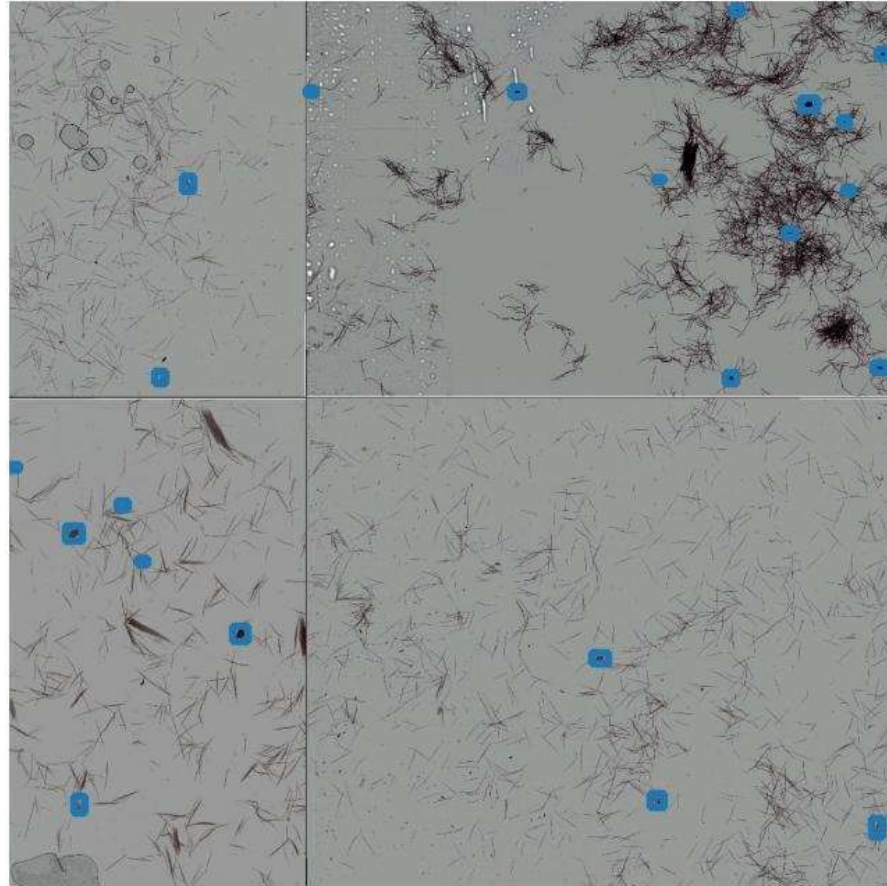


Figure 12: Mosaic data augmentation, where the blue boxes denote vessel elements. Source: [Nie+24a].

Our experiments revealed that data augmentation techniques significantly improved mean Average Precision (mAP). In particular, mosaic augmentation provided substantial performance gains. Figure 12 demonstrates how mosaic augmentation works in our dataset, where four training images are combined into a single training instance. Together with mosaic augmentation, we implemented several additional techniques: color jittering in HSV (Hue, Saturation, Value) color space, geometric transformations including image translation and scaling, and horizontal flips. For a broader discussion of data augmentation techniques and their theoretical foundations, see section 2.2.5.

After determining optimal hyperparameters, we tested the trained model on the test dataset, achieving stable results across both validation and test datasets. The test dataset mAP was 71.85%, with 77.63% precision and 72.98% recall.

Detection performance varied significantly across genera, as shown in table 5. For instance, *Hevea* exhibited low precision (50.60%) but high recall (90.37%), while genera like *Liquidambar* and *Salix* showed the opposite trend. These

Genus	Precision	Recall	F2
Liquidambar	0.8885	0.6145	0.6549
Salix	0.9109	0.6317	0.6730
Fagus	0.9357	0.6799	0.7192
Populus	0.9578	0.6855	0.7268
Eucalyptus	0.8125	0.7629	0.7723
Hevea	0.5060	0.9037	0.7809
Schima	0.8736	0.8537	0.8576
Betula	0.8961	0.8581	0.8654
Acacia	0.8753	0.8950	0.8910

Table 5: Detection performance for individual genera, ranked by F2 score on the test dataset.

differences can be attributed to the size of the vessels and their frequency in the images.

Precision can be improved by increasing the confidence threshold, while recall benefits from lowering it. Our analysis identified three primary error sources:

- Dense vessel packing in genera like *Fagus* and *Liquidambar* created detection challenges, particularly in areas where vessels overlap or cluster tightly together
- Images with low brightness resulted in reduced recall, as the model struggled to distinguish vessel boundaries in poorly illuminated regions
- False positives occurred when fibers morphologically resembled vessel elements, leading to incorrect detections of non-vessel structures

In real-world wood identification applications, achieving perfect detection of individual vessel elements is not critical. The key requirement is ensuring sufficient detection accuracy across the vessel population to enable reliable genus-level identification. As long as the overall distribution of detected vessel elements approximates the true distribution, minor errors in individual detections or classifications remain acceptable for practical purposes.

Classification

We extracted patches from all microscope images using the ground truth bounding boxes and first evaluated different preprocessing strategies.

Of the three preprocessing approaches introduced earlier, resizing with padding proved most effective. The padding-only approach, while theoretically preserving all original features, proved impractical due to excessive memory requirements – with our largest vessels exceeding 3000 pixels, batch training became impossible even on 40GB GPUs. Direct resizing, though computationally efficient, introduced significant distortions in vessel morphology, leading to a 1.2% decrease in macro F1 score compared to our resize-with-padding approach.

After establishing resize-with-padding as our method, we investigated optimal target resolutions. As shown in figure 13, increasing resolution from 224×224 to 800×800 pixels improved the macro F1 score by approximately 12%. Interestingly, further increases showed diminishing returns while substantially increasing computational costs. We hypothesize this plateau occurs because

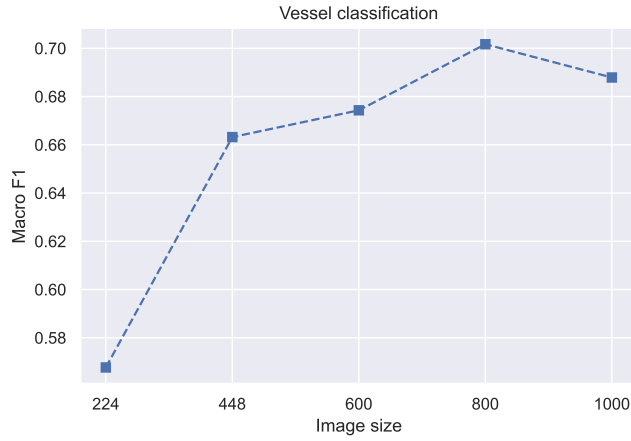


Figure 13: Effect of image resolution on the macro F1 score (cross validation). The “image size” refers to the square height/width in pixels, with a range from 224×224 to 800×800 pixels. A resolution of 800×800 pixels provides the best balance between performance and computational cost. Beyond this, further resolution increases yield diminishing returns. Source: [Nie+24a].

most vessel features are captured at 800×800 resolution, with additional pixels primarily padding empty space. Recall from table 1 that, on average, most vessels are smaller than 1000×1000.

Following our preprocessing strategy analysis, we evaluated various approaches to improve accuracy:

- **Grayscale Conversion:** As anticipated, eliminating color information improved performance by 1%, confirming our hypothesis that staining variations potentially introduce noise rather than useful features. While grayscale conversion does not reduce GPU memory usage, it decreases the file size of the processed images.
- **Focal Plane Integration:** Among tested configurations in table 6, individual plane processing with prediction averaging achieved optimal results.

The superior performance of grayscale processing suggests that structural features, rather than staining characteristics, are the primary drivers of classification accuracy. This aligns with expert wood anatomist approaches, where color is rarely used as a diagnostic feature.

Focal plane	Macro F1
1st, 2nd, 3rd	0.6669
1st, 3rd, 4th	0.6615
3rd	0.6602
Average	0.7017
Maximum	0.7014

Table 6: Results of focal plane configurations for vessel classification (cross validation). Averaging predictions yields the highest macro F1 score, while concatenating focal planes as RGB images (123, 134) or using a single focal plane (3) did not give good results.

Our focal plane experiments revealed that while concatenating multiple planes performed similarly to using only plane 3, averaging predictions across

all planes provided the most robust results. In our approach, different focal planes serve a similar purpose to test-time augmentation (TTA) [Mos+20]: each plane provides a unique view of the vessel, capturing complementary spatial and structural details. By averaging predictions from these distinct views, we effectively aggregate diverse information, enhancing classification reliability and robustness.

Architecture	Macro F1
ConvNeXt-tiny	0.7017
DenseNet-121	0.6441
ResNet-34	0.5958
EfficientNet-Bo	0.6472
EfficientNet-B1	0.6698
EfficientNet-B2	0.6632

Table 7: Architecture experiments (cross validation). The table shows that ConvNeXt-tiny achieved the highest Macro F1 score.

As anticipated from our architecture discussion, ConvNeXt-tiny achieved the highest performance. Table 7 shows the results for cross validation.

Similarly, as discussed in the Battle of the Backbones study [Gol+23], EfficientNet has also proven to be a strong choice.

The final model achieved a macro F1 score of 64.61% on the test dataset (slightly lower than the cross validation results). Analysis of the confusion matrix (figure 14) revealed expected challenges with morphologically similar genera:

- *Liquidambar-Schima-Fagus* group had substantial cross-confusion, with up to 23% misclassification between pairs
- *Populus-Salix* showed consistent mutual confusion patterns

These confusion patterns mirror expert classification challenges, particularly for individual vessels lacking distinctive features. While most genera achieved accuracy exceeding 80%, the challenging groups showed lower performance. By combining frequently confused genera into unified classes, we achieved over 90% accuracy, though at the cost of reduced taxonomic resolution. This trade-off between accuracy and taxonomic precision represents a key consideration for practical applications.

3.4.2 WoodYOLO

Building upon the baseline results, we developed WoodYOLO to improve the detection accuracy achieved with YOLOv7. In all experiments, we use the F2 score as the primary evaluation metric, instead of the traditional mean Average Precision (mAP), to better align with the practical requirements of vessel detection.

We begin by evaluating the final object detection accuracy. As shown in table 8, our baseline YOLOv7-W6 achieved an F2 score of 0.783. However, by switching to WoodYOLO, we observe a 6% improvement, with an F2 score of 0.848. Interestingly, newly developed models like YOLOv10 did not surpass YOLOv7 in performance. One potential reason for this is YOLOv10’s decision to eliminate non-maximum suppression, which might have negatively impacted its ability to handle dense and overlapping objects.

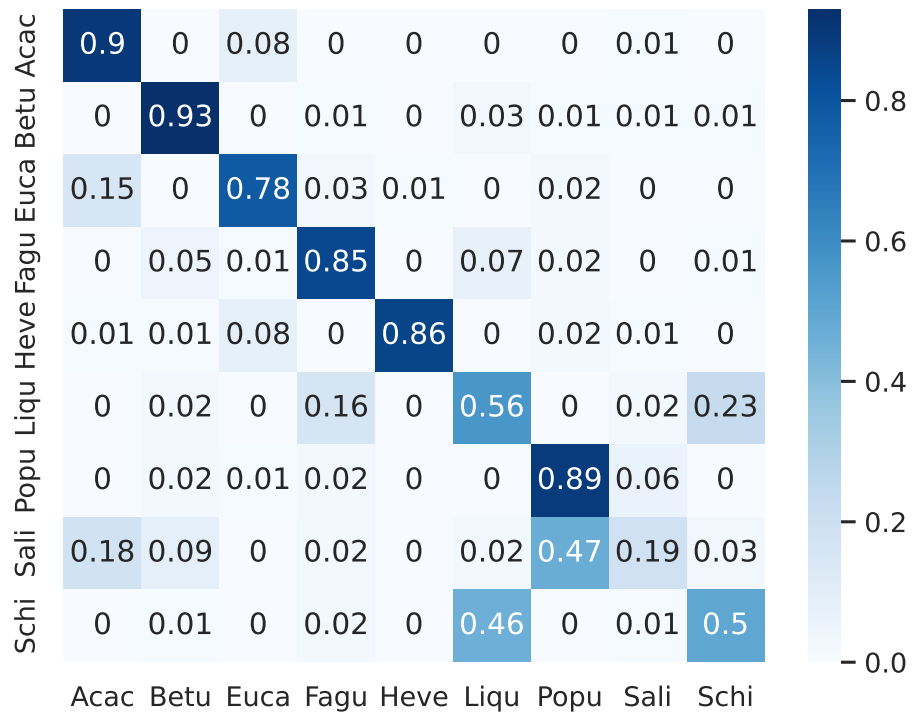


Figure 14: Confusion matrix showing the predicted versus actual classes (test dataset). The model struggles with distinguishing between the “*Schima*” and “*Liquidambar*” genera, misclassifying them in 23% of cases. Similar challenges arise for other morphologically similar genera like *Populus* and *Salix*. This reflects typical difficulties encountered by human experts in wood anatomy. Source: [Nie+24a].

Another notable aspect of WoodYOLO is its ability to operate at a lower image resolution, reducing VRAM requirements. While YOLOv7-W6 required a resolution of 5184x5184 and an A100 GPU for training, WoodYOLO operates at 2048x2048, requiring only around 10GB of RAM for training. Therefore, we could train on higher batch sizes to further improve training stability. We also tested increasing the resolution to see if it improved performance. Interestingly, the results plateaued at 2048x2048, where we saw a 5% improvement compared to 1024x1024. However, increasing the resolution further to 4096x4096 resulted in a slight performance decline, suggesting that higher resolutions are not necessary for this task, which is beneficial for computational efficiency.

The choice of the backbone network was crucial to achieving high performance. After evaluating various architectures, we found VGG11-bn to be the optimal choice. This backbone strikes a balance between model complexity, parameter count, and memory usage. We discussed VGG11 in more detail in section 2.2.4. The simplicity of VGG11-bn, particularly the absence of complex components like skip connections or Squeeze-and-Excitation blocks, complements the PANet neck, which already incorporates complex features. This design choice suggests that complex components in the backbone may not be necessary, as the PANet neck architecture is sufficient for combining features effectively.

Our optimization experiments, detailed in table 9, revealed several interesting insights. Contrary to conventional object detection practices, increasing the number of neighboring grid cells during training actually reduced performance. Additionally, switching from the baseline GIoU to CIoU caused a

Architecture	F2 Score
Ours	0.848
YOLOv7-W6	0.783
YOLOv7-tiny	0.723
YOLOv10-M	0.719
YOLOv10-S	0.691

Table 8: Comparison of model architectures based on F2 score.

Component	Modification	F2 Score Change (%)
Architecture	VGG11-bn (Baseline)	0.00
	ConvNeXt-Nano	-0.38
	EfficientNet-Bo	-1.42
	RepVGG-A0	-1.78
	YOLOv7-tiny	-2.04
	ResNet-18	-2.65
Training	Standard Training (Baseline)	0.00
	CIoU	-0.24
	No Maximum Size Constraint	-0.83
	Gradient Accumulation	-1.18
	2 neighbors	-1.70
	Mosaic Augmentation	-7.31

Table 9: Ablation study: Impact of architectural and training modifications on F2 score.

slight decrease in performance. Overall, we found that the choice of IoU loss function had a marginal impact, with less than a 1% change in performance. In the network’s outputs, we define two parameters, m_w and m_h , which set the maximum allowed dimensions for the bounding boxes. Removing this constraint resulted in a slight decrease of 0.83% in the F2 score.

In our experiments, mosaic augmentation showed varying effectiveness despite its widespread recognition in improving object detection performance. On general real-world datasets, both the standard YOLOv7 model (pretrained on COCO) and our WoodYOLO model achieved small gains with this technique. Yet when applied to our specialized wood identification dataset, WoodYOLO’s performance decreased substantially by 7-8%. Since the same augmentation technique proved beneficial in other contexts, this decline cannot be attributed to implementation issues. Instead, it suggests that specialized domains like wood microscopy may benefit more from domain-specific optimizations than general-purpose augmentation techniques.

WoodYOLO’s superior performance can be attributed to several key design decisions. We specialized the architecture specifically for vessel localization, eliminating unnecessary complexity while preserving essential features. Rather than optimizing for the traditional mAP metric, we focused on the F2 score, which better aligns with the practical requirements of vessel detection. The choice of a streamlined backbone network enabled more stable training and improved scaling characteristics.

While models like YOLOv7 and YOLOv10 excel across diverse applications, our results highlight the value of domain-specialized architectures. Through careful optimization of both architecture and training objectives, we achieved better recall and precision than the base YOLOv7 model.

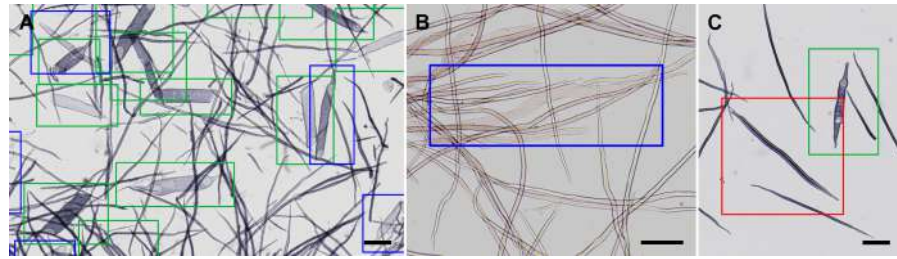


Figure 15: Representative error cases in vessel detection. (A) Dense vessel arrangement in *Fagus*: Successfully detected vessels (green boxes) alongside missed detections (blue boxes) in areas of high vessel density. (B) Image quality effects in *Liquidambar*: Reduced contrast and brightness leading to missed detections (blue boxes). (C) Feature similarity in *Eucalyptus*: False positive detections (red boxes) caused by cohesive fibers exhibiting vessel-like characteristics. Scale bars = 200 μm . Source: [Nie+24a].

Despite these improvements, achieving perfect detection remains challenging, as evidenced by three primary error categories (figure 15). First, in genera such as *Fagus* and *Liquidambar*, high vessel density leads to detection failures, particularly when vessels are closely spaced (figure 15A). Second, suboptimal image characteristics, notably low brightness and contrast, significantly impact detection reliability (figure 15B). Third, structural similarities between fibers and vessel elements result in false positive detections (figure 15C).

To address these limitations, particularly in cases of densely packed or overlapping vessels, we propose that incorporating rotated bounding boxes could offer a promising direction for future improvements. This approach would better accommodate the natural variation in vessel orientations and potentially reduce detection failures in complex arrangements. Furthermore, stronger data augmentation could prove beneficial as well.

Having developed a wood identification system, the next logical step is to focus on explaining its decision-making process. Here, the emphasis is on the classifier rather than the detector. While the definition of what constitutes a vessel is relatively clear, minor ambiguities may arise, such as whether a slightly damaged vessel still qualifies as one. However, these distinctions are generally intuitive. The greater challenge lies in distinguishing between genera, a task handled by the classifier.

Wood anatomists base their analyses on various features, including the 163 structural attributes defined by the International Association of Wood Anatomists (IAWA) [Com89]. In the long term, it would be valuable to identify novel features or determine which features are most significant for neural networks compared to those used by human experts. As a step toward this goal, we identify interpretability techniques for convolutional neural networks, evaluate their effectiveness, and explore opportunities for improvement.

This research was carried out in three phases:

1. Comparison of Methods: Conducted a comprehensive evaluation of various interpretability techniques and showed shortcomings of current metrics [Nie+24b].
2. Evaluation Metric: Introduced a metric for assessing the quality of interpretability methods [NSK24a].
3. Method: Constrain neural networks to improve interpretability [NSK24b].

This chapter synthesizes findings from the first two phases. The next chapter introduces the architectural perspective.

4.1 ATTRIBUTION METHODS

4.1.1 *Fundamentals and Terminology*

As discussed in section 2.4, model interpretability encompasses both transparency and post-hoc interpretability. Convolutional neural networks (CNNs), despite operating directly on raw image data without requiring sophisticated feature engineering, lack transparency due to their complex architectures involving millions of parameters, multiple processing layers, and non-linear transformations. To understand these non-transparent models, we must rely on post-hoc interpretation techniques. Attribution methods, which we discuss in this section, represent one such class of post-hoc interpretation approaches. They provide intuitive explanations by mapping the network’s decision-making process back to the original image space, highlighting which regions were most influential for the model’s prediction.

For a neural network $f : \mathbb{R}^d \rightarrow \mathbb{R}^C$ mapping an input $x \in \mathbb{R}^d$ to logits for C classes, an attribution method generates a map $A : \mathbb{R}^d \rightarrow \mathbb{R}^d$ assigning importance scores to input features. These scores indicate each feature’s contribution to the model’s decision for a target class c .

4.1.2 Basic Gradient Methods

Feature attribution through gradient analysis measures how changes in input features affect a model's output predictions. For a target class c and neural network f , the gradient attribution can be computed as [SVZ14]:

$$A_{grad}(x) = \left| \frac{\partial f^{(c)}(x)}{\partial x} \right|$$

This method, called *Gradients* or *Vanilla Gradients*, uses the absolute value of gradients to capture the magnitude of feature influence. However, alternative formulations are possible: using the raw gradients without absolute values reveals whether features positively or negatively influence the prediction, while applying a ReLU activation highlights only positively contributing features. In linear models where $f(x) = w^T x$, gradients directly correspond to feature importance through model weights.

However, deep neural networks introduce additional complexities. A significant challenge arises from the chain rule multiplication across multiple layers, which often results in diminishing gradient magnitudes [PMB13] (refer to section 2.1.3 and equation (5)). This phenomenon can obscure the distinction between meaningful feature contributions and numerical noise in attribution maps. To address this limitation, researchers have developed enhanced backpropagation techniques [ZF13; Spr+15]. In standard backpropagation, the ReLU gradient simply passes through positive inputs while blocking negative ones. Guided Backpropagation extends this approach by implementing a more stringent gradient flow criterion [Spr+15]. Given the loss function L , the gradient flow is defined as:

$$\frac{\partial ReLU(x)}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \text{ and } \frac{\partial L}{\partial ReLU(x)} > 0 \\ 0 & \text{otherwise} \end{cases}$$

By allowing gradient flow only when both input and incoming gradient are positive, this modification effectively filters out noisy or contradictory signals. While this approach produces clearer visualizations, it does deviate from exact gradient computation.

4.1.3 Path Integration Methods

Path integration methods address the diminishing gradient problem by accumulating gradients along a path from a reference point to the input. This approach helps capture feature importance even when individual gradients at any single point are small. Integrated Gradients (IG) [STY17] exemplifies this approach by defining a straight-line path from a baseline image x' to the input x :

$$IG_i(x) = (x_i - x'_i) \int_{\alpha=0}^1 \frac{\partial f^{(c)}(\gamma(\alpha))}{\partial \gamma_i(\alpha)} d\alpha$$

where $\gamma(\alpha) = x' + \alpha(x - x')$ parameterizes the path. The subscript i denotes a specific feature index in the input. The baseline image x' is typically chosen as a zero (black) image, though this choice can be application-dependent. In practice, this integral is approximated using a Riemann sum with m steps:

$$IG_i(x) \approx (x_i - x'_i) \sum_{k=1}^m \frac{\partial f^{(c)}(x' + \frac{k}{m}(x - x'))}{\partial x_i} \cdot \frac{1}{m}$$

The choice of step count m significantly impacts attribution quality, with higher values providing more accurate approximations at the cost of increased computation time. Recent variants like Blur Integrated Gradients [XVS20] use a blurred version of the input as the baseline, while Guided Integrated Gradients [Kap+21] employs an adaptive path based on model behavior.

4.1.4 Class Activation Methods

Class Activation Mapping (CAM) methods build upon gradient-based approaches by focusing on feature maps in the final convolutional layers. Rather than computing gradients through the entire network, these methods analyze how the final convolutional layer’s activations contribute to class predictions. This design leverages the observation that later convolutional layers typically capture high-level semantic features.

The original CAM method [Zho+15a] requires a specific network architecture where global average pooling is performed on convolutional feature maps before the final classification layer. For a target class c , CAM computes attributions as:

$$A_{CAM}^{(c)}(x) = \sum_k w_k^{(c)} F_k$$

where $w_k^{(c)}$ represents the weight connecting the k -th feature map to class c in the final layer, and F_k are the feature maps of some chosen layer.

GradCAM [Sel+19] generalizes this approach by removing the architectural constraints of CAM. Instead of using classifier weights directly, GradCAM derives importance weights through gradients. For a selected layer, the attribution is:

$$A_{GradCAM}^{(c)}(x) = ReLU\left(\sum_k \alpha_k^{(c)} F_k\right)$$

The importance weights $\alpha_k^{(c)}$ are calculated using gradients:

$$\alpha_k^{(c)} = \frac{1}{Z} \sum_{i=1}^u \sum_{j=1}^v \frac{\partial f^{(c)}}{\partial F_k(i, j)}$$

where $Z = u \times v$ represents the spatial dimensions. The resulting attribution map is upsampled to match input dimensions using bilinear interpolation.

GradCAM++ [Cha+18] refines this approach by incorporating second-order derivatives in the weight calculation.

$$A_{GradCAM++}^{(c)}(x) = \sum_k a_k^{(c)} F_k$$

For a target class c , the coefficient $a_k^{(c)}$ is defined as:

$$a_k^{(c)} = \sum_{i=1}^u \sum_{j=1}^v v_k^{(c)}(i, j) \cdot ReLU\left(\frac{\partial f^{(c)}}{\partial F_k(i, j)}\right)$$

The weight $v_k^{(c)}(i, j)$ incorporates second-order gradients to better capture the relative importance of feature map activations. This modification helps GradCAM++ produce more precise attribution maps, particularly for cases involving multiple instances of the same class in an image.

The success of class activation maps has inspired numerous variants in recent years. These include SS-CAM [Wan+20a], IS-CAM [Nai+20], Ablation-CAM [DR20], FD-CAM [Li+22], Group-CAM [ZRY21], Poly-CAM [ECD22], Zoom-CAM [Shi+20], Recipro-CAM [BL23], and EigenCAM [MY20]. Other notable approaches include ScoreCAM [Wan+20b], which eliminates gradient dependence, and Smooth GradCAM++ [Ome+19], which incorporates Gaussian noise similar to SmoothGrad. LayerCAM [Jia+21] and XGradCAM [Fu+20] have also contributed alternative weighting schemes for gradient computation.

4.1.5 Ensemble and Perturbation Methods

SmoothGrad [Smi+17] reduces noise in gradient-based attributions by averaging over multiple perturbed inputs:

$$A_{smooth}(x) = \frac{1}{n} \sum_{i=1}^n A_{grad}(x + \epsilon_i)$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$ represents random noise samples with standard deviation σ . The number of samples n is a crucial hyperparameter that balances attribution quality with computational cost, typically ranging from 25 to 50 samples in practice.

RISE (Random Input Sampling for Explanation) [PDS18] generates attribution maps by randomly masking the input and aggregating predictions:

$$A_{RISE}(x) = \frac{1}{N} \sum_{i=1}^N m_i \cdot f^{(c)}(x \odot m_i),$$

where m_i represents random binary masks that partially occlude regions of the input image, and \odot denotes element-wise multiplication. These masks are generated using a sliding window approach: a small window moves across the input image in fixed steps, and at each position, the algorithm randomly decides whether to preserve or mask out the pixels within that window. The process creates multiple masks, each revealing different combinations of image regions. By aggregating the model’s predictions across these various masked versions of the input, RISE builds a comprehensive attribution map that highlights the regions most important for the model’s decision.

