DEBUGGING IMAGE CLASSIFICATION ALGORITHMS USING HUMAN ASSISTED FEEDBACK

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KADIRCAN BECEK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2022

Approval of the thesis:

DEBUGGING IMAGE CLASSIFICATION ALGORITHMS USING HUMAN ASSISTED FEEDBACK

submitted by **KADIRCAN BECEK** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil KALIPÇILAR
Dean, Graduate School of **Natural and Applied Sciences** ______________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ______________

Prof. Dr. Gözde Bozdağı Akar
Supervisor, **Electical and Electronics Engineering, METU** ______________

Examining Committee Members:

Prof. Dr. İlkay Ulusoy
Electical and Electronics Engineering, METU ______________

Prof. Dr. Gözde Bozdağı Akar
Electical and Electronics Engineering, METU ______________

Prof. Dr. Ece Güran Schmidt
Electical and Electronics Engineering, METU ______________

Assoc. Prof. Dr. Elif Vural
Electical and Electronics Engineering, METU ______________

Assoc. Prof. Dr. Seniha Esen Yüksel Erdem
Electical and Electronics Engineering, Hacettepe University ______________

Date:15.12.2022

Name, Surname: Kadircan Becek

Signature :

ABSTRACT

DEBUGGING IMAGE CLASSIFICATION ALGORITHMS USING HUMAN ASSISTED FEEDBACK

Becek, Kadircan
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Gözde Bozdağı Akar

December 2022, 88 pages

Neural Network debugging is a relatively newly emerged field of deep learning which tries to improve upon learned parameters with external assistance. Two academic fields are particularly active in addressing this issue: the field of feature visualizations and explainable artificial intelligence. While these methods can improve how to interpret a neural network, they can not provide the expert opinion of a human to improve the accuracy of a neural network. Therefore, while being interpreted correctly, these networks can have undesired outputs. For instance, they may have biases against some classes or may not work effectively in the wild due to overfitting.

In this thesis, we propose a method with two steps to tackle this problem: (1) presenting feature contribution using visualization, (2) taking feedback from humans, and retraining the network by disabling irrelevant or adversarial features.

Keywords: Explainable AI, Feature Visualization, debugging

ÖZ

İNSAN YARDIMLI GERİ BİLDİRİM KULLANARAK RESİM SINIFLANDIRMA ALGORİTMALARINDA HATA AYIKLAMA

Becek, Kadircan
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Aralık 2022 , 88 sayfa

Sinir Ağı hata ayıklaması, öğrenilen parametreleri harici yardımla geliştirmeye çalışan, nispeten yeni ortaya çıkmış bir derin öğrenme alanıdır. Bu konuyu ele almada özellikle aktif olan iki akademik alan vardır: görsel analitik ve Açıklanabilir Yapay Zeka. Bu yöntemler bir sinir ağının nasıl yorumlanacağını iyileştirebilirken, sinir ağının doğruluğunu artırmak için bir insanın uzman görüşünü sağlayamaz. Dolayısıyla bu ağlar yorumlanırken istenmeyen çıktılara sahip olabilir. Örneğin, bazı sınıflara karşı önyargıları olabilir veya aşırı öğrenme nedeniyle etkili bir şekilde çalışmayabilirler.

Bu tezde, bu sorunun üstesinden gelmek için iki adımlı bir yöntem öneriyoruz: (1) görselleştirme kullanarak özellik katkısını sunmak, (2) insanlardan geri bildirim almak ve alakasız veya çekişmeli özellikleri devre dışı bırakarak ağı yeniden eğitmek.

Anahtar Kelimeler: Açıklanabilir YZ, Özellik Görselleştirme, hata ayıklama

To my friends and family

ACKNOWLEDGMENTS

I would like to express my deep and sincere gratitude to my supervisor Prof. Dr. Gözde Bozdağı Akar for her guidance, assistance, support, and patience throughout the thesis studies.

I would like to thank my family and friends for supporting and believing in me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	Two Dimensional
AI	Artificial Intelligence
ALICE	Active Learning with Contrastive Explanations
AM	Activation Maximization
ANN	Artificial Neural Networks
BCE	Binary Cross Entropy
CAM	Class Activation Map
CBM	Concept Bottleneck Models
CNN	Convolutional Neural Networks
CPPN	Compositional Pattern Producing Networks
CPU	Central Processing Unit
CV	Computer Vision
DGN-AM	Deep Generative Network for Activation Maximization
DL	Deep Learning
EC	Explanation Capacity
FIND	Feature Inspection aNd Disabling
FPR	False Positive Rate
FNR	False Negative Rate
FV	Feature Visualization
GAN	Generative Adversarial Networks
GPU	Graphics Processing Unit
HINT	Human Importance-aware Network Tuning
HP	High Priority
LIME	Local Interpretable Model Agnostic Explanation

LP	Low Priority
LRP	Layer-wise Relevance Propagation
ML	Machine Learning
MP	Medium Priority
MSE	Mean Squared Error
NLL	Negative Log Likelihood
NLP	Natural Language Processing
NN	Neural Network
PC	Personal Computer
PR	Pattern Recognition
RRR	Right for right reasons
SENN	Self Explaining Neural Networks
SIMD	Single Instruction Multiple Dataset
SGD	Stochastic Gradient Descent
TL	Transfer Learning
TPU	Tensor Processing Unit
VQA	Visual Question Answering
XAI	Explainable Artificial Intelligence
XIL	Explanatory Interactive Learning

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Deep Learning (DL) is a powerful method that empowered the capabilities of various research fields such as Computer Vision (CV), Natural Language Processing (NLP), Pattern Recognition (PR), etc. Despite the great success of deep neural networks, little is explored for debugging internal problems that affect the reliability and robustness of the decision-making strategies they employ. Deep learning models can perform remarkably well when given enough high-quality training data. Other than ideal cases like benchmark results obtained by AlexNet [13], DenseNet [14], Inception [15], SqueezeNet [16] on ImageNet-1K [17] in this case is diverse, rich and huge; diversity in most datasets used by real-life applications is scarce. Usually, in real-life cases, datasets are small and biased towards the person or group who collects and labels them. This bias in data collection and annotation can lead to models which are not generalized to their intended use case. For example, the model can overfit to one class due to erroneous data collection which requires collecting and annotating additional data or balancing classes by removing data from the dataset which is already small in size. This erroneous behavior created a new problem to be solved: debugging the decision-making mechanism of deep neural networks by inspecting the internals of hidden layers. In this thesis, we will analyze this behavior and try to solve this by regulating networks using expert knowledge, hence debugging the network.

To overcome problems of datasets, there are methods that can transfer knowledge from one trained network to another to reduce training time and use learned knowledge from a better-trained network. These methods which are gathered under a single topic as Transfer Learning (TL) [18, 19] use knowledge source (or in most cases a pre-trained network whose performance is known to work better in generalizing) on one domain and transfer it to a specific and smaller domain to benefit from the success of source and have the best result with the limited amount of data. However, these methods have limited external control to influence the training process and inference outcome.

Debugging deep learning models [20, 21, 22, 11, 10, 12, 23] aims to find errors that create vulnerability and try to correct them to avoid problems. Neural network debugging can be utilized in many areas such as CV where sub-tasks are object detection and recognition, image classification and generation, and NLP where sub-tasks are text comprehension, semantics analysis, etc.

To get rid of erroneous parameters, debug and improve the model, previous works looked into different methods such as analyzing symmetry and unconsumed information on an image by filters [20] and calculating the importance of features by dissecting and aligning results with a pre-trained GAN (Generative Adversarial Networks) [21].

To debug the internals of deep learning models, one needs to understand what is learned throughout the training process. The early method which is still used as a baseline algorithm to get a comprehension of what gets the most confident result on a neuron in a model is Activation Maximization (AM) [24] which tries to maximize the response of a neuron by trying to optimize the input with respect to that neuron.

This created the field Feature Visualization (FV) [25] where random Gaussian input is fed to the network and optimized until maximum value generating input is created for a particular neuron, channel, or layer of the network. Also to reason with outputs or intermediate results of the network, Explainable AI (XAI) [6, 26] has emerged.

With the help of Feature Visualization (FV) and Explainable AI (XAI), the models can be finally peeked through to get a sense of which parts of the input model's internal parameters take into consideration while making the final decision; hence, allowing human knowledge to be integrated with higher level. Model debugging utilizing human feedback has can be inspected in two different types: modifying the model through the dataset and modifying the model directly. The former utilizes feedback by sampling results and displaying them to humans to give input so that it can be fixed at the next iteration in the dataset. These methods change or append data to the dataset itself to improve the quality of data or change model parameters indirectly. CAIPI [22] is one of the methods where it utilizes active learning to enable humans correct labels predicted from an unlabeled dataset or correct the explanation made by an XAI method named LIME [27]. LIME takes input and segments into an image where the target node or neuron is positively or negatively activated. Then, the dataset is extended with corrected labels and augmentations made from corrected explanations until the maximum number of iterations is reached or the model is optimal. Latter employ feedback to whether create new pathway [11] or delete pathways that could confuse or mislead the model [12]. FIND [12] is a NLP-based model debugging method that utilizes an XAI method named Layer-wise Relevance Propagation (LRP) [28] to understand the behavior of features and model's prediction. After compiling the most activated words or phrases for every feature from the results of LRP, they will be presented to humans to get their opinion on whether those features are relevant to the decision they make. If the feature clashes with the opinion of humans, the feature gets disabled and the model is retrained with a modified feature vector.

In this thesis, we propose a method that enables humans to integrate their knowledge by eliminating useless or harmful features of the feature vector extracted before being processed by the last classification layer of image classification models. This approach has a similar idea to FIND [12]; however, due to differences between text and image, our approach differs in explanation method where Activation Maximization [24, 25] and LIME is used to represent feature patterns. Also, while FIND decides how a feature supports final outputs by inspecting weights, in our approach, we used the contribution of features to the final layer as features could be negatively activated by input and still be positively contributed to the final nodes by negative weights.

1.2 The Outline of the Thesis

The thesis contains 5 chapters. In Chapter 1, brief descriptions of Feature Visualization, Neural Network debugging, and Explainable Artificial Intelligence are provided.

The shortcomings of existing methods are identified, and our contributions are summarized. In Chapter 2, background information is provided for the concepts used and mentioned throughout the thesis. Also, a literature review for Feature Visualization, Neural Network debugging, and Explainable Artificial Intelligence is provided. Convolutional Neural Network (CNN) debugging and analysis methods are discussed in terms of taxonomy and application areas. Image classification datasets and evaluation metrics are described. In Chapter 3, The architectural and training details of the baseline method used in this study are thoroughly explained. This chapter also explains the proposed novel approach. In Chapter 4, experimental results are provided. The analysis is performed for the dataset, obtaining human expert feedback, and integrating this feedback into the network to improve its decision-making ability. The baseline method and the proposed method are theoretically and practically compared. Finally, in Chapter 5, We provide a summary of the research done for this thesis, draw some conclusions, and talk about potential follow-up studies for this research. In Appendix A, we provide user study results gathered, visualizations provided at user study, and contributions to each class for every investigated feature. In Appendix B, confusion matrices for each trial are provided.

1.3 Contributions and Novelties

This thesis focuses on debugging methods for deep image classification algorithms. Our contributions are as follows:

- Experimenting with FIND [12] and modifying its supporting algorithms like explanations with LRP [28] and word clouds to see if it could be a viable algorithm for image classification while preserving its core idea.
- Using visualization of features via LIME [27] to present information to humans who interact with debugging since parameters themselves are hard to comprehend by only inspecting numerical values at feature maps. Therefore, a thorough visualization with examples and contributions by weight and biases is provided to debuggers to clarify any weight's or feature's purpose and contribution toward the final decision.

- A masking operation for weights or parameters for disabling them entirely to stop their irrelevant or adversarial contribution to the model's success. This helps the model to drop adverse features which confuse the network in the decision-making process.

CHAPTER 2

BACKGROUND INFORMATION AND LITERATURE REVIEW

2.1 Introduction

In this section, brief information is provided on neural network visualization and debugging. It is discussed how neural networks developed through time to become the powerful deep learning models we use today. Because it is crucial for network debugging, the neural network training method is explained. Convolutional neural networks are thoroughly discussed as they are employed in the experiments. The fundamental information of visualization is explained since visualization is used to portray learned features as familiar notions to humans with expert knowledge.

2.2 Brief History of Neural Networks

Part of a series on Machine Learning (ML) and Artificial Neural Networks (ANN) are computational models inspired by the way that biological neurons work. They have been around for the past 30 years, and have proven themselves to be a powerful tool for data processing in various fields including pattern recognition, forecasting, and optimization. The basis of their operation can be traced back to research done in 1943 by Princeton researchers Warren McCulloch and Walter Pitts, who described how to achieve a “logical” design that could mimic how neurons in the brain process information where it is called "threshold logic" with electrical circuits [29]. Hebbs extended their work in 1949 by suggesting that learning behavior can be replicated by repetitive iterations, which is known as hebbian learning [30]. Perceptron which can be seen in Figure 2.1 proposed by Frank Rosenblatt in 1958 [31] is seen as the first program to classify information using hebbian learning and advanced threshold logic circuit. In its first version which is named Mark 1 Perceptron had only 2 layers: input and output layer. Perceptron uses Equation 2.1 to calculate the result.

$$\mathbf{y} = \sum_{n=0}^{m} f(\omega_n * x_n) \tag{2.1}$$

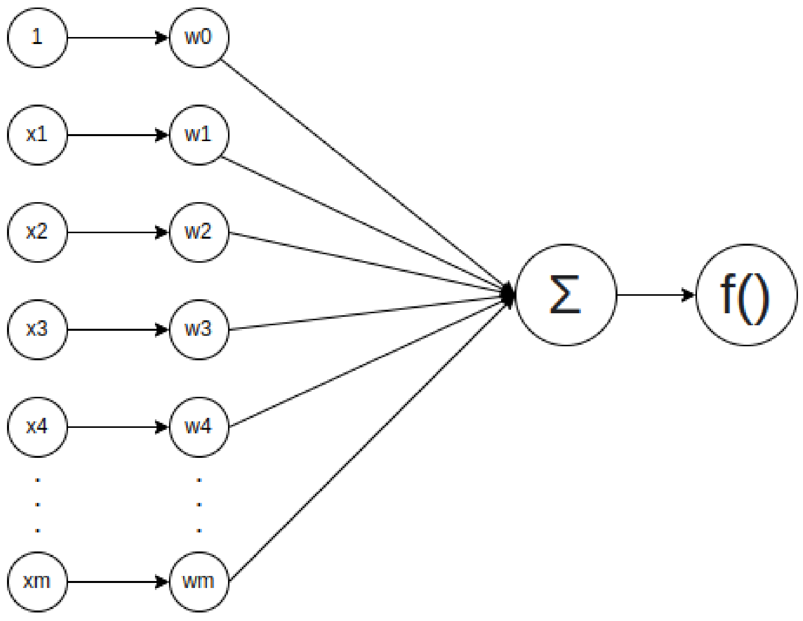


Figure 2.1: Perceptron illustration

Because NN models are hand-engineered and there is no viable backpropagation implementation, their development tumbles back and forth until the 1980s. The introduction of backpropagation and multi-layer perceptron gets the popularity of the field back at that time, however, it will never be as popular as today until 2012 with AlexNet [13] due to lack of data and computational power. As the internet is popularized and reachable by the public, while data becomes digitalized and structured, data being both personal and anonymous became available to researchers and companies which wanted to employ data-based algorithms. This excessive amount of data allowed researchers to split their data to generalize their decision-making algorithm for their use case. The computation problem is solved by the parallel processing power of the Graphics Processing Unit (GPU)s and its single instruction multiple dataset (SIMD) ability due to processing graphics. After the popularization of GPUs, there were other solutions such as neural network accelerators and TPU (Tensor Processing Unit)s inspired by parallel computation and tailored for neural network units.

2.3 Training Neural Networks

Neural networks should have their parameters tuned for the task they are tailored for. To achieve this, there is a learning process with two different steps: forward propagation and backpropagation. The first one is the same as the decision-making process. Input is fed to the model and it goes through the layers and modules the model has in an intended way while inferring the output of the model with the output of the final layer or combination of layers depending on the model architecture. In backpropagation, the error from the output which calculates the loss function whether it is from ground truth in the supervised learning or reward function in reinforcement learning and so on is propagated through the input. There are many loss functions for many purposes such as Mean Squared Error (MSE) which can be represented in Equation 2.2 for regression, Binary Cross Entropy (BCE) which can be represented in Equation 2.3 for binary classification or Negative Log Likelihood (NLL) which can be represented in Equation 2.4 for multiclass classification.

$$MSE = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{2.2}$$

$$BCE = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y})) \tag{2.3}$$

$$NLL(y) = -\text{log}(p(y)) \tag{2.4}$$

Once the loss function is chosen for the appropriate algorithm, it is propagated by taking its gradient throughout from the last layer to the first layer with respect to each parameter such as weight and bias residing within the model. The gradient can be seen in Equation 2.5 where J is the cost function and ω is the parameter in the model for the gradient. For multi-layered networks gradient works in chain rule as can be seen in Equation 2.6. This step shows how much each parameter in each module in the model is needed to change to achieve learning.

$$\nabla\omega_m = \frac{\delta J}{\delta\omega_m} \tag{2.5}$$

$$\nabla\omega_m = \frac{\delta J}{\delta\omega_{final}}\frac{\delta\omega_{final}}{\delta\omega_{final-1}}...\frac{\delta\omega_{m+1}}{\delta\omega_m} \tag{2.6}$$

To actually achieve an iterative learning process, there are many optimizers that try to modify parameters in each step to improve the model. The most common examples are Stochastic Gradient Descent (SGD) which can be represented in Equation 2.7 where η is learning rate, Momentum [32] which can be represented in Equation 2.8 where ν is a gradient from the previous iteration and γ is momentum parameter which is used to control residual gradient, Adaptive Momentum (Adam) [33] which can be represented in Equation 2.9 where m and ν are moving averages of gradient and squared gradient, β parameters are hyper-parameters for controlling moving average amount.

$$\omega_t = \omega_t - \eta\nabla\omega_t \tag{2.7}$$

$$\begin{aligned} \nu_t &= \gamma\nu_t - \eta\nabla\omega_t \\ \omega_{t+1} &= \omega_t - \nu_t \end{aligned} \tag{2.8}$$

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1)\nabla\omega_t \\ \nu_t &= \beta_2\nu_{t-1} + (1-\beta_2)\nabla\omega_t^2 \\ \hat{m_t} &= \frac{m_t}{1-\beta_1^t} \\ \hat{\nu_t} &= \frac{\nu_t}{1-\beta_2^t} \\ \omega_t + 1 &= \omega_t - \eta\frac{\hat{m_t}}{\sqrt{\hat{v_t}}+\epsilon} \end{aligned} \tag{2.9}$$

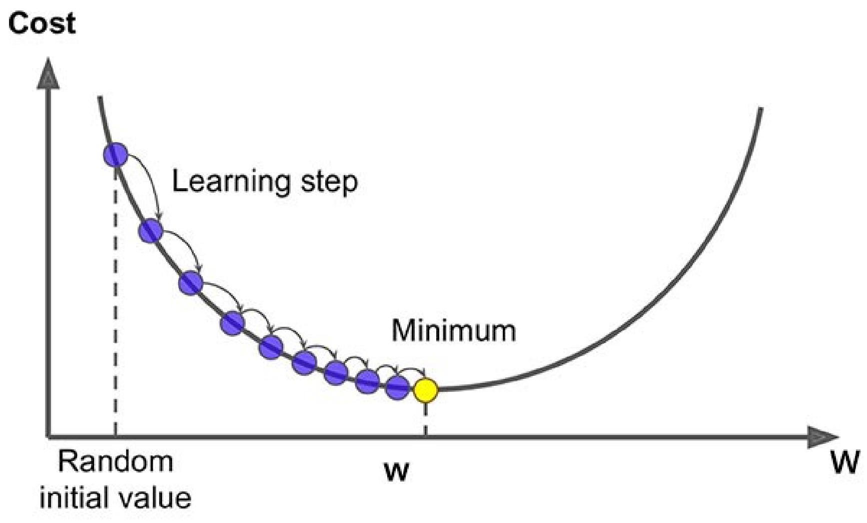


Figure 2.2: Gradient Descent Visualized [1]

The optimizer updates the parameters of the model with their function for each step scaled with a predefined value named learning rate. The learning rate is a hyperparameter that defines how fast or slow a parameter is updated during training in the backpropagation phase. Learning with gradient descent can be seen in Figure 2.2.

