



BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

**METHODS SUPPORTING INTERPRETABLE
AND EXPLAINABLE MACHINE LEARNING**

METODY PODPORUJÍCÍ INTERPRETOVATELNÉ A VYSVĚTLITELNÉ STROJOVÉ UČENÍ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Bc. ŠIMON ŠMÍDA

SUPERVISOR

VEDOUCÍ PRÁCE

prof. Ing. LUKÁŠ SEKANINA, Ph.D.

BRNO 2025

Master's Thesis Assignment



165030

Institut: Department of Computer Systems (DCSY)
Student: **Šmída Šimon, Bc.**
Programme: Information Technology and Artificial Intelligence
Specialization: Machine Learning
Title: **Methods Supporting Interpretable and Explainable Machine Learning**
Category: Artificial Intelligence
Academic year: 2024/25

Assignment:

1. Familiarize yourself with the concepts of interpretability and explainability in machine learning (ML).
2. Deliver a study on definitions, methods, evaluation practices, and typical case studies dealing with interpretability and explainability in ML.
3. Select at least three methods that can increase the interpretability or explainability of ML and implement them in a suitable development platform.
4. Evaluate these methods on selected relevant case studies.
5. Propose suitable extensions of one of the selected methods to improve its performance. Compare the obtained results with the reference method.
6. Discuss the results obtained in the project.

Literature:

- According to the instructions of the supervisor.

Requirements for the semestral defence:

- Items 1 and 2 of the assignment.

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Sekanina Lukáš, prof. Ing., Ph.D.**
Head of Department: Sekanina Lukáš, prof. Ing., Ph.D.
Beginning of work: 1.11.2024
Submission deadline: 21.5.2025
Approval date: 20.2.2025

Abstract

The increasing complexity of machine learning systems poses significant challenges in understanding their decision-making processes – a phenomenon known as the black-box problem. This thesis systematically investigates existing methods for interpretability and explainability of machine learning models, with a focus on convolutional neural networks and image data. It covers intrinsic, post-hoc, and mechanistic interpretability paradigms, key techniques (e.g., LIME, SHAP, gradient-based methods), and evaluation strategies for interpretability methods (both qualitative and quantitative). The conducted experimental analyses assess the impact of model quality on explanations, compare the performance of interpretability methods across different datasets (MNIST, ImageNet), and evaluate their robustness against various types of input data perturbations. A notable contribution of this work is the proposal and experimental validation of the Adaptive-Metric Softmax (AMS) Ensemble method, which is a novel approach that enables a dynamic combination of multiple interpretability methods to enhance the reliability of explanations. This thesis provides theoretical insights and empirical evidence supporting the development of more transparent and trustworthy machine learning models.

Abstrakt

Narastajúca zložitosť systémov na báze strojového učenia predstavuje dôležité výzvy pri snahe porozumieť procesom ich rozhodovania – jav známy ako “problém čiernej skrinky”. Diplomová práca systematicky skúma existujúce metódy interpretability a vysvetliteľnosti modelov strojového učenia, so zameraním na konvolučné neurónové siete a obrazové dáta. Práca pokrýva intrinzičné, post-hoc a mechanistické paradigmy interpretovateľnosti, kľúčové techniky (napr. LIME, SHAP, metódy založené na gradientoch) a stratégie vyhodnotenia metód interpretability (kvalitatívne a kvantitatívne). Vykonané experimentálne analýzy posudzujú vplyv kvality modelu na vysvetlenia, porovnávajú výkonnosť metód interpretability na dátových rôznych sadách (MNIST, ImageNet) a hodnotia ich robustnosť voči rôznym typom perturbácií vstupných dát. Prínosom práce je aj návrh a experimentálna validácia metódy Adaptive-Metric Softmax (AMS) Ensemble, čo je nový prístup umožňujúci dynamicky kombinovať viacero metód interpretability s cieľom zlepšiť spoľahlivosť vysvetlení. Táto práca prináša teoretické poznatky a empirické dôkazy smerujúce k vývoju transparentnejších a dôveryhodnejších modelov strojového učenia.

Keywords

Explainable Artificial Intelligence, XAI, Interpretable Machine Learning, Gradient-Based Methods, XAI Ensembles, XAI Evaluation, Convolutional Neural Networks

Kľúčové slová

vysvetliteľná umelá inteligencia, interpretovateľné strojové učenie, gradienté metódy interpretability, kombinované metódy interpretability, vyhodnocovanie XAI, konvolučné neurónové siete

Reference

ŠMÍDA, Šimon. *Methods Supporting Interpretable and Explainable Machine Learning*. Brno, 2025. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Lukáš Sekanina, Ph.D.

Rozšírený abstrakt

Moderné systémy strojového učenia, predovšetkým hlboké neurónové siete, dosahujú pozoruhodné výsledky v širokom spektre aplikácií, od rozpoznávania obrazu až po spracovanie prirodzeného jazyka. S rastúcou komplexnosťou týchto modelov sa zároveň zvyšuje aj náročnosť pochopenia ich vnútorných rozhodovacích mechanizmov. Tento jav, často označovaný ako “problém čiernej skrinky” (angl. *black-box problem*), predstavuje výraznú prekážku pre ich nasadenie v kritických oblastiach, kde je koncepcia transparentnosti a dôveryhodnosti systému kľúčová. Vysvetliteľná umelá inteligencia (XAI) sa preto stáva nevyhnutnou súčasťou výskumu s cieľom rozvinúť metódy a techniky, ktoré umožnia nahliadnuť do fungovania týchto komplexných modelov a interpretovať ich predikcie.

Táto diplomová práca sa systematicky venuje problematike interpretovateľnosti a vysvetliteľnosti modelov strojového učenia, so špecifickým zameraním na konvolučné neurónové siete (CNN) a ich aplikácie pri spracovaní obrazových dát (klasifikačné úlohy). Cieľom práce je poskytnúť komplexný prehľad súčasných XAI prístupov, analyzovať ich silné a slabé stránky prostredníctvom experimentálneho vyhodnotenia a navrhnúť rozšírenie existujúcich metód s cieľom zlepšiť vlastnosti generovaných vysvetlení.

Práca je štruktúrovaná tak, aby pokryla teoretické základy, praktickú implementáciu a experimentálne overenie vybraných metód XAI. V úvodných kapitolách sú podrobne rozpracované fundamentálne koncepty interpretovateľnosti a vysvetliteľnosti, vrátane definícií, motivácie a klasifikácie existujúcich metód. Podrobnejšie sa práca venuje trom hlavným paradigmám: intrinzickej interpretovateľnosti (modely transparentné v rámci svojho návrhu, napr. lineárne modely, rozhodovacie stromy), post-hoc metódam (aplikovaným na už natrénované modely, napr. LIME, SHAP, metódy založené na gradientoch) a mechanickej interpretovateľnosti (zameranej na reverzné inžinierstvo neurónových sietí s cieľom odhaliť nimi implementované algoritmy). Pre každú kategóriu sú diskutované kľúčové techniky, ich princípy fungovania a typické prípady použitia.

Významná časť práce sa zaoberá problematikou vyhodnocovania metód interpretovateľnosti. Medzi existujúce stratégie patria kvalitatívne prístupy, založené na ľudskom úsudku a vizuálnej inšpekcii, a kvantitatívne metódy, ktoré využívajú formálne definované metriky na meranie vlastností výstupov metód interpretability, ako sú vernosť (*fidelity*), robustnosť, lokalizácia, komplexnosť, či citlivosť na náhodné javy. V práci je predstavený a využitý súbor techník Quantus pre systematické kvantitatívne hodnotenie XAI metód.

Jadrom praktického prínosu práce je motivácia, návrh a implementácia novej metódy Adaptive-Metric Softmax (AMS) Ensemble, ktorá rozširuje sľubné prístupy kombinovania jednoduchých metód s cieľom spojiť ich pozitívne aspekty do výsledného robustnejšieho vysvetlenia. Tento prístup rieši problém stability a nekonzistencie jednotlivých XAI metód dynamickým kombinovaním ich výstupov. AMS Ensemble priradzuje váhy jednotlivým metódam na základe ich aktuálnej kvality pre dané vstupné dáta, meranej pomocou súboru kvantitatívnych metrík, a následne agreguje ich vysvetlenia pomocou softmax váhovania korigovaného o konfidenčný faktor.

Experimentálna časť práce bola navrhnutá s cieľom systematicky preskúmať a porovnať rôzne aspekty metód XAI. Realizované boli štyri hlavné skupiny experimentov:

1. **Vplyv kvality modelu na kvalitu vysvetlenia:** Analýza bola zameraná na skúmanie toho, ako sa mení charakter a spoľahlivosť vysvetlení generovaných pre modely s rôznou úrovňou natrénovania (nedotrénovaný, pretrénovaný a správne natrénovaný) na dátovej sade MNIST.

2. **Porovnanie XAI metód na rôznych dátových sadách:** Boli kvalitatívne a kvantitatívne zhodnotené vlastnosti vybraných metód (Vanilla Gradients, Integrated Gradients, SHAP varianty, Guided Grad-CAM) na dátových sadách rôznej zložitosti: MNIST (s vlastným CNN modelom), ImageNet (s predtrénovaným modelom ResNet).
3. **Robustnosť vysvetlení:** Bola analyzovaná stabilita vysvetlení voči rôznym typom perturbácií vstupných dát: aditívny Gaussovský šum simulujúci prirodzené variácie v dátach, testy invariantnosti voči transformáciám zachovávajúcim predikciu modelu a odolnosť voči adversariálnym útokom (angl. *adversarial attacks*) cieľným na manipuláciu vysvetlení.
4. **Vyhodnotenie kombinovaných XAI metód:** Bola porovnaná výkonnosť navrhutej metódy AMS Ensemble s existujúcimi kombinovanými prístupmi (NormEnsembleXAI, AutoWeighted) a jednotlivými XAI metódami (Vanilla Gradients, Integrated Gradients, SHAP, Guided Grad-CAM), najmä z hľadiska zlepšenia robustnosti a dôveryhodnosti vysvetlení na dátovej sade ImageNet.

Všetky experimenty zahŕňali vizuálnu analýzu generovaných výstupov XAI metód (salienčné mapy) a systematické kvantitatívne merania pomocou knižnice Quantus.

Experimentálne výsledky potvrdili viacero kľúčových hypotéz. Ukázalo sa, že kvalita skúmaného modelu strojového učenia významne ovplyvňuje zmyslupnosť a spoľahlivosť generovaných vysvetlení; nedostatočne alebo nesprávne natrénované modely produkujú neinformatívne, prípadne zavádzajúce atribúcie, pretože metódy vysvetliteľnosti reflektujú inherentné vlastnosti daného modelu. Takéto vysvetlenia môžu byť stále využiteľné napríklad na odhalenie chýb a súvisiacich problémov modelu. Porovnanie jednotlivých XAI metód umožnilo určiť ich špecifické charakteristiky, napríklad metódy ako Integrated Gradients a SHAP konzistentne poskytovali koherentnejšie a dôveryhodnejšie vysvetlenia v porovnaní s jednoduchými metódami založenými na gradientoch (Vanilla Gradients), aj keď ich relatívna výkonnosť sa mohla líšiť v závislosti od dátovej sady a zvolených kvantitatívnych metrík. Testy robustnosti preukázali, že mnohé štandardné metódy sú citlivé na šum a adversariálne perturbácie vstupov, čo zdôrazňuje potrebu robustnejších prístupov.

Navrhnutá metóda AMS Ensemble preukázala v experimentoch schopnosť efektívne zlepšiť vlastnosti vybraných metód XAI. Vďaka adaptívnemu váhovaniu, ktoré zohľadňuje aktuálnu výkonnosť jednoduchých metód na konkrétnom dátovom vstupe, bola metóda AMS Ensemble schopná potlačiť artefakty a šum z menej spoľahlivých jednotlivých metód XAI a vyťažiť silné stránky robustnejších techník. Výsledné kombinované vysvetlenia dosahovali lepšie výsledky vo viacerých kvantitatívnych metrikách (napr. vernosť, zložitosť, lokalizácia, randomizácia) v porovnaní s jednotlivými skúmanými metódami.

Diplomová práca prináša okrem uceleného teoretického prehľadu problematiky vysvetliteľnej umelej inteligencie so zameraním na obrazové dáta, aj konkrétny metodologický prínos vo forme AMS Ensemble. Empirické výsledky podporujú záver, že kombinácie rôznorodých prístupov k XAI a adaptívne stratégie agregácie predstavujú sľubnú cestu k vývoju transparentnejších, dôveryhodnejších a v konečnom dôsledku bezpečnejších systémov strojového učenia. Práca prispieva k prebiehajúcemu úsiliu o demystifikáciu “čiernych skriniek” a posilnenie dôvery v pokročilé technológie strojového učenia.

Methods Supporting Interpretable and Explainable Machine Learning

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of prof. Ing. Lukáš Sekanina, Ph.D. I have listed all the literary sources, publications, and other sources that were used during the preparation of this thesis.

.....
Šimon Šmída
May 21, 2025

Acknowledgements

I would like to express sincere gratitude to my supervisor, prof. Ing. Lukáš Sekanina, Ph.D., for his support, valuable feedback, and open-minded approach. I am especially thankful that he found the proposed topic worth pursuing and that he provided thoughtful guidance throughout the development of this thesis. The work has given me the opportunity to deeply explore a field I am passionate about, has significantly expanded my understanding of it, and has opened up new possibilities for future research and potential doctoral studies. This work was also supported by the Czech Science Foundation project 24-10990S.

Contents

1	Introduction	7
2	Interpretability and Explainability in Machine Learning	8
2.1	Core Definitions	9
2.2	Taxonomy of Interpretability Methods	10
2.3	Importance and Applications	11
2.4	Challenges in Interpretability	12
2.5	Societal and Ethical Implications	14
3	Intrinsic Interpretability Methods	15
3.1	Linear Models	15
3.2	Generalized Additive Models	17
3.3	Decision Trees	17
3.4	Strengths and Limitations	18
4	Post-hoc Interpretability Methods	19
4.1	Taxonomy of Post-hoc Interpretability Methods	19
4.2	Model-Agnostic Methods	20
4.3	Model-Specific Methods	26
5	Mechanistic Interpretability	29
5.1	Reverse-engineering Neural Networks	29
5.2	Features as model building blocks	30
5.3	Monosemantic vs. Polysemantic Neurons	32
5.4	Circuits Analysis	32
5.5	Comparison to Intrinsic and Post-hoc Methods	34
6	Proposed extension: XAI Ensembles	36
6.1	Why ensemble explanations?	36
6.2	Ensemble Techniques	37
6.3	Limitations that motivate a new method	38
6.4	Proposed Extension: Adaptive-Metric Softmax (AMS)	39
7	Evaluation of the Explanations	43
7.1	Qualitative Evaluation: Interpretability	43
7.2	Quantitative Evaluation: Completeness	44
7.3	Quantus: XAI Toolkit	48
7.4	Challenges in Explanation Evaluation	48

8	Experimental Analysis	49
8.1	General Experimental Setup	49
8.2	Experiment 1: Model Quality vs. Explanation Quality	52
8.3	Experiment 2: Comparative Analysis of XAI Methods	58
8.4	Experiment 3: Robustness of Explanations	67
8.5	Experiment 4: Enhancing Explanations via Ensembles	73
9	Conclusion	78
	Bibliography	79
A	CNN Architecture (MNIST)	85
A.1	Dataset and Pre-processing	86
A.2	Training Configurations	86
B	Input Invariance Test Statistics	87
C	Experiment 4: Visualizations	88

List of Figures

2.1	The proposed concept of XAI by DARPA (adapted from [22]).	8
2.2	Overview of ML interpretability methods, categorized into intrinsic methods and post-hoc methods, which are further divided into model-agnostic and model-specific approaches. Inspired by [42].	10
2.3	Is there a trade-off between the performance and interpretability of ML models? Adapted from [67].	12
3.1	Feature contributions in a linear regression model: Coefficient values β_i are represented as bars, where positive contributions (blue) increase the model's prediction, and negative contributions (red) decrease it.	16
4.1	Explanation of an image classification prediction made by Google's Inception neural network using LIME. The top three predicted classes: Electric Guitar ($p = 0.32$), Acoustic guitar ($p = 0.24$) and Labrador ($p = 0.21$), from [52].	21
4.2	Overview of SHAP methods, including model-agnostic approaches such as Kernel SHAP, and model-specific methods including Linear SHAP, Low-Order SHAP, Max SHAP, and Deep SHAP as introduced in [37], with Tree SHAP added later in [38].	24
4.3	Deep SHAP applied to the VGG16 vision model [59] for classifying the input image. The trained network showed uncertainty between two classes, "Dowitcher" and "Red-backed Sandpiper", both belonging to the same family (<i>Scolopacidae</i>) in scientific classification. The SHAP values highlight the specific features that contributed to each prediction. Reproduced from [37].	24
4.4	Vanilla Saliency Map: Visualization of pixel importance using the gradient of the class score with respect to input pixels (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].	26
4.5	Smooth Saliency Map: An enhanced version of the vanilla saliency map reduces noise by averaging gradients computed over multiple slightly perturbed versions of the input (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].	27
4.6	Integrated Gradients: Attribution map (right) shows the cumulative feature importance across the input image. The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].	27

4.7	Grad-CAM: Class activation map generated using gradient-weighted importance of feature maps in a convolutional layer (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].	28
5.1	Feature visualization by optimization: an input image (initialized with random noise) is iteratively updated to maximally activate a targeted neuron (InceptionV1 [66], layer <code>mixed4a</code> , unit 11). From Olah et al. [48], CC BY 4.0.	30
5.2	Progressive feature development across layers of InceptionV1 [66] trained on ImageNet [14]. Early layers detect edges and textures, while deeper layers capture object parts and high-level concepts. Adapted from [48], CC BY 4.0.	31
5.3	Circuit-based composition of a car detector in InceptionV1 [66] neural network. This visualization illustrates how a high-level Car Detector (<code>4c:447</code>) emerges from lower-level feature detectors through learned connections. The Windows (<code>4b:237</code>), Car Body (<code>4b:491</code>), and Wheels (<code>4b:373</code>) units contribute excitatory (red) and inhibitory (blue) signals, forming a neural circuit that integrates these components into a coherent car representation. From Olah et al. [47], CC BY 4.0.	33
6.1	Visualization of the NormEnsembleXAI method. From [27].	37
8.1	Underfitting model: Incorrect prediction for a single MNIST test sample (true label 7). From left to right: (1) original 28×28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to [0,1] range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The map lacks structure, is noisy, and fails to highlight task-relevant regions.	53
8.2	Overfitting model: Misclassifies digit 7 as 3. From left to right: (1) original 28×28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to [0,1] range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The saliency map is sharper but highlights misleading features, reflecting memorized patterns, not a generalizable structure.	54
8.3	Properly trained model: Correctly classifies digit 7. From left to right: (1) original 28×28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to [0,1] range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The produced saliency map is less sharp than the map of the overfitting model (likely due to the smoother decision surface of the proper model), and highlights meaningful and task-relevant pixels.	54
8.4	Side-by-side comparison of saliency maps from underfitting and overfitting to properly-trained CNN. Heatmap attribution values are normalized to the [0,1] range, indicating each pixel’s influence on the predicted logit. High-performing models yield cleaner, more localized explanations.	55

8.5	Saliency map comparison across underfit, overfit, and properly trained models for four selected test samples (first column). Each row shows the original image and saliency overlays for the predicted class by each model. Heatmap attribution values are normalized to the $[0,1]$ range, indicating each pixel's influence on the predicted logit. Differences in attribution patterns reflect variation in learned representations and model reliability.	55
8.6	Radar plot comparing the explanation quality of underfitted, overfitted, and properly-trained CNN models using five quantitative metrics, based on Quantus [23]. Each axis represents a normalized score (0 to 1), with higher values indicating better performance.	56
8.7	DeepSHAP analysis of input digit 4, comparing a correct prediction (top) with an incorrect one (bottom) across ten digit classes (0-9). Red/blue indicate positive/negative attributions. In the top row, the model confidently predicts correct class 4 ($p = 1.00$); SHAP values highlight that the absence of a top connection (a feature typical for digit 9) strongly supports identifying this digit as a 4. In the bottom row, a slight variation in writing (presence of the top connection) confuses the model, resulting in a misclassification favoring digit 9 ($p = 0.55$) over digit 4 ($p = 0.45$).	59
8.8	Comparison of the selected attribution methods on a correctly classified MNIST digit 8. Columns show: Original input, Attributions for class 8 (correct), Attributions for class 3 (incorrect), Input with top 10% influential pixels (for class 8) masked. Red/blue indicate positive/negative attributions (normalized $[-1, 1]$). The attribution maps highlight important regions per method; the masked samples reveal the effect of removing them.	59
8.9	Attribution quality measured by the change in log-odds for the predicted class after masking the top 10% most salient pixels ($\Delta\log\text{-odds}$). Higher values indicate that more influential pixels were identified. Boxplots show distribution over 100 MNIST test samples for Vanilla Gradients, Integrated Gradients, and SHAP.	60
8.10	Violin plot showing the distribution of normalized pixel-wise attribution values for Vanilla Gradients, Integrated Gradients, and SHAP across $N = 100$ test samples (MNIST). The plot displays density estimates, with wider sections indicating higher concentrations of values. The thicker blue line indicates the mean.	61
8.11	Aggregated quantitative evaluation for Vanilla Gradients (Saliency), Integrated Gradients, and SHAP on MNIST over $N = 100$ samples. Values are scaled $[0, 1]$, where 1.0 (outer edge) indicates the best performance on that metric axis.	62
8.12	Qualitative comparison on ImageNet sample (three input images, column 1) for Vanilla Gradients (teal), Integrated Gradients (orange), GradientSHAP (blue). Attributions are normalized to $[-1, 1]$	64
8.13	Aggregated quantitative evaluation for Vanilla Gradients (Saliency), Integrated Gradients, and SHAP on ImageNet over $N = 100$ samples. Values are scaled $[0, 1]$, where 1.0 (outer edge) indicates the best performance on that metric axis.	65
8.14	Rank-based comparison of the results from quantitative evaluation analysis of the selected XAI methods. Higher rank means better relative performance.	66

8.15	Stability of selected XAI methods' explanations under Gaussian noise for a CNN prediction on an MNIST digit 8. Columns depict increasing noise levels (standard deviation σ , left to right from 0 to 1). The first row shows the input images as the noise is being introduced. Subsequent rows display explanation heatmaps for Vanilla Gradients, Integrated Gradients, GradientSHAP, and DeepSHAP, all generated with respect to the true class label. Visual degradation illustrates the sensitivity of each method to noise.	68
8.16	Comparison of explanation stability under Gaussian noise using SSIM. The plot tracks the mean SSIM scores (100 test samples, shaded areas indicate ± 1 standard deviation) for Vanilla Gradients, Integrated Gradients, GradientSHAP, and DeepSHAP. SSIM scores compare each method's explanation for the original MNIST input ($\sigma = 0$) against explanations generated under increasing noise levels (σ). Higher SSIM indicates greater structural similarity to the original explanation (higher stability).	69
8.17	Constant-Shift Invariance Test. Attribution maps for Vanilla Gradients, Integrated Gradients (zero/black baselines) and GradientSHAP (zero/black baselines) on MNIST before (top row) and after (bottom row) applying a uniform shift of $-1/\text{std}$ with bias compensation. Vanilla Gradients and all black-baseline methods remain identical under shift, whereas zero-baseline methods "leak" the uniform offset into their heatmaps.	71
8.18	Checkerboard-Pattern Attack. Attribution maps for the XAI methods on MNIST digit before (top row) and after (bottom row) adding a 2×2 checkerboard shift. None of the selected methods stays invariant; the checkerboard pattern "leaks" into attributions of all methods.	72
8.19	Qualitative comparison on ImageNet sample image (class 'tree frog') of singular XAI methods (Vanilla Gradients, Integrated Gradients, GradientSHAP, GuidedGradCAM) and ensembles (NormEnsembleXAI, AutoWeighted, and proposed AMS).	74
8.20	Qualitative comparison on three ImageNet sample images of ensemble methods (NormEnsembleXAI, AutoWeighted, and proposed AMS). AMS was able to exploit the localization property of GuidedGradCAM more effectively than the other ensembles.	74
8.21	Quantitative comparison on $N = 100$ ImageNet samples of singular methods (Vanilla Gradients, Integrated Gradients, GradientSHAP, GuidedGradCAM) and ensembles (NormEnsembleXAI, AutoWeighted, and proposed AMS).	75
C.1	Qualitative comparison of singular XAI methods on three ImageNet samples.	88
C.2	Comparison of both singular and ensemble methods on four ImageNet samples.	88

Chapter 1

Introduction

The emergence of Artificial Intelligence (AI) represents an important step in the evolution of computing, aiming to address the limitations inherent in traditional approaches [39]. Historically, practitioners solved individual problems by designing task-specific algorithms, incorporating explicitly defined rules and domain expertise. This approach is often straightforward for structured, well-defined tasks, but it struggles with ambiguous or difficult-to-formalize problems such as recognizing patterns in images or understanding human language [33]. To overcome these challenges, researchers drew inspiration from the structure and function of biological brains, and AI-based systems, such as Deep Neural Networks (DNNs), became capable of learning from data rather than relying on predefined rules.

This paradigm shift enabled significant progress in various fields, where models of machine learning (ML), a subset of AI, could be successfully used to solve problems previously thought to be infeasible. Nevertheless, this power comes at a cost. Even though humans design the model architectures and define the mathematics behind them, the complexity often obscures the reasoning behind their decisions. These systems have become “black boxes”, producing outputs that are difficult (even for experts) to interpret [11, 35].

The need to interpret and explain AI-related methods arises from two fundamental reasons. First, humans are naturally curious and seek to understand how things work. Early investigations into neural networks (NNs) revealed that vision models learn representations such as edges and simple shapes, progressively combining them into complex objects aligned with those in biological vision systems, offering insights into both AI and human cognition [28, 72]. Second, understanding AI’s decision-making is critical for safety, trust, and accountability, especially in high-stakes domains like healthcare and finance, which are increasingly regulated for fairness and transparency [12]. In this way, humans can reduce risks and build confidence in results.

This thesis aims to investigate and advance methods that enhance the interpretability and explainability of machine learning models. It begins with a theoretical foundation covering core concepts, motivations, and taxonomies in Chapter 2. Subsequent chapters present a structured review of established techniques, focusing on intrinsically interpretable models (Chapter 3) and post-hoc methods (Chapter 4), with a dedicated chapter on mechanistic interpretability (Chapter 5). A novel extension, an ensemble-based approach, is proposed and described in Chapter 6. Selected methods are implemented and rigorously evaluated using both qualitative and quantitative assessments outlined in Chapter 7. The experimental analysis in Chapter 8 examines the strengths, limitations, and behavior of selected methods under diverse conditions. Finally, Chapter 9, summarizes the findings, discusses their implications, and outlines promising directions for future research.

Chapter 2

Interpretability and Explainability in Machine Learning

Machine learning systems continue growing ever more complex in research and real-world applications. The complexity naturally raises the question of how to understand and explain the behavior of such systems, especially in critical domains. This chapter offers a view on the foundational concepts of interpretability and explainability, exploring proposed definitions, societal impact, and challenges of this relatively new field. The fundamentals enable further discussions of specific methods and their applications in the subsequent chapters.

XAI. The term eXplainable Artificial Intelligence (XAI) became widely known following the establishment of the Defense Advanced Research Projects Agency (DARPA) XAI program in 2015. It was motivated by growing concerns over the black-box nature of high-performance yet obscure models (particularly DNNs). The program aimed to develop methods that combine accuracy with understandable, trustworthy explanations (see [Figure 2.1](#)). The official Broad Agency Announcement (DARPA-BAA-16-53)¹, released in 2016, clearly outlined these goals and led to significant research efforts.

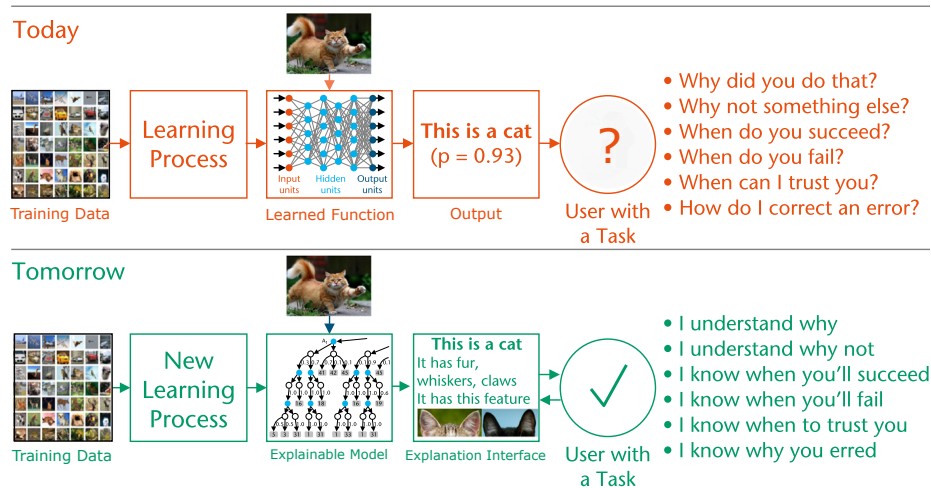


Figure 2.1: The proposed concept of XAI by DARPA (adapted from [22]).

¹DARPA XAI program announcement: <https://www.darpa.mil/attachments/DARPA-BAA-16-53.pdf>

Although DARPA popularized the term XAI, efforts toward interpretable AI models precede this initiative, dating back to rule-based expert systems and medical diagnostic systems developed in the 1970s and 1980s. DARPA’s program, however, provided renewed momentum and visibility, encouraging a broad spectrum of academic and industrial research focused explicitly on interpretability and explainability. This resulted in numerous frameworks, practical toolkits, and standards aimed at providing clear, actionable insights into model behavior, ultimately enabling stakeholders from diverse backgrounds to understand, trust, and effectively interact with AI systems [29, 35, 42].

2.1 Core Definitions

Interpretability and explainability are often used interchangeably, yet they encompass subtly different concepts essential for understanding ML systems. The definitions for these terms vary across the literature and are not standardized, which leads to ambiguity [21].