4.1.6 Limitations

A significant limitation of attribution maps lies in their inherent inability to capture feature interdependencies. Attribution maps, by design, reduce complex feature relationships to a spatial map of individual importance scores. This “flattening” of feature relationships into independent importance values fails to represent how neural networks actually process information through feature interactions.

Consider a model classifying dog images: while an attribution map might correctly identify ears and tail as important features by assigning high importance scores to these regions, it cannot represent how these features work together in the model’s decision-making process. The map format itself imposes this limitation, as it cannot explicitly represent the combined relationships between features that the model has learned.

Another fundamental limitation is the inability of attribution maps to represent the importance of feature absence in model decisions. Neural networks

often learn to recognize patterns not only through the presence of certain features but also through their absence. For example, when classifying dog breeds, the absence of specific features (such as a long snout for a chihuahua) might be as crucial to the model’s decision as the presence of others.

Most attribution methods focus exclusively on highlighting present features that contribute positively to model predictions. For this reason, the CAMs make use of the ReLU (maximum) function. This one-sided representation fails to capture how the lack of certain features might influence the model’s decision-making process.

4.2 CONSISTENCY OF ATTRIBUTION MAPS

The interpretation of deep neural networks through attribution maps would be greatly simplified if different attribution methods produced consistent results. In such an ideal scenario, the choice of attribution method would be largely inconsequential. However, our empirical analysis reveals that attribution maps can vary substantially across different methods, highlighting the critical importance of systematic evaluation approaches.

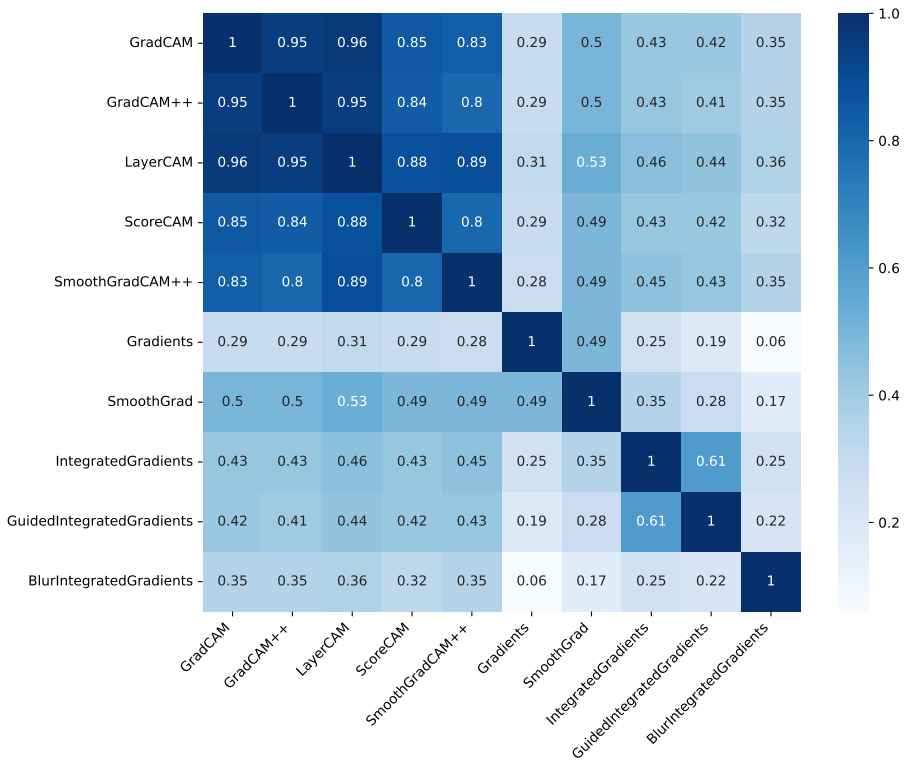


Figure 16: This plot illustrates the degree of similarity among all attribution maps. The matrix was computed by averaging the individual correlation results across all attribution maps in the ImageNet dataset with ResNet-50. Source: [NSK24a].

To quantify the differences between attribution maps, we conducted a comprehensive similarity analysis using the Pearson correlation coefficient. For two attribution maps with pixels x_i and y_i , the correlation is defined as:

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

where \bar{x} and \bar{y} represent the respective sample means. The coefficient r_{xy} ranges from -1 to 1, where:

- $r_{xy} = 1$ indicates perfect similarity
- $r_{xy} = 0$ suggests uncorrelated maps (effectively random noise)
- $r_{xy} = -1$ represents completely inverted attribution patterns

We conducted our analysis on a diverse subset of 1,000 images from the ImageNet dataset, examining the consistency across attribution maps for both ResNet-50 and ConvNeXt-tiny architectures (see section 2.2.4 for an overview of architectures). The results, visualized in Figure 16 for ResNet-50, reveal remarkably low levels of consistency. The average similarity across attribution methods for ResNet-50 was only 48%. Even more striking, the ConvNeXt-tiny architecture exhibited an even lower average similarity of 22%. We provide additional qualitative results in the appendix figure 37 and figure 38.

These findings demonstrate that attribution maps can produce substantially different interpretations of the same model’s decision-making process. Such significant variations underscore the necessity of developing robust evaluation metrics to determine which attribution methods most accurately reflect the true decision-making process of convolutional neural networks.

4.3 EVALUATING ATTRIBUTION METHODS

The evaluation of attribution methods presents unique challenges due to the lack of absolute ground truth data for feature importance. While model predictions can be validated against known labels, there is no definitive “true” attribution map for comparison. This fundamental challenge has led to the development of various evaluation approaches.

4.3.1 *Qualitative Evaluation*

Expert assessment remains a cornerstone of attribution method evaluation. Domain specialists can assess whether attribution maps highlight regions that align with their professional knowledge and experience. In the context of wood identification, for instance, wood anatomists can evaluate whether attribution maps emphasize known discriminative features such as vessel pit arrangements or ray characteristics.

However, relying solely on expert evaluation has important limitations. Experts may be biased toward known features, potentially overlooking novel patterns that neural networks might discover. Furthermore, qualitative assessments are inherently subjective and can vary between experts, making it difficult to establish consistent comparisons between different attribution methods.

4.3.2 *Quantitative Evaluation*

To address the limitations of qualitative assessment, several quantitative evaluation approaches have been developed. These methods can be broadly categorized into two groups: perturbation-based metrics and annotation-based comparisons.

Perturbation-based metrics evaluate attribution maps by measuring how modifications to highly attributed regions affect model predictions. The Dele-

tion metric [FV17; PDS18] progressively removes pixels in order of their attribution values:

$$Deletion(x) = \frac{1}{L} \sum_{l=1}^L f^{(c)}(x_l),$$

where x_l represents the input with the top l most attributed pixels removed. The metric operates by iteratively masking the most significant regions with zero values and measuring the model’s class probability after each modification. Lower scores indicate better attribution maps, as removing important features should rapidly decrease model confidence. This process is visualized in Figure 17, where highly attributed regions (shown in red in the saliency map) are systematically removed in sequential stages.

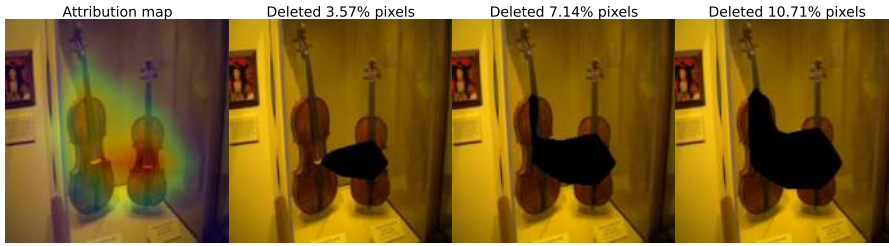


Figure 17: Visualization of the Deletion method’s progressive evaluation process. From left to right: (1) Original image overlaid with saliency map highlighting attribution importance, (2-4) Sequential stages showing incremental deletion of regions, from most to least significant, based on their attribution scores. Red regions in the first image indicate higher attribution values. Source: [NSK24a].

The Insertion metric [FV17; PDS18] operates in the opposite direction, measuring confidence increase as highly attributed pixels are progressively added to a baseline image:

$$Insertion(x) = \frac{1}{L} \sum_{l=1}^L f^{(c)}(x'_l)$$

where x'_l contains only the top l most attributed pixels. The process typically begins with a Gaussian-blurred version of the original image and gradually restores pixels according to their attribution values.

As demonstrated in Figure 18, the model’s confidence evolves from zero when the image is fully blurred to its original prediction value as pixels are restored. Higher Area Under Curve (AUC) values indicate more effective attribution maps, suggesting that restoring important pixels quickly reconstructs the original prediction.

A second quantitative approach involves comparing attribution maps against expert annotations or segmentation masks. For applications where such annotations exist, metrics like Intersection over Union (IoU) can measure alignment between attribution maps and expert-identified regions:

$$IoU(a, e) = \frac{\text{Area of Overlap}}{\text{Total Area Covered}}$$

where a represents the thresholded attribution map and e represents the expert-annotated mask. The thresholding of attribution maps is typically performed using either a fixed percentile of attribution values (e.g., top 20%) or an adaptive threshold based on the distribution of attribution values.

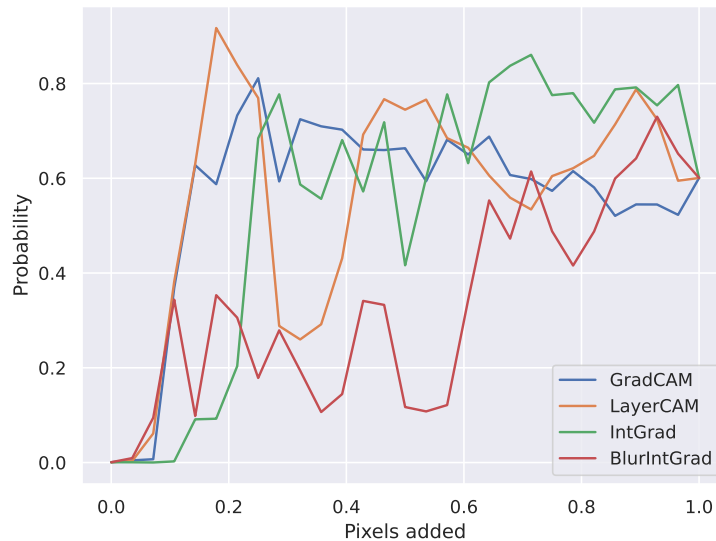


Figure 18: Standard Insertion metric. The graph shows model confidence (probability) over the progressive restoration of image pixels, ordered by their attribution importance. Higher Area Under Curve (AUC) values indicate more effective attribution maps, as they suggest that restoring important pixels quickly reconstructs the original prediction.

However, this approach has significant limitations. Most crucially, it assumes that the neural network bases its decisions on the same features that experts use, which may not be the case. Neural networks might discover novel discriminative features that experts have not identified, or might use combinations of features differently than human experts. Therefore, low alignment with expert annotations does not necessarily indicate poor attribution quality – it might instead reveal new, valid decision-making patterns.

There are many other metrics, but with similar shortcomings [Zha+16; RR22].

4.4 IMPROVED MASKING FOR EVALUATING ATTRIBUTION METHODS

Building on our previous discussion of quantitative evaluation methods, this section presents an in-depth analysis of deletion and insertion metrics to develop a more robust evaluation framework. While we introduced these metrics in their basic form earlier, we now examine their theoretical and empirical limitations in detail. Based on this analysis, we will present our enhanced metric, which addresses several key shortcomings of these existing approaches, though some fundamental limitations remain beyond its scope.

4.4.1 Foundations in Human Vision Research

Our choice of deletion and insertion metrics as a foundation stems from their alignment with established findings in human vision research. The fundamental principle behind these metrics – progressively removing or adding image regions – finds strong theoretical support in established experimental paradigms from cognitive psychology that investigate human visual processing under partial information conditions.

Most notably, the “Bubbles” technique [GS01] provides a direct methodological precedent for the evaluation framework. In these experiments, researchers

systematically revealed or masked different image regions to identify which areas were diagnostic for human recognition tasks. The technique demonstrated that human recognition performance degrades in predictable ways as diagnostic regions are masked, while remaining robust when key features are visible. This empirical finding suggests two important principles for attribution evaluation:

1. The systematic manipulation of visual information through controlled revelation or masking can effectively probe the importance of different image regions
2. The degradation of recognition performance under such manipulations can serve as a reliable metric for feature importance

These principles directly inform our attribution evaluation framework. Just as the Bubbles technique measures human recognition performance under controlled information removal, deletion metrics assess how neural network confidence changes as pixels deemed important by attribution methods are systematically removed. Similarly, insertion metrics parallel experiments where visual information is gradually revealed, measuring how quickly recognition performance improves as critical features become visible.

4.4.2 Symbol Grounding Problem

While the human vision research provides compelling support for our evaluation approach, implementing these insights in neural networks reveals a deeper challenge. Attribution evaluation must bridge two fundamentally different types of information processing: the statistical information of pixels and the semantic understanding of image content. While deletion and insertion metrics operate at the pixel level, the neural networks they analyze make decisions based on high-level semantic features. This disconnect manifests as a specific instance of the symbol grounding problem [Hargo], which addresses how meaningful concepts emerge from raw sensory patterns (here: pixel data).

At the statistical level, we deal with raw numerical data: pixel values representing brightness and color. When pixels are deleted, information loss is monotonic and measurable, which makes intuitive sense: as we progressively remove more pixels, we irreversibly lose more information about the original image. This monotonicity can be formally understood through the Data Processing Inequality (DPI):

Definition 4.4.1 (Data Processing Inequality (DPI) [CT06]). Let X , Y , and Z form a Markov chain $X \rightarrow Y \rightarrow Z$. Then the mutual information I between these variables satisfies:

$$I(X; Y) \geq I(X; Z),$$

where $I(X; Y) = H(Y) - H(Y | X)$, and $H(\cdot)$ denotes entropy.

While the DPI is traditionally stated for stochastic Markov chains, it also applies in deterministic settings where each variable is a function of the previous one. Specifically, if $Y = f_1(X)$ and $Z = f_2(Y)$ are deterministic transformations, then Y is fully determined by X , and Z is fully determined by Y . In such cases, the conditional entropy $H(Y | X)$ is zero, so the mutual information simplifies to $I(X; Y) = H(Y)$. Similarly, $I(X; Z) = H(Z)$, and the DPI becomes: $H(Y) \geq H(Z)$.

Intuitively, the DPI tells us that as information flows through a chain of transformations, the statistical dependency between variables can only decrease. Consider an image sequence $X_0, X_{\text{del}_1}, \dots, X_{\text{del}_k}$, where each $X_{\text{del}_{i+1}}$ is derived from X_{del_i} through pixel deletion based on a fixed attribution ranking (with $X_{\text{del}_0} = X_0$ being the original image). Since each deletion step removes a deterministic subset of pixels from the previous image without introducing new information, each successive image $X_{\text{del}_{i+1}}$ contains strictly less information about the original image X_0 than its predecessor X_{del_i} . By the DPI, this guarantees that $I(X_0; X_{\text{del}_1}) \geq I(X_0; X_{\text{del}_2}) \geq \dots \geq I(X_0; X_{\text{del}_k})$, where the mutual information between each corrupted version and the original image decreases monotonically as more pixels are removed.

A similar information-theoretic perspective can be applied to the feed-forward processing within a neural network itself. In a simple sequential architecture, the layers form a processing cascade where the input to layer L_{i+1} is exclusively the output of layer L_i . This structure forms a Markov chain: $X \rightarrow L_1 \rightarrow L_2 \dots \rightarrow L_n$, where X is the input image and L_i is the activation map of the i -th layer. Because each layer cannot create new information about the original input X and can only transform the information received from the preceding layer, the DPI directly applies.

Note that skip connections (e.g., ResNet) create a directed acyclic graph (DAG) rather than a simple Markov chain $L_i \rightarrow L_{i+1} \rightarrow L_{i+2}$. However, the DPI still applies to any sub-chain within this DAG. While multiple information paths exist between layers, no single path can increase mutual information about the original input.

To empirically validate this information-theoretic perspective, we introduce a “noisy channel” at each layer by applying dropout only at layer i during inference (figure 19).

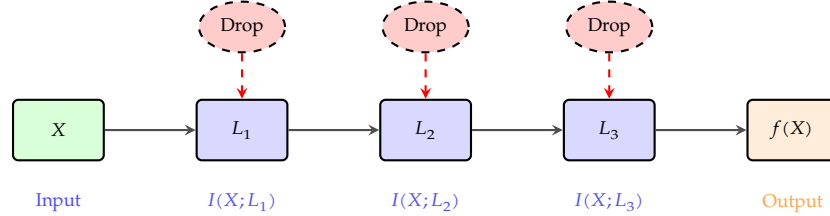


Figure 19: Dropout-based validation of the Data Processing Inequality in neural networks. Injecting noise at earlier layers (which contain more mutual information about the input) should cause larger drops in output confidence, following the pattern $\Delta f(X; L_1) \geq \Delta f(X; L_2) \geq \Delta f(X; L_3)$.

We then measure how much the model’s predicted probability for the true class falls. By the DPI, corrupting an earlier layer can only remove information that all later layers depend on, so it should never cause less damage to the final output than corrupting a later layer. In practice, we therefore expect that the confidence drop $\Delta f(X; L_i) = f(X) - f(X | L_i \text{ corrupted})$ should be largest for early layers, with $\Delta f(X; L_1) \geq \Delta f(X; L_2) \geq \Delta f(X; L_3) \geq \dots \geq \Delta f(X; L_n)$, where $f(X)$ represents the model’s confidence on the original input.

We conducted a systematic analysis across four neural architectures (ResNet18, ConvNext-Tiny, EfficientNet-Bo, and DenseNet121), targeting four representative layers in each model from early feature extraction to late semantic processing. For each layer, we inject controlled corruption by applying dropout at rates from 0% to 100%, randomly zeroing activations before continuing forward propagation.

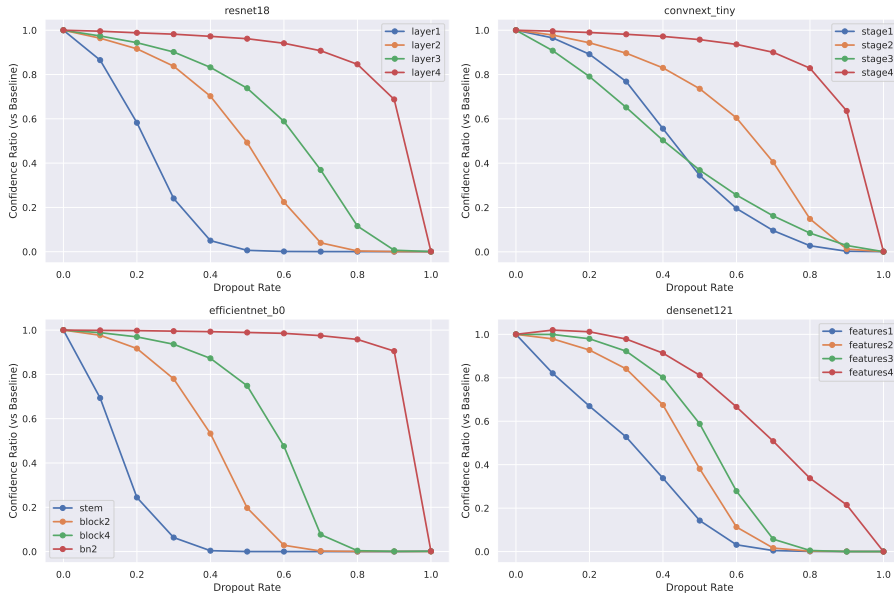


Figure 20: Layer-wise dropout provides evidence for the Data Processing Inequality in neural networks: early layers show steeper confidence drops under input corruption, consistent with higher mutual information $I(X; L_i) \geq I(X; L_{i+1})$ closer to the input.

As shown in Figure 20, the results support the DPI-based prediction: early layers exhibit significantly steeper confidence degradation compared to deeper layers when corrupted. This monotonic pattern across all architectures is consistent with the expectation that $I(X; L_i)$ decreases with depth, reflecting progressively weaker input-output dependencies. The findings also align with the distributed information property discussed in section 2.3.1, which suggests greater information redundancy in later layers.

Apart from this statistical information flow, neural networks must also process semantic information: the high-level recognition that “this is a dog” or “this contains a face.” Unlike statistical information, semantic understanding often requires contextual relationships and prior knowledge that cannot be directly extracted from individual pixel values. Consider, for example, starting with a black image and progressively revealing pixels. When we first reveal what appears to be a tail, the model might assign a 20% confidence to the “dog” classification. However, when we reveal additional features showing this tail belongs to a cat, the model’s confidence in “dog” drops to 0%. This non-monotonic relationship demonstrates how features are inherently interconnected – the interpretation of the tail depends critically on its relationship with other features in the image.

CNNs must bridge this gap between statistical and semantic information, learning to map from raw pixel information to semantic concepts. As discussed in section 2.3.2, neural networks accomplish this through hierarchical processing: early layers detect basic features like edges and textures, middle layers combine these into more complex shapes and patterns, and deeper layers assemble these into high-level semantic concepts. While the DPI constrains each successive layer to contain less mutual information about the raw input, the network transforms this information by progressively discarding statistically superfluous details (e.g., lighting variations, pixel noise).

According to the Information Bottleneck theory, networks learn to compress input information while preserving task-relevant features [TZ15; ST17]. This creates a tension: while the DPI ensures that statistical mutual information $I(X; L_i)$ decreases with depth, the network simultaneously increases semantic mutual information $I(Y; L_i)$ between layers and the target labels. The result is that later layers become increasingly specialized for the classification task.

This analysis reveals two fundamentally different information types in neural networks. Statistical information behaves predictably due to DPI. Semantic information, however, is far more complex: it emerges from interconnected features in high-dimensional space, where individual pixels cannot be interpreted independently of their broader context.

Table 10 summarizes these contrasting behaviors, highlighting the core challenge for attribution methods and metrics.

Information Type	Through Layers (L_i)	Under Deletion (X_{del_i})
Statistical	Decreases monotonically DPI: $I(X; L_i) \geq I(X; L_{i+1})$	Decreases monotonically DPI: $I(X; X_{\text{del}_i}) \geq I(X; X_{\text{del}_{i+1}})$
Semantic	Increases with depth (edges \rightarrow textures \rightarrow parts \rightarrow concepts)	Non-monotonic (depends on feature interactions and context)

Table 10: Information flow in neural networks: statistical information follows the Data Processing Inequality (DPI). X is the original input image, X_{del_i} represents progressively more corrupted versions (e.g., with more pixels deleted), and L_i is a layer output. While statistical information degrades predictably, semantic interpretation exhibits more complex behavior.

This statistical-semantic distinction has direct implications for attribution evaluation methods. The main insight lies in recognizing that deletion metrics operate at the statistical level rather than the semantic level. Attribution methods do not manipulate semantically meaningful concepts like “tail” or “head” but instead remove fixed-size pixel subsets based on attribution scores. For instance, in figure 17, each step removes 3.57% of the highest-attributed pixels, regardless of their semantic coherence.

This pixel-based deletion process constitutes purely statistical information removal. By the Data Processing Inequality, such deletions guarantee monotonic information loss: $I(X; X_{\text{del}_1}) \geq I(X; X_{\text{del}_2}) \geq \dots \geq I(X; X_{\text{del}_n})$, where each successive deletion step X_{del_i} contains strictly less mutual information about the original image X .

While translating this statistical monotonicity to neural network confidence is non-trivial due to the highly non-linear nature of the mapping function $f(\cdot)$, our dropout experiments provide strong empirical evidence for an analogous monotonic confidence degradation: $\mathbb{E}[f(X)] \geq \mathbb{E}[f(X_{\text{del}_1})] \geq \dots \geq \mathbb{E}[f(X_{\text{del}_k})]$.

The key insight from our layer-wise corruption analysis is that increasing statistical corruption (dropout rate) monotonically decreases confidence across all network depths, despite the complex non-linear transformations applied by subsequent layers. The reason lies in the type of corruption applied. Dropout randomly zeroes layer activations without regard to their semantic meaning, creating purely statistical corruption. Unlike semantic manipulations that might reveal new interpretations (e.g., occluding a “head” to increase the confidence of “cat”), unstructured dropout cannot generate semantic reinter-

pretations. This mirrors attribution-based pixel deletion: both methods apply unstructured corruption, which do not respect semantic boundaries.

While individual images may exhibit non-monotonic patterns due to random semantic feature interactions, the statistical nature of pixel deletion ensures these effects average out across sufficiently large datasets. Therefore, our theoretical framework predicts reliable monotonic degradation for statistical corruption. We demonstrate both individual-level monotonicity patterns (table 11) and average monotonic behavior (figure 21) in our evaluation section.

Despite this theoretical guarantee, current deletion implementations often fail to achieve the expected monotonicity, particularly on small sample sizes (figure 18). The observed variations are too severe to be attributed solely to minor semantic interactions. Instead, the root cause lies in distribution shift: the evaluation process generates images with partially revealed regions that differ substantially from the natural images used during training. These artificial, patchy images fall outside the model’s training distribution, leading to unreliable confidence estimates that no longer accurately reflect the underlying statistical information content [Nie+24b; NSK24a].

This analysis reveals why attribution evaluation faces fundamental challenges. While we seek to understand semantic feature importance, our metrics are constrained to statistical information manipulation at the pixel level. The information framework shows that this limitation cannot be fully overcome as long as attribution maps operate on pixels rather than semantic concepts. However, we can work to ensure reliable evaluations within these constraints by addressing the distribution shift problem that disrupts even the expected statistical monotonicity.

4.4.3 Properties

Given that we must operate at the pixel level, the key question becomes how we should modify input images to evaluate attribution methods. The standard Deletion metric simply sets pixel values to zero, but this is just one possible approach among many potential masking operators. Our analysis of information processing in neural networks, from human vision research to the Data Processing Inequality, suggests that we need masking operators that respect how information degrades in these systems. To formalize these requirements, we propose three essential properties that any masking operator must satisfy to enable reliable attribution evaluation.

Let $f : \mathbb{R}^{h \times w \times c} \rightarrow [0, 1]$ be a neural network classifier that maps an input image to its normalized confidence score for a target class. Let I denote the set of all pixel coordinates in the image. For any subset $S \subseteq I$, let $M_S : \mathbb{R}^{h \times w \times c} \rightarrow \mathbb{R}^{h \times w \times c}$ be a masking operator that modifies the pixels at coordinates in S . We define:

Property 4.4.1 (Statistical Monotonicity). For any two coordinate subsets $S_1, S_2 \subseteq I$ such that $S_1 \subseteq S_2$:

$$\mathbb{E}_X[f(M_{S_1}(X))] \geq \mathbb{E}_X[f(M_{S_2}(X))]$$

where X is a random variable over the image space $\mathbb{R}^{h \times w \times c}$, and $f(M_{S_i}(X))$ denotes the model’s confidence after applying the mask to the subset S_i . Note that while this property holds in expectation, individual images may exhibit variance due to random semantic interactions between features.

This property formalizes the monotonic behavior observed in our dropout experiments (figure 20) and aligns with experiments on human vision. It

ensures that removing more pixels cannot increase the metric score, preventing the erratic fluctuations identified in Deletion/Insertion.

Property 4.4.2 (Completeness). For any input image $x \in \mathbb{R}^{h \times w \times c}$:

$$f(M_\emptyset(x)) = f(x), \quad f(M_I(x)) = 0$$

These conditions specify how the masking operator should behave at extremes: applying no mask leaves the neural network’s confidence unchanged, while masking all pixels reduces the confidence to zero. This property prevents trivial solutions, such as the identity operator, which would simply return the original image regardless of the attribution scores.