2.4 Visualizing Learned Features

Deep learning neural networks are generally opaque, which means that while they can make useful and skillful predictions, it is unclear how or why a particular prediction was made. With the success of neural networks and their success due to empirical methods they employ, the need for investigating neural networks by their learned features has grown further. There are early methods where intermediate network layer outputs are extracted to show what each feature in each layer actually represents.

2.4.1 Activation Maximization

Activation Maximization [24] is a method to generate artificial input that maximizes the activation of a specific neuron in a specific layer of a neural network. The optimization problem is expressed in Equation 2.10 where v is the input, $\hat{v}$ is generated input, $a(\cdot)$ is the activated parameter function which can be whole end-to-end neural network or a part of it. It is possible to use gradient-based [34] or non-gradient [35] approaches to try to identify a local maximum for this non-convex optimization issue. Applying a gradient ascent [24, 5, 36, 37] for maximization with an update rule is a straightforward strategy which is given in Equation 2.11 where t denotes iteration count, v_t denotes image generated until iteration t, γ denotes step size. This gradient ascent is very similar to the gradient descent that is widely used in deep neural network optimization on the backpropagation step. However, in this case, we are applying an optimization step to input with respect to the model's output value.

$$\hat{v} = \underset{v \epsilon \mathbb{R}^v}{\mathbf{argmax}}\, a(v) \tag{2.10}$$

$$v_{t+1} = v_t + \gamma \frac{\partial a(v_t)}{\partial v_t} \tag{2.11}$$

While this method generates images as can be seen in Figure 2.3 and 2.4, this interpretation can be incomprehensible to humans as it includes too much high-frequency information for the human mind to perceive the concept.

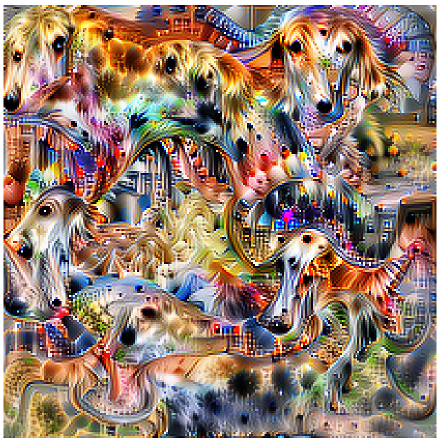


Figure 2.3: Activation maximization example with respect to class "Saluki, gazelle hound" from ImageNet-1K optimized to model VGG-16 [2]

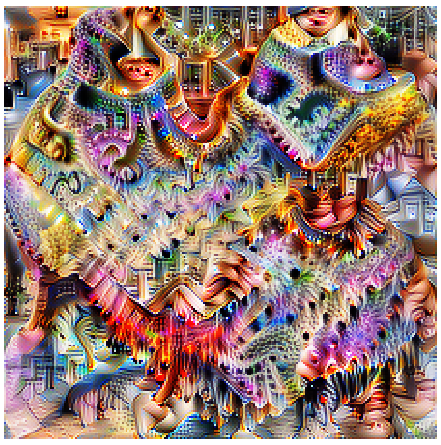


Figure 2.4: Activation maximization example with respect to class "Poncho" from ImageNet-1K optimized to the final layer of model VGG-16 [2]

To get more interpretable results, Compositional Pattern Producing Networks (CPPN) [38] can be added as an additional constraint for visualization as can be seen in Figure 2.5. CPPN inputs latent features and pixel position to produce a scalar value for the generated image's inputted coordinates. These networks can be combined with the Generator-Discriminator concept of GANs to generate interpretable images [4]. The weights and biases of the CPPN network determine the image produced. Depending on the architecture chosen for the CPPN, pixels in the resulting image are required to share, to some constraint, the color of their neighbors [3]. An example can be seen for the same gazelle hound in Figure 2.6 that visuals are not high frequency and they are more interpretable for humans to understand learned features. This process requires training a new network for generating the image which introduces more black box modules to the overall system; therefore, this is not a preferred method in this thesis.

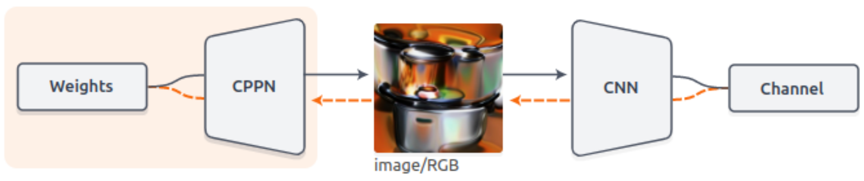


Figure 2.5: CPPN integrated into CNN network for visualization [3, 4]

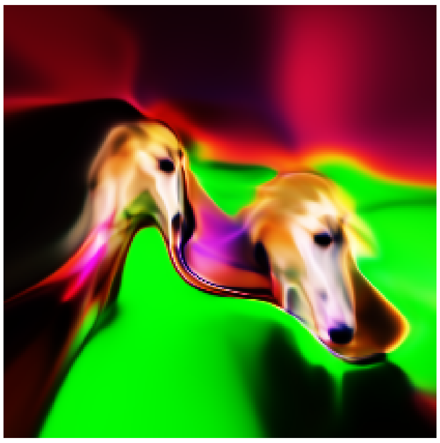


Figure 2.6: Activation maximization example with respect to class "Saluki, gazelle hound" from ImageNet-1K optimized to model VGG-16 optimized using CPPN model as constraint

Similar to CPPN [38], Generative networks can be utilized for synthesizing images that can activate images for better comprehension such as DGN-AM [5] and PPGN [39]. The architecture for synthesizing can be seen in Figure 2.7 which is called Deep Generative Network for Activation Maximization (DGN-AM). The formula for optimization can be seen in Equation 2.12 where y^l is the entire layer to generate an image from viable code, Φ is the target network, h is the target neuron, G_l is the generator for the target layer, λ is regularization parameter. This procedure is costly due to training different generator networks required for every single layer that needs to be visualized and because they are another neural network trained, similar to the CPPN case, this arises more black box concept than explainability; therefore, it is not preferred in this thesis.

$$\hat{v} = \underset{y^l}{\mathbf{argmax}}\, \Phi_h(G_l(y^l) - \lambda \|y^l\|) \tag{2.12}$$

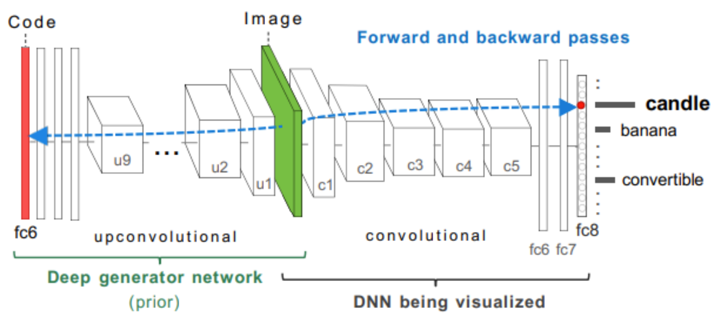


Figure 2.7: Prior generative network (DGN-AM) proposed by A. Nguyen [5] for visualizing DL model

2.4.2 Explainable Artificial Intelligence

Since the beginning of AI research, scientists have argued that intelligent systems should explain the AI results, particularly when it comes to decisions [40]. Users also have the right to demand an explanation for decisions according to European Union's AI Act [41] which states every AI that breaches the fundamental rights and safety of data subjects (users in other terms) that is classified as high-risk AI should have a certain degree of transparency.

To achieve this transparency for the inner workings of black box explainable artificial intelligence, Charlisle Scott et al. [6] set guidelines for explanations of decision-making production systems in 1977 and a diagram for prototype explanation system as can be seen in Figure 2.8.

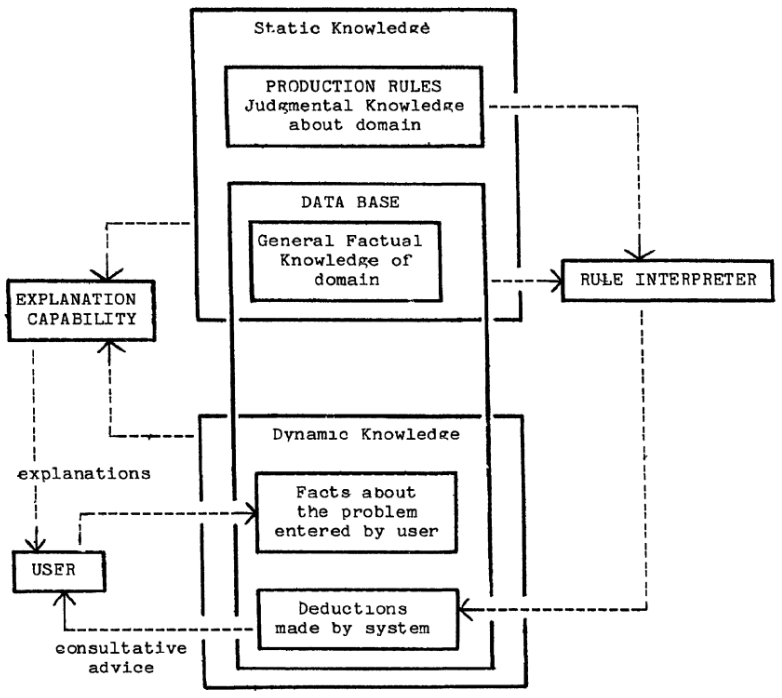

Static Knowledge
PRODUCTION RULES
Judgmental Knowledge about domain
DATA BASE
General Factual Knowledge of domain
RULE INTERPRETER
EXPLANATION CAPABILITY
Dynamic Knowledge
explanations
Facts about the problem entered by user
USER
Deductions made by system
consultative advice

Figure 2.8: A Production-Based Consultation System with Explanation Capability proposed by Scott et al [6]

After seeds of interpretation are planted with these guidelines, a field emerged after 40 years when Deep Learning became more successful, popularized, and used by both researchers to improve it and companies to provide services in several areas. Without explanations, modern artificial intelligence systems are completely black boxes that are only inspired by classical methods for working principles and nothing else meaning whose internal inference processes are unknown to the observer and cannot be interpreted by humans. Therefore, the network and its output can not be explained to the developer or user.

This paved the way for Explainable Artificial Intelligence [6, 42, 43] field where the purpose is to find a way to inspect the internals of black box systems or make networks transparent and interpretable so that external observers can comprehend network's decision making process. XAI can be categorized in three terms [7]: scope (global [44, 45] and local [26, 46]), methodology (backpropagation-based [47, 48, 49, 50, 46] and perturbation-based [27]), usage (post-hoc [47, 48, 49] and model-intrinsic

[45, 44, 51]) as can be seen in Figure 2.9.

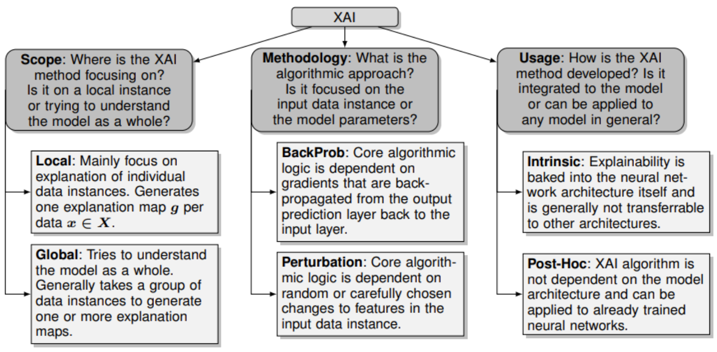

XAI
Scope: Where is the XAI method focusing on? Is it on a local instance or trying to understand the model as a whole?
Methodology: What is the algorithmic approach? Is it focused on the input data instance or the model parameters?
Usage: How is the XAI method developed? Is it integrated to the model or can be applied to any model in general?
Local: Mainly focus on explanation of individual data instances. Generates one explanation map g per data $x \in X$.
Global: Tries to understand the model as a whole. Generally takes a group of data instances to generate one or more explanation maps.
BackProb: Core algorithmic logic is dependent on gradients that are back-propagated from the output prediction layer back to the input layer.
Perturbation: Core algorithmic logic is dependent on random or carefully chosen changes to features in the input data instance.
Intrinsic: Explainability is baked into the neural network architecture itself and is generally not transferrable to other architectures.
Post-Hoc: XAI algorithm is not dependent on the model architecture and can be applied to already trained neural networks.

Figure 2.9: A general categorization of XAI taken from [7]

2.4.3 Concept Bottleneck Models and Self-Explaining Neural Networks

After Explainable AI achieved significant success in justifying decisions for neural networks by highlighting parts of input or tracing paths through the output, the need for interpretable networks has arisen. Among these networks, there is one that is named Self-Explaining Neural Networks [8] which tries to achieve creating unsupervised concepts for inputs before making the final decision. These networks encode the information into a small set of explainable features as can be seen in Figure 2.10.

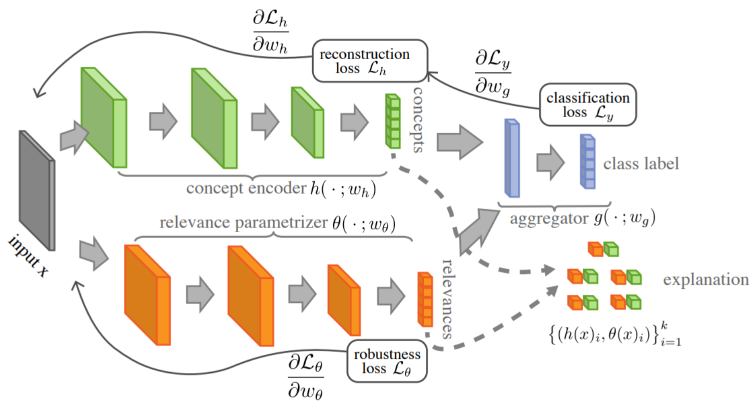

$\frac{\partial \mathcal{L}_h}{\partial w_h}$
reconstruction loss $\mathcal{L}_h$
$\frac{\partial \mathcal{L}_y}{\partial w_g}$
classification loss $\mathcal{L}_y$
concepts
class label
concept encoder $h(\cdot\,;w_h)$
aggregator $g(\cdot\,;w_g)$
relevance parametrizer $\theta(\cdot\,;w_\theta)$
input x
relevances
explanation
$\{(h(x)_i, \theta(x)_i)\}_{i=1}^k$
robustness loss $\mathcal{L}_\theta$
$\frac{\partial \mathcal{L}_\theta}{\partial w_\theta}$

Figure 2.10: A general architecture of Self Explaining Neural Networks (SENN) [8]

While unsupervised self-explaining networks give results by combining their concepts and providing examples while not giving explicit explanations, Concept Bottleneck Models [9] try to learn pre-engineered features to learn and judge using disentangled features as can be seen in Figure 2.11. These models also enable external intervention on intermediatory parameters called concepts and enable models to be more interpretable while most other SotA NNs are end-to-end and difficult to interpret. While this method can be viable due to an intrinsic set of parameters called concepts, they add another level of supervision and labeling. Therefore, they are not used in this thesis.

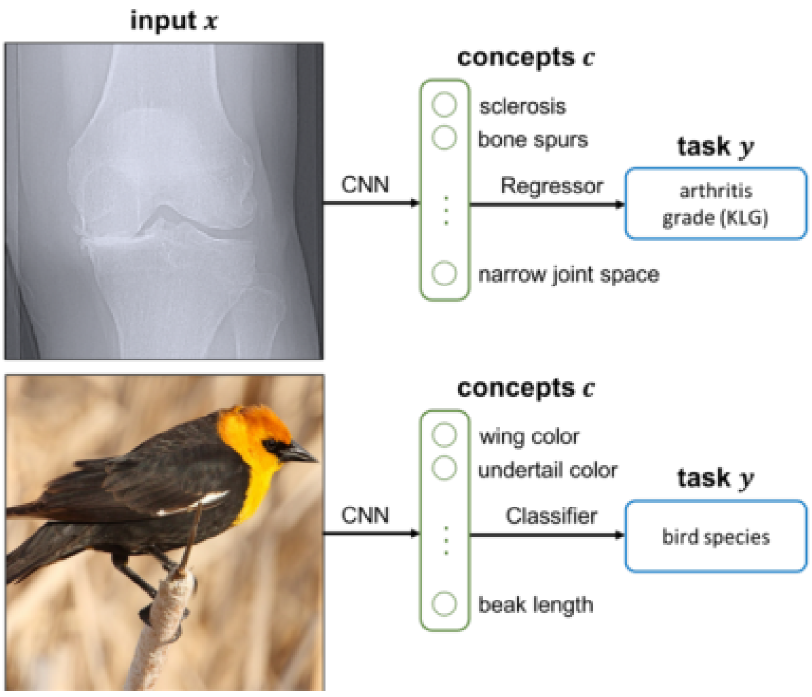


Figure 2.11: A general working principle of Concept Bottleneck Models (CBM) [9]

2.5 Model Debugging

With the growing success of NNs arrived a growing need to expand the research into different areas. This expansion caused the need for troubleshooting. In the earliest phases, machine learning parameter-based problems could have been solved by identifying basic problems such as over-fitting or under-fitting, bias or variance-based problems, or learning-based problems where the learning rate becomes so steep it steers gradients into unlearning some of the training. These problems could have been solved with general methods such as the regularization of weights in updates (L1 in Equation 2.13, L2 in Equation 2.14 regularization), canceling random weights in layers to train others every iteration with Dropout [52] or using learning rate schedulers such as Cosine Annealing [53] where learning rate η_t decays with cosine function modulation $cos(\frac{T_{cur}}{T_i}\pi)$ as can be seen in Equation 2.15

$$MSE_{L1} = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^{m} \|\omega_j\| \tag{2.13}$$

$$MSE_{L2} = \sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \lambda \sum_{j=0}^{m} \omega_j^2 \tag{2.14}$$

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + cos(\frac{T_{cur}}{T_i}\pi)) \tag{2.15}$$

With the emergence of XAI [6], model debugging expanded its methods to include humans in the process which is called Explanatory Interactive Learning(XIL) [22]. This method combines active learning with XAI and uses LIME [27] to explain inference results for unlabelled images and correct results, explanation, or both if necessary. Then, the model continues training until the model is accurate enough or the max iteration number is reached. There are 3 different feedback types in XIL:

- **Right for the right reasons:** The prediction and the explanation are both correct. No feedback is requested.
- **Wrong for the wrong reasons:** The prediction is wrong. As in active learning, we ask the user to provide the correct label. The explanation is also necessarily wrong, but we currently do not require the user to act on it.
- **Right for the wrong reasons:** The prediction is correct but the explanation is wrong. We ask the user to provide an explanation correction C.

This operation is costly due to generating counterexamples and storing them in training. To overcome this problem, an improved version of XIL [23] uses GRAD-CAM [49] instead of LIME [27] for explanations and focuses on RRR(Right for right reasons) type of feedback in original XIL [22] paper. Change of explanation method from perturbation-based to backpropagation-based method gets rid of extra operation caused by sampling superpixels. Also choosing only RR relevant and other decisions irrelevant; therefore punishing weights for non-contributing in training makes training faster rather than analyzing and manipulating data as in the original paper

For Vision-Language models, there is a method called HINT(Human Importance-aware Network Tuning) [10] which tries to remove the bias of explanations by using human feedback which is a heatmap to reflect explanation to possible output to tune VQA(Visual Question Answering) [54] as can be seen in Figure 2.12. This system uses ranking loss to reward the model if the annotator and model agree on the explanation of the result.

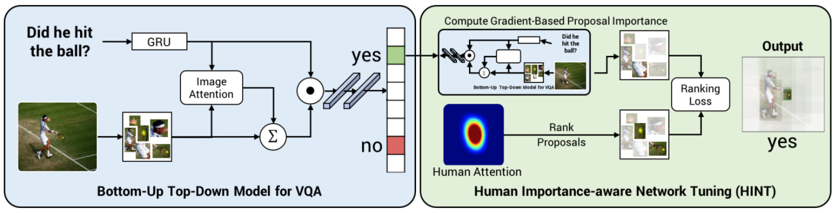

Did he hit the ball?
GRU
Image Attention
Σ
yes
no
Bottom-Up Top-Down Model for VQA
Compute Gradient-Based Proposal Importance
Ranking Loss
Rank Proposals
Human Attention
Output
yes
Human Importance-aware Network Tuning (HINT)

Figure 2.12: Working diagram of HINT system taken from [10]

ALICE (Active Learning with Contrastive Explanations) [11] relies on contrasting explanations while facilitating Active Learning [55]. For each interaction round, the machine chooses a small number of class pairs—often those that the model does not discriminate well—and requests that an annotator describes in normal language the characteristics that allow them to tell the two classes apart. The feedback is then transformed into rules via semantic parsing, and integrated by "morphing" the model architecture appropriately as can be seen in Figure 2.13.