Interpretability generally refers to the ability to understand the *internal mechanisms* of a model. This view aligns closely with Lipton’s notion of *simulatability*, the property of a model being sufficiently simple for a human to follow its reasoning entirely [35]. Examples of interpretable models include linear regression (Section 3.1) or decision trees (Section 3.3), whose decision-making processes are transparent and straightforward to follow.

Explainability, in contrast, focuses on making a model’s *behavior* understandable. Explainability methods, such as LIME [52] and SHAP [37] (both discussed in Chapter 4), produce insights into model decisions without necessarily clarifying internal mechanisms [36]. Thus, explainability methods often result in simplified approximations and emphasize comprehensibility over completeness.

In summary, the key questions distinguishing these terms are:

- **Interpretability:** “How does the model work internally?”
- **Explainability:** “Why did the model make this decision?”

Transparency refers explicitly to the structural clarity of the model, denoting the degree to which its operations and structure can be directly inspected and understood [35]. Transparency is intrinsic to the model design itself.

Comprehensibility describes how effectively explanations provided by a model align with human cognitive abilities. A transparent model might still be incomprehensible if it involves too many features or intricate relationships that surpass human cognitive limits [36]. Models optimized for comprehensibility prioritize human cognitive constraints, thus often favoring simplified explanations.

But what if the model is orders of magnitude more complex, with billions of trainable parameters, as in deep neural networks? Here, *post-hoc* explanations (discussed in Chapter 4) become essential. They provide approximations of the model behavior *after* training. Although these explanations might lack completeness (they do not fully describe internal mechanisms), they still offer valuable insights into specific predictions, supporting the trust of users.

2.2 Taxonomy of Interpretability Methods

Interpretability methods can be broadly classified along three primary dimensions [42]:

1. **Intrinsic vs. Post-hoc** Interpretability: Is interpretability built into the model or applied after training?
2. **Model-Specific vs. Model-Agnostic** Methods: Does a method rely on internal model details or treat the model as a black box?
3. **Global vs. Local** Interpretability: Do the explanations cover an overall model or a single prediction?

These divisions are *not* mutually exclusive; they can overlap. For instance, SHAP is typically considered a post-hoc, model-agnostic method, but its output can be used for both local (individual instance) and global (dataset-wide) explanations. Figure 2.2 offers a rough visual overview of the techniques and shows how different types relate to each other.

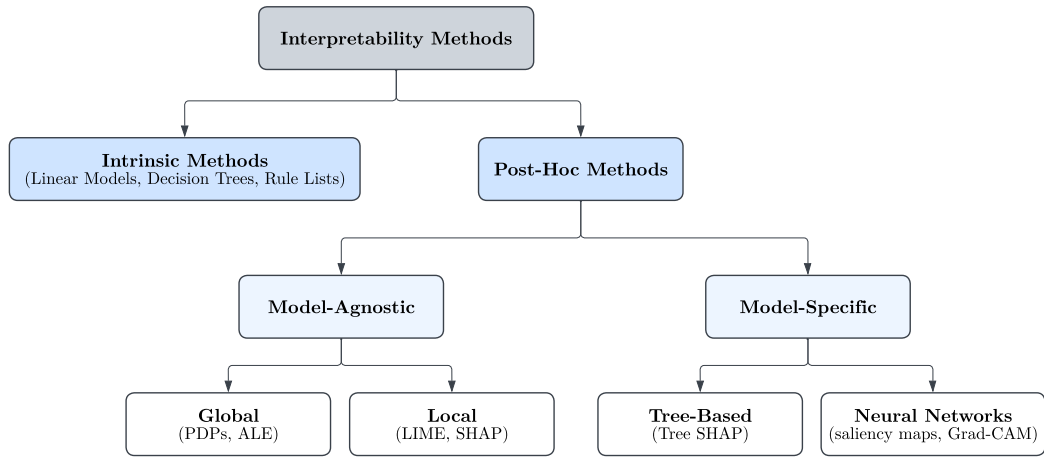


Figure 2.2: Overview of ML interpretability methods, categorized into intrinsic methods and post-hoc methods, which are further divided into model-agnostic and model-specific approaches. Inspired by [42].

2.2.1 Intrinsic vs. Post-hoc Interpretability

Intrinsic Interpretability. This category refers to models that are *inherently understandable*, featuring simple and transparent structures. Such models are deliberately *designed to be interpretable* from the ground up. This allows users to comprehend how inputs are transformed into outputs. An example is a decision tree model where the decision-making process can be directly visualized and followed step-by-step. More detailed analysis of intrinsic methods can be found in Chapter 3.

Post-hoc Interpretability. These methods involve applying interpretability methods to more complex models *after* training. The methods aim to extract explanations without altering the original model, for example, by using LIME [52] to explain the predictions of a DNN. For a closer examination of post-hoc techniques, see Chapter 4.

2.2.2 Model-Specific vs. Model-Agnostic Methods

Model-Specific Methods. These techniques are used to interpret particular types of ML models by utilizing their internal structures and properties. For example, one can visualize gradient-based saliency maps or use methods like Grad-CAM to highlight important pixels in an input image, providing insights into which regions influence the model’s decisions. Further information on model-specific methods can be found in [Section 4.3](#).

Model-Agnostic Methods. This category can be applied to *any* ML model, treating it as a black box, which is possible because these methods do not rely on model architecture or its parameters. For example, SHAP values, computed using Kernel SHAP, can explain any model’s predictions by quantifying the contribution of each input feature to the final prediction through a game-theoretic approach [37]. See [Section 4.2](#) for more insights into model-agnostic methods.

2.2.3 Global vs. Local Interpretability

Global Interpretability. Global methods aim to characterize the model behavior across the entire input space. They provide insights into overall decision-making logic by analyzing feature importances, decision boundaries, or surrogate models trained on full model outputs. These methods answer: *“How does the model generally make decisions?”*

Local Interpretability. Local methods aim to explain the model’s behavior in the neighborhood of a specific input. They provide per-instance attributions that highlight which features were most influential for a particular prediction. For example, LIME [52] fits an interpretable model locally around a single data point to approximate the black-box model’s decision. These methods answer: *“Why did the model make this prediction?”*

2.3 Importance and Applications

Machine learning models have been widely applied across various areas in our everyday lives, and oftentimes, the decisions they make can have significant consequences. In healthcare, for instance, understanding why a model predicts a certain diagnosis can give clinicians additional and potentially highly important information, ensuring accountability and improving patient outcomes [2]. Similarly, in finance, regulators require explainable credit decisions to detect discrimination and maintain trust in automated lending [69]. Autonomous systems, such as self-driving cars, also benefit from interpretable components (e.g., rule-based modules or explanation interfaces) to diagnose failures or near-miss events and improve safety protocols [5].

Beyond these high-stakes scenarios, transparency allows users to build trust in such systems. Users are more likely to rely on automated systems if they can understand or challenge the reasoning behind a model’s output. However, interpretability is not universally necessary. In low-risk domains, such as product recommendations, spam detection, or predictive maintenance, the emphasis often shifts toward optimizing performance metrics rather than ensuring model transparency. In such cases, black-box models may be acceptable, provided their impact on users is minimal and well understood [42, 20].

2.4 Challenges in Interpretability

The main challenges in ML interpretability arise from the sheer complexity of specific models. DNNs are often referred to as “black-box” models because their decision-making process is currently considered incomprehensible, despite experts in the field understanding the mathematics behind their architectures and learning processes.

Complexity of Models vs. Simplicity of Explanations. Models such as DNNs operate in high-dimensional spaces, often involving billions of parameters [10, 25]. It is difficult to follow through on how these parameters interact to produce a certain prediction. The count of the parameters and the number of their interactions is just too large. The key factors that contribute to the “black-box” nature of these models include the following:

- **High Dimensionality:** As the number of parameters grows, it becomes harder to interpret how they interact and influence the model’s behavior. This phenomenon is often referred to as *the curse of dimensionality*.
- **Non-linearity:** The models typically employ non-linear transformations, as it helps them capture complex patterns hidden in the data, but it also makes the relationship between inputs and outputs less straightforward.
- **Distributed Representations:** In DNNs, information is scattered across multiple neurons and layers, with no single neuron being solely responsible for a decision.
- **Emergent Behavior:** During training, the models learn patterns and relationships from data that were not explicitly programmed. This emergent behavior may lead to unexpected outcomes, complicating explanations for specific decisions.

Performance-Interpretability Trade-off. Increasing a model’s predictive performance often comes at the expense of interpretability [35]. Figure 2.3 provides an illustrative depiction of this trade-off to convey the general intuition.

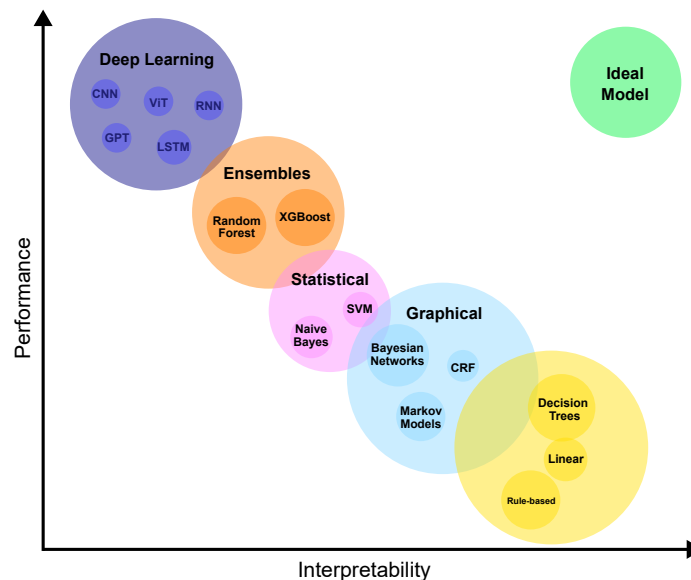


Figure 2.3: Is there a trade-off between the performance and interpretability of ML models? Adapted from [67].

The more complex the model is, the more accurate predictions it typically produces for certain applications, but it comes at the cost of our ability to understand *why* or *how* the model made a particular prediction. For example, deep learning models often outperform simpler models in tasks such as image recognition, natural language processing, and complex time-series prediction. Interpretable models like linear regression or decision trees may provide transparency, but often do not perform as well on complex tasks or datasets.

However, this view of an inevitable trade-off is not obvious and not generally agreed on. It has been challenged by several researchers, including Cynthia Rudin et al. [54], who argue that, in many cases, especially for tabular data, simpler models can achieve comparable results to black-box models. Rudin suggests that the field of interpretable AI should focus on inherently interpretable models and use black-box models only as benchmarks for performance. By doing so, inherently interpretable models could be developed with the same level of accuracy as the black-box counterparts, providing both transparency and performance in critical applications.

This debate resonates with R. Sutton’s *The Bitter Lesson* [65], which emphasizes that general, computation-driven methods often outperform those relying on human ingenuity or domain-specific design. It explains the dominance of complex models, but Rudin and others argue that interpretable models *can* still achieve comparable performance, especially when domain knowledge is effectively utilized. This balance between scalability and interpretability remains a key challenge for the AI research community.

Evaluation of Interpretability Methods. Evaluating interpretability methods is another significant challenge in the field. How do we *measure* interpretability? What would it even mean? The interpretations or explanations of some concepts can mean different things to different people. There is no universally accepted framework for measuring how interpretable a model or explanation is. As mentioned in [Chapter 2](#), there are also no generally agreed-on definitions of the terms *interpretability* and *explainability* in AI. In the absence of a ground truth for what constitutes a “correct” explanation, the evaluation of interpretability remains an open problem [42]. Doshi-Velez and Kim [16] propose a taxonomy for evaluating interpretability at three levels: *application-grounded* (real tasks with domain experts), *human-grounded* (simplified tasks with lay users), and *functionally-grounded* (proxy metrics such as fidelity or complexity). This emphasizes that evaluation is highly context-dependent.

Generally, it is agreed that the explanations provided by a method should exhibit *fidelity*, meaning the method should accurately reflect how the model made a decision. However, higher fidelity often comes at the cost of increased complexity, reducing interpretability. As models grow in complexity, explanations must increasingly compress and abstract information to remain comprehensible, creating a trade-off between fidelity and simplicity. This compression (inherently lossy) highlights the challenge of capturing sufficient detail without overwhelming the user [56]. Methods like LIME or SHAP (discussed in [Section 4.2](#)) provide local explanations that approximate the behavior of the original model. However, a simpler explanation may abstract important details, reducing fidelity to the model’s true behavior. Beyond fidelity, other desirable properties include *stability* (similar inputs should yield similar explanations) and *comprehensibility* (the explanation should align with human intuition and domain knowledge). Since interpretability is primarily intended for human understanding, user studies and human-centered evaluation metrics are important for assessing whether explanations are actually helpful in decision-making.

2.5 Societal and Ethical Implications

The growing use of machine learning in sensitive areas also raises questions of fairness, bias, and accountability. Decisions informed by uninterpretable ML systems, whether in hiring processes, credit scoring, or criminal justice, may result in outcomes that reflect and amplify existing societal biases present in the training data or model design [40]. Interpretability methods can help to identify and rectify such biases and provide insights into the decision-making process (e.g., how specific features influence model outcomes).

Moreover, legal frameworks, such as the European General Data Protection Regulation (GDPR) [19], underscore the need for transparency. Article 22 of the GDPR suggests that individuals have the right not to be subject to solely automated decisions without an explanation of the logic involved. By providing interpretable insights into model decisions, organizations can align with ethical guidelines and legal requirements, building trust and legitimacy in AI-driven systems.

However, it is important to keep in mind that current interpretability methods alone do not guarantee fairness or unbiased outcomes. Interpretability is a tool, not a cure-all. When used properly, such methods can reveal potential discrimination or error patterns, but they can also mislead or cause users to overestimate the reliability of the explanations. Researchers warn of *fairwashing*, where the presence of a superficial explanation gives a false impression of fairness, masking underlying biases or unjust behavior [4]. Furthermore, explanations themselves can be manipulated by adversaries or misinterpreted by end-users, leading to incorrect assumptions about model behavior or validity [61].

In summary, interpretability is key to ensuring the reliability and ethical use of machine learning, especially in critical or regulated areas. This chapter laid the groundwork by defining core concepts, categorizing methods, and addressing major challenges and societal implications. The next step is a deeper exploration of specific interpretability methods. The following chapter introduces *intrinsic* approaches designed to be transparent by nature, followed by *post-hoc* and *mechanistic* techniques in subsequent chapters.

Chapter 3

Intrinsic Interpretability Methods

Intrinsic interpretability refers to machine learning models that are transparent by design. Their structure makes it possible to directly understand how inputs influence outputs, which is especially valuable in high-stakes domains such as healthcare, finance, and law, where trust, accountability, and compliance are critical.

These models prioritize simplicity and structural clarity, which enables easier debugging, integration of domain knowledge, and user trust. However, this often limits their capacity to model complex, nonlinear relationships in high-dimensional data, leading to a trade-off between interpretability and predictive performance (discussed in [Section 2.4](#)).

While there is no strict boundary between simple and complex models, a useful guideline is whether the decision function can be fully described without omitting key details. When models become too intricate (due to size, feature interactions, or opaque representations), they require post-hoc explanations instead.

This chapter introduces core families of intrinsically interpretable models, including linear models, decision trees, and Generalized Additive Models (GAMs), and discusses their interpretability mechanisms, strengths, limitations, and applications.

3.1 Linear Models

A **linear model** assumes a linear relationship between the input features (independent variables) and the target (dependent variable). The general form of a linear model for predicting an output y is represented by the following equation:

$$y = \beta_0 + \sum_{i=1}^n \beta_i x_i \quad (3.1)$$

where β_0 is the constant term (intercept), β_i are the coefficients (weights) of the input features x_i , and n is the number of features.

Interpretability. Each coefficient β_i reflects the strength and direction of the feature's contribution to the target variable y . Specifically, the *sign* indicates a positive or negative impact, while the *magnitude* represents the feature's overall influence. Because of these properties, linear models are considered to be *transparent*, as the relationship between inputs and outputs is explicitly encoded in the model itself.

3.1.1 Linear Regression

A typical example of linear models is *linear regression*. It is a widely used statistical method for modeling the relationship between a dependent variable y and one or more independent variables x_i [44]. The objective of the method is to determine the optimal weights (β_i) for input features (x_i) that minimize the prediction error (loss). The resulting weights offer insights into individual feature importance. To use linear regression for prediction tasks, one can directly follow the general equation for linear models (see Equation 3.1).

Example: To demonstrate the interpretability of the linear regression model, consider a model with three features x_1, x_2, x_3 , and the following equation:

$$\text{Prediction: } y = 2.5 - 1.5x_1 + 1.0x_2 - 0.5x_3 \quad (3.2)$$

where:

- $\beta_0 = 2.5$: Intercept, the predicted value of y when all features (x_1, x_2, x_3) are zero.
- $\beta_1 = -1.5$: One-unit increase in x_1 *decreases* the predicted value of y by 1.5, holding all other features constant.
- $\beta_2 = 1.0$: One-unit increase in x_2 *increases* the predicted value of y by 1.0, holding all other features constant.
- $\beta_3 = -0.5$: One-unit increase in x_3 *decreases* the predicted value of y by 0.5, holding all other features constant.

The coefficients β_i in the Equation 3.2 illustrate how feature contributions (shown visually in Figure 3.1) determine the model's predictions.

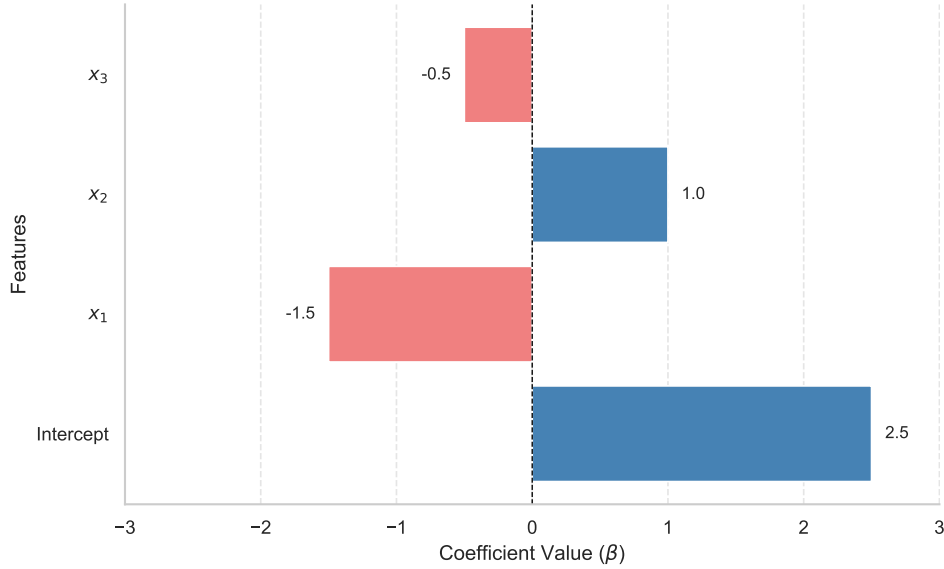


Figure 3.1: Feature contributions in a linear regression model: Coefficient values β_i are represented as bars, where positive contributions (blue) increase the model's prediction, and negative contributions (red) decrease it.

In [Equation 3.2](#), the coefficients (β_i) provide *global interpretability* by showing the effects of the features across the entire dataset. For *local interpretability*, the same weights can explain an individual prediction (e.g., y for a specific instance).

Limitations. Practitioners value linear models for their simplicity and offer a level of interpretability, but because of their linear nature, these methods are limited to specific use cases. For example, they underperform when the relationship between the input features and the target variable is inherently nonlinear. Additionally, linear models tend to fail when applied to datasets with intricate feature interactions where the effect of one feature on the target variable can depend on the values of other features.

3.2 Generalized Additive Models

Generalized Additive Models (GAMs) are a class of models that extend linear models (discussed in [Section 3.1](#)) by allowing for *non-linear* relationships between the input features (x_i) and the target variable (y), thus replacing linear terms $\beta_i x_i$ with functions $f_i(x_i)$. The output y of the model is predicted as the sum of *smooth* functions of the input features:

$$y = \beta_0 + \sum_{i=1}^n f_i(x_i) \quad (3.3)$$

where:

- β_0 is the intercept,
- $f_i(x_i)$ are smooth functions (e.g., splines) representing the effect of each feature x_i ,
- n is the number of features.

The additive structure of the model ensures that each feature’s contribution can be independently examined. In this way, GAMs allow the capture of non-linear relationships while maintaining interpretability.

3.3 Decision Trees

Decision trees are machine-learning models representing individual algorithmic decisions and their possible consequences in a tree-like structure. The model recursively splits the input data based on feature values. This way, the decision tree forms a series of decision rules leading to the predicted output.

Each node applies a decision rule on a feature (e.g., “is x_i less than some value θ ?”), and based on the outcome, the data follows a branch to either the left or right child node. This splitting continues until a leaf node is reached, which outputs a prediction. What makes this a typical, inherently interpretable model is that the mentioned decision process can be visualized in an intuitive, directly understandable way that is natural to follow. Of course, if a tree grows too deep (too many rules), it can become as inscrutable as a black-box.

Decision trees also address the problem of linear models, which lies in their tendency to fail in situations where the relationship between the features (x_i) and the prediction (y) is not linear or where there is interaction among the individual features [\[42\]](#).

3.4 Strengths and Limitations

Inherently interpretable models represent the desired set of properties that people want from machine learning systems. If a model is designed with the goal of being understandable while maintaining high performance in specific applications, it can be considered a success. Cynthia Rudin’s work [54] demonstrates that interpretable models can achieve performance comparable to the black-box models (particularly in structured data settings). However, these models are not without limitations, as they often struggle with unstructured data or complex relationships, and oftentimes, for real-world problems, they themselves can grow complex beyond straightforward understanding.

Strengths:

- **Direct Interpretability:** Because these models are inherently simple (e.g., linear models, decision trees, or rule-based systems), their internal logic can be inspected and followed directly without *post-hoc* techniques (discussed in [Chapter 4](#)).
- **Transparency and Trust:** Stakeholders can more easily trust decisions made by a model whose outputs can be traced back to human-readable features or rules. This is especially beneficial in high-stakes or regulated domains (e.g., healthcare or finance).
- **Ease of Debugging and Validation:** If predictions seem incorrect, practitioners can quickly identify which features or rules likely caused the decision, speeding up model troubleshooting [54].
- **Seamless Incorporation of Domain Knowledge:** Rule-based systems and simple interpretable models allow domain experts to inject constraints or insights directly, ensuring the model aligns with real-world requirements or ethical guidelines [12].

Limitations:

- **Reduced Predictive Power in Complex Tasks:** As problem complexity grows (e.g., high-dimensional data, intricate relationships), purely interpretable models may underperform compared to more complex “black-box” algorithms.
- **Limited Capacity for Non-Linear Interactions:** Even though some methods like GAMs (see [Section 3.2](#)) partially address non-linearity, strictly linear or rule-based approaches often struggle with highly non-linear dependencies among features.
- **Overfitting with Large Rule Sets:** Even rule-based models or large decision trees can become incomprehensible, losing practical interpretability if they grow too complex or produce overly intricate rules.
- **Risk of Oversimplification:** In forcing a model to be simple, it might omit important nuances in the data, leading to poorer predictions or ignoring subtle patterns. Striking the right balance between simplicity and predictive accuracy remains an ongoing challenge.

Intrinsic interpretability is a step in the right direction and inspires the creation of new inherently interpretable models aiming to achieve similar performance as models considered uninterpretable. Currently, however, these methods still struggle when dealing with unstructured data (e.g., images or text). This is where other approaches are needed, and one of them, *post-hoc* interpretability, is described in the following chapter.

Chapter 4

Post-hoc Interpretability Methods

Post-hoc interpretability is an approach that offers techniques that are applied *after* an ML model has been trained. These methods focus on providing insights into the model’s predictions and behavior behind the decision-making process without modifying the inner model structure. This contrasts with the inherently interpretable models (discussed in [Chapter 3](#)), developed to be interpretable by design. Post-hoc methods are specifically valuable because they address the tension between increasing model complexity and the growing demand for transparency [\[35\]](#). Depending on whether the focus lies on the model’s internal structure or solely on its input-output behavior (disregarding the internal mechanisms entirely), the post-hoc methods can be categorized into *model-agnostic* and *model-specific* approaches [\[42\]](#), which are discussed further in this chapter (see [Section 4.2](#) and [Section 4.3](#) respectively).

4.1 Taxonomy of Post-hoc Interpretability Methods

As mentioned in [Chapter 2](#), there is not a clear, single categorization of the existing methods in the field of interpretable and explainable AI. This thesis will stick to the largely agreed-on categories and their properties. Following Christoph Molnar’s *Interpretable Machine Learning* [\[42\]](#) post-hoc methods can be structured based on their scope and dependency on the model structure:

Scope of Interpretability. Post-hoc methods can be categorized by their scope:

- **Local** methods focusing on explaining individual predictions (e.g., LIME, SHAP)
- **Global** methods aiming to describe overall model behavior (e.g., computing overall feature importance scores via permuting inputs, or training a global surrogate decision tree to mimic the entire model)

Model Dependency. Based on their dependency on the model structure:

- **Model-agnostic** methods working with any model (e.g., SHAP, LIME)
- **Model-specific** methods tailored to specific architectures (e.g., Grad-CAM for CNN)

4.2 Model-Agnostic Methods

Model-agnostic methods offer a flexible approach to interpreting various ML models. Unlike previously described intrinsically interpretable models (see [Chapter 3](#)) or model-specific techniques (discussed in [Section 4.3](#)), model-agnostic methods are not tied to a particular model type or structure. Instead, they are designed to work with *any* ML model, regardless of its architecture [52]. Their versatility can be particularly useful in practical settings when evaluating the interpretability of diverse models. They allow consistent assessment across different types without being constrained by model-specific techniques. These approaches do not require peeking into the model’s internal parameters; they only need the ability to query the model’s outputs for given inputs. They often operate by observing how changes in input affect the output, thus inferring the importance of various features.

4.2.1 LIME

Local Interpretable Model-agnostic Explanations (LIME) was introduced in 2016 by Ribeiro et al. [52]. LIME was designed to explain individual predictions of ML models, regardless of the underlying architecture. The term “**Local**” refers to LIME’s focus on providing explanations for a specific input instance rather than explaining the global behavior of the model on a dataset as a whole. The local approximations generated by LIME can help find insights about why the model made a particular prediction for a given input. The “**Model-agnostic**” part means that LIME can be applied to almost *any* type of ML model, ranging from decision trees and ensemble methods to neural network models, as discussed in [Section 4.2](#). LIME approximates the local decision boundary of the original (black-box) model using a simpler, inherently interpretable **surrogate model** (such as linear regression or decision tree). This surrogate model acts as a proxy, as it provides an interpretable explanation of how the input features influenced the prediction.

Mechanism. The mechanics of LIME can be understood by understanding the core principles of this method, which can be summarized in three key steps:

1. **Perturbation of Input Data:** The process starts with a specific input that is expected to be explained. LIME perturbs (modifies) the input features in order to generate a group of similar instances (neighborhood) around the original input. The perturbed instances aim to simulate realistic variations of the input while preserving the local context of the input.
2. **Generating Predictions and Fitting a Surrogate Model:** The obtained perturbed data instances are consequently passed to the original (black-box) model to make corresponding predictions. LIME then fits the surrogate model (e.g., linear regression) to approximate the local behavior of the original model. The surrogate is trained using the perturbed data (inputs) and their black-box model’s predictions (targets, labels). In this way, LIME approximates the local decision boundary of the black-box model around the original input instance.
3. **Interpretation:** Depending on the selected surrogate model, the results can be analyzed and interpreted. For the linear regression model, the coefficients provide insight into the influence of each input feature on the prediction. Positive coefficients highlight features that increase the prediction, while negative coefficients correspond to decreasing contributions. For more details on inherently interpretable models see [Chapter 3](#).

Figure 4.1 illustrates an example of using LIME to generate sparse linear explanations for an image classification task performed by Google’s pre-trained Inception neural network. The method highlights areas with positive weights toward each of the top three predicted classes, providing insights into why the model made specific predictions. This visualization helps build trust in the classifier by demonstrating that its decisions align with human intuition, even when the top prediction is incorrect.

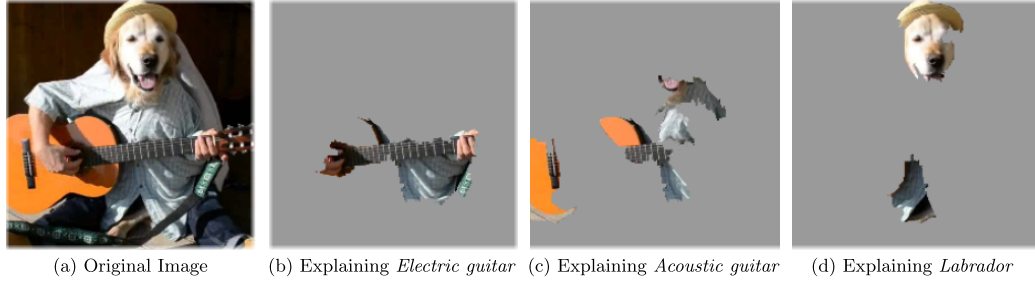


Figure 4.1: Explanation of an image classification prediction made by Google’s Inception neural network using LIME. The top three predicted classes: **Electric Guitar** ($p = 0.32$), **Acoustic guitar** ($p = 0.24$) and **Labrador** ($p = 0.21$), from [52].

Relevance to Interpretability. Ideally, LIME provides human-understandable explanations and insights for individual predictions by approximating the complex decision boundary of a black-box model with a simpler, more interpretable surrogate model (e.g, linear models, discussed in Section 3.1). Users can better understand which inputs influenced a specific decision the most by highlighting the *local contributions* of features.

Strengths:

- **Model-agnostic:** LIME is designed to explain predictions of *any* ML model.
- **Local Explanations:** Interpretable explanations are provided for each prediction.
- **Flexibility:** LIME works with various surrogate models, for example, linear regression (see Section 3.1), or decision trees (see Section 3.3).

Limitations:

- **Sensitivity to Perturbations:** The quality of explanations depends heavily on the choice of perturbations (a poor choice may lead to misleading explanations)
- **High-Dimensional Data:** LIME struggles with complex datasets containing many features, as fitting a surrogate model becomes computationally expensive, or less interpretable itself.
- **Local Validity:** The individual explanations are only valid locally. They may not generalize well to the entire dataset.