Property 4.4.3 (Significance). Let $A, R : \mathbb{R}^{h \times w \times c} \rightarrow \mathbb{R}^{h \times w}$ be attribution maps, where A assigns higher importance scores to relevant features and R assigns scores randomly. Then there exists a threshold $k_0 > 0$ such that for any $k \geq k_0$ and any input image $x \in \mathbb{R}^{h \times w \times c}$:

$$f(M_{S_A^k}(x)) < f(M_{S_R^k}(x))$$

where S_A^k and S_R^k are k -element subsets of I containing the coordinates of the k highest-scoring pixels according to A and R respectively.

This property ensures that masking pixels according to attribution scores produces measurably different results than random masking, providing a clear baseline for evaluation. It guarantees that our evaluation framework can distinguish meaningful attribution methods from random feature selection.

By focusing explicitly on the masking operator’s behavior, we establish clear criteria for valid attribution evaluation while avoiding the distributional artifacts that plague Deletion and Insertion.

4.4.4 Masking Operators

Having established the key properties for valid masking operators, we begin by formally defining several masking operators that can be used in attribution evaluation.

Let $x \in \mathbb{R}^{h \times w \times c}$ be an input image.

Definition 4.4.2 (Gaussian Blur). The Gaussian blur mask M^{GB} creates a local averaging effect in the masked regions:

$$M_S^{\text{GB}}(x)_{i,j} = \begin{cases} (G_\sigma * x)_{i,j} & \text{if } (i,j) \in S \\ x_{i,j} & \text{otherwise} \end{cases} \quad (10)$$

where G_σ represents a Gaussian kernel with standard deviation σ , and $*$ denotes the convolution operation.

Definition 4.4.3 (Additive Noise). The additive noise mask M^{AN} perturbs pixels by adding random noise:

$$M_S^{\text{AN}}(x)_{i,j} = \begin{cases} x_{i,j} + \eta, \eta \sim P & \text{if } (i,j) \in S \\ x_{i,j} & \text{otherwise} \end{cases} \quad (11)$$

where P is a probability distribution from which the noise η is drawn. A common choice is the Gaussian distribution $\mathcal{N}(0, \sigma^2)$.

Definition 4.4.4 (Brightness Adjustment). The brightness adjustment mask M^{BA} controls the pixel intensity in masked regions:

$$M_S^{\text{BA}}(x)_{i,j} = \begin{cases} \alpha x_{i,j} & \text{if } (i,j) \in S \\ x_{i,j} & \text{otherwise} \end{cases} \quad (12)$$

where α determines the brightness level. Setting $\alpha = 0$ yields complete darkness.

Definition 4.4.5 (Identity). The identity mask M^{I} preserves all pixel values:

$$M_S^{\text{I}}(x)_{i,j} = x_{i,j} \quad \forall (i,j) \quad (13)$$

Note that this mask violates the Completeness property, as it cannot effectively reduce model confidence even when masking the entire image.

These masking operators represent different approaches from the image processing literature for modifying image content. For attribution evaluation, the Insertion method uses Gaussian blur while the Deletion method uses black pixels (brightness adjustment with $\alpha = 0$).

While some of these standard masking operators are widely used, they each face fundamental limitations. Gaussian blur introduces artificial smoothing that may create unrealistic feature interactions. Additive noise can generate pixel values and patterns that lie outside the natural image distribution. Brightness adjustment creates sharp transitions at mask boundaries that rarely occur in natural photographs. These artifacts become particularly problematic when masks are applied to arbitrary image regions, where maintaining local coherence is crucial.

A natural question is whether more sophisticated approaches, such as inpainting, could serve as masking operators. Consider a sequence of photographs where a truck gradually moves in front of a cat, eventually completely obscuring it. While this represents a perfect way of removing information naturally, creating such masking artificially through inpainting would require understanding the semantic content, generating plausible occluding objects, and placing them correctly in 3D space. This becomes especially problematic since attribution maps typically highlight pixels of varying shapes and sizes as important, as shown in figure 17. It creates a circular dependency where effective masking would first require solving the very problem of image understanding that attribution methods aim to explain.

These limitations motivate the need for a masking operator that can remove information while preserving the natural image statistics and avoiding artificial artifacts. In the following section, we introduce such an approach based on adversarial perturbations.

4.4.5 Adversarial Perturbation Mask

Instead of relying on traditional image transformations, we propose leveraging adversarial perturbations as a masking operator (refer to section 2.3.4 for an introduction into these attacks). The key insight is that we can remove information from the image by making minimal, carefully targeted changes that maximally affect the network’s output while preserving the natural image statistics.

Definition 4.4.6 (Adversarial Perturbation). The adversarial perturbation mask M^{AP} applies minimal pixel-wise changes that maximize the network’s loss with respect to the target class:

$$M_S^{\text{AP}}(x)_{i,j} = \begin{cases} x_{i,j} + \epsilon \cdot \text{sgn}\left(\frac{\partial L}{\partial x_{i,j}}\right) & \text{if } (i,j) \in S \\ x_{i,j} & \text{otherwise} \end{cases} \quad (14)$$

where L is the loss function for the target class, and ϵ controls the perturbation magnitude. For 8-bit images, setting $\epsilon = \frac{1}{255}$ ensures the smallest possible discrete perturbation of ± 1 per pixel.

The adversarial perturbation mask can be viewed as a sophisticated generalization of traditional masking approaches, particularly the brightness adjustment mask. While brightness adjustment applies a uniform scaling factor α to all pixels in the masked region, adversarial perturbation implements a local, content-aware operator that selectively modifies each pixel by $\pm\epsilon$ based on its impact on the network’s prediction. This makes it analogous to an adaptive brightness adjustment that operates at a per-pixel level, but with two key advantages: it targets the network’s learned features rather than raw pixel intensities, and it requires only minimal perturbations to achieve the desired masking effect.

This approach addresses the fundamental limitations of previous masking operators in several ways. First, by using the Fast Gradient Sign Method (FGSM) [GSS15], we modify each masked pixel by the minimal amount that still affects the network’s output. Unlike arbitrary transformations like blur or noise that can create patterns the network wasn’t trained on, these perturbations work within the network’s learned feature space. Second, when $\epsilon = \frac{1}{255}$, the changes are comparable to subtle lighting variations that naturally occur in photographs, ensuring the perturbed images remain within the training distribution. This is particularly important when applying masks to arbitrary patches, where maintaining local coherence becomes crucial.

Critically, this masking operator maintains smooth transitions at patch boundaries since the pixel-wise changes are imperceptible and locally adaptive. The magnitude of perturbation can be controlled through ϵ , though larger values may begin to introduce visible artifacts and shift away from the natural image distribution. This makes the adversarial perturbation mask particularly suitable for attribution evaluation, as it allows us to systematically remove information without confounding the network’s response with distribution artifacts.

4.5 EVALUATION OF ATTRIBUTION MAP METRICS

Given the black-box nature of neural networks, it is not feasible to mathematically prove whether a particular mask satisfies specific properties. Instead, we perform empirical evaluations across different architectures and datasets to assess the validity of various masking operators.

4.5.1 Statistical Monotonicity

We conducted a systematic evaluation using centered occlusion to assess monotonicity. This approach leverages the fact that objects in ImageNet are typically centered, allowing us to progressively expand a square mask from the image center to analyze how different masking operators affect network confidence

across varying levels of occlusion. Our experimental design compared four masking operators: adversarial perturbation (our proposed method), Gaussian noise, brightness reduction ($\alpha = 0$, i.e. black pixels), and Gaussian blur. The occlusion rate varied from 0.0 (no masking) to 1.0 (complete masking), tracking the relationship between information removal and model confidence.

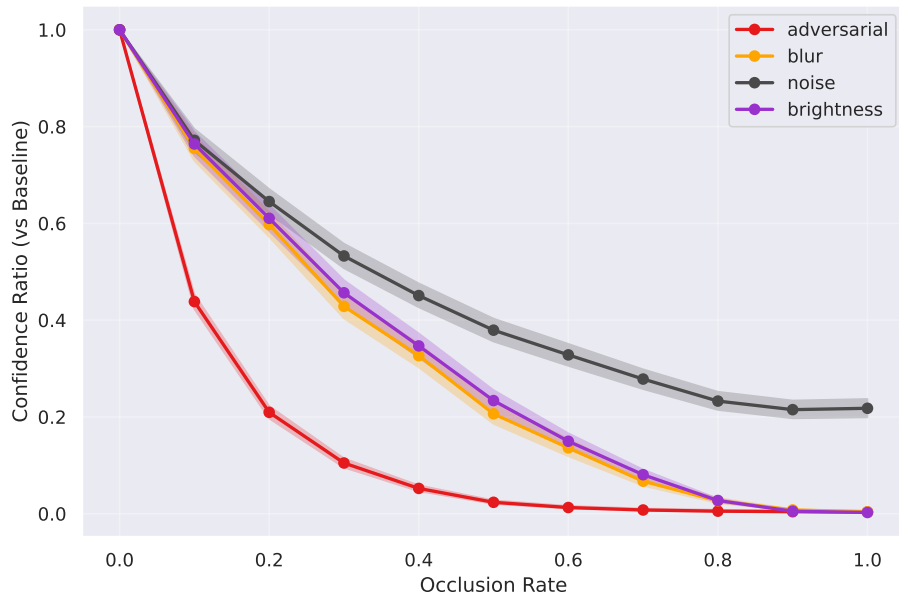


Figure 21: The confidence ratio $r(x) = \frac{f(x')}{f(x)}$ across occlusion rates for different masking operators, where $f(x')$ and $f(x)$ are the model confidences on perturbed and original images respectively. Shaded regions show 95% confidence intervals. Analysis performed on 1500 ImageNet images. Our approach adversarial masking shows the smallest variance.

Figure 21 presents the results for ResNet-18 on ImageNet, revealing several critical insights about masking behavior. While all methods show monotonic decrease in their mean confidence as occlusion increases, they differ fundamentally in their stability patterns. Most notably, adversarial perturbation demonstrates a dramatically steeper initial confidence reduction compared to traditional methods, dropping to around 0.2 confidence ratio by just 20% occlusion. This stark difference in behavior persists throughout the experiment, with traditional methods showing much more gradual declines.

Given that ImageNet images consistently position their primary objects in the center, we would expect any masking of these central regions to cause immediate degradation in confidence. Our adversarial perturbation mask confirms this expectation, while traditional masking approaches show more gradual declines. This finding is further supported by the confidence reduction patterns observed in ResNet-18’s early layers, as shown by the blue lines in figure 20. This parallel between dropout effects and adversarial perturbation suggests that our adversarial approach may more accurately capture the progressive degradation of visual information in the network.

The methods also exhibit distinct variance patterns, visible in their confidence bands. While all traditional methods (brightness, blur, and noise) show wider confidence bands than adversarial perturbation, the differences are most pronounced with Gaussian noise, which displays the highest variance throughout the experiment. Adversarial perturbation maintains notably

narrower and more consistent confidence bands, suggesting it provides more reliable and predictable information removal.

These results demonstrate that adversarial perturbation achieves both faster confidence reduction and more consistent behavior across samples, better aligning with the theoretical predictions of information processing chains. While individual samples may still experience semantic interactions, the narrower confidence bands suggest adversarial perturbation more effectively aggregates these effects across samples.

Having established the behavior of different masking operators on ResNet-18 with ImageNet, we now extend our analysis across multiple architectures and datasets to validate the generality of our findings. In [NSK24a], we conducted experiments across three diverse datasets: ImageNet [Rus+15], Oxford-IIIT Pet Dataset [Par+12], and ChestX-ray8 [Wan+17]. For each dataset, we evaluated five CNN architectures: ResNet-50 [He+15a], EfficientNet-Bo [TL19], DenseNet-121 [HLW16], ConvNeXt-Tiny [Liu+22], and RepVGG-Bo [Din+21].

To quantify the consistency of masking operators at the individual sample level, we define two metrics:

Definition 4.5.1 (Sample-level Monotonicity). For a sequence of masked images x_1, \dots, x_n with increasing mask size, the monotonicity score measures the fraction of consecutive pairs that maintain the expected confidence decrease:

$$\text{Monotonicity}(f, x_i) = \frac{1}{n-1} \sum_{i=1}^{n-1} \mathbf{1}[f(x_{i+1}) \leq f(x_i)],$$

where f is the model’s confidence function and $\mathbf{1}[\cdot]$ is the indicator function.

Definition 4.5.2 (Sample-level Smoothness). The smoothness score measures the average deviation from the expected linear degradation in confidence:

$$\text{Smoothness}(f, x_i) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n-1} (f(x_{i+1}) - f(x_i) - \mu)^2},$$

where μ is the mean difference in confidence between consecutive steps.

We evaluate four masking configurations:

- $M_{\text{Del}}^{\text{BA}}$: Deletion using brightness adjustment ($\alpha = 0$), starting from original image
- $M_{\text{Ins}}^{\text{GB}}$: Insertion using Gaussian blur, starting from fully blurred image
- $M_{\text{Ins}}^{\text{BA}}$: Insertion using brightness adjustment ($\alpha = 0$), starting from black image
- $M_{\text{Ins}}^{\text{AP}}$ (ours): Insertion using adversarial perturbation, starting from fully perturbed image

The results in Table 11 demonstrate that the direction of masking (insertion vs. deletion) has minimal impact on monotonicity. We also see that our adversarial perturbation approach achieves significantly higher monotonicity and smoothness scores across all datasets and architectures. While some fluctuations remain due to semantic interactions in the feature space, they are substantially reduced compared to traditional masking approaches.

Masking Operator	Monotonicity \uparrow	Smoothness \downarrow
$M_{\text{Del}}^{\text{GB}}$	0.648	1.712
$M_{\text{Ins}}^{\text{GB}}$	0.671	1.658
$M_{\text{Ins}}^{\text{BA}}$	0.654	1.238
$M_{\text{Ins}}^{\text{AP}}$ (ours)	0.967	0.891

Table 11: Sample-level monotonicity and smoothness scores averaged across all datasets and architectures. Higher monotonicity and lower smoothness scores indicate more reliable masking behavior.

4.5.2 Completeness

Our experimental results in figure 21 also validate the Completeness property. We observe that at zero occlusion rate, all masking operators preserve the baseline confidence (ratio = 1.0), satisfying the first condition. At full occlusion, adversarial perturbation, brightness reduction, and Gaussian blur effectively reduce the model’s confidence to near zero, satisfying the second condition. While Gaussian noise demonstrates less complete masking at our chosen variance ($\sigma = 50$), increasing the noise magnitude would achieve lower final confidence but at the cost of higher variance in intermediate occlusion rates.

4.5.3 Significance

The Significance property requires that masking important image regions (according to attribution) should affect the model’s confidence more than masking random regions. This provides a basic test for attribution evaluation methods. We can test this property in two ways using baseline attribution maps:

First, we generate a *uniform* baseline by sampling attribution scores from a uniform distribution. This directly corresponds to the random attribution map R in the property definition – if a masking operator is valid, removing features based on actual attribution scores should produce larger confidence drops than removing uniformly random features.

Second, we introduce a more nuanced test using a *canny* baseline [Can86] that assigns importance based purely on edge detection. This baseline captures low-level image structure but lacks semantic understanding, providing an intermediate reference point between random selection and meaningful attribution. A valid masking operator should still show larger confidence impacts from semantic attribution compared to this structure-only baseline.

To evaluate masking operators against these baselines, we conducted experiments across the same architectures and datasets we already used for table 11. For each configuration, we tested whether the operator maintained the theoretically expected ordering: meaningful attribution methods should outperform the Canny baseline, which should in turn outperform uniform random selection.

Table 12 shows how often each masking operator achieved this ordering:

The results show clear differences between masking operators. Progressive deletion with black pixels ($M_{\text{Del}}^{\text{BA}}$) performs poorly, maintaining correct ordering in only 2-3 of 15 cases. Switching to progressive insertion ($M_{\text{Ins}}^{\text{BA}}$) works better against random selection but struggles with the edge-based baseline. Gaussian blur ($M_{\text{Ins}}^{\text{GB}}$) improves baseline discrimination overall, but its occasional failures point to underlying weaknesses.

Method	Canny 2nd to last \uparrow	Uniform last \uparrow
$M_{\text{Del}}^{\text{BA}}$	3	2
$M_{\text{Ins}}^{\text{BA}}$	4	11
$M_{\text{Ins}}^{\text{GB}}$	12	12
$M_{\text{Ins}}^{\text{AP}}$ (ours)	15	15

Table 12: Number of cases in which Canny edge detector and uniform came last. The maximum achievable score is 15 (3 datasets with 5 architectures).

Only our perturbation-based method ($M_{\text{Ins}}^{\text{AP}}$) achieves perfect baseline discrimination across all configurations, consistently ranking meaningful attribution above edge detection and random selection. The consistent failure of traditional methods to maintain proper baseline ordering suggests that apparently reasonable masking strategies may still fail to provide reliable attribution evaluation.

4.5.4 Effect of perturbation strength

The effectiveness of adversarial perturbation masking depends critically on the perturbation strength parameter ϵ , which determines the maximum allowed change in pixel values. This parameter presents a fundamental trade-off: larger values increase the likelihood of changing model predictions but risk disrupting the underlying data distribution that we aim to analyze.

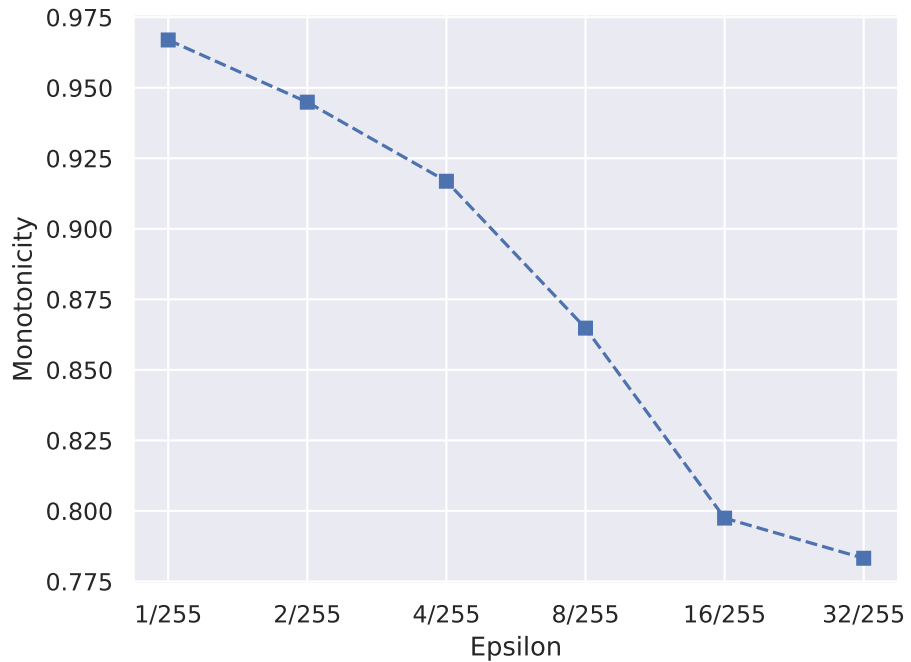


Figure 22: Analysis of scoring function monotonicity across different perturbation strengths. The vertical axis shows the monotonicity score, while the horizontal axis represents attack strength. Results demonstrate optimal behavior at $\epsilon = \frac{1}{255}$. Source: [NSK24a].

Our empirical investigation reveals that minimal perturbation strengths ($\epsilon = \frac{1}{255}$) provide the most reliable results for attribution evaluation. As illustrated in figure 22, at this strength level, our scoring function demonstrates

consistent monotonic behavior, indicating reliable progressive information removal.

To validate these findings, we compared two common adversarial attack methods: the Fast Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD). Both methods operate under the ℓ_∞ -norm constraint (see section 2.3.4). In our experiments across multiple datasets, PGD consistently improved the attack success rate (defined as the percentage of images where the model’s prediction changed) compared to FGSM. For instance, on the ChestX-ray8 dataset, the success rate increased from 95.80% with FGSM to 100% with PGD, while maintaining the same relative ranking in our scoring function.

However, stronger perturbations introduce undesirable artifacts. The ℓ_2 perturbations particularly compromise evaluation reliability by allowing non-uniform pixel modifications. Therefore, we advocate for using the weakest possible attack strength $\epsilon = \frac{1}{255}$ with FGSM as the default approach.

4.5.5 Attribution maps

Having validated our masking operator’s reliability, we now apply it to evaluate a broad range of attribution methods. We analyze both traditional CAM-based approaches that require upsampling and gradient-based methods that produce attributions at input resolution. Table 13 presents the evaluation scores averaged across all datasets and architectures.

Attribution Method	Perturb (ours) \uparrow
GradCAM [Sel+19]	0.477 \pm 0.08
GradCAM++ [Cha+18]	0.496 \pm 0.106
LayerCAM [Jia+21]	0.494 \pm 0.094
PolyCAMm [ECD22]	0.430 \pm 0.098
PolyCAMp [ECD22]	0.438 \pm 0.110
PolyCAMpm [ECD22]	0.440 \pm 0.102
ReciproCAM [BL23]	0.495 \pm 0.137
ScoreCAM [Wan+20b]	0.492 \pm 0.113
SmoothGradCAM++ [Ome+19]	0.441 \pm 0.017
BlurintegratedGradients [XVS20]	0.454 \pm 0.136
Gradients [SVZ14]	0.544 \pm 0.137
GuidedintegratedGradients [Kap+21]	0.452 \pm 0.138
IntegratedGradients [STY17]	0.488 \pm 0.135
SmoothGrad [Smi+17]	0.562 \pm 0.148
Canny [Can86]	0.315 \pm 0.072
Uniform	0.281 \pm 0.061

Table 13: Evaluation scores averaged across all datasets and architectures. The first group of attribution methods require upsampling, while the second group outputs an attribution map in input resolution. Our evaluation shows that SmoothGrad achieves the highest score, while the baseline methods (Canny and Uniform) score significantly lower as expected.

The results reveal several interesting patterns. Within the CAM family, ReciproCAM, GradCAM++, and LayerCAM show similar performance (scores around 0.49), while PolyCAM variants perform notably lower (0.43-0.44). Among gradient-based methods, SmoothGrad achieves the highest overall

score of 0.562, followed by vanilla Gradients at 0.544. The integrated gradient variants cluster together with scores between 0.45-0.49.

Importantly, all attribution methods maintain clear separation from our baseline tests – Canny edge detection (0.315) and uniform random attribution (0.281). This consistent margin validates that these methods extract meaningful features rather than just detecting edges or randomly selecting pixels.

In appendix figure 39, we demonstrate the practical value of our metric by using it to optimize SmoothGrad’s smoothing parameter σ on a medical pneumonia dataset.

4.6 VISUAL EXPLANATIONS FOR WOOD SPECIES CLASSIFICATION

After establishing SmoothGrad as the most reliable attribution method for understanding CNN decisions, we apply it to analyze our wood species classification network. Our objective is twofold: first, to generate attribution maps revealing the features that influence the model’s species identification decisions; second, to evaluate whether the model’s focus aligns with established wood anatomical features used by human experts.

For this analysis, we utilize our optimized ConvNeXt classification model (detailed in section 3.4.1). To establish a ground truth for comparison, expert wood anatomists annotated key anatomical features within a carefully curated subset of 270 samples from our dataset. These samples were specifically selected for their clear and distinctive anatomical characteristics.

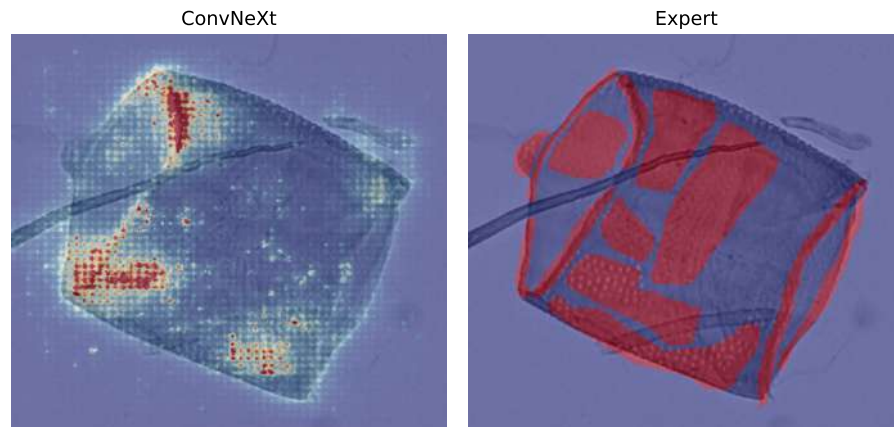


Figure 23: Comparison between SmoothGrad attribution map (left) and expert annotation (right) for an *Acacia* specimen. The contrast shows the model’s more localized focus compared to expert annotations.

The analysis of *Acacia* in figure 23 reveals a striking contrast between expert and model attention patterns. While experts comprehensively annotate entire anatomical structures, SmoothGrad highlights significantly fewer regions with a more localized focus. The attribution maps exhibit a certain degree of noise, which can be attributed to the backpropagation process inherent to SmoothGrad. As the method traces gradients backward through the network layers to identify influential input features, numerical instabilities and gradient accumulation can introduce noise patterns in the resulting attribution maps.

The *Eucalyptus* analysis in figure 24 reveals an intriguing divergence between model and expert feature utilization. While experts focus primarily on vessel structures, our model also highlights parenchyma cells (the object next to the vessel). This difference initially appears to contradict traditional wood

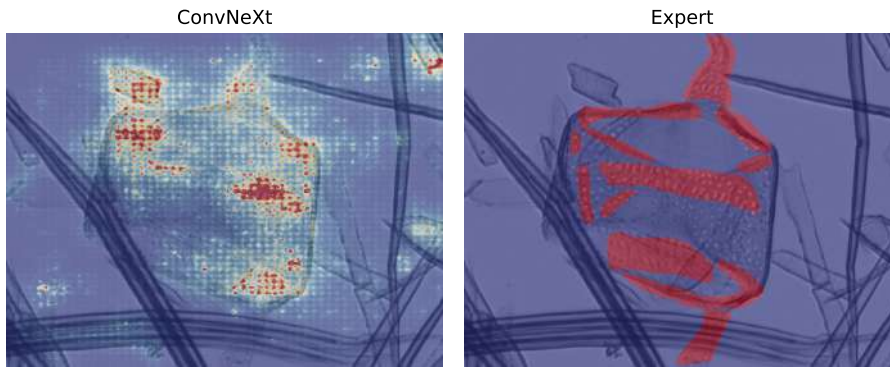


Figure 24: SmoothGrad attribution (left) and expert annotation (right) for *Eucalyptus*, showing distinct differences in feature recognition approaches.

anatomy practices, but expert consultation reveals a more nuanced reality: parenchyma cells do exhibit genus-specific variations, but these have been historically understudied. Wood anatomists have traditionally focused on vessels because they offer more readily distinguishable features and have been well-documented over the past century. While individual fibers and parenchyma cells possess distinctive characteristics, their features alone are typically insufficient for definitive species identification, especially given that wood samples usually contain mixed cellular compositions.

The model's attention to parenchyma cells might therefore indicate its ability to detect subtle, genus-specific variations that, while real, have remained largely unexplored in traditional wood anatomy. This suggests that machine learning approaches might help uncover new anatomically relevant features that complement established classification methods.

