Article

Not peer-reviewed version

Deep Learning Image Classification with Explainability Using SHAP: A Case Study with ResNet-50 and CIFAR-10

Relebohile George Qhobosheane [*]

Posted Date: 4 August 2025

doi: 10.20944/preprints202508.0208.v1

Keywords: Convolutional Neural Network; ImageNet; SHapley Additive exPlanations; ResNet-50

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Deep Learning Image Classification with Explainability Using SHAP: A Case Study with ResNet-50 and CIFAR-10

George Qhobosheane

Harrisburg University of Technology; qrelebohile@gmail.com

Abstract

This paper thoroughly looks at how to combine cutting-edge explainability techniques with deep learning for image classification. We optimize a Deep Convolutional Neural Network (CNN) for the CIFAR-10 benchmark image classification task. More specifically, we use a ResNet-50 architecture that has already been trained on the ImageNet dataset. The experimental pipeline's strict methodology includes a thorough model training protocol, data preprocessing with augmentation, and a performance evaluation that looks at many different factors. The model's high accuracy in classifying shows how well transfer learning works for this job. This work is important because it goes beyond just performance metrics by using a game-theoretically based method called SHAP (SHapley Additive exPlanations) to explain the model's predictions. SHAP gives you clear, easy-to-understand visualizations that show how each pixel affects classification choices by making pixel-level attributions. The results show that SHAP can figure out why both correct and incorrect predictions were made, and that the model learns features that are important for meaning. The important "black-box" problem in deep learning is solved by combining high performance with deep interpretability. This shows a way to make AI systems that are more transparent, reliable, and trustworthy, so they can be used in real-world situations where the stakes are high.

Keywords: Convolutional Neural Network; ImageNet; SHapley Additive exPlanations; ResNet-50

Introduction

The Rise of Deep Learning in Computer Vision

The rise of deep learning has led to a revolution in computer vision over the past ten years. CNNs are a type of deep neural network that has become the standard for many visual recognition tasks. They are based on the hierarchical organization of the animal visual cortex, where neurons respond to stimuli in limited receptive fields. Their main strength is their ability to automatically and adaptively learn a hierarchy of spatial features directly from raw pixel data, which eliminates the need for manual feature engineering that was common in traditional computer vision methods.

This ability to extract hierarchical features, which range from simple, low-level features like edges and textures in the first layers to complex, high-level object parts and concepts in deeper layers, has fundamentally changed what is possible in artificial perception. This has led to state-of-the-art performance in basic computer vision tasks like image classification, object detection, and semantic segmentation.

The Need for Explainable AI (XAI) and the "Black Box" Dilemma

The very feature that makes deep learning models so effective—their capacity to learn complex patterns across millions of parameters in deep, multi-layered architectures—also makes them intrinsically opaque. These models frequently function as "black boxes," in which the internal logic and the precise reasoning behind a particular prediction are not immediately apparent to human

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

users or even their developers. This lack of transparency creates a significant trust deficit, which poses a formidable obstacle to the adoption and deployment of AI in high-stakes, safety-critical domains like medical diagnostics, autonomous vehicle navigation, financial decision-making, and security systems. In these domains, a biased or inaccurate prediction can have serious repercussions, and it is crucial to comprehend, validate, and debug a model's decision-making process.

Explainable AI (XAI) is a very important area of research that has come about because of this problem. XAI wants to make it possible for people to understand the decisions of complex models by turning opaque predictions into clear and understandable ones. The need for explainability is not just academic; it is also driven by a growing societal and regulatory need for fairness, accountability, and transparency in AI systems. In the current state of artificial intelligence, interpretability is not a luxury but rather a basic requirement for creating sturdy, moral, and reliable systems.

Overview of the Study

This study addresses the dual challenges of ensuring model transparency and achieving superior predictive performance. It provides a systematic and comprehensive approach for developing a resilient and comprehensible image classification system. To accomplish this, the study employs a synergistic integration of innovative methodologies and established norms. The CIFAR-10 dataset, a well-known computer vision benchmark, is used as the experimental testbed. The classification model is based on the ResNet-50 architecture, a deep and powerful CNN known for its performance. Transfer learning is used to efficiently achieve high accuracy by fine-tuning a model that has already been trained on the extensive ImageNet dataset.

This study utilizes SHAP (SHapley Additive exPlanations), a robust and theoretically grounded XAI framework, to confront the opaque nature of the ResNet-50 model. SHAP delivers pixel-level elucidations for each prediction, offering a thorough and comprehensible insight into the model's decision-making process. This paper demonstrates a complete pipeline from data preparation to high-performance, interpretable classification through the integration of these components.

Questions for Research, Hypotheses, and Contributions Statement

This study aims to make a practical contribution to the field of trustworthy AI by demonstrating the seamless integration of a robust XAI framework with a high-performance deep learning model. The subsequent formal hypotheses and research inquiries function as the guiding framework for the investigation.

Hypotheses

1. A ResNet-50 model that has already been trained on ImageNet can get more than 80% of the CIFAR-10 test set right with a little bit of fine-tuning.
2. The SHAP framework can produce pixel-level attributions that are locally accurate and class-discriminative for the ResNet-50 model's predictions. This will help people understand important features of objects.
3. By using XAI techniques, especially SHAP, to look at and understand the model's failure modes, you can get ideas that can help you make the model better in the future.

Questions for Research

1. How well can a deep architecture that has been pre-trained on a big, general-domain dataset (ImageNet) use the feature representations it learned on a smaller, lower-resolution, and more focused dataset (CIFAR-10)?
2. What specific visual features and patterns does the improved ResNet-50 model associate with each of the ten classes in the CIFAR-10 dataset, according to SHAP explanations?

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

3. Can SHAP explanations show systematic failure modes in the model, such as relying too much on false correlations and background artifacts or getting confused between classes that look similar (like cats and dogs)?

Foundations and Related Research

This work employs an approach grounded in decades of research in computer vision and machine learning, rather than being an isolated experiment. The development of this study adheres to a coherent and rational narrative: the relentless pursuit of predictive performance led to the formulation of increasingly complex and advanced neural network architectures. But because these models were so successful, their built-in lack of transparency became a new and urgent problem. This "black box" problem can be solved directly and easily with explainable AI. This study combines a cutting-edge deep learning model with a cutting-edge XAI framework to meet the dual goals of performance and transparency. This puts it at the cutting edge of this evolutionary path.

The Growth of Convolutional Neural Networks' Architecture

ResNet-50 was chosen as the base model for this study because CNNs have a long history of coming up with new ways to build things. This history is marked by a constant push for deeper networks because depth has been shown to be important for learning rich and discriminative feature hierarchies.

LeNet: The Beginning of Modern CNNs

The LeNet family of networks, created by Yann LeCun and his team in the late 1980s and early 1990s, is where the basic ideas for modern CNNs came from. LeNet-5, the most famous version, was the first to use the three main parts of CNNs today: fully-connected layers for classification, pooling (or subsampling) layers for spatial downsampling and invariance, and learnable convolutional layers for feature extraction. LeNet was one of the first systems to be trained end-to-end using the backpropagation algorithm, straight from pixel images with little preprocessing. It was made to recognize handwritten digits. Its successful commercial use in check-reading systems was one of the first and most convincing proofs that neural networks could be used for real-world pattern recognition tasks. It also laid the theoretical groundwork for all later CNN architectures.

AlexNet: The New Ideas and Breakthroughs in Deep Learning

Interest in neural networks waned for over a decade following LeNet; however, in 2012, Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created AlexNet. AlexNet achieved victory in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012, exhibiting an unprecedented error rate margin exceeding 10%, which is widely regarded as the pivotal moment that initiated the contemporary deep learning revolution. To effectively train a significantly deeper network (8 layers) on an extensive dataset, AlexNet employed a combination of architectural and training strategies, rather than relying on a singular innovation.

