Explainable Transfer Learning for CNNs and Transformers

Forrest Bicker
Harvey Mudd College
fbicker@hmc.edu

Sahil Rane
Harvey Mudd College
srane@hmc.edu

Rohan Subramanian
Harvey Mudd College
rsubramanian@hmc.edu

Abstract

The field of computer vision was first revolutionized by the advent of Convolutional Neural Networks (CNNs) such as AlexNet and ResNet. Even more recently, vision transformers (ViT) have achieved unprecedented levels of accuracy on standard image classification tasks. However, training these models from scratch requires large amounts of data and computational resources. In this work, we first compare transfer learning for AlexNet, ResNet34 and ViT, pre-trained on large datasets such as ImageNet and transferred to smaller image recognition tasks such as CIFAR-10. Next, we use explainability methods Grad-CAM on CNNs and Layer-wise Relevance Propagation on ViT to explain differences in performance. We conclude that transformers were better for fine-tuning on the CIFAR-10 task. Both CNNs and transformers were successfully modified to explain predictions in an intuitive manner. Our results demonstrate the utility of both types of networks and superior performance of vision transformers. We show that explainable predictions are both insightful and feasible. Furthermore, explainable predictions can lead insight into salient features and patterns of errors to inform future attempts to improve performance. Future work could extend these methods to a variety of sensitive computer vision tasks such as medical image analysis, making them trustworthy and transparent to human users.

1 Introduction and related work

Image classification is one of the most fundamental problems in computer vision. Humans are able to process the world around them and recognize a vast variety of objects in different forms, contexts and with limited information. Previous research has sought to replicate these properties of human vision by developing models to recognize objects in images composed of pixels. Machine learning methods for image classification can be used to automate analysis and interpretation of images at large scales, which is highly relevant in medical imaging, surveillance, social media and advertising. In addition to replicating properties of human vision, machine learning methods have the potential to identify structures that are difficult for the human eye to detect.

These architectures must preserve special properties of images including translation invariance, meaning that object features could be located anywhere, and local structures, meaning that the importance of each pixel depends on its neighbors. Before the advent of neural networks, image classification was limited to basic patterns and shapes. Starting in the 1990s, neural networks inspired by the human brain rose in popularity. The first breakthrough in the field was the invention of Convolutional Neural Networks (CNNs) in 1998, which were first applied by LeCun et al. to handwritten digit recognition [LeCun et al., 1989]. While standard neural networks use fully connected layers that process the entire input at once, CNNs use convolutional layers, which apply filters to local areas of the input data. These layers make it particularly effective for processing data with spatial relationships like images, preserving translation invariance and local structures.

With the advent of powerful GPUs and increased computational capabilities in the 2000s, it became feasible to train deeper and more complex neural networks. Microsoft's 2012 AlexNet, which consists of 5 convolutional layers and 3 fully connected layers, improved performance, but it was still difficult to train deeper networks due to the vanishing gradient problem [Krizhevsky et al., 2012]. Residual networks are another type of neural network that use skip connections to jump over some layers, enabling the training of much deeper networks by alleviating the vanishing gradient problem and improving learning. Google's 2015 ResNet combined CNNs with these skip connections, enabling them to train much deeper networks and achieve high accuracy [He et al., 2015]. In this project, we will dive deeper into the workings of ResNet34, which has 34 layers.

The most recent development in the field has been yet another type of network known as transformers. Introduced in 2017, transformers rely on a mechanism called 'attention', which combines the benefits of fully connected and convolutional layers [Vaswani et al., 2017]. This mechanism allows the model to focus on relevant parts of the input while ignoring less relevant parts, making it effective for tasks that involve understanding relationships and long-range dependencies in the data. Transformer architectures have become the state-of-the-art for natural language processing, and recent research demonstrates their effectiveness in computer vision. Vision transformers (ViT) were introduced in 2020 in the paper "An Image Is Worth 16x16 Words" and are rapidly gaining popularity [Dosovitskiy et al., 2020]. We will also investigate the performance of ViT in this project.

We sought to address three main problems in this project. Firstly, we sought to compare the performance of CNNs and transformers on standard image classification tasks and determine if there are noticeable differences. While both CNNs and transformers are more efficient to train than standard neural networks due to parallel processing of input data, training still requires large amounts of data and computational power. We sought to apply a transfer learning technique known as fine-tuning, where a pre-trained model is reused for a similar task, reducing training time and improving performance by leveraging previously learned knowledge. We will investigate the effectiveness of transfer learning for ResNet34 and ViT.

Finally, these models are highly complex and often seen as "black boxes," as it is challenging to understand how they arrive at specific decisions or predictions. As such, it is often unclear to how they why these models sometimes make mistakes, hindering their application to sensitive tasks that require the transparent decisions and trust of users. Previous research has proposed many methods to achieve "explainability" of both CNNs and transformers. For CNNs, previous research has attempted visualization of intermediate layers ("kernels") to understand feature representation [Zeiler and Fergus, 2013]. Another technique is Grad-CAM (Gradient-weighted Class Activation Mapping), which uses gradients of the target class flowing into the final convolutional layer to produce a map of "important" areas of the image [Selvaraju et al., 2016].