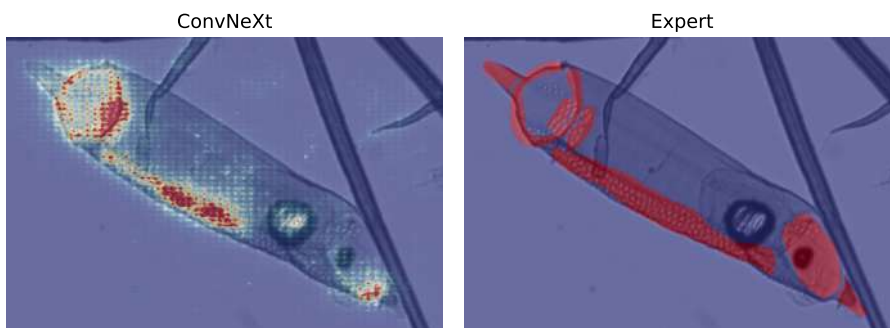


Figure 25: Attribution patterns for *Populus*, showing the contrast between model's localized focus (left) and expert's comprehensive annotation (right).

The *Populus* analysis in figure 25 further reinforces the observed patterns in model behavior. Expert annotations show comprehensive coverage of vessel structures, while the model's attention is concentrated on specific points along these structures. This fragmented attention pattern is consistent across all analyzed specimens, suggesting it is a characteristic feature of the model's classification approach rather than a specimen-specific phenomenon.

These qualitative analyses reveal important insights about our model's behavior. First, the model consistently relies on fewer, more localized features compared to the comprehensive approach of human experts. Second, the attribution maps are much noisier due to the backpropagation process. Finally, both experts and SmoothGrad tend to focus on the same areas, but sometimes

there are other features which experts do not consider. However, interpreting these differences requires careful consideration.

While experts provided comprehensive annotations of anatomical structures in our dataset, this documentation style likely represents a formalization of their knowledge rather than reflecting the actual recognition process. To understand whether these different patterns truly represent different approaches to recognition, we would need controlled experiments similar to those in human vision research, where the ability to identify species from partial anatomical features could be systematically tested. Without such experiments, we cannot conclude whether neural networks and experts process features differently merely from comparing attribution maps to documentation patterns.

Our findings do reveal three important insights about feature processing: First, neural networks can successfully classify species using localized regions rather than complete anatomical structures. Second, these regions largely overlap with expert-identified features, suggesting the model learns relevant anatomical patterns. Third, in some cases like the parenchyma cells in *Eucalyptus*, the model identifies potentially overlooked features that merit further investigation.

UNDERSTANDING NEURAL DECISIONS: A PATH-BASED FRAMEWORK

In the previous chapter, we explored attribution maps in depth and validated the effectiveness of adversarial perturbations as masking operations in attribution evaluation metrics. Building on this foundation, we now introduce a novel framework that moves beyond simple attribution maps to capture the complex, interdependent nature of neural network decision-making.

This chapter presents three main contributions that together advance our understanding of interpretability in neural networks:

1. Development of a systematic approach for identifying and analyzing decision paths across different neural architectures.
2. Introduction of Top-GAP [NSK24b], a novel architectural modification that constrains spatial information processing to improve network interpretability.
3. Evaluation of these methods both on our wood identification dataset and ImageNet, demonstrating how different architectural choices affect both model performance and interpretability.

5.1 DECISION PATHS

5.1.1 *Motivation*

Our analysis of attribution maps highlighted a fundamental limitation (see section 4.1.6 and section 4.4.2): they represent importance as a two-dimensional map of scalar values, where each spatial location receives an independent score. While these maps can highlight relevant image regions, they fail to capture how these regions interact in the network’s decision process.

Consider a neural network classifying bird species. An attribution map might assign high importance to the beak (0.9) and moderate importance to the wing pattern (0.6). This representation suggests we can evaluate each feature’s contribution independently. However, the network’s actual decision process is more nuanced – it might require seeing both the distinctive beak shape and wing pattern together to identify the species correctly. Neither feature alone would be sufficient, regardless of their individual attribution scores.

This interdependence extends beyond simple pairs. The network might correctly classify an image using either:

- The beak shape combined with tail pattern
- The wing pattern combined with overall body shape
- A combination of several moderately important features

Traditional attribution maps cannot represent these alternative pathways to correct classification, nor can they capture how features of varying importance scores might work together effectively.

To address these limitations, we need a framework that explicitly captures the combinatorial nature of neural network decision-making. Rather than asking “which pixels are important?”, we should ask “what combinations of regions are important?”.

5.1.2 Mathematical Definition

We introduce decision paths as specific combinations of image regions that collectively maintain the network’s original classification decision. Our analysis utilizes adversarial perturbation masking (section 4.4.5) as it provides a principled way to remove information while preserving natural image statistics.

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^C$ be a neural network classifier that maps inputs to class probabilities, where $f^{(c)}(x)$ represents the probability assigned to class c for input x . We define the grid coordinate space $I = \{(i, j) \mid 1 \leq i, j \leq n\}$ for an $n \times n$ grid overlaid on the input image.

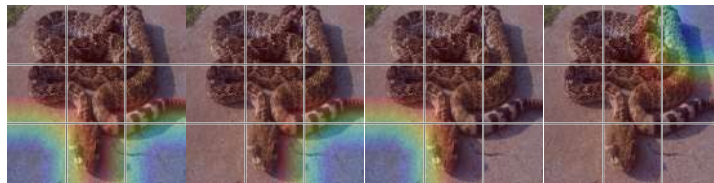
Definition 5.1.1 (Decision Path). For input image x , let M^{AP} be the adversarial perturbation masking operator. A decision path $P \subseteq I$ is defined as a subset of grid coordinates that, when revealed from an initially fully perturbed image, maintains the original classification:

$$\arg \max_c f^{(c)}(M_{I \setminus P}^{\text{AP}}(x)) = \arg \max_c f^{(c)}(x),$$

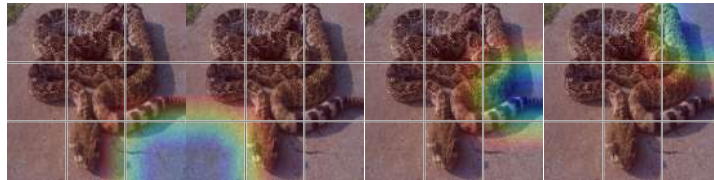
where $M_{I \setminus P}^{\text{AP}}$ indicates that all regions except those in P are adversarially perturbed.

The size of a decision path $|P|$ is simply the number of grid coordinates it contains. Of particular interest are minimal paths – those decision paths that maintain classification while using the fewest possible regions.

Figure 26 illustrates this concept, showing all decision paths discovered for two different neural architectures on the same input image. The visualizations use color overlays to indicate masked regions (blue) and visible regions (red), demonstrating how different architectures may utilize different combinations of regions to reach the same classification decision.



(a) ResNet-18: One decision path using 7 out of 9 patches, three paths using 8 patches



(b) EfficientNet-Bo: Four decision paths, each using 8 out of 9 patches

Figure 26: Visualization of all decision paths in a 3×3 grid for two architectures. Heat map overlays indicate regions that enable correct classification (blue = hidden by M^{AP} , red = shown).

In this specific example for ResNet-18, we can observe that the minimal decision path of size 7 (shown in the first visualization) is contained within the second and third 8-patch paths, but not in the fourth path which includes a different region in the corner as shown by the blue overlay.

Definition 5.1.2 (Frequency Map). For a set of decision paths S , the frequency map $F : I \rightarrow [0, 1]$ assigns to each grid coordinate i the fraction of decision paths that include it:

$$F(i) = \frac{|\{P \in S : i \in P\}|}{|S|}$$

A frequency of 1 indicates that a region appears in every decision path and is thus necessary for classification, while a frequency of 0 means the region never appears in any path.

Figure 27 illustrates a frequency map for a classification. The map reveals three central regions with frequency 1.0, indicating their presence in every decision path. The bottom-left (0.67) and bottom-right / top-right (0.78) cells show lower frequencies, corresponding to areas where the snake is less visible.

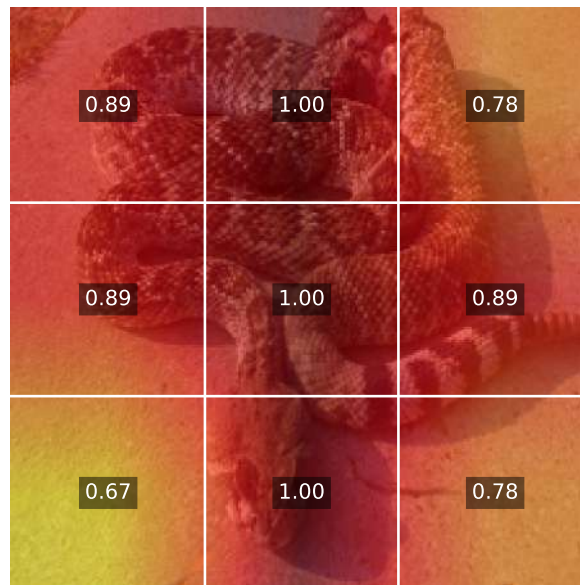


Figure 27: Frequency map showing regional participation in decision paths. Values indicate the fraction of decision paths incorporating each region (1.0 represents regions present in all paths).

5.1.3 Computation

Finding all decision paths requires evaluating every possible subset of grid coordinates, essentially exploring the power set of the grid. While one might consider approximating this search through random sampling, exhaustive enumeration offers crucial advantages. It guarantees discovery of minimal paths and enables precise analysis of how regions work together to maintain classification.

Our algorithm (algorithm 5.1) employs a systematic approach: starting from an adversarially perturbed version of the input image, we iteratively reveal original patches and test if the network's classification is restored.

Algorithm 5.1 Find Decision Paths

Require: Neural network f , original image x , grid size n , perturbation strength ϵ

Ensure: Set of decision paths S

```

1:  $y_{\text{orig}} \leftarrow \arg \max_c f^{(c)}(x)$  ▷ Original classification
2:  $x_{\text{adv}} \leftarrow \text{FGSM}(f, x, y_{\text{orig}}, \epsilon)$  ▷ Create targeted adversarial example
3:  $S \leftarrow \emptyset$  ▷ Initialize empty set of decision paths
4: for  $k \leftarrow 1$  to  $n^2$  do ▷ Iterate through region counts
5:   for  $P \in \text{combinations}(n^2, k)$  do ▷ All k-sized combinations
6:      $x_{\text{masked}} \leftarrow x_{\text{adv}}$  ▷ Start with adversarial image
7:     for  $(i, j) \in P$  do ▷ Reveal original patches at selected positions
8:        $h \leftarrow \text{height}/n$ 
9:        $w \leftarrow \text{width}/n$ 
10:       $x_{\text{masked}}[ih : ih+h, jw : jw+w] \leftarrow x[ih : ih+h, jw : jw+w]$ 
11:    end for
12:    if  $\arg \max_c f^{(c)}(x_{\text{masked}}) = y_{\text{orig}}$  then ▷ Correct prediction?
13:       $S \leftarrow S \cup P$  ▷ Add new decision path
14:    end if
15:  end for
16: end for
17: return  $S$ 

```

The algorithm operates by progressively testing combinations of increasing size. For each size k from 1 to n^2 , we evaluate all possible k -combinations of grid positions. When a combination restores the original classification, we've identified a decision path. The masking operation works by replacing regions of the original image with their adversarially perturbed counterparts.

The computational complexity of this search is $O(2^{n^2})$ where n is the grid size, as we must evaluate all possible subsets. For each subset, we perform a forward pass through the network, making the total complexity $O(2^{n^2} \cdot T_f)$ where T_f is the time for a single forward pass.

This exponential complexity necessitates careful choice of grid resolution. Attribution methods like GradCAM typically produce 7×7 attribution maps based on the last convolutional layer's feature map size. Exhaustively searching such a grid would require evaluating 2^{49} possible combinations, making the computation intractable. Other methods like SmoothGrad can produce attribution maps at input resolution (224×224), which would make exhaustive search even more challenging.

For practical analysis, we overlay a 3×3 grid on our images, resulting in a manageable search space of 512 possibilities while still capturing essential spatial relationships in the network's decision-making process.

5.1.4 Path Distribution

Having defined decision paths, we now examine their general distribution. Through analyzing 500 randomly sampled images from the ImageNet validation set, we observed several distinctive patterns in how paths are distributed across different path sizes. Figure 28 presents these distributions for ResNet-18 and ConvNext-tiny.

Most strikingly, while the probability of finding decision paths increases monotonically with path size, this relationship exhibits a characteristic nonlin-

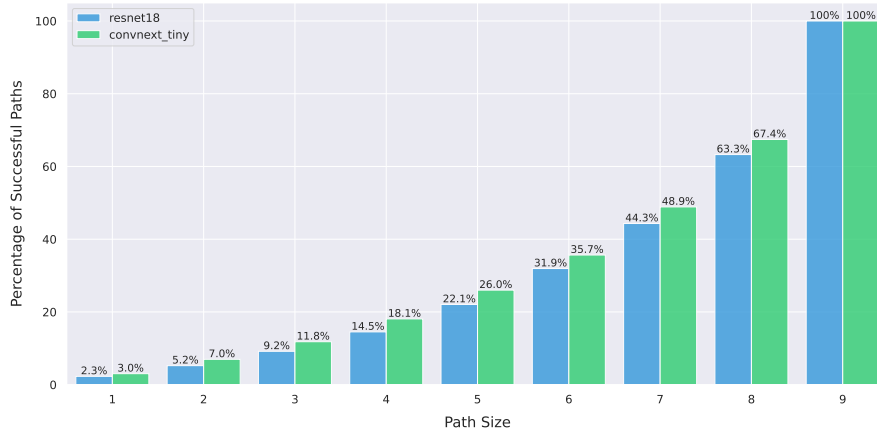


Figure 28: Distribution of decision paths by size for ResNet-18 and ConvNext-tiny architectures. For each path size k , the success rate represents the fraction of decision paths (those maintaining original classification) among all possible $\binom{9}{k}$ combinations of k regions.

ear progression. For small paths (1-3 regions), decision paths are remarkably rare, with success rates below 12% for both architectures. This scarcity of small paths indicates that neither architecture can reliably maintain classification based on isolated regions of the original content.

As path size increases, we observe a distinctive acceleration in success rates, particularly between 5-8 regions. This transition marks a critical point where networks gain access to sufficient contextual information to reconstruct their original classifications. The gradual nature of this increase, rather than a sharp step function, suggests that these architectures progressively integrate information from multiple regions. The success rate reaches 100% at path size 9, representing the trivial case where no regions are masked.

Notably, ConvNext-tiny consistently demonstrates higher success rates across this range compared to ResNet-18, indicating it maintains more robust feature representations even with partial information.

The smooth progression in success rates reveals that these architectures rely on distributed evidence patterns rather than critical individual features. This observation parallels our findings about confidence degradation under dropout (figure 20), where we saw that neural networks exhibit gradual degradation when losing information rather than catastrophic failures that would indicate reliance on specific critical features.

5.1.5 Relationship to Deletion and Insertion Metrics

The path-based framework provides valuable insights into traditional attribution evaluation metrics, particularly the insertion and deletion metric. An important conceptual point is that a decision path P consists of all regions that are not hidden (masked) in the image. This means when we talk about “adding a region to P ”, we are revealing more and more of the image.

Let’s examine how insertion metrics relate to decision paths through a concrete example of classifying a bird. Consider a 3×3 grid with attribution scores and corresponding bird features:

$$\begin{bmatrix} 0.9 \text{ (head)} & 0.8 \text{ (beak)} & 0.3 \text{ (background)} \\ 0.7 \text{ (breast)} & 0.6 \text{ (wing)} & 0.2 \text{ (tail)} \\ 0.5 \text{ (feet)} & 0.4 \text{ (branch)} & 0.1 \text{ (background)} \end{bmatrix}$$

The insertion metric follows a fixed trajectory determined by attribution scores:

1. Initially, all regions are masked, so $P = \emptyset$ (path size 0)
Network confidence for “bird”: 0.01
2. Reveal highest-scored region (0.9, head) $\rightarrow P_1 = \{(1, 1)\}$
Network confidence: 0.15
3. Add second-highest region (0.8, beak) $\rightarrow P_2 = \{(1, 1), (1, 2)\}$
Network confidence: 0.45
4. Add third-highest region (0.7, breast) $\rightarrow P_3 = \{(1, 1), (1, 2), (2, 1)\}$
Network confidence: 0.92 (correct classification achieved)
5. Continue until P_9 contains all regions (confidence: 0.98)

The key difference between insertion metrics and our decision path framework lies in how they explore possible region combinations. The insertion metric follows a single predetermined path through the space of region combinations, dictated by attribution scores. It begins with no regions and progressively adds them in order of their attribution scores until all regions are included.

While this process might discover a decision path along the way (like when it achieves correct classification at step 3 in our example), it cannot explore alternative combinations that might maintain classification with fewer regions.

Our path distribution analysis (figure 28) provides crucial context for interpreting these metrics. Any sequence of region additions will eventually yield a successful decision path at size 9 when the complete image is revealed.

A high-quality attribution map should therefore guide us to successful decision paths early in the sequence, before reaching the path sizes where random combinations typically succeed. Once a decision path is found (as at step 3 in our example with confidence 0.92), the addition of further regions becomes irrelevant.

This observation provides theoretical justification for why insertion metrics are effective: they inherently reward attribution maps that achieve high confidence with smaller path sizes. The AUC calculation gives more weight to early confidence gains, aligning with our observation that finding decision paths early (before the probability of random success increases) is a key indicator of attribution map quality.

5.2 TOP-GAP: CONSTRAINING NEURAL DECISIONS

5.2.1 *Motivation*

Our analysis of decision paths reveals two fundamental issues in neural decision-making. First, networks develop numerous redundant pathways to classification. For a 3×3 grid, we have $2^9 = 512$ possible combinations of regions. Looking at figure 28, for path size 5, we observe approximately 22% success rate. Since there are $\binom{9}{5} = 126$ possible combinations of 5 regions, this

means around 28 different combinations maintain the original classification just at this path size alone.

The second issue is that these decision paths can be very long. As shown in figure 26, both ResNet and EfficientNet require sometimes nearly complete images (8 out of 9 patches) to maintain classification, making it difficult to identify which regions are truly essential for the network’s decision.

From a theoretical perspective, interpretability would be maximized when a model has a single decision path that requires viewing only a minimal portion of the image. Consider the extreme case: a model with just one decision path that only needs to look at one grid cell to make its classification. In this scenario, we would have perfect interpretability since we could definitively say which part of the image drives the model’s decision.

Beyond decision paths, current architectures also present challenges for attribution methods. Since gradients can flow through all spatial locations during backpropagation, attribution maps often highlight regions that weren’t actually critical for the network’s decision. This leads to noisy attributions that may not reflect the network’s true decision-making process.

This stark contrast between current architectures and the theoretical ideal motivates our development of Top-GAP, a novel architectural approach that constrains how networks process spatial information.

5.2.2 Method

In standard convolutional networks, the global average pooling (GAP) layer treats all spatial locations as equally important contributors to the final decision (see section 2.2.2 for an overview on different layers). This architectural choice implicitly encourages networks to develop multiple redundant decision paths, as observed in our path analysis. By modifying this fundamental component, we can guide networks toward more focused and interpretable decision-making processes.

Top-GAP introduces spatial constraints through selective pooling operations. Instead of averaging all spatial locations, the network selects and utilizes only the most informative regions:

Definition 5.2.1 (Top-k Pooling). Given a feature map $X \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels, and $H \times W$ are the spatial dimensions, the Mean-Top-k Pooling operation $M_k : \mathbb{R}^{C \times H \times W} \rightarrow \mathbb{R}^C$ is defined as:

$$M_k(X)_c = \frac{1}{k} \sum_{i=1}^k X_{c, \text{Top}_k(X_c)[i]},$$

where $X_c \in \mathbb{R}^{H \times W}$ represents channel c ’s feature map flattened to a vector, and $\text{Top}_k(X_c)$ returns indices of the k largest values in X_c .

However, simply replacing GAP with Top-k pooling is insufficient. The network must learn which spatial locations are truly important, not just which ones have the highest activation values. This requires two additional architectural components: multi-scale feature integration and sparsity regularization.

Multi-scale feature integration is implemented through a Feature Pyramid Network (FPN), which combines information across different spatial resolutions before the Top-k pooling operation (refer to figure 29). This ensures the network can identify important features regardless of their scale. The FPN outputs feed into a final convolutional layer that produces class-specific activation maps.

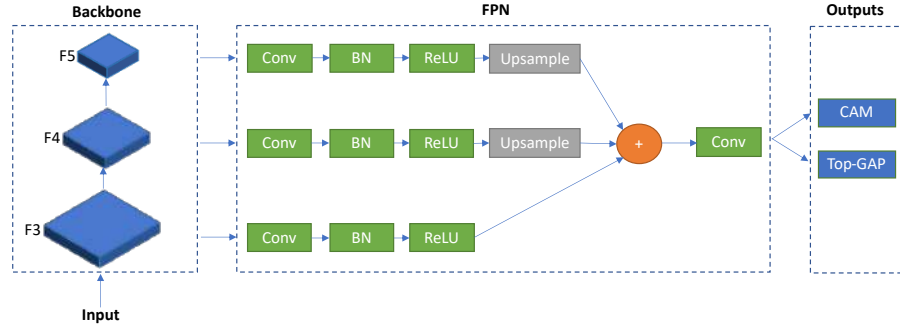


Figure 29: Example of our architecture applied to a backbone with 3 feature maps (e.g. 7×7 , 14×14 , 28×28). For all convolutions except the final one, a kernel size of 3 and 256 filters is used. The last convolution employs a kernel size of 1, with the number of filters set to match the number of output classes. The CAM is as large as the biggest feature map (here F3). Our pooling layer (“Top-GAP”) averages the CAMs given by the last convolutional layer (“Conv”) to create a vector containing the probability for each class. For the CAM, we disable “Top-GAP” and perform min-max scaling.

Sparsity regularization addresses a fundamental challenge in training: without constraints, the network might arbitrarily choose which k locations to use, leading to unstable training. We add L1 regularization to the loss function:

$$L_{\text{total}} = L_{\text{CE}} + \lambda \|X\|_1$$

This creates an adversarial dynamic during training. The cross-entropy loss L_{CE} encourages high activations for correct classification, while the L1 term penalizes all activations. The network must learn to maintain high activations only for truly discriminative features. The parameter λ controls this trade-off, with $\lambda = 1$ typically providing sufficient regularization. In section 2.4.4, we explored the relationship between accuracy and model interpretability.

The combination of these components – Top-k pooling, FPN, and L1 regularization – is crucial. The FPN enables robust feature detection across scales, Top-k pooling enforces spatial selectivity, and L1 regularization ensures stability and sparsity. Each component addresses a specific aspect of the interpretability challenge, and removing any one of them significantly degrades performance.

For completeness, we explicitly state the two operational modes shown in figure 29:

1. Prediction mode (Top-GAP): Top-k pooling is applied to generate class probabilities
2. Visualization mode (CAM): The pre-pooling activations are used to produce class activation maps

These modes arise naturally from our architecture’s use of a 1×1 convolutional layer, making it structurally similar to the original CAM method. Due to this architecture, CAM and GradCAM would produce equivalent visualizations in our network, as both methods would identify the same k spatial locations that were selected during the forward pass. This equivalence occurs because gradients can only flow through these k selected locations during backpropagation.

However, our key contribution lies in constraining the network’s behavior through Top-k pooling and L1 regularization, rather than in the visualization method itself. This architectural constraint benefits any attribution method, as gradients are naturally focused on only the k most relevant spatial locations used in the forward pass. We show additional results in appendix A.2, where we focus on CAM instead of SmoothGrad to prove this.

The effectiveness of Top-GAP depends on choosing an appropriate value for k . A small k forces the network to be highly selective but might prevent it from capturing sufficient context for accurate classification. Conversely, a large k reduces the constraining effect. We explore this trade-off empirically in the evaluation section.

5.3 EVALUATION

5.3.1 Analysis of Decision Paths in Neural Networks

To understand how different architectures process spatial information and how this relates to model performance and interpretability, we begin with a large-scale analysis of neural networks.

Our analysis of 644 models from the timm collection [Wig19] reveals a fundamental relationship in neural networks. Better-performing models tend to maintain correct classification with smaller regions of the input image, demonstrated by a significant negative correlation ($r = -0.5542$, $R^2 = 0.307$) between minimal path size and ImageNet accuracy. For instance, while lower-performing models might need to see most of an image to identify the class, high-performing models can often make correct classifications from just small regions.

As shown in table 14, this relationship holds consistently across different ImageNet variants, with correlations ranging from $r = -0.59$ for ImageNet-Rendition Clean to $r = -0.40$ for ImageNet-Adversarial. Notably, minimal path size proves to be a better predictor of model accuracy than the number of parameters.

Dataset	Number of Paths	Minimal Path Size	Parameter Count
IN	0.40	-0.55	0.38
IN-Real	0.43	-0.58	0.32
IN-R	0.47	-0.57	0.55
IN-R Clean	0.43	-0.59	0.34
IN-A	0.26	-0.40	0.56
IN-A Clean	0.42	-0.57	0.34
IN-v2	0.41	-0.57	0.41
IN-Sketch	0.44	-0.55	0.53
Average	0.41	-0.55	0.43

Table 14: Correlation between accuracy and Number of Paths / Minimal Path Size / Parameter Count across 644 models. Datasets: IN = ImageNet [Rus+15]; IN-Real = ImageNet-Real [Bey+20]; IN-R = ImageNet-Rendition [Hen+21a]; IN-A = ImageNet-Adversarial [Hen+21b]; IN-v2 = ImageNet-v2 [Rec+19]; IN-Sketch = ImageNet-Sketch [Wan+19b].

Given these findings, one might conclude that the optimal approach would be to design neural networks that deliberately focus on small regions rather than considering the entire image.

However, this insight comes with an interesting trade-off: we discovered a strong negative correlation ($r = -0.8983$) between minimal path size and the number of decision paths. Networks that achieve smaller minimal paths tend to develop more alternative pathways to reach their decisions. This makes intuitive sense: when each decision path uses less information (smaller minimal path size), the network needs more such paths to capture the full complexity of the visual task.

This trade-off becomes visually evident when examining the unnormalized frequency maps of VGG11 and ConvNeXt-large (figure 30). While our earlier definition of frequency maps involved normalization (see definition 5.1.2), here we present the raw counts. This unnormalized perspective is particularly valuable, as it reveals not only how many decision paths utilize each region but also how the total number of paths varies between architectures.

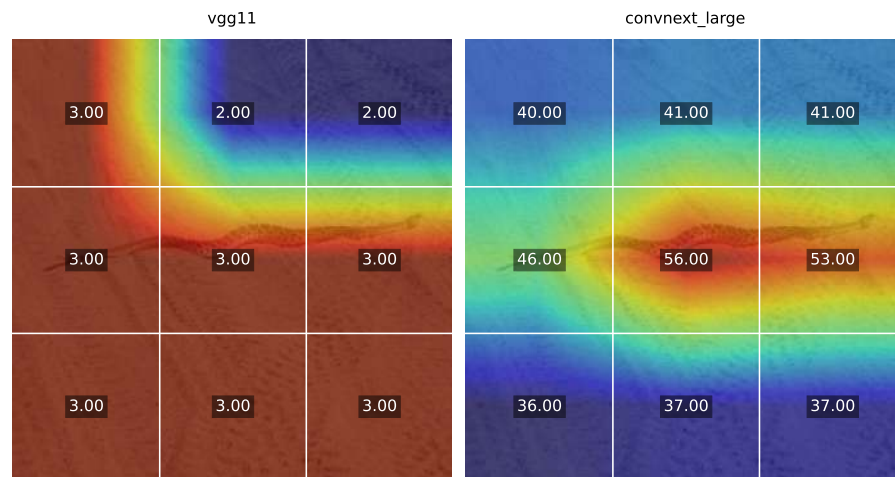


Figure 30: Unnormalized frequency map showing regional participation in decision paths for VGG11 (left) and ConvNeXt-large (right). Values indicate the number of decision paths incorporating each region.