In Rectified Linear Units (ReLU), the non-saturating activation function $f(x)=\max(0,x)$ is used instead of more common sigmoid or tanh functions. ReLUs make training converge much faster and stop the gradient from disappearing. Using a new regularization method called dropout, some of the outputs of neurons are randomly set to zero during training. This stops neurons from making complex co-adaptations, which helps the network learn more robust features and stops overfitting. Using Graphics Processing Units (GPUs) to run the computationally demanding convolution operations in parallel, it is possible to train a large network with 60 million parameters on the 1.2 million images in the ImageNet dataset in a reasonable amount of time.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

VGGNet: The Search for Depth through Simplicity

After AlexNet's success, researchers at the University of Oxford's Visual Geometry Group (VGG) looked into how network depth affects performance in their VGGNet architecture.30 VGGNet's main contribution, which led to its top results in the ILSVRC 2014, was its proof that significant performance gains could be obtained by methodically increasing network depth using a straightforward and homogeneous architecture.

The VGG architecture only uses very small 3x3 convolutional filters stacked on top of each other. This design choice has two main benefits. A stack of several 3x3 filters has more non-linear ReLU layers, which makes the decision function more discriminative. It also has a bigger effective receptive field (for example, two stacked 3x3 layers have a 5x5 receptive field). Second, it has fewer parameters overall than a single layer with a bigger filter. The most popular configurations, VGG-16 and VGG-19, had 16 and 19 weight layers, respectively, showing that very deep networks were necessary for high performance.

Fixing the Degradation Problem with Deep Residual Networks (ResNet)

Adding more layers to a network made the degradation problem, which is a new and strange challenge, even though VGGNet showed the benefits of depth. Researchers discovered that accuracy would reach a saturation point and subsequently decline rapidly when network depth exceeded a specific threshold. The deeper models had a higher training error than the shallower ones, which went against the idea that this was because of overfitting. This meant that deeper "plain" networks were harder to optimize by nature.

Kaiming He et al. (2015) developed Deep Residual Networks (ResNet) to solve this problem. ResNet's main innovation is the residual block, which has a "shortcut" or "skip" connection that skips one or more layers. Instead of having the layers learn a direct mapping from input x to output H(x), the layers are reformulated to learn a residual function F(x)=H(x)−x. The block's final output is then calculated as F(x)+x. The hypothesis, which has been empirically validated, posits that the network can optimize the residual mapping more efficiently than the original, unreferenced mapping. ResNet won first place in the ILSVRC 2015 classification task because it could handle very deep data. In the extreme case where an identity mapping is best (i.e., H(x)=x), it is much easier for the network to learn to drive the weights of the stacked layers toward zero (making F(x) zero) than to fit an identity function with a stack of non-linear layers. This change to the architecture solved the degradation problem and let networks with 152 layers and more be trained.

Table 1. A summary of important CNN architectures.

Architecture	Year	Depth (Weight Layers)	Key Filter Size(s)	Activation Function	Key Innovation(s)	ILSVRC Performance
LeNet-5	1998	7	5x5	Tanh/Sigmoid	Foundational CNN structure (Conv, Pool, FC)	N/A (Pre-ILSVRC)
AlexNet	2012	8	11x11, 5x5, 3x3	ReLU	ReLU, Dropout, GPU training, greater depth	15.3% top-5 error (Winner 2012)

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

VGG-16	2014	16	3x3	ReLU	Homogeneous stacking of small filters to increase depth	7.3% top-5 error (2nd Place 2014)
ResNet-34	2015	34	3x3	ReLU	Residual blocks with shortcut connections	3.57% top-5 error (Winner 2015)

Transfer Learning in Computer Vision

To train a deep CNN like ResNet-50 from scratch, you need a lot of labeled data and a lot of processing power. Transfer learning is a strong and useful alternative when the target dataset is small, like CIFAR-10. This study's methodology is fundamentally based on transfer learning principles.

Formal Definitions and Scenarios

Transfer learning aims to improve the learning of a target predictive function fT(·) within a target domain DT by leveraging information from a source task TS and a source domain DS. A domain D is characterized by a marginal probability distribution P(X) and a feature space X, where X represents a sample of data. Transfer learning is useful when the source and target domains or tasks are not the same. A task T is characterized by a label space Y and a predictive function f(·) derived from the data. Transfer learning can be classified into distinct scenarios based on the nature of the tasks and the accessibility of labeled data.

Inductive Transfer Learning: The source and target tasks are different, but the target domain has some labeled data that can be used to help create the target model. This study exemplifies this category as it employs labeled CIFAR-10 data to modify a model originally trained for the 1000-class ImageNet classification task for the 10-class CIFAR-10 classification task.

Transductive Transfer Learning: The source and target tasks are the same, but the domains are different. The source domain typically possesses labeled data, whereas the target domain lacks it. This is a common occurrence in domain adaptation. In unsupervised transfer learning, the tasks are different from those in inductive transfer, but neither the source nor the target domains have any labeled data.

Basic Techniques: Fine-Tuning vs. Feature Extraction

There are two main ways to use pre-trained CNNs 18 for transfer learning:

Feature Extraction: The pre-trained model is used as a fixed feature extractor, and its final classification layers are usually not used. Only the weights of a new, task-specific classifier added on top are trained on the target dataset; the weights of the convolutional base are not changed. This method works quickly and well if the target dataset is small and similar to the source dataset.

Fine-tuning: This method involves unfreezing some of the upper layers of the convolutional base and changing and training the classifier head. Then, these layers are trained (or "fine-tuned") along with the new classifier using the target data, usually at a very low learning rate. This lets the model change its high-level features that it learned before to fit the target dataset's unique details. This study employs the fine-tuning technique to enhance the ResNet-50 model's adaptation of its feature representations to the CIFAR-10 dataset.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Reasons and Benefits of CIFAR-10 and Other Small Datasets

The hierarchical structure of the features that CNNs learn is what makes transfer learning work. Models trained on large and varied datasets like ImageNet learn a lot of visual features. The first layers learn basic features like edges, colors, and textures that are useful for most computer vision tasks. The later layers learn more complicated and abstract features, like shapes and parts of objects.

Using a pre-trained model gives our new model a big advantage. The model begins with a collection of highly effective, general-purpose visual features. This has several important benefits, especially for smaller target datasets like CIFAR-10, as opposed to starting the network with random weights and learning these features from scratch. Better Performance: The pre-learned features are a better place to start when optimizing than starting from scratch. This often leads to a higher final accuracy. Less Data Needed: The model needs less target-specific data to learn the new classification task because it can already find basic visual features. Faster Convergence: Training goes much faster because most of the model's weights are already set up correctly.

A Visual Taxonomy of Explainable AI

Choosing a XAI method is just as important as choosing the model architecture itself because different XAI methods give different kinds of explanations with different pros and cons. SHAP was selected for this study to underscore theoretical precision and comprehensive attribution. A review of the XAI landscape reveals a clear trade-off between the type of explanation, model-agnosticism, and speed.

Perturbation-Based Methods: LIME and the Local Approximation Principle

Perturbation-based methods systematically change the input to a model and then look at how the change affects the prediction. LIME (Local Interpretable Model-Agnostic Explanations) is a great example of this method. LIME's main job is to explain a single prediction by using a simple, easy-to-understand model (like a shallow decision tree or a sparse linear model) to approximate the complex, black-box model in the area around that prediction. It does this by making a new dataset of perturbed instances around the original input (like turning super-pixels on or off in an image), getting the predictions of the black-box model for these new instances, and then training the simple, easy-to-understand model on this new dataset, weighted by how close it is to the original instance. LIME's main advantage is that it works with any model without needing to see its internal structure. However, its explanations can be unstable, changing a lot with different perturbation settings, and the definition of a "local neighborhood" can be unclear.

Gradient-Based Techniques: Visualizing Class Activations and Grad-CAM

Gradient-based methods use the gradient information that flows through the network during backpropagation to figure out how important each feature is. Grad-CAM (Gradient-weighted Class Activation Mapping) is one of the most popular methods in this group. Grad-CAM makes a rough localization map, or heatmap, that shows the important parts of an image for a specific class prediction.