LIME introduced a flexible, model-agnostic approach for generating *local explanations* by approximating black-box models with more interpretable surrogate models. However, its reliance on perturbations and its focus on local fidelity can limit its robustness (particularly in high-dimensional settings). These shortcomings lead to the development of a more theoretically grounded framework, called SHAP [37], discussed next.

4.2.2 SHAP

SHAP, short for **SH**apley **A**dditive **eX**planations, is one of the most notable model-agnostic interpretability techniques. It was introduced by Lundberg and Lee in 2017 [37] to provide theoretically grounded and consistent explanations for machine learning models, aiming to address some of the limitations of earlier methods like LIME (discussed in [Section 4.2.1](#)).

SHAP unifies several existing interpretability methods based on specific properties into a class named **additive feature attribution methods**. These methods explain model predictions by assigning each feature a numeric contribution to the model’s output, mathematically defined as:

$$g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i \quad (4.1)$$

where:

- M is the total number of features
- z'_i represents the *presence* or *absence* of a feature in the simplified input representation
- ϕ_0 is the baseline value representing the model output when no features are present
- ϕ_i is the contribution of feature i to the prediction

The additive feature attribution method class ensures interpretability by summing contributions from all features to match the model’s output. The authors surprisingly discovered that the class contains a *single unique solution*, derived from the field of cooperative game theory and corresponding to Shapley values – the theoretical foundation of SHAP values. This solution satisfies three desirable properties: local accuracy, missingness, and consistency. They were derived and formalized from the principles of Shapley values [37] and had not been previously identified in relation to other additive attribution methods. Ensuring these properties, the method aligns not only with the model’s predictions but also provides *fair* and *consistent* feature attributions.

Local Accuracy ensures that the method’s explanation aligns with the model’s prediction for a given input. The *explanation model* $g(z')$, which uses simplified input z' , is designed to match the output of the *original model* $f(x)$ when z' corresponds to x . Formally, there is a mapping $x = h_x(z')$ such that:

$$f(h_x(z')) = g(z') = \phi_0 + \sum_{i=1}^M \phi_i z'_i \quad (4.2)$$

where:

- $f(x)$ is the original model’s prediction
- $g(z')$ is the explanation model approximating $f(x)$ in the simplified space
- $h_x(\cdot)$ maps the simplified input z' back to the original input x

The terms ϕ_0 , ϕ_i and z'_i are defined in the same way as in [Equation 4.1](#).

Local Accuracy requires two conditions:

1. The simplified input z' must properly map back to the original input x via $x = h_x(z')$.
2. The model's prediction and the explanation must match whenever z' represents x , i.e. $f(h_x(z')) = g(z')$.

When both conditions hold, $f(x)$ and $g(z')$ align exactly for that input, thus satisfying *local accuracy*.

Missingness property requires that if a feature is not present in the input ($z'_i = 0$), it should have no attribution impact ($\phi_i = 0$). It must not contribute to the explanation. In this way, only the *relevant* features are considered in the explanation.

Consistency property requires that if a feature becomes more important in the model, its attribution value must increase or remain unchanged, ensuring fairness. Formally, for any two models f and f' , if the marginal contribution of feature i under f' is at least as high as under f for all subsets, then the attribution value of feature i should not decrease:

$$\phi_0(f', x) \geq \phi_i(f, x) \quad (4.3)$$

Mechanism. **Shapley values** uniquely satisfy all the discussed properties, ensuring fair, consistent attributions aligned with the model's behavior. **SHAP values** are a practical adaptation of Shapley values for ML models. Shapley values originate from cooperative game theory, where a group of players collectively earns a reward that must be *fairly distributed* among them based on their individual contributions. The ML model's prediction represents the collective reward, while the features of the input act as the individual players who contribute to the reward. The Shapley value then ensures that each feature is assigned an attribution that is proportional to its unique contribution across all possible combinations of features. In this way, features influencing the prediction more significantly receive higher attributions, while less influential features are assigned lower attributions.

SHAP values adapt Shapley values to use conditional expectations for feature attributions in ML models, quantifying the contribution of each feature to the prediction. For a specific feature i , this contribution (ϕ_i) is determined by examining what happens when this feature is introduced to different combinations of other features. The calculation takes every possible group of features (subset S) and measures the **change in the model's prediction when feature i is added to that group**. Each of these changes is then assigned a **weight** based on the group size, creating a *fair* system that considers all possible ways features could be combined. Consequently, Shapley value can be defined as follows:

$$\underbrace{\phi_i}_{\text{contribution}} = \sum_{S \subseteq F \setminus \{i\}} \underbrace{\frac{|S|!(|F| - |S| - 1)!}{|F|!}}_{\text{weight}} \underbrace{[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]}_{\text{marginal contribution}} \quad (4.4)$$

where:

- ϕ_i represents the feature i 's **contribution** to the prediction
- The **weight** term accounts for all possible orderings of features
- The **marginal contribution** shows how much feature i changes the prediction

Computing Shapley values is, however, computationally expensive because the number of possible subsets of features grows exponentially with the number of features. For M features, this requires 2^M model evaluations, as this value represents the total number of subsets $|S|$ in Equation 4.4. This exponential growth makes the computation infeasible for models with a large number of features and led the authors of [37] to the development of multiple SHAP variants designed for greater efficiency, as shown in Figure 4.2.

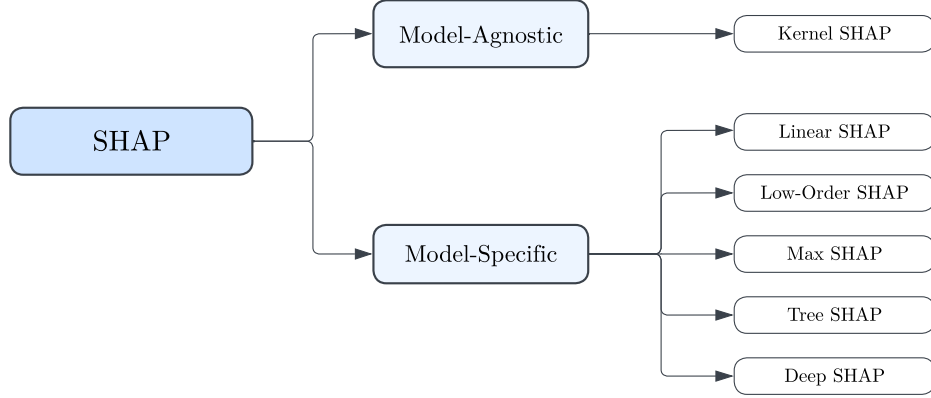


Figure 4.2: Overview of SHAP methods, including model-agnostic approaches such as Kernel SHAP, and model-specific methods including Linear SHAP, Low-Order SHAP, Max SHAP, and Deep SHAP as introduced in [37], with Tree SHAP added later in [38].

Deep SHAP, for instance, provides valuable insights into the decision-making processes of DNN models. In vision tasks, where the model predicts the class of a given input image, uncertainty can arise when multiple classes are similar. This is illustrated in Figure 4.3, where the model exhibits uncertainty between two different bird species that belong to the same family in scientific classification.

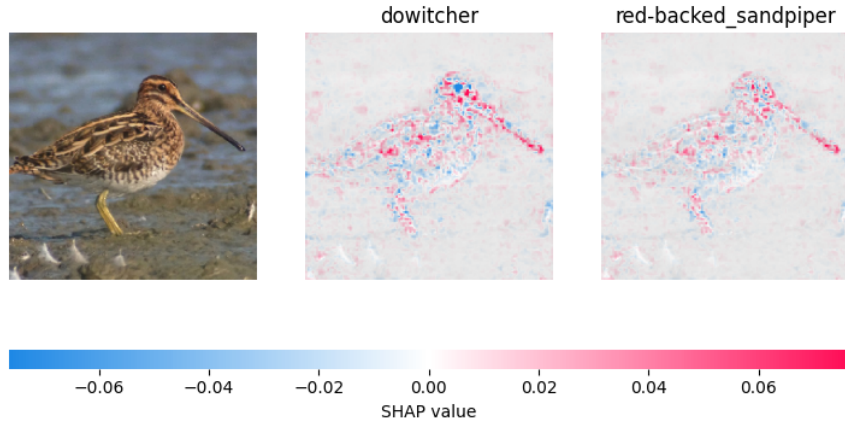


Figure 4.3: Deep SHAP applied to the VGG16 vision model [59] for classifying the input image. The trained network showed uncertainty between two classes, “Dowitcher” and “Red-backed Sandpiper”, both belonging to the same family (*Scolopacidae*) in scientific classification. The SHAP values highlight the specific features that contributed to each prediction. Reproduced from [37].

The SHAP values allow precise analysis of the model’s behavior, especially in cases of class ambiguity. The identification of the features (pixels) driving each potential classification allows understanding of why the model was confused and how it differentiates between classes.

Relevance to Interpretability. SHAP ensures that feature attributions are *consistent*, *fair*, and aligned with the model’s predictions. This is achieved by satisfying all the mentioned properties:

- **Local Accuracy:** The sum of feature attributions matches the model’s prediction.
- **Missingness:** Features that are absent get zero attribution.
- **Consistency:** If a model changes so that a feature’s contribution increases, its attribution does not decrease.

Satisfying the properties distinguishes SHAP from methods like LIME, which do not offer formal guarantees of fairness and consistency. SHAP is applicable for both **local interpretability** (explaining a single prediction) and **global interpretability** (summarizing overall feature importance across predictions).

Strengths:

- **Theoretical Guarantees:** SHAP is build upon Shapley values, which ensures fair and consistent attributions.
- **Versatility:** SHAP supports both local and global explanations.
- **Model Flexibility:** SHAP supports model-agnostic methods (Kernel SHAP) and model-specific optimizations (Tree SHAP, Deep SHAP).

Limitations:

- **Computational Complexity:** Exact Shapley value computation has exponential complexity 2^M , where M is the number of features. This limits the scalability of SHAP.
- **Approximation Errors:** Approximations like Kernel SHAP trade accuracy for efficiency.

Comparison with LIME. While LIME focuses on approximating local explanations with surrogate models, it can lead to inconsistent attributions due to reliance on perturbations and lack of theoretical guarantees. SHAP overcomes these issues by ensuring explanations are theoretically grounded, fair, and consistent.

SHAP unites strong theoretical foundations with practical applicability and has become a powerful, widely used method for transparent, reliable, and fair explanations of ML models.

4.3 Model-Specific Methods

Model-specific methods are selected techniques tailored for specific types of ML models and are not generally applicable across different architectures. These methods focus on the internal mechanisms and structures unique to a particular model type, such as an artificial neural network (ANN). Neural network-based models represent the main areas of interest in model-specific interpretability research of AI, and the presented methods are aligned with this fact. While model-agnostic methods (see [Section 4.2](#)) might be considered more flexible (in their ability to provide information about *any* ML model), they often offer less direct insights into the intricate decision-making processes of DNNs.

4.3.1 Gradient-Based Methods

When a neural network generates a prediction, it is often helpful to identify which input features (e.g., image pixels) had the most significant influence on the prediction. For instance, if a CNN trained for digit recognition classifies an image as a “7”, it can be insightful to determine which specific pixels contributed most to this decision. An intuitive approach for interpreting individual predictions might be using *gradients*. Specifically, computing the gradient $\frac{\partial F(x)}{\partial x_i}$ of the model’s output $F(x)$ with respect to each input feature x_i provides information on how slight modifications in x_i impact the model’s prediction. However, this straightforward method often results in poor explanations due to the *saturation* problem: neural networks utilizing activation functions like ReLU or sigmoid can experience gradient saturation, meaning important features may have near-zero gradients despite their significant influence on the output. The limitation led to more sophisticated gradient-based techniques that build upon the intuitive idea of gradients.

Saliency Maps. Saliency maps, also commonly referred to as pixel attribution methods, sensitivity maps (and others), were popularized by Simonyan et al. [58] as a simple gradient-based technique to visualize which input features most strongly influence a neural network’s output. They are a specific subset of the *general feature attributions*, a set of methods that attribute each feature (pixels, words, or tabular data) according to how much it altered the model’s prediction. The raw saliency map method computes the gradient of the target output (e.g., class score) with respect to each input pixel. These gradient values are then aggregated into a heatmap, where higher intensities indicate higher importance for the model’s decision. This “Vanilla Saliency” approach is visualized in [Figure 4.4](#).

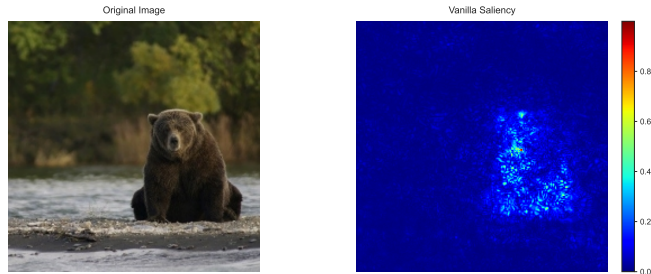


Figure 4.4: **Vanilla Saliency Map:** Visualization of pixel importance using the gradient of the class score with respect to input pixels (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].

Smooth Gradients (SmoothGrad) extend this approach by reducing noise. Instead of using a single gradient computation, they average gradients over multiple perturbed versions of the input. This enhancement results in more robust and less noisy visualizations compared to the vanilla version, as demonstrated in Figure 4.5.



Figure 4.5: **Smooth Saliency Map:** An enhanced version of the vanilla saliency map reduces noise by averaging gradients computed over multiple slightly perturbed versions of the input (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].

Pros: The advantages lie in the method’s simplicity and in the intuitive way of providing interpretable visualization of feature importance.

Cons: The saliency map approach focuses only on the first-order gradients, thus ignoring higher-order (potentially useful) interactions among features. They are also sensitive to noise; small perturbations in the input can lead to large changes in the saliency map.

Integrated Gradients. Integrated Gradients [64] address some of the limitations of raw saliency maps, such as noisy or zero gradients in saturated regions. The approach defines a *baseline* input (e.g., a black image for vision tasks) and gradually interpolates from the baseline to the actual input. At each step of this interpolation, the method computes the gradients and then *integrates* these values. Figure 4.6 illustrates the resulting attribution map, showing cumulative feature importance across the input image.



Figure 4.6: **Integrated Gradients:** Attribution map (right) shows the cumulative feature importance across the input image. The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].

Pros: This approach reduces issues with saturation and zero gradients, which leads to smoother attributions. The method also provides its theoretical basis, as integrated gradients satisfy desirable axioms (e.g., sensitivity).

Cons: Integrated Gradients are more computationally expensive than a single-step saliency map because of the need to evaluate multiple gradients (based on the number of interpolation steps). Another problem is that the resulting explanation can be influenced by the choice of baseline.

Grad-CAM (Gradient-weighted Class Activation Mapping) Grad-CAM [57] is specifically designed for CNNs, where it exploits the spatial information in the final convolutional layers. Grad-CAM calculates the gradient of the target class with respect to feature maps, averages these gradients to produce importance weights, and combines them with feature maps to generate a class activation map. Figure 4.7 depicts a Grad-CAM heatmap, illustrating the spatial regions that contributed most to the classification.,



Figure 4.7: **Grad-CAM:** Class activation map generated using gradient-weighted importance of feature maps in a convolutional layer (right). The original image (left) is from the ImageNet dataset [14], and the saliency map was computed using the VGG16 network [59].

Pros: Grad-CAM produces intuitive, spatially coherent heatmaps that closely match human perception in image tasks. Compared to raw saliency maps, the Grad-CAM approach is often more robust to noise.

Cons: The problem is that the results of this method depend on the choice of the convolutional layer. It does not explicitly capture interactions among multiple layers.

Each of the three gradient-based techniques discussed offers distinct capabilities tailored to specific interpretability needs, while collectively contributing to a comprehensive understanding of feature importance and model behavior in the context of visual data. There is not a single “best” approach to this, as the choice of method often depends on the specific model architecture, the nature of the task, and the desired level of granularity or abstraction in the explanations. They often work well in combination with other methods, as interpretability is ultimately subjective and resides in the mind of the beholder, shaped by the user’s goals, background, and expectations [62].

Chapter 5

Mechanistic Interpretability

Apart from the previously discussed *intrinsic* interpretability, the design of inherently transparent models (Chapter 3) and *post-hoc* interpretability, explaining black-box models after training (Chapter 4), there is a third perspective on the black-box problem: **mechanistic interpretability**. This approach moves from observing behavior (post-hoc methods) to examining the inner structure of the model (mechanistic methods).

5.1 Reverse-engineering Neural Networks

Mechanistic interpretability is typically referred to as the study of reverse-engineering trained neural networks to uncover the algorithms they implement [46]. This means examining the individual components of the network (neurons, weights) and their interactions (features, circuits) rather than just focusing on inputs and outputs. In this way, researchers can get closer to uncovering *why* models work, not just *that* they work.

As discussed in Section 2.4, the interpretability must overcome the **curse of dimensionality** – exponentially large high-dimensional input spaces in neural networks. To efficiently understand model behavior across these spaces, researchers simplify the problem by studying smaller neural networks [45, 71] or by analyzing specific input instances (see Section 4.3). Mechanistic interpretability addresses this similarly to traditional software reverse engineering (see Table 5.1), where programmers interpret complex software through finite, structured representations (program code). Likewise, mechanistic interpretability examines neural networks via finite parameter sets (the model’s “source code”), avoiding explicit enumeration of all inputs.

Table 5.1: Analogies between regular computer programs and neural networks from the perspective of mechanistic interpretability. Adapted from [46].

Regular Computer Programs	Neural Networks
Reverse Engineering	Mechanistic Interpretability
Program Binary	Network Parameters
VM / Processor / Interpreter	Network Architecture
Program State / Memory	Layer Representation / Activations
Variable / Memory Location	Neuron / Feature Direction

Just as understanding compiled binaries in reverse engineering reveals the behavior of traditional programs, examining a neural network’s learned *weights* can demonstrate how specific computations arise between layers. In this view, weights are analogous to compiled assembly code: they specify the low-level transformations that shape higher-level behavior [68].

Although modern neural networks often have billions of parameters [10, 25], understanding them remains potentially achievable through systematic analysis. Interpretability results are typically expected to yield simple, easy-to-follow explanations [41]. However, mechanistic interpretability, similar to neuroscience analysis of biological brains, can be as challenging as reverse engineering complex software [46]. It involves decomposing high-dimensional neuron activations into independently interpretable components. Such decomposition enables a separate understanding and interpretation of network behavior. Continuing the analogy, computer programs use memory layouts where data is meaningfully aligned. Similarly, neural networks often align meaningful features with individual neurons (a phenomenon called a *privileged basis* [18]), which makes interpretation more approachable. Challenges remain, however, such as the existence of **polysemantic neurons** that encode multiple unrelated features [47] (discussed in Section 5.2).

5.2 Features as model building blocks

At the core of mechanistic interpretability lies the idea that DNNs can be understood in terms of their internal *features*, units of representation that often correspond to individual neurons or directions in activation space [47]. These features serve as the model’s internal vocabulary, capturing visual or conceptual patterns that compose its understanding of the world. A practical, human-centric view defines a feature as something clearly interpretable, like an edge detector or a cat face detector in a convolutional network [48]. However, many features defy simple interpretation. Thus, more generally, features can be thought of as directions in activation space that consistently encode information and serve as fundamental building blocks for downstream computations. In this sense, features are analogous to the “atoms” of neural network representations: discrete, potentially composable elements that carry semantically meaningful units of computation.

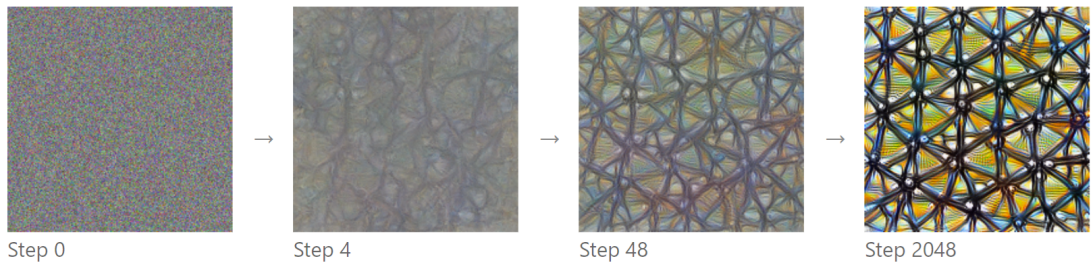


Figure 5.1: Feature visualization by optimization: an input image (initialized with random noise) is iteratively updated to maximally activate a targeted neuron (InceptionV1 [66], layer `mixed4a`, unit 11). From Olah et al. [48], CC BY 4.0.

Feature visualization is a foundational tool for studying features mechanistically. Exploiting the differentiability of neural networks allows for optimizing an input to maximize the activation of a specific neuron or channel. The result is an image that often reveals the kinds of patterns the neuron seems to be “looking for” (see Figure 5.1). While these

synthetic images can sometimes be dominated by noise or adversarial patterns, careful regularization and diversity-promoting techniques have been proposed, which have made such visualizations increasingly more realistic and interpretable [48].

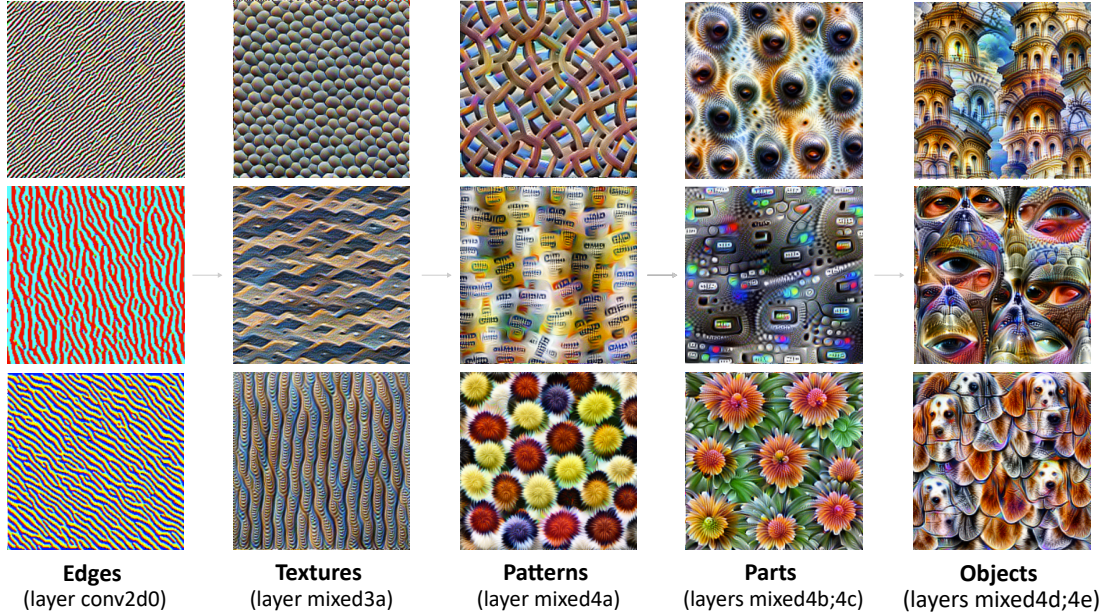


Figure 5.2: Progressive feature development across layers of InceptionV1 [66] trained on ImageNet [14]. Early layers detect edges and textures, while deeper layers capture object parts and high-level concepts. Adapted from [48], CC BY 4.0.

In vision models (specifically CNNs, e.g., InceptionV1 [66]), early layers have been found to respond to low-level visual features such as edges and textures, whereas deeper layers encode increasingly complex patterns, including object parts and entire objects (Figure 5.2). This hierarchical progression reflects a compositional abstraction pipeline, where each layer builds upon previous ones. Importantly, not all features are monosemantic. As already mentioned, many neurons tend to encode multiple unrelated features, which complicates interpretation and motivates techniques like sparse feature discovery and alternative basis analysis to disentangle these overlapping representations.

Recent research suggests that neuron–feature alignment, what Elhage et al. call a *privileged basis* [18], is not accidental but something that emerges during training. This alignment allows individual features to be interpretable and modular, which is central to the potential of mechanistic interpretability.

It is worth noting that while neurons often act as the primary unit of analysis, features need not be neuron-aligned. Directions in activation space (linear combinations of neurons) can be just as meaningful, and sometimes more so [49]. Mechanistic interpretability aims to identify and study these semantically coherent directions, whether they are embedded in individual neurons or distributed across them. Understanding features as model building blocks is a prerequisite for studying more complex structures like circuits (see Section 5.4), which describe how features *interact* to implement specific algorithms. In this framing, features are the vocabulary, and circuits are the syntax of neural computation.

5.3 Monosemantic vs. Polysemantic Neurons

Olah et al. [47] explicitly attribute polysemanticity to a phenomenon called *superposition*, in which neural networks deliberately reuse neurons for multiple unrelated features to compress representations efficiently. This reveals that polysemanticity is an emergent property of representational efficiency rather than arbitrary confusion, suggesting targeted interpretability techniques such as sparse feature decomposition or disentanglement to resolve overlapping activations effectively.

A central challenge in mechanistic interpretability is that many neurons do *not* represent a single, coherent concept. Instead, they often activate for multiple, seemingly unrelated patterns; they are *polysemantic* [47]. In contrast, *monosemantic neurons* respond selectively to a single interpretable feature, such as “curve edges” or “cat faces,” making their roles in computation easier to follow.

Polysemanticity is not a bug but a consequence of representational efficiency. Neural networks often operate in a capacity-limited setting, where there are fewer neurons than the number of features the network needs to represent. To compensate, the model *reuses* individual neurons to encode multiple directions in activation space. This results in **superposition** [17], where a neuron’s activation is the sum of several feature directions that are linearly combined. Rather than each neuron aligning with a single semantic axis, many features are “packed” into overlapping directions.

While superposition increases representational capacity, it complicates interpretation: features can no longer be mapped cleanly to individual neurons. This limits the effectiveness of neuron-level tools like *activation maximization* or *saliency maps* (see [Subsection 4.3.1](#)). As a response, researchers have developed techniques to identify a more **sparse and disentangled basis** for the activation space. These include non-negative matrix factorization (NMF) or sparse autoencoders [9]. The goal is to recover an alternative basis in which features are more orthogonal and semantically distinct, approaching monosemanticity not in the original neuron basis but in a learned interpretable one.

Understanding the distinction between monosemantic and polysemantic neurons and feature superposition geometry has significant implications. It not only reframes neuron analysis but also guides architectural choices (e.g., expanding layer width or introducing sparsity priors) that promote disentangled representations. In doing so, researchers move closer to the goal of reverse-engineering networks into understandable building blocks.

5.4 Circuits Analysis

While features can be viewed as the fundamental units of a neural network’s internal vocabulary, *circuits* constitute its grammar – the rules and patterns by which features are composed to implement more sophisticated computations. In mechanistic interpretability, circuits are defined as structured subgraphs within the network, comprising neurons (or more generally, feature directions) and the weighted connections between them, which together instantiate an identifiable algorithm or sub-function [47].

Rather than treating neurons as isolated detectors, the circuits perspective asks: “*how do these units combine to compute something greater than the sum of their parts?*” This shift in perspective mirrors a broader scientific pattern. Just as biology “zoomed in” to uncover the cell as the building block of life, interpretability research is beginning to zoom in on the circuit as a building block of learned neural computation [47]. This level of

granularity transforms interpretability from descriptive labeling into a form of algorithmic reverse engineering.

In practice, circuits are often small: they may consist of just a handful of features across two or three layers. Yet, even at this scale, they often reveal precise, modular behaviors. For example, a neuron detecting curved shapes might be constructed from a lower-level set of edge detectors aligned in a tangent configuration, a circuit that, when examined via its weights, resembles a classical computer vision algorithm for curve detection. But circuits can also represent more high-level concepts, e.g., car detectors as shown in [Figure 5.3](#).

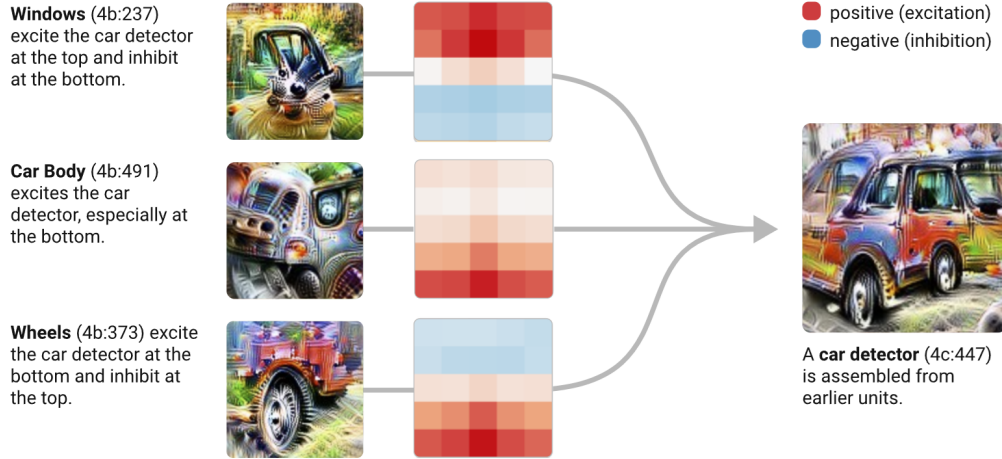


Figure 5.3: Circuit-based composition of a car detector in InceptionV1 [66] neural network. This visualization illustrates how a high-level Car Detector (4c:447) emerges from lower-level feature detectors through learned connections. The Windows (4b:237), Car Body (4b:491), and Wheels (4b:373) units contribute excitatory (red) and inhibitory (blue) signals, forming a neural circuit that integrates these components into a coherent car representation. From Olah et al. [47], [CC BY 4.0](#).

5.4.1 Circuit Motifs

Recurring architectural patterns called *circuit motifs* offer insight into the design principles underlying learned representations [47]. These include:

- **Equivariance:** Redundant subcircuits that detect pattern variations (e.g., curve detectors rotated across orientations), suggesting distributed symmetry.
- **Union over cases:** Multiple subdetectors for distinct but semantically related inputs (e.g., left- and right-facing dog heads) combined into an invariant representation.
- **Superposition motifs:** Circuits where multiple concepts share neurons, contributing to polysemanticity and complicating clean interpretability.