VGG11’s frequency map reveals a highly uniform pattern: almost all grid cells participate in exactly 3 decision paths, with only two cells participating in 2 paths. This uniformity suggests that the network requires nearly the entire image to make decisions, as almost every region is used in every available path.

In contrast, ConvNeXt-large shows a clear distinction between regions that correspond to the object and those that contain background. The middle region (snake) participates in 53/56 different decision paths, while the surrounding regions appear in only 36 to 41 paths.

This pattern aligns with the correlations shown in the table: ConvNeXt-large achieves high accuracy (86% on ImageNet), which corresponds to a low minimal path size and a high number of paths. Conversely, VGG11, with lower accuracy (70.4% on ImageNet), exhibits a larger minimal path size but has a lower overall number of paths. This supports the broader trend observed across 644 models, where accuracy is negatively correlated with minimal path size (average correlation: -0.55) and positively correlated with the number of paths (average correlation: 0.41).

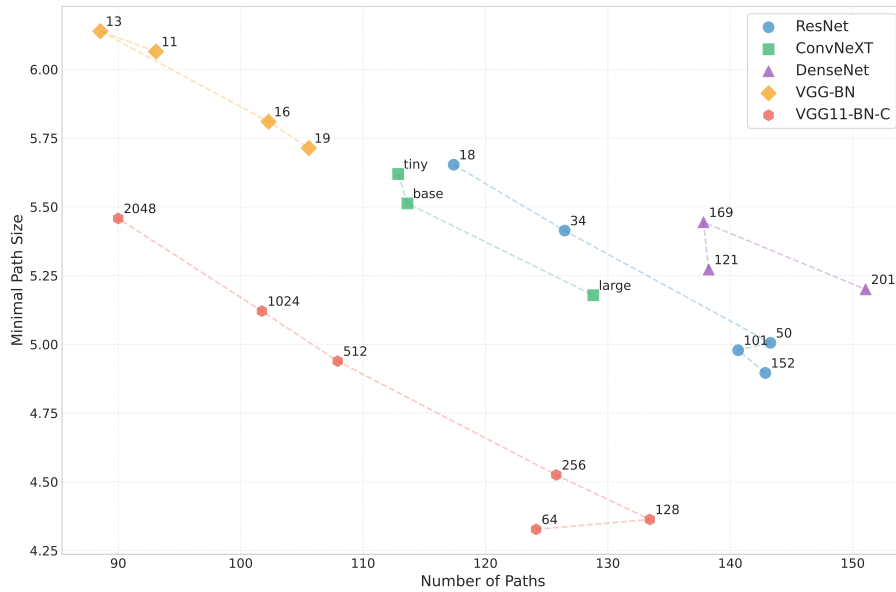


Figure 31: Analysis of minimal path size versus number of paths across architectures. The y-axis shows the minimal regions (out of 9 in a 3×3 grid) needed for classification. Standard architectures all require many regions (5.2-6.0) while maintaining numerous paths (90-150). Top-GAP using VGG11-bn (VGG11-BN-C) enables systematic control over both measures as k varies from 2048 to 64.

To further explore how these patterns vary across architectures, we analyze minimal path size and number of paths across model families (figure 31).

VGG-BN models require nearly 6 out of 9 possible regions while maintaining on average 90-110 different decision paths. DenseNet achieves smaller minimal path sizes (around 5.2-5.4 regions) but develops more paths (135-150). ConvNeXT models strike a middle ground, utilizing 5.2-5.6 regions with 115-130 paths.

Furthermore, the plot reveals how neural networks naturally control spatial processing through model scaling. For instance, DenseNet shows a progression from DenseNet-121 to DenseNet-201, with larger models tending to use less regions but maintaining more paths. Similarly, ResNet (from ResNet-18 to ResNet-152) and ConvNeXT (tiny to large) demonstrate how architectural scaling influences the balance between minimal path size and number of paths.

Top-GAP enables direct control of this balance through its k parameter, unlike traditional architectures that rely on increasing parameter size to improve model behavior.

The value of k directly determines how many spatial locations the network can utilize, providing explicit control over the trade-off between path size and number of paths. Setting $k = 512$ achieves an optimal balance: the model uses only 5.0 regions while maintaining 110 paths, placing it closer to the lower-left corner of figure 31 than traditional architectures. Lower k values (256, 128, 64) further reduce the regions needed but at the cost of increasing decision paths to 120-130.

This trade-off reflects a fundamental limitation in neural network interpretability: it is impossible to minimize both path size and number of paths simultaneously. A perfectly interpretable model would theoretically use a single, minimal path to make its decisions. However, the visual complexity of

ImageNet-scale tasks necessitates both sufficient spatial coverage (path size) and alternative decision strategies (number of paths) for robust classification.

5.3.2 Attribution maps and Frequency maps

Having established the effect of different architectures on decision paths, we now examine how attribution maps relate to our path-based analysis. For this, we compute the Spearman correlation coefficient between the path frequency map and attribution maps, normalized by a random baseline to ensure comparability across architectures.

Architecture	Variant	Correlation coefficient \uparrow
ConvNeXT-T	FB	0.362
	FB22k	0.323
	HNF	0.329
	IN12k	0.350
DenseNet-121	TV	0.455
	RA	0.445
EfficientNet-Bo	RA	0.486
	TF-AA	0.461
	TF-NS	0.438
	TF	0.471
ResNet-18	A1	0.515
	A2	0.527
	RA2	0.477
	FB-SSL	0.516
	FB-SWSL	0.522
	Gluon	0.476
	SK-RA	0.488
TV	0.482	
VGG-11	TV	0.466
	TV-BN	0.506
	TV-BN (k=64)	0.473
	TV-BN (k=128)	0.510
	TV-BN (k=256)	0.537
	TV-BN (k=512)	0.580
	TV-BN (k=1024)	0.601
TV-BN (k=2048)	0.573	

Table 15: Correlation coefficient between frequency map and attribution maps across different architectures and variants (k variants indicate our Top-GAP method). Results are averaged over 700 ImageNet validation images and four attribution methods.

We evaluated these metrics across 700 images from the ImageNet validation dataset, averaging results across four attribution methods: ReciproCAM, two variants of SmoothGrad (with $\sigma = 0.1$ and $\sigma = 0.01$), and VanillaGradients. The results in table 15 reveal several interesting patterns.

Traditional architectures like ResNet-18 show relatively high correlations (0.476-0.527) across different variants, suggesting consistent alignment between attribution maps and frequency maps. Interestingly, ConvNeXT-tiny

exhibits lower correlations (0.323-0.362) despite having lower number of paths than ResNet-18 and slightly lower path size, as shown in figure 31.

This apparent discrepancy highlights that attribution maps and decision paths provide complementary views of the network’s decision process. The lower correlation might also be influenced by ConvNeXT’s sensitivity to different SmoothGrad noise levels used in our evaluation.

EfficientNet-B0 variants show moderate correlations (0.438-0.486), while DenseNet variants cluster around 0.45. The consistency within architectural families suggests that the relationship between attribution maps and frequency maps is primarily determined by fundamental architectural design choices.

Most notably, Top-GAP with VGG11-bn demonstrates a clear pattern as k varies across different scales. The correlation begins at a moderate level with $k = 64$ (0.473) and increases steadily until reaching its peak at $k = 1024$ (0.601), before declining again at $k = 2048$ (0.573). This peak achieves the highest correlation among all tested architectures.

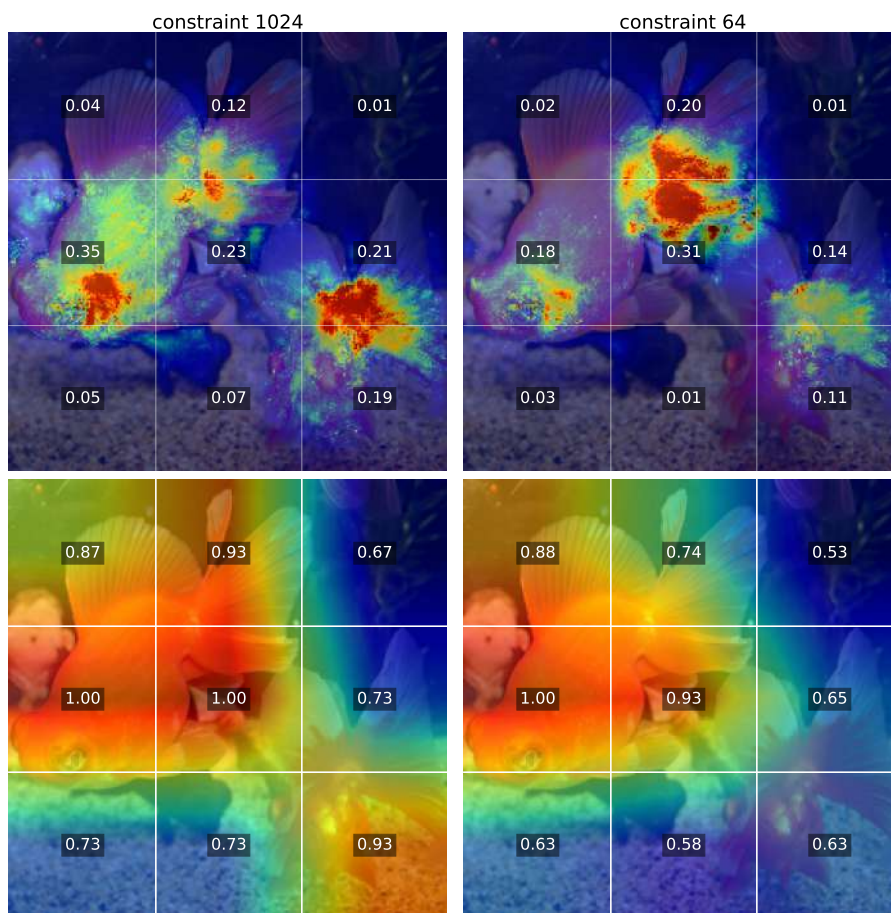


Figure 32: SmoothGrad attribution maps (top) and frequency maps (bottom) under Top-GAP constraints of 1024 (left) and 64 (right). In the attribution maps, the stricter constraint (64) yields sparser patterns with 0.11 versus 0.14 values per grid cell. The frequency maps similarly show more focused highlighting under constraint 64.

To understand this difference, we performed a visual analysis comparing attribution and frequency maps under two different Top-GAP constraints. Figure 32 shows this comparison for $k = 64$ and $k = 1024$ using a single ImageNet validation image, with SmoothGrad attribution maps in the top

row and corresponding frequency maps below. For meaningful comparison with our 3×3 grid-based decision paths, we averaged the SmoothGrad values within each grid cell.

Unlike figure 30, we show in the plots here the normalized frequency map values, which better reveal the regions the network focuses on. These normalized values do not show the number of paths.

With constraint $k = 1024$, the model develops 15 different ways (decision paths) to classify the image. In the frequency map, a value of 1.0 for a grid cell means that this cell is used in all 15 decision paths. We observe two such cells (bottom left): one corresponding to the fish’s head and one to its tail fin / body. The SmoothGrad map (top left) shows on average 0.14 values per grid cell.

When we reduce the constraint to $k = 64$, we force the network to use fewer regions in each decision path. To compensate for this restriction, the network develops more alternative paths to still achieve successful classification. The network now has 57 decision paths.

The frequency map shows only one grid cell (the fish’s head) with a value of 1.0 (bottom right), meaning this cell appears in all 57 paths. The SmoothGrad map shows sparser patterns with on average 0.11 values per grid cell, and the frequency map similarly shows more focused highlighting.

This analysis reveals the same trade-off we observed earlier in figure 31: stricter constraints ($k=64$) force networks to use fewer, more focused regions per decision, while developing more alternative pathways to achieve successful classification.

5.3.3 Accuracy and Robustness

Having analyzed how Top-GAP enables control over decision paths, we now examine how these architectural constraints affect model performance. Table 16 compares the accuracy of standard VGG-11 architectures against Top-GAP variants across different constraint values ($k = 64, 128, \dots, 2048$).

Model Name	Variant	Number of Paths	Minimal Path Size	Accuracy \uparrow
VGG-11	TV	112.26 \pm 155.03	5.88 \pm 2.45	69.0
	TV-BN	93.09 \pm 130.13	6.07 \pm 2.31	70.4
VGG-11-Con	64	124.14 \pm 137.41	4.33 \pm 2.25	68.7
	128	133.42 \pm 145.16	4.36 \pm 2.28	69.7
	256	125.79 \pm 140.85	4.53 \pm 2.30	70.0
	512	107.93 \pm 134.78	4.94 \pm 2.32	69.6
	1024	101.74 \pm 132.11	5.12 \pm 2.29	68.7
	2048	90.00 \pm 126.25	5.46 \pm 2.36	66.7

Table 16: Comparison of decision path characteristics across different neural architectures. Standard deviations reflect variation across 500 randomly sampled ImageNet validation images. Accuracy refers to all images in the ImageNet validation dataset.

The comparison between VGG-11 variants reveals interesting effects of batch normalization: it increases minimal path size (5.88 \rightarrow 6.07), decreases the number of paths (112.26 \rightarrow 93.09), and improves accuracy (69.0% \rightarrow 70.4%). This illustrates how architectural components like batch normalization can significantly influence a network’s decision-making patterns.

In the Top-GAP variants, we observe a clear manifestation of the negative correlation ($r=-0.55$) between minimal path size and accuracy for $k \geq 256$. As k increases from 256 to 2048, minimal path size grows steadily ($4.53 \rightarrow 5.46$) with a corresponding decline in accuracy ($70.0\% \rightarrow 66.7\%$). Notably, at $k = 256$, Top-GAP nearly matches the accuracy of VGG-11-BN (70.0% vs 70.4%) while requiring significantly fewer regions (4.53 vs 6.07).

While Top-GAP shows slightly lower performance on ImageNet compared to the baselines (e.g., 70.0% vs 70.4% for VGG11-bn), its benefits become apparent when examining specialized datasets. We compared the following datasets:

- COCO [Lin+15b]: We turned this segmentation dataset into a classification dataset by excluding images with more than one object. Furthermore, we kept only classes with a minimum of 20 samples per class. The resulting subset comprises 53 classes.
- Wood identification dataset: This dataset was introduced in section 3.1.4.
- Oxford-IIIT Pet Dataset [Par+12]: The task is to differentiate among 37 breeds of dogs and cats.
- CUB-200-2011 [Wah+11]: 200 classes of birds have to be distinguished.

The results are shown in table 17. We trained all the networks with 5-fold cross validation and averaged the values.

Dataset	Arch	Accuracy \uparrow	Accuracy (ours) \uparrow
COCO [Lin+15b]	EN	0.801 ± 0.009	0.803 ± 0.006
	CN	0.939 ± 0.006	0.940 ± 0.005
	RN	0.853 ± 0.004	0.868 ± 0.005
Wood	EN	0.672 ± 0.037	0.681 ± 0.041
	CN	0.721 ± 0.030	0.724 ± 0.033
Oxford [Par+12]	EN	0.854 ± 0.008	0.863 ± 0.010
	RN	0.861 ± 0.007	0.862 ± 0.007
CUB [Wah+11]	EN	0.76 ± 0.01	0.77 ± 0.005
	RN	0.69 ± 0.014	0.685 ± 0.006
	CN	0.862 ± 0.007	0.854 ± 0.005
Average		0.801	0.805

Table 17: Performance comparison between baseline models and Top-GAP variants across multiple datasets. EN = EfficientNet-Bo, CN = ConvNeXt-tiny, RN = ResNet-18, VG = VGG11-bn.

For almost all datasets and architectures, we see small improvements in classification performance. On average, the accuracy increases by around 0.4%.

Importantly, across all these specialized datasets, optimal performance was achieved with constraints $k \geq 256$, never with smaller values like $k=64$. This aligns with our ImageNet findings about the importance of maintaining sufficient spatial flexibility while constraining attention.

Beyond specialized datasets, we investigated whether Top-GAP’s spatial constraints enhance model robustness to distribution shifts. Table 18 presents results across three scenarios: CUB to Waterbirds (background shifts) [Sag+20],

ImageNet to Sketch (style shifts) [Wan+19b], and ImageNet to ImageNet-C (corruption shifts) [HD19].

Dataset	Arch	Acc \uparrow	Acc \uparrow (ours)
CUB \rightarrow Waterbirds	EN	0.521	0.564
	CN	0.722	0.737
	RN	0.468	0.520
ImageNet \rightarrow Sketch	VG	0.179	0.200
	RN	0.206	0.236
ImageNet \rightarrow ImageNet-C	VG	0.494	0.498
	RN	0.513	0.535
Average	-	0.443	0.470

Table 18: Evaluation of the out-of-distribution accuracy by using images outside the original dataset. $X \rightarrow Y$ means train on X and validate on Y .

The results demonstrate consistent improvements across all distribution shift scenarios. On the Waterbirds dataset, where background changes test the model’s ability to focus on relevant object features, Top-GAP shows substantial gains across all architectures. Similar improvements appear on ImageNet-Sketch, where ResNet-18 achieves a 3 percentage point gain (20.6% to 23.6%), and on ImageNet-C, where accuracy improves by up to 2.2 percentage points.

5.3.4 Spatial Focus and Receptive Field

Our analysis has shown that Top-GAP with strong constraints (low k) leads to networks with many decision paths but each path uses very few regions. While small paths are desirable for interpretability since they pinpoint specific important regions, having many alternative paths makes it harder to understand all possible decision strategies. For instance, with $k=64$, we observed 124 different decision paths compared to just 90 paths with $k=2048$ (table 16).

However, we also found that networks with strong constraints show improved accuracy on specialized datasets and better robustness to distribution shifts (table 18). This suggests that despite having more paths, these networks might be focusing on relevant objects rather than background features.

While the theoretical receptive field (see section 2.2.3) tells us how large of an input region can influence each output position, research has shown that in practice, not all pixels within this region have equal influence [Luo+16]. To understand how our architectural constraints affect spatial processing patterns, we need to examine the effective receptive field (ERF), which measures how strongly each input pixel actually influences the network’s computations.

We calculate this influence through partial derivatives: $\frac{\partial X_{i^{(n)},j^{(n)}}^{(n)}}{\partial X_{i^{(1)},j^{(1)}}^{(1)}}$, which captures how much a change in input pixel $(i^{(1)},j^{(1)})$ affects the output at position $(i^{(n)},j^{(n)})$.

To visualize the ERF, we place a gradient signal of 1 at specific locations in the final feature map and backpropagate to the input. Figure 33 shows this for both a standard ResNet-18 and one trained with Top-GAP.

To quantify these differences, we introduce the ERF distance metric:

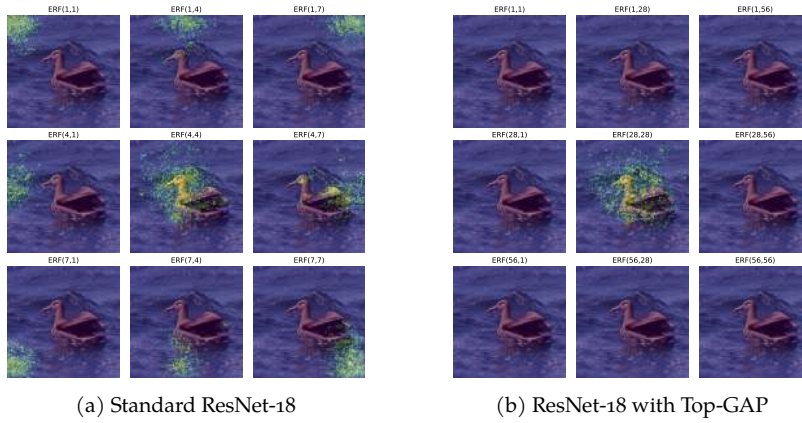


Figure 33: ERF for various locations in the output feature map. While the standard model shows uniform influence across positions, Top-GAP exhibits stronger central influence, suggesting more focused processing of object regions versus background.

Definition 5.3.1 (ERF distance [NSK24b]). Let $\text{ERF}(1, 1)$ be the average absolute gradient from all input pixels to the output corner position $(1, 1)$:

$$\text{ERF}(1, 1) = \frac{1}{hw} \sum_{i,j} \left| \frac{\partial X_{1,1}^{(n)}}{\partial X_{i,j}^{(1)}} \right|$$

Similarly, let $\text{ERF}(h/2, w/2)$ be the average absolute gradient to the output center position, where h and w are the height and width of the output feature map. The ERF distance is then defined as: $\text{ERF}(h/2, w/2) - \text{ERF}(1, 1)$.

This metric captures how uniformly the network processes different spatial regions. A high ERF distance indicates stronger central influence – the network focuses more on central regions where objects typically appear. A low or negative distance suggests uniform influence across positions, indicating less focused spatial processing.

Dataset	Arch	ERF distance \uparrow	ERF distance (ours) \uparrow
COCO [Lin+15b]	EN	0.108	0.447
	CN	0.072	0.288
	RN	0.273	0.399
Oxford [Par+12]	EN	0.013	0.383
	RN	0.060	0.443
CUB-200-2011 [Wah+11]	EN	-0.033	0.480
	CN	-0.034	0.242
	RN	0.092	0.529

Table 19: The table shows that our approach leads to a different ERF. The center has a stronger effect than the corner of the image. “Ours” is our approach (with pixel constraint, ℓ_1 loss and the changes to the model). The other column is the standard model without any changes. EN = EfficientNet-Bo, CN = ConvNeXt-tiny, RN = ResNet-18.

As shown in table 19, networks trained with Top-GAP consistently show larger ERF distances across all architectures and datasets. This reveals an im-

portant aspect of how Top-GAP influences network behavior: while strong constraints lead to many decision paths, these paths are not uniformly distributed across the image. Instead, they tend to concentrate on regions containing relevant objects.

This pattern aligns with our earlier frequency map analysis (figure 32), where we observed slightly higher number of regions highlighted for weaker Top-GAP constraints. The ERF analysis now provides quantitative evidence that this spatial bias persists across different architectures and datasets.

5.3.5 Spatial Focus and Frequency maps

Another way to analyze the spatial focus of Top-GAP and to compare it with other models is through the peak-to-average ratio of frequency maps. The frequency maps show us how many paths go through each region. As we observed in figure 30, VGG11 exhibits relatively uniform spatial processing while ConvNeXt-large shows more focused processing. However, not all examples show such clear visual distinctions. For instance, in figure 32 the differences are more subtle. Therefore, we need a quantitative approach to analyze spatial processing patterns across large numbers of images.

The peak-to-average ratio provides such a measure, quantifying how effectively a network distinguishes between foreground and background regions. This analysis complements our earlier ERF investigation, offering another perspective on spatial processing patterns.

Definition 5.3.2 (Peak-to-Average Ratio). Given a frequency map $F : I \rightarrow [0, 1]$ that assigns to each grid coordinate i the fraction of decision paths that include it, the peak-to-average ratio is defined as:

$$R = \frac{\max_{i \in I} F(i)}{\frac{1}{|I|} \sum_{i \in I} F(i)}$$

A ratio of 1.0 indicates uniform spatial processing (all regions used equally often), while higher ratios indicate more focused processing (some regions consistently used more than others).

Analysis of peak-to-average ratios in table 20 reveals several key patterns. First, Top-GAP with strong constraints (low k) consistently achieves higher ratios than standard architectures, indicating more focused spatial processing. The effect is systematic, with ratios decreasing monotonically as k increases from 64 to 2048. This aligns with our earlier observation that these networks show improved robustness to distribution shifts because they have learned to identify the most relevant image regions.

Second, we observe clear relationships between architectural choices and spatial focus. Within both the ResNet and ConvNeXt families, larger models achieve higher peak-to-average ratios (ResNet101: 1.168 vs ResNet18: 1.142; ConvNeXt-Large: 1.207 vs ConvNeXt-Tiny: 1.172). This suggests that increased model capacity enables more selective spatial processing, although the effect is less pronounced than that achieved through explicit constraints.

The peak-to-average ratio shows a moderate negative correlation ($r = -0.411$) with minimal path size across architectures, suggesting that networks capable of making decisions using fewer regions tend to process spatial information more selectively. This relationship is exemplified by Top-GAP with $k=64$, which achieves both the highest peak-to-average ratio (1.289) and one of the smallest minimal path sizes (4.327) among all architectures (see also table 16).

Architecture	Variant	Peak-to-Avg \uparrow
Top-GAP (VGG11-BN)	k=64	1.289 \pm 0.178
	k=128	1.268 \pm 0.176
	k=256	1.264 \pm 0.174
	k=512	1.245 \pm 0.171
	k=1024	1.239 \pm 0.171
	k=2048	1.217 \pm 0.166
VGG11	Standard	1.141 \pm 0.124
	BN	1.152 \pm 0.135
ResNet	18	1.142 \pm 0.123
	34	1.149 \pm 0.124
	50	1.166 \pm 0.148
	101	1.168 \pm 0.144
ConvNeXt	Tiny	1.172 \pm 0.144
	Base	1.190 \pm 0.154
	Large	1.207 \pm 0.161

Table 20: Comparison of peak-to-average ratios across architectures. Higher ratios indicate more focused spatial processing. Standard deviations calculated across ImageNet validation images.

The consistency of this pattern across different architectures suggests that constraining the spatial extent of decision-making naturally leads to more focused processing strategies.

5.3.6 Ablation study

Our proposed method, Top-GAP, integrates three key components: Feature Pyramid Networks (FPN), an ℓ_1 loss, and a top-k pooling mechanism. To evaluate the contribution of each component to the overall performance, we conducted an ablation study on two datasets: wood identification and COCO. This analysis provides insight into how each element of Top-GAP affects accuracy.

FPN	ℓ_1 loss	Top-GAP	Acc _{Wood} \uparrow	Acc _{COCO} \uparrow
\times	\times	\times	0.672	0.801
\times	\times	\checkmark	0.626	0.681
\times	\checkmark	\times	0.676	0.799
\times	\checkmark	\checkmark	0.676	0.796
\checkmark	\times	\times	0.676	0.796
\checkmark	\times	\checkmark	0.614	0.532
\checkmark	\checkmark	\times	0.671	0.790
\checkmark	\checkmark	\checkmark	0.681	0.803

Table 21: Ablation study using the wood identification and COCO dataset with EfficientNet-Bo and size 224x224.

Table 21 demonstrates that the complete Top-GAP configuration, combining FPN, ℓ_1 loss, and top-k pooling, achieves the highest accuracy on both datasets.

5.4 VISUAL EXPLANATIONS FOR WOOD SPECIES CLASSIFICATION

In section 4.6, we analyzed our wood classification network using SmoothGrad attribution maps. While these maps identified potentially important regions through scalar importance scores, they represent each spatial location independently. This simplified view cannot capture how different regions interact in the network’s decision process.

The decision path framework introduced in this chapter addresses this limitation by examining what combinations of regions together maintain classification decisions when other regions are adversarially perturbed. Additionally, Top-GAP constraints offer a way to influence how networks process spatial information, leading to different patterns of feature utilization.

For a detailed understanding of these different approaches to feature detection, we analyze an *Acacia mangium* sample using attribution maps and decision paths.