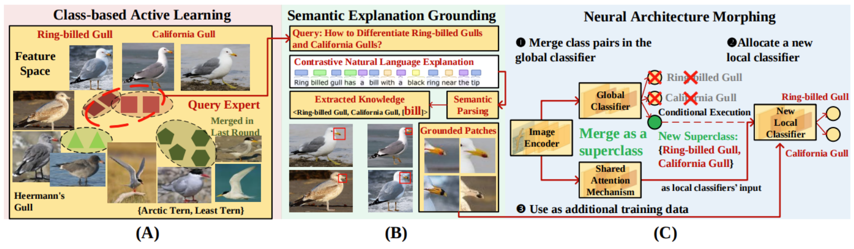

Class-based Active Learning
Ring-billed Gull
California Gull
Feature Space
Query Expert
Merged in Last Round
Heermann's Gull
{Arctic Tern, Least Tern}
(A)
Semantic Explanation Grounding
Query: How to Differentiate Ring-billed Gulls and California Gulls?
Contrastive Natural Language Explanation
Ring billed gull has a bill with a black ring near the tip
Extracted Knowledge
<Ring-billed Gull, California Gull, {bill}>
Semantic Parsing
Grounded Patches
(B)
Neural Architecture Morphing
❶ Merge class pairs in the global classifier
❷ Allocate a new local classifier
Ring-billed Gull
California Gull
Global Classifier
Image Encoder
Merge as a superclass
Conditional Execution
New Superclass: {Ring-billed Gull, California Gull}
New Local Classifier
Ring-billed Gull
California Gull
Shared Attention Mechanism
as local classifiers' input
❸ Use as additional training data
(C)

Figure 2.13: Working diagram of ALICE system taken from [11]

There is another model focused on NLPs mainly to inspect latent features after the feature extraction process called FIND (Feature Inspection aNd Disabling) [12]. This framework extracts word clouds from latent features as it ranks words or short phrases comprised of 2 or 3 words. Then these word clouds are presented to the user or expert to let them decide whether this feature is relevant to the decision made by the model or not. If it is irrelevant, the feature is completely disabled by masking the weights and disabling connections to the final layer. Then, the model is trained with masked weights ensuring that those layers are not included by both training and inference operations. The overall process can be seen in Figure 2.14

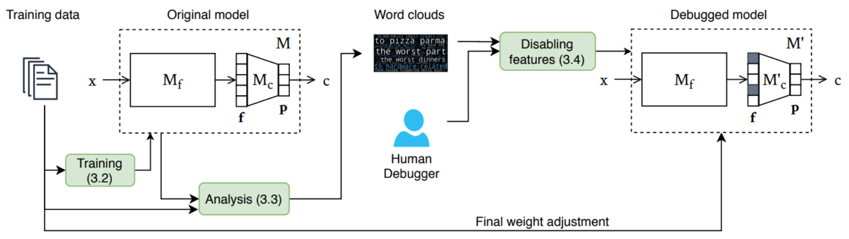


Figure 2.14: Working diagram of FIND taken from [12]

CHAPTER 3

PROPOSED METHODOLOGY

3.1 Motivation

In this chapter, we will give the details of the proposed methodology to debug the image classification algorithms using human feedback for disabling irrelevant features. If we are aware of the patterns or attributes of the input that each feature captures, we may appreciate the model's overall functionality since the classification part's linear layer then becomes understandable. This is made possible by utilizing Activation Maximization and Explainable AI method called LIME [27] in this thesis. Humans can determine whether each feature's discovered input patterns are pertinent for categorization by inspecting the explanation of the model. In order for the following linear layer to support the appropriate classes, the features should be attributed with correct weights per classes that they represent. If those features are not supporting the classes correctly, debugging can be carried out by turning off any features that the model may have that could be misleading. This methodology answers 2 questions:

- How to interpret and visualize latent features?
- How to match human knowledge and the model's explanation for the final decision and improve the model's training and fine-tuning?

Generally, deep image classifiers are comprised of two parts: feature extractors which perform identifying and transforming image data to feature vectors which store meaningful information about the image and are comprised of either fully connected layers, convolutional layers, or a newly shining field, transformers, and classifiers which uses feature vectors to a fully connected layer with a non-linear activation which is preferably softmax to get predictions and their respective probabilities. This algorithm is not a see-through, transparent algorithm by itself where anyone can peek through and see what the feature vector represents. Therefore, they can not be understood without the help of XAI and even if we get a glimpse of what this means, it can not be modified directly to give the perfect result, only guided through correct answers. However, with the usage of human knowledge and feature interpretation using XAI, and fusing this knowledge to guide the model to become better or understand the importance of features, we could make gray-box models' working principle more comprehensible.

Table 3.1: Table of Notations

Notation	Description
d	number of features in feature vector
N	number of classes model outputs
D	Feature Disabling Matrix with the shape of dxd
W	Weight of M_c with shape of Nxd
x	Input
M	Model
M_f	Feature Extraction part of model
M_c	Classification part of model
M_c'	Classification part of the model, modified after fine-tuning
F	Feature vector comprised of d features, the output of M_f
f_i	i^{th} feature instance from F = $[f_0, f_1, ..., f_i, ..., f_{d-1}]$

3.2 Algorithm

A three-stage algorithm as can be seen in Figure 3.1 is introduced to address and answer the questions above which is similar to FIND [12] framework.

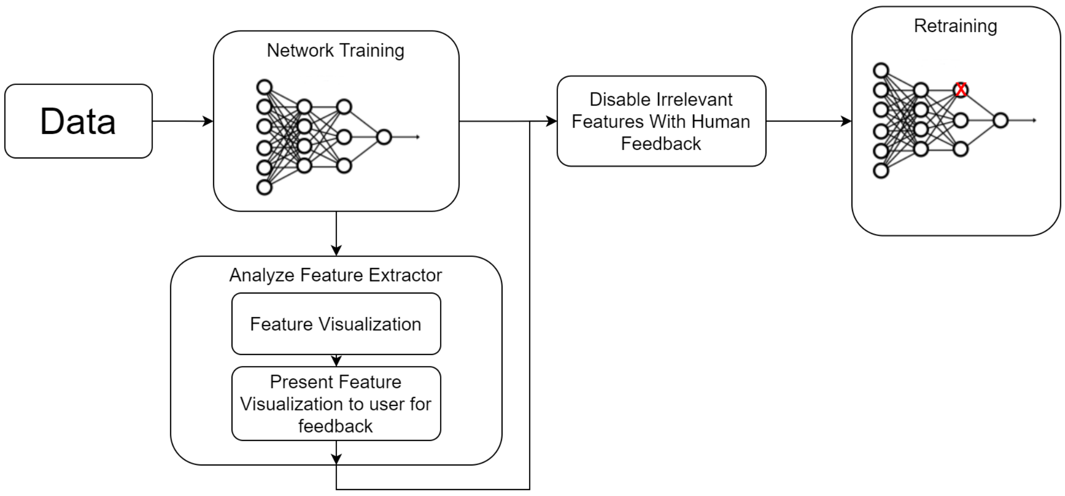

Data
Network Training
Disable Irrelevant Features With Human Feedback
Retraining
Analyze Feature Extractor
Feature Visualization
Present Feature Visualization to user for feedback

Figure 3.1: Proposed System

3.2.1 Training Network

The first one is the learning phase which is described in Chapter 2 Section 2.3, training a convolutional neural network for image classification whose notation will be M with the model formulation in Equation 3.1 where all notation can be seen in Table 3.1 from scratch or using transfer learning.

$$M(x) = M_c(M_f(x)) = \hat{y} \tag{3.1}$$

3.2.2 Interpreting Features

For the second part of the algorithm, the feature extracting part (M_f) is isolated from the model as in Equation 3.1 and visualized using Activation Maximization [24] or LIME [27] for each feature f_i calculated at feature extracting part as in Equation 3.2, the entire validation dataset is scanned to find images that activate the features the most and visualized weights to see that which class this feature actually contributes the most as can be seen in Figure 3.4. For LIME, we sample a limited number of inputs that activates feature highly and contributes positively to the final decision. Therefore for each feature f_i, a subset of the validation set denoted as x has run through and collaged into an image as can be seen in Figure 3.2

$$M_f(x) = F \tag{3.2}$$

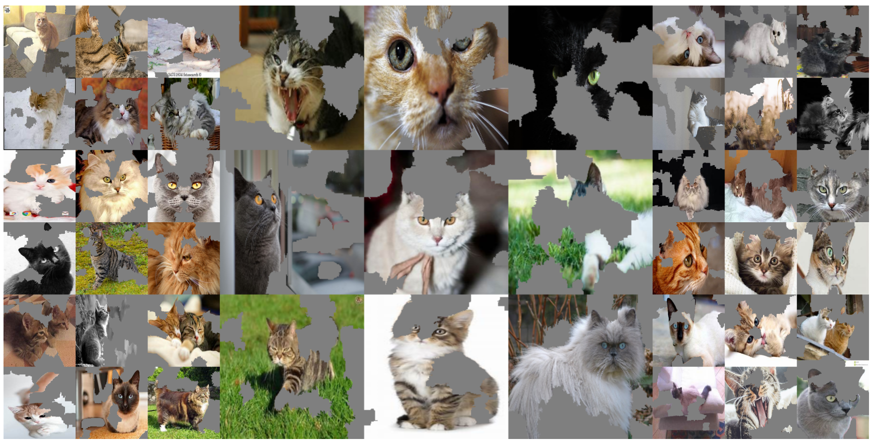
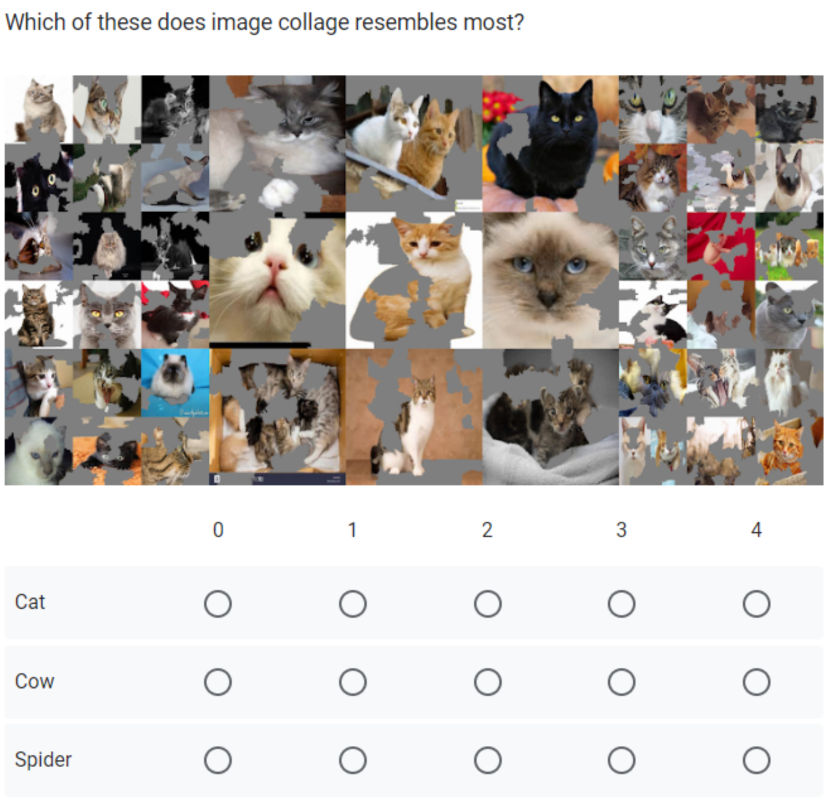


Figure 3.2: Visualization of a feature with LIME

3.2.3 Disabling Features

The investigation of features is done by a human expert or experts about whether the feature is useful or misguiding to the classification part by presenting a visualization of each feature f_i and collecting responses on features about which class they represent and how likely to the target class it is as can be seen in Figure 3.3 and disabling part is done according to results. For disabling part, if survey results clash with that of the model itself, that means the model is lacking to give good decisions on that feature, therefore it should be disabled. For example, consider feature representation in Figure 3.2 that clearly represents "Cat" class may be classified as one of the other classes in the classification part. That means this feature misleads the classification part to misclassify this. Therefore if it is removed, the model will be guided to make better decisions for classifying instances of images. To establish disabling part, we use a disabling matrix D and $D \in \mathbb{R}^{Nxd}$ where d is the number of features and N is the number of classes that model M classifies. To disable a feature f_i, i^{th} column of D matrix is set to zero. Disable matrix D masks the final layer's weight W is a matrix of shape Nxd where d is the number of features in the feature vector which is the output of M_f. The model is retrained with disabled weights as in Equation 3.3 to fine-tune the model. At this phase, M_f's parameters are frozen completely to only change parameters of M_c.

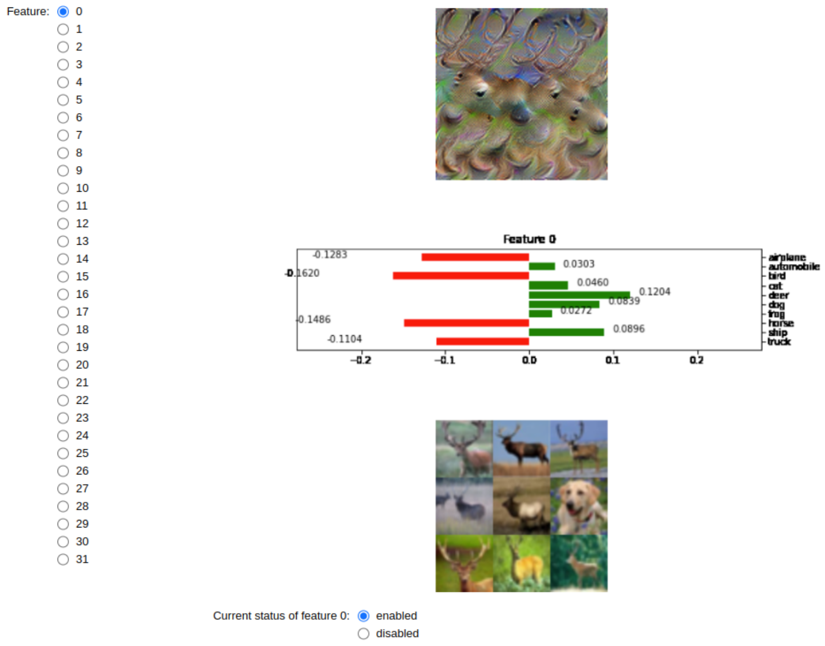
$$M'(x) = M'_c(F) = softmax((W \odot D) * F + b) \tag{3.3}$$



Which of these does image collage resembles most?

	0	1	2	3	4
Cat	○	○	○	○	○
Cow	○	○	○	○	○
Spider	○	○	○	○	○

Figure 3.3: Survey question conducted to receive opinions about what class represents, 0-Not representing at all, 4-Truly represents the class



Feature: [✓ 0] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 0: [✓ enabled] disabled

Figure 3.4: Feature Visualization and disabling screen presented to user, with feature vector on left, Activation Maximization result, weight visualization per feature, maximum activated images on right, feature disabling option at bottom

CHAPTER 4

EXPERIMENTAL RESULTS

4.1 Implementation Details

The models are trained and fine-tuned on a PC with NVIDIA GeForce RTX 3070 GPU with 8 GB memory and Intel Core i7-11800H CPU.

4.2 Datasets

We used CIFAR-10 [56] for initial experimentation due to their diverse classes which can be distinguishable when features are visualized. CIFAR includes 32x32 RGB images that have 6000 training and 1000 test images per class with a total of 60000 training images and 10000 testing images. CIFAR-10 has 10 classes which are Airplanes, Cars, Birds, Cats, Deer, Dogs, Frogs, Horses, Ships, and Trucks.

For more experimentation and better visuals, the dataset is switched to Animal-10 [57] which has medium-quality images which have higher resolution than CIFAR-10 [56]. This new dataset has 10 classes which are Butterfly, Cat, Chicken, Cow, Dog, Elephant, Horse, Sheep, Spider, and Squirrel.

4.3 Base Architecture

For experiments modified version of ResNet-18 [58] is used. Feature extractor part before Global Average Pooling [59] layer is cut off and three more layers added as can be seen in Table 4.1 and Figure 4.1. This modification allowed for better investigation of features due to the fact that there are less number of features to investigate rather than having 512 features, we have 32 to analyze and disable if necessary. The model is trained for 15 epochs with batch size 64 and learning rate $\eta = 0.001$.

Table 4.1: Model Architecture

Layer	Properties
$conv2d$	[7,7,64], stride 2
$maxpool2d$	[2,2], stride 2
$conv2d$	[3,3,64]
$conv2d$	[3,3,64]
$conv2d$	[3,3,64]
$conv2d$	[3,3,64]
$conv2d$	[3,3,128], stride 2
$conv2d$	[3,3,128]
$conv2d$	[3,3,128]
$conv2d$	[3,3,128]
$conv2d$	[3,3,256], stride 2
$conv2d$	[3,3,256]
$conv2d$	[3,3,256]
$conv2d$	[3,3,256]
$conv2d$	[3,3,512], stride 2
$conv2d$	[3,3,512]
$conv2d$	[3,3,512]
$conv2d$	[3,3,512]
$conv2d$	[1,1,32]
$globalavgpool$	[1,1,32]
$fclayer$	[32,10]

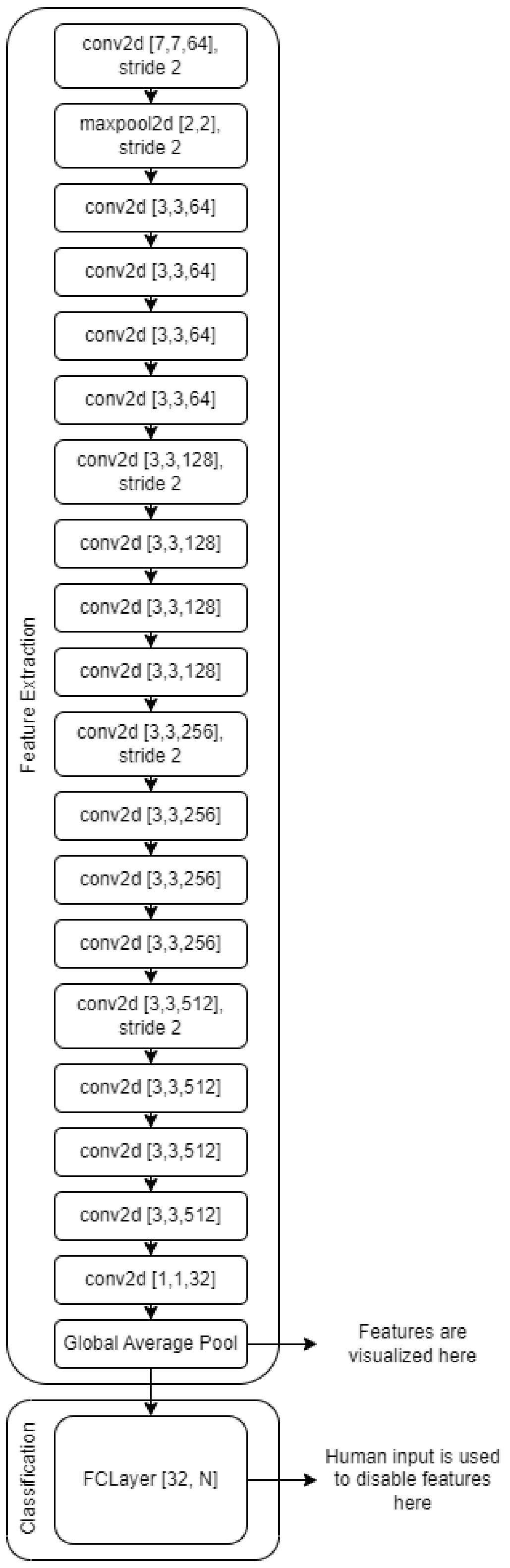


Figure 4.1: Model Architecture with parts indicated where features are visualized and where human input is utilized

4.4 Experiment 1: Feature Disabling and Retraining with CIFAR-10

For this experiment, we trained custom ResNet-18 with CIFAR-10 inputs. To get better results on feature visualization, images are resized from 32x32 to 128x128 before being fed to the model. This proved to give better results as the resolution for input is increased learned features could be better represented as can be seen in Figure 4.2