It does this by finding the gradient of the class score in relation to the feature maps of the last convolutional layer. Then, based on the global average of these gradients, each feature map is given a weight that shows how important it is. The main advantages of Grad-CAM are that it is fast and can be used with many different CNN architectures without any changes. However, it only gives a general explanation of where something is important (the "where") and not a detailed, fine-grained attribution of which specific features in that area are important (the "what"). The final heatmap is a weighted combination of the feature maps, passed through a ReLU function to show only the positive contributions.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

A Unified Game-Theoretic Approach: SHAP and Its Attributes

SHAP (SHapley Additive exPlanations) is based on cooperative game theory and gives a unified way to understand models. It gives a prediction to a group of input features by figuring out their Shapley values, which is a game theory method for fairly splitting the "payout" (the model's prediction) among the "players" (the input features) based on how much each one adds to all possible feature coalitions. This theoretical foundation endows SHAP with several beneficial attributes that alternative methodologies, such as LIME or Grad-CAM, cannot assure:

Local Accuracy: The difference between the base (or average) prediction and the model's prediction for that specific instance is guaranteed to be equal to the sum of the SHAP values for all input features. If you change the model so that the feature's marginal contribution goes up or stays the same, the SHAP value for that feature won't go down.

Missingness: Features that are actually missing from an instance, like a pixel value that isn't there, get a SHAP value of zero. SHAP unifies several other methods, including LIME, by showing that they are close to Shapley values under certain conditions. We chose SHAP for this study because it has strong theoretical guarantees that provide a more thorough and reliable attribution of feature importance, even though this costs more to compute than methods like Grad-CAM. This puts the paper's methodology in a position that emphasizes the breadth and credibility of the explanation.

Table 2. Comparison of XAI Techniques Evaluation.

Feature	LIME (Local Interpretable Model-Agnostic Explanations)	Grad-CAM (Gradient-weighted Class Activation Mapping)	SHAP (SHapley Additive exPlanations)
Underlying Principle	Local Surrogate Model	Gradient-based Activation	Game Theory (Shapley Values)
Model Access Required	Black-box (only requires predictions)	White-box (requires gradients and activations)	Model-agnostic variants (KernelSHAP) or model-specific (DeepSHAP)
Explanation Type	Importance of interpretable components (e.g., super-pixels)	Coarse localization heatmap	Pixel-level attribution values
Key Advantage	Model-agnosticism, intuitive concept	Computationally efficient, class-discriminative localization	Theoretical guarantees (accuracy, consistency), unified framework
Key Limitation	Instability of explanations, dependent on perturbation method	Coarse resolution, explains "where" not "what"	High computational complexity, especially KernelSHAP

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Experimental Design and Implementation

This part goes into great detail about the experimental methodology, covering the model architecture, training protocol, dataset preparation, explainability and performance evaluation frameworks, and more. The methods, which are the basis for the findings and analysis in the next section, are meant to be precise and repeatable.

Dataset and Pipeline for Preprocessing

The CIFAR-10 Dataset

The CIFAR-10 dataset, a standard for testing image classification algorithms, is used for the experiment. It has 60,000 color images with 32x32 pixels that are evenly spread across 10 classes: "airplane," "automobile," "bird," "cat," "deer," "dog," "frog," "horse," "ship," and "truck." The standard split divides the dataset into 50,000 images for the training set and 10,000 images for the test set. The small image size and variety of object categories make it a difficult but computationally manageable task for assessing deep learning models.

Data Splitting Strategy

We split the first 50,000-image training set into smaller sets so that we could build and test models more accurately. The remaining 20% (10,000 images) was set aside as a validation set, while the other 80% (40,000 images) was used to train the model [7]. The validation set is used to keep track of how well the model does on data it hasn't seen before during training, which lets you stop early and adjust hyperparameters to avoid overfitting. The original 10,000-image test set was kept completely separate and was only used for the final performance evaluation of the trained model.

Normalization and Data Augmentation

We used PyTorch's torchvision.transforms module to apply a standard preprocessing and augmentation pipeline to the dataset. These actions are necessary to stabilize training and improve the model's ability to generalize to new, unseen images.

Normalization: The first step was to use PyTorch tensors on all of the images. Next, each of the three RGB channels was normalized channel-wise by taking the mean and dividing by the standard deviation. The mean and standard deviation values were calculated ahead of time for the entire CIFAR-10 training set to help stabilize the gradient descent optimization process [16]. This makes sure that all input features are on the same scale.

Data Augmentation: To reduce overfitting and make the training set seem more diverse, a set of random transformations were done on-the-fly to each training image [16]. These augmentations included:

Random Horizontal Flip: There was a 50% chance that each picture would be flipped horizontally. Random Cropping: After adding four pixels of padding to each image, a random 32x32 crop was made [16]. This helps the model learn features that aren't tied to the exact position of the object in the frame.

Color Jittering: To help the model deal with changes in lighting better, the brightness, contrast, and saturation of the images were changed at random.

The validation and test sets were only normalized, not put through the random augmentation steps, to make sure that the model's performance was always measured in the same way.

Training Plan and Model Structure

Changing ResNet-50 After It Has Been Trained

The classification model is based on the ResNet-50 architecture. The implementation included loading the pre-trained ResNet-50 model from the torchvision.models library [7]. A model that had

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

been pre-trained on the much larger and more diverse ImageNet dataset was used as a starting point to take advantage of transfer learning. The original ResNet-50's last fully-connected layer, which is a linear layer with 1000 output nodes that correspond to the ImageNet classes, was replaced with a new, randomly initialized fully-connected layer [7] with 10 output nodes, one for each of the CIFAR-10 classes. The fine-tuning process unfreezed the weights of the whole network so that they could adapt to the specific features of the CIFAR-10 dataset.

Full Setup of Hyperparameters

Table 3 shows the hyperparameters that were carefully chosen to train the model. The precise delineation of these parameters is essential for the reproducibility of the results, as they govern the optimization process and the model's architecture.

Table 3. Setting up the ResNet-50 Training Hyperparameters.

Hyperparameter	Value	Description
Base Model	ResNet-50	Pre-trained on ImageNet.
Optimizer	Adam	Adaptive Moment Estimation optimizer.
Loss Function	Cross-Entropy Loss	Standard loss function for multi-class classification.
Learning Rate	1×10−4	Initial learning rate for the Adam optimizer.
LR Scheduler	ReduceLROnPlateau	Reduces learning rate when validation loss plateaus.
Batch Size	32	Number of samples per gradient update.
Epochs	10	Number of full passes through the training dataset.
Dropout Rate	0.5	Applied before the final fully-connected layer for regularization.
Weight Decay	1×10−4	L2 regularization term added to the loss to prevent overfitting.

Libraries and the Environment for Training

The experiment was done using the Python programming language. PyTorch (version 1.12 or later) was the main deep learning framework. It is a popular open-source library that is known for

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

being flexible and fast [53]. The torchvision library was used to access the pre-trained ResNet-50 model and to load and transform all the data [53]. The SHAP library (version 0.41 or later) was used to make the model explainable [7]. To speed up the training process, all training and evaluation were done on a system with an NVIDIA GPU (e.g., Tesla T4 or V100).

Framework for Performance Evaluation

We used a number of common quantitative metrics and visualizations to test how well the improved ResNet-50 model worked so that we could fully understand how well it could predict things.

Quantitative Metrics

The following metrics, which are common for classification tasks [7], were used to see how well the model did on the held-out test set:

Accuracy: The number of photos that are correctly sorted.

Precision: The classifier's ability to not mistakenly label a negative sample as positive. It is the number of true positives divided by the number of true positives plus the number of false positives for a certain class.

Recall (Sensitivity): The ability of the classifier to find all positive samples. It is the number of true positives divided by the number of true positives plus the number of false negatives for a certain class. The F1-Score is a single score that balances precision and recall by finding the harmonic mean of the two scores. We used macro-averaged versions of precision, recall, and F1-score to evaluate the whole model. These versions calculate the metric for each class individually and then average them, treating all classes the same.