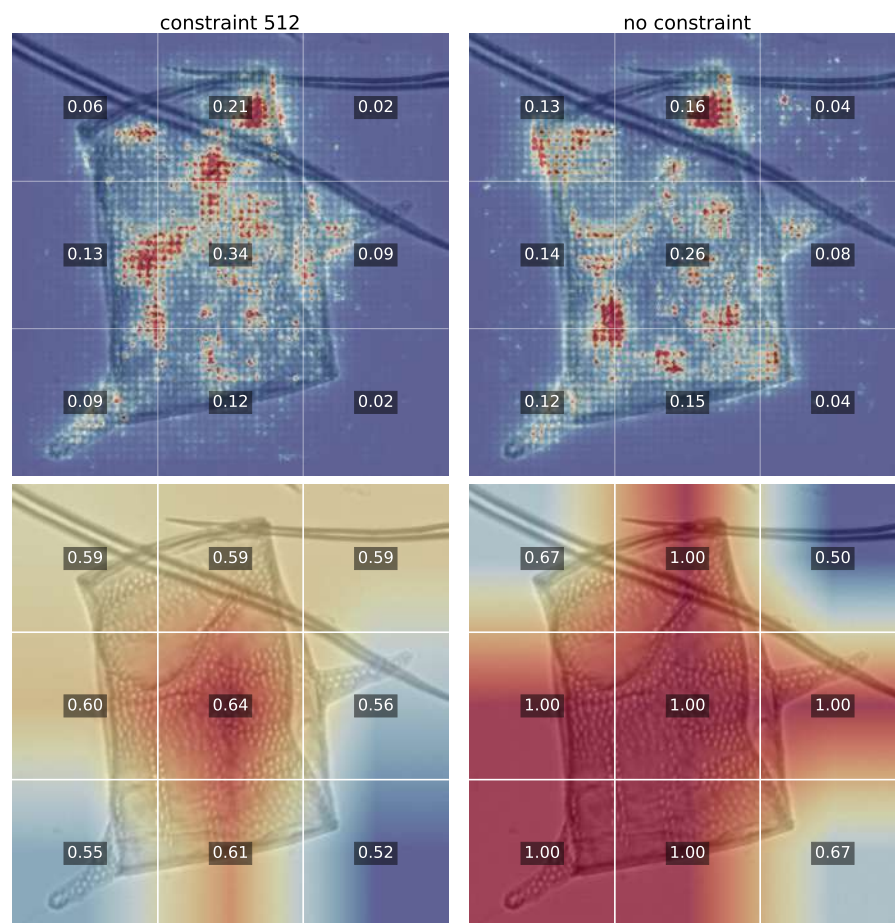


Figure 34: Comparison of attribution patterns in wood identification. Top row shows SmoothGrad attribution maps. Bottom row displays frequency maps indicating decision path participation rates. In the attribution maps, the constraint 512 yields 0.12 values per grid cell versus 0.126 for no constraint. The constrained frequency map also highlights fewer regions than the unconstrained one.

Figure 34 reveals how constrained and unconstrained networks (baseline ConvNeXt) develop distinctly different strategies for processing anatomical features. The top row shows attribution maps with grid cell values indicating

feature importance, while the bottom row displays frequency maps showing how often each region participates in successful classification.

The attribution maps demonstrate that both networks identify similar structures, with the unconstrained network showing slightly higher attribution values (0.126) compared to the constrained version (0.12). At first glance, the similar appearance of these attribution maps might suggest that the constraint had minimal impact on how the networks process information. However, the frequency maps reveal a fundamentally different picture of how these networks actually make their decisions.

The unconstrained network exhibits a rigid pattern where six regions have maximum frequency (1.00), indicating an inflexible decision process. This reveals a crucial limitation of attribution maps: while SmoothGrad shows us the independent importance of each region, it cannot tell us how these regions work together in the actual classification process. The frequency maps fill this gap by revealing that these independently identified features must actually all be present together for successful classification. If even one of these critical regions is missing, the network's ability to correctly classify the image would fail.

In contrast, the constrained network shows a much more balanced distribution of frequency values, ranging from 0.64 for the most frequently used cell to 0.52 for the least used. This narrow range indicates that no single region is absolutely essential. Instead, the network has learned to correctly identify the wood species using various combinations of features.

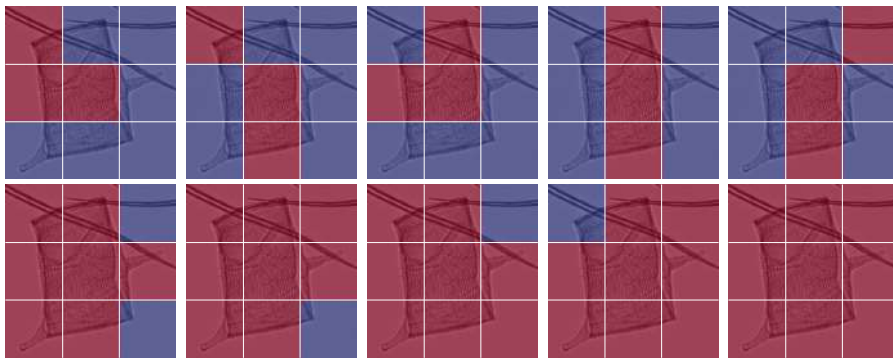


Figure 35: Visualization of decision paths for *Acacia mangium*. Top row: Constrained network ($k=512$) showing multiple valid three-cell combinations for correct classification. Bottom row: Unconstrained network requiring at least seven specific regions, indicating less flexibility in feature combination. Blue regions indicate masked areas, red shows regions used for classification.

Figure 35 further illustrates this distinction by showing specific decision paths. The unconstrained network requires a minimum of seven grid cells for correct classification, including regions containing key anatomical features such as ray patterns. The constrained network, however, can achieve correct classification with just three grid cells.

Given the constrained network's 345 decision paths, we selected and visualized the five shortest paths for clarity. In the unconstrained network, which has only five paths, all were displayed. Figure 36 shows a human expert annotation overlay for comparison.

Comparing figure 35 with this expert annotation, we see that the first decision path of the unconstrained variant (bottom row) shows strong alignment with expert features by covering most grid cells. While this comprehensive

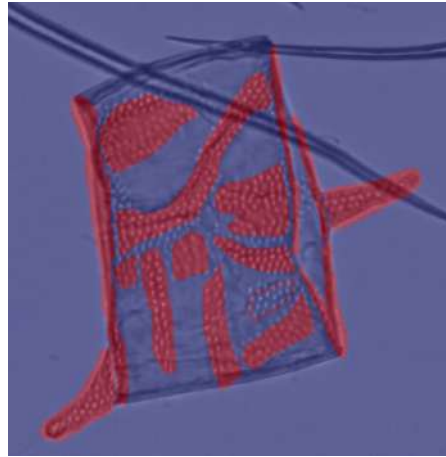


Figure 36: Expert overlay

path pattern appears as a subset in most paths, path 4 notably differs by excluding the upper-left corner region.

Even though the constrained network’s decision paths also correspond to expert-identified features, they exhibit partial overlaps and require significantly fewer cells for classification. Therefore, the network achieves correct classification with minimal spatial information.

At first glance, analyzing individual samples (here *Acacia mangium*, with another example of *Eucalyptus* provided in appendix A.3) might appear limited. However, our extensive analysis throughout this chapter has already demonstrated that these patterns are fundamental to how neural networks process spatial information. Through quantitative evaluation of hundreds of models across multiple datasets, we established that neural networks consistently operate along two main axes: path size (how many regions are required) and path count (how many alternative paths exist). The behavioral differences between constrained and unconstrained networks observed in these samples directly reflect these broader architectural properties. These examples thus serve to illustrate in detail the general principles we established through our comprehensive analysis.

Building on these general principles, our analysis reveals two key insights specifically for wood species identification. First, as demonstrated in section 4.6, attribution maps show strong alignment with expert-identified features. However, our current analysis demonstrates that attribution values alone provide an incomplete picture of how networks actually process features.

To address this limitation, frequency maps offer a more precise understanding of feature importance by revealing how regions participate in actual classification decisions. Unlike attribution maps, which provide noisy gradient values, frequency maps give clear, interpretable measurements: a value of 0 indicates a region is never used in classification, while 1 indicates the region is essential for all valid decision paths.

The combination of frequency maps and decision paths provides practical insights into how networks process information. By identifying specific combinations of regions sufficient for classification, decision paths reveal concrete strategies the network uses for species identification. This represents a significant advance over attribution-based analysis alone.

These methodological advances demonstrate how architectural constraints can guide neural networks toward more interpretable and efficient decision strategies. The integration of attribution maps, frequency maps, and decision paths provides users with a comprehensive framework for understanding and validating classification systems. This deeper understanding of neural network decision-making opens new possibilities for improving both neural network architectures and interpretability practices in automated species identification.

DISCUSSION AND CONCLUSION

This thesis addressed two interconnected challenges: developing an automated system for microscopic wood identification using deep learning, and advancing our understanding of interpretability in neural networks. Our work was motivated by increasing regulatory pressures, particularly the EU regulation on deforestation-free supply chains (EUDR) [Par23], which demands reliable methods for wood species verification in global supply chains.

Wood anatomy presents an ideal domain for exploring interpretability due to its controlled experimental conditions in this project, including standardized microscopy protocols and consistent imaging parameters. Unlike many computer vision tasks, we had access to wood anatomy experts who could evaluate our findings and provide domain knowledge about anatomical structures. This enabled meaningful discussions about whether our networks were learning relevant patterns for identification.

The following sections detail our specific contributions in both areas, analyze their theoretical and practical implications, and discuss future directions for research. We begin by examining our advances in wood identification, followed by our contributions to neural network interpretability, before synthesizing these findings and discussing their broader impact on both fields.

6.1 WOOD IDENTIFICATION

6.1.1 *Summary*

This research presented a comprehensive approach to microscopic wood identification through a two-stage pipeline combining vessel detection and classification. Through collaboration with the Thünen Institute of Wood Research, we established a substantial dataset comprising 1,614 microscopy images with 118,287 annotated vessel elements across 13 genera, focusing particularly on nine significant genera. The dataset creation process incorporated an active learning approach to optimize annotation efforts, while careful attention to sample preparation and imaging protocols ensured data quality and consistency.

In the detection phase, we first established a baseline using YOLOv7 and subsequently developed WoodYOLO, a specialized architecture optimized for vessel detection. The baseline YOLOv7-W6 model achieved an F2 score of 0.783, operating at 5184×5184 resolution and requiring substantial computational resources. WoodYOLO significantly improved upon this performance, achieving an F2 score of 0.848 (a 6% improvement) while operating at a lower 2048×2048 resolution and requiring less VRAM. This enhancement was achieved by rewriting the original YOLOv7 code and removing redundant code, while at the same time also adjusting the architecture, metrics and tuning the parameters. Notably, our experiments revealed that established techniques like mosaic augmentation, while effective in general object detection, proved detrimental in our specialized domain, highlighting the importance of domain-specific optimization.

The classification stage employed a ConvNeXt-tiny architecture, achieving a macro F1 score of 64.61% on the test dataset. Through extensive experi-

mentation, we identified optimal preprocessing strategies including resize-with-padding to 800×800 pixels, grayscale conversion (improving accuracy by 1%), and prediction averaging across multiple focal planes. The classification results revealed consistent confusion patterns between morphologically similar genera, particularly the *Liquidambar-Schima-Fagus* group and the *Populus-Salix* pair, which mirrors challenges encountered by human experts in wood anatomy.

Several technical innovations emerged from this research. Our WoodYOLO architecture demonstrated that domain-specialized models can significantly outperform general-purpose architectures while reducing computational requirements. Our analysis of focal plane integration strategies and preprocessing techniques established best practices for handling multi-focal microscopy data in classification tasks.

6.1.2 Discussion

Our research advances both the theoretical understanding and practical implementation of automated microscopic wood identification. The development of WoodYOLO and our classification pipeline directly addresses the growing need for reliable wood species verification in global supply chains, particularly in response to regulations like the EUDR. The results demonstrate that automated systems can achieve accuracy levels comparable to human expert analysis, while offering advantages in processing speed and consistency.

These findings emerged through systematic experimentation and development, yielding valuable insights into the requirements for effective wood species identification. Most notably, we discovered that successful vessel detection does not necessarily require extensive training data, as demonstrated by our first baseline model. By adapting a YOLOv7 model pretrained on the COCO dataset of natural images to microscopic wood samples, we achieved promising detection performance with a limited set of annotated images. This success reflects standard results in the ML literature [Yos+14] that show that fundamental image features learned from natural objects, such as edge detection and shape recognition, can transfer effectively to different datasets.

Now with our expanded dataset of over 100,000 vessels from 1,614 microscope images, our models can serve as powerful pre-trained networks for transfer learning. Rather than starting from scratch when detecting other anatomical structures in wood microscopy images, these pre-trained models can be fine-tuned with significantly fewer annotations, as they have already learned relevant features from our extensive dataset.

However, there is also one limitation. Our dataset, though substantial in size, was captured using a single microscope under consistent imaging parameters. This introduces significant correlations between images, as true diversity would require data from multiple devices and imaging conditions.

Despite these constraints, several characteristics of microscopic imaging support our expectation of robust real-world performance. Microscope imagery fundamentally differs from natural photography in its variability characteristics. While natural photographs of objects vary dramatically with lighting, camera angles, backgrounds, and environmental conditions, microscope imagery operates within more controlled parameters. The imaging process follows standardized protocols, with specimens prepared according to established procedures and captured using calibrated equipment. While variations exist in factors such as staining intensity, focal plane alignment, and image brightness, the fundamental morphological characteristics of wood vessels

remain consistent, providing a strong foundation for reliable automated analysis.

This inherent stability in microscopic imaging suggests that our model should generalize effectively to new samples, even with limited training data. Our data augmentation techniques likely capture a substantial portion of the real-world variation we expect to encounter in practice.

A particularly interesting aspect of our findings is the parallel between human and neural network classification strategies. Initially, we attempted to incorporate color information into the classification process, assuming that additional features would enhance accuracy. However, our experiments demonstrated that, similar to human experts, color information provided no additional discriminative power. Similarly, our initial approach of image resizing proved less effective than maintaining consistent scale relationships, mirroring the behavior of human experts.

6.1.3 *Limitations and Future Directions*

Our research opens several promising avenues for advancement, which we organize into three key areas:

Architectural Optimization

Our finding that shallow networks proved remarkably effective for vessel detection suggests opportunities for further optimization. Since the task focuses on general shape identification rather than complex feature extraction, exploring modernized versions of simpler architectures like VGG or investigating depthwise blocks could improve inference speed without compromising accuracy. The neck component, responsible for aggregating features from different feature maps, also warrants investigation to determine essential components for optimal performance.

Real-World Deployment Enhancement

For practical implementation, several challenges require attention:

- **Mixed Sample Analysis:** Developing robust post-processing methods for handling samples containing multiple species, including determining ideal confidence thresholds for production environments. The system needs to provide accurate percentage distributions of species within an image.
- **Expert-Like Selection:** While experts typically focus on a small subset of high-confidence vessels, our classifier currently processes all vessels detected by WoodYOLO. To better mimic expert behavior, we could better filter the bounding boxes using specialized heuristics.
- **Cross-Device Compatibility:** Enhancing system robustness across different microscopy conditions through domain adaptation techniques and testing across various microscope types.
- **Computational Efficiency:** Optimizing the classification process, particularly for non-GPU deployments, through techniques like model quantization. This is especially important for images containing hundreds of vessels requiring individual classification.

Extension of Capabilities

Future work should focus on expanding the system’s capabilities in several directions:

1. **Softwood Identification:** Adapting the system for softwood samples, which present different challenges due to their fiber-like structures and smaller identifying features. Furthermore, they have no vessel elements. This would require adjustments to both architecture and resolution handling.
2. **Novel Feature Discovery:** Exploring beyond traditional vessel-based classification to potentially uncover new anatomical markers for species identification. While this represents a departure from standard practices, deep learning systems might identify previously unknown discriminative features.
3. **Enhanced Multi-Focal Integration:** Developing more sophisticated methods for combining multi-focal plane data, potentially through specialized architectural components or advanced ensemble techniques for prediction aggregation.
4. **Expansion of Hardwood genera:** Increasing the number of genera included in the training dataset.

This roadmap for future development balances immediate practical improvements with longer-term research objectives, ensuring both academic advancement and industrial applicability.

6.2 NEURAL NETWORK INTERPRETABILITY

6.2.1 Summary

Our investigation into neural network interpretability began with a systematic evaluation of attribution methods across multiple standard computer vision datasets. Through comprehensive empirical analysis, we discovered a striking lack of consistency between these methods. The average similarity between different attribution approaches was only 48% for ResNet-50 and even lower at 22% for ConvNeXt-tiny. This significant variation highlighted the need for robust evaluation metrics to determine which methods most accurately reflect neural networks’ decision-making processes.

In developing such metrics, we drew inspiration from human vision research, particularly studies where researchers systematically revealed or masked different image regions to identify areas diagnostic for human recognition tasks. This empirical approach suggested that systematic manipulation of visual information through controlled revelation or masking could effectively probe feature importance. Traditional metrics like Deletion and Insertion were based on this principle. However, our analysis revealed a crucial flaw: since these metrics operate at the pixel level, they should demonstrate monotonic behavior in model confidence as information is progressively removed or added. Our experiments showed they failed to exhibit this expected characteristic.

This observation led us to formally define essential properties for masking operators, including statistical monotonicity, completeness, and significance. After analyzing various masking approaches, from Gaussian blur to brightness adjustment, we introduced our novel adversarial perturbation mask.

Through extensive evaluation across multiple architectures and datasets, we demonstrated that our approach achieved superior performance in terms of consistency, monotonicity, and baseline discrimination. Using this improved evaluation framework, we established that SmoothGrad consistently outperforms other attribution methods.

Having established SmoothGrad as the most reliable attribution method through our comprehensive evaluation framework, we applied it to our primary research focus: understanding how neural networks process wood anatomical features. Our analysis revealed three key insights about feature processing. First, neural networks predominantly focused on regions that significantly overlapped with expert-identified features. Second, while experts provided comprehensive annotations of complete anatomical structures in our dataset, neural networks identified specific discriminative regions – though this difference in documentation may not reflect actual visual processing strategies. Third, we found cases where neural networks highlighted anatomically relevant regions that experts had not traditionally emphasized, suggesting potential new diagnostic features for wood identification.

This pattern emerged clearly in our analysis of vessel structures across different genera. In *Acacia* specimens, while experts documented complete vessel outlines and arrangements, SmoothGrad attribution maps highlighted specific points along these structures that were most discriminative for classification. A notable example of novel feature detection occurred in *Eucalyptus* samples, where our model identified diagnostic patterns in parenchyma cells that could serve as additional characteristics for species identification.

While these analyses provided valuable insights into how neural networks process wood anatomical features, our investigation revealed a fundamental limitation of attribution maps: they treat pixels independently, failing to capture how different anatomical regions interact in the network's decision process. This insight led to the development of our decision paths framework, which moved beyond single-feature analysis to understand how neural networks combine multiple anatomical features for classification. By examining these paths, we gained a new perspective on network behavior through frequency maps, which reveal how often each region participates in successful classification.

The decision paths framework yielded surprising insights into general neural network behavior. Through analysis of 644 models, we discovered that larger, better-performing models tend to use smaller regions but develop many redundant paths, while smaller models require larger regions but maintain fewer paths. This relationship manifested as a significant negative correlation ($r=-0.55$) between minimal path size and accuracy. Importantly, we found that no network consistently maintains classification using only a single small region, suggesting that robust visual recognition inherently requires multiple feature combinations.

Based on these insights about feature combinations, we introduced Top-GAP, an architectural modification designed to guide how networks process spatial information. While we successfully encouraged networks to focus on smaller regions, we discovered an unexpected trade-off: constraining region size led to an increase in the number of alternative paths. This revealed a fundamental limitation in neural network interpretability – we could not force networks to use both small regions and few paths simultaneously.

Nevertheless, Top-GAP demonstrated significant benefits. Through analysis of effective receptive fields and peak-to-average ratios, we showed that constrained networks exhibit more selective spatial processing. This selective

attention translated to improved robustness, with Top-GAP networks showing better performance under distribution shifts. Furthermore, we observed that larger models naturally develop more selective spatial processing, suggesting that increased model capacity enables more sophisticated feature selection strategies.

These findings reveal that we do not necessarily make models more interpretable by restricting them to use a small set of clearly defined features. Instead, our work suggests that the complexity of visual recognition inevitably manifests either through many small regions in different combinations or through fewer, larger regions. Moreover, our analysis suggests that the flexibility in feature selection we observe in our models may be fundamental to effective visual recognition. The discovery of potentially overlooked feature combinations, such as subtle patterns in parenchyma cells, opens new directions for wood anatomical research by highlighting anatomical characteristics that merit closer scientific investigation.

6.2.2 Discussion

The findings of our thesis reveal fundamental insights about neural network architecture, evaluation, and the nature of interpretability itself. Our analysis begins with the observation that current approaches to characterizing neural networks rely heavily on metrics such as parameter count, computational cost (FLOPs), and various performance measures. While these metrics often correlate with model accuracy and appear to capture important aspects of model capability, our research demonstrates their insufficiency. The case of VGG16 serves as a compelling example: despite its 138 million parameters, modern architectures with far fewer parameters consistently achieve superior results. This disparity highlights how parameter count alone provides an incomplete picture of model capability.

Building on this understanding, our analysis introduces a crucial new dimension to the evaluation of neural architectures by examining how networks process visual information. We discovered that models can achieve robust processing through two distinct strategies: either by developing many small-region redundant paths or by utilizing fewer large-region paths. This insight helps explain why architectures with similar parameter counts or FLOPs can perform very differently – they may be implementing fundamentally different visual processing strategies. These findings suggest that a comprehensive evaluation of neural networks must consider not just performance metrics, but also how they process spatial information and develop redundant pathways.

This more nuanced understanding of network behavior leads us to reconsider traditional approaches to neural network interpretability. While methods like attribution maps have been widely adopted for highlighting regions important for classification, our research reveals their fundamental limitations.

These methods provide only a static snapshot of network behavior, failing to distinguish between necessary and sufficient features for classification. For instance, while an attribution map might highlight a dog’s tail as important, it cannot tell us whether the network could still correctly classify the image if the tail were hidden. Our decision path analysis addresses this limitation by revealing all possible combinations of features that lead to correct classification, showing both which feature sets are sufficient (any complete path) and which features are truly necessary (appearing in all paths). Our decision path analysis represents a step forward by revealing how networks combine multiple features, yet even this more sophisticated approach cannot fully characterize

network decision-making. This limitation parallels a fundamental challenge in understanding human visual processing: experts' post-hoc explanations of their decisions may not fully reflect their intuitive recognition process.

Rather than pursuing complete interpretability as an end goal, our findings suggest a more pragmatic approach: developing targeted analytical tools for specific needs. This might involve using attribution maps for feature discovery, designing focused tests for detecting spurious correlations, and employing decision path analysis to understand feature combinations. Such a multi-faceted approach acknowledges that perfect interpretability, whether for neural networks or human experts, may be an unrealistic goal while still providing practical insights for different applications.

The implications of our research extend to the commonly cited tradeoff between accuracy and interpretability. This tradeoff is often illustrated through oversimplified comparisons with linear models, but our work demonstrates how this framing breaks down when considering complex visual tasks. Consider what appears to be a simple approach: using a CNN's first layer as a feature extractor followed by linear regression. While this creates a technically simpler model, it doesn't actually achieve interpretability. Even though we understand these first-layer features as edge detectors, the specific feature combinations and their interactions remain opaque. This reveals a fundamental challenge: robust visual recognition appears to inherently require sophisticated feature processing that resists reduction to simple, human-interpretable rules.

Our findings ultimately point toward a more nuanced approach to interpretability, one guided by task requirements. For simple tasks where interpretable features naturally exist, we should strive for truly interpretable models where both features and their combinations are meaningful. Such models can serve as valuable baselines for more complex tasks, demonstrating what performance is achievable with fully explainable features and decision rules before moving to sophisticated architectures needed for real-world robustness. While complex visual tasks may never be fully interpretable, we can continue developing better tools to understand their behavior at different levels of abstraction, from pixel-level attribution to semantic-level concepts. This multi-level approach acknowledges the inherent complexity of visual recognition while providing practical paths forward for improving our understanding of neural network behavior.

6.2.3 *Limitations*

There are several methodological limitations that warrant discussion, though we believe they do not impact our main findings.

First, our path framework employs a relatively coarse 3×3 grid, which limits precise feature localization. While this resolution was necessary due to computational constraints in our brute force search across grid cells, our quantitative and qualitative analyses demonstrate that the framework still produces consistent and meaningful results. This resolution choice also influenced our model selection process.

Second, regarding model coverage, we evaluated approximately 650 models from the timm collection, representing roughly half of the available models. Our focus on architectures trained at 224×224 resolution was deliberate, as higher resolutions would result in even coarser grid representations relative to the input. Nevertheless, our selection encompasses all commonly used models in practical applications, including vision transformers.

Third, our path analysis methodology relies on successful adversarial attacks, which presents certain constraints. Specifically, for adversarially trained models where our FGSM attack fails, we cannot perform the required perturbation analysis. However, as demonstrated in section 4.5.4, both PGD and FGSM generate similar perturbations (± 1) in terms of raw pixel values. While alternative approaches like using black pixels were considered, we opted for our current method based on the superior reliability of our perturbation operator, as shown in section 4.5.

Fourth, regarding dataset sample size, we conducted many experiments only on a subset of the total 50,000 ImageNet validation images. Our analysis revealed that increasing the sample size beyond approx. 1000-2000 images did not yield significantly different results. For ImageNet experiments, we primarily used VGG11 for training Top-GAP due to its computational efficiency. While expanding to more architectures is possible, the challenge lies in hyperparameter tuning to achieve higher accuracy than standard trained networks across all architectures. We partially addressed this limitation by conducting extensive training across diverse real-world datasets.

Despite these limitations, we maintain that our core findings remain valid. While certain aspects could benefit from more detailed analysis, both our attribution maps and path framework findings are supported by robust empirical evidence across multiple experiments and datasets.

6.2.4 *Future Directions*

Our findings open several promising avenues for future research in neural network interpretability and architecture design. We organize these directions into three main themes: architectural innovations, methodological improvements, and theoretical advances.

Architectural Innovations for Robust Feature Processing

While our research revealed the importance of redundant pathways in neural networks, we need to better understand how to systematically develop and control these redundancies. Future work should investigate how different architectural components influence pathway formation and robustness. Of particular interest is the role of normalization techniques (batch normalization, layer normalization) and skip connections in shaping how networks develop multiple pathways for classification. These components might affect not just model performance, but fundamentally alter how networks learn to process information redundantly.

Additionally, we should explore architectural modifications that explicitly encourage beneficial forms of redundancy. While our Top-GAP approach demonstrated that we can influence spatial processing through regularization, future research could investigate other architectural modifications that shape pathway formation. We have shown that larger models naturally develop more pathways, but there might be other more efficient ways to encourage path redundancy.

Methodological Improvements in Network Analysis

Our current decision path analysis framework provides valuable insights into network behavior, but there are important aspects of pathway development that remain to be explored. A particularly promising direction is investigat-

ing how pathways evolve during the training process. We hypothesize that networks might begin with more rigid, limited pathways early in training before developing richer redundancies as training progresses. Understanding this evolution could provide crucial insights into how networks learn to process information robustly and how different training regimes might influence pathway development.