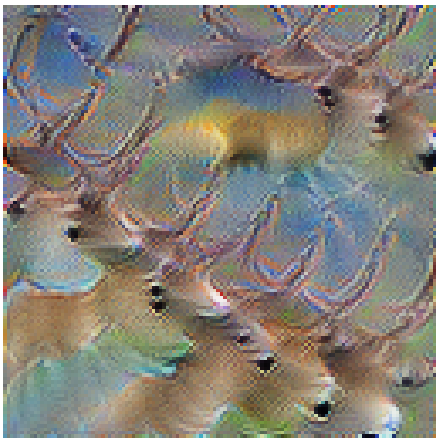
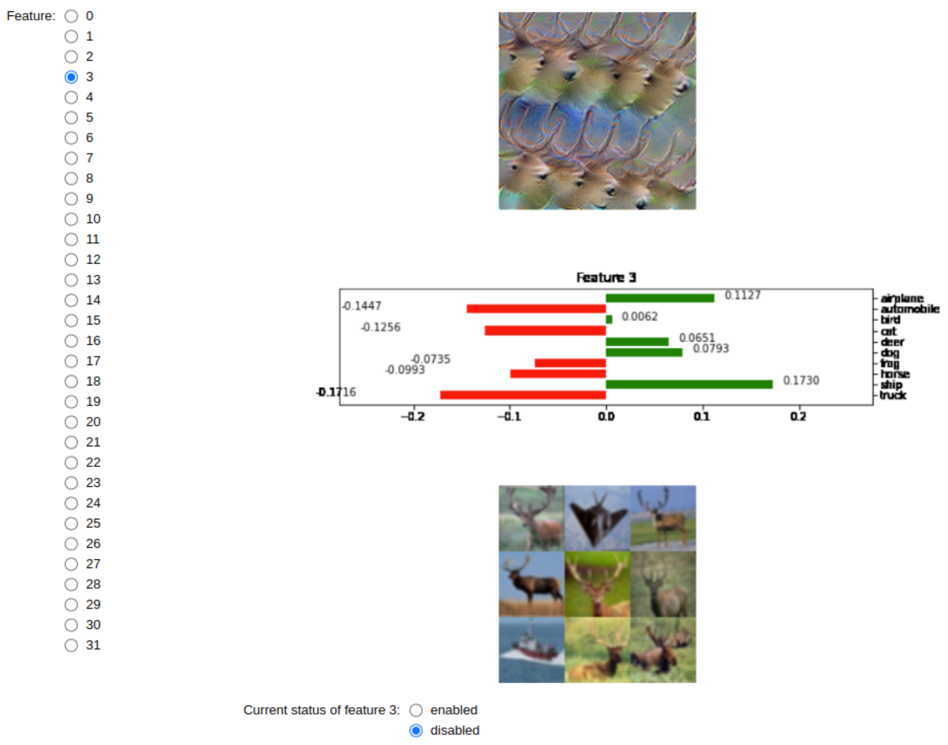


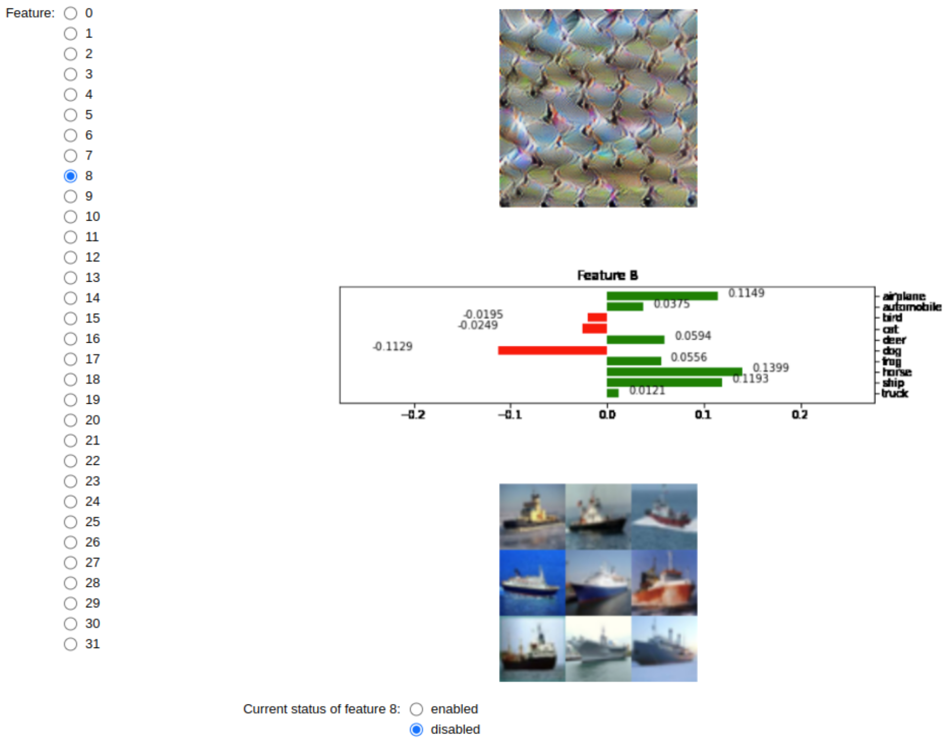
Figure 4.2: Feature Visualization taken from f_3

We tried a mostly data-agnostic approach by using class weights per feature and visualizing features using the Activation Maximization [24] method to get positive parts that contribute to the feature itself. After careful investigation, features are disabled regarding their visualization, most activating images in the dataset, and their contribution to the final layer by looking at the weight graph on whether they are misleading or not. An example of a feature disabling screen with assisting data can be seen in Figure 4.3. Disabled features in this case are indexed as $[3, 8, 11, 12, 13]$ out of 32 layers. Misleading examples can be seen in Figures 4.3, 4.4, 4.5, 4.6, 4.7. Misleading examples can be investigated in 2 categories: misattribution to a different class, and ambiguous features. Misattribution is when a feature does not contribute to the class as intended whether it negatively attributes to it or some other class attributes more than the class feature represents as can be seen in Figures 4.3, 4.6, 4.7. Ambiguity is caused when feature visualization is not interpretable and most contributed images are not represented in the weight plot as can be seen in Features 4.4, 4.5.


Feature: 0 1 2 [✓ 3] 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 3: enabled [✓ disabled]

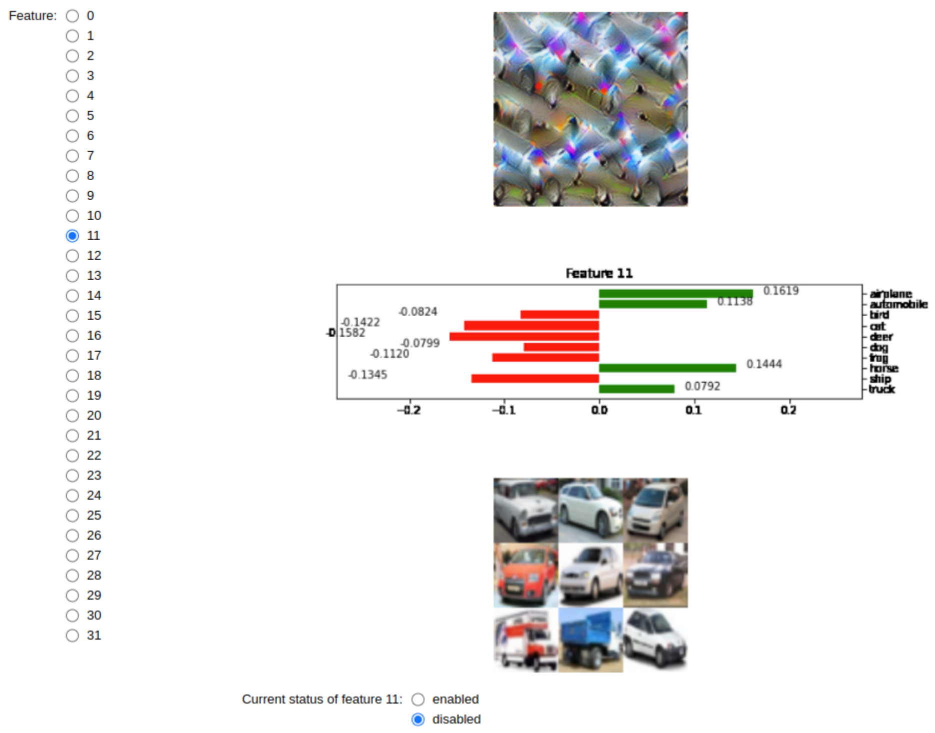
Figure 4.3: Feature Visualization taken from f_3: Problem with this layer is it is clearly attributed as deer, however, it mostly contributes to "ship" class.



Feature: 0 1 2 3 4 5 6 7 [✓ 8] 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 8: enabled [✓ disabled]

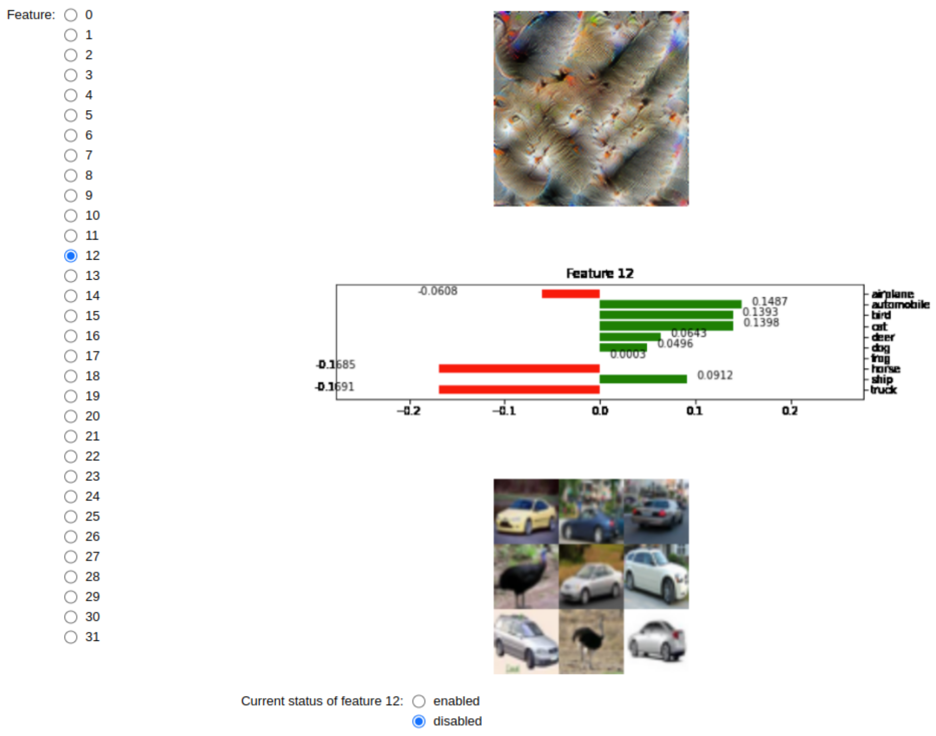
Figure 4.4: Feature Visualization taken from f_8: Problem with this layer is it is not clear to deduce which class it belongs to from FV and contributions are mostly to "horse" class while maximum activations are all "ships".



Feature: 0 1 2 3 4 5 6 7 8 9 10 [✓ 11] 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 11: enabled [✓ disabled]

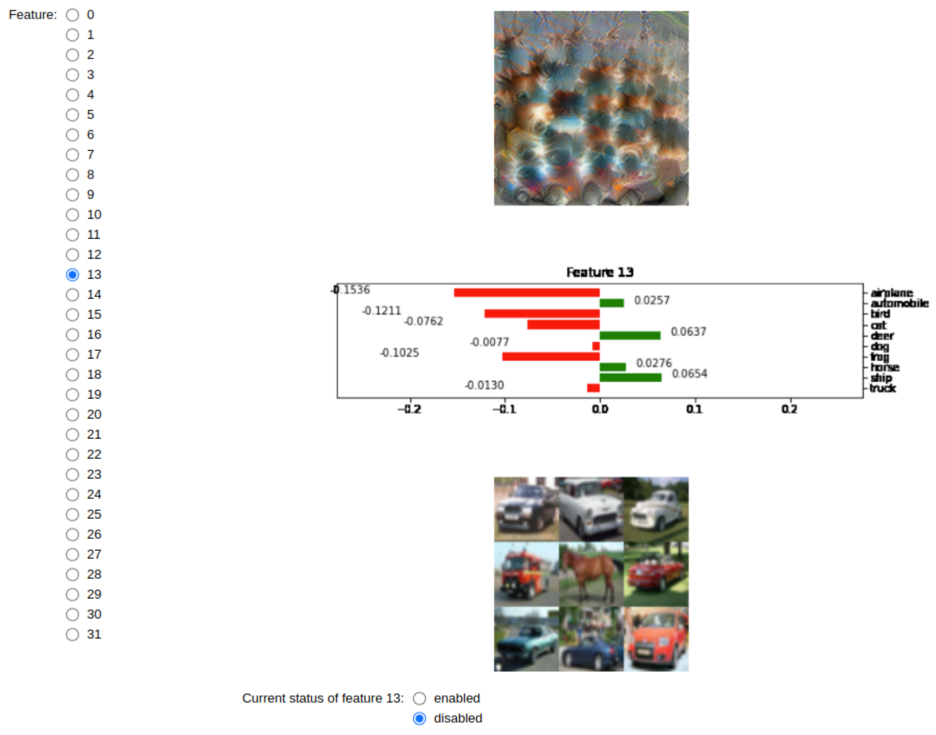
Figure 4.5: Feature Visualization taken from f_{11}: Problem with this layer is it is not clear to deduce which class it belongs to from FV and contributions are mostly to "horse" class while maximum activations are all "automobiles".



Feature: 0 1 2 3 4 5 6 7 8 9 10 11 [✓ 12] 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 12: enabled [✓ disabled]

Figure 4.6: Feature Visualization taken from f_{12}: Problem with this layer is it is clearly attributed as "cats" while most activated images are all "cars" and the contributions are contested for 3 different classes. Therefore this feature is misattributed.


Feature: 0 1 2 3 4 5 6 7 8 9 10 11 12 [✓ 13] 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Current status of feature 13: enabled [✓ disabled]

Figure 4.7: Feature Visualization taken from f_{13}: The problem with this layer is it is clearly attributed as "ships" as FV patterns can be recognizable as sea colors while most activated images are all "cars" and the contributions are contested for 2 different classes. Therefore this feature is misattributed.

After disabling and retraining the model with masked features for 10 epochs with Early Stopping of 2 epochs, validation accuracy improved from 88.68% to 88.99%, and macro f1 improved from 0.8873 to 0.8911. Inspecting Table 4.2, Airplane, Automobile, Cat, Horse, Ship, and Truck classes are improved significantly, Deer class is negatively affected by feature disabling, and other classes are affected insignificant amounts in terms of F1 score metric. In this experiment, we considered weights per feature rather than contribution by running it with the validation set. Therefore, while overall results are improved by retraining and adapting to changes, possible useful features are discarded due to this data-agnostic approach.

Table 4.2: Results with CIFAR-10

Training Type	Accuracy	F1	FPR	FNR	Class	Per Class F1
Initial	88.68	0.8873	0.015	0.084	airplane	0.8899
					automobile	0.9287
					bird	0.8585
					cat	0.7884
					deer	0.9003
					dog	0.8319
					frog	0.9088
					horse	0.9254
					ship	0.9176
					truck	0.9118
Fine-Tune	88.99	0.8911	0.052	0.166	airplane	0.8921
					automobile	0.9506
					bird	0.8590
					cat	0.7948
					deer	0.8764
					dog	0.8384
					frog	0.9144
					horse	0.9107
					ship	0.9495
					truck	0.9417

As has been demonstrated, using too many classes with too few features in debugging phase causes entanglement; therefore, disabling features with considering only one class is problematic as it would disable useful contributions to other classes. Also, Feature Visualization [25] with Activation Maximization [24] causes high-frequency outputs that could be hard to interpret and decide whether the feature is useful or not.

4.5 Experiment 2: Human Opinion Collection, Feature Ranking and Disabling with Reduced Animals-10

After inspecting the effects of Feature Visualization and weight comparison assisted elimination on CIFAR-10 [56], a thorough experiment is conducted with a reduced set of Animals-10 [57]. This set only contained 3 of 10 classes of the original dataset which are Cat, Cow, and Spider. In this part, a sample of data for each feature is selected in accordance with their contribution to the final decision via the help of Algorithm 2. Then, they are processed with LIME [27] to get positive parts that contribute to the final decision. The processed samples are combined to represent what feature stores and provides for the final decision part as can be seen in Figure 4.8. Also, in this part, instead of debugging and disabling with the opinion of one person, we used the expertise of 16 different individuals to gather more data by surveying whether a feature represents any of the given classes and giving points with respect to the representation of classes as seen in Figure 3.3. For this part, only features that have at least one positive influence on a class are used as can be seen in Table 4.4. With this data at hand, features are ranked for each class as high priority (HP) where the feature is important for classes selection at the final stage, medium priority (MP) where it does not have a decisive role but still contributes small amounts, low priority (LP) where the feature does not contribute or negatively contributes to decision part on the specific class as can be seen for each class in Figures 4.9, 4.10 and 4.11. All visualization, human votings, and actual contributions of features can be seen in Appendix A.

Table 4.3: Sample Amounts in reduced Animal-10

Class	Train	Val	Total
Cat	1501	167	1668
Cow	1679	187	1846
Spider	4338	483	4821

Algorithm 1 Probability list creator for Sample algorithm

Input: prob_list = $[p_1, \ldots p_N]$ -> probability of each element getting selected.

Output: range = $[r_1, \ldots r_N]$ -> Each elements' probability accumulated to achieve weighted random

```
1: function $probability\_range(prob\_list)$
2:     $range \leftarrow []$
3:     $cumulative\_prob \leftarrow 0$
4:     for all $j = 1$ to size($prob\_list$) do
5:         $cumulative\_prob \leftarrow cumulative\_prob + prob\_list[j]$
6:         $range.insert(cumulative\_prob)$
7:     end for
8:     return $range$
9: end function
```

Table 4.4: Feature Indexes ranked according to their contribution

Class	Priority Level		
	Low	Medium	High
Cat	2,4,8,9,13,14,15,17,22	1,5,11,12,16,20,21,24,29	0,3,6,7,25,26,27,28,31
Cow	0,8,15,17,20,21,22,24,29	1,2,3,5,6,7,13,14,25	4,9,11,12,16,26,27,28,31
Spider	0,3,11,16,25,26,27,28,31	1,2,4,5,6,7,9,12,21	8,13,14,15,17,20,22,24,29
No positive contribution	10,18,19,23,30		

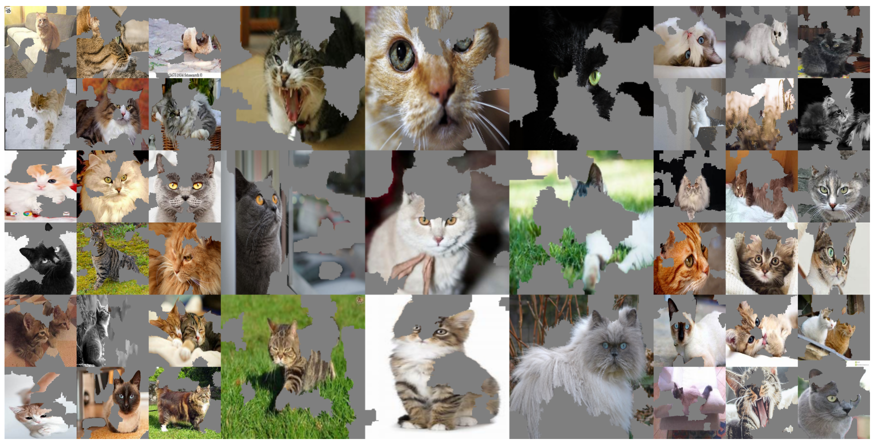


Figure 4.8: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance their positive contribution as in Algorithm 2

Algorithm 2 Sampling algorithm

Input: predicted_results = [(x_1, c_1), $(x_2, c_2), \dots$ (x_N, c_N)] -> validation dataset elements x with the contribution c to model's final prediction, N being the size of validation dataset.