These motifs resemble common structures in neuroscience, e.g., retinotopic mappings and lateral inhibition, and hint at the existence of a deeper circuit-level ontology for neural computation [28].

5.4.2 From Features to Circuits

Features provide the atoms; circuits provide the chemistry. A circuit might connect a horizontal edge detector, a vertical edge detector, and a corner detector into a higher-level shape recognizer. These compositional hierarchies reveal how abstract representations (such as object categories) are synthesized from reusable mid-level parts. For example, in InceptionV1 [66], curve detectors are implemented via weighted sums of oriented edge detectors. The learned weights encode specific tangent alignments, forming smooth arcs. When these weights are visualized, they exhibit structure: excitatory lobes aligned with the curve and inhibitory margins that suppress off-pattern activation. Such structured patterns imply not just correlation but computation.

5.4.3 Methodologies

Circuit discovery can proceed via:

- **Manual inspection**, guided by feature visualization and weight alignment.
- **Automated tools** such as Network Dissection [7], which match neurons to human-understandable concepts.
- **Weight tracing** and “expanded weights” analysis [68], revealing how signals propagate through multiple layers.

While manual approaches are more accurate, they are time-intensive. Automation, though still in early stages, holds promise for scaling circuit analysis to modern architectures.

5.4.4 Implications and Limitations

Circuit-level analysis provides a mechanistic lens through which we can verify hypotheses about learned behavior. In the case of InceptionV1’s curve detectors, Olah et al. [47] applied seven converging lines of evidence (feature visualizations, synthetic inputs, tuning curves, circuit inspection, and more) to validate that specific neurons indeed implement curve detection algorithms. Yet limitations remain. Polysemanticity, where circuits reuse the same neurons for multiple roles, undermines compositional clarity. This is particularly prevalent in later layers or under-parameterized regimes. Moreover, circuits identified in relatively shallow vision models may not generalize cleanly to large language models, where learned representations are highly entangled and abstract.

Despite these challenges, the core insight remains: by zooming in on the level of weights and connections, we begin to recover the hidden algorithms that give rise to model behavior. Circuits offer a path not just toward interpretation, but toward mechanistic understanding, transforming the black box into something closer to software with source code.

5.5 Comparison to Intrinsic and Post-hoc Methods

Mechanistic interpretability differs fundamentally from *intrinsic* and *post-hoc* approaches. Instead of simplifying models or approximating their behavior externally, it analyzes trained networks directly to uncover the internal computations responsible for their outputs.

Intrinsic Interpretability. Intrinsic methods (see [Chapter 3](#)) prioritize transparency *by design*. Models like linear regression or decision trees are interpretable because their decision-making logic is explicit and human-readable. While this often limits expressiveness on complex tasks, mechanistic interpretability addresses more complex models, seeking structure within their learned representations (e.g., features, circuits) without requiring simplicity at design time.

Post-hoc Interpretability. Post-hoc methods (see [Chapter 4](#)) such as SHAP [37] or LIME [52] explain predictions by analyzing input-output behavior, often treating models as black boxes. They are scalable and practical but provide indirect and sometimes misleading explanations [56]. Mechanistic interpretability, by contrast, seeks to trace how specific parameters, activations, and structures give rise to model behavior.

Goals and Methodology. Intrinsic and post-hoc methods focus on usability and trust, especially in practical or regulated contexts. Mechanistic interpretability aims for a deeper, algorithmic understanding akin to reverse-engineering software. While intrinsic methods are interpretable by construction and post-hoc methods typically explain from the outside, mechanistic methods inspect the model itself, via tools such as feature visualization, circuit tracing, or activation patching.

Strengths and Limitations. Mechanistic interpretability offers detailed, faithful insights but is time-consuming, lacks tooling, and scales poorly to large models. Post-hoc methods are widely applicable but can oversimplify. Intrinsic methods are transparent but limited in expressivity. Each has different trade-offs suited to different contexts.

Complementarity. These paradigms are complementary: intrinsic methods offer built-in transparency, post-hoc methods enable broad accessibility, and mechanistic interpretability provides deep causal understanding. Together, they span a spectrum: from design-time clarity to black-box explanation to internal reverse engineering, offering layered insights into model behavior.

As machine learning systems grow in complexity and societal relevance, mechanistic interpretability adds a crucial *epistemic* dimension: not only explaining *what* a model does, but understanding *how* and *why* it works.

Chapter 6

Proposed extension: XAI Ensembles

The ability to *combine* multiple explanations into a more stable and informative ensemble aligns with a broader goal in interpretable machine learning: enhancing the trustworthiness and usability of explanations without requiring extensive modifications to existing models. This chapter systematically explores several ensemble strategies in the field of XAI, critically evaluates their advantages and limitations, and introduces an adaptive extension designed to overcome key shortcomings observed in current ensemble approaches.

6.1 Why ensemble explanations?

Combining explanation methods is not just a convenience, it is a principled response to the known weaknesses of individual techniques, which often suffer from instability, limited scope, and method-specific artifacts:

- **Instability of single methods:** Gradient-based methods (e.g., Vanilla Gradients) exhibit high sensitivity to input perturbations, and small noise can cause large attribution shifts. Perturbation-based methods (e.g., Occlusion [72]) depend on parameters such as patch size or mask resolution, which significantly alter the resulting saliency maps and explanation granularity.
- **Complementary strengths:** GradientSHAP [37] produces explanations that are faithful but might result in high-frequency noise. Occlusion [72] tends to blur fine details and local relevance. Integrated Gradients [64] can better capture long-range interactions, while Grad-CAM [57] emphasizes coarse class-discriminative regions robustly but ignores pixel-level localization.
- **Observation from Rieger and Hansen [53]:** Empirically, averaging attribution maps from multiple explanation methods already improved *robustness* to adversarial perturbations. This suggests that ensemble-based combination, even using simple averaging, can reduce method-specific artifacts and noise. A more principled ensemble may further improve *faithfulness*, *localization*, and *stability*.

6.2 Ensemble Techniques

In recent years, multiple approaches have been proposed for combining explanations.

NormEnsembleXAI [27] offers a structured, simple pipeline for combining explanation maps (depicted in Figure 6.1).

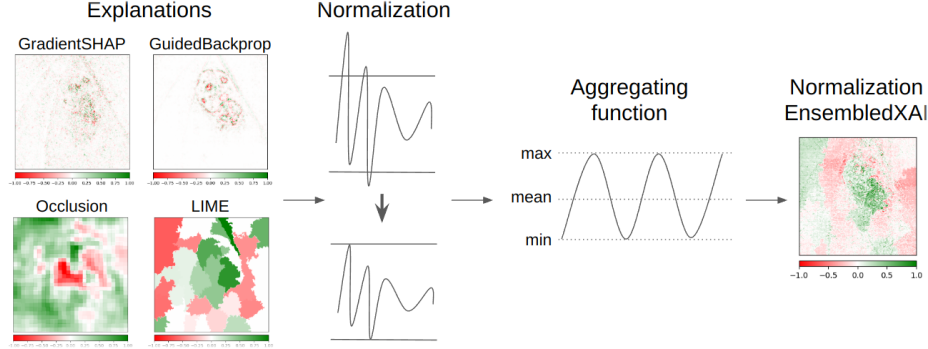


Figure 6.1: Visualization of the NormEnsembleXAI method. From [27].

The method operates in two main stages:

- **Normalization:** Each XAI method’s attribution maps are standardized across the dataset by computing *global* statistics (mean, standard deviation) and applying z-score normalization. Formally, given an explanation map ϕ_e for method e , the normalized map $\hat{\phi}_e$ is computed as:

$$\hat{\phi}_e = \frac{\phi_e - \mu_e}{\sigma_e} \quad (6.1)$$

where μ_e and σ_e are the global mean and standard deviation over the training set. This ensures that explanations produced by methods with different dynamic ranges or attribution scales become comparable.

- **Aggregation:** After normalization, the ensemble attribution is formed by aggregating the normalized maps. Proposed aggregation strategies include simple mean, elementwise minimum, and maximum aggregation:

$$\phi_{\text{ens}} = \text{agg}_e(\hat{\phi}_e) \quad (6.2)$$

where agg denotes either mean, min, or max.

NormEnsembleXAI is simple, architecture-agnostic, and significantly reduces the impact of outlier methods. However, a critical limitation is that it treats all contributing methods *equally*, without considering their actual explanation quality. As a result, unreliable or noisy methods can still negatively affect the final ensemble attribution, particularly when no filtering or quality control is applied.

AutoWeighted [8] ensemble offers an extended approach compared to NormEnsembleXAI. It introduces *quality-aware weighting* of the individual explanations. Instead of aggregating all methods equally, each explanation map is evaluated across multiple quantitative metrics

(e.g., faithfulness, robustness, or sparsity), and a weighted average is computed based on these assessments. The weights in AutoWeighted are selected *manually*, typically reflecting the relative importance assigned to different evaluation criteria. Each method e receives an *ensemble score* ES_e :

$$ES_e = \sum_k w_k m_{k,e} \quad (6.3)$$

where $m_{k,e}$ is the value of quantitative metric k and w_k are user-defined importance weights, which are *global* (image-independent).

Let $\hat{\phi}_e$ denote the normalized attribution map from XAI method e , as defined in NormEnsembleXAI (Equation 6.1). Let α_e denote the weight assigned to method e based on its ensemble score ES_e , normalized across all methods (indexed by e') to ensure weights sum to 1. The final ensemble attribution ϕ_{ens} is weighted sum over all normalized attributions:

$$\phi_{\text{ens}} = \sum_e \alpha_e \hat{\phi}_e, \quad \text{where } \alpha_e = \frac{ES_e}{\sum_{e'} ES_{e'}} \quad (6.4)$$

While this approach addresses the major limitation of blind aggregation, it still relies on *global* quality evaluations, meaning that a method’s weight is constant across all inputs, regardless of per-sample variability. Consequently, methods that are generally reliable but fail on specific inputs may still influence the final explanation in undesirable ways.

Supervised Ensemble Approaches treat explanation combination as a supervised learning problem, incorporating regression models trained to predict high-quality explanations based on *ground-truth* labels (e.g., segmentation masks). Approaches include **SupervisedXAI** [74] utilizing Kernel-Ridge Regressor, and **CNN-based Ensemble** [26] utilizing Convolutional Neural Network. These methods can learn complex aggregation strategies but come with notable limitations:

- **Requirement of ground truth:** They require pixel-level explanation labels, which are often unavailable or impractical to obtain in real-world settings. Since no universally agreed-upon ground-truth explanations exist for most datasets, supervised ensembles risk learning artifacts that do not generalize well or truly reflect model behavior.
- **Training complexity:** Training ensemble models introduces additional computational overhead and potential overfitting risks.

6.3 Limitations that motivate a new method

While ensemble methods offer clear advantages, existing approaches suffer from critical limitations that reduce their effectiveness and generalizability.

- L1. **Input-Agnostic Weights.** Existing methods compute global aggregation weights that remain *fixed* across all samples. They cannot dynamically downweight an explanation that fails on a particular input, leading to degraded ensemble quality when one method underperforms locally.
- L2. **Ignorance of Metric Trade-offs.** Simple aggregation (min, max, mean) implicitly assumes that all quantitative metrics (e.g., faithfulness, robustness, or localization) can

be jointly optimized, which is often not the case. Prioritizing one metric may worsen others (e.g., maximizing localization can reduce robustness), and fixed aggregation rules cannot adapt to these trade-offs per input.

- L3. Need for Ground-Truth Labels in Supervised Approaches.** Supervised ensembling strategies require ground-truth explanation masks for training, which are unavailable for most tasks and datasets. This restricts their practical applicability and risks overfitting to specific label distributions instead of capturing genuine model behavior.

Together, these limitations reveal the need for an **input-adaptive, metric-aware, label-free** ensemble method that can dynamically select and combine explanations based on their quality on each individual input, without supervision.

6.4 Proposed Extension: Adaptive-Metric Softmax (AMS)

To overcome the limitations discussed in [Section 6.3](#) (namely, input-agnostic weights, ignorance of metric trade-offs, and the need for ground-truth labels in supervised approaches), this work introduces **Adaptive-Metric Softmax (AMS)** Ensemble. AMS is a novel ensemble technique that dynamically adapts the combination of explanation methods based on their per-sample quantitative quality assessment. The method operates in five systematic steps, outlined below.

Input: For a given input sample i , let $\{\phi_e^{(i)}\}_{e=1}^E$ denote the attribution maps produced by E diverse base explanation methods (e.g., gradient-based, perturbation-based). It is assumed that each individual attribution map $\phi_e^{(i)}$ has its pixel values normalized to a common range (e.g., $[0,1]$ or $[-1,1]$) to ensure fair contribution during the final fusion stage.

Each explainer’s map $\phi_e^{(i)}$ is evaluated using a collection $\mathcal{M} = \{m_1, \dots, m_K\}$ of K distinct quantitative metrics¹. This process yields a metric vector $\mathbf{m}_e^{(i)}$ for each explainer e on sample i :

$$\mathbf{m}_e^{(i)} = [m_{1,e}^{(i)}, \dots, m_{K,e}^{(i)}] \in \mathbb{R}^K$$

Step 1: Per-Metric Z-Score Normalization. To ensure all metrics contribute on a comparable scale regardless of their original units or ranges, each metric k is standardized across the E explanation methods for the current sample i :

$$\hat{m}_{e,k}^{(i)} = \frac{m_{e,k}^{(i)} - \mu_k^{(i)}}{\sigma_k^{(i)} + \varepsilon}$$

where $\mu_k^{(i)}$ and $\sigma_k^{(i)}$ are the sample mean and standard deviation of metric k ’s scores across all E methods for sample i . The constant ε (e.g., 10^{-8}) ensures numerical stability.

This z-score normalization centers and scales the metric values, offering greater stability against outliers compared to min-max scaling, especially for small E .

¹Any differentiable or black-box metric is admissible. Common choices from libraries such as **Quantus** [23] include those assessing faithfulness, robustness, localization, sparsity, and others. The selection should align with the desired characteristics of the final ensemble explanation.

Step 2: Utility Score Computation. The normalized metric scores $\hat{m}_{e,k}^{(i)}$ are aggregated into a single scalar *utility score* $s_e^{(i)}$ for each method e . This is achieved using a user-defined priority vector $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_K] \in \Delta^K$ (the K -dimensional probability simplex, where $\lambda_k \geq 0$ and $\sum_k \lambda_k = 1$):

$$s_e^{(i)} = \sum_{k=1}^K \lambda_k \hat{m}_{e,k}^{(i)}$$

The priority vector $\boldsymbol{\lambda}$ encodes the relative importance of each metric, allowing AMS to be tailored to specific interpretability goals or application contexts (e.g., prioritizing faithfulness for model debugging versus localization for object identification). The priority vector $\boldsymbol{\lambda}$ can be set based on domain expertise, tuned on a validation set, or left uniform ($\lambda_k = 1/K$) if no specific preference exists.

Step 3: Confidence Correction. The reliability of metric evaluations can vary, especially for stochastic metrics (e.g., those based on random perturbations or sampling). To account for this, an uncertainty estimate $\sigma_{e,k}^{(i)}$ is obtained for each metric $m_{e,k}^{(i)}$ that is stochastic, typically by computing its standard deviation using B bootstrap resamples:

$$\sigma_{e,k}^{(i)} = \text{BootstrapSD}(m_{e,k}^{(i)}, B)$$

For deterministic metrics, $\sigma_{e,k}^{(i)}$ is zero. These uncertainties are then aggregated into an overall uncertainty score $\sigma_e^{(i)}$ for each method e , weighted by the squared metric priorities:

$$\sigma_e^{(i)} = \sqrt{\sum_{k=1}^K \lambda_k^2 (\sigma_{e,k}^{(i)})^2}$$

This overall uncertainty is used to compute a *confidence factor* $c_e^{(i)}$:

$$c_e^{(i)} = \exp(-\sigma_e^{(i)})$$

The factor $c_e^{(i)}$, ranging $(0, 1]$, will downweight methods whose utility scores are based on unstable or noisy metric evaluations.

Step 4: Softmax Weighting. The final weight $w_e^{(i)}$ for each explanation method e is derived using a temperature-scaled softmax function, incorporating both the utility score $s_e^{(i)}$ and the confidence factor $c_e^{(i)}$:

$$w_e^{(i)} = \frac{\exp(\tau s_e^{(i)}) c_e^{(i)}}{\sum_{e'=1}^E \exp(\tau s_{e'}^{(i)}) c_{e'}^{(i)}}$$

The temperature parameter $\tau > 0$ (default $\tau = 1$) controls the sharpness of the distribution: higher τ values lead to a more “winner-takes-all” assignment, while lower values produce softer weights. This ensures that methods with higher utility and higher confidence receive exponentially larger weights.

Step 5: Fusion. The final ensemble attribution map $\phi_{\text{AMS}}^{(i)}$ is computed as a combination of the base attribution maps $\phi_e^{(i)}$, using the dynamically computed weights $w_e^{(i)}$:

$$\phi_{\text{AMS}}^{(i)} = \sum_{e=1}^E w_e^{(i)} \phi_e^{(i)}$$

Since $\sum_e w_e^{(i)} = 1$ and $w_e^{(i)} \geq 0$, the final attribution $\phi_{\text{AMS}}^{(i)}$ is guaranteed to lie within the convex hull of the base explanation maps. This is an important property, as it prevents the ensemble from introducing artifacts outside the range from its constituent explanations.

The procedure for the AMS Ensemble method is summarized in Algorithm 1.

Algorithm 1 Adaptive-Metric Softmax (AMS) Ensemble

```

1: Inputs: attributions  $\{\phi_e\}_{e=1}^E$ , metrics  $\mathcal{M} = \{m_1, \dots, m_K\}$ , priority vector  $\lambda \in \Delta^K$ ,
   softmax temperature  $\tau$ , bootstrap size  $B$ 
2: Output: ensemble attribution  $\phi_{\text{AMS}}$ 

3: for  $e = 1$  to  $E$  do
4:   for  $k = 1$  to  $K$  do
5:      $m_{e,k} \leftarrow m_k(x, \phi_e)$ 
6:     if  $m_k$  is stochastic then
7:        $\sigma_{e,k} \leftarrow \text{BOOTSTRAPSD}(m_{e,k}, B)$ 
8:     end if
9:   end for
10: end for
11: Compute  $\mu_k$  and  $\sigma_k$  (mean and std.) over  $e = 1, \dots, E$ 
12: for  $e = 1$  to  $E$  do
13:   for  $k = 1$  to  $K$  do
14:      $\hat{m}_{e,k} \leftarrow (m_{e,k} - \mu_k) / (\sigma_k + \varepsilon)$ 
15:   end for
16:    $s_e \leftarrow \sum_{k=1}^K \lambda_k \hat{m}_{e,k}$  ▷ utility
17:    $\sigma_e \leftarrow \sqrt{\sum_{k=1}^K \lambda_k^2 \sigma_{e,k}^2}$  ▷ uncertainty
18:    $c_e \leftarrow \exp(-\sigma_e)$  ▷ confidence
19: end for

20:  $w_e \leftarrow \frac{\exp(\tau s_e) c_e}{\sum_{j=1}^E \exp(\tau s_j) c_j}$  ▷ softmax weights
21:  $\phi_{\text{AMS}} \leftarrow \sum_{e=1}^E w_e \phi_e$  ▷ fusion

22: return  $\phi_{\text{AMS}}$ 

```

AMS therefore delivers an **input-specific, metric-prioritized, uncertainty-aware** ensemble. It is broadly applicable to *any* collection of base explanation methods and quantitative metrics, without requiring ground-truth annotations for training. The resulting ensemble explanation is a convex combination of its constituents, adapting dynamically to leverage the strengths of individual explainers on a per-sample basis.

It is acknowledged that AMS does not provide *the* perfect explanation addressing every issue with existing methods and comes with advantages and limitations that define it:

Advantages:

- **Dynamic Adaptivity:** Weights are tailored per input sample, enabling AMS to emphasize the most suitable explainer for a given instance, unlike methods with fixed or globally determined weights (e.g., NormEnsembleXAI [27], AutoWeighted [8]).
- **Customizable Priorities:** The metric priority vector (λ) allows users to explicitly define the desired trade-offs between different aspects of explanation quality (e.g., Faithfulness vs. Sparsity), aligning the ensemble with specific interpretability goals.
- **Uncertainty Management:** By incorporating a confidence factor derived from metric stability, AMS can downweight explanations whose quality assessments are themselves unreliable or noisy.
- **Generality and Flexibility:** AMS can combine any set of base XAI methods and any quantitative evaluation metrics, offering broad applicability.
- **Preservation of Attribution Range:** The convex combination fusion ensures that the ensemble does not introduce new attribution artifacts outside the range spanned by the base explanations.
- **Unsupervised Nature:** The method is label-free, which makes it practical in scenarios where ground-truth explanations are unavailable (unlike SupervisedXAI [74], or CNN-based explanation ensembling [26]).

Limitations and Considerations:

- **Dependence on Metric Quality:** The performance of AMS relies on the selection of the quantitative metrics and base explainers. If the metrics do not accurately capture the desired characteristics of a “good” explanation for the task, or if base explainers are uniformly poor, the ensemble’s utility will be limited.
- **Computational Cost:** The primary computational overhead lies in the per-sample evaluation of $E \times K$ metrics, especially if these metrics (and their bootstrap estimates for uncertainty) are intrinsically expensive to compute. This cost needs to be weighed against the potential improvements in explanation quality.
- **Parameter Sensitivity and Configuration:** Choices of base explainers, metrics, priority vector λ , and temperature τ are important configuration decisions. Optimal settings, particularly for λ , might require domain expertise or careful tuning.
- **Requirement for Multiple Explainers:** As an ensemble method, AMS requires at least two, and ideally a diverse set of base explanation methods as input. It cannot generate an ensemble from a single explainer.
- **Interpretability of the Ensemble Process:** While the AMS process is transparent, understanding why specific weights were assigned to base methods for a particular input requires analyzing the interplay of all K metrics for all E methods on that input, which can add a layer of meta-interpretation.

In summary, AMS offers an additional view of creating adaptive XAI ensembles and aims for enhanced reliability and relevance. Its practical effectiveness depends on careful configuration and the inherent quality of its constituent components and evaluation criteria.

Chapter 7

Evaluation of the Explanations

As mentioned in [Section 2.4](#), evaluating interpretability methods is inherently challenging. Explanations are generally considered *subjective* and have a context-dependent nature. Over the years, multiple approaches to the evaluation in XAI have been proposed; however, they have not reached a clear, universal, and generally agreed-upon consensus [\[16, 36\]](#). Why is evaluating interpretability and explainability methods relevant? Effective explanations lead to improved human decisions [\[51\]](#), but incorrect ones can actively mislead users, potentially leading to negative outcomes [\[30\]](#). The main goal is establishing trust, transparency, and reliability of ML systems, especially in high-stakes decision-making situations (healthcare, finance) and autonomous systems. A comprehensive evaluation approach typically involves both *qualitative* (human-centered) and *quantitative* (metric-based) assessments to holistically measure the effectiveness of explanations [\[62, 21\]](#).

The goal of evaluating interpretability in AI is twofold: to measure how accurately explanations reflect the true decision-making process of the ML model (*completeness*) and to determine how effectively the explanations facilitate human understanding (*interpretability*). This dichotomy often introduces a challenging trade-off, as highly *complete* explanations can become complex and less comprehensible, whereas simpler explanations risk omitting critical model behavior details [\[21, 54\]](#). Evaluations should cautiously weigh this trade-off to avoid misleading users.

This chapter outlines both *qualitative* and *quantitative* approaches to evaluating interpretability methods, followed by a section on challenges encountered in the evaluation process.

7.1 Qualitative Evaluation: Interpretability

Qualitative evaluation methods emphasize human judgment and practical usability. Such evaluations measure the *interpretability* of an explanation by assessing how effectively users can understand, trust, and utilize it [\[63, 21\]](#). Qualitative methods commonly include user studies, interviews, and expert assessments, enabling direct engagement with stakeholders to determine how well explanations facilitate practical decision-making.

Key considerations include:

- **Comprehensibility:** How easily do users grasp the explanation without excessive cognitive effort?

- **Actionability:** Can users practically utilize the explanation in their decision-making processes?
- **Contextual Alignment:** How well do explanations resonate with users’ existing knowledge and task-specific goals?

The notion of *comprehensibility* is central to qualitative evaluation. If an explanation is comprehensible, users can interpret and act upon it. Comprehensibility mainly depends on the cognitive abilities, background knowledge, and specific objectives of the people of interest [62]. Human-centered evaluations typically employ interactive, dialogue-based approaches, allowing users to provide direct feedback and adapt explanations dynamically. In the context of computer vision (specific focus of this thesis) and image data, qualitative explanations often take the form of visualizations (saliency maps, activation overlays, and feature-visualization plots), which highlight input regions influencing the model’s prediction.

Qualitative evaluations rely on human intuition, being inherently subjective and context-specific. Studies show humans might find persuasive yet misleading explanations convincing, causing evaluations to reflect perceived transparency rather than actual clarity or accuracy [50, 32]. Therefore, qualitative evaluations should be complemented by quantitative methods, which offer objective and reproducible metrics for interpretability assessment [16].

7.2 Quantitative Evaluation: Completeness

Quantitative evaluation serves as a complementary approach to the qualitative assessments discussed in Section 7.1. It aims to measure how accurately an explanation reflects the true decision-making process of a model, a notion referred to as *completeness*. The quantitative metrics are designed to be objective and reproducible, which facilitates direct comparisons across interpretability methods. However, it’s important to recognize that quantitative metrics are not infallible. The absence of ground truth explanations and the sensitivity of some metrics to hyperparameter choices can lead to inconsistent, or even misleading evaluations [70, 23]. Therefore, although quantitative methods might provide valuable insights, they should be applied carefully and in conjunction with qualitative assessments to ensure a holistic evaluation of XAI methods. Following Hedström et al. [23], quantitative measures can be categorized into six groups: faithfulness, robustness, localization, complexity, randomization, and axiomatic approach.

7.2.1 Faithfulness

Faithfulness quantifies how well an explanation captures the internal logic of the model. The assumption is that more important features should have a stronger impact on predictions.

Pixel Flipping [6] evaluates faithfulness by progressively removing input features (replacing with uninformative baseline values) in descending order of their attributed relevance and observing the corresponding degradation in the model’s output. For each step k , a new input $x^{(k)}$ is created by perturbing the top- k most important features in x , and the model output $f_{\text{model}}(x^{(k)})$ is recorded.

$$\text{PF} = \sum_{k=1}^K f_{\text{model}}(x^{(k)}) \quad (7.1)$$

The metric is computed as the area under the resulting degradation curve, where a *lower area* indicates *higher faithfulness*. Thus, more relevant features are removed earlier, causing the prediction to drop quickly.

Insight: Pixel Flipping directly tests whether explanations identify the causal drivers of predictions. Faithful explanations lead to a rapid drop in model confidence as top-attributed features are perturbed, which makes this metric especially informative for visual or attribution-based explanations.

7.2.2 Robustness

Robustness measures the stability of explanations under small *perturbations* to the input. A robust method should produce consistent outputs when the input is slightly altered.

Local Lipschitz Estimate [3] is a commonly used metric to quantify robustness. It captures the local sensitivity of an explanation method f by measuring how much the explanation changes in response to small input changes. Given an input x , the local Lipschitz constant $\hat{L}(x)$ is estimated as:

$$\hat{L}(x) = \max_{x' \in B_\epsilon(x)} \frac{\|f_{\text{expl}}(x) - f_{\text{expl}}(x')\|_2}{\|x - x'\|_2} \quad (7.2)$$

where $B_\epsilon(x)$ is a small neighborhood around x , and $f_{\text{expl}}(x) \in \mathbb{R}^d$ is the explanation vector assigning attribution scores to each of the d input features. The $\|\cdot\|_2$ represents the Euclidean (L2) norm. A lower value of $\hat{L}(x)$ indicates higher robustness.

Insight: Robust explanations are more trustworthy in practice, particularly in safety-critical applications where reliability under input noise matters. A low local Lipschitz estimate ensures that the explanation is stable and not overly sensitive to minor, potentially irrelevant perturbations in the input.

7.2.3 Localization

Localization measures whether an explanation correctly identifies spatially relevant input regions, typically evaluated in image tasks. A well-localized explanation highlights areas that semantically correspond to the object or concept responsible for a prediction.

Pointing Game [73] assesses whether the most salient pixel in an explanation aligns with the ground-truth location of the target object. Given an input image and a target class, an explanation method produces an attention map. The Pointing Game identifies the pixel with the maximum saliency and checks whether it lies within a ground-truth mask or bounding box. Formally, the localization accuracy (LA) can be expressed as:

$$\text{LA} = \frac{\#\text{Hits}}{\#\text{Hits} + \#\text{Misses}} \quad (7.3)$$

A *hit* occurs when the most salient point falls inside the annotated region. Otherwise, it's a *miss*. This metric is label-aware but architecture-agnostic and directly measures the spatial alignment between the explanation and annotated object locations.

Insight: The Pointing Game isolates the discriminative focus of an explanation without requiring the full object extent to be highlighted. It is especially valuable for evaluating the spatial precision of top-down attention maps in multi-object scenes.

7.2.4 Complexity

Complexity measures how concise an explanation is, favoring sparse, minimal explanations that are easier to interpret. Sparse explanations assign relevance to only a few dominant features, reducing cognitive load and aiding human understanding.

Sparsity (Gini Index) [13] quantifies the inequality of attribution values, capturing how concentrated the explanation is on a few input features. Given a non-negative attribution vector $f(x) \in \mathbb{R}^d$, where d is the number of input features, the Gini Index is computed as:

$$\text{Gini}(f(x)) = \frac{\sum_{i=1}^d (2i - d - 1) \cdot |f(x)_{(i)}|}{d \cdot \sum_{i=1}^d |f(x)_i|} \quad (7.4)$$

where $|f(x)_{(i)}|$ denotes the i -th smallest component of the sorted absolute attribution vector (i.e., sorted such that $|f(x)_{(1)}| \leq \dots \leq |f(x)_{(d)}|$), and $|f(x)_i|$ refers to the i -th component in the original (unsorted) attribution vector.

Insight: Higher Gini scores reflect explanations focusing on a few influential features, enhancing clarity. As demonstrated by Chalasani et al. [13], sparsity also correlates with robustness: adversarially trained models tend to produce sparser, more stable explanations, suggesting a potential link between interpretability and model reliability.