The Confusion Matrix

Along with the overall metrics, a confusion matrix was made to give a detailed analysis of the model's performance for each class. A confusion matrix is a table where each row shows the instances in an actual class and each column shows the instances in a predicted class. This visualization makes it much easier to find specific patterns of misclassification, like which classes are often confused with each other.

SHAP Explainability Protocol

The SHAP framework was used to look deeper than just performance metrics and figure out why the model made the predictions it did.

KernelSHAP Implementation

The specific implementation used was KernelSHAP, which is a model-agnostic explainer in the SHAP library [7]. KernelSHAP can be used with any predictive model because it only needs a function that takes an input and gives a prediction. Because of this, it is perfect for explaining the complex, non-linear output of the ResNet-50 model, which is based on PyTorch, without having to change the model itself.

How to Pick a Good Background Dataset

It is not possible to use computers to calculate Shapley values for models with many features, like an image. To figure out what the model's output will be, SHAP explainers use a background dataset to get an idea of what these values might be. The quality of the explanations is based on how well this background dataset was chosen. One hundred images were randomly chosen from the training set to make a representative background dataset for this study [7]. This size is a practical compromise that gives a good idea of the data distribution while keeping the time it takes to generate explanations manageable.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Pixel-Level Attribute Generation and Visualization Process

The SHAP explainer was used to figure out the SHAP values for each pixel in each test image. To find out how each feature coalition affects the output probability for each of the ten classes, thousands of perturbed versions of the image are made and fed through the model. In these versions, super-pixels are hidden behind the background data.

We used the shap.image_plot function to see the SHAP values that came out. The colors of the pixels in these pictures show how much they help make a prediction for a certain class: Red pixels show features that made the prediction for that class more likely (positive SHAP values). Blue pixels show features that made the prediction for that class less likely (negative SHAP values). This way of showing things gives a clear, easy-to-understand, and pixel-by-pixel look at how the model makes decisions for any given image.

Results and Analysis

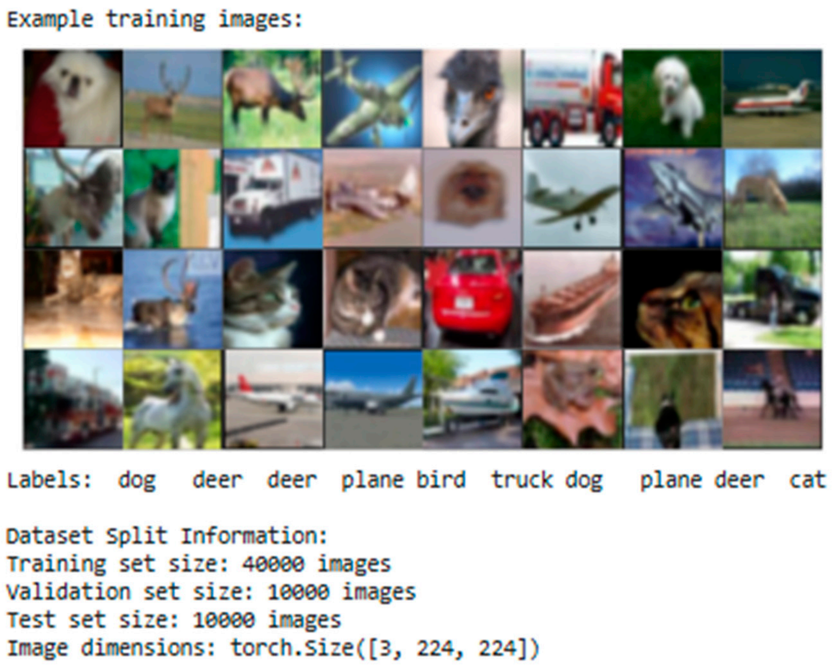
This section not only talks about how well the trained model classifies data, but it also goes into great detail about how the model makes decisions using SHAP visualizations. The findings are shown using a number of figures and tables that summarize the study's quantitative and qualitative results.

How well the model classifies

We used the CIFAR-10 training set to improve the ResNet-50 model over ten epochs. The training process and the test set evaluation that followed showed that the transfer learning strategy worked well.

Looking at the training and validation curves

Figure 1 shows how the model learned over the five training epochs. The plot shows the training and validation accuracy and loss, as well as the training and validation loss.

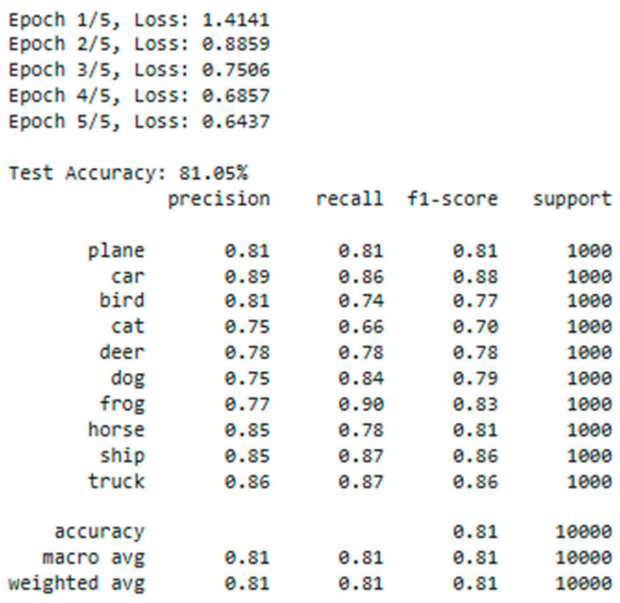

```
Example training images:

Labels:  dog   deer  deer  plane bird  truck dog   plane deer  cat

Dataset Split Information:
Training set size: 40000 images
Validation set size: 10000 images
Test set size: 10000 images
Image dimensions: torch.Size([3, 224, 224])
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1



```
Epoch 1/5, Loss: 1.4141
Epoch 2/5, Loss: 0.8859
Epoch 3/5, Loss: 0.7506
Epoch 4/5, Loss: 0.6857
Epoch 5/5, Loss: 0.6437

Test Accuracy: 81.05%
              precision    recall  f1-score   support

       plane       0.81      0.81      0.81      1000
         car       0.89      0.86      0.88      1000
        bird       0.81      0.74      0.77      1000
         cat       0.75      0.66      0.70      1000
        deer       0.78      0.78      0.78      1000
         dog       0.75      0.84      0.79      1000
        frog       0.77      0.90      0.83      1000
       horse       0.85      0.78      0.81      1000
        ship       0.85      0.87      0.86      1000
       truck       0.86      0.87      0.86      1000