Output: sample_list= [(x_1, c_1), $(x_2, c_2), \dots$ (x_M, c_M)] -> sampled elements from predicted_results with M<N

```
1:  $candidate\_list \leftarrow []$
2:  $probability\_list \leftarrow []$
3:  for all $j = 1$ to N do
4:      $x_j, c_j \leftarrow predicted\_results[j]$
5:      if $c_j > 0$ then
6:          $candidate\_list.insert((x_j, c_j))$
7:          $probability\_list.insert(c_j)$
8:      end if
9:  end for
10: $probability\_list\_normalised \leftarrow probability\_list/sum(probability\_list)$
11: $prob\_range \leftarrow probability\_range(probability\_list\_normalised)$
12: for all $i = 1$ to M do
13:     $prob \leftarrow random.generate(0, 1)$
14:     for all $j = 1$ to size($candidate\_list$) do
15:         if $prob < prob\_range[j]$ then
16:             $sample\_list.insert(candidate\_list[j])$
17:             $probability\_list.pop(j)$
18:             $candidate\_list.pop(j)$
19:             break
20:         end if
21:     end for
22:     $probability\_list\_normalised \leftarrow probability\_list/sum(probability\_list)$
23:     $prob\_range \leftarrow probability\_range(probability\_list\_normalised)$
24: end for
```

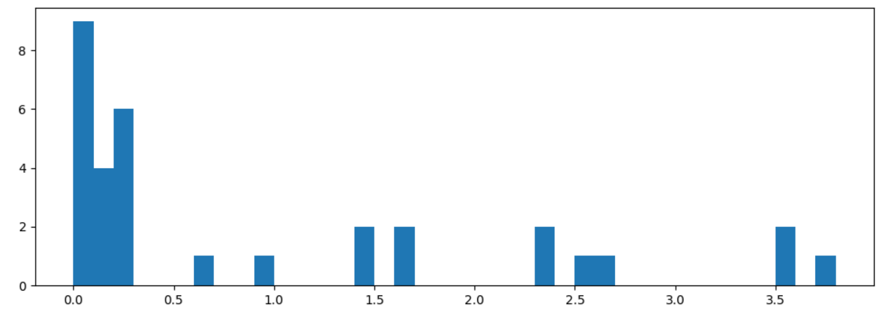


Figure 4.9: Features ranked with respect to their resemblance to "Cat" class

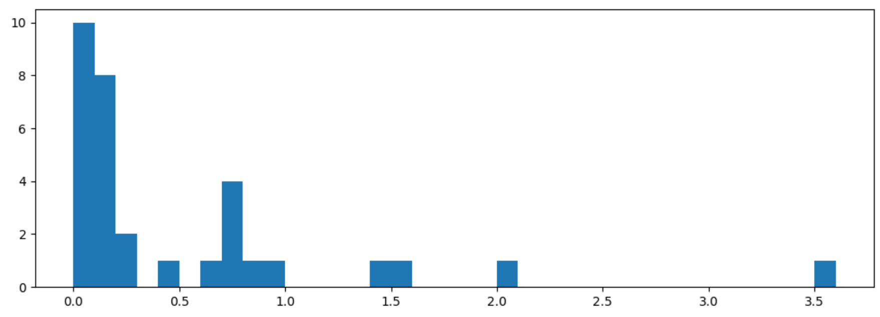


Figure 4.10: Features ranked with respect to their resemblance to "Cow" class

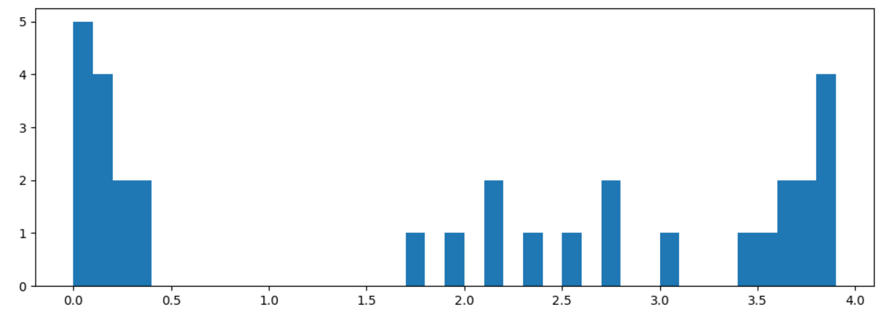
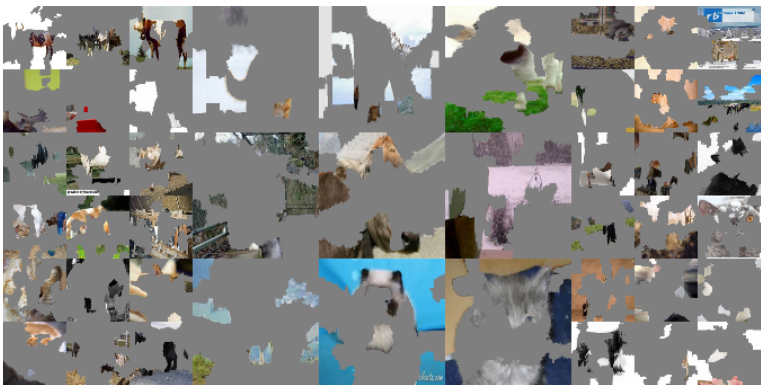


Figure 4.11: Features ranked with respect to their resemblance to "Spider" class

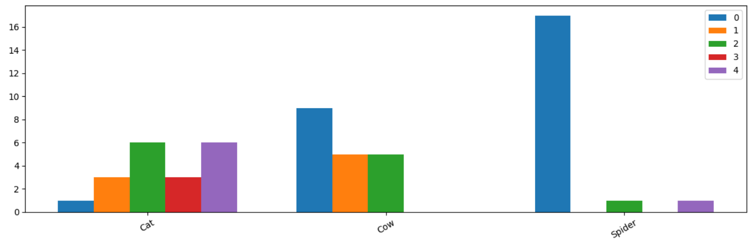
4.5.1 Effect of disabling features according to human knowledge

After ranking each feature according to the answers and points as can be seen in Figures 4.9, 4.10 and 4.11 and Table 4.4 given to each feature with questions provided such as in Figure 3.3, we disabled each priority group for each class except the non-positive contributing ones both individually and in combinations of two and both class weight only and whole feature such as disabling only cat weight for low priority (LP) or disabling all weights calculating feature for medium and high priority (MP & HP) ranked features of cow class. For Confusion matrices and results for this experiment can be seen in Appendix B. The results are taken without retraining or fine-tuning after disabling to investigate the effects of features. For this experiment, we choose metrics F1 per class, Accuracy, and Macro F1

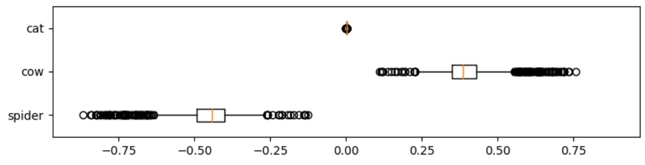
Investigating results for the Cat class, medium priority (MP) classes actually contribute more to the class than high priority(HP) classes where features are expected to mostly only contribute to the class itself. This is because of the entanglement of features in combination with more than one class. For example, Feature 28 as can be seen in Figure 4.12 voted more contributing to the Cat class as it is placed as high priority feature for the Cat class actually contributes less than feature 11 and 21 as stats can be seen in Figure 4.14 and 4.13. However, due to the sampling process and both features actually contributing to cow (Feature 28) and spider(Feature 11) classes same or more amount, these features can not be solely voted as the Cat contributing classes. Therefore, when high-priority(HP) features are disabled, there is little change whereas when mediums(MP) are disabled there is a bigger change. When both priorities are combined, the positive amount reduces to zero as there will be no contributing feature remaining to make a positive impact on the Cat class; therefore making the F1 score for the Cat class NaN. Also, without retraining, disabling low-priority(LP) features' weights that contribute to the Cat class, improves overall metrics slightly. Investigating further by splitting medium-priority features into 3 equal parts, we can see that features 11,12,21 are heavily contributing to the Cat class while others are slightly supporting in the decision-making part as can be seen in Table 4.5. The results for the Cat class can be seen in Table 4.6.



(a) Feature Visualization

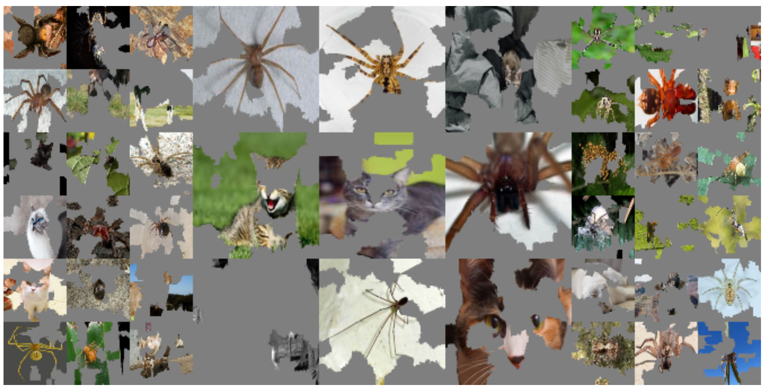


(b) Survey Answers

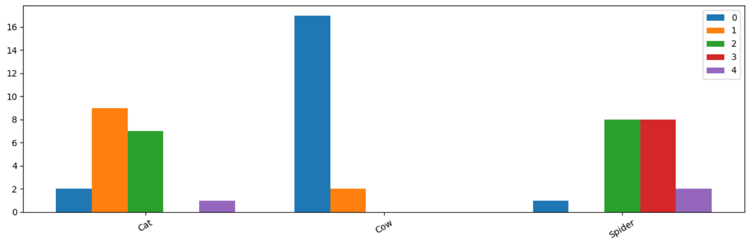


(c) Contribution of Feature

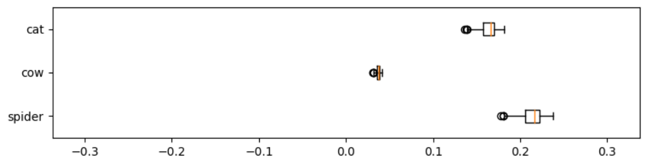
Figure 4.12: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 28 and answers in survey, and also contributions of feature index 28



(a) Feature Visualization

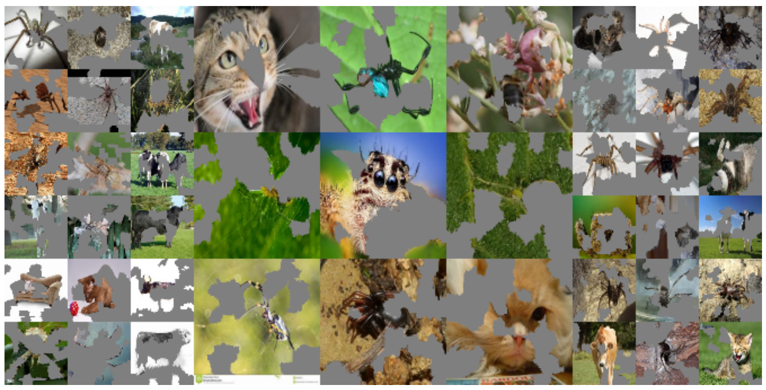


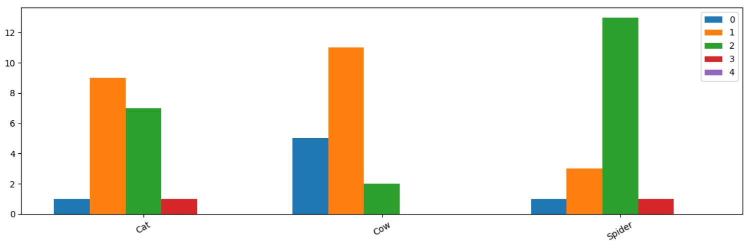
(b) Survey Answers



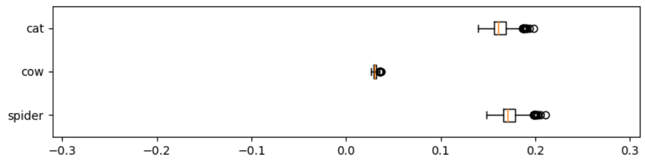
(c) Contribution of Feature

Figure 4.13: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 21 and answers in survey, and also contributions of feature index 21


(a) Feature Visualization



(b) Survey Answers



(c) Contribution of Feature

Figure 4.14: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 11 and answers in survey, and also contributions of feature index 11

Table 4.5: Results with Animal-10 for disabling different parts of medium priority features according to Cat

Model: ResNet-18	Dataset: Animals-10 reduced to 3 classes				
	Cat F1	Cow F1	Spider F1	Macro F1	Accuracy
Original	0.97	0.981	0.993	0.981	0.986
Disabling Cat Weights only					
Disabling 20,29,1	0.97	0.979	0.992	0.98	0.984
Disabling 16,24,5	0.97	0.981	0.993	0.981	0.986
Disabling 12,21,11	0.97	0.979	0.992	0.98	0.984
Disabling Whole Feature					
Disabling 20,29,1	0.973	0.981	0.994	0.983	0.987
Disabling 16,24,5	0.97	0.981	0.993	0.981	0.986
Disabling 12,21,11	0.134	0.976	0.866	0.659	0.812

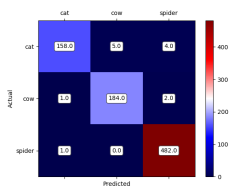
Table 4.6: Results with Animal-10 for disabling according to Cat

Model: ResNet-18	Dataset: Animals-10 reduced to 3 classes				
	Cat F1	Cow F1	Spider F1	Macro F1	Accuracy
Original	0.97	**0.981**	0.993	0.981	0.986
Disabling Cat Weights only					
Disabling LP	**0.973**	**0.981**	**0.994**	**0.983**	**0.987**
Disabling MP	0.953	**0.981**	0.988	0.974	0.98
Disabling HP	0.963	**0.981**	0.991	0.978	0.983
Disabling LP&MP	0.972	**0.981**	**0.994**	**0.983**	**0.987**
Disabling LP&HP	0.966	**0.981**	0.992	0.98	0.984
Disabling MP&HP	0.94	0.976	0.986	0.967	0.975
Disabling Whole Feature					
Disabling LP	0.967	0.233	0.851	0.683	0.795
Disabling MP	0.175	0.974	0.869	0.673	0.817
Disabling HP	0.957	0.979	0.988	0.974	0.98
Disabling LP&MP	0.333	NaN	NaN	0.111	0.2
Disabling LP&HP	0.963	NaN	0.83	0.597	0.763
Disabling MP&HP	NaN	0.976	0.856	0.611	0.798

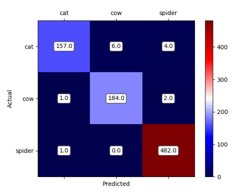
Examining Table 4.7, medium priority (MP) and low priority (LP) features impact low on classes' success individually, while high priority features are playing a decisive role in the classification of Cow. While there are disentangled features on high-priority features, it can be said that there are more features entangled with the Spider class than with the Cat class amongst high-priority features as those features break the balance in favor of the Spider class as can be seen in Figure 4.15c; therefore, creating more false positives for Spider class and reducing its F1 score more than Cat's F1 score. Similar to the Cat class, disabling low-priority (LP) features contributing to the Cow class improves the overall performance of the model. Looking into disabling only high-priority (HP) features, this features completely removes both Cat and Cow classes' positive predictions. From this, we can deduce that high-priority Cow features include weights that support positive contributions to the Cat class. Also, medium-priority features have little to no impact neither negative nor positive on the Cow class' prediction.

Table 4.7: Results with Animal-10 for disabling according to Cow

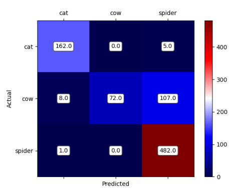
Model: ResNet-18	Dataset: Animals-10 reduced to 3 classes				
	Cat F1	Cow F1	Spider F1	Macro F1	Accuracy
Original	0.97	0.981	0.993	0.981	0.986
Disabling Cow Weights only					
Disabling LP	0.966	0.979	0.993	0.979	0.984
Disabling MP	0.963	0.976	0.993	0.977	0.983
Disabling HP	0.959	0.556	0.895	0.803	0.855
Disabling LP&MP	0.963	0.974	0.994	0.977	0.983
Disabling LP&HP	0.97	0.947	0.979	0.965	0.97
Disabling MP&HP	0.967	0.916	0.97	0.951	0.958
Disabling Whole Feature					
Disabling LP	**0.973**	**0.984**	**0.995**	**0.984**	**0.988**
Disabling MP	**0.973**	0.981	0.994	0.983	0.987
Disabling HP	NaN	NaN	0.732	0.244	0.577
Disabling LP&MP	0.969	0.979	0.994	0.981	0.986
Disabling LP&HP	0.204	NaN	0.743	0.2516	0.6
Disabling MP&HP	NaN	NaN	0.732	0.244	0.577



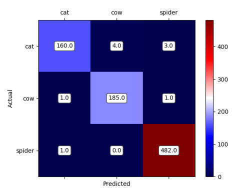
(a) Disabled LP features for only cow weights



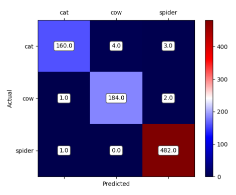
(b) Disabled MP features for only cow weights



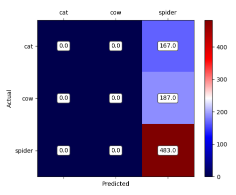
(c) Disabled HP features for only cow weights



(d) Disabled LP features for all weights



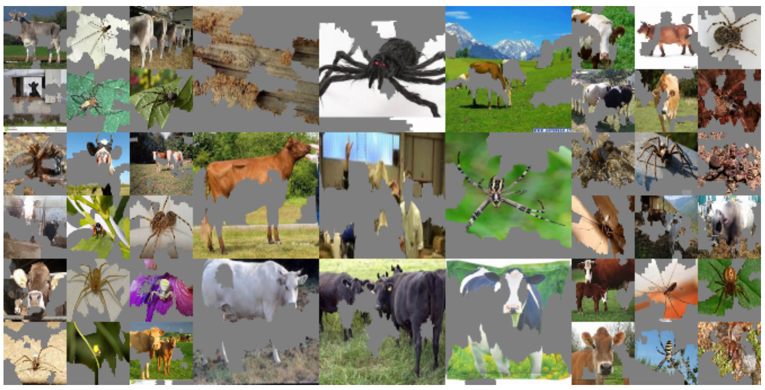
(e) Disabled MP features for all weights



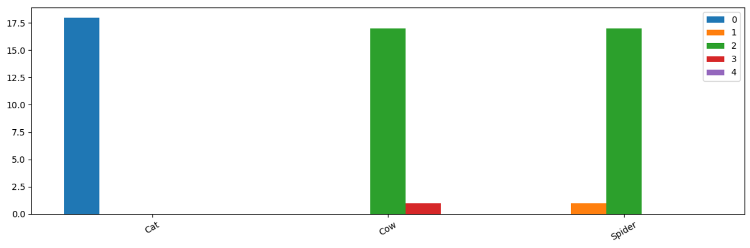
(f) Disabled HP features for all weights

Figure 4.15: Confusion Matrix disabling one priority rank features for Cow class

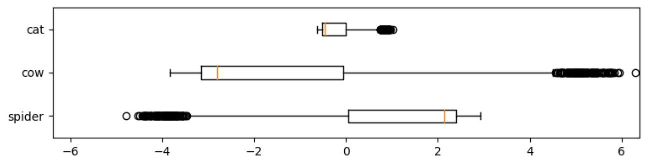
Inspecting results on Table 4.8, similar to the Cat class, medium-priority features are contributing more than high-priority features. In most of those features, the Spider class shares the feature with other classes. In the specific example of Feature 4, the feature is visualized as contributing Cows and Spiders equally while in Figure 4.16 it can be seen that Cow contribution is mostly negative with the exception for some outliers above the last quartile. Therefore, while removing MP features' Spider class weights does not change metrics where HP features compensate completely, deleting MP features completely eliminates positive Cow predictions and changes most of the predictions to the Cat class. Hence, it can be derived that the Spider class is more entangled with the Cow class than the Cat class in MP features. Producing the same result as Cow results' Disabling HP features in Table 4.7, Spider's LP and MP combination includes all HP features from the Cow class. This provides insight that some of the Cow class is entangled with the Spider class in their most contributing features. In another example, Disabling the MP&HP features of Spider produces the same results as disabling the LP&MP features of Cat classes which removes both Spider and Cow classes' positive predictions.