7.2.5 Randomization (Sensitivity)

Randomization metrics evaluate whether explanations appropriately degrade when either the model’s parameters or the input-output relationship is randomized. A good explanation method should produce significantly different explanations under randomization, thus showing that it depends on the model’s learned behavior and not just on visually salient or statistically common patterns in the input.

Random Logit Test (RLT) [60] assesses whether explanations specifically highlight features relevant to the predicted class or whether they remain similar even when an unrelated class is artificially assigned. This test compares saliency maps generated for the model’s predicted logit against maps for randomly selected logits. The similarity between these two explanations is quantified using the Structural Similarity Index Measure (SSIM):

$$\text{Random Logit SSIM}(f(x), f_{\text{rand}}(x)) = \text{SSIM}(E(f(x)), E(f_{\text{rand}}(x))) \quad (7.5)$$

where $E(f(x))$ is the explanation generated for the predicted logit, and $E(f_{\text{rand}}(x))$ denotes the explanation for a randomly chosen logit. Lower SSIM values indicate explanations strongly tied to the specific predicted class, signifying higher sensitivity and reliability.

Insight: The Random Logit Test effectively serves as a sanity check against explanations that trivially reflect general model behavior or dataset statistics rather than genuine class-specific reasoning. Methods producing significantly different explanations when logits are randomized show higher sensitivity to the actual predictive structure of the trained model.

Model Parameter Randomisation Test (MPRT) [1] probes whether explanations depend on learned weights by randomising parameters layer-wise (either from output to input or independently) and measuring how much the attributions change. Let $E(f(x))$ be the explanation under the original model and, for each layer l , let $E(f_l(x))$ be the explanation after randomising that layer’s parameters.

The similarity is computed via Spearman rank correlation (default in Quantus) or SSIM:

$$\text{MPRT}_{\text{sim}}(f(x)) = \frac{1}{L} \sum_{l=1}^L \text{sim}(E(f(x)), E(f_l(x))) \quad (7.6)$$

Optionally, one may report only the final (fully randomized) layer:

$$\text{MPRT}_{\text{sim}}^{\text{last}}(f(x)) = \text{sim}(E(f(x)), E(f_L(x))) \quad (7.7)$$

Lower values indicate higher sensitivity to parameter perturbations, thus greater fidelity to the model’s true reasoning.

Insight: Explanations that remain similar after cascading (from output to input) or independent layer randomisation (high MPRT_{sim}) fail to reflect parameter-specific behaviour and instead capture architectural or dataset priors. Passing the MPRT (i.e. low average or last-layer similarity) demonstrates that explanations truly depend on the learned weights rather than on input statistics.

7.2.6 Axiomatic Metrics

Axiomatic metrics test whether explanations satisfy the theoretical properties expected of attribution methods. These include:

- **Completeness** [64]: Requires that total attribution matches the model output difference from a baseline.
- **Input Invariance** [31]: Expects explanations to remain unchanged under transformations that do not affect the model’s output.

Insight: These metrics formalize desirable behaviors that explainers should satisfy, though they are often stricter than what is needed in practice.

A summary of commonly used evaluation categories and representative metrics following the Quantus library [23] is provided in [Table 7.1](#).

Category	Purpose	Selected Metrics
Faithfulness	Are key features causally important?	Pixel Flipping [6]
Robustness	Are explanations stable under perturbation?	Local Lipschitz Est. [3]
Localization	Do explanations align with task-relevant regions?	Pointing Game [73]
Complexity	Are explanations concise and minimal?	Sparsity (Gini Index) [13]
Randomization	Do explanations fail gracefully with noise?	RLT [60], MPRT [1]

Each metric group provides a different lens on explanation completeness, and their joint use enables more reliable and multi-dimensional evaluation. In practice, the selection of metrics should be guided by the specific use case, data modality, and risks of failure modes relevant to the domain [63, 55].

7.3 Quantus: XAI Toolkit

To support rigorous and reproducible evaluation of XAI methods, Hedström et al. [23] introduced **Quantus**, a Python library focused on quantitative evaluation of XAI-based explanations. Unlike libraries that generate explanations (e.g., Captum), **Quantus** provides over 30 metrics across six categories (described in [Section 7.2](#)): faithfulness, robustness, localization, complexity, randomization, and axiomatic soundness.

Quantus aims at standardization and usability: it supports both **PyTorch** and **TensorFlow**, includes sanity checks and visualization tools, and allows single-line execution of evaluation routines. It also encourages careful metric selection by highlighting parameter sensitivity and potential pitfalls. As such, **Quantus** helps fill the methodological gap in XAI evaluation and provides a consistent and extensible framework for assessing explanation quality across tasks and methods.

7.4 Challenges in Explanation Evaluation

One fundamental problem for evaluating interpretability methods’ results is the general lack of available ground truth for what constitutes a “correct” explanation [21]. Evaluating explanations confronts several critical challenges:

- **Trade-offs between Interpretability and Completeness:** Highly interpretable explanations may sacrifice accuracy, and comprehensive explanations may lose interpretability.
- **Subjectivity and Context Dependency:** Interpretability heavily relies on user background, which complicates standardized assessments.
- **Computational Complexity:** Many quantitative metrics require extensive computation, restricting scalability.
- **Evaluation Multiplicity:** Multiple explanations or views may enhance understanding or overwhelm users, requiring careful evaluation of cognitive impacts.

Addressing these challenges requires selecting appropriate metrics and acknowledging the human-centered nature of interpretability. Future evaluations should incorporate interactive, adaptive methods that evolve through user feedback. Combining qualitative insights with quantitative metrics in a unified framework enables a more reliable, transparent, and context-sensitive evaluation of explanations.

Chapter 8

Experimental Analysis

This chapter demonstrates and systematically evaluates the selected interpretability methods across various CNN architectures, datasets, and experimental conditions. The primary goal is to explore their behavior, assess their strengths and weaknesses, as well as their robustness, reliability, and practical usefulness by addressing the following research questions explicitly:

- How does the model quality (under/overfit, proper) influence the explanation?
- How do various interpretability methods differ qualitatively and quantitatively? Can they be compared meaningfully?
- How robust are explanations under noise and adversarial perturbations? Are explanations stable under input changes that preserve the model’s output?
- Why ensemble explanations?

8.1 General Experimental Setup

Experimental Flow. The experiments follow a structured, incremental approach to progressively build understanding:

- **Experiment 1:** Model Quality vs. Explanation Quality (MNIST)
- **Experiment 2:** Comparative Analysis of XAI Methods across increasing dataset complexity:
 - a) XAI method comparison on MNIST
 - b) XAI method comparison on ImageNet
- **Experiment 3:** Robustness of Explanations under diverse perturbations:
 - a) Stability under natural Gaussian noise
 - b) Input invariance tests
 - c) Robustness to adversarial attacks
- **Experiment 4:** Enhancing Explanations via Ensemble Techniques

This progression aims to gain insights into interpretability and explainability across multiple dimensions (complexity, reliability, robustness, and others).

Hardware. The hardware components utilized in the experiments include:

- CPU: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
- GPU: NVIDIA GeForce GTX 1650
- RAM: 32 GB

Models and Architectures. To isolate and evaluate interpretability techniques, CNNs of varying complexity and training quality were utilized. Two main approaches to the selection of models are considered:

- **Custom-trained:** Models trained locally from scratch on MNIST. These allow for controlled experimentation and manipulation of model quality (e.g., underfitting, overfitting, adversarial vulnerability).
- **Pretrained:** Models such as ResNet-18 pre-trained on ImageNet, enabling experiments in high-dimensional settings, moving from toy examples closer to real-world situations.

This mix of models enables comparisons across scale and learning dynamics and supports observations on how model structure and performance influence explanation fidelity.

Datasets. The models used in the experiments are trained and evaluated on datasets representing varying complexity levels:

- **MNIST** [34]: Baseline evaluation, consisting of:
 - 70 000 images of handwritten digits
 - 60 000 train, 10 000 test split
 - each image is 28×28 pixels
- **ImageNet (ILSVRC2012)** [14]: High complexity, large-scale dataset, consisting of:
 - 1 281 167 training, 50 000 validation, and 100 000 test images
 - 1 000 object classes organized by WordNet synsets
 - variable-resolution RGB images (preprocessed to 224×224 for CNN inputs)

All datasets are preprocessed using standard normalization pipelines.

Explanation Methods. Selected methods span gradient-based, advanced post-hoc approaches (see [Chapter 4](#)):

- **Vanilla Gradients (Saliency):** Baseline gradient-based attribution.
- **Integrated Gradients (IG):** Improved reliability via path-integrated gradients.
- **SHAP (SHapley Additive exPlanations):** Game-theoretic, theoretically-grounded feature importance.
- **Grad-CAM** variants: Layer-specific visualization of CNN activations.

The chosen XAI methods represent distinct interpretability paradigms and provide complementary insights into model explanations. **Vanilla Gradients** serves as a *baseline*, directly illustrating the raw gradient-based attributions. **Integrated Gradients** improves upon Vanilla Gradients by smoothing attributions through cumulative gradient integration. By doing this, the noise and fragmentation are addressed, and it enables more stable and coherent explanations. **SHAP** offers a theoretically grounded, model-agnostic method rooted in cooperative game theory. It provides precise, feature-level interpretations suitable for detailed, pixel-level analysis. **Grad-CAM** is an explicitly CNN-specific method. It identifies semantically meaningful regions most influential for the classification decision.

Evaluation Strategy. The selected interpretability methods are evaluated using both *qualitative* and *quantitative* approaches, following the strategies and best practices described in [Chapter 7](#). It results in recognizing the strengths and limitations of each evaluation approach.

Qualitative evaluation: Interpretability is assessed through visual inspection of explanation heatmaps. Human-centered criteria include:

- **Clarity:** Are important regions clearly highlighted without excessive noise?
- **Alignment:** Do explanations correspond to semantically meaningful image regions?
- **Comparability:** Can different methods be distinguished based on visual evidence?

This subjective evaluation complements metric-based assessment and helps identify artifacts or unintuitive behavior not easily captured by numerical scores.

Quantitative evaluation: Explanations are evaluated using metrics from the **Quantus** library [23], covering:

- **Faithfulness:** Do important features have a causal influence on predictions?
- **Robustness:** Are explanations stable under small input changes?
- **Localization:** Do the highlighted regions align with relevant objects?
- **Complexity:** Are explanations sparse and focused?
- **Randomization:** Do explanations fail on randomized models?

Quantitative evaluation ensures reproducibility and provides a solid basis for comparative analysis across methods and settings.

Limitations. As described in [Section 7.4](#), while qualitative evaluation supports intuitive understanding, it remains subjective and context-dependent. Quantitative metrics, though rigorous, may be sensitive to hyperparameters or fail to reflect human judgment. A balanced interpretation of results thus requires *both* perspectives.

8.2 Experiment 1: Model Quality vs. Explanation Quality

The first experiment investigates a key principle in interpretable machine learning: the usefulness and structure of XAI explanations depend on the quality of the model underlying them [42]. Explanation methods aim to faithfully represent the model’s learned function, regardless of whether the model itself is accurate and properly trained. An *underfitting* model, which fails to capture relevant structure in the data, typically produces attributions that highlight non-discriminative or noisy regions. An *overfitted* model may “memorize” spurious correlations or dataset-specific artifacts, which can lead to sharp but misleading explanations. In contrast, a well-regularized, properly trained model that learns generalizable representations tends to produce more coherent and task-relevant explanations. Thus, explanation quality is inherently constrained by what the model has learned [43].

This experiment uses a single XAI method, *vanilla gradients* (saliency, see Subsection 4.3.1), due to its simplicity and high sensitivity to small variations in the model’s decision surface [31], which makes it well-suited for exposing differences in learned representations across model training regimes. As a *local* explanation method (see Section 2.2), saliency operates per individual input and yields instance-specific attributions.

The dataset (MNIST) was selected for its low input dimensionality, allowing controlled experiments. The goal is to demonstrate how the interpretability and task relevance of explanations change with model quality, using both visual inspection and quantitative evaluation via the Quantus [23] library (see Section 7.3).

8.2.1 Setup

- **Dataset:** MNIST, used for training and evaluating all three models under controlled, low-dimensional conditions.
- **Models:** Underfit, Overfit, and Properly Trained CNN, identical architecture (Listing A.1), varying training setups (Table 8.1).
- **Explanation Method:** Vanilla Gradients (absolute value gradients, no smoothing).
- **Quantitative Metrics:** Metrics are carefully selected based on recommendations from MetaQuantus [24] using the Quantus toolkit [23]. Arrows indicate metric interpretation direction (↑: higher scores better; ↓: lower scores better):
 - *Robustness* (Local Lipschitz Estimate ↓)
 - *Faithfulness* (Pixel Flipping ↓)
 - *Randomization* (Random Logit ↓)
 - *Localization* (Pointing Game ↑)
 - *Complexity* (Sparseness ↑)

All attributions were converted to absolute values, normalized per map, and aggregated over $N = 100$ test images. See Section 7.2 for formal definitions of all metrics.

- **Objective:** Investigate how differences in model training quality influence the *task relevance* of explanations, using both visual inspection and quantitative metrics.

Model Comparison. All three models (underfit, overfit, and proper) employed in this experiment share an identical CNN architecture (Listing A.1) but differ in their training regimes (Table 8.1) to achieve desired properties.

The *underfit* model was intentionally trained on a limited subset of just 200 training samples, a single epoch, and dropout regularization (to reduce complexity further). Consequently, its low training and validation accuracies indicate that the model could not learn meaningful patterns from the data, demonstrating the underfitting.

The *overfit* model was excessively trained (20 epochs) on the same small subset of 200 samples without dropout. This resulted in high training accuracy (likely memorization), yet poor generalization to unseen validation data (substantially lower validation accuracy).

The *properly trained* model was carefully optimized using the complete training set (60 000 samples), moderate training epochs (8) guided by loss monitoring, and dropout to balance complexity. This approach yielded consistently high accuracy for both training and validation sets, demonstrating generalization and reliably learned representations. To achieve the desired behaviors of underfitting and overfitting while sharing the same CNN architecture, Table 8.1 shows how individual models differed in their training.

Table 8.1: Training regimes for underfit, overfit, and properly-trained CNN models. They differ mainly in training data size, epochs, and dropout. Test-set accuracies are reported as mean $\pm$ standard deviation over $N = 30$ independent runs with different random seeds.

Model	Train Samples	Epochs	Dropout	Train Acc. (%)	Test Acc. (%)
Underfit	200	1	0.25 / 0.50	8.50	33.87 ± 9.98
Overfit	200	20	0.00 / 0.00	100.00	83.03 ± 1.28
Proper	60,000	8	0.25 / 0.25	99.40	99.10 ± 0.07

8.2.2 Results

The saliency map produced by the **underfitting model** (shown for a single sample in Figure 8.1) is highly noisy and fails to highlight task-relevant pixels. Due to insufficient training, the model incorrectly classifies digit 7 and produces a near-uniform probability distribution over all classes, which reflects its high uncertainty and lack of discriminative power.

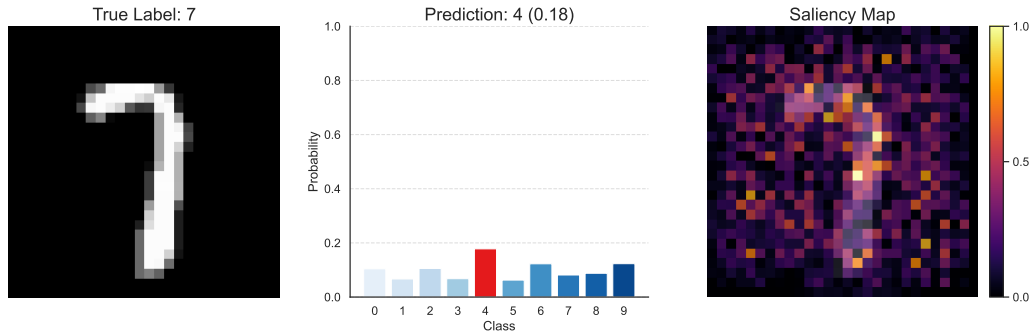


Figure 8.1: **Underfitting model:** Incorrect prediction for a single MNIST test sample (true label 7). From left to right: (1) original 28×28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to $[0,1]$ range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The map lacks structure, is noisy, and fails to highlight task-relevant regions.

In contrast, the **overfitting model** (shown for a single sample in Figure 8.2) produces a much sharper saliency map. The model misclassifies the digit 7 as a 3 and, interestingly, highlights pixel regions reminiscent of the digit 3. This behavior indicates the model’s *memorization* of specific training samples’ properties rather than learning generalizable features. The resulting explanations might become deceptive yet visually plausible.

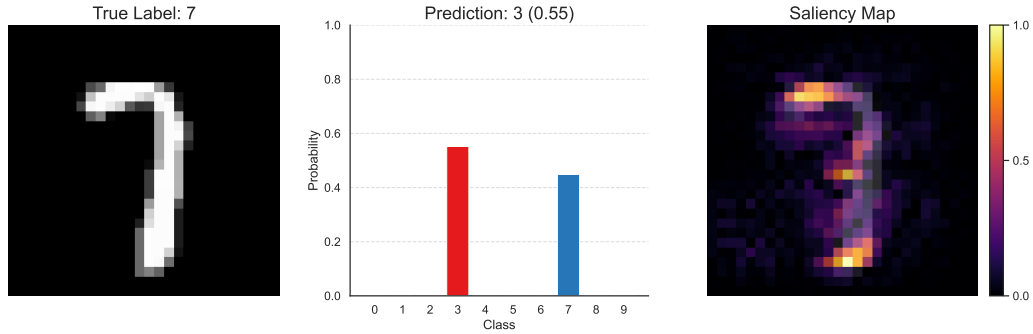


Figure 8.2: **Overfitting model:** Misclassifies digit 7 as 3. From left to right: (1) original 28x28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to [0,1] range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The saliency map is sharper but highlights misleading features, reflecting memorized patterns, not a generalizable structure.

Finally, the saliency map of the **properly trained model** (shown for a single sample in Figure 8.3) highlights relevant pixels corresponding to the digit’s outline. This correctly classified digit demonstrates the model’s learned generalizable patterns and provides an explanation that is neither misleading nor highly noisy.

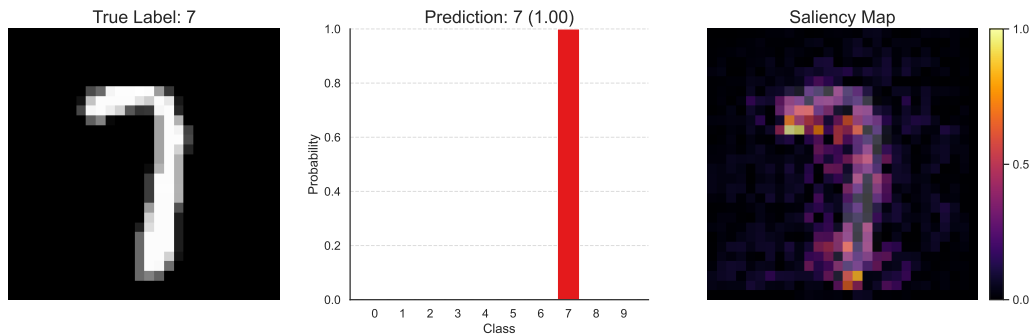


Figure 8.3: **Properly trained model:** Correctly classifies digit 7. From left to right: (1) original 28x28 input image; (2) model’s predicted class probabilities; (3) corresponding saliency map (attributions normalized to [0,1] range; colorbar: 0 = low, 1 = high relevance) indicating each pixel’s influence on the predicted logit. The produced saliency map is less sharp than the map of the overfitting model (likely due to the smoother decision surface of the proper model), and highlights meaningful and task-relevant pixels.

To directly compare the quality gap, Figure 8.4 presents an overlay visualization demonstrating the difference in attribution sharpness and localization between the models. It visually reinforces the core claim of this experiment: task-relevant, more trustworthy explanations emerge only from models that have learned meaningful patterns.

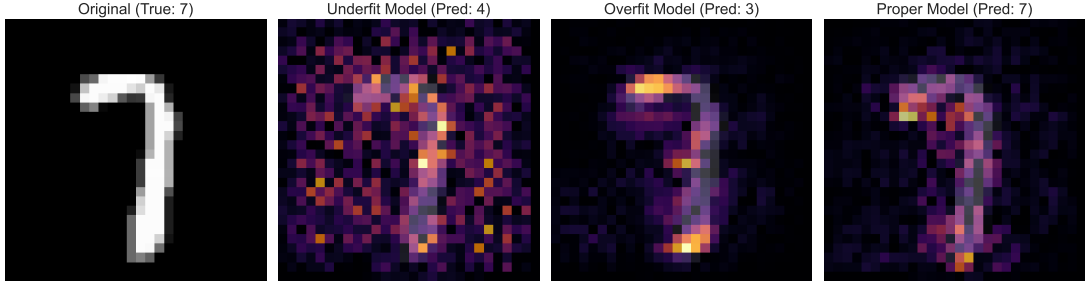


Figure 8.4: Side-by-side comparison of saliency maps from underfitting and overfitting to properly-trained CNN. Heatmap attribution values are normalized to the $[0,1]$ range, indicating each pixel’s influence on the predicted logit. High-performing models yield cleaner, more localized explanations.

In poorly performing models, gradient-based attributions fail to reliably focus on informative input regions. Multiple input samples (four randomly selected test images) across the models are compared in [Figure 8.5](#) to further support this conclusion.

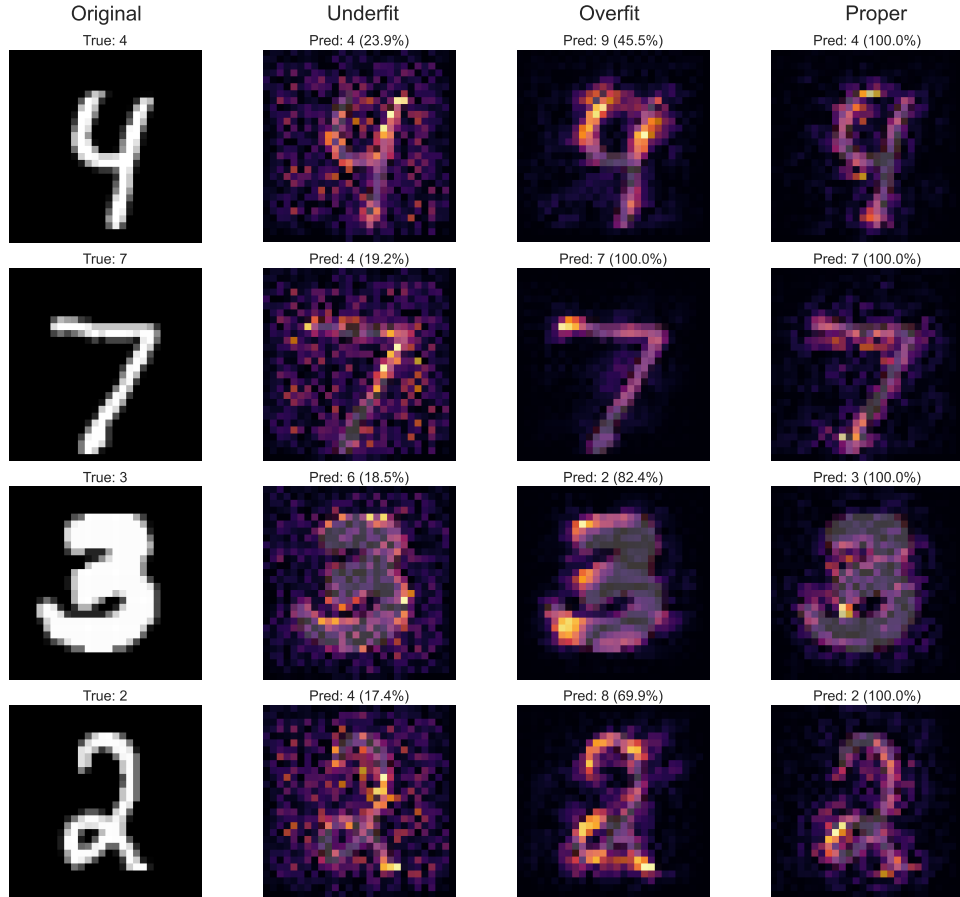


Figure 8.5: Saliency map comparison across underfit, overfit, and properly trained models for four selected test samples (first column). Each row shows the original image and saliency overlays for the predicted class by each model. Heatmap attribution values are normalized to the $[0,1]$ range, indicating each pixel’s influence on the predicted logit. Differences in attribution patterns reflect variation in learned representations and model reliability.

In addition to visual inspection, obtained explanations were evaluated quantitatively using the **Quantus** library [23] with **Figure 8.6** summarizing the results visually (radar plot). To obtain the metrics scores, all attributions were converted to absolute values, scaled to $[0, 1]$ range per map, and aggregated over $N = 100$ test samples. Metrics defined as “lower is better” (Robustness, Faithfulness, Randomization, see **Subsection 8.2.1**) were inverted, so that on every axis of the radar plot, higher denotes better performance.

Before evaluation, two artifacts needed to be addressed. Pixel Flipping metric originally measures the area under the softmax probability $\hat{p}_y(x)$ curve (AUC) as top-attribution pixels are iteratively erased. For an underfit model, with nearly uniform softmax outputs (due to flat logits), the AUC is artificially low (seemingly “good” faithfulness). Random Logit compares attributions for the predicted logit $f_y(x)$ against random logits $f_k(x)$; when all logits are similar, saliency maps also appear similar, falsely inflating the score. To correct these issues, the evaluations are considered only for correctly classified samples, avoiding the confound of misclassification. Then, each metric’s deletion (Pixel Flipping) or insertion (Random Logit) curve was normalized by the class margin:

$$\Delta(x) = f_y(x) - \max_{k \neq y} f_k(x)$$

where $f_k(x)$ is the model’s raw output (logit) for class k

This process ensured anchoring the metric on the model’s logit-margin rather than its softmax scale. For both Pixel Flipping and Random Logit, AUC was computed over the Δ -normalized curve, which in the case of this experiment reflects explanation quality more accurately than the original, unnormalized metric. The resulting radar plot in **Figure 8.6** reveals interpretable trends that align with theoretical expectations when evaluating explanation quality across the three models.

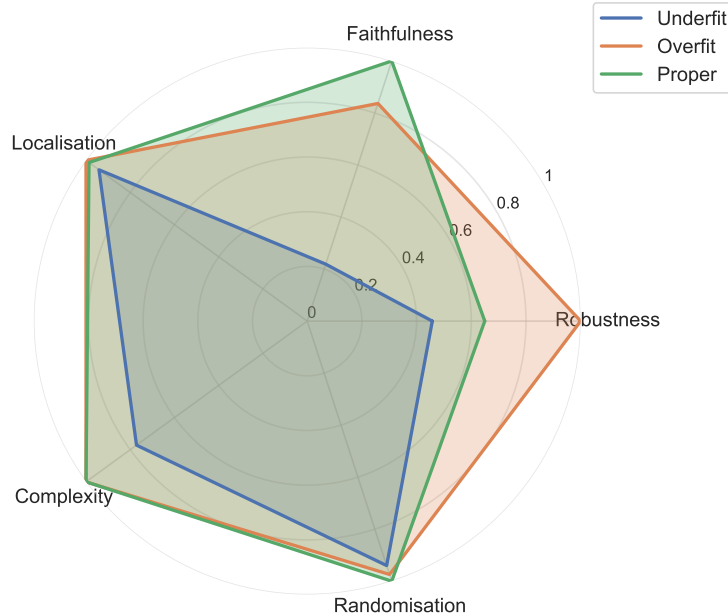


Figure 8.6: Radar plot comparing the explanation quality of underfitted, overfitted, and properly-trained CNN models using five quantitative metrics, based on **Quantus** [23]. Each axis represents a normalized score (0 to 1), with higher values indicating better performance.

The properly trained model’s explanations generally achieve the highest or near-highest scores across the selected metrics, closely followed by the overfitted model’s explanations, while the underfitted model consistently underperforms (except in metrics that originally failed to penalize degenerate outputs, but whose behavior was addressed in this experiment).

Robustness (LocalLipschitzEstimate [3]): The overfitted model scores best on the Local Lipschitz Estimate, indicating that its explanations are most stable under local perturbations. This stems from its extremely peaked logits and locally flat decision surface (even off the training set), so small input noise rarely flips its prediction or significantly alters gradients. The underfit model, by contrast, produces highly erratic gradients due to its inability to learn any coherent decision surface, resulting in the worst robustness score. The properly trained model sits between these extremes.

Faithfulness (PixelFlipping [6]): After correcting for confidence-scale distortion, the properly trained model dominates in the Pixel Flipping metric. Its explanations identify truly influential pixels whose removal most effectively degrades the predicted class score, validating that its attributions reflect the model’s actual reasoning. The overfitted model performs reasonably well, reflecting some learned structure, albeit spurious. The underfit model ranks worst once the metric is adjusted, showing that its attribution maps are not faithful, despite earlier misleading scores caused by flat logits.

Randomization (RandomLogit [60]): The underfit model’s explanations correlate with random logit directions, yielding a poor (but now appropriately penalized) score. The properly trained model outperforms both alternatives, indicating that its saliency maps are class-discriminative and do not generalize across arbitrary class logits, as desired.

Localization (PointingGame [73]): Models produce comparable scores, and the results depend on the choice of the segmentation mask needed by the metric, which likely limits the metric’s sensitivity. Still, the observed order, Proper $\approx$ Overfit $>$ Underfit, matches expectations: meaningful explanations should focus more precisely on the digit’s structure.

Complexity (Sparseness [13]): The properly trained model and overfitted model both yield highly sparse saliency maps, while the underfit model produces more diffuse, low-contrast heatmaps. This suggests that higher model quality leads to more selective and structured attributions, which aligns with visual inspection.

8.2.3 Summary and Discussion

The results support the hypothesis that explanation quality is fundamentally constrained by the quality of the underlying model. When the model fails to learn meaningful patterns, as in the underfitting case, its explanations become noisy, non-discriminative, or misleading, though still faithful to what the model internally represents. The overfitted model achieves higher scores due to sharp but potentially deceptive attributions tied to memorized patterns. Only the properly trained model yields consistently strong performance across all metrics, producing explanations that are both faithful and task-relevant. This underscores a key principle: **trusting explanations for model decision-making requires confidence in the model’s predictive reliability.**

At the same time, explanations remain diagnostically valuable even when the model is flawed. Underfitting produces noisy, unstable maps that reveal a failure to learn structure, prompting retraining or architecture revision. Overfitting results in sharp but spurious saliency patterns, often tied to dataset-specific artifacts, which can inform better regularization or data augmentation. In this sense, XAI can reveal *how* a model fails and suggest corrective actions, even when the model’s predictions should not yet be trusted.