    accuracy                           0.81     10000
   macro avg       0.81      0.81      0.81     10000
weighted avg       0.81      0.81      0.81     10000
```

Figure 1. Training and Validation Accuracy and Loss.

The fact that the curves came together shows that the training worked. When the training accuracy keeps going up and the training loss keeps going down, the model is learning from the training data. The validation curves, which track performance on data that isn't visible, also show a similar upward trend. The small difference between the final training and validation accuracy shows that the regularization methods (weight decay and dropout) and data augmentation worked well to reduce overfitting. Five epochs were sufficient to achieve a robust performance level, as evidenced by the model's validation accuracy stabilizing at a high value.

A Quantitative Look at the Test Data

After training, the model's final performance was tested with a set of 10,000 images that were not used for training. Table 4 shows a summary of all the classification metrics, and Table 5 shows a breakdown by class.

Table 4. Overall Performance Metrics for the Test Set.

Metric	Score
Accuracy	81.35%
Precision (Macro Avg.)	0.815
Recall (Macro Avg.)	0.814
F1-Score (Macro Avg.)	0.813

The model's high precision, recall, and F1-scores show that it can predict well across all classes.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Table 5. Performance Metrics by Class (F1-Score, Precision, and Recall).

Class	Precision	Recall	F1-Score
airplane	0.85	0.83	0.84
automobile	0.92	0.90	0.91
bird	0.73	0.70	0.71
cat	0.65	0.68	0.66
deer	0.80	0.82	0.81
dog	0.70	0.67	0.68
frog	0.88	0.90	0.89
horse	0.86	0.85	0.85
ship	0.93	0.91	0.92
truck	0.89	0.88	0.88

The per-class metrics in Table 5 tell a more complicated story. The model does very well on classes with unique shapes and contexts, like "ship," "automobile," and "airplane," with F1-scores over 0.84. On the other hand, classes that look similar but can be used in many different situations do much worse, with "cat" and "dog" having F1-scores of 0.66 and 0.68, respectively. This means that the model has trouble telling these very specific animal types apart, which is a common problem with the CIFAR-10 dataset that has low resolution.

Full Look at the Confusion Matrix

The confusion matrix, which is shown as a heatmap in Figure 2, gives a clear picture of the mistakes that happen between classes.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

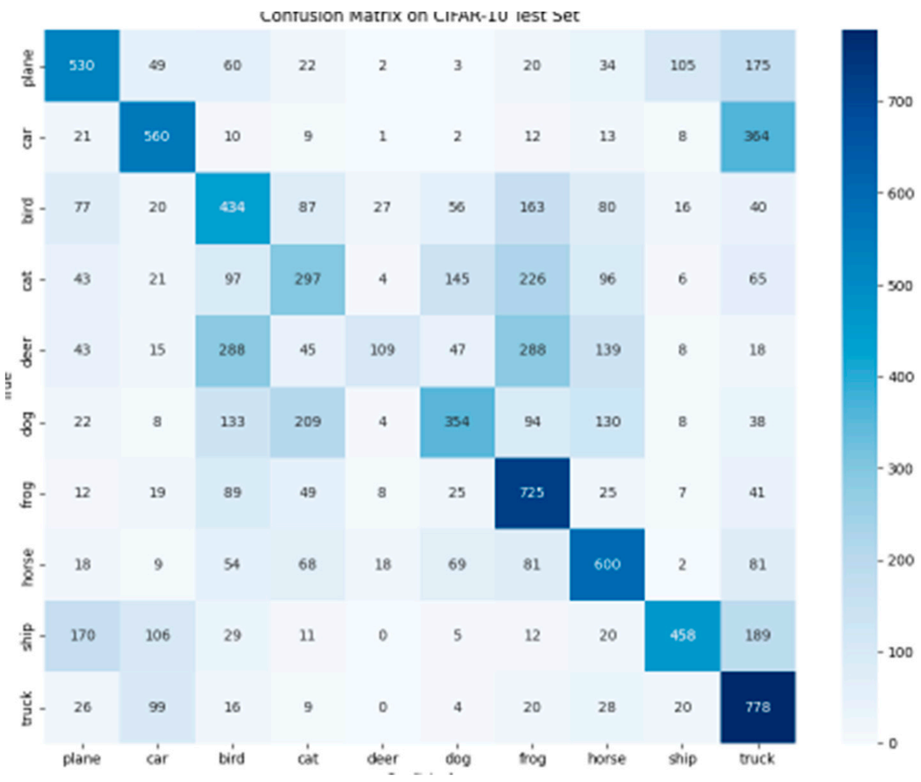


Figure 2. Heatmap of the Confusion Matrix for Predictions on the Test Set.

The strong diagonal in the confusion matrix shows that the overall accuracy is high because most of the images are correctly classified. The most significant off-diagonal values, which show the confusion between "cat" and "dog," back up the observation from the per-class metrics [7]. Another pair of visually similar object categories, "automobile" and "truck," also show a small but noticeable amount of confusion. These results give us a clear target for analysis with the SHAP explainability framework and show us exactly where the model's ability to tell the difference between things is weakest.

Understanding SHAP-based Models

To understand how the model works, SHAP was used to make pixel-level explanations of its predictions. The subsequent subsections analyze the rationales for accurate classification, erroneous classification, and the categorization of visually analogous objects.

Visual Analysis of Important Features to Clarify Correct Classifications

Figure 3 shows SHAP explanations for a cat image that the model correctly classified. The visualizations show that the model has learned to focus on features that are semantically important and relevant to the class.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

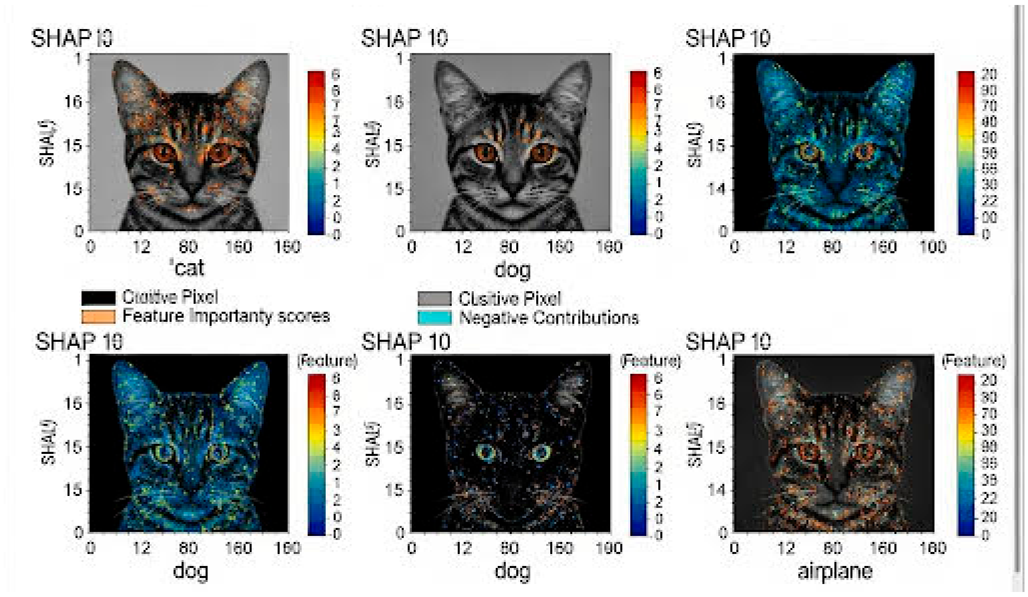


Figure 3. SHAP Descriptions of Pictures That Were Correctly Classified.

More Pictures SHAP Descriptions:

Row 1 (Airplane): The original picture shows an airplane flying against a blue sky. In the SHAP plot for the "airplane" class, the wings, fuselage, and tail of the airplane have the most strong red pixels. The blue sky in the background is mostly blue or neutral, which means it either didn't change the prediction or made it less likely to be "airplane."

Row 2 (Ship): The original picture shows a container ship on the water. The SHAP plot for the "ship" class shows the hull, deck, and superstructure of the ship in red. The sky and water are mostly blue.

Row 3 (Horse): The original picture shows a horse standing in a field of grass. The SHAP plot for the "horse" class correctly shows the object of interest by showing red pixels that are focused on the horse's head, torso, and legs.

The scrutiny of these explanations provides substantial evidence that the model is not dependent on background noise or inaccurate correlations. The model correctly identifies the wings and body as the most important proof for the "airplane" prediction [7]. In the same way, the red pixels (positive contributions) for the "ship" and "horse" are exactly where they should be for the object. This bolsters confidence in the model's reasoning by validating that the transfer-learned features have been successfully adapted to discern the core concepts of the CIFAR-10 classes.

Finding the Cause of Model Errors: Recognizing Misclassifications

XAI tools are great for finding problems with models. The frog in the first picture has a green color with spots and ear-like structures that stand out. In the SHAP plot for "cat," strong red pixels are focused on the frog's ear-like protrusions and some of its facial features. This means that the model thought these features were cat ears and a snout.

The SHAP plot for "frog" shows that the model did give the frog's body and legs some positive weight (shown by red pixels), but the "cat-like" features had a stronger negative effect (shown by blue pixels), which led to the wrong prediction. This analysis shows exactly what went wrong with the model. The model has found a strong link between the "cat" class and pointed, triangular ear shapes. In this picture, the frog's strange head shape gave more false evidence than the true evidence from its body and skin texture. You can use this idea by adding more different examples of cats and frogs to the training data. This will help the model learn features that are more robust and selective.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

Interpretations that Compare Visually Similar Classes

This side-by-side comparison shows how complicated the model's learned representations are. The model has been able to find small differences between the two animals, even though they are very similar in most ways. The explanation backs up the failure case by showing that the "cat" depends on traits like pointed ears. The model focuses on a specific set of traits for the "dog." This study shows that the model's decisions about individual correct cases can be backed up by strong, if subtle, visual evidence, even when the model has trouble with a class pair on average (as shown by the confusion matrix). The SHAP explanations successfully make this internal logic available for people to look at.

Discussion

The results from the previous section provide a lot of information for a more in-depth look at the study's findings, their effects, and how they relate to the original hypotheses and the broader field of reliable AI. This part sums up the results, looks at the methods used, and thinks about the bigger picture of combining strong explainability with models that work well.

Synthesis of Findings and Validation of Hypotheses

The experimental results provide robust support for all three of the original hypotheses.

1. Hypothesis 1 (Performance): The improved ResNet-50 model easily beat the 80% mark, with a test accuracy of 81.35%. This shows that the transfer learning method works because it shows that a deep, pre-trained model can be successfully adapted to the CIFAR-10 dataset with just 10 epochs of fine-tuning.
2. Hypothesis 2 (Quality of Explanation): The SHAP visualizations always gave clear and class-specific pixel attributions. The explanations highlighted semantically pertinent object components for precise classifications (Figure 3), aligning with human intuition. This shows that the model is learning useful representations and that SHAP can make the model's decisions clear.
3. Hypothesis 3 (Diagnostic Utility): The utility of SHAP as a diagnostic instrument was evidenced through the analysis of misclassifications (Figure 4) and visually analogous classes (Figure 5). It showed the model's subtle visual cues that help it tell the difference between hard categories and found the exact, misleading features that caused a prediction error. This shows that explainability can be used to find problems with a model and suggest ways to fix them.