(a) Feature Visualization



(b) Survey Answers



(c) Contribution of Feature

Figure 4.16: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 4 and answers in survey, and also contributions of feature index 4

Table 4.8: Results with Animal-10 for disabling according to Spider

Model: ResNet-18	Dataset: Animals-10 reduced to 3 classes				
	Cat F1	Cow F1	Spider F1	Macro F1	Accuracy
Original	0.97	0.981	0.993	0.981	0.986
Disabling Spider Weights only					
Disabling LP	0.963	0.979	0.99	0.977	0.982
Disabling MP	0.97	0.981	0.993	0.981	0.986
Disabling HP	**0.973**	0.981	0.994	0.983	0.987
Disabling LP&MP	0.963	0.981	0.991	0.978	0.983
Disabling LP&HP	0.966	0.981	0.992	0.98	0.984
Disabling MP&HP	0.964	0.979	0.99	0.978	0.982
Disabling Whole Feature					
Disabling LP	0.957	0.976	0.987	0.973	0.978
Disabling MP	0.356	NaN	0.23	0.195	0.274
Disabling HP	**0.973**	**0.984**	**0.995**	**0.984**	**0.988**
Disabling LP&MP	NaN	NaN	0.732	0.244	0.577
Disabling LP&HP	0.966	0.981	0.992	0.98	0.984
Disabling MP&HP	0.333	NaN	NaN	0.111	0.2

CHAPTER 5

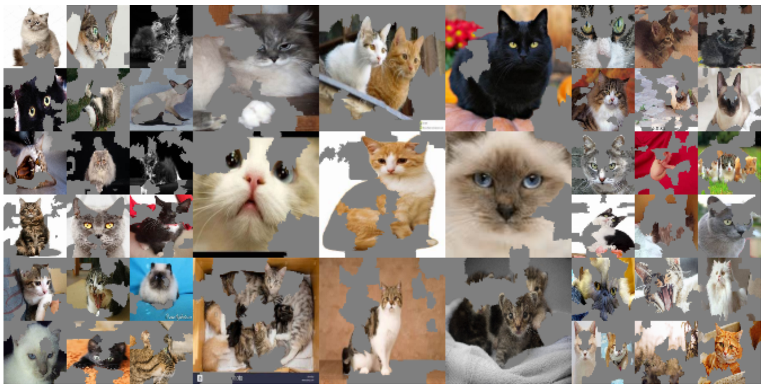
CONCLUSIONS

In this study, we have analyzed the effects of human-guided feature disabling with the assistance of FV and XAI. We have provided results and analysis for retraining DL-based image classification algorithms. During experiments, we performed the visual analysis of the model and decided which feature is irrelevant using FV, ranking images by activation amounts, and weight visualization. Recently, active learning methods are popular for neural network debugging via manipulating the network itself [23] or labeling unlabelled classes after inferring them with model [22]. However, these models are not taking advantage of feature visualization, and they correct the model by inspecting examples instead of the network itself. Therefore, FIND [12] which employs LRP (Layer-wise Relevance Propagation) [28] to find global explanations for each feature. This does not work as expected in image-based problems as there are no "embedding" vectors or layers that can encode the image in a way they are completely distinguishable in latent space before it is fed to the model. Therefore, Activation Maximization [24] is employed for feature visualization to attribute latent features some familiarity to the user who debugs the system. However, using only Activation Maximization [24] for visualization of layers could confuse the users as it can produce high-frequency patterns which probably have no familiarity to the user. To overcome this problem, maximum activating images from the dataset are brought together in a collage to make a decision about to which class this feature contributes the most easier. To complete the decision, we presented the classifier module's weights to show to which class that feature actually contributes.

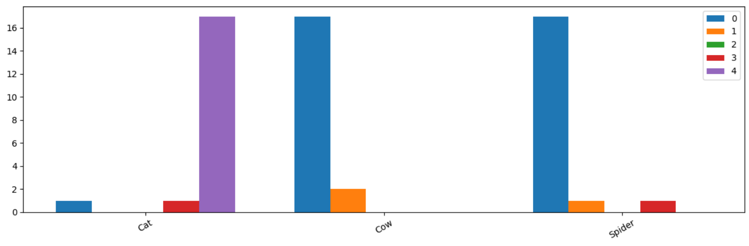
During the experiment phase, we used a custom ResNet-18 [58] and trained and tested it with CIFAR-10 [56] and Animal-10 [57]. We disabled features that are either ambiguous, misleading or have high-frequency FV that are not distinguishable. This improved overall F1 score and most classes' F1 scores after fine-tuning with masked features. Results show that FV is not enough to determine whether a feature is relevant or not. Therefore, we employed LIME [27] which is an XAI method for highlighting relevant parts of the input for Experiment 2 to test human expertise vs the model's understanding of each feature. Even though LIME proved useful for visualization, it is computationally expensive to create visuals for each feature due to the individual processing of inputs. To overcome this, we take weighted samples on the validation set and used them for visualization. This experimentation showed that we can use XAI techniques for the diagnosis of errors in decision-making and debugging of image classification algorithms. For future studies, Self Explaining Neural Networks [8] or Concept Bottleneck Models [9], if there is a labeled set of concepts, could be employed to investigate samples from the dataset to improve visualization. Moreover, rather than disabling features completely, regularization methods or rewards and punishments for specific weights could be applied to get more interpretable and better-contributing features.

APPENDIX A

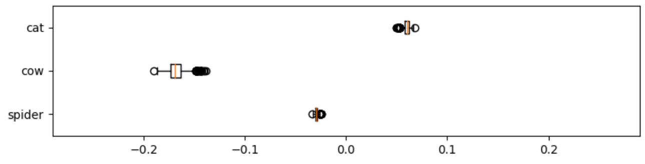
FEATURE VISUALIZATIONS VIA LIME IN ANIMAL-10



(a) Feature Visualization

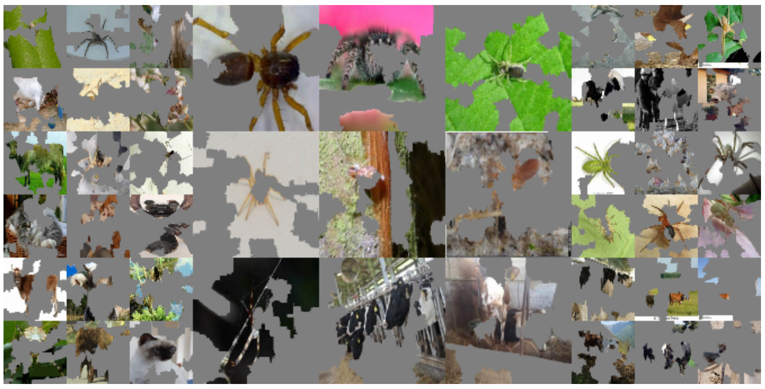


(b) Survey Answers

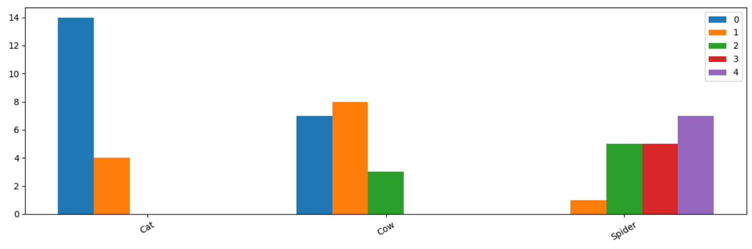


(c) Contribution of Feature

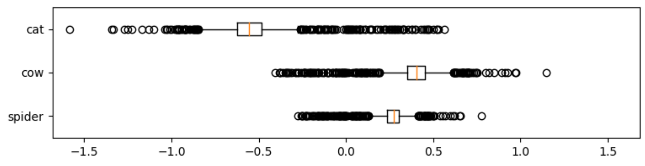
Figure A.1: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 0 and answers in survey, and also contributions of feature index 0



(a) Feature Visualization

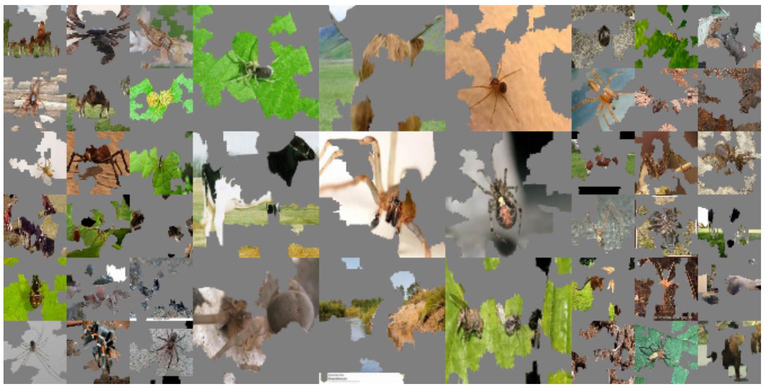


(b) Survey Answers

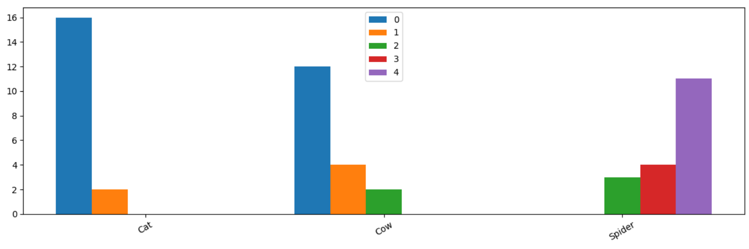


(c) Contribution of Feature

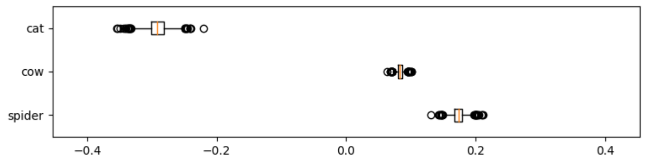
Figure A.2: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 1 and answers in survey, and also contributions of feature index 1



(a) Feature Visualization

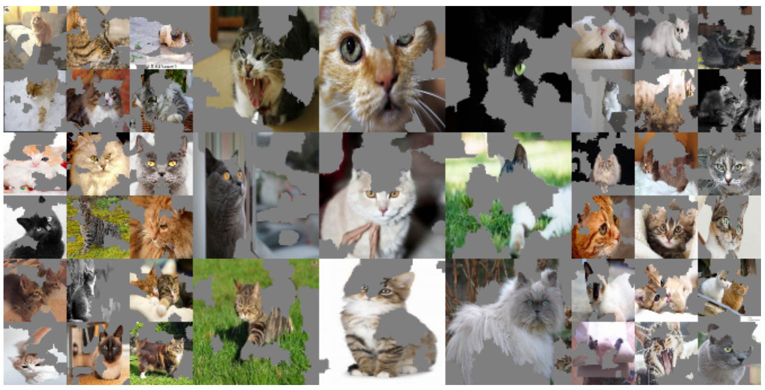


(b) Survey Answers

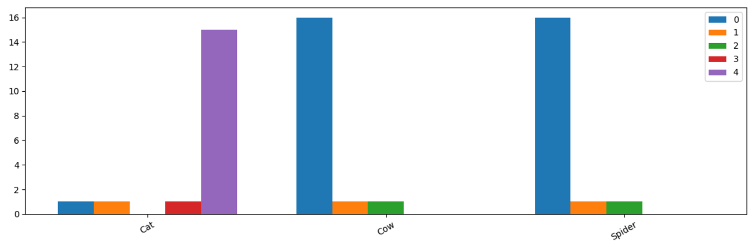


(c) Contribution of Feature

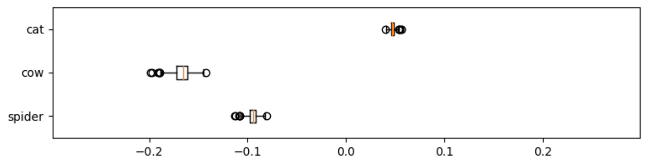
Figure A.3: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 2 and answers in survey, and also contributions of feature index 2



(a) Feature Visualization



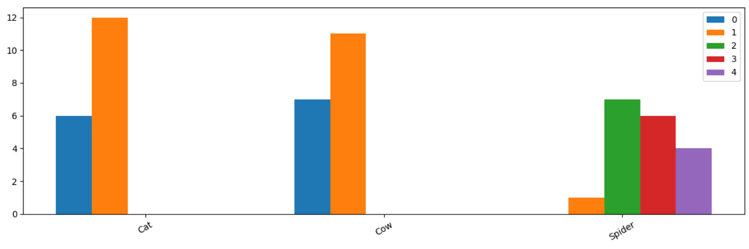
(b) Survey Answers



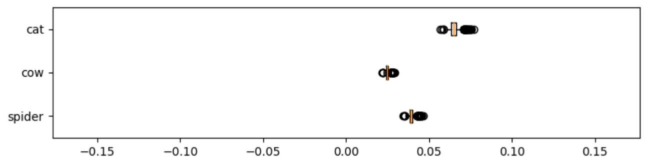
(c) Contribution of Feature

Figure A.4: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 3 and answers in survey, and also contributions of feature index 3

(a) Feature Visualization

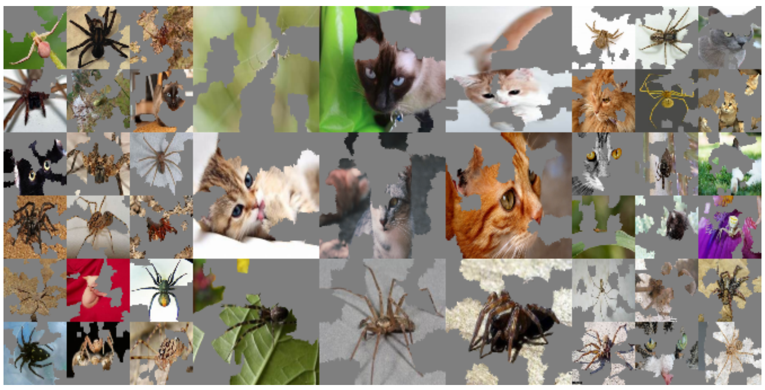


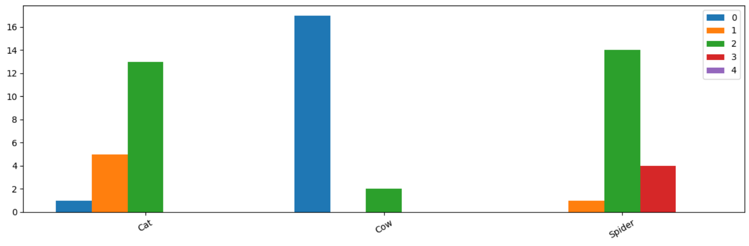
(b) Survey Answers



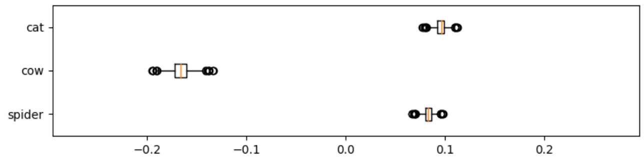
(c) Contribution of Feature

Figure A.5: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 5 and answers in survey, and also contributions of feature index 5


(a) Feature Visualization

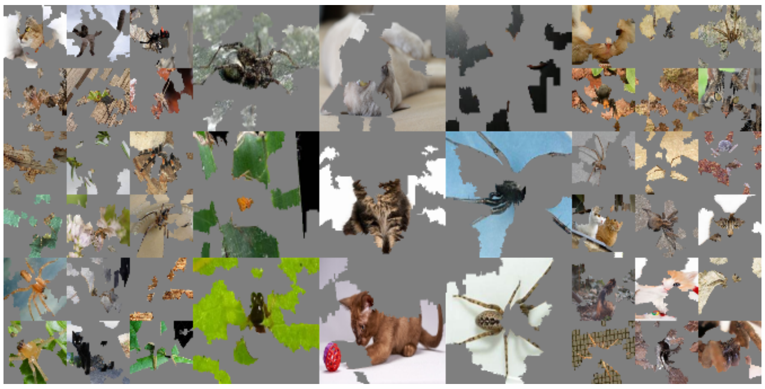


(b) Survey Answers

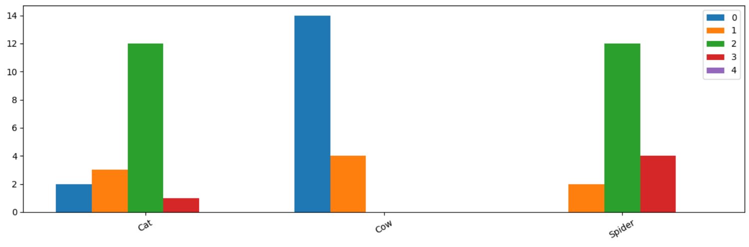


(c) Contribution of Feature

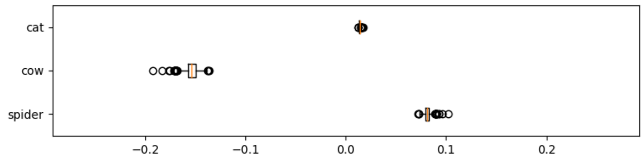
Figure A.6: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 6 and answers in survey, and also contributions of feature index 6



(a) Feature Visualization

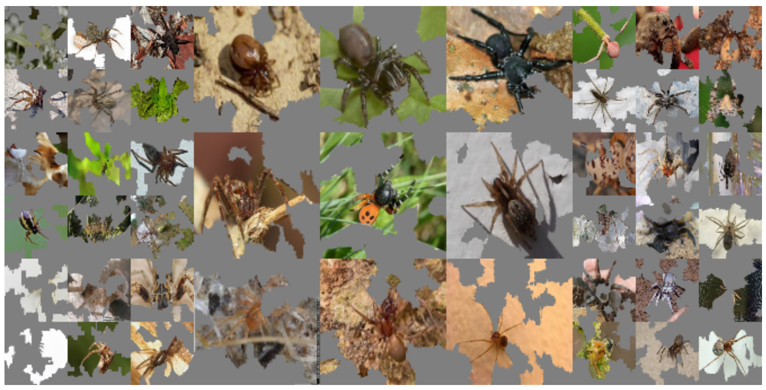


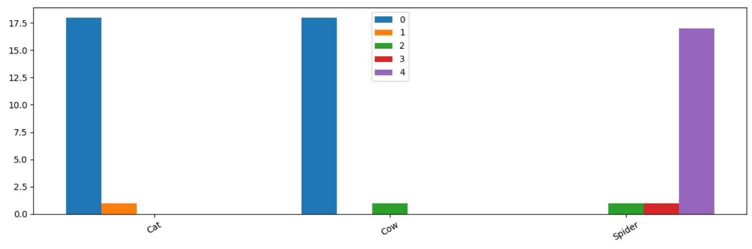
(b) Survey Answers



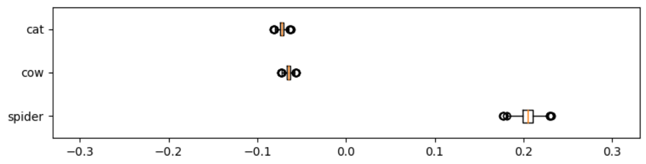
(c) Contribution of Feature

Figure A.7: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 7 and answers in survey, and also contributions of feature index 7


(a) Feature Visualization

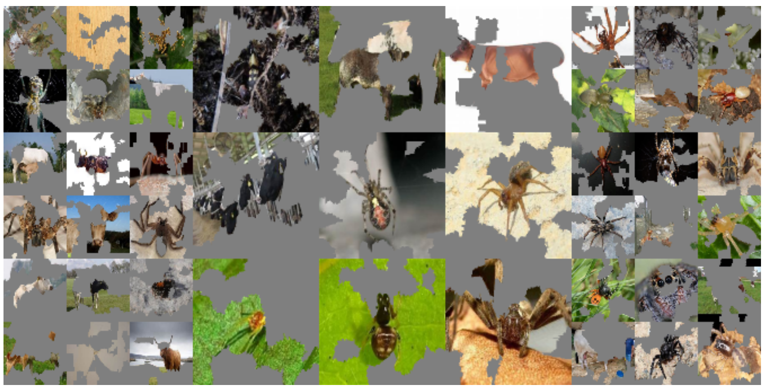


(b) Survey Answers

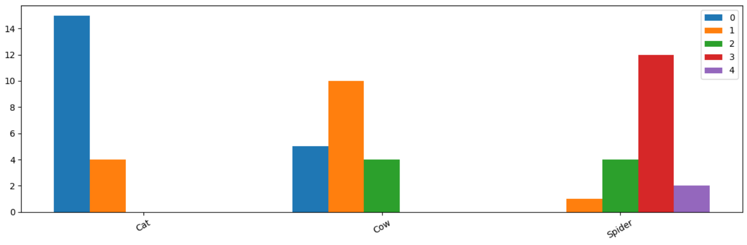


(c) Contribution of Feature

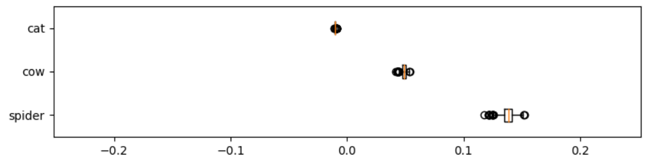
Figure A.8: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 8 and answers in survey, and also contributions of feature index 8