8.3 Experiment 2: Comparative Analysis of XAI Methods

Following Experiment 1 (Section 8.2), which analyzed the importance of model performance and how it affects corresponding explanations, this experiment focuses on comparing the inherent characteristics of different XAI methods. Experiment 2 assumes a fixed, properly trained model for each image classification task to ensure observed differences in explanations stem primarily from the *methods* themselves, not model deficiencies.

Objective: To qualitatively and quantitatively compare selected XAI methods under varying dataset complexity (MNIST $\rightarrow$ ImageNet), assessing their strengths, limitations, and suitability for model understanding.

The analysis proceeds in two stages:

- (a) **Low-complexity** (Subsection 8.3.1): Fundamental attribution behaviors, MNIST [34].
- (b) **High-complexity** (Subsection 8.3.2): Scalability and performance, ImageNet [14].

8.3.1 Experiment 2a: XAI Method Comparison (MNIST)

The first subexperiment evaluates and compares selected XAI methods on MNIST.

Setup.

- **Dataset:** MNIST (model training and evaluation).
- **Model:** Custom CNN (architecture from Listing A.1, test accuracy $\approx 98\%$).
- **Methods:**
 - Vanilla Gradients (Saliency): simple, gradient-based (baseline)
 - Integrated Gradients: axiom-satisfying variant designed for improved attribution
 - SHAP (SHapley Additive exPlanations) variants: Game-theoretic approaches (DeepSHAP, GradientSHAP) providing theoretically grounded explanations

The method selection represents distinct families of XAI techniques commonly used for image classification, and enables their meaningful evaluation and comparison.

- **Quantitative Metrics:** The choice of metrics follows the recommendations from MetaQuantus [24], and their calculation is done via Quantus toolkit [23]. Arrows indicate metric interpretation direction ($\uparrow$: higher scores better; $\downarrow$: lower scores better):
 - *Robustness* (Local Lipschitz Estimate $\downarrow$)
 - *Faithfulness* (Pixel Flipping $\downarrow$)
 - *Randomization* (Random Logit $\downarrow$)
 - *Localization* (Pointing Game $\uparrow$)
 - *Complexity* (Sparseness $\uparrow$)

All attributions were converted to absolute values, normalized to the range $[-1, 1]$, and aggregated over $N = 100$ test images to ensure direct comparability across methods and consistent qualitative and quantitative assessments. See Section 7.2 for formal definitions of all metrics.

Why attribution methods? To understand how the XAI methods might be helpful, the analysis first examines the explanatory potential using DeepSHAP (Figure 8.7). The figure shows SHAP’s ability to provide fine-grained, contrastive explanations, differentiating pixel contributions that support the predicted class versus alternatives. DeepSHAP demonstrates how specific features (or their absence) influence the prediction and how minor input variations can lead to misclassifications, indicating model sensitivity or dataset imbalances.

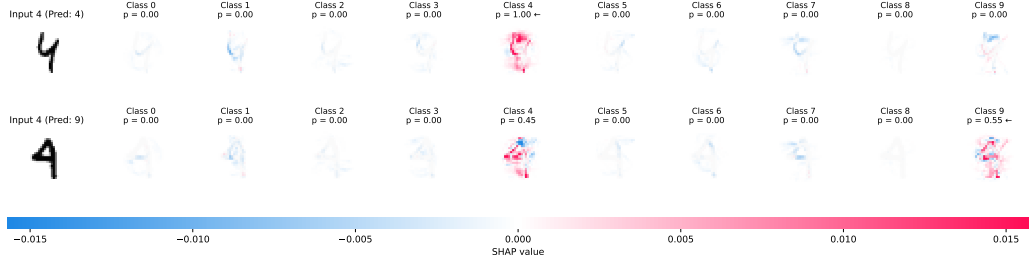


Figure 8.7: **DeepSHAP analysis** of input digit 4, comparing a correct prediction (top) with an incorrect one (bottom) across ten digit classes (0-9). Red/blue indicate positive/negative attributions. In the top row, the model confidently predicts correct class 4 ($p = 1.00$); SHAP values highlight that the absence of a top connection (a feature typical for digit 9) strongly supports identifying this digit as a 4. In the bottom row, a slight variation in writing (presence of the top connection) confuses the model, resulting in a misclassification favoring digit 9 ($p = 0.55$) over digit 4 ($p = 0.45$).

With the demonstrated explanatory potential of XAI methods (DeepShap), the analysis continues with qualitative and quantitative assessments of the selected XAI methods.

Qualitative Evaluation. Figure 8.8 visually compares Vanilla Gradients, Integrated Gradients, and SHAP on a correctly classified digit 8. All three methods produce both positive and negative attributions, but differ in their behavior.

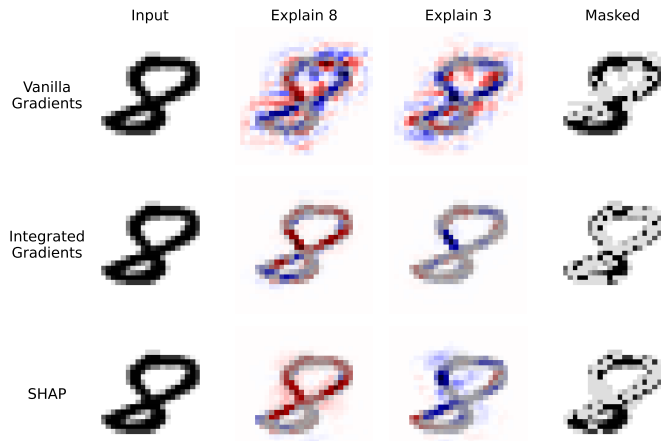


Figure 8.8: Comparison of the selected attribution methods on a correctly classified MNIST digit 8. Columns show: Original input, Attributions for class 8 (correct), Attributions for class 3 (incorrect), Input with top 10 % influential pixels (for class 8) masked. Red/blue indicate positive/negative attributions (normalized $[-1, 1]$). The attribution maps highlight important regions per method; the masked samples reveal the effect of removing them.

The comparison illustrates method-specific interpretability characteristics, and key visual differences emerge:

- **Vanilla Gradients:** Produces noisy, diffuse attributions, making it difficult to pinpoint crucial features.
- **Integrated Gradients:** Results in sharper, more coherent attribution maps that highlight the structural elements of digit 8.
- **SHAP:** Provides detailed attributions, effectively separating positive (red, supporting '8') and negative (blue, contradicting '8', e.g., features suggesting '3') evidence.

Masking the top 10% pixels (last column) visually confirms these characteristics: the regions identified by Integrated Gradients and SHAP appear more structurally relevant to the digit 8 compared to the scattered pixels highlighted by Vanilla Gradients.

Quantitative Evaluation. Figure 8.9 displays the distribution of the change in the model's log-odds score for the predicted class after masking the top 10% most salient pixels identified by each method (based on absolute attribution value, see Figure 8.8). The metric plotted is:

$$\Delta \log\text{-odds} = \log\text{-odds}(\text{original}) - \log\text{-odds}(\text{masked}) \quad (8.1)$$

A *higher positive* value indicates a larger drop in the model's confidence after masking, signifying that the method successfully identified highly influential pixels crucial for the original prediction.

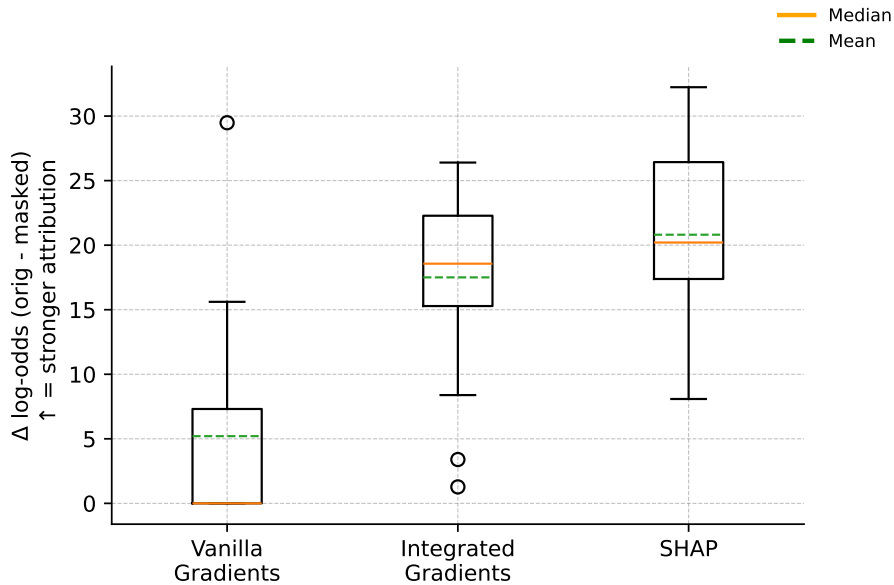


Figure 8.9: Attribution quality measured by the change in log-odds for the predicted class after masking the top 10% most salient pixels ($\Delta \log\text{-odds}$). Higher values indicate that more influential pixels were identified. Boxplots show distribution over 100 MNIST test samples for Vanilla Gradients, Integrated Gradients, and SHAP.

The boxplot reveals:

- **Integrated Gradients** and **SHAP** achieve significantly higher median $\Delta\log$ -odds values compared to Vanilla Gradients. This confirms they more effectively identify pixels critical to the model’s prediction, as removing these pixels causes a larger drop in confidence.
- **Integrated Gradients** method exhibits the tightest distribution – smallest Interquartile Range (IQR), which indicates the most consistent performance in identifying influential pixels across different test samples.
- **SHAP** shows comparable median performance to Integrated Gradients but displays greater variability (larger IQR and wider whisker range), suggesting its effectiveness, while high on average, is less consistent sample-to-sample compared to Integrated Gradients in this specific test setting.

Further insight into method behavior comes from analyzing the distribution of normalized pixel-wise attribution values, as shown in [Figure 8.10](#). The width of the violin shape indicates the density of attribution values at that level.

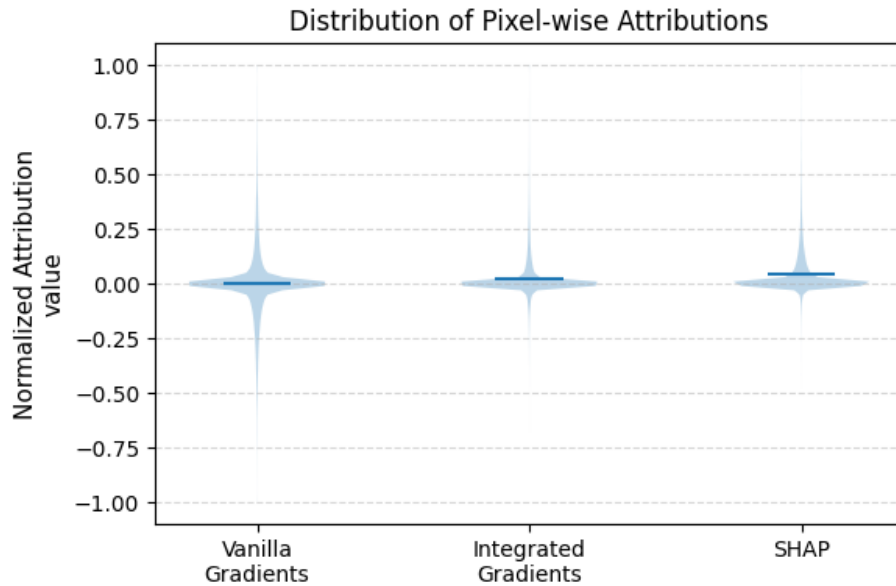


Figure 8.10: Violin plot showing the distribution of normalized pixel-wise attribution values for Vanilla Gradients, Integrated Gradients, and SHAP across $N = 100$ test samples (MNIST). The plot displays density estimates, with wider sections indicating higher concentrations of values. The thicker blue line indicates the mean.

Interpreting the distributions shown in the violin plot:

- **Vanilla Gradients:** Exhibits the widest overall spread, with long, thin tails extending towards high positive and negative values (-1 and 1). However, the central density is sharply peaked around zero. This suggests that while most pixels receive near-zero attribution, a non-negligible number receive extreme scores, consistent with the visually noisy and fragmented heatmaps observed earlier.

- **Integrated Gradients:** Displays the narrowest distribution, tightly concentrated around zero with very short tails. This indicates that Integrated Gradients assigns significant importance scores (far from zero) to relatively few pixels, aligning with its characteristic of producing smoother, more focused, and coherent attribution maps.
- **SHAP:** Shows an intermediate spread, wider than Integrated Gradients but narrower than Vanilla Gradients, particularly in the tails. The distribution is relatively symmetric around zero. This reflects SHAP’s ability to assign both positive and negative importance scores across a broader range of pixels than Integrated Gradients, capturing finer details, but without the extreme outliers seen in Vanilla Gradients.

These distribution shapes reinforce the qualitative differences observed, linking Integrated Gradients’ focus to its narrow value range, Vanilla Gradients’ noise to its extreme value tails, and SHAP’s detailed nature to its intermediate, balanced spread.

The radar plot in [Figure 8.11](#) synthesizes the quantitative evaluation across five distinct criteria via the Quantus [\[23\]](#) framework, as outlined in the experimental setup. The raw metric scores were scaled such that higher values (closer to 1.0) consistently represent better performance on the plot for all axes.

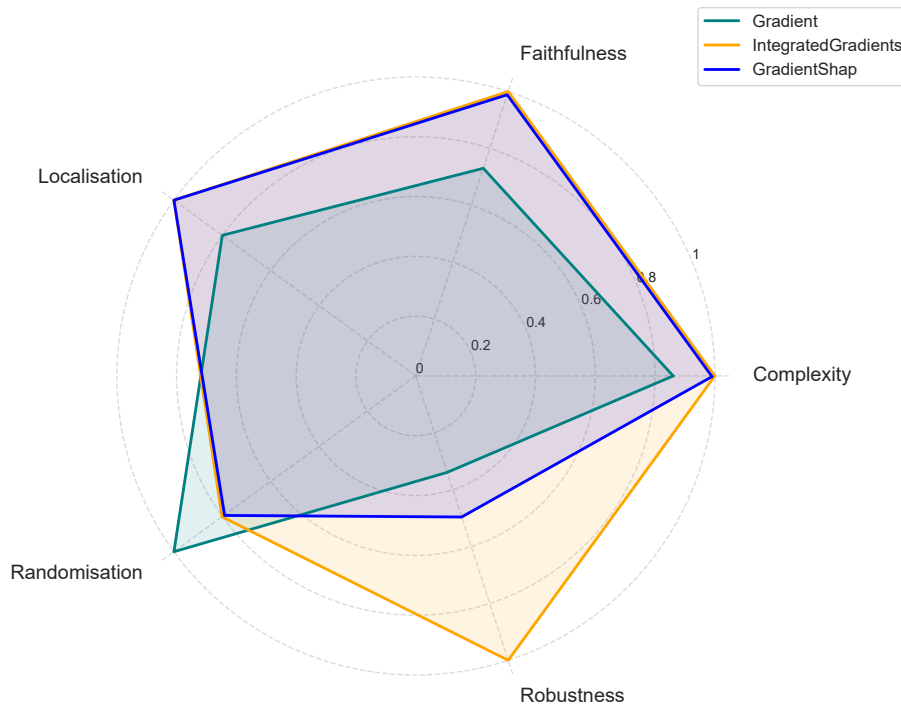


Figure 8.11: Aggregated quantitative evaluation for Vanilla Gradients (Saliency), Integrated Gradients, and SHAP on MNIST over $N = 100$ samples. Values are scaled $[0, 1]$, where 1.0 (outer edge) indicates the best performance on that metric axis.

Key takeaways from the radar plot, reflecting the scaled scores:

- **Integrated Gradients:** Shows dominating performance (scaled score ≈ 1.0) on all axes except for Randomization. High values on Faithfulness and Robustness indicate

that its explanations strongly correlate with model behavior and are stable under input perturbations.

- **SHAP:** Achieves similar scores to Integrated Gradients, but underperforms on Robustness.
- **Vanilla Gradients (Saliency):** Underperforms along all axes compared to both Integrated Gradients and SHAP (smallest total area on the plot), except for the Randomization.

Integrated Gradients and SHAP outperform the Vanilla Gradients method on all selected quantitative metrics, except Randomization. Their superior performance in Faithfulness and Robustness, coupled with high scores in Complexity, suggests they offer more reliable explanations overall.

8.3.2 Experiment 2b: XAI Method Comparison (ImageNet)

This experiment extends from the low-dimensional MNIST scenario to analyze interpretability methods on a more complex and realistic dataset, **ImageNet**, using a pretrained ResNet model. ImageNet’s higher dimensionality (224×224 RGB images) and greater visual complexity present an important evaluation setting to test how well selected XAI methods can generalize and scale from simpler benchmarks closer to real-world applications and whether the performance trends observed on MNIST (Subsection 8.3.1) persist with ImageNet.

Setup.

- **Dataset:** ImageNet ILSVRC 2012 (used for the pretrained ResNet50 model).
- **Model:** ResNet50 (from `torchvision`) pretrained on ImageNet.
- **Methods:**
 - Vanilla Gradients (Saliency)
 - Integrated Gradients
 - SHAP (GradientSHAP)

The XAI method selection process was the same as in Subsection 8.3.1.

- **Quantitative Metrics:** The choice of metrics follows the recommendations from **MetaQuantus** [24], and their calculation is done via **Quantus** toolkit [23]. Arrows indicate metric interpretation direction ($\uparrow$: higher scores better; $\downarrow$: lower scores better):
 - *Robustness* (Local Lipschitz Estimate $\downarrow$)
 - *Faithfulness* (Pixel Flipping $\downarrow$)
 - *Randomization* (Model Parameter Randomisation test $\downarrow$)
 - *Localization* (Pointing Game $\uparrow$)
 - *Complexity* (Sparseness $\uparrow$)

All attributions were converted to absolute values, normalized to the range $[-1, 1]$, and aggregated over $N = 100$ test images to ensure direct comparability across methods and consistent qualitative and quantitative assessments. See Section 7.2 for formal definitions of all metrics.

Qualitative Evaluation. To provide qualitative insights into the selected XAI methods, [Figure 8.12](#) displays explanation heatmaps from the methods for three ImageNet input samples. All methods produce semantically plausible attributions, yet differ in sharpness, focus, and alignment with class-relevant regions.

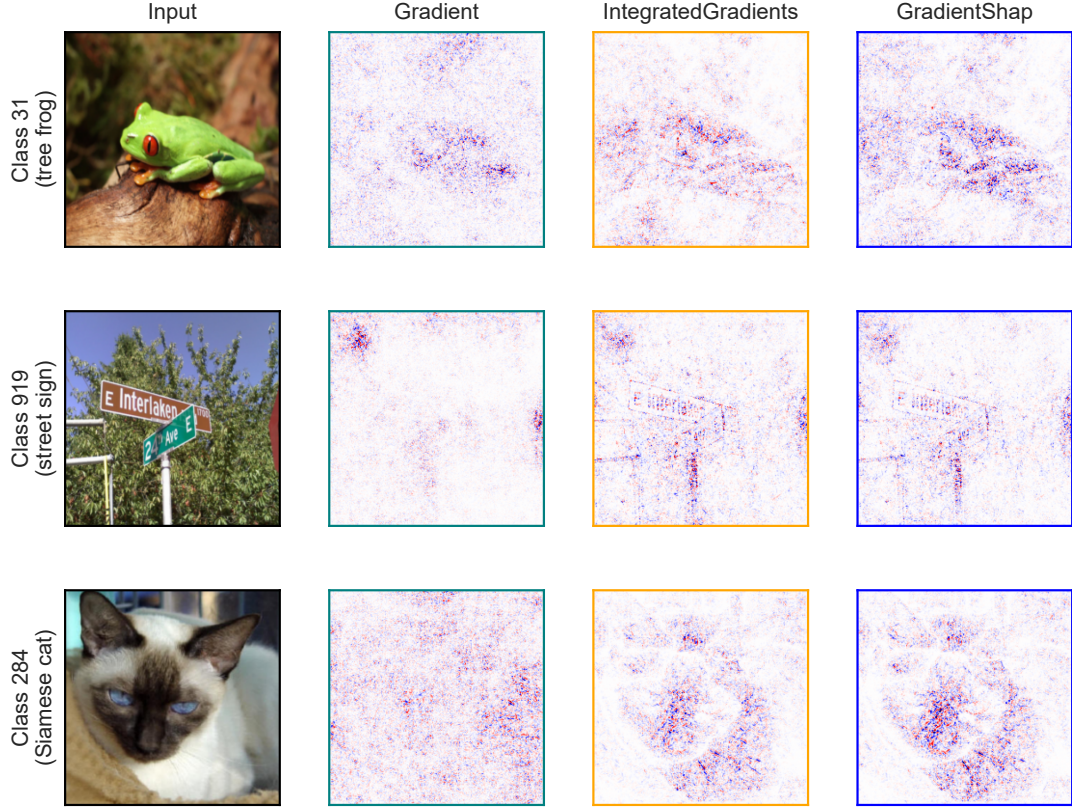


Figure 8.12: Qualitative comparison on **ImageNet** sample (three input images, column 1) for Vanilla Gradients ([teal](#)), Integrated Gradients ([orange](#)), GradientSHAP ([blue](#)). Attributions are normalized to $[-1, 1]$.

From the visual assessment above, similar trends as in the previous experiment with MNIST data (see [Subsection 8.3.1](#)) can be observed, namely, the Vanilla Gradients resulted in fragmented, highly noisy attribution values, which are less precisely aligned with object-relevant regions compared to the other two methods. Integrated Gradients and GradientSHAP attributions closely resemble each other. Both methods are more focused, more effectively highlighting object-specific areas, and are overall more coherent compared to Vanilla Gradients.

Although the visual properties of the selected XAI methods are observable, it is difficult to unambiguously prefer one method purely based on visual assessment. Thus, qualitative evaluation, which is subjective, is again complemented with a more systematic evaluation approach via quantitative metrics.

Quantitative Evaluation. As mentioned in the setup of this experiment, the quantitative evaluation is based on the **Quantus** framework [23] with the specific quantitative metric selection recommended in **MetaQuantus** [24].

The radar plot in **Figure 8.13** summarizes the quantitative comparison on **ImageNet** subset of $N = 100$ samples using the obtained metric scores across the five selected criteria: *Faithfulness*, *Robustness*, *Localization*, *Complexity*, and *Randomization* (discussed in **Section 7.2**).

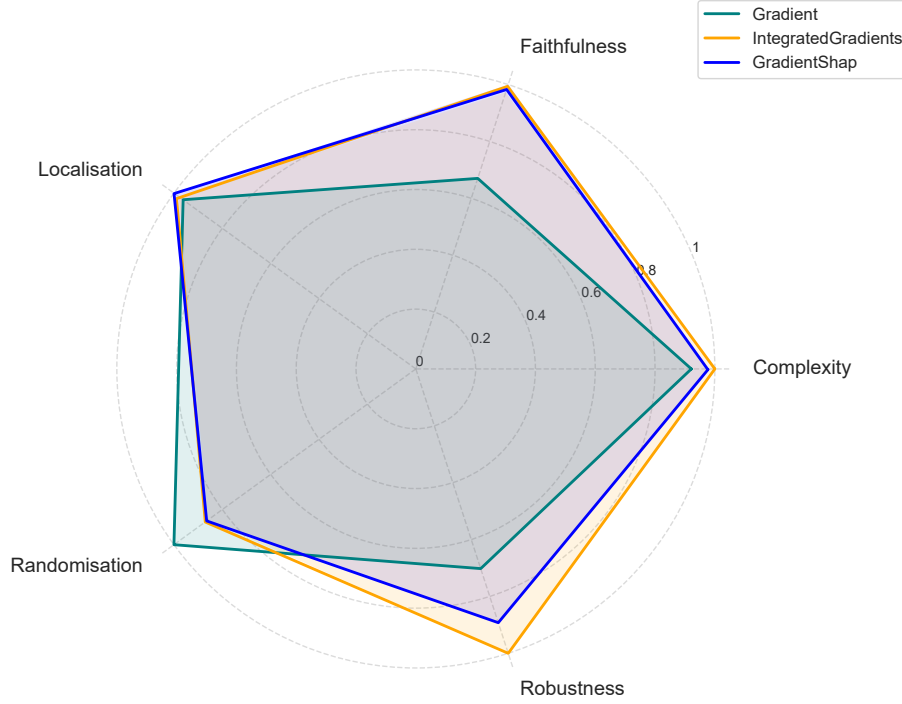


Figure 8.13: Aggregated quantitative evaluation for Vanilla Gradients (Saliency), Integrated Gradients, and SHAP on **ImageNet** over $N = 100$ samples. Values are scaled $[0, 1]$, where 1.0 (outer edge) indicates the best performance on that metric axis.

From the quantitative assessment, it can be observed that the results for **ImageNet** experiment are similar to the results for **MNIST** discussed previously. Vanilla Gradients method again underperforms both Integrated Gradients and GradientSHAP, except for the Randomization axis. Integrated Gradients and GradientSHAP achieve similar metric scores, except for the Robustness metric, where GradientSHAP again underperforms.

Experiment 2 Summary: Method Comparison Across Complexities. This experiment systematically compared Vanilla Gradients (Saliency), Integrated Gradients, and SHAP variants across **MNIST** and **ImageNet** datasets.

Visual inspection of the selected XAI methods demonstrated the behavior of the selected XAI methods. Vanilla Gradients consistently produced noisier attributions, while Integrated Gradients and GradientSHAP resulted in explanations similar to each other, better focusing on task-relevant regions.

The quantitative assessment confirmed similar behavior of the XAI methods across task complexity (MNIST + simple CNN vs. ImageNet + pretrained ResNet50), which is highlighted in [Figure 8.14](#), showing *rank-based* ordering of the obtained quantitative metric scores. Values are scaled so that for all axes, higher means better.

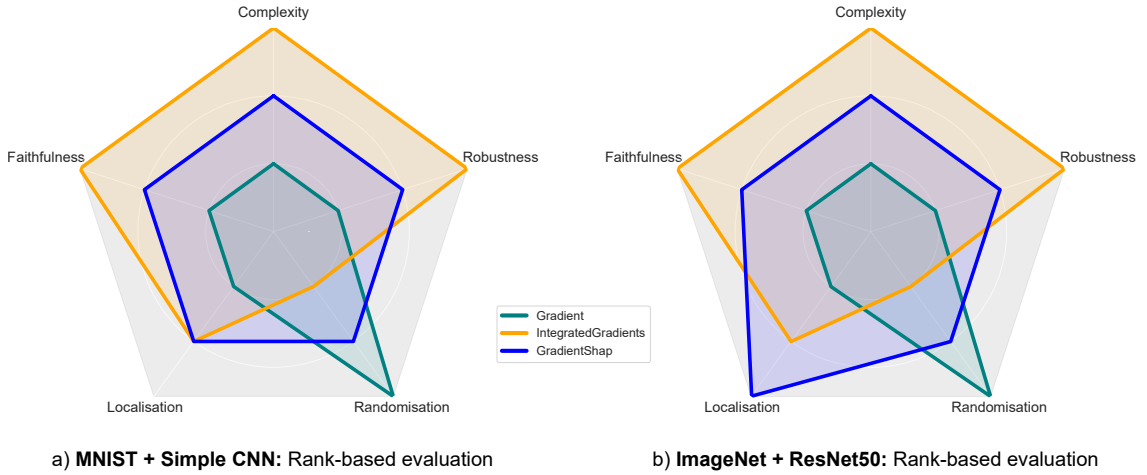


Figure 8.14: Rank-based comparison of the results from quantitative evaluation analysis of the selected XAI methods. Higher rank means better relative performance.

The overall behavior of the selected XAI methods was preserved, except for the Localization, where GradientSHAP took the lead. In the MNIST scenario, both Integrated Gradients and GradientSHAP achieved a perfect score (Localization is highly dependent on the choice of segmentation mask, see [Section 7.2](#)), thus ranking on the same level.

- **Consistency:** The relative performance ranking observed on MNIST largely held on the more complex ImageNet dataset: $IG \gtrsim SHAP \gg \text{Saliency}$.
- **Integrated Gradients:** Consistently emerged as the top performer, delivering explanations that were qualitatively clearer and quantitatively outperformed other methods in terms of faithfulness and robustness. Its strengths persisted across dataset complexity and model architectures.
- **SHAP (DeepSHAP/GradientSHAP):** Provided valuable fine-grained and contrastive insights (especially DeepSHAP on MNIST) and performed well quantitatively, particularly on localization and complexity metrics.
- **Vanilla Gradients (Saliency):** Served as a useful baseline but consistently produced noisy explanations and scored poorly on most quantitative metrics, limiting its practical utility for deep insights.

The results highlight the necessity of combining qualitative inspection with rigorous quantitative metrics. The analysis also demonstrated that quantitative rankings can depend on the choices of metric and their parameter settings, which emphasized the need for careful evaluation design and robustness checks.

8.4 Experiment 3: Robustness of Explanations

This set of experiments investigates the property of robustness in XAI methods. Reliable explanations should remain stable or predictable under common input perturbations encountered in real-world conditions. Robustness is examined from two perspectives. Firstly, the *stability under natural random noise* (Subsection 8.4.1) simulates minor environmental changes. Second, *input invariance testing* (Subsection 8.4.2), examines whether explanations change when inputs are modified in ways that leave the model’s prediction unchanged.

8.4.1 Experiment 3a: Stability under Natural Noise

Evaluating the reliability of XAI methods requires understanding their behavior under realistic input variations. Noise can naturally arise from sensor imperfections, transmission errors, or minor environmental fluctuations. This experiment assesses how the selected explanation methods degrade under increasing levels of Gaussian noise.