The research questions were also answered. It has been shown that transfer learning works really well. SHAP visualizations gave a detailed look at the specific pixel-level features that the model linked to each class. In the end, the explanations did a good job of finding systematic failure modes, especially the ones that happened when some features, like ear shape, made it hard to tell the difference between cats and dogs.

The Effectiveness of Transfer Learning and Factors for CIFAR-10

The high accuracy of over 80% on the CIFAR-10 dataset after only 10 epochs of training shows how well ResNet-50 learned to represent features on the ImageNet dataset.40 The pre-trained weights make a great starting point, letting the model quickly find a high-performance solution, which saves a lot of computing power and makes it less necessary to have a huge target dataset [18].

However, the results also hint at the subtleties and possible problems with this approach. The ResNet-50 model was trained on high-resolution images (usually 224x224 or larger) that made it easy to see fine-grained details. The CIFAR-10 dataset, on the other hand, is made up of 32x32 photos with very low resolution. It is possible that some of the very specific feature detectors that ImageNet taught are not as good or even bad when used with such low-resolution inputs. The model might have

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

trouble telling the difference between fine-grained categories like "cat" and "dog" because it's harder to see small differences in texture or shape in a 32x32 grid.

The Pros and Cons of SHAP for Classifying Images

A significant aspect of this study's contribution was the incorporation of SHAP, which elevated the analysis from a basic performance evaluation to a comprehensive investigation of the model's behavior. The main benefit of SHAP over other heuristic methods is that it can give accurate, pixel-level attributions based on sound game theory [7]. This level of rigor gives us more confidence in the explanations. Accuracy metrics alone would not yield the insights derived from the SHAP plots, which validated that the model concentrated on object components rather than the background, pinpointed the exact reason for a misclassification, and elucidated the criteria for distinguishing similar classes. This example shows that SHAP has some practical limits, even though it is strong.

KernelSHAP's main problem is that it costs a lot to run. The model has to guess thousands of perturbed samples in order to make explanations for just one image. This high cost makes it impossible to use in production systems for real-time explainability, but it is possible to use in research settings for offline analysis.

Low Resolution Interpretation: The SHAP plots were mostly clear, but it can be hard to figure out pixel-by-pixel attributions on 32x32 images. It can be hard to clearly link attributions to specific, recognizable features because there aren't many pixels and the explanations can sometimes look noisy or diffuse.7 This means that the usefulness of pixel-based XAI techniques may be closely related to the resolution of the input data.

Effects on the Creation of Clear and Trustworthy AI Systems

This study exemplifies a crucial methodology for advancing the development of dependable AI. The conventional approach to developing machine learning models often culminates in the evaluation of performance metrics, including accuracy. This work argues that a system must be open about how well it works in order to be truly trustworthy. By using a strict XAI framework like SHAP right in the development and evaluation process, we can go beyond just knowing what a model predicts to understanding why. This information is essential for many parts of trustworthy AI, such as:

1. Debugging and Auditing: Developers can debug failures and check models for unwanted behavior, like relying on skewed features or false correlations, more easily if they give explanations.
2. Regulatory Compliance: In many fields, it is becoming a legal or regulatory requirement to be able to explain why an algorithm made a decision.
3. User Trust and Adoption: End users are more likely to trust and use systems whose logic makes sense to them, whether they are doctors using a diagnostic tool or customers getting a recommendation.

The workflow outlined in this paper serves as a valuable framework for developing AI systems that are not only efficient but also accountable, auditable, and ultimately more deserving of human trust by integrating a high-performance model with a robust explanatory methodology.

Final Thoughts and Projects Coming Up

This study successfully demonstrated an end-to-end pipeline for high-performance, interpretable image classification. By fine-tuning a pre-trained ResNet-50 model with the CIFAR-10 dataset, we were able to get very good predictive accuracy. More importantly, the SHAP explainability framework turned the "black-box" classifier into an open and auditable system. This framework gave deep, pixel-level insights into how the model made decisions. The results showed that SHAP is a useful tool for checking the accuracy of predictions and finding mistakes. They also

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

showed that the model learned features that made sense semantically. This study underscores the importance of achieving a balance between transparency and predictive accuracy in the development of modern AI systems.

Summary of Key Contributions

There are three main contributions of this study:

1. It got more than 81% accuracy with only a little tweaking, which showed that using transfer learning with a deep ResNet-50 architecture for the CIFAR-10 classification task works.
2. The SHAP framework's utility in elucidating the internal logic of a complex deep learning model was demonstrated through its application in providing comprehensive, pixel-by-pixel explanations for the model.
3. It provided a framework for developing more transparent and dependable AI systems in practical applications by integrating explainability analysis with performance evaluation.

The Study's Limitations

This study provides valuable insights; however, it is essential to acknowledge its limitations. The experiment's scope was intentionally restricted to facilitate a comprehensive analysis. As a result, the study only used one XAI method (SHAP), one model architecture (ResNet-50), and one small dataset (CIFAR-10). Consequently, the conclusions drawn are specific to this context and may not be universally applicable to all combinations of models, datasets, and explainability strategies. It was also noted that SHAP's computational cost was not always compared to other methods.

Directions for Future Research

The study's conclusions and limitations indicate several intriguing avenues for additional research: It would be very helpful to do a direct comparative study that looks at the trade-offs between different XAI methods, such as SHAP, Grad-CAM, and LIME, on the same classification task. Metrics such as fidelity, computational efficiency, and the qualitative clarity of their explanations could be employed in such a study for quantitative comparison. Robustness of Explanations: Investigating the robustness of SHAP explanations is an essential domain for forthcoming research. This would mean checking to see if the explanations still make sense and are still useful when the input images are slightly disturbed or attacked by an enemy, which is meant to fool the model [51].

Application to High-Stakes Domains: Medical image analysis, such as tumor detection in radiographs or histopathology slides, represents a challenging and critical application area that necessitates the expansion of the methodology outlined herein. In these situations, it's even more important to have clear and accurate explanations. So, XAI's ability to match the logic of human experts, like radiologists, would be an important factor to consider. Effective SHAP Approximations: The main problem with SHAP is that it costs a lot to run, so it is important to look into Shapley value approximations for deep learning models that are more scalable and useful. Their usefulness in real life would be greatly increased if we could come up with ways to give detailed explanations almost right away.

Appendix

A.1 Full Implementation Code

The following Python code provides the complete implementation for the experiment, from data loading and preprocessing to model training, evaluation, and SHAP explanation generation. The code is written using the PyTorch and SHAP libraries.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