(a) Feature Visualization

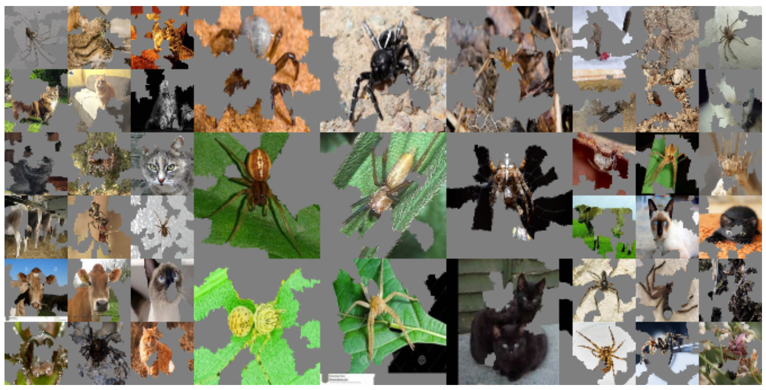


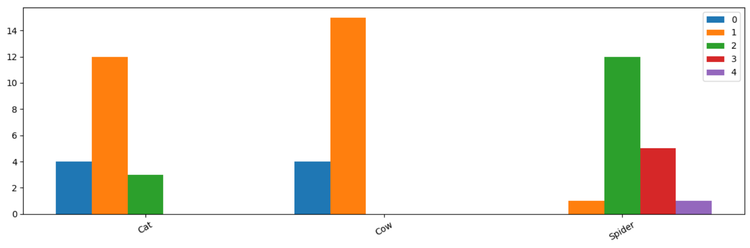
(b) Survey Answers



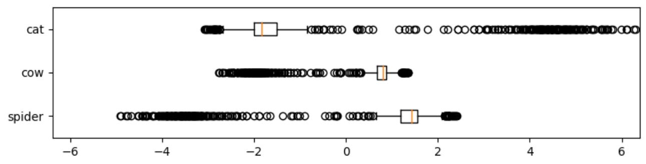
(c) Contribution of Feature

Figure A.9: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 9 and answers in survey, and also contributions of feature index 9


(a) Feature Visualization

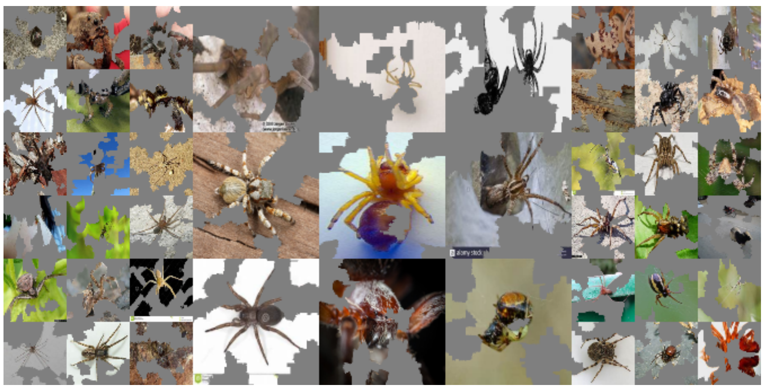


(b) Survey Answers

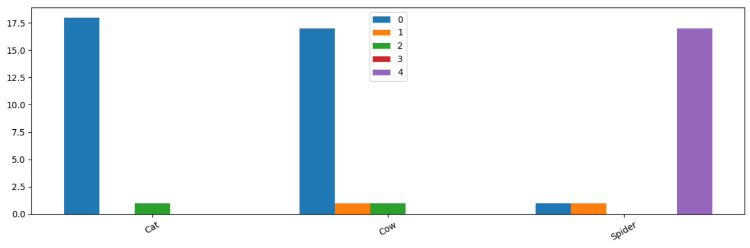


(c) Contribution of Feature

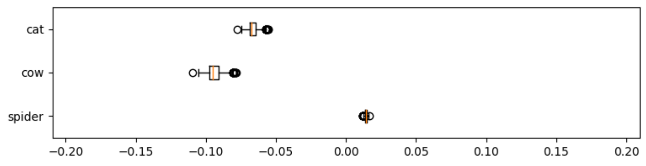
Figure A.10: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 12 and answers in survey, and also contributions of feature index 12



(a) Feature Visualization

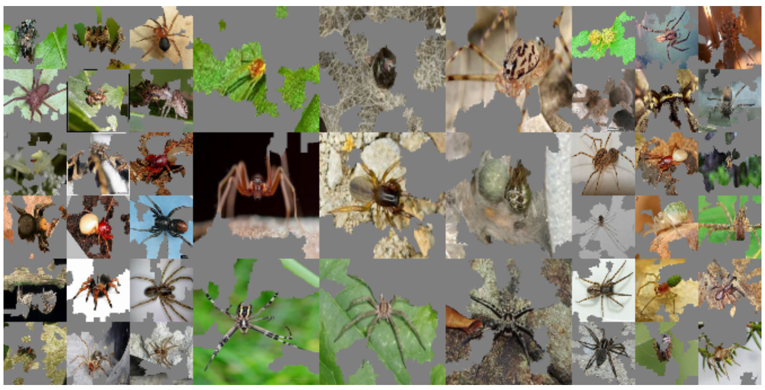


(b) Survey Answers

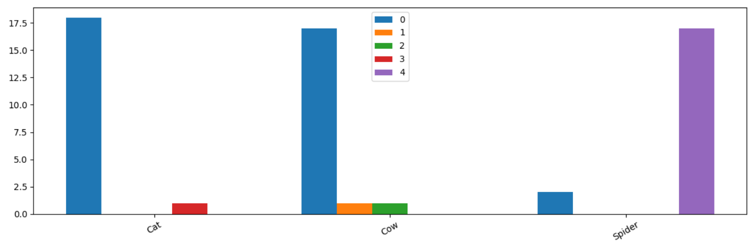


(c) Contribution of Feature

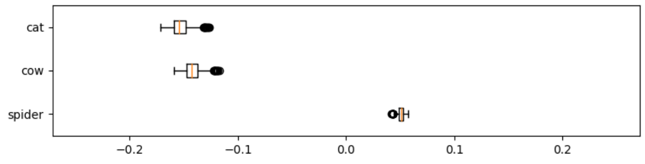
Figure A.11: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 13 and answers in survey, and also contributions of feature index 13



(a) Feature Visualization

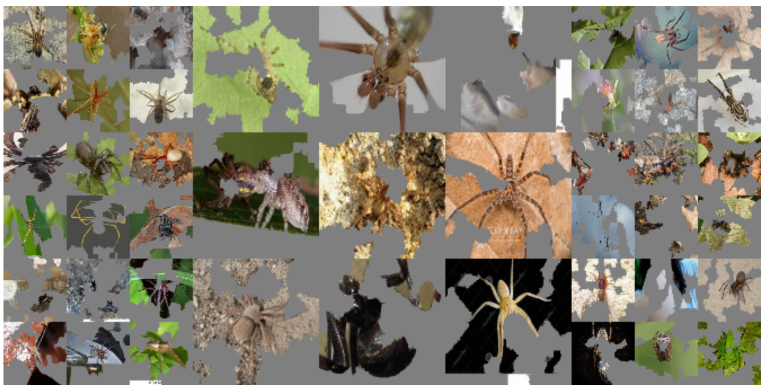


(b) Survey Answers

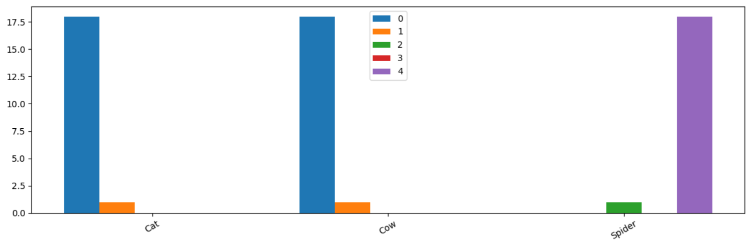


(c) Contribution of Feature

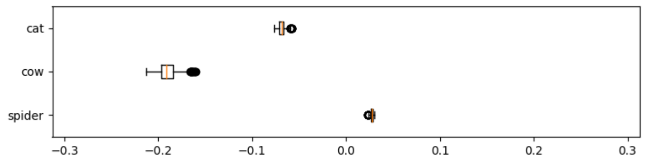
Figure A.12: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 14 and answers in survey, and also contributions of feature index 14



(a) Feature Visualization

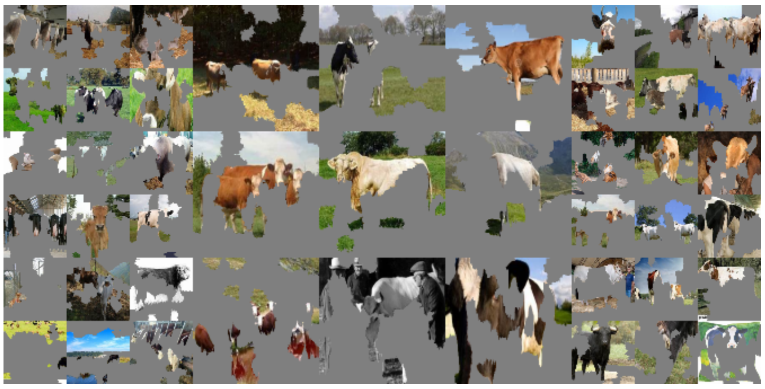


(b) Survey Answers

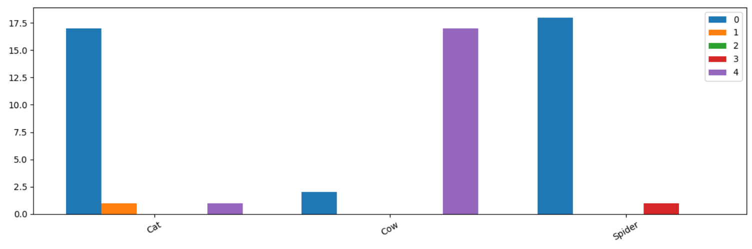


(c) Contribution of Feature

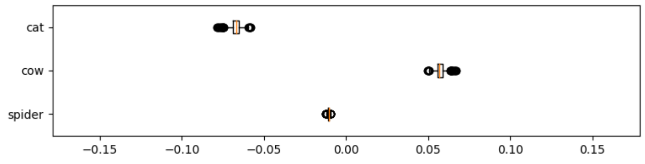
Figure A.13: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 15 and answers in survey, and also contributions of feature index 15



(a) Feature Visualization

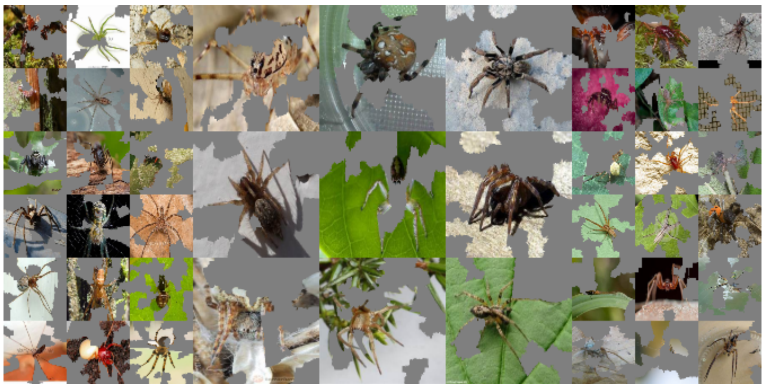


(b) Survey Answers

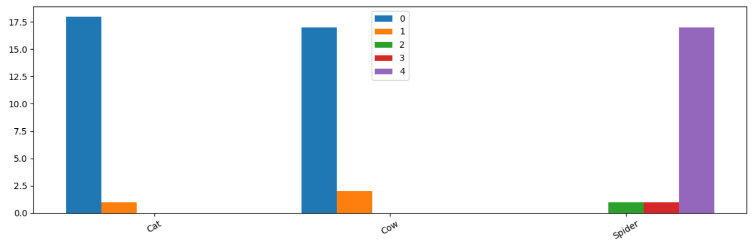


(c) Contribution of Feature

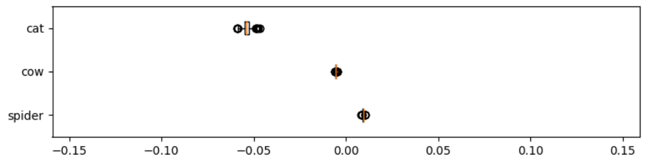
Figure A.14: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 16 and answers in survey, and also contributions of feature index 16



(a) Feature Visualization



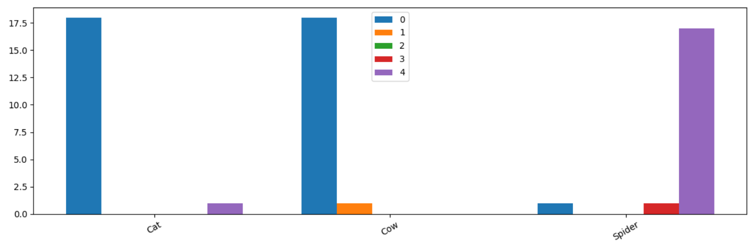
(b) Survey Answers



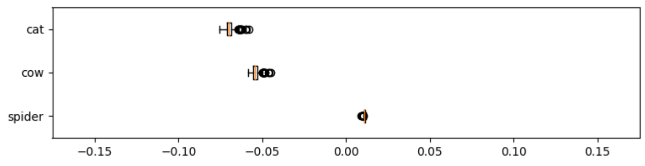
(c) Contribution of Feature

Figure A.15: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 17 and answers in survey, and also contributions of feature index 17

(a) Feature Visualization

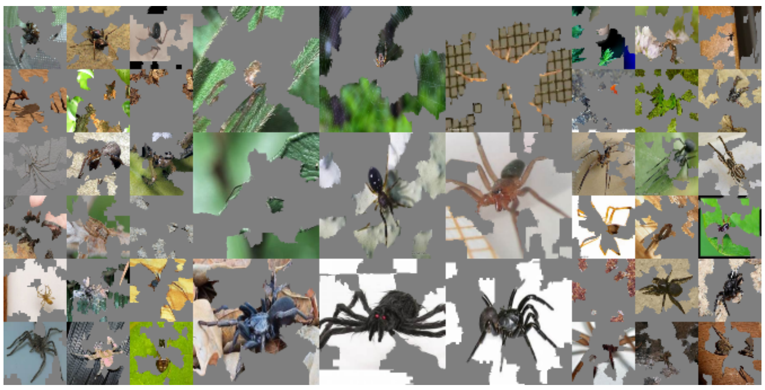


(b) Survey Answers

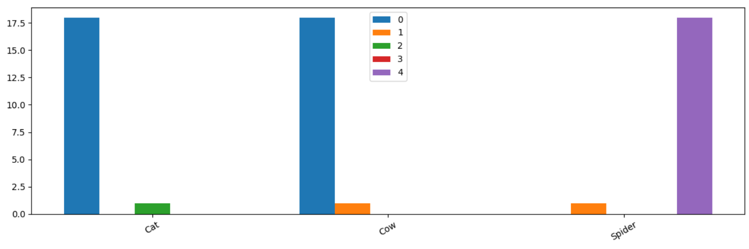


(c) Contribution of Feature

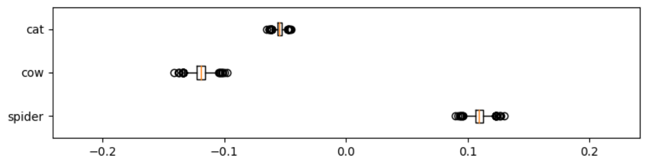
Figure A.16: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 20 and answers in survey, and also contributions of feature index 20



(a) Feature Visualization

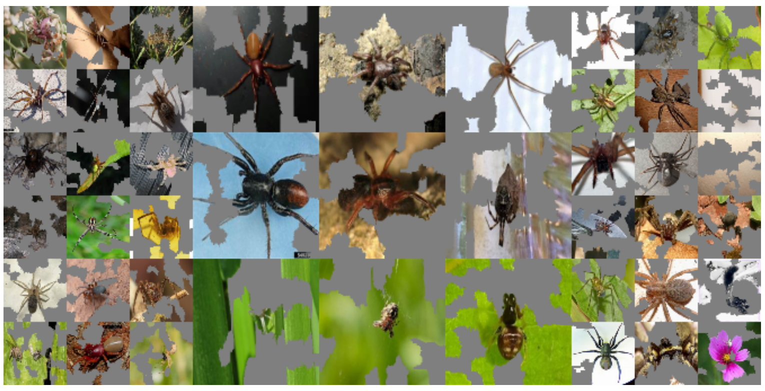


(b) Survey Answers

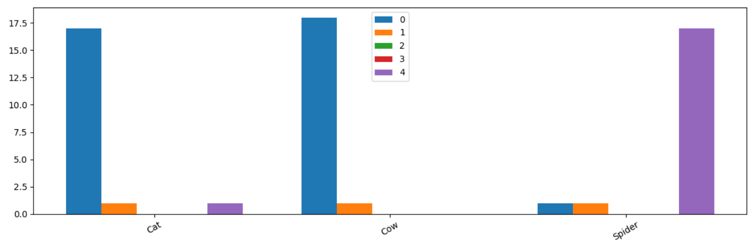


(c) Contribution of Feature

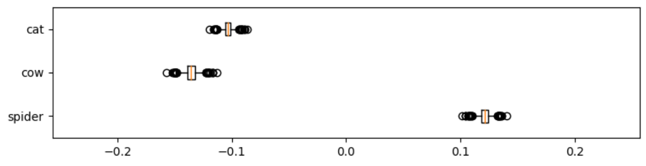
Figure A.17: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 22 and answers in survey, and also contributions of feature index 22



(a) Feature Visualization

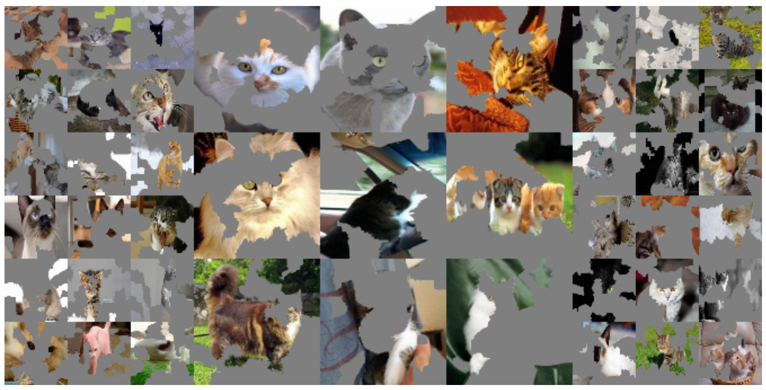


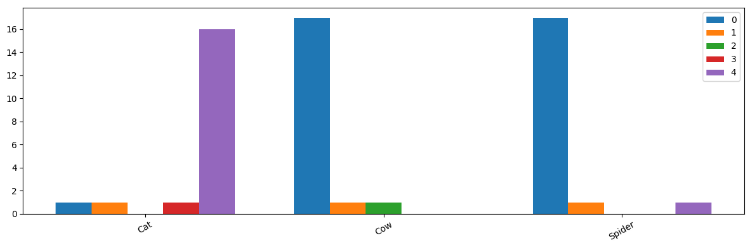
(b) Survey Answers



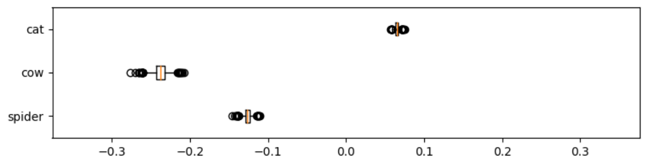
(c) Contribution of Feature

Figure A.18: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 24 and answers in survey, and also contributions of feature index 24