Hypothesis. It is expected that all selected XAI methods will exhibit a decrease in explanation stability (measured by SSIM score) as the level of Gaussian noise (σ) applied to the input increases. Gradient-based methods will show varying sensitivities. Specifically, Vanilla Gradients are expected to be the most sensitive, with their explanations degrading rapidly and becoming noisy. Integrated Gradients, due to its averaging nature across an integration path, will demonstrate improved stability over Vanilla Gradients. SHAP-based methods (GradientSHAP and DeepSHAP) are expected to maintain higher structural similarity with the original explanation even at significant noise levels.

Setup.

- **Dataset:** MNIST (used for both model training and evaluation).
- **Model:** Custom CNN (architecture from Listing A.1, test accuracy $\approx 98\%$).
- **Noise protocol:** additive i.i.d. Gaussian noise $\sigma \in \{0, 0.05, \dots, 1\}$; images are clipped to the normalized range.
- **Methods:**
 - Vanilla Gradients (Saliency)
 - Integrated Gradients
 - GradientSHAP (Gaussian baseline $\mathcal{N}(0, 0.1)$)
 - DeepSHAP (background = 100 random test images)
- **Metrics:**
 - *SSIM* to measure structural similarity between the clean and noisy explanation heatmaps. Its values range from -1 to 1, with values closer to 1 indicating greater robustness under noise.
- **Sample size:** $N = 100$ test sample images from MNIST.

Small sensor irregularities are ubiquitous; robustness to such *natural* noise therefore indicates explanation reliability.

Results. Figure 8.15 shows a single MNIST digit 8 being progressively corrupted with additive independent and identically distributed Gaussian noise. For each noisy input x' , the attributions $f(x')$ are computed using Vanilla Gradients, Integrated Gradients, SHAP variants (GradientSHAP and DeepSHAP), and compared with the clean reference attribution $f(x)$ (corresponding to $\sigma = 0$). The attributions in this figure are generated with respect to the ground truth label of the digit.

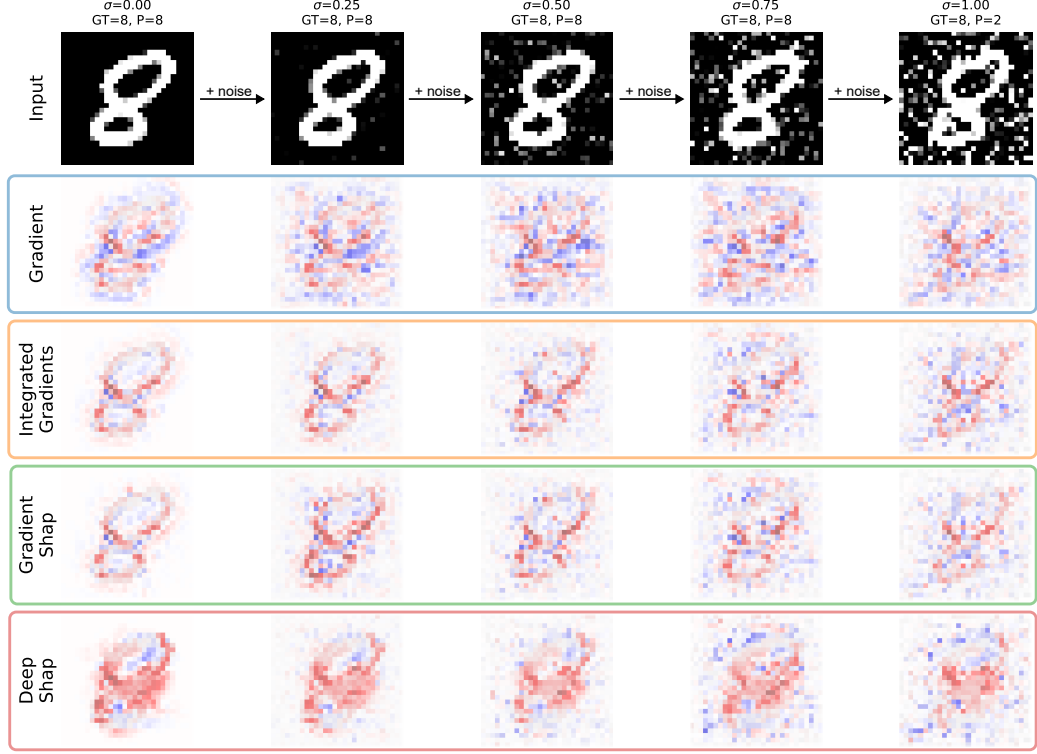


Figure 8.15: Stability of selected XAI methods' explanations under Gaussian noise for a CNN prediction on an MNIST digit 8. Columns depict increasing noise levels (standard deviation σ , left to right from 0 to 1). The first row shows the input images as the noise is being introduced. Subsequent rows display explanation heatmaps for Vanilla Gradients, Integrated Gradients, GradientSHAP, and DeepSHAP, all generated with respect to the true class label. Visual degradation illustrates the sensitivity of each method to noise.

Visually, as expected, *all* of the selected XAI methods demonstrate sensitivity to the introduced noise; the attribution maps degrade. Vanilla Gradients show high sensitivity even at relatively low noise levels (e.g., $\sigma = 0.25$). The heatmap, which already starts noisy, begins to lose focus and incorporate noise patterns, essentially highlighting pixels irrelevant to the digit's structure. Integrated Gradients and GradientSHAP mostly preserve the initial digit-relevant regions, up to approximately $\sigma = 0.5$ where the explanations continue to degrade further.

The quantitative results are depicted in Figure 8.16 (SSIM scores vs. σ) and are further summarized by the Area Under the Stability Curve (AUC-robustness) scores (Table 8.2) calculated from these plots, which provides an additional view on the subject.

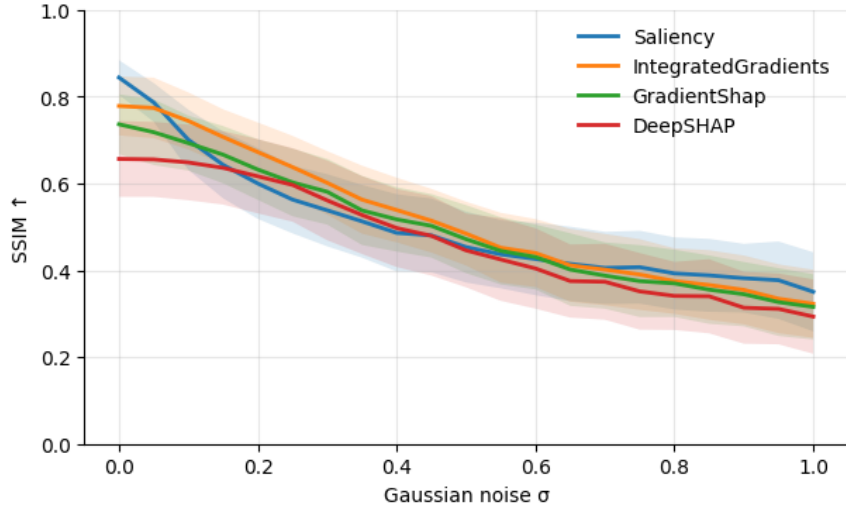


Figure 8.16: Comparison of explanation stability under Gaussian noise using SSIM. The plot tracks the mean SSIM scores (100 test samples, shaded areas indicate ± 1 standard deviation) for Vanilla Gradients, Integrated Gradients, GradientSHAP, and DeepSHAP. SSIM scores compare each method’s explanation for the original MNIST input ($\sigma = 0$) against explanations generated under increasing noise levels (σ). Higher SSIM indicates greater structural similarity to the original explanation (higher stability).

The AUC-robustness scores were calculated from the obtained SSIM results for each of the explanation methods, and are summarized in [Table 8.2](#).

Table 8.2: AUC-robustness scores for each XAI method.

Method	AUC $\uparrow$	Rank
Integrated Gradients	0.530	#1
GradientSHAP	0.509	#2
DeepSHAP	0.468	#3
Vanilla Gradients	0.466	#4

Discussion. Methods that integrate along a path (Integrated Gradients) or ensemble multiple baselines (GradientSHAP) average out high-frequency gradient noise and thus remain reliable under realistic sensor noise. DeepSHAP inherits baseline sensitivity; using a domain-specific background could potentially close the gap.

This experiment demonstrates that the selection of explanation methods should carefully consider the anticipated noise characteristics of the deployment environment. Methods that are highly sensitive to input variations (e.g., Vanilla Gradients) may produce inconsistent explanations in noisy real-world settings, which could undermine user trust and interpretation accuracy. While stability against random noise is a crucial aspect of reliability, another fundamental characteristic is how explanations behave under transformations that, ideally, should leave them unchanged. This consideration motivates the investigation of **input invariance** properties, which will be explored in the following experiment.

8.4.2 Experiment 3b: Input Invariance Tests

Beyond stability to random noise, a more fundamental test of explanation reliability is *input invariance*. Input invariance is the requirement that explanations should not change under transformations that leave the model’s output unchanged [31]. If an explanation changes despite identical model outputs, it suggests the method is attributing importance to factors irrelevant to the model’s actual decision process (encoding artifacts of the chosen baseline, sampling strategy, or implementation). This experiment investigates the invariance of different methods under specific, controlled input shifts.

Hypothesis. Gradient-based methods that evaluate the gradient at the input (Vanilla Gradients) will satisfy input invariance under a constant shift with bias correction. Path-integral methods with a fixed zero baseline (IG-zero, GS-zero) will “leak” the shift pattern into their attributions, whereas using a black-image baseline (IG-black, GS-black) will cancel the shift and yield nearly identical attributions (up to sampling noise for GS-black, which relies on stochastic sampling around the baseline).

Setup.

- **Dataset:** MNIST (model training and evaluation).
- **Input:** a single MNIST digit $x \in [0, 1]^{28 \times 28}$ used for the input invariance test.
- **Models:**
 - a) *Model 1:* Reference CNN model f_1 (based on Listing A.1, test accuracy $\approx 98\%$).
 - b) *Model 2:* “Shifted twin” model f_2 with input $x' = x + m$, first-layer bias is adjusted to $b_2 = b_1 - w^\top m$ such that $f_1(x) = f_2(x')$ for all x despite the input shift m (creating valid invariance test scenario). The constant shift is set to $m = -1/\text{std}(\mathcal{D})$, preserving output equivalence across the network pair.
- **Transformations:**
 - *Constant shift:* m (to test basic input invariance).
 - *Pattern shift:* checkerboard (to test for invariance under structured but irrelevant input modifications).
- **Attribution Methods:**
 - Vanilla Gradients (Saliency)
 - Integrated Gradients (zero vs. black baseline)
 - GradientSHAP (zero vs. black baseline)
- **Metrics:** For each method, the absolute deviation $\Delta = A(x) - A(x')$, which is the element-wise difference between the two attribution maps (for original and shifted inputs), is computed and summarized via $\max |\Delta|$ and Pearson correlation of the two maps. Additional statistics are provided in Appendix B.

Results. The *constant-shift* experiment for input x (digit 9) and its shifted counterpart x' is visually demonstrated in Figure 8.17. Applying a uniform offset of $-1/\text{std}$ to every pixel and compensating the bias in the first convolutional layer produces two functionally identical networks. In the top-row heatmaps (original input) versus the bottom row (shifted input + bias fix), it can be seen that Vanilla Gradients remains unchanged. Integrated Gradients with zero baseline (IG-zero) “leaks” the constant shift: its attributions on the shifted input show a uniform patch of spurious importance, correlating almost zero with the original map. In contrast, IG with a black-image baseline (dataset minimum, IG-black) cancels the offset because the baseline tracks the shift and its maps under shift overlay identically with the originals. GradientSHAP (zero baseline, GS-zero) mirrors IG-zero’s failure: the fixed zero baseline does not track the shift, so GS-zero highlights the uniform shift region. GS-black, however, inherits the black baseline’s invariance: its shifted vs. original maps align up to tiny Monte Carlo noise (corr > 0.99, with 12 800 samples, see Table 8.3).

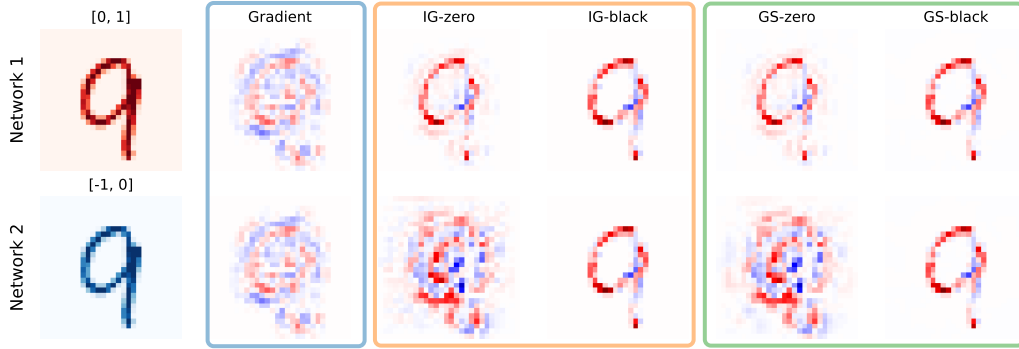


Figure 8.17: **Constant-Shift Invariance Test.** Attribution maps for Vanilla Gradients, Integrated Gradients (zero/black baselines) and GradientSHAP (zero/black baselines) on MNIST before (top row) and after (bottom row) applying a uniform shift of $-1/\text{std}$ with bias compensation. Vanilla Gradients and all black-baseline methods remain identical under shift, whereas zero-baseline methods “leak” the uniform offset into their heatmaps.

The numerical summary is given in Table 8.3, which demonstrates that only Gradient and black-baseline methods achieve $\max|\Delta| \approx 0$ and $\text{corr} \approx 1$.

Table 8.3: Constant-shift invariance summary (100 test images). A method is considered input invariant if it meets both criteria, $\max|\Delta| < 10^{-2}$, $\bar{\rho} > 0.999$, supported by visual assessment [31]. Full descriptive statistics are provided in Appendix B.

Method	$\overline{\max \Delta }$	$\bar{\rho}$ (Pearson)	Invariant?
Gradient	$4.6 \times 10^{-4} \pm 4.5 \times 10^{-3}$	$1.0 \pm 9.0 \times 10^{-5}$	✓
IG-zero	$8.06 \times 10^{-1} \pm 2.51 \times 10^{-1}$	-0.030 ± 0.047	✗
IG-black	$3.6 \times 10^{-5} \pm 1.9 \times 10^{-4}$	$1.0 \pm 1.6 \times 10^{-7}$	✓
GS-zero	$7.93 \times 10^{-1} \pm 2.48 \times 10^{-1}$	-0.026 ± 0.049	✗
GS-black	$5.13 \times 10^{-3} \pm 2.08 \times 10^{-3}$	$1.0 \pm 9.1 \times 10^{-6}$	✓

The next test is to apply a spatially varying checkerboard mask on top of the input, *without* any bias correction. Unlike a uniform offset (which could be absorbed by adjusting each filter’s bias), non-uniform perturbations cannot be cancelled by a single per-filter bias term.

The input invariance test experiment is depicted in Figure 8.18.

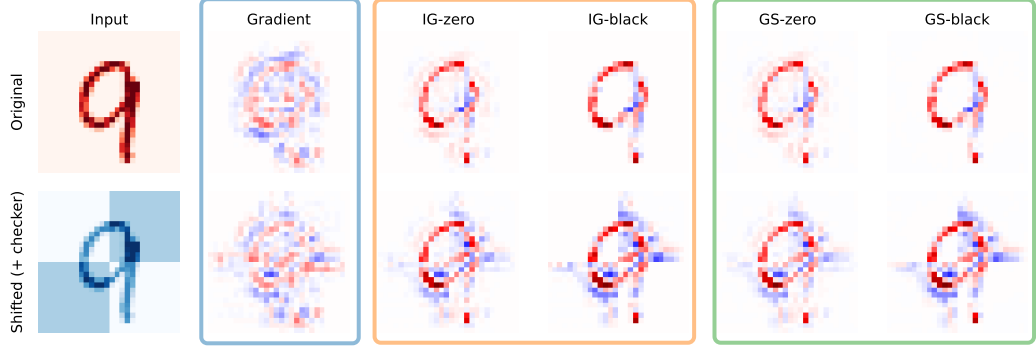


Figure 8.18: **Checkerboard-Pattern Attack.** Attribution maps for the XAI methods on MNIST digit before (top row) and after (bottom row) adding a 2×2 checkerboard shift. None of the selected methods stays invariant; the checkerboard pattern “leaks” into attributions of all methods.

The network’s prediction remains unchanged, but now *all* methods, even those using a black baseline, “leak” the checker pattern into their attributions. This structured-shift test confirms that black baselines only guarantee invariance for offsets that the baseline itself tracks (constant shifts), but not for arbitrary input perturbations.

Table 8.4: Checkerboard perturbation: invariance summary (100 test images). A method is input invariant if it meets both criteria: $\max |\Delta| < 10^{-2}$ and $\bar{\rho} > 0.999$, supported by visual assessment [31]. Full descriptive statistics are provided in Appendix B.

Method	$\overline{\max \Delta }$	$\bar{\rho}$ (Pearson)	Invariant?
Gradient	$2.27 \times 10^{-1} \pm 5.43 \times 10^{-2}$	0.796 ± 0.051	✗
IG-zero	$2.73 \times 10^{-1} \pm 7.62 \times 10^{-2}$	0.881 ± 0.035	✗
IG-black	$3.02 \times 10^{-1} \pm 8.46 \times 10^{-2}$	0.908 ± 0.026	✗
GS-zero	$2.59 \times 10^{-1} \pm 7.20 \times 10^{-2}$	0.889 ± 0.032	✗
GS-black	$2.88 \times 10^{-1} \pm 8.07 \times 10^{-2}$	0.914 ± 0.024	✗

The results confirm the critical role of the *baseline*, namely:

- Fixed baselines “lock in” spurious structure (IG-zero, GS-zero).
- Data-dependent baselines cancel constant shifts but not structured ones.
- In this setup, none of the methods were truly agnostic to arbitrary input changes.

Implication: Users should be aware that baseline choice can dominate saliency maps under seemingly irrelevant transformations.

Invariance failure does not imply a method is invalid, but suggests the attribution may reflect design choices (e.g., baseline) more than the model’s actual decision process. In sensitive domains, this can mislead users or obscure behavior. Conversely, invariance alone doesn’t guarantee explanation quality and should be considered as a minimal reliability check ensuring consistency under output-preserving transformations. Alongside noise robustness, input invariance forms a complementary axis of explanation reliability. The work

of Dombrowski et al. [15] demonstrated dangers of adversarial scenarios targeted on XAI explanations, and resulted in a promising yet simple idea of aggregating attribution maps into ensemble explanations, which resulted in better robustness toward adversarial examples. The idea of ensemble explanations is presented in Chapter 6, and the following experiment analyzes the behavior of selected ensemble approaches.

8.5 Experiment 4: Enhancing Explanations via Ensembles

Previous experiments (Section 8.4) mentioned that simple ensemble strategies can enhance the reliability of explanation methods. In particular, the AGG-Mean [53], a straightforward averaging of attribution maps from different methods, exhibited increased robustness to adversarial perturbations compared to individual XAI techniques. These findings suggest that ensemble approaches, even simple ones, can mitigate some of the key weaknesses of single-explanation methods, including sensitivity to noise and vulnerability to manipulation.

This experiment compares individual XAI methods with *ensemble* approaches, along with a proposed ensemble extension (introduced in Chapter 6).

8.5.1 Setup

- **Dataset:** Imagenet ILSVRC 2012 (used for ResNet50 pretraining).
- **Model:** Pretrained ResNet50 (from torchvision).
- **Singular XAI Methods:**
 - Vanilla Gradients (Saliency)
 - Integrated Gradients (IG, baseline 0)
 - GradientSHAP (GS, baseline blur*0.4)
 - Guided Grad-CAM (LAYER4)
- **Ensemble Methods:**
 - NormEnsembleXAI [27], with *mean* aggregation function (NE-Mean)
 - AutoWeighted [8] (AW, weights $w_k = [0.5, 0.2, 0.2, 0.1, 0]$)
 - **Adaptive-Metric Softmax** (AMS, $\tau = 1$, λ same values as w_k above)
- **Quantitative Metrics:** The choice of metrics follows the recommendations from MetaQuantus [24], and their calculation is done via Quantus toolkit [23]. Arrows indicate metric interpretation direction ($\uparrow$: higher scores better; $\downarrow$: lower scores better):
 - *Robustness* (Local Lipschitz Estimate $\downarrow$)
 - *Faithfulness* (Pixel Flipping $\downarrow$)
 - *Randomization* (Model Parameter Randomisation test $\downarrow$)
 - *Localization* (Pointing Game $\uparrow$)
 - *Complexity* (Sparseness $\uparrow$)

All attributions were normalized to $[-1, 1]$, and aggregated over $N = 100$ test images to ensure direct comparability across methods and consistent qualitative and quantitative assessments. See Section 7.2 for formal definitions of all metrics. These metrics were also used for AutoWeighted and AMS ensemble weighting (see Chapter 6).

8.5.2 Results

Qualitative Evaluation. Figure 8.19 displays explanation heatmaps from the selected XAI methods (individual, ensembles, and proposed extension) for a single ImageNet sample (class ‘tree frog’). All methods produce semantically plausible attributions, yet differ in sharpness, focus, and alignment with class-relevant regions.

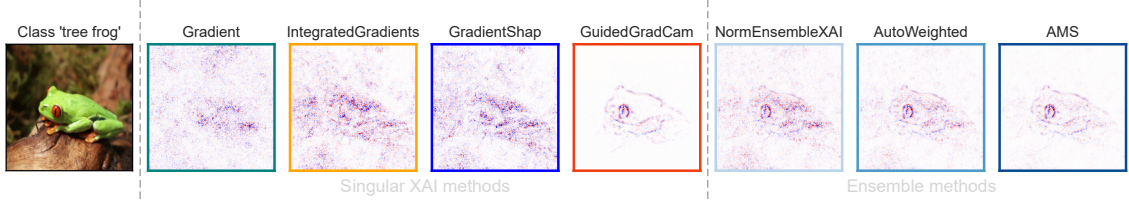


Figure 8.19: Qualitative comparison on ImageNet sample image (class ‘tree frog’) of singular XAI methods (Vanilla Gradients, Integrated Gradients, GradientSHAP, GuidedGradCAM) and ensembles (NormEnsembleXAI, AutoWeighted, and proposed AMS).

To see the singular XAI methods on multiple ImageNet samples more closely, refer to Appendix C.1. Detailed visualization of ensemble attribution maps over three selected ImageNet samples is depicted in Figure 8.20.

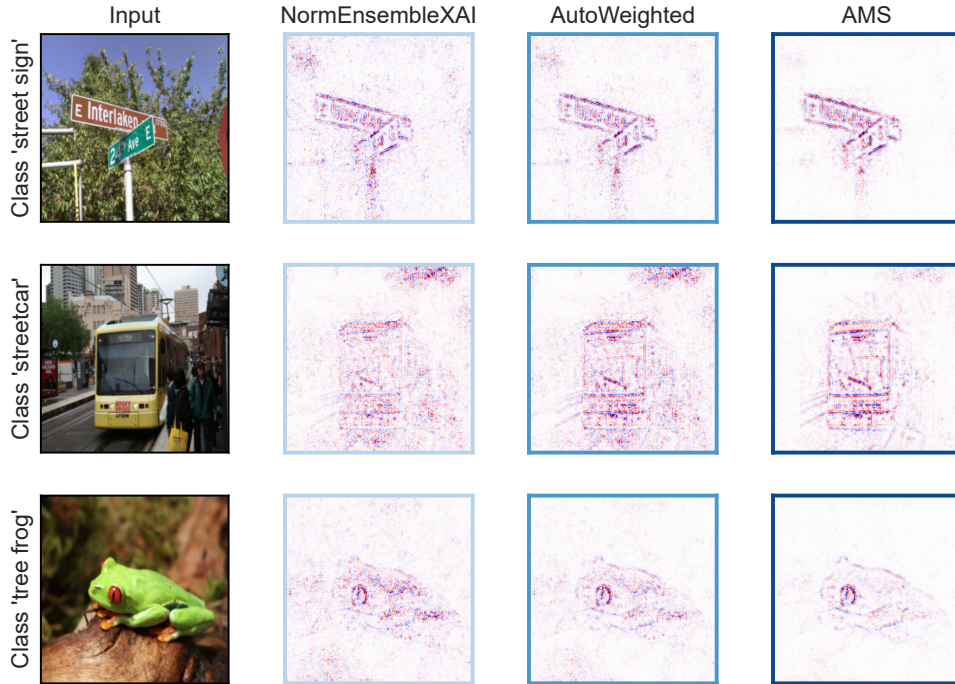


Figure 8.20: Qualitative comparison on three ImageNet sample images of ensemble methods (NormEnsembleXAI, AutoWeighted, and proposed AMS). AMS was able to exploit the localization property of GuidedGradCAM more effectively than the other ensembles.

Compared to the other two ensemble approaches (NormEnsembleXAI and AutoWeighted), the proposed AMS relied more on the positive properties of the Guided Grad-CAM method, which resulted in less noisy attribution maps. But from visual inspections, it might be

difficult to unambiguously prefer one method over another. Thus, qualitative evaluation is again complemented with a more systematic evaluation via quantitative metrics.

Quantitative Evaluation. Table 8.5 presents the quantitative results across five key evaluation metrics. The AMS ensemble method demonstrates competitive performance, achieving the best faithfulness score while maintaining strong results in other dimensions.

Table 8.5: Quantitative evaluation of interpretability methods across five metrics. Lower values are better for columns marked ($\downarrow$); higher values are better for columns marked ($\uparrow$).

Method	Robustness ($\downarrow$)	Complexity ($\uparrow$)	Faithfulness ($\downarrow$)	Localization ($\uparrow$)	Randomization ($\downarrow$)
<i>Singular methods</i>					
Gradient	0.097	0.559	0.101	0.372	0.486
IntegratedGradients	0.077	0.607	0.086	0.365	0.569
GradientShap	0.079	0.606	0.087	0.360	0.558
GuidedGradCam	0.016	0.841	0.042	0.644	0.754
<i>Ensemble methods</i>					
NormEnsembleXAI	0.066	0.588	0.066	0.389	0.554
AutoWeighted	0.038	0.599	0.059	0.411	0.617
AMS	0.076	0.645	0.041	0.488	0.524

Figure 8.21 visualizes a rank-based radar summary across all metrics, highlighting the relative strengths and trade-offs among the evaluated methods.

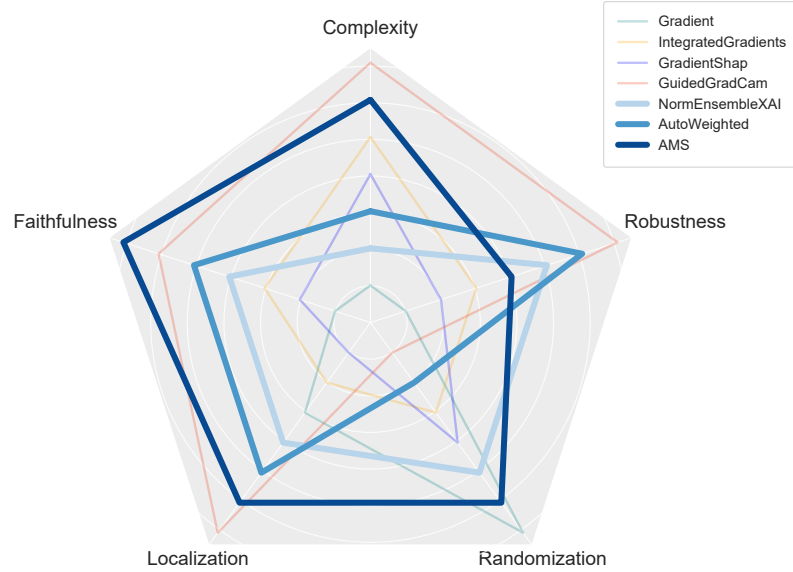


Figure 8.21: Quantitative comparison on $N = 100$ ImageNet samples of singular methods (Vanilla Gradients, Integrated Gradients, GradientSHAP, GuidedGradCAM) and ensembles (NormEnsembleXAI, AutoWeighted, and proposed AMS).

In the setting of this experiment, AMS ranked better than both other analyzed ensemble approaches, NormEnsembleXAI (mean) and AutoWeighted, across all axes except for Robustness. Overall, AMS maintained close to top performance, compared to both singular and ensemble methods.

8.5.3 Computational Complexity of Ensemble Explainers

Ensembling explanations is *inherently* more expensive than producing a single attribution: firstly, the E base maps must be generated, and then an aggregation step must be performed. Because the base maps are a fixed prerequisite for *every* ensemble, the fair point of comparison is the *aggregation stage only*. To contrast the three ensembles studied in this experiment (NormEnsembleXAI, AutoWeighted, and the proposed AMS), the four underlying singular attribution methods (Vanilla Gradients, Integrated Gradients, GradientSHAP, Guided GradCAM) will be considered.

Computational Complexity: Let E be the number of explainers (4), K the number of quantitative metrics, B the bootstrap rounds used by AMS for uncertainty, and $P = C \cdot H \cdot W$ the number of attribution pixels. Ignoring constant factors shared by all methods, the per-input-sample complexities are:

Table 8.6: Asymptotic aggregation cost (per input).

Ensemble	Metric stage	Fusion stage
NormEnsembleXAI	–	$\mathcal{O}(EP)$
AutoWeighted	$\mathcal{O}(EK)$ (dataset-level)	$\mathcal{O}(EP)$
AMS	$\mathcal{O}(BEK)$ (sample-level)	$\mathcal{O}(EP)$

AMS adds overhead only in the metric-computation stage, and this is only when the bootstrap parameter, which is optional, is $B > 1$. If $B = 1$, AMS’s complexity equals that of AutoWeighted. Its soft-max fusion stage has the same linear pixel cost as the other two ensemble methods.

Protocol.

- (i) ImageNet inputs (224×224); $N = 32$ representative samples.
- (ii) Pre-computed $E = 4$ base attribution maps loaded into GPU memory.
- (iii) Five Quantus [23] metrics ($K = 5$), same across all methods.
- (iv) Each aggregation stage is timed over 30 repetitions; wall-clock mean $\pm$ std reported.

Table 8.7: Average time (in seconds) of ensembling explanations (base maps pre-computed).