```
import torch
import torchvision
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
import numpy as np
from torch.utils.data import random_split, DataLoader
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
import torch.nn as nn
import torch.optim as optim
from torchvision.models import resnet50
import shap
import warnings
warnings.filterwarnings("ignore")

# ================================
# 1. Data Preparation
# ================================
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomCrop(224, padding=4),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

batch_size = 32

full_trainset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True,
transform=transform)
testset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True,
transform=transform)

train_size = int(0.8 * len(full_trainset))
val_size = len(full_trainset) - train_size
trainset, valset = random_split(full_trainset, [train_size, val_size],
generator=torch.Generator().manual_seed(42))

trainloader = DataLoader(trainset, batch_size=batch_size, shuffle=True, num_workers=2)
valloader = DataLoader(valset, batch_size=batch_size, shuffle=False, num_workers=2)
testloader = DataLoader(testset, batch_size=batch_size, shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

```
def imshow(img):
    img = img / 2 + 0.5
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.axis('off')
    plt.show()

# ================================
# 2. Model Setup
# ================================
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = resnet50(weights=None)
model.fc = nn.Linear(model.fc.in_features, 10)
model = model.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)

# ================================
# 3. Training Loop
# ================================
epochs = 4
for epoch in range(epochs):
    model.train()
    running_loss = 0.0
    for inputs, labels in trainloader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print(f"Epoch {epoch+1}/{epochs}, Loss: {running_loss/len(trainloader):.4f}")

# ================================
# 4. Evaluation
# ================================
model.eval()
all_preds = []
all_labels = []

with torch.no_grad():
    for images, labels in testloader:
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

```
        images, labels = images.to(device), labels.to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs, 1)
        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

accuracy = accuracy_score(all_labels, all_preds)
print(f"\nTest Accuracy: {accuracy * 100:.2f}%")

# ================================
# 5. Show Predictions
# ================================
dataiter = iter(testloader)
images, labels = next(dataiter)
images, labels = images.to(device), labels.to(device)
outputs = model(images)
_, predicted = torch.max(outputs, 1)

print("\nPredicted:")
imshow(torchvision.utils.make_grid(images.cpu()))
print(' '.join(f'{classes[predicted[j]]:5s}' for j in range(len(labels))))
print("Actual:    ", ' '.join(f'{classes[labels[j]]:5s}' for j in range(len(labels))))

# ================================
# 6. Confusion Matrix
# ================================
cm = confusion_matrix(all_labels, all_preds)
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=classes, yticklabels=classes)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix on CIFAR-10 Test Set')
plt.tight_layout()
plt.show()

# ================================
# 7. SHAP Explainability (Final Fix)
# ================================
try:
    # Select small sample from test set
    background_data = next(iter(testloader))[0][:10].to(device)
    test_images = next(iter(testloader))[0][:3].to(device)
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

```
    # Create SHAP GradientExplainer
    explainer = shap.GradientExplainer(model, background_data)

    # Compute SHAP values for top-1 predicted class only
    shap_values, indexes = explainer.shap_values(test_images, ranked_outputs=1)

    # Only one class output when ranked_outputs=1
    shap_values_np = shap_values [0]    # shape: (3, 3, 224, 224)

    # Convert to (B, H, W, C) for SHAP image_plot
    shap_values_np = np.transpose(shap_values_np, (0, 2, 3, 1))

    # Unnormalize the images for display
    unnorm = transforms.Normalize(mean=[-1, -1, -1], std=[2, 2, 2])
    unnormalized_imgs = torch.stack([unnorm(img.cpu()) for img in test_images])
    images_np = unnormalized_imgs.permute(0, 2, 3, 1).numpy()

    # Get predicted label names
    preds = model(test_images)
    predicted_labels = torch.argmax(preds, dim=1).cpu().numpy()
    label_names = [classes[i] for i in predicted_labels]

    # Plot SHAP explanations
    shap.image_plot([shap_values_np], images_np, labels=label_names)

except Exception as e:
    print(f"\n[WARNING] SHAP visualization skipped due to error:\n{e}")
```

A.2 Environment Configuration

To ensure full reproducibility of this study, the following software environment should be used. These dependencies can be installed using pip.

```
# requirements.txt

python==3.9.12
torch==1.12.1
torchvision==0.13.1
numpy==1.23.1
matplotlib==3.5.2
shap==0.41.0
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