Bridging Information Theory and Semantic Understanding

A fundamental challenge in neural network interpretability lies in connecting information-theoretic aspects with semantic understanding. Current attribution methods and evaluation metrics operate primarily at the pixel level, failing to capture semantic relationships that emerge in deeper network layers. This limitation reflects the broader “symbol grounding problem” in artificial intelligence (see section 4.4.2) – how do we connect low-level features to high-level semantic concepts?

A key challenge in attribution methods is bridging the gap between pixel-level attributions and the semantic concepts learned by networks. While current attribution maps effectively identify important pixels or regions, they don’t directly connect to the high-level semantic concepts that the network has learned to recognize. This creates a fundamental disconnect: the network operates at a semantic level, making decisions based on learned concepts, but our attribution methods can only show us pixel-level importance.

Future research should focus on developing attribution methods that can better align with the network’s semantic understanding. This could involve:

- Creating attribution methods that directly map to learned semantic concepts rather than just highlighting important pixels
- Developing techniques to connect pixel-level attributions with the network’s internal representations
- Investigating how different network architectures affect the relationship between pixel-level attributions and semantic understanding

Theoretical Foundations

Finally, we need stronger theoretical foundations for understanding the relationship between network architecture, redundant pathways, and robust classification. Key questions include:

- What are the fundamental limits on the number of paths and path size?
- How does the trade-off between path size and number of paths scale with model capacity?
- Can we develop a formal framework for understanding how networks balance information compression with semantic representation?
- What are the theoretical connections between path redundancy and generalization?

These theoretical investigations could help guide practical architecture design while deepening our understanding of neural network behavior. By connecting empirical observations about pathway formation with formal theories of learning and information processing, we might develop more principled approaches to network design and analysis.

The field of neural network interpretability stands at an exciting juncture. While our current tools have revealed important insights about network behavior, significant work remains in developing more sophisticated methods for understanding and guiding how networks process information. Progress in these directions could lead not only to more interpretable models but also to fundamental advances in our understanding of machine learning and artificial intelligence.

6.3 BROADER IMPACT

This thesis demonstrates how neural network interpretability and wood anatomical research can mutually strengthen each other. By developing frameworks to analyze how networks process wood anatomical features, we have shown how ML can contribute meaningfully to wood anatomy while remaining grounded in expert knowledge.

Our work demonstrates that neural networks can do more than just automate wood identification – they can provide new perspectives on anatomical features when we understand their decision-making processes. Through close collaboration with wood anatomists, we’ve developed methods to reveal how networks process and combine features, enabling experts to validate and evaluate these computational approaches to wood analysis.

The timing of this research is particularly relevant given increasing regulatory pressures like the EU regulation on deforestation-free supply chains (EUDR) [Par23]. As society demands more accountability in timber supply chains, the need for interpretable wood identification systems becomes more pressing. Our work shows that it’s possible to develop systems that both perform accurate species identification and provide insights into their analysis process.

Looking beyond wood identification, this research suggests a framework for deploying neural networks in other domains requiring specialized expertise, such as medical imaging or cellular microscopy. The key lies in developing interpretable systems that can be validated by domain experts while potentially revealing new patterns or relationships in their respective fields.

This combination of neural network interpretability research with wood anatomy expertise provides a model for future work in specialized domains.

BIBLIOGRAPHY

- [AB18] Guillaume Alain and Yoshua Bengio. *Understanding intermediate layers using linear classifier probes*. 2018. arXiv: 1610.01644 [stat.ML]. URL: <https://arxiv.org/abs/1610.01644> (cit. on p. 19).
- [Ara+20] Eric Arazo, Diego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. *Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning*. 2020. arXiv: 1908.02983 [cs.CV]. URL: <https://arxiv.org/abs/1908.02983> (cit. on p. 28).
- [BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML]. URL: <https://arxiv.org/abs/1607.06450> (cit. on p. 11).
- [Bai+21] Yutong Bai, Jieru Mei, Alan Yuille, and Cihang Xie. *Are Transformers More Robust Than CNNs?* 2021. arXiv: 2111.05464 [cs.CV]. URL: <https://arxiv.org/abs/2111.05464> (cit. on p. 47).
- [BSF94] Y. Bengio, P. Simard, and P. Frasconi. „Learning long-term dependencies with gradient descent is difficult“. In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166. DOI: 10.1109/72.279181 (cit. on p. 10).
- [Bey+20] Lucas Beyer, Olivier J. Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. *Are we done with ImageNet?* 2020. arXiv: 2006.07159 [cs.CV]. URL: <https://arxiv.org/abs/2006.07159> (cit. on p. 89).
- [Bjo+18] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. „Understanding Batch Normalization“. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf (cit. on p. 11).
- [BHH20] Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. „Accurately computing the log-sum-exp and softmax functions“. In: *IMA Journal of Numerical Analysis* 41.4 (Aug. 2020), pp. 2311–2330. ISSN: 0272-4979. DOI: 10.1093/imanum/draa038. eprint: <https://academic.oup.com/imajna/article-pdf/41/4/2311/40758053/draa038.pdf>. URL: <https://doi.org/10.1093/imanum/draa038> (cit. on p. 9).
- [BWL20] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV]. URL: <https://arxiv.org/abs/2004.10934> (cit. on pp. 17, 36, 39).
- [BL23] Seok-Yong Byun and Wonju Lee. *Recipro-CAM: Fast gradient-free visual explanations for convolutional neural networks*. 2023. arXiv: 2209.14074 [cs.CV] (cit. on pp. 60, 77).
- [Can86] John Canny. „A computational approach to edge detection“. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on pp. 75, 77).

- [Cas+20] Paola Cascante-Bonilla, Fuwen Tan, Yanjun Qi, and Vicente Ordonez. *Curriculum Labeling: Revisiting Pseudo-Labeling for Semi-Supervised Learning*. 2020. arXiv: 2001.06001 [cs.LG]. URL: <https://arxiv.org/abs/2001.06001> (cit. on p. 28).
- [Cha+18] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. „Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks“. In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, Mar. 2018. DOI: 10.1109/wacv.2018.00097 (cit. on pp. 59, 77).
- [Com89] IAWA Committee. „IAWA List of Microscopic Features for Hardwood Identification“. In: *IAWA Bulletin* 10 (1989), pp. 219–332 (cit. on pp. 3, 57).
- [CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. USA: Wiley-Interscience, 2006. ISBN: 0471241954 (cit. on p. 65).
- [CH20] Francesco Croce and Matthias Hein. *Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks*. 2020. arXiv: 2003.01690 [cs.LG]. URL: <https://arxiv.org/abs/2003.01690> (cit. on p. 20).
- [DR20] Saurabh Desai and Harish G. Ramaswamy. „Ablation-CAM: Visual Explanations for Deep Convolutional Network via Gradient-free Localization“. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 972–980. DOI: 10.1109/WACV45572.2020.9093360 (cit. on p. 60).
- [DZR12] James J. DiCarlo, Davide Zoccolan, and Nicole C. Rust. „How Does the Brain Solve Visual Object Recognition?“ In: *Neuron* 73.3 (Feb. 2012), pp. 415–434. ISSN: 0896-6273. DOI: 10.1016/j.neuron.2012.01.010. URL: <http://dx.doi.org/10.1016/j.neuron.2012.01.010> (cit. on pp. 18, 19).
- [Din+21] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. „RepVGG: Making VGG-style ConvNets Great Again“. In: *CoRR* abs/2101.03697 (2021). arXiv: 2101.03697. URL: <https://arxiv.org/abs/2101.03697> (cit. on p. 74).
- [Dos+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: <https://arxiv.org/abs/2010.11929> (cit. on pp. 16, 47).
- [DBR20] Gintare Karolina Dziugaite, Shai Ben-David, and Daniel M. Roy. *Enforcing Interpretability and its Statistical Impacts: Trade-offs between Accuracy and Interpretability*. 2020. arXiv: 2010.13764 [cs.LG]. URL: <https://arxiv.org/abs/2010.13764> (cit. on p. 23).
- [DGR16] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M. Roy. *A study of the effect of JPG compression on adversarial images*. 2016. arXiv: 1608.00853 [cs.CV]. URL: <https://arxiv.org/abs/1608.00853> (cit. on p. 20).

- [ECD22] Alexandre Englebert, Olivier Cornu, and Christophe De Vleeschouwer. *Poly-CAM: High resolution class activation map for convolutional neural networks*. 2022. DOI: 10.48550/ARXIV.2204.13359. URL: <https://arxiv.org/abs/2204.13359> (cit. on pp. 60, 77).
- [Eve+10] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. „The Pascal Visual Object Classes (VOC) Challenge“. In: *Int. J. Comput. Vision* 88.2 (June 2010), pp. 303–338. ISSN: 0920-5691. DOI: 10.1007/s11263-009-0275-4. URL: <https://doi.org/10.1007/s11263-009-0275-4> (cit. on p. 44).
- [FKS23] Alex Fang, Simon Kornblith, and Ludwig Schmidt. *Does progress on ImageNet transfer to real-world datasets?* 2023. arXiv: 2301.04644 [cs.CV]. URL: <https://arxiv.org/abs/2301.04644> (cit. on p. 47).
- [Fen+21] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang. *TOOD: Task-aligned One-stage Object Detection*. 2021. arXiv: 2108.07755 [cs.CV]. URL: <https://arxiv.org/abs/2108.07755> (cit. on p. 41).
- [FV17] Ruth Fong and Andrea Vedaldi. „Interpretable Explanations of Black Boxes by Meaningful Perturbation“. In: *CoRR abs/1704.03296* (2017). arXiv: 1704.03296. URL: <http://arxiv.org/abs/1704.03296> (cit. on p. 63).
- [FC19] Jonathan Frankle and Michael Carbin. *The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks*. 2019. arXiv: 1803.03635 [cs.LG]. URL: <https://arxiv.org/abs/1803.03635> (cit. on p. 18).
- [Fra45] GL Franklin. „Preparation of thin sections of synthetic resins and wood-resin composites, and a new macerating method for wood“. In: *Nature* 155.3924 (1945), pp. 51–51 (cit. on pp. 3, 26).
- [Fu+20] Ruigang Fu, Qingyong Hu, Xiaohu Dong, Yulan Guo, Yinghui Gao, and Biao Li. „Axiom-based Grad-CAM: Towards Accurate Visualization and Explanation of CNNs“. In: *CoRR abs/2008.02312* (2020). arXiv: 2008.02312. URL: <https://arxiv.org/abs/2008.02312> (cit. on p. 60).
- [Fuk80] Kunihiko Fukushima. „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. In: *Biological Cybernetics* 36.4 (Apr. 1980), pp. 193–202. ISSN: 1432-0770. DOI: 10.1007/bf00344251. URL: <http://dx.doi.org/10.1007/BF00344251> (cit. on p. 13).
- [GG16] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2016. arXiv: 1506.02142 [stat.ML]. URL: <https://arxiv.org/abs/1506.02142> (cit. on p. 12).
- [GIG17] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. *Deep Bayesian Active Learning with Image Data*. 2017. arXiv: 1703.02910 [cs.LG]. URL: <https://arxiv.org/abs/1703.02910> (cit. on p. 27).
- [Gao+21] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. *Fast Convergence of DETR with Spatially Modulated Co-Attention*. 2021. arXiv: 2108.02404 [cs.CV]. URL: <https://arxiv.org/abs/2108.02404> (cit. on p. 32).

- [Ge+21a] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. *OTA: Optimal Transport Assignment for Object Detection*. 2021. arXiv: 2103.14259 [cs.CV]. URL: <https://arxiv.org/abs/2103.14259> (cit. on p. 41).
- [Ge+21b] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. *YOLOX: Exceeding YOLO Series in 2021*. 2021. arXiv: 2107.08430 [cs.CV]. URL: <https://arxiv.org/abs/2107.08430> (cit. on pp. 36–38, 40, 41).
- [Gir+14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV]. URL: <https://arxiv.org/abs/1311.2524> (cit. on p. 32).
- [GB10] Xavier Glorot and Yoshua Bengio. „Understanding the difficulty of training deep feedforward neural networks.“ In: *AISTATS*. Ed. by Yee Whye Teh and D. Mike Titterton. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. URL: <http://dblp.uni-trier.de/db/journals/jmlr/jmlrp9.html#GlorotB10> (cit. on p. 10).
- [Gol+23] Micah Goldblum, Hossein Souri, Renkun Ni, Manli Shu, Viraj Prabhu, Gowthami Somepalli, Prithvijit Chattopadhyay, Mark Ibrahim, Adrien Bardes, Judy Hoffman, Rama Chellappa, Andrew Gordon Wilson, and Tom Goldstein. *Battle of the Backbones: A Large-Scale Comparison of Pretrained Models across Computer Vision Tasks*. 2023. arXiv: 2310.19909 [cs.CV]. URL: <https://arxiv.org/abs/2310.19909> (cit. on pp. 47, 53).
- [GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2015. arXiv: 1412.6572 [stat.ML]. URL: <https://arxiv.org/abs/1412.6572> (cit. on pp. 20, 72).
- [GS01] Frédéric Gosselin and Philippe G. Schyns. „Bubbles: a technique to reveal the use of information in recognition tasks“. In: *Vision Research* 41.17 (2001), pp. 2261–2271. ISSN: 0042-6989. DOI: [https://doi.org/10.1016/S0042-6989\(01\)00097-9](https://doi.org/10.1016/S0042-6989(01)00097-9). URL: <https://www.sciencedirect.com/science/article/pii/S0042698901000979> (cit. on p. 64).
- [Harg0] Stevan Harnad. „The symbol grounding problem“. In: *Physica D: Nonlinear Phenomena* 42.1 (1990), pp. 335–346. ISSN: 0167-2789. DOI: [https://doi.org/10.1016/0167-2789\(90\)90087-6](https://doi.org/10.1016/0167-2789(90)90087-6). URL: <https://www.sciencedirect.com/science/article/pii/0167278990900876> (cit. on p. 65).
- [HTF09] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. ISBN: 9780387848587. DOI: 10.1007/978-0-387-84858-7. URL: <http://dx.doi.org/10.1007/978-0-387-84858-7> (cit. on pp. 7, 30).
- [He+18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV]. URL: <https://arxiv.org/abs/1703.06870> (cit. on p. 32).
- [He+15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. „Deep Residual Learning for Image Recognition“. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: <http://arxiv.org/abs/1512.03385> (cit. on pp. 16, 47, 74).

- [He+15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV]. URL: <https://arxiv.org/abs/1502.01852> (cit. on p. 10).
- [He+24] Xin He, Daniël M. Pelt, JingRan Gao, Barbara Gravendeel, PeiQi Zhu, SongYang Chen, Jian Qiu, and Frederic Lens. „Machine learning-based wood anatomy identification: towards anatomical feature recognition“. In: *IAWA Journal* 45.4 (Apr. 2024), pp. 457–475. ISSN: 2294-1932. DOI: 10.1163/22941932-bja10157. URL: <http://dx.doi.org/10.1163/22941932-bja10157> (cit. on p. 3).
- [Hel+18] Stephanie Helmling, Andrea Olbrich, Immo Heinz, and Gerald Koch. „Atlas of vessel elements: Identification of Asian timbers“. In: *Iawa Journal* 39.3 (2018), pp. 249–352 (cit. on pp. 3, 25, 26).
- [Hel+16] Stephanie Helmling, Andrea Olbrich, Lena Tepe, and Gerald Koch. „Qualitative and quantitative characteristics of macerated vessels of 23 mixed tropical hardwood (MTH) species: a data collection for the identification of wood species in pulp and paper“. In: *Holzforschung* 70.9 (2016), pp. 839–844 (cit. on p. 26).
- [Hen+21a] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. *The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization*. 2021. arXiv: 2006.16241 [cs.CV]. URL: <https://arxiv.org/abs/2006.16241> (cit. on p. 89).
- [HD19] Dan Hendrycks and Thomas Dietterich. *Benchmarking Neural Network Robustness to Common Corruptions and Perturbations*. 2019. arXiv: 1903.12261 [cs.LG]. URL: <https://arxiv.org/abs/1903.12261> (cit. on p. 96).
- [HG23] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs)*. 2023. arXiv: 1606.08415 [cs.LG]. URL: <https://arxiv.org/abs/1606.08415> (cit. on pp. 9, 17).
- [Hen+21b] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. *Natural Adversarial Examples*. 2021. arXiv: 1907.07174 [cs.LG]. URL: <https://arxiv.org/abs/1907.07174> (cit. on p. 89).
- [HMR86] Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. „Distributed Representations“. In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. Ed. by David E. Rumelhart and James L. McClelland. Cambridge, MA: MIT Press, 1986, pp. 77–109 (cit. on p. 18).
- [Hoc91] Sepp Hochreiter. „Untersuchungen zu dynamischen neuronalen Netzen“. In: *Master’s thesis, Institut für Informatik, Technische Universität, München 1* (1991), pp. 1–150 (cit. on p. 10).
- [HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. „Multilayer feedforward networks are universal approximators“. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8). URL: <https://www.sciencedirect.com/science/article/pii/0893608089900208> (cit. on p. 8).

- [How+17] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. arXiv: 1704.04861 [cs.CV]. URL: <https://arxiv.org/abs/1704.04861> (cit. on p. 14).
- [Hu+19] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. *Squeeze-and-Excitation Networks*. 2019. arXiv: 1709.01507 [cs.CV]. URL: <https://arxiv.org/abs/1709.01507> (cit. on p. 16).
- [HLW16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. „Densely Connected Convolutional Networks“. In: *CoRR abs/1608.06993* (2016). arXiv: 1608.06993. URL: <http://arxiv.org/abs/1608.06993> (cit. on pp. 16, 47, 74).
- [Hub+19] Martin A Hubbe, Richard P Chandra, Dilek Dogu, and STJ Van Velzen. „Analytical staining of cellulosic materials: a review“. In: *BioResources* 14.3 (2019), pp. 7387–7464 (cit. on p. 26).
- [HW62] D. H. Hubel and T. N. Wiesel. „Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex“. In: *The Journal of Physiology* 160.1 (Jan. 1962), pp. 106–154. ISSN: 1469-7793. DOI: 10.1113/jphysiol.1962.sp006837. URL: <http://dx.doi.org/10.1113/jphysiol.1962.sp006837> (cit. on p. 13).
- [Ilv95] Marja-Sisko Ilvessalo-Pfäffli. *Fiber atlas: identification of papermaking fibers*. Springer Science & Business Media, 1995 (cit. on pp. 3, 25).
- [Ily+19] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. *Adversarial Examples Are Not Bugs, They Are Features*. 2019. arXiv: 1905.02175 [stat.ML]. URL: <https://arxiv.org/abs/1905.02175> (cit. on p. 20).
- [IS15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: <https://arxiv.org/abs/1502.03167> (cit. on p. 10).
- [JGH20] Arthur Jacot, Franck Gabriel, and Clément Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. 2020. arXiv: 1806.07572 [cs.LG]. URL: <https://arxiv.org/abs/1806.07572> (cit. on p. 22).
- [Jia+21] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. „LayerCAM: Exploring Hierarchical Class Activation Maps for Localization“. In: *IEEE Transactions on Image Processing* 30 (2021), pp. 5875–5888. DOI: 10.1109/TIP.2021.3089943 (cit. on pp. 60, 77).
- [Kap+21] Andrei Kapishnikov, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. „Guided Integrated Gradients: An Adaptive Path Method for Removing Noise“. In: *CoRR abs/2106.09788* (2021). arXiv: 2106.09788. URL: <https://arxiv.org/abs/2106.09788> (cit. on pp. 59, 77).

- [KN22] Sayash Kapoor and Arvind Narayanan. *Leakage and the Reproducibility Crisis in ML-based Science*. 2022. arXiv: 2207.07048 [cs.LG]. URL: <https://arxiv.org/abs/2207.07048> (cit. on p. 30).
- [KB17] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: <https://arxiv.org/abs/1412.6980> (cit. on p. 8).
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf (cit. on p. 13).
- [LJH15] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. *A Simple Way to Initialize Recurrent Networks of Rectified Linear Units*. 2015. arXiv: 1504.00941 [cs.NE]. URL: <https://arxiv.org/abs/1504.00941> (cit. on p. 10).
- [LeC+89] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. „Backpropagation Applied to Handwritten Zip Code Recognition“. In: *Neural Computation* 1 (1989), pp. 541–551 (cit. on p. 13).
- [LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. „Deep learning“. In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: <http://dx.doi.org/10.1038/nature14539> (cit. on p. 8).
- [Li+22] Hui Li, Zihao Li, Rui Ma, and Tieru Wu. *FD-CAM: Improving Faithfulness and Discriminability of Visual Explanation for CNNs*. 2022. DOI: 10.48550/ARXIV.2206.08792. URL: <https://arxiv.org/abs/2206.08792> (cit. on p. 60).
- [LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. *Network In Network*. 2014. arXiv: 1312.4400 [cs.NE]. URL: <https://arxiv.org/abs/1312.4400> (cit. on p. 14).
- [Lin+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV]. URL: <https://arxiv.org/abs/1612.03144> (cit. on p. 37).
- [Lin+18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV]. URL: <https://arxiv.org/abs/1708.02002> (cit. on p. 32).
- [Lin+15a] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: <https://arxiv.org/abs/1405.0312> (cit. on p. 35).

- [Lin+15b] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV] (cit. on pp. 95, 97).
- [Lip17] Zachary C. Lipton. *The Mythos of Model Interpretability*. 2017. arXiv: 1606.03490 [cs.LG]. URL: <https://arxiv.org/abs/1606.03490> (cit. on pp. 21, 22).
- [Liu+18] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV]. URL: <https://arxiv.org/abs/1803.01534> (cit. on p. 37).
- [Liu+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. „SSD: Single Shot MultiBox Detector“. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2 (cit. on p. 32).
- [Liu+22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. „A ConvNet for the 2020s“. In: *CoRR abs/2201.03545 (2022)*. arXiv: 2201.03545. URL: <https://arxiv.org/abs/2201.03545> (cit. on pp. 11, 16, 74).
- [LH19] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: <https://arxiv.org/abs/1711.05101> (cit. on p. 13).
- [Luo+16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. „Understanding the Effective Receptive Field in Deep Convolutional Neural Networks“. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf (cit. on pp. 15, 96).
- [MHN13] A.L. Maas, A.Y. Hannun, and A.Y. Ng. „Rectifier Nonlinearities Improve Neural Network Acoustic Models“. In: *Proceedings of the International Conference on Machine Learning*. Atlanta, Georgia, 2013 (cit. on p. 9).
- [Mad+19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019. arXiv: 1706.06083 [stat.ML]. URL: <https://arxiv.org/abs/1706.06083> (cit. on p. 20).
- [Mic+18] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. *Mixed Precision Training*. 2018. arXiv: 1710.03740 [cs.AI]. URL: <https://arxiv.org/abs/1710.03740> (cit. on p. 31).

- [Mos+20] Nikita Moshkov, Botond Mathe, Attila Kertesz-Farkas, Reka Hollandi, and Peter Horvath. „Test-time augmentation for deep learning-based cell segmentation on microscopy images“. In: *Scientific Reports* 10.1 (Mar. 2020). ISSN: 2045-2322. DOI: 10.1038/s41598-020-61808-3. URL: <http://dx.doi.org/10.1038/s41598-020-61808-3> (cit. on p. 53).
- [MY20] Mohammed Bany Muhammad and Mohammed Yeasin. „Eigen-CAM: Class Activation Map using Principal Components“. In: *CoRR abs/2008.00299* (2020). arXiv: 2008.00299. URL: <https://arxiv.org/abs/2008.00299> (cit. on p. 60).
- [MH21] Samuel G. Müller and Frank Hutter. *TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation*. 2021. arXiv: 2103.10158 [cs.CV]. URL: <https://arxiv.org/abs/2103.10158> (cit. on p. 17).
- [Nai+20] Rakshit Naidu, Ankita Ghosh, Yash Maurya, Shamanth R. Nayak K, and Soumya Snigdha Kundu. „IS-CAM: Integrated Score-CAM for axiomatic-based explanations“. In: *CoRR abs/2010.03023* (2020). arXiv: 2010.03023. URL: <https://arxiv.org/abs/2010.03023> (cit. on p. 60).
- [Nay+22] Niv Nayman, Avram Golbert, Asaf Noy, Tan Ping, and Lihi Zelnik-Manor. *Diverse Imagenet Models Transfer Better*. 2022. arXiv: 2204.09134 [cs.CV]. URL: <https://arxiv.org/abs/2204.09134> (cit. on p. 47).
- [Nie+24a] Lars Nieradzic, Jördis Sieburg-Rockel, Stephanie Helmling, Janis Keuper, Thomas Weibel, Andrea Olbrich, and Henrike Stephani. „Automating Wood Species Detection and Classification in Microscopic Images of Fibrous Materials with Deep Learning“. In: *Microscopy and Microanalysis* 30.3 (May 2024), pp. 508–520. ISSN: 1435-8115. DOI: 10.1093/mam/ozae038. URL: <http://dx.doi.org/10.1093/mam/ozae038> (cit. on pp. 5, 25, 26, 28, 49, 50, 52, 54, 56).
- [NSK24a] Lars Nieradzic, Henrike Stephani, and Janis Keuper. „Reliable Evaluation of Attribution Maps in CNNs: A Perturbation-Based Approach“. In: *International Journal of Computer Vision* (Nov. 2024). ISSN: 1573-1405. DOI: 10.1007/s11263-024-02282-6. URL: <http://dx.doi.org/10.1007/s11263-024-02282-6> (cit. on pp. 5, 57, 61, 63, 69, 74, 76).
- [NSK24b] Lars Nieradzic, Henrike Stephani, and Janis Keuper. *Top-GAP: Integrating Size Priors in CNNs for more Interpretability, Robustness, and Bias Mitigation*. 2024. arXiv: 2409.04819 [cs.CV]. URL: <https://arxiv.org/abs/2409.04819> (cit. on pp. 5, 57, 81, 97).
- [Nie+24b] Lars Nieradzic, Henrike Stephani, Jördis Sieburg-Rockel, Stephanie Helmling, Andrea Olbrich, and Janis Keuper. „Challenging the Black Box: A Comprehensive Evaluation of Attribution Maps of CNN Applications in Agriculture and Forestry“. In: *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2024, pp. 483–492. DOI: 10.5220/0012363400003660. URL: <http://dx.doi.org/10.5220/0012363400003660> (cit. on pp. 5, 57, 69).