(a) Feature Visualization

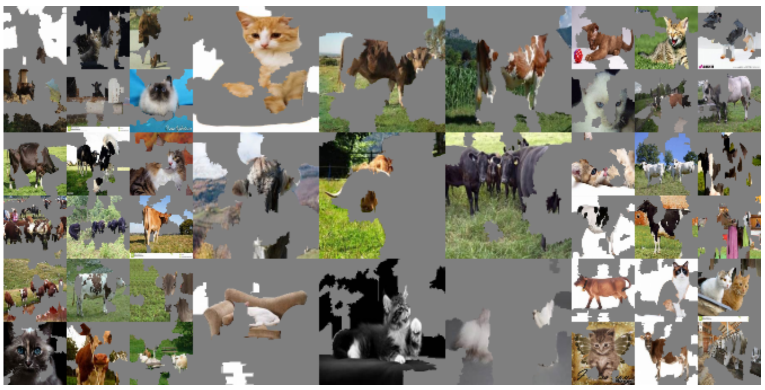


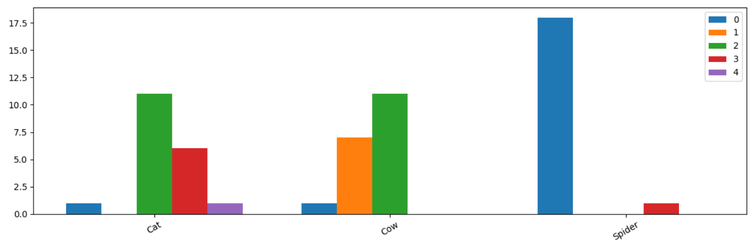
(b) Survey Answers



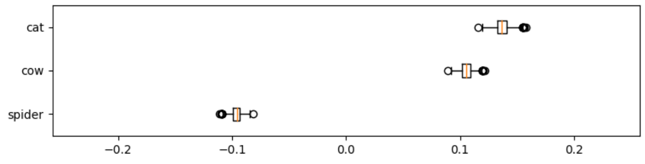
(c) Contribution of Feature

Figure A.19: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 25 and answers in survey, and also contributions of feature index 25


(a) Feature Visualization



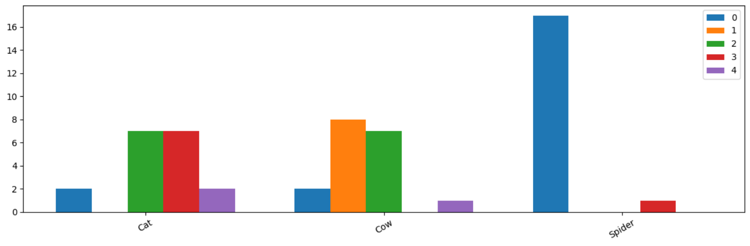
(b) Survey Answers



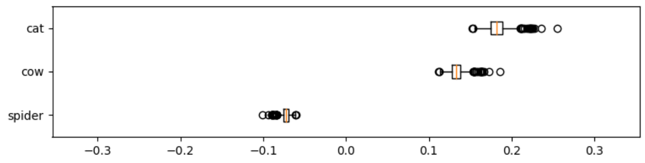
(c) Contribution of Feature

Figure A.20: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 26 and answers in survey, and also contributions of feature index 26

(a) Feature Visualization

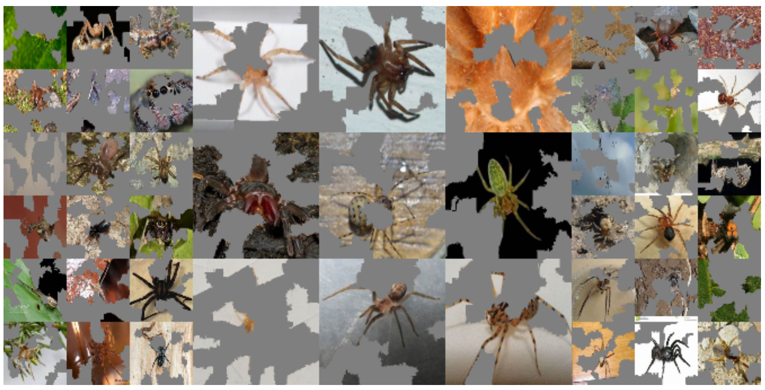


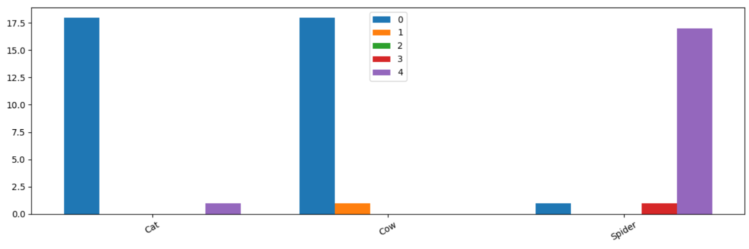
(b) Survey Answers



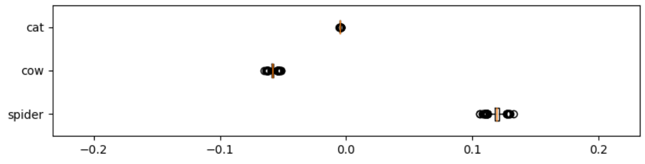
(c) Contribution of Feature

Figure A.21: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 27 and answers in survey, and also contributions of feature index 27


(a) Feature Visualization

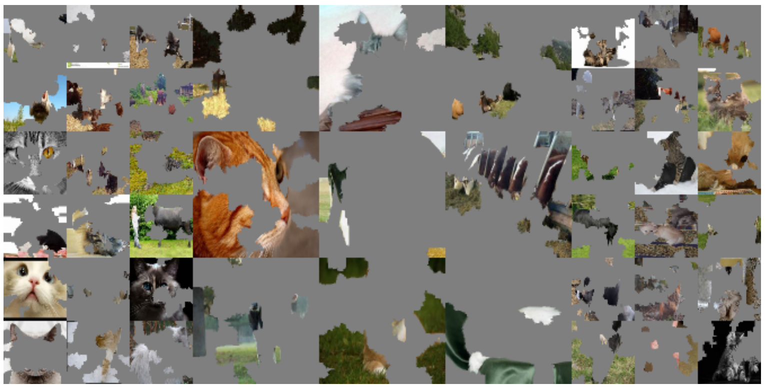


(b) Survey Answers

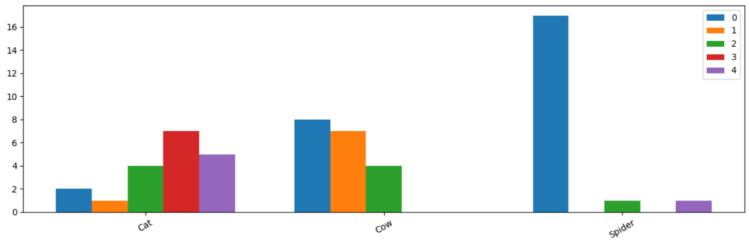


(c) Contribution of Feature

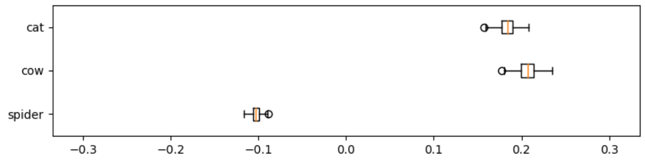
Figure A.22: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 29 and answers in survey, and also contributions of feature index 29



(a) Feature Visualization



(b) Survey Answers



(c) Contribution of Feature

Figure A.23: Feature Visualization via samples from reduced Animal-10 processed with LIME in accordance with their positive contribution for feature index 31 and answers in survey, and also contributions of feature index 31

APPENDIX B

CONFUSION MATRICES AND RESULTS FROM ANIMAL-10

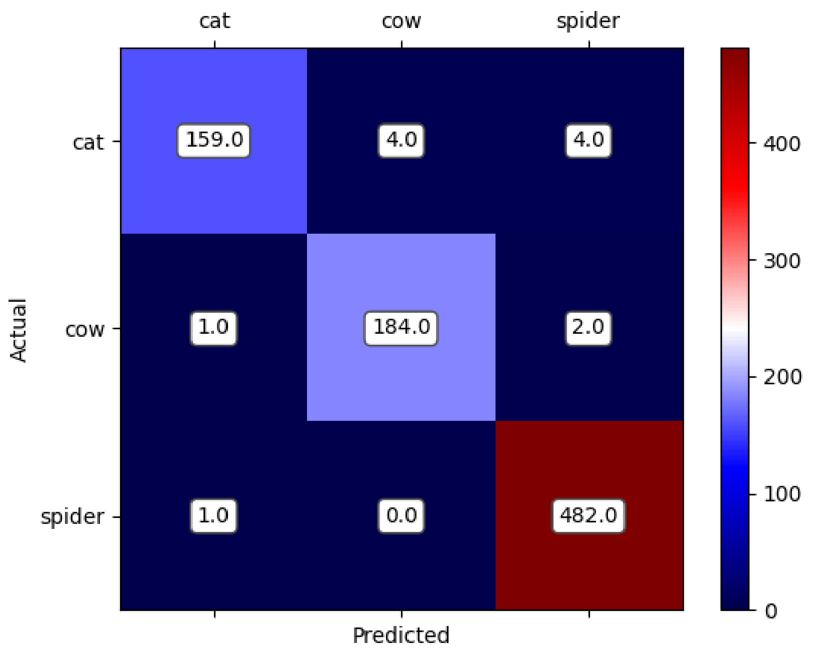
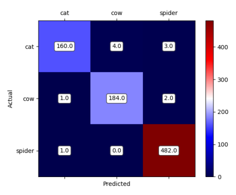
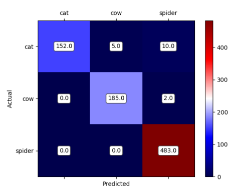


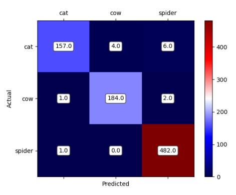
Figure B.1: Confusion Matrix for Original Model with no disabled features



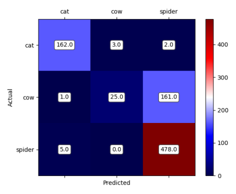
(a) Disabled LP features for only cat weights



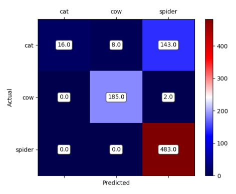
(b) Disabled MP features for only cat weights



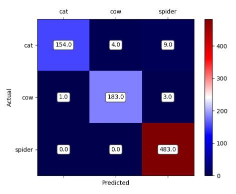
(c) Disabled HP features for only cat weights



(d) Disabled LP features for all weights

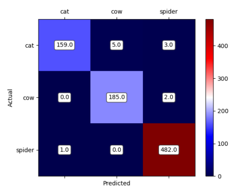


(e) Disabled MP features for all weights

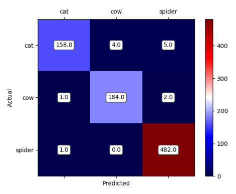


(f) Disabled HP features for all weights

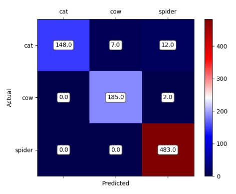
Figure B.2: Confusion Matrix disabling one priority rank features for Cat class



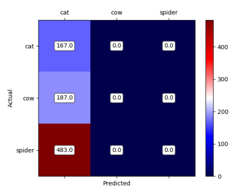
(a) Disabled LP & MP features for only cat weights



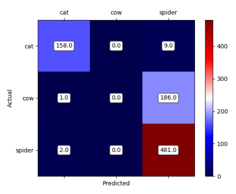
(b) Disabled LP & HP features for only cat weights



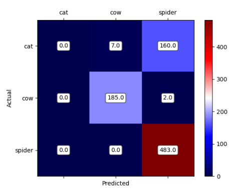
(c) Disabled MP & HP features for only cat weights



(d) Disabled LP & MP features for all weights

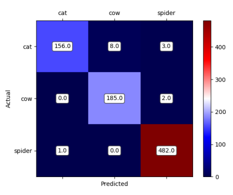


(e) Disabled LP & HP features for all weights

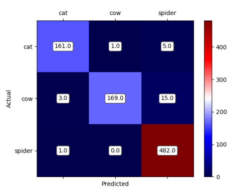


(f) Disabled MP & HP features for all weights

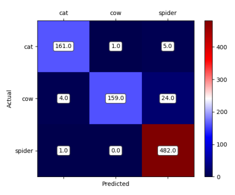
Figure B.3: Confusion Matrix disabling combination of two priority rank features for Cat class



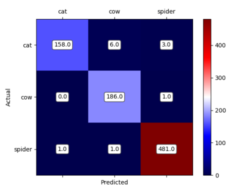
(a) Disabled LP & MP features for only cow weights



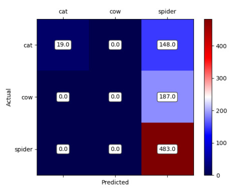
(b) Disabled LP & HP features for only cow weights



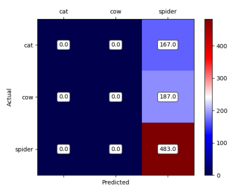
(c) Disabled MP & HP features for only cow weights



(d) Disabled LP & MP features for all weights

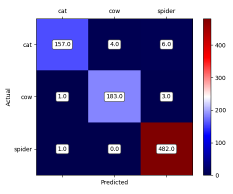


(e) Disabled LP & HP features for all weights

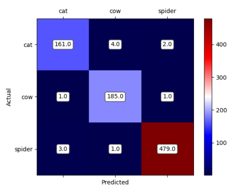


(f) Disabled MP & HP features for all weights

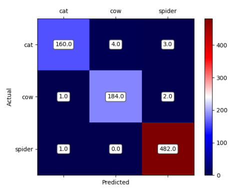
Figure B.4: Confusion Matrix disabling combination of two priority rank features for Cow class



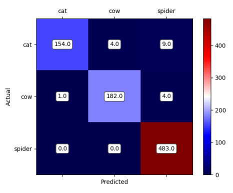
(a) Disabled LP features for only spider weights



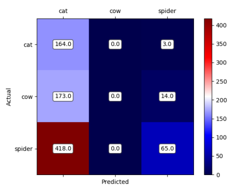
(b) Disabled MP features for only spider weights



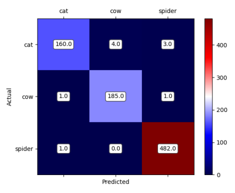
(c) Disabled HP features for only spider weights



(d) Disabled LP features for all weights

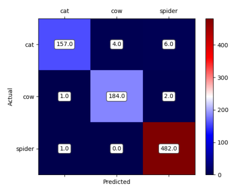


(e) Disabled MP features for all weights

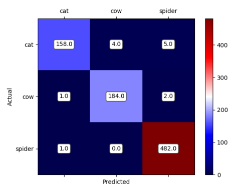


(f) Disabled HP features for all weights

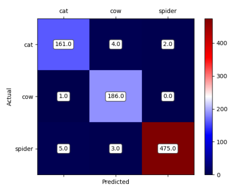
Figure B.5: Confusion Matrix disabling one priority rank features for Spider class



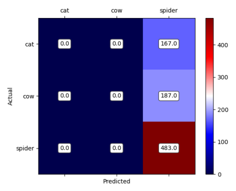
(a) Disabled LP & MP features for only spider weights



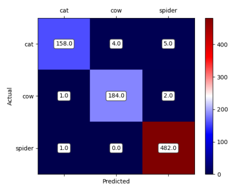
(b) Disabled LP & HP features for only spider weights



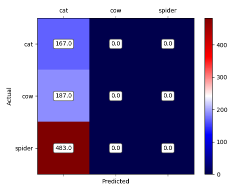
(c) Disabled MP & HP features for only spider weights



(d) Disabled LP & MP features for all weights



(e) Disabled LP & HP features for all weights



(f) Disabled MP & HP features for all weights

Figure B.6: Confusion Matrix disabling combination of two priority rank features for Spider class

REFERENCES

[1] "What does gradient descent actually mean," 2020.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] A. Mordvintsev, N. Pezzotti, L. Schubert, and C. Olah, "Differentiable image parameterizations," *Distill*, 2018. https://distill.pub/2018/differentiable-parameterizations.

[4] D. Ha, "Generating large images from latent vectors," *blog.otoro.net*, 2016.

[5] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[6] A. C. Scott, W. J. Clancey, R. Davis, and E. H. Shortliffe, "Explanation capabilities of production-based consultation systems," *American Journal of Computational Linguistics*, pp. 1–50, Feb. 1977. Microfiche 62.

[7] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[8] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, "Concept bottleneck models," in *International Conference on Machine Learning*, pp. 5338–5348, PMLR, 2020.

[10] R. R. Selvaraju, S. Lee, Y. Shen, H. Jin, S. Ghosh, L. Heck, D. Batra, and D. Parikh, "Taking a hint: Leveraging explanations to make vision and language models more grounded," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2591–2600, 2019.

[11] W. Liang, J. Zou, and Z. Yu, “Alice: Active learning with contrastive natural language explanations,” *arXiv preprint arXiv:2009.10259*, 2020.

[12] P. Lertvittayakumjorn, L. Specia, and F. Toni, “Find: Human-in-the-loop debugging deep text classifiers,” *arXiv preprint arXiv:2010.04987*, 2020.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS’12, (Red Hook, NY, USA), p. 1097–1105, Curran Associates Inc., 2012.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[18] D. N. Perkins, G. Salomon, *et al.*, “Transfer of learning,” *International Encyclopedia of Education*, vol. 2, pp. 6452–6457, 1992.

[19] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, “A comprehensive survey on transfer learning,” *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[20] B. Alsallakh, N. Kokhlikyan, V. Miglani, S. Muttepawar, E. Wang, S. Zhang, D. Adkins, and O. Reblitz-Richardson, “Debugging the internals of convolu-

tional networks," in *eXplainable AI approaches for debugging and diagnosis.*, 2021.

[21] D. Bau, J.-Y. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[22] S. Teso and K. Kersting, "Explanatory interactive machine learning," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 239–245, 2019.

[23] P. Schramowski, W. Stammer, S. Teso, A. Brugger, F. Herbert, X. Shao, H.-G. Luigs, A.-K. Mahlein, and K. Kersting, "Making deep neural networks right for the right scientific reasons by interacting with their explanations," *Nature Machine Intelligence*, vol. 2, no. 8, pp. 476–486, 2020.

[24] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *Technical Report, Univeristé de Montréal*, 01 2009.

[25] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[26] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, pp. 211–222, 2017.

[27] M. T. Ribeiro, S. Singh, and C. Guestrin, ""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, 2016.

[28] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[29] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[30] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Psychology Press, 2005.

[31] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychological Review*, vol. 65 6, pp. 386–408, 1958.

[32] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[33] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[34] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, “Intriguing properties of neural networks,” *arXiv preprint arXiv:1312.6199*, 2013.

[35] A. M. Nguyen, J. Yosinski, and J. Clune, “Understanding innovation engines: Automated creativity and improved stochastic optimization via deep learning,” *Evolutionary Computation*, vol. 24, pp. 545–572, 2016.

[36] A. Nguyen, J. Yosinski, and J. Clune, “Innovation engines: Automated creativity and improved stochastic optimization via deep learning,” 07 2015.

[37] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” *arXiv preprint arXiv:1506.06579*, 2015.

[38] K. O. Stanley, “Compositional pattern producing networks: A novel abstraction of development,” *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[39] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, “Plug & play generative networks: Conditional iterative generation of images in latent space,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4467–4477, 2017.

[40] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, *Explainable AI: A Brief*

Survey on History, Research Areas, Approaches and Challenges, pp. 563–574. 09 2019.

[41] "Regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts." `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A52021PC0206`.

[42] W. R. Swartout, *Explaining and Justifying Expert Consulting Programs*, pp. 254–271. New York, NY: Springer New York, 1985.

[43] D. Gunning and D. Aha, "Darpa's explainable artificial intelligence (xai) program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, 2019.

[44] L. Grosenick, S. Greer, and B. Knutson, "Interpretable classifiers for fmri improve prediction of purchases," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 16, no. 6, pp. 539–548, 2008.

[45] V. Schetinin, J. E. Fieldsend, D. Partridge, T. J. Coats, W. J. Krzanowski, R. M. Everson, T. C. Bailey, and A. Hernandez, "Confident interpretation of bayesian decision tree ensembles for clinical applications," *IEEE Transactions on Information Technology in Biomedicine*, vol. 11, no. 3, pp. 312–319, 2007.

[46] H. Li, Y. Tian, K. Mueller, and X. Chen, "Beyond saliency: Understanding convolutional neural networks from saliency prediction on layer-wise relevance propagation," *Image and Vision Computing*, vol. 83, pp. 70–86, 2019.

[47] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[48] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International Conference on Machine Learning*, pp. 3319–3328, PMLR, 2017.

[49] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

[50] L. Brocki and N. C. Chung, "Concept saliency maps to visualize relevant features in deep generative models," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pp. 1771–1778, IEEE, 2019.

[51] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4699–4711, 2021.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[53] I. Loshchilov and F. Hutter, "Sdgr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[54] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2425–2433, 2015.

[55] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, 1996.

[56] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[57] C. Alessio, "Animals-10." https://www.kaggle.com/datasets/alessiocorrado99/animals10.

[58] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[59] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.