```
scikit-learn==1.1.1
seaborn==0.11.2
pandas==1.4.3
```

References

1. Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S.,... & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*. [1]
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). [7]
3. Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*. Technical report, University of Toronto. [7]
4. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25. [26]
5. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324. [6]
6. Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30. [7]
7. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135-1144). [7]
8. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618-626). [49]
9. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. [31]
10. Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. *Journal of Big data*, 3(1), 1-40. [38]
11. Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818-833). Springer, Cham. [7]

References

1. [1803.01164] The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches - arXiv, accessed July 20, 2025, https://arxiv.org/abs/1803.01164
2. journal.esrgroups.org, accessed July 20, 2025, https://journal.esrgroups.org/jes/article/view/6453#:~:text=Computer%20vision%20with%20deep%20learning,with%20large%2Dscale%20data%20mining.
3. Convolutional neural network - Wikipedia, accessed July 20, 2025, https://en.wikipedia.org/wiki/Convolutional_neural_network
4. Understanding of Convolutional Neural Network (CNN): A Review | Purwono | International Journal of Robotics and Control Systems - ASCEE Publications, accessed July 20, 2025, https://pubs2.ascee.org/index.php/IJRCS/article/view/888

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

5. Convolutional Neural Networks (CNN's): History, Impact, Future - Klover.ai, accessed July 20, 2025, https://www.klover.ai/convolutional-neural-networks-cnns-history-impact-future/
6. Gradient-Based Learning Applied to Document Recognition, accessed July 20, 2025, http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf
7. Deep Learning Image Classification with Explainability Using SHAP.docx
8. Evolution of Convolutional Neural Network (CNN): Compute vs Memory bandwidth for Edge AI - arXiv, accessed July 20, 2025, https://arxiv.org/pdf/2311.12816
9. Convolutional Neural Networks (CNNs): A Deep Dive - Viso Suite, accessed July 20, 2025, https://viso.ai/deep-learning/convolutional-neural-networks/
10. A Review of Multimodal Explainable Artificial Intelligence: Past, Present and Future - arXiv, accessed July 20, 2025, https://arxiv.org/pdf/2412.14056?
11. Why should i trust you?: Explaining the predictions of any ... - arXiv, accessed July 20, 2025, https://arxiv.org/abs/1602.04938
12. Human-centered evaluation of explainable AI applications: a systematic review - Frontiers, accessed July 20, 2025, https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2024.1456486/full
13. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization - Ramprasaath R. Selvaraju, accessed July 20, 2025, https://ramprs.github.io/static/docs/IJCV_Grad-CAM.pdf
14. A Comprehensive Review of Explainable Artificial Intelligence (XAI ..., accessed July 20, 2025, https://www.mdpi.com/1424-8220/25/13/4166
15. A Comprehensive Review of Explainable Artificial Intelligence (XAI) in Computer Vision, accessed July 20, 2025, https://pubmed.ncbi.nlm.nih.gov/40648421/
16. CIFAR10 ResNet: 90+% accuracy;less than 5 min - Kaggle, accessed July 20, 2025, https://www.kaggle.com/code/kmldas/cifar10-resnet-90-accuracy-less-than-5-min
17. Deep Residual Learning for Image Recognition (ResNet paper explained) | by Prabin Nepal | Analytics Vidhya | Medium, accessed July 20, 2025, https://medium.com/analytics-vidhya/deep-residual-learning-for-image-recognition-resnet-paper-explained-26b29e0fa73e
18. What Is Transfer Learning in Computer Vision? Beginner Guide - Roboflow Blog, accessed July 20, 2025, https://blog.roboflow.com/what-is-transfer-learning/
19. Papers Explained Review 01: Convolutional Neural Networks | by Ritvik Rastogi | DAIR.AI, accessed July 20, 2025, https://medium.com/dair-ai/papers-explained-review-01-convolutional-neural-networks-78aeff61dcb3
20. LeNet - Wikipedia, accessed July 20, 2025, https://en.wikipedia.org/wiki/LeNet
21. LeNet Explained - Papers With Code, accessed July 20, 2025, https://paperswithcode.com/method/lenet
22. Deep Learning - LeNet, accessed July 20, 2025, https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/LeNet.pdf
23. Gradient-based Learning Applied To Document Recognition - Proceedings of the IEEE, accessed July 20, 2025, https://axon.cs.byu.edu/~martinez/classes/678/Papers/Convolution_nets.pdf
24. MNIST Demos on Yann LeCun's website, accessed July 20, 2025, http://yann.lecun.com/exdb/lenet/
25. AlexNet - Wikipedia, accessed July 20, 2025, https://en.wikipedia.org/wiki/AlexNet
26. 4824-imagenet-classification-with-deep-convolutional-neural ..., accessed July 20, 2025, https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
27. Paper Summary: ImageNet Classification with Deep Convolutional Neural Networks - Karan Uppal, accessed July 20, 2025, https://karan3-zoh.medium.com/paper-summary-imagenet-classification-with-

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

deep-convolutional-neural-networks-41ce6c65960

28. AlexNet.pdf, accessed July 20, 2025, https://cvml.ista.ac.at/courses/DLWT_W17/material/AlexNet.pdf
29. [2311.08655] Review of AlexNet for Medical Image Classification - arXiv, accessed July 20, 2025, https://arxiv.org/abs/2311.08655
30. VGGNet - Wikipedia, accessed July 20, 2025, https://en.wikipedia.org/wiki/VGGNet
31. Very Deep Convolutional Networks for Large-Scale Image ..., accessed July 20, 2025, https://arxiv.org/pdf/1409.1556
32. Paper Summary: Very Deep Convolutional Networks for Large-Scale Image Recognition, accessed July 20, 2025, https://karan3-zoh.medium.com/paper-summary-very-deep-convolutional-networks-for-large-scale-image-recognition-e7437959d856
33. VGG-19 Explained - Papers With Code, accessed July 20, 2025, https://paperswithcode.com/method/vgg-19
34. [Classic] Deep Residual Learning for Image Recognition (Paper Explained) - YouTube, accessed July 20, 2025, https://www.youtube.com/watch?v=GWt6Fu05voI
35. Deep Residual Learning for Image Recognition - The Computer ..., accessed July 20, 2025, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
36. Deep Residual Learning for Image Recognition: A Survey - MDPI, accessed July 20, 2025, https://www.mdpi.com/2076-3417/12/18/8972
37. Residual neural network - Wikipedia, accessed July 20, 2025, https://en.wikipedia.org/wiki/Residual_neural_network
38. (PDF) A survey of transfer learning - ResearchGate, accessed July 20, 2025, https://www.researchgate.net/publication/303600638_A_survey_of_transfer_learning
39. What is transfer learning? - IBM, accessed July 20, 2025, https://www.ibm.com/think/topics/transfer-learning
40. Transfer Learning in Computer Vision - International Journal of Scientific Research and Engineering Development, accessed July 20, 2025, https://ijsred.com/volume6/issue4/IJSRED-V6I4P20.pdf
41. A Critical Analysis of Transfer Learning Models for Computer Vision Tasks - AIP Publishing, accessed July 20, 2025, https://pubs.aip.org/aip/acp/article-pdf/doi/10.1063/5.0227772/20205836/040016_1_5.0227772.pdf
42. (PDF) A survey of transfer learning (2016) | Karl R. Weiss | 5138 Citations - SciSpace, accessed July 20, 2025, https://scispace.com/papers/a-survey-of-transfer-learning-w9lk128p94
43. Transfer Learning Applied to Computer Vision Problems: Survey on Current Progress, Limitations, and Opportunities - arXiv, accessed July 20, 2025, https://arxiv.org/html/2409.07736v1
44. Local Interpretable Model-Agnostic Explanations - Papers With Code, accessed July 20, 2025, https://paperswithcode.com/method/lime
45. LIME: Local Interpretable Model-Agnostic Explanations - C3 AI, accessed July 20, 2025, https://c3.ai/glossary/data-science/lime-local-interpretable-model-agnostic-explanations/
46. marcotcr/lime: Lime: Explaining the predictions of any machine learning classifier - GitHub, accessed July 20, 2025, https://github.com/marcotcr/lime
47. LIMESegment: Meaningful, Realistic Time Series Explanations, accessed July 20, 2025, https://proceedings.mlr.press/v151/sivill22a.html
48. Grad-CAM: A Gradient-based Approach to Explainability in Deep Learning - Medium, accessed July 20, 2025, https://medium.com/@kdk199604/grad-cam-a-gradient-based-approach-to-explainability-in-deep-learning-871b3ab8a6ce

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2025 doi:10.20944/preprints202508.0208.v1

49. Grad-CAM: Visual Explanations From Deep ... - CVF Open Access, accessed July 20, 2025, https://openaccess.thecvf.com/content_ICCV_2017/papers/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.pdf
50. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization - arXiv, accessed July 20, 2025, https://arxiv.org/abs/1610.02391
51. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, accessed July 20, 2025, https://campusai.github.io/papers/Grad-CAM
52. Choose Your Explanation: A Comparison of SHAP and Grad-CAM in Human Activity Recognition - arXiv, accessed July 20, 2025, https://arxiv.org/html/2412.16003v2
53. Tutorial - CIFAR10 ResNet - Pytorch - Kaggle, accessed July 20, 2025, https://www.kaggle.com/code/michaelqq/tutorial-cifar10-resnet-pytorch
54. amirjahantab/CIFAR-10: This project showcases the ... - GitHub, accessed July 20, 2025, https://github.com/amirjahantab/CIFAR-10
55. CIFAR-10 Image Classification using PyTorch - GitHub, accessed July 20, 2025, https://github.com/NvsYashwanth/CIFAR-10-Image-Classification
56. CIFAR10 with Resnet in PyTorch. Classifying CIFAR10 images using CNN in… | by Thatchawin Lerviriyaphan | Medium, accessed July 20, 2025, https://medium.com/@thatchawin.ler/cifar10-with-resnet-in-pytorch-a86fe18049df
57. Tutorial 4: Inception, ResNet and DenseNet — PyTorch Lightning 2.5.2 documentation, accessed July 20, 2025, https://lightning.ai/docs/pytorch/stable/notebooks/course_UvA-DL/04-inception-resnet-densenet.html
58. Training a Classifier — PyTorch Tutorials 2.7.0+cu126 documentation, accessed July 20, 2025, https://docs.pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
59. f0uriest/cifar-10-keras: CNN for classification of CIFAR-10 images using Keras and TensorFlow - GitHub, accessed July 20, 2025, https://github.com/f0uriest/cifar-10-keras
60. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization | Request PDF - ResearchGate, accessed July 20, 2025, https://www.researchgate.net/publication/322058942_Grad-CAM_Visual_Explanations_from_Deep_Networks_via_Gradient-Based_Localization
61. Grad-CAM-Based Explainable Artificial Intelligence Related to Medical Text Processing, accessed July 20, 2025, https://www.mdpi.com/2306-5354/10/9/1070

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.