Ensemble	Metric stage [s]	Fusion stage [s]
NormEnsembleXAI	–	0.0012 ± 0.0011
AutoWeighted	28.372 ± 4.469	0.0004 ± 0.0005
AMS	28.746 ± 3.593	0.0004 ± 0.0004

NormEnsembleXAI requires only a global z -score and elementwise mean, completing fusion in ≈ 1.2 ms per batch with no metric overhead. AutoWeighted computes the full suite of five Quantus metrics once over the entire batch, which is 28.37 s of metric computation, after which its fusion step takes only 0.4 ms. Because the bootstrap $B = 1$, AMS’s metric-stage cost is nearly identical (28.75 s) to AutoWeighted’s, and its fusion via softmax remains

equally small at 0.4ms. In other words, the dominant cost for both AutoWeighted and AMS is in the high-fidelity metric evaluation, their aggregation stages are negligible by comparison.

Memory footprint. All three methods store one additional $E \times P$ float tensor and therefore share identical $\mathcal{O}(EP)$ memory complexity.

Optimization. AMS’s cost scales linearly with both K and B . Reducing B (e.g. from 16 to 4) or caching deterministic metrics eliminates most of the overhead while preserving adaptive weighting. A batched bootstrap could also be implemented in future work. Further ablation studies, e.g., varying K , B , metric combinations, and fusion strategies, could provide deeper insights into the trade-offs and performance characteristics of the proposed method.

8.5.4 Discussion

AMS adapts its weights per input sample and dynamically favors the most reliable explanation according to evaluated metrics. Unlike NormEnsembleXAI (mean) and AutoWeighted, which apply global or fixed weighting, AMS:

- Suppresses explainers that fail locally (e.g., under noise or perturbation).
- Prioritizes faithfulness and robustness when necessary, without sacrificing localization.
- Maintains convex aggregation, ensuring no artificial amplification beyond base methods.

Limitations.

- Requires at least two diverse attribution methods as input (as any XAI ensemble).
- Performance depends on the choice of evaluation metrics (λ); task-specific tuning may further optimize results.
- Introduces additional computational cost compared to simple aggregation schemes.

8.5.5 Summary

Ensembling in general is a promising approach to interpretability, aiming to exploit the positive aspects of singular methods. NormEnsemble stabilizes maps but cannot exploit per-image strengths. AutoWeighted introduces quality awareness yet remains global. The proposed *Adaptive-Metric Softmax* improves on both axes: provides instance adaptivity, metric-driven trade-offs, and label-free applicability, resulting in competitive results in faithfulness on ImageNet with modest overhead.

Chapter 9

Conclusion

This Master’s thesis addressed the growing demand for interpretability in machine learning by combining a structured theoretical review with empirical evaluation of explanation methods for complex “black-box” models.

It began with a detailed examination of core concepts and taxonomies. *Intrinsic* methods (e.g., linear models) were assessed for built-in transparency; *post-hoc* methods (e.g., SHAP) for providing insights into opaque models; and *mechanistic* interpretability for revealing internal structure via feature tracing and causal decomposition. Common use cases, challenges, and evaluation approaches (qualitative and quantitative) were also covered.

The experimental section systematically compared selected XAI methods (Vanilla Gradients, Integrated Gradients, SHAP variants, Grad-CAM variants) on datasets of increasing complexity (MNIST and ImageNet) using both custom and pre-trained CNNs. The analysis covered effects of model quality (underfit, overfit, and properly trained) on explanations, compared methods across varying data complexity, and tested robustness to noise, adversarial manipulations, and sensitivity input invariances.

Motivated by observed limitations in existing methods, the thesis proposed the Adaptive-Metric Softmax (AMS) Ensemble, a novel extension of ensemble techniques that dynamically weights multiple explanation methods based on per-sample performance across key metrics, without requiring ground-truth labels. AMS demonstrated improved performance across qualitative and quantitative criteria.

This work highlights persistent challenges in XAI, such as the trade-off between simplicity and fidelity, subjectivity in interpretation, and computational costs. It synthesizes theoretical insights across interpretability paradigms, delivers a thorough empirical evaluation under diverse settings, and proposes an adaptive ensemble technique to improve explanation properties. All code and documentation are publicly available on GitHub¹.

Results confirm that no single method suffices universally, but principled selection, evaluation, and combination of explanation strategies can yield more robust, faithful, and useful explanations. Future work may extend these findings to other data modalities, model architectures (e.g., transformer-based), and human-centered evaluations.

Moreover, this work was selected for presentation at the Excel@FIT 2025 student conference on innovation, technology, and science².

¹<https://github.com/simon-smida/DP-Explainable-AI>

²<https://excel.fit.vutbr.cz/2025/sbornik/>

Bibliography

- [1] ADEBAYO, J.; GILMER, J.; MUELLY, M.; GOODFELLOW, I.; HARDT, M. et al. *Sanity Checks for Saliency Maps*. 2020. Available at: <https://arxiv.org/abs/1810.03292>.
- [2] AHMAD, M. A.; TEREDESAI, A. and ECKERT, C. Interpretable Machine Learning in Healthcare. *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, 2018, p. 447–447. Available at: <https://api.semanticscholar.org/CorpusID:51721535>.
- [3] ALVAREZ MELIS, D. and JAAKKOLA, T. S. *On the Robustness of Interpretability Methods*. 2018. Available at: <https://arxiv.org/abs/1806.08049>.
- [4] ANDERS, C. J.; PASLIEV, P.; DOMBROWSKI, A.-K.; MÜLLER, K.-R. and KESSEL, P. *Fairwashing Explanations with Off-Manifold Detergent*. 2020. Available at: <https://arxiv.org/abs/2007.09969>.
- [5] ATAKISHIYEV, S.; SALAMEH, M.; YAO, H. and GOEBEL, R. Explainable Artificial Intelligence for Autonomous Driving: A Comprehensive Overview and Field Guide for Future Research Directions. *IEEE Access*, 2021, vol. 12, p. 101603–101625. Available at: <https://api.semanticscholar.org/CorpusID:245385604>.
- [6] BACH, S.; BINDER, A.; MONTAVON, G.; KLAUSCHEN, F.; MÜLLER, K.-R. et al. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE*, 2015, vol. 10. Available at: <https://api.semanticscholar.org/CorpusID:9327892>.
- [7] BAU, D.; ZHOU, B.; KHOSLA, A.; OLIVA, A. and TORRALBA, A. *Network Dissection: Quantifying Interpretability of Deep Visual Representations*. 2017. Available at: <https://arxiv.org/abs/1704.05796>.
- [8] BOBEK, S.; BAŁAGA, P. and NALEPA, G. J. Towards model-agnostic ensemble explanations. In: PASZYNSKI, M.; KRANZLMÜLLER, D.; KRZHIZHANOVSKAYA, V.; DONGARRA, J. J. and SLOOT, P. M., ed. *Computational Science – ICCS 2021*. Springer, Cham, 2021, vol. 12745, p. 39–51. Lecture Notes in Computer Science. Available at: https://doi.org/10.1007/978-3-030-77970-2_4.
- [9] BRICKEN, T.; TEMPLETON, A.; BATSON, J.; CHEN, B.; JERMYN, A. et al. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Transformer Circuits Thread*, 2023. Available at: <https://transformer-circuits.pub/2023/monosemantic-features>.
- [10] BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J. et al. *Language Models are Few-Shot Learners*. 2020. Available at: <https://arxiv.org/abs/2005.14165>.

- [11] BURRELL, J. How the machine ‘thinks’: Understanding opacity in machine learning algorithms. *Big Data & Society*, 2016, vol. 3, no. 1. Available at: <https://doi.org/10.1177/2053951715622512>.
- [12] CARUANA, R.; LOU, Y.; GEHRKE, J.; KOCH, P.; STURM, M. et al. Intelligible Models for Healthcare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In: *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. Sydney, NSW, Australia: Association for Computing Machinery, 2015, p. 1721–1730. Available at: <https://doi.org/10.1145/2783258.2788613>.
- [13] CHALASANI, P.; CHEN, J.; CHOWDHURY, A. R.; JHA, S. and WU, X. *Concise Explanations of Neural Networks using Adversarial Training*. 2020. Available at: <https://arxiv.org/abs/1810.06583>.
- [14] DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K. et al. ImageNet: A large-scale hierarchical image database. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, p. 248–255.
- [15] DOMBROWSKI, A.-K.; ALBER, M.; ANDERS, C. J.; ACKERMANN, M.; MÜLLER, K.-R. et al. *Explanations can be manipulated and geometry is to blame*. 2019. Available at: <https://arxiv.org/abs/1906.07983>.
- [16] DOSHI VELEZ, F. and KIM, B. *Towards A Rigorous Science of Interpretable Machine Learning*. 2017. Available at: <https://arxiv.org/abs/1702.08608>.
- [17] ELHAGE, N.; HUME, T.; OLSSON, C.; SCHIEFER, N.; HENIGHAN, T. et al. Toy Models of Superposition. *Transformer Circuits Thread*, 2022. Available at: https://transformer-circuits.pub/2022/toy_model.
- [18] ELHAGE, N.; NANDA, N.; OLSSON, C.; HENIGHAN, T.; JOSEPH, N. et al. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, 2021. Available at: <https://transformer-circuits.pub/2021/framework/index.html>.
- [19] EUROPEAN PARLIAMENT AND COUNCIL OF THE EUROPEAN UNION. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. 2016. Available at: <https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng>. Accessed: 2025-01-19.
- [20] GAROUANI, M.; MOTHE, J.; BARHRHOUI, A. and ALIGON, J. Investigating the Duality of Interpretability and Explainability in Machine Learning. In: *2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, October 2024, p. 861–867. Available at: <http://dx.doi.org/10.1109/ICTAI62512.2024.00125>.
- [21] GILPIN, L. H.; BAU, D.; YUAN, B. Z.; BAJWA, A.; SPECTER, M. A. et al. Explaining Explanations: An Overview of Interpretability of Machine Learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, 2018, p. 80–89. Available at: <https://api.semanticscholar.org/CorpusID:59600034>.

- [22] GUNNING, D. and AHA, D. DARPA’s Explainable Artificial Intelligence (XAI) Program. *AI Magazine*, Jun. 2019, vol. 40, no. 2, p. 44–58. Available at: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2850>.
- [23] HEDSTRÖM, A.; WEBER, L.; KRAKOWCZYK, D.; BAREEVA, D.; MOTZKUS, F. et al. Quantus: An Explainable AI Toolkit for Responsible Evaluation of Neural Network Explanations and Beyond. *Journal of Machine Learning Research*, 2023, vol. 24, no. 34, p. 1–11. Available at: <http://jmlr.org/papers/v24/22-0142.html>.
- [24] HEDSTRÖM, A.; BOMMER, P.; WICKSTRØM, K. K.; SAMEK, W.; LAPUSCHKIN, S. et al. *The Meta-Evaluation Problem in Explainable AI: Identifying Reliable Estimators with MetaQuantus*. 2023. Available at: <https://arxiv.org/abs/2302.07265>.
- [25] HOFFMANN, J.; BORGEAUD, S.; MENSCH, A.; BUCHATSKAYA, E.; CAI, T. et al. *Training Compute-Optimal Large Language Models*. 2022. Available at: <https://arxiv.org/abs/2203.15556>.
- [26] HRYNIEWSKA GUZIK, W.; LONGO, L. and BIECEK, P. *CNN-based explanation ensembling for dataset, representation and explanations evaluation*. 2024. Available at: <https://arxiv.org/abs/2404.10387>.
- [27] HRYNIEWSKA GUZIK, W.; SAWICKI, B. and BIECEK, P. *NormEnsembleXAI: Unveiling the Strengths and Weaknesses of XAI Ensemble Techniques*. 2024. Available at: <https://arxiv.org/abs/2401.17200>.
- [28] HUBEL, D. H. and WIESEL, T. N. Receptive fields of single neurones in the cat’s striate cortex. *The Journal of Physiology*, october 1959, vol. 148, no. 3, p. 574–591.
- [29] INC., K. *XAITK: Explainable AI Toolkit*. 2021. Available at: <https://xaitk.org/>. Accessed: 2025-01-22.
- [30] JACOVI, A. and GOLDBERG, Y. *Towards Faithfully Interpretable NLP Systems: How should we define and evaluate faithfulness?* 2020. Available at: <https://arxiv.org/abs/2004.03685>.
- [31] KINDERMANS, P.-J.; HOOKER, S.; ADEBAYO, J.; ALBER, M.; SCHÜTT, K. T. et al. *The (Un)reliability of saliency methods*. 2017. Available at: <https://arxiv.org/abs/1711.00867>.
- [32] LAKKARAJU, H. and BASTANI, O. “How do I fool you? ,: Manipulating User Trust via Misleading Black Box Explanations. 2019. Available at: <https://arxiv.org/abs/1911.06473>.
- [33] LECUN, Y.; BENGIO, Y. and HINTON, G. E. Deep learning. *Nature*. Nature Publishing Group, May 2015, vol. 521, no. 7553, p. 436–444.
- [34] LECUN, Y.; CORTES, C. and BURGESS, C. MNIST handwritten digit database. *ATT Labs [Online]*, 2010, vol. 2. Available at: <http://yann.lecun.com/exdb/mnist>.
- [35] LIPTON, Z. C. The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability Is Both Important and Slippery. *Queue*, june 2018, vol. 16, no. 3, p. 31–57. ISSN 1542-7730.

- [36] LONGO, L.; BRCIC, M.; CABITZA, F.; CHOI, J.; CONFALONIERI, R. et al. Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*. Elsevier BV, june 2024, vol. 106, p. 102301. ISSN 1566-2535. Available at: <http://dx.doi.org/10.1016/j.inffus.2024.102301>.
- [37] LUNDBERG, S. and LEE, S.-I. *A Unified Approach to Interpreting Model Predictions*. 2017. Available at: <https://arxiv.org/abs/1705.07874>.
- [38] LUNDBERG, S. M.; ERION, G. G. and LEE, S.-I. *Consistent Individualized Feature Attribution for Tree Ensembles*. 2019. Available at: <https://arxiv.org/abs/1802.03888>.
- [39] MCCARTHY, J.; MINSKY, M. L.; ROCHESTER, N. and SHANNON, C. E. A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, August 31, 1955. *AI Magazine*, Dec. 2006, vol. 27, no. 4, p. 12. Available at: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1904>.
- [40] MEHRABI, N.; MORSTATTER, F.; SAXENA, N.; LERMAN, K. and GALSTYAN, A. *A Survey on Bias and Fairness in Machine Learning*. 2022. Available at: <https://arxiv.org/abs/1908.09635>.
- [41] MILLER, T. *Explanation in Artificial Intelligence: Insights from the Social Sciences*. 2018. Available at: <https://arxiv.org/abs/1706.07269>.
- [42] MOLNAR, C. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. 2nd ed. 2022. Available at: <https://christophm.github.io/interpretable-ml-book>.
- [43] MOLNAR, C.; KÖNIG, G.; HERBINGER, J.; FREIESLEBEN, T.; DANDL, S. et al. *General Pitfalls of Model-Agnostic Interpretation Methods for Machine Learning Models*. 2021. Available at: <https://arxiv.org/abs/2007.04131>.
- [44] MONTGOMERY, D. C.; PECK, E. A. and VINING, G. G. *Introduction to Linear Regression Analysis*. 6th ed. Hoboken, NJ: John Wiley & Sons, 2021. ISBN 978-1-119-57872-7.
- [45] OLAH, C. *Neural Networks, Manifolds, and Topology*. 2014. Available at: <https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/>. Accessed: 2025-03-12.
- [46] OLAH, C. *Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases*. 2022. Available at: <https://www.transformer-circuits.pub/2022/mech-interp-essay>.
- [47] OLAH, C.; CAMMARATA, N.; SCHUBERT, L.; GOH, G.; PETROV, M. et al. Zoom In: An Introduction to Circuits. *Distill*, 2020. Available at: <https://distill.pub/2020/circuits/zoom-in>.
- [48] OLAH, C.; MORDVINTSEV, A. and SCHUBERT, L. Feature Visualization. *Distill*, 2017. Available at: <https://distill.pub/2017/feature-visualization>.

- [49] OLAH, C.; SATYANARAYAN, A.; JOHNSON, I.; CARTER, S.; SCHUBERT, L. et al. The Building Blocks of Interpretability. *Distill*, 2018. Available at: <https://distill.pub/2018/building-blocks>.
- [50] PRUTHI, D.; GUPTA, M.; DHINGRA, B.; NEUBIG, G. and LIPTON, Z. C. *Learning to Deceive with Attention-Based Explanations*. 2020. Available at: <https://arxiv.org/abs/1909.07913>.
- [51] RAY, A.; YAO, Y.; KUMAR, R.; DIVAKARAN, A. and BURACHAS, G. *Can You Explain That? Lucid Explanations Help Human-AI Collaborative Image Retrieval*. 2019. Available at: <https://arxiv.org/abs/1904.03285>.
- [52] RIBEIRO, M. T.; SINGH, S. and GUESTRIN, C. “Why Should I Trust You? „; *Explaining the Predictions of Any Classifier*. 2016. Available at: <https://arxiv.org/abs/1602.04938>.
- [53] RIEGER, L. and HANSEN, L. K. *A simple defense against adversarial attacks on heatmap explanations*. 2020. Available at: <https://arxiv.org/abs/2007.06381>.
- [54] RUDIN, C. *Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead*. 2019. Available at: <https://arxiv.org/abs/1811.10154>.
- [55] SAMEK, W.; BINDER, A.; MONTAVON, G.; BACH, S. and MÜLLER, K.-R. *Evaluating the visualization of what a Deep Neural Network has learned*. 2015. Available at: <https://arxiv.org/abs/1509.06321>.
- [56] SARKAR, A. Is Explainable AI a Race Against Model Complexity? 192-199. *ArXiv*, 2022, abs/2205.10119. Available at: <https://api.semanticscholar.org/CorpusID:248302066>.
- [57] SELVARAJU, R. R.; DAS, A.; VEDANTAM, R.; COGSWELL, M.; PARIKH, D. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 2016, vol. 128, p. 336 – 359. Available at: <https://api.semanticscholar.org/CorpusID:15019293>.
- [58] SIMONYAN, K.; VEDALDI, A. and ZISSERMAN, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR*, 2013, abs/1312.6034. Available at: <https://api.semanticscholar.org/CorpusID:1450294>.
- [59] SIMONYAN, K. and ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv preprint arXiv:1409.1556*, 2014. Available at: <https://arxiv.org/abs/1409.1556>.
- [60] SIXT, L.; GRANZ, M. and LANDGRAF, T. *When Explanations Lie: Why Many Modified BP Attributions Fail*. 2024. Available at: <https://arxiv.org/abs/1912.09818>.
- [61] SLACK, D.; HILGARD, S.; JIA, E.; SINGH, S. and LAKKARAJU, H. *Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods*. 2020. Available at: <https://arxiv.org/abs/1911.02508>.

- [62] SOKOL, K. and FLACH, P. *Explainability Is in the Mind of the Beholder: Establishing the Foundations of Explainable Artificial Intelligence*. 2022. Available at: <https://arxiv.org/abs/2112.14466>.
- [63] SOKOL, K. and FLACH, P. *Explainability Is in the Mind of the Beholder: Establishing the Foundations of Explainable Artificial Intelligence*. 2022. Available at: <https://arxiv.org/abs/2112.14466>.
- [64] SUNDARARAJAN, M.; TALY, A. and YAN, Q. Axiomatic Attribution for Deep Networks. In: *International Conference on Machine Learning*. 2017. Available at: <https://api.semanticscholar.org/CorpusID:16747630>.
- [65] SUTTON, R. *The Bitter Lesson*. 2019. Available at: <http://incompleteideas.net/IncIdeas/BitterLesson.html>. Accessed: 2025-01-22.
- [66] SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. et al. Going Deeper with Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, p. 1–9.
- [67] VISWAN, V.; SHAFFI, N.; MAHMUD, M.; SUBRAMANIAN, K. and HAJAMOHIDEEN, F. Explainable Artificial Intelligence in Alzheimer’s Disease Classification: A Systematic Review. *Cognitive Computation*, november 2023, vol. 16.
- [68] VOSS, C.; CAMMARATA, N.; GOH, G.; PETROV, M.; SCHUBERT, L. et al. Visualizing Weights. *Distill*, 2021. Available at: <https://distill.pub/2020/circuits/visualizing-weights>.
- [69] WEBER, P. M.; CARL, K. V. and HINZ, O. Applications of Explainable Artificial Intelligence in Finance – a systematic review of Finance, Information Systems, and Computer Science literature. *Management Review Quarterly*, 2023, vol. 74, p. 867–907. Available at: <https://api.semanticscholar.org/CorpusID:257324912>.
- [70] WICKSTRØM, K.; HÖHNE, M. M.-C. and HEDSTRÖM, A. *From Flexibility to Manipulation: The Slippery Slope of XAI Evaluation*. 2024. Available at: <https://arxiv.org/abs/2412.05592>.
- [71] WOLFRAM, S. *What’s Really Going On in Machine Learning? Some Minimal Models* Stephen Wolfram Writings. 2024. Available at: <https://writings.stephenwolfram.com/2024/08/whats-really-going-on-in-machine-learning-some-minimal-models/>. Accessed: 2025-03-12.
- [72] ZEILER, M. D. and FERGUS, R. *Visualizing and Understanding Convolutional Networks*. 2013. Available at: <https://arxiv.org/abs/1311.2901>.
- [73] ZHANG, J.; LIN, Z.; BRANDT, J.; SHEN, X. and SCLAROFF, S. *Top-down Neural Attention by Excitation Backprop*. 2016. Available at: <https://arxiv.org/abs/1608.00507>.
- [74] ZOU, L.; GOH, H. L.; LIEW, C. J. Y.; QUAH, J. L.; GU, G. T. et al. Ensemble Image Explainable AI (XAI) Algorithm for Severe Community-Acquired Pneumonia and COVID-19 Respiratory Infections. *IEEE Transactions on Artificial Intelligence*, 2023, vol. 4, no. 2, p. 242–254.

Appendix A

CNN Architecture (MNIST)

The shared CNN architecture of the models:

```
class CNNModel(nn.Module):
    """Two-conv CNN for 28x28 grayscale digits (MNIST)."""
    def __init__(self, dropout1=0.25, dropout2=0.5):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 32, 3) # Kaiming-uniform init
        self.conv2 = nn.Conv2d(32, 64, 3) # Kaiming-uniform init
        self.pool = nn.MaxPool2d(2)
        self.do1 = nn.Dropout2d(dropout1)
        self.do2 = nn.Dropout(dropout2)
        self.fc1 = nn.Linear(64*12*12, 128) # Kaiming-uniform init
        self.fc2 = nn.Linear(128, 10) # Kaiming-uniform init
    def forward(self, x):
        x = F.relu(self.conv1(x)) # ReLU
        x = F.relu(self.conv2(x)) # ReLU
        x = self.pool(x)
        x = self.do1(x)
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x)) # ReLU
        x = self.do2(x)
        return self.fc2(x) # logits
```

Listing A.1: Shared CNN architecture used by all three models (underfit, overfit, proper). Despite identical structures (and a total of 1 199 882 trainable parameters), each model’s training configuration differs, see [Table A.1](#).

Weight initialization. All layers use **Kaiming uniform** (He) initialization with `fan_in` mode and `a=0` (ReLU assumption), as implemented in `torch.nn.init.kaiming_uniform_`. Biases are initialized to zero.

Activation functions. Every hidden layer employs the `torch.nn.functional.relu`. No other non-linearities or normalization layers are present.

Global random seed. `torch.manual_seed(42)` and `numpy.random.seed(42)` before any RNG access, to guarantee identical weight and data shuffles across runs.

A.1 Dataset and Pre-processing

Dataset MNIST [34], 60 000 train / 10 000 test images.

Transform `ToTensor()` $\rightarrow$ `Normalize( $\mu = 0.1307$ ,  $\sigma = 0.3081$ )`.

A.2 Training Configurations

All three regimes (underfit, overfit, proper) in Experiment 1 share the architecture above but vary *training data*, *epochs*, and *dropout*. Table A.1 makes the differences explicit.

Table A.1: Training hyper-parameters per regime.

Regime	Train set	Epochs	Dropout (p_1, p_2)	Optimizer	LR	Batch
Underfit	200 subset	1	(0.25, 0.50)	Adam	10^{-3}	128
Overfit	200 subset	20	(0, 0)	Adam	10^{-3}	128
Proper	Full 60k	8	(0.25, 0.25)	Adam	10^{-3}	128

All code, random seeds, related analyses, and figure-generation scripts are provided in the companion GitHub¹ repository.

¹<https://github.com/simon-smida/DP-Explainable-AI>

Appendix B

Input Invariance Test Statistics

Table B.1 reports mean, standard deviation, and 95th percentile for all five drift metrics over the same 100 images used in Experiment 3b (Subsection 8.4.2).

Table B.1: Input-invariance under a uniform shift of $-1/\sigma$. Values are *mean* $\pm$ *std* over 100 images; P_{95} is the 95-th percentile. Bold rows meet the invariance criterion.

Method	$\overline{\max \Delta } (\pm\sigma)$	$P_{95}[\max \Delta]$	$\bar{\rho} (\pm\sigma)$
Gradient	$4.6 \times 10^{-4} \pm 4.5 \times 10^{-3}$	0	$1.0000 \pm 9.0 \times 10^{-5}$
IG-zero	$8.1 \times 10^{-1} \pm 2.5 \times 10^{-1}$	1.28	$-3.0 \times 10^{-2} \pm 4.7 \times 10^{-2}$
IG-black	$3.6 \times 10^{-5} \pm 1.9 \times 10^{-4}$	1.3×10^{-4}	$1.0000 \pm 1.6 \times 10^{-7}$
GS-zero	$8.0 \times 10^{-1} \pm 2.5 \times 10^{-1}$	1.28	$-2.6 \times 10^{-2} \pm 4.9 \times 10^{-2}$
GS-black	$2.8 \times 10^{-2} \pm 1.2 \times 10^{-2}$	4.8×10^{-2}	$1.0000 \pm 2.8 \times 10^{-4}$

Table B.2 shows the corresponding mean, standard deviation, and 95th percentile for all five drift metrics over the same 100 images used in Subsection 8.4.2, for the checkerboard perturbation.

Table B.2: Invariance statistics for a 2×2 checkerboard perturbation (raw amplitude 0.30).

Method	$\overline{\max \Delta } (\pm\sigma)$	$P_{95}[\max \Delta]$	$\bar{\rho} (\pm\sigma)$
Gradient	$2.27 \times 10^{-1} \pm 5.43 \times 10^{-2}$	3.24×10^{-1}	$7.96 \times 10^{-1} \pm 5.06 \times 10^{-2}$
IG-zero	$2.73 \times 10^{-1} \pm 7.62 \times 10^{-2}$	3.87×10^{-1}	$8.81 \times 10^{-1} \pm 3.51 \times 10^{-2}$
IG-black	$3.02 \times 10^{-1} \pm 8.46 \times 10^{-2}$	4.28×10^{-1}	$9.08 \times 10^{-1} \pm 2.57 \times 10^{-2}$
GS-zero	$2.62 \times 10^{-1} \pm 7.41 \times 10^{-2}$	3.81×10^{-1}	$8.88 \times 10^{-1} \pm 3.26 \times 10^{-2}$
GS-black	$2.89 \times 10^{-1} \pm 8.18 \times 10^{-2}$	4.27×10^{-1}	$9.13 \times 10^{-1} \pm 2.39 \times 10^{-2}$

Appendix C

Experiment 4: Visualizations

Detailed visualizations of the singular XAI methods on three ImageNet samples (Figure C.1) and the combined (singular + ensembles) version on four ImageNet samples (Figure C.2).

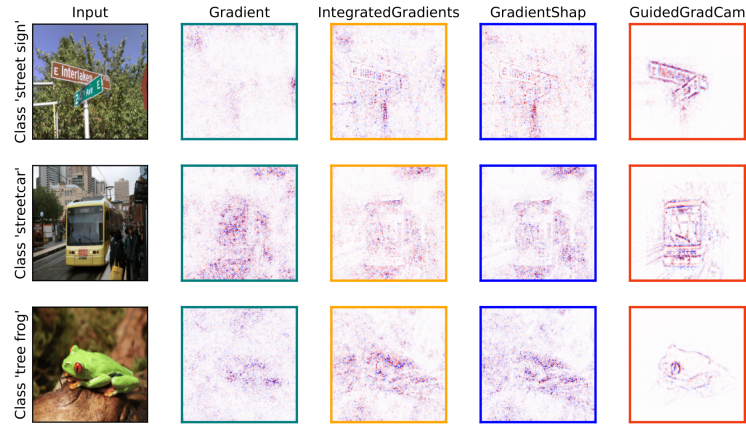


Figure C.1: Qualitative comparison of singular XAI methods on three ImageNet samples.

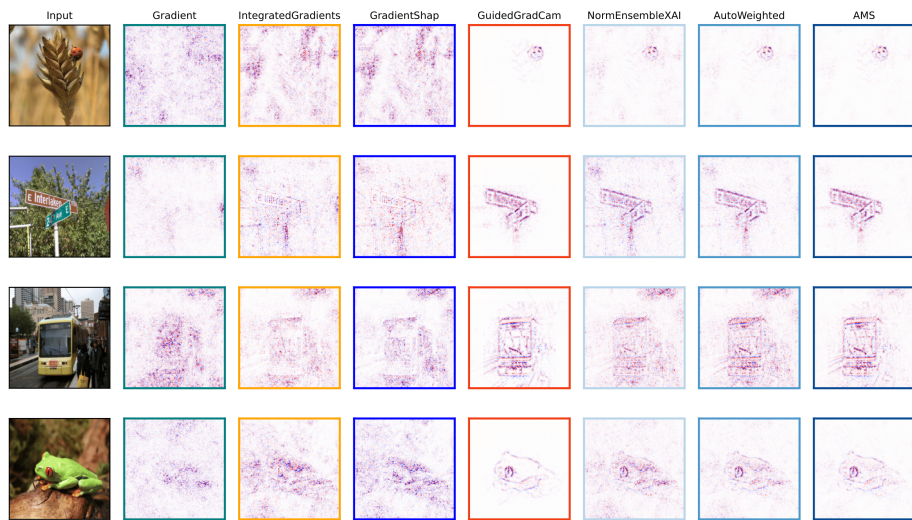


Figure C.2: Comparison of both singular and ensemble methods on four ImageNet samples.