- [Nie+24c] Lars Nieradzick, Henrike Stephani, Jördis Sieburg-Rockel, Stephanie Helmling, Andrea Olbrich, Stephanie Wrage, and Janis Keuper. „WoodYOLO: A Novel Object Detector for Wood Species Detection in Microscopic Images“. In: *Forests* 15.11 (Oct. 2024), p. 1910. ISSN: 1999-4907. DOI: 10.3390/f15111910. URL: <http://dx.doi.org/10.3390/f15111910> (cit. on pp. 5, 25, 43, 45).
- [Ome+19] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldemariam. „Smooth Grad-CAM++: An Enhanced Inference Level Visualization Technique for Deep Convolutional Neural Network Models“. In: *CoRR abs/1908.01224* (2019). arXiv: 1908.01224. URL: <http://arxiv.org/abs/1908.01224> (cit. on pp. 60, 77).
- [OB21] Juri Opitz and Sebastian Burst. *Macro F1 and Macro F1*. 2021. arXiv: 1911.03347 [cs.LG]. URL: <https://arxiv.org/abs/1911.03347> (cit. on p. 48).
- [Par+12] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. „Cats and Dogs“. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2012 (cit. on pp. 74, 95, 97).
- [Par23] European Parliament. „Regulation (EU) 2023/1115 of the European Parliament and of the Council of 31 May 2023 on the making available on the Union market and the export from the Union of certain commodities and products associated with deforestation and forest degradation and repealing Regulation (EU) No 995/2010“. In: *Off. J. Eur. Union* 150 (2023), pp. 206–247 (cit. on pp. 3, 105, 114).
- [PMB13] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. 2013. arXiv: 1211.5063 [cs.LG]. URL: <https://arxiv.org/abs/1211.5063> (cit. on pp. 10, 58).
- [PDS18] Vitali Petsiuk, Abir Das, and Kate Saenko. *RISE: Randomized Input Sampling for Explanation of Black-box Models*. 2018. DOI: 10.48550/ARXIV.1806.07421. URL: <https://arxiv.org/abs/1806.07421> (cit. on pp. 60, 63).
- [RR22] Lassi Raatikainen and Esa Rahtu. *The Weighting Game: Evaluating Quality of Explainability Methods*. 2022. DOI: 10.48550/ARXIV.2208.06175. URL: <https://arxiv.org/abs/2208.06175> (cit. on p. 64).
- [Rad+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. *Learning Transferable Visual Models From Natural Language Supervision*. 2021. arXiv: 2103.00020 [cs.CV]. URL: <https://arxiv.org/abs/2103.00020> (cit. on p. 47).
- [RZL17] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: 1710.05941 [cs.NE]. URL: <https://arxiv.org/abs/1710.05941> (cit. on p. 9).
- [Rav+20] Prabu Ravindran, Blaise J Thompson, Richard K Soares, and Alex C Wiedenhoft. „The XyloTron: flexible, open-source, image-based macroscopic field identification of wood products“. In: *Frontiers in plant science* 11 (2020), p. 1015 (cit. on p. 3).

- [Rec+19] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. *Do ImageNet Classifiers Generalize to ImageNet?* 2019. arXiv: 1902.10811 [cs.CV]. URL: <https://arxiv.org/abs/1902.10811> (cit. on p. 89).
- [Red+16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV]. URL: <https://arxiv.org/abs/1506.02640> (cit. on pp. 32, 34, 35, 38, 40).
- [RF16] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV]. URL: <https://arxiv.org/abs/1612.08242> (cit. on p. 39).
- [RF18] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV]. URL: <https://arxiv.org/abs/1804.02767> (cit. on pp. 36, 39).
- [Ren+16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: <https://arxiv.org/abs/1506.01497> (cit. on p. 32).
- [Rez+19] Hamid Rezaatofghi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*. 2019. arXiv: 1902.09630 [cs.CV]. URL: <https://arxiv.org/abs/1902.09630> (cit. on pp. 34, 40).
- [Rud19] Cynthia Rudin. *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. 2019. arXiv: 1811.10154 [stat.ML]. URL: <https://arxiv.org/abs/1811.10154> (cit. on pp. 22, 23).
- [RC19] Flavio Ruffinatto and Alan Crivellaro. *Atlas of macroscopic wood identification: with a special focus on timbers used in Europe and CITES-listed species*. Springer Nature, 2019 (cit. on p. 3).
- [RM86] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. The MIT Press, 1986. ISBN: 9780262291408. DOI: 10.7551/mitpress/5236.001.0001. URL: <http://dx.doi.org/10.7551/mitpress/5236.001.0001> (cit. on p. 18).
- [Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. *ImageNet Large Scale Visual Recognition Challenge*. 2015. arXiv: 1409.0575 [cs.CV]. URL: <https://arxiv.org/abs/1409.0575> (cit. on pp. 74, 89).
- [Sag+20] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. *Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization*. 2020. arXiv: 1911.08731 [cs.LG]. URL: <https://arxiv.org/abs/1911.08731> (cit. on p. 95).
- [San+19] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML]. URL: <https://arxiv.org/abs/1805.11604> (cit. on p. 11).

- [SMB10] Dominik Scherer, Andreas Müller, and Sven Behnke. „Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition“. In: *Artificial Neural Networks – ICANN 2010*. Ed. by Konstantinos Diamantaras, Wlodek Duch, and Lazaros S. Iliadis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 92–101. ISBN: 978-3-642-15825-4 (cit. on p. 14).
- [SSH21] Robin M. Schmidt, Frank Schneider, and Philipp Hennig. *Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers*. 2021. arXiv: 2007.01547 [cs.LG]. URL: <https://arxiv.org/abs/2007.01547> (cit. on p. 7).
- [Sch+20] Nele Schmitz, Hans Beeckman, Céline Blanc-Jolivet, Laura Boeschoten, Jez WB Braga, José-Antonio Cabezas, Gilles Chaix, Simon Crameri, Bernd Degen, Victor Deklerck, et al. *Overview of current practices in data analysis for wood identification-A guide for the different timber tracking methods*. 2020 (cit. on p. 3).
- [Sel+19] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. „Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization“. In: *International Journal of Computer Vision* 128.2 (Oct. 2019), pp. 336–359. DOI: 10.1007/s11263-019-01228-7 (cit. on pp. 59, 77).
- [Shi+20] Xiangwei Shi, Seyran Khademi, Yunqiang Li, and Jan van Gemert. „Zoom-CAM: Generating Fine-grained Pixel Annotations from Image Labels“. In: *CoRR abs/2010.08644* (2020). arXiv: 2010.08644. URL: <https://arxiv.org/abs/2010.08644> (cit. on p. 60).
- [SK19] Connor Shorten and Taghi M. Khoshgoftaar. „A survey on Image Data Augmentation for Deep Learning“. In: *Journal of Big Data* 6.1 (July 2019). ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: <http://dx.doi.org/10.1186/s40537-019-0197-0> (cit. on p. 17).
- [ST17] Ravid Shwartz-Ziv and Naftali Tishby. *Opening the Black Box of Deep Neural Networks via Information*. 2017. arXiv: 1703.00810 [cs.LG]. URL: <https://arxiv.org/abs/1703.00810> (cit. on p. 68).
- [Sil+22] José Luís Silva, Rui Bordalo, José Pissarra, and Paloma de Palacios. „Computer Vision-Based Wood Identification: A Review“. In: *Forests* 13.12 (2022), p. 2041 (cit. on p. 3).
- [SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. „Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps“. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2014. URL: <http://arxiv.org/abs/1312.6034> (cit. on pp. 58, 77).
- [SZ15] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: <https://arxiv.org/abs/1409.1556> (cit. on p. 16).

- [Smi+17] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. „SmoothGrad: removing noise by adding noise“. In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: <http://arxiv.org/abs/1706.03825> (cit. on pp. 60, 77).
- [Spr+15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. „Striving for Simplicity: The All Convolutional Net“. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6806> (cit. on p. 58).
- [Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. „Dropout: A Simple Way to Prevent Neural Networks from Overfitting“. In: *Journal of Machine Learning Research* 15:56 (2014), pp. 1929–1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html> (cit. on pp. 12, 18).
- [STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. „Axiomatic Attribution for Deep Networks“. In: *CoRR* abs/1703.01365 (2017). arXiv: 1703.01365. URL: <http://arxiv.org/abs/1703.01365> (cit. on pp. 58, 77).
- [Sze+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV]. URL: <https://arxiv.org/abs/1312.6199> (cit. on p. 19).
- [TL19] Mingxing Tan and Quoc V. Le. „EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks“. In: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: <http://arxiv.org/abs/1905.11946> (cit. on pp. 16, 47, 74).
- [Tia+19] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. *FCOS: Fully Convolutional One-Stage Object Detection*. 2019. arXiv: 1904.01355 [cs.CV]. URL: <https://arxiv.org/abs/1904.01355> (cit. on p. 40).
- [TZ15] Naftali Tishby and Noga Zaslavsky. *Deep Learning and the Information Bottleneck Principle*. 2015. arXiv: 1503.02406 [cs.LG]. URL: <https://arxiv.org/abs/1503.02406> (cit. on pp. 19, 68).
- [Tou+21] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. *Going deeper with Image Transformers*. 2021. arXiv: 2103.17239 [cs.CV]. URL: <https://arxiv.org/abs/2103.17239> (cit. on p. 17).
- [TK15] Satoru Tsuchikawa and Hikaru Kobori. „A review of recent application of near infrared spectroscopy to wood science and technology“. In: *Journal of Wood Science* 61.3 (2015), pp. 213–220 (cit. on p. 3).
- [ÜÖÇ19] F. Özge Ünel, Burak O. Özkalayci, and Cevahir Çiğla. „The Power of Tiling for Small Object Detection“. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019, pp. 582–591. DOI: 10.1109/CVPRW.2019.00084 (cit. on p. 31).

- [Vas+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: <https://arxiv.org/abs/1706.03762> (cit. on pp. 9, 11).
- [VWB16] Andreas Veit, Michael Wilber, and Serge Belongie. *Residual Networks Behave Like Ensembles of Relatively Shallow Networks*. 2016. arXiv: 1605.06431 [cs.CV]. URL: <https://arxiv.org/abs/1605.06431> (cit. on p. 16).
- [VJ01] P. Viola and M. Jones. „Rapid object detection using a boosted cascade of simple features“. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517 (cit. on p. 22).
- [VSL24] Kirill Vishniakov, Zhiqiang Shen, and Zhuang Liu. *ConvNet vs Transformer, Supervised vs CLIP: Beyond ImageNet Accuracy*. 2024. arXiv: 2311.09215 [cs.CV]. URL: <https://arxiv.org/abs/2311.09215> (cit. on p. 47).
- [Wah+11] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. July 2011 (cit. on pp. 95, 97).
- [Wan+24] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. *YOLOv10: Real-Time End-to-End Object Detection*. 2024. arXiv: 2405.14458 [cs.CV]. URL: <https://arxiv.org/abs/2405.14458> (cit. on pp. 25, 32).
- [WBL21] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *Scaled-YOLOv4: Scaling Cross Stage Partial Network*. 2021. arXiv: 2011.08036 [cs.CV]. URL: <https://arxiv.org/abs/2011.08036> (cit. on p. 37).
- [WBL22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. 2022. arXiv: 2207.02696 [cs.CV]. URL: <https://arxiv.org/abs/2207.02696> (cit. on pp. 28, 32, 35).
- [WLY22] Chien-Yao Wang, Hong-Yuan Mark Liao, and I-Hau Yeh. *Designing Network Design Strategies Through Gradient Path Analysis*. 2022. arXiv: 2211.04800 [cs.CV]. URL: <https://arxiv.org/abs/2211.04800> (cit. on p. 37).
- [Wan+19a] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. arXiv: 1911.11929 [cs.CV]. URL: <https://arxiv.org/abs/1911.11929> (cit. on p. 36).
- [WYL24] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*. 2024. arXiv: 2402.13616 [cs.CV]. URL: <https://arxiv.org/abs/2402.13616> (cit. on p. 32).
- [Wan+20a] Haofan Wang, Rakshit Naidu, Joy Michael, and Soumya Snigdha Kundu. *SS-CAM: Smoothed Score-CAM for Sharper Visual Feature Localization*. 2020. DOI: 10.48550/ARXIV.2006.14255. URL: <https://arxiv.org/abs/2006.14255> (cit. on p. 60).

- [Wan+20b] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. *Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks*. 2020. arXiv: 1910.01279 [cs.CV] (cit. on pp. 60, 77).
- [Wan+19b] Haohan Wang, Songwei Ge, Eric P. Xing, and Zachary C. Lipton. *Learning Robust Global Representations by Penalizing Local Predictive Power*. 2019. arXiv: 1905.13549 [cs.CV]. URL: <https://arxiv.org/abs/1905.13549> (cit. on pp. 89, 96).
- [Wan+23] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, XiaoWei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. *InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions*. 2023. arXiv: 2211.05778 [cs.CV]. URL: <https://arxiv.org/abs/2211.05778> (cit. on p. 32).
- [Wan+17] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammad-hadi Bagheri, and Ronald M. Summers. „ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases.“ In: *CVPR*. IEEE Computer Society, 2017, pp. 3462–3471. ISBN: 978-1-5386-0457-1. URL: <http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#WangPLLBS17> (cit. on pp. 74, 132).
- [Wie20] Alex C Wiedenhoeft. „The XyloPhone: toward democratizing access to high-quality macroscopic imaging for wood and other substrates“. In: *Iawa Journal* 41.4 (2020), pp. 699–719 (cit. on p. 3).
- [Wig19] Ross Wightman. *PyTorch Image Models*. <https://github.com/rwightman/pytorch-image-models>. 2019. DOI: 10.5281/zenodo.4414861 (cit. on pp. 47, 89).
- [WH18] Yuxin Wu and Kaiming He. *Group Normalization*. 2018. arXiv: 1803.08494 [cs.CV]. URL: <https://arxiv.org/abs/1803.08494> (cit. on p. 11).
- [XVS20] Shawn Xu, Subhashini Venugopalan, and Mukund Sundararajan. „Attribution in Scale and Space“. In: *CoRR abs/2004.03383* (2020). arXiv: 2004.03383. URL: <https://arxiv.org/abs/2004.03383> (cit. on pp. 59, 77).
- [Yos+14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG]. URL: <https://arxiv.org/abs/1411.1792> (cit. on pp. 19, 106).
- [ZF13] Matthew D. Zeiler and Rob Fergus. „Visualizing and Understanding Convolutional Networks“. In: *CoRR abs/1311.2901* (2013). arXiv: 1311.2901. URL: <http://arxiv.org/abs/1311.2901> (cit. on pp. 19, 58).
- [Zha+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. *Understanding deep learning requires rethinking generalization*. 2017. arXiv: 1611.03530 [cs.LG]. URL: <https://arxiv.org/abs/1611.03530> (cit. on p. 19).

- [Zha+22] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. *DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection*. 2022. arXiv: 2203.03605 [cs.CV]. URL: <https://arxiv.org/abs/2203.03605> (cit. on p. 32).
- [Zha+16] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. „Top-down Neural Attention by Excitation Back-prop“. In: *CoRR abs/1608.00507* (2016). arXiv: 1608.00507. URL: <http://arxiv.org/abs/1608.00507> (cit. on p. 64).
- [ZRY21] Qing-Long Zhang, Lu Rao, and Yubin Yang. „Group-CAM: Group Score-Weighted Visual Explanations for Deep Convolutional Networks“. In: *CoRR abs/2103.13859* (2021). arXiv: 2103.13859. URL: <https://arxiv.org/abs/2103.13859> (cit. on p. 60).
- [Zha+24] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. *DETRs Beat YOLOs on Real-time Object Detection*. 2024. arXiv: 2304.08069 [cs.CV]. URL: <https://arxiv.org/abs/2304.08069> (cit. on p. 32).
- [Zhe+19] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. *Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression*. 2019. arXiv: 1911.08287 [cs.CV]. URL: <https://arxiv.org/abs/1911.08287> (cit. on pp. 34, 40).
- [Zhe+21] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. *Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation*. 2021. arXiv: 2005.03572 [cs.CV]. URL: <https://arxiv.org/abs/2005.03572> (cit. on pp. 34, 40).
- [Zho+15a] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. *Learning Deep Features for Discriminative Localization*. 2015. arXiv: 1512.04150 [cs.CV]. URL: <https://arxiv.org/abs/1512.04150> (cit. on p. 59).
- [Zho+15b] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. *Object Detectors Emerge in Deep Scene CNNs*. 2015. arXiv: 1412.6856 [cs.CV]. URL: <https://arxiv.org/abs/1412.6856> (cit. on p. 18).

APPENDIX

A.1 EVALUATION OF ATTRIBUTION MAPS

While we have already presented quantitative results demonstrating the variability of different attribution maps and metrics, here we provide some practical examples to illustrate these differences.

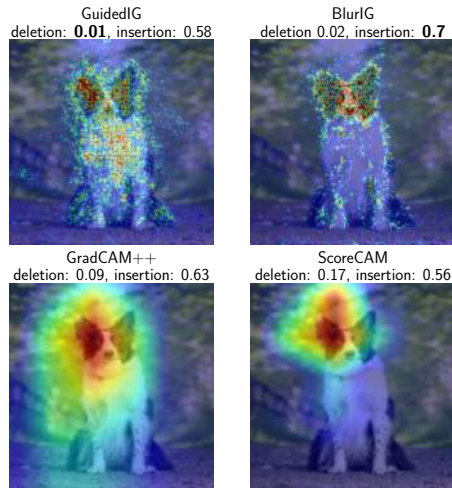


Figure 37: Comparison of four attribution maps (AM) on an ImageNet sample.

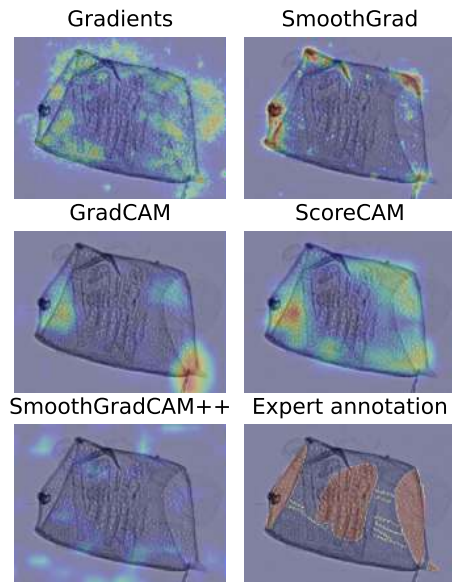


Figure 38: Attribution maps applied to a vessel element, demonstrating substantial variation in highlighted regions.

The plot figure 37 shows divergent results: while the Deletion method (lower is better) favors Guided Integrated Gradients (IG), the Insertion method

(higher is better) indicates Blur IG as optimal. This divergence further validates our findings about the limitations of traditional evaluation metrics.

Similar in figure 38, we see that each method focuses on different anatomical features, with none achieving complete alignment with expert annotations. This variability reinforces our findings about the importance of robust evaluation metrics for attribution methods. We note, however, that for SmoothGrad we did use the default value and did not tune its σ parameter.

To demonstrate the practical utility of our metric (insertion combined with perturbation operator), we analyze its application on the ChestX-ray8 dataset [Wan+17]. We focus on the SmoothGrad attribution map, which requires careful tuning of its noise level parameter σ . The choice of this parameter is crucial, as inappropriate values can lead to misleading results. We compare two approaches for determining the optimal parameter: the deletion metric using black pixel masking, and our insertion metric using adversarial perturbation.

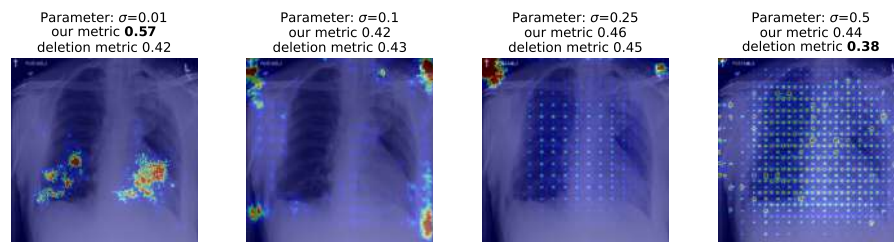


Figure 39: The optimal σ values for SmoothGrad are determined by two evaluation methods: Deletion (\downarrow) and our metric (\uparrow). While visual analysis clearly suggests the first image as the most appropriate choice, the Deletion metric unexpectedly favors the last image as the preferred option.

The results demonstrate that our metric more reliably identifies the optimal σ parameter, aligning better with visual intuition.

A.2 TOP-GAP

Our Top-GAP method architecturally constrains the neural network to process only a limited set of spatial locations (e.g., 64 pixels). While we primarily focus on SmoothGrad in this thesis, having demonstrated it produces the best attribution maps according to our metric, we also examine other attribution methods. In particular, we analyze here the standard class activation map (CAM/GradCAM).

In both figures figure 40 and figure 41, we see the effect of our constraint.

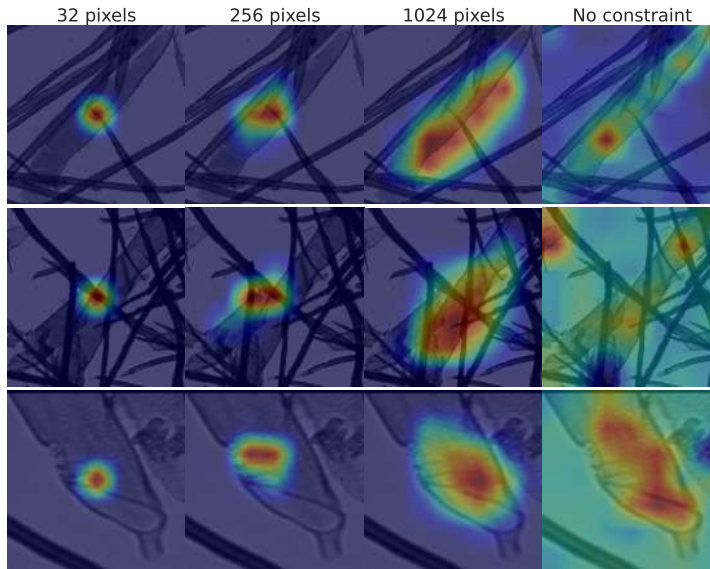


Figure 40: Impact of pixel constraint on CAM (Wood identification dataset). “No constraint” denotes a standard unmodified EfficientNet-Bo model using CAM/GradCAM. The object in the center, known as a vessel should be highlighted. Without our method, the background containing fibers is also highlighted.

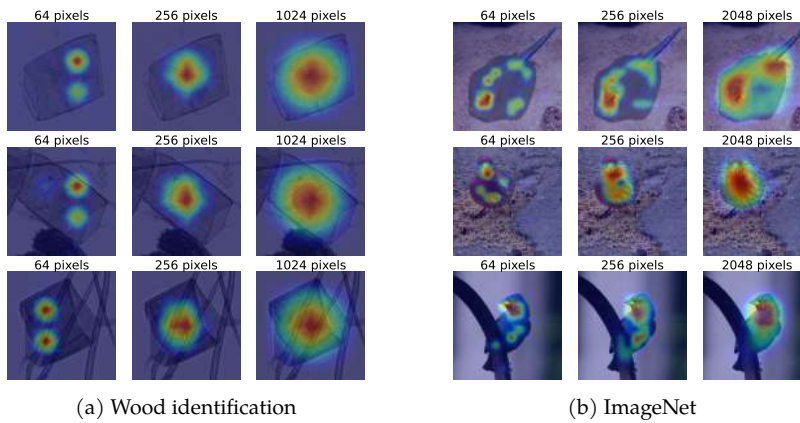


Figure 41: Another example for our wood dataset (a) and ImageNet (b), where we limit the locations in the output feature map that the CNN can use to make predictions. Increasing the allowed pixel count leads to more pixels being highlighted in the class activation map (CAM).

We quantitatively repeated this experiment for other datasets as well. The following plot shows that by increasing the constraint k , more and more pixels are highlighted in the class activation map.

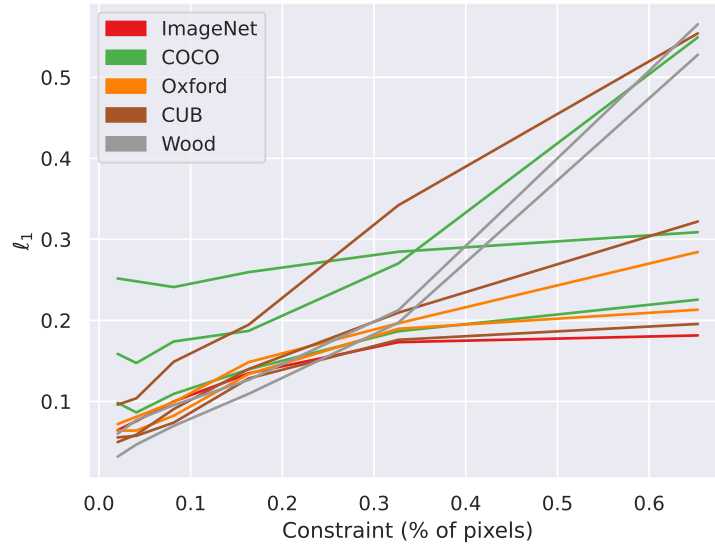


Figure 42: Each line in the graph represents a dataset+architecture combination. The x-axis shows the normalized k value (e.g. $\frac{64}{56^2}$) for the constraint, while the y-axis represents the ℓ_1 norm.

A.3 PATH FRAMEWORK

To further illustrate how architectural constraints influence decision-making patterns, we analyze the decision paths for a *Eucalyptus* sample. Figure 43 reveals differences between constrained and unconstrained networks in how they process anatomical features.

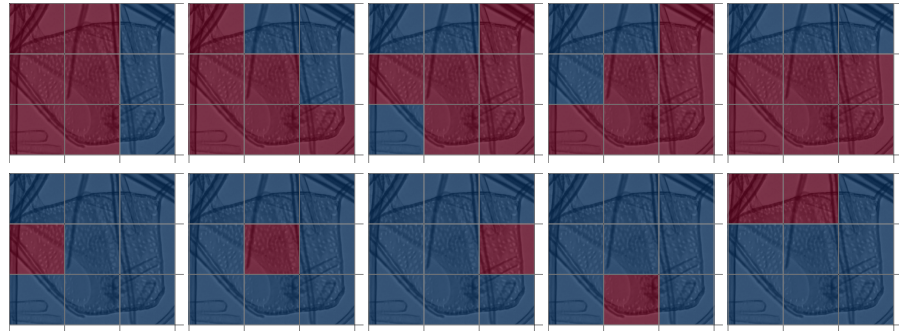


Figure 43: Visualization of decision paths for *Eucalyptus*. Top row: Constrained network ($k=64$). Bottom row: Unconstrained network. Blue regions indicate masked areas, red shows regions used for classification.

The unconstrained network requires at least 6 regions for successful classification. Furthermore, it shows a sharp transition in success rates: from 0% with 5 regions to 97.22% with 7 regions. The network relies heavily on seeing most of the image to maintain correct classification. The following table 22 shows this in more detail.

While these success rates are different than those observed in our ImageNet analysis figure 28 (where success rates for 1-3 regions were below 12%), this

Path Size	Successful Paths	Total Possible	Success Rate (%)
1	0	9	0.00
2	0	36	0.00
3	0	84	0.00
4	0	126	0.00
5	0	126	0.00
6	5	84	5.95
7	35	36	97.22
8	9	9	100.00
9	1	1	100.00

Table 22: Decision path statistics for the original (unconstrained) network. Path size indicates the number of regions used, while success rate shows the percentage of valid paths among all possible combinations of that size.

difference is expected given the sample size. Our ImageNet statistics were averaged across 500 images, while these tables represent performance on a single *Eucalyptus* sample.

The constrained network ($k=64$) achieves successful classification with as few as one region (44.44% success rate) and reaches near-perfect performance with just three regions (94.05% success rate).

Path Size	Successful Paths	Total Possible	Success Rate (%)
1	4	9	44.44
2	30	36	83.33
3	79	84	94.05
4	126	126	100.00
5	126	126	100.00
6	84	84	100.00
7	36	36	100.00
8	9	9	100.00
9	1	1	100.00

Table 23: Decision path statistics for the network with Top-GAP constraint $k=64$. The constrained network achieves successful classification with significantly fewer regions, as shown by the high success rates for small path sizes.

This example reinforces our broader findings about how Top-GAP constraints guide networks toward more focused spatial processing. While the constrained network develops more alternative pathways, these paths consistently overlap in regions containing discriminative features. This increased redundancy, rather than indicating unfocused processing, actually demonstrates more precise spatial attention.