For transformers, attention visualization, analogous to kernel visualization in CNNs, can provide insight into which parts of the input the model is focusing on [Abnar and Zuidema, 2020]. Chefer et al. 2021 extend existing methods such as LRP (Layer-wise Relevance Propagation) to transformers [Chefer et al., 2020]. This permits backtracking the model's prediction back to the input level, highlighting important pixels in image classification. In this project, we will apply Grad-CAM to ResNet34 and LRP to ViT to evaluate their effectiveness and lend insight into how different types of models make decisions.

2 Datasets

2.1 ImageNet-21K

ImageNet-21K is a dataset used widely in image recognition and computer vision tasks. It contains 21,000 categories with millions of labeled 224×224 images. The images cover a wide range of objects, animals, scenes, and abstract concepts. We did not work directly with this dataset, but it is relevant to mention since we initialized pre-trained ResNet34 and ViT models with weights from pre-training on ImageNet-21K. It is a common practice to pre-train computer vision models with large datasets such as ImageNet-21K and then use transfer learning for donwstream tasks.

2.2 CIFAR-10

CIFAR-10 is a smaller, more manageable dataset commonly used in machine learning and computer vision for bench-marking image classification algorithms. It consists of 60,000 32×32 color images in 10 different classes. The ten classes are airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. We imported the data from the `datasets` Python library provided by Hugging Face.

We chose this dataset to fine-tune our models on as it has fewer classes that allow us to look at each class in-depth. Due to computational constraints on the server, we used a subset of the dataset. Specifically, we randomly selected 4500 images for the training (90%) and validation set (10%) and 2000 images for the test set. In 1, we see that the class distribution is not explicitly stratified bit fairly balanced in each set.

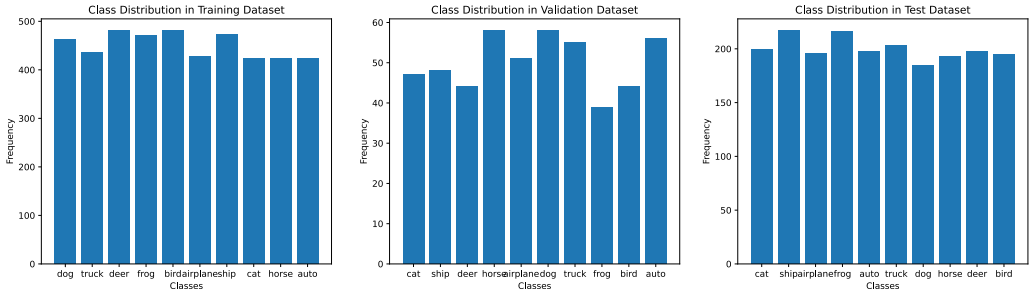


Figure 1: Class distribution in each section of the CIFAR-10 dataset subset.

3 Methods

3.1 ResNet34 Architecture and Fine-Tuning

ResNet34 is a convolutional neural network (CNN) architecture that was introduced in the paper "Deep Residual Learning for Image Recognition" [He et al., 2015]. ResNet was designed to address the challenges of training deep neural networks by using a residual learning framework. The foundational idea is that instead of trying to learn a target function that directly maps from an input x to an output $\mathcal{H}(x)$, ResNet learns the "residual" function $\mathcal{F}(x) = \mathcal{H}(x) - x$ then adds the input back to the output to recover the original function $\mathcal{H} = \mathcal{F}(x) + x$. Doing so helps mitigate the vanishing gradient problem and therefore allows for the training of deeper networks.

More specifically, the foundational residual learning paper presents ResNet34, a 34-layer network for image classification. The network starts with a convolutional layer with a 7×7 kernel to capture large spatial features. Batch normalization and ReLU activation are applied then max-pooling with a 3×3 kernel and a stride of 2 are used to down-sample. There are then several residual stages each consisting of multiple residual blocks. A residual block consists of two convolutional layers with 3×3 filters and a "shortcut" connection that adds the input to the output (to recover the desired mapping $\mathcal{H} = \mathcal{F}(x) + x$). Batch normalization and ReLU activation functions are typically applied after each convolutional layer. A residual stage consists of several residual blocks, and each stage halves the feature map size with more stride 2 convolutional layers before passing to the next stage to

allow the network to learn more abstract features as the input propagates. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax.

To train this network, we start by rebuilding the ResNet34 architecture from "Deep Residual Learning for Image Recognition" and loading the pre-trained model weights the paper presents for the ImageNet 2012 classification dataset with 1000 classes. While these weights were leaned for a different classification problem, we figure the parsing of high-level patterns should transfer across classification problems enough for the weighs to serve as a reasonable initialization.

Since CIFAR-10 only has 10 output classes opposed to ImageNet's 1000, we had to customize the final layer of the ResNet34 architecture to produce the appropriate number of output neurons. We replaced the head of the model with a linear layer of size 10 using Xavier initialization. With Xavier initialization, weights are drawn from a distribution with zero mean and a variance of $Var(W) = \frac{2}{n_{in}+n_{out}}$, where n_{in} and n_{out} are the number of input and output units in the weight matrix, respectively. The final layer uses the sigmoid activation function. We also applied transforms to resize the CIFAR-10 image from 32×32 to 224×224 and normalize the RBG values to have a mean and standard deviation of 0.5, as suggested in the original paper.

From there, we fine-tune the pre-trained weight to specialize our model to CIFAR-10. This is achieved by running gradient descent with Adam optimizer, cross-entropy loss, learning rate of 1e-5 and weight decay of 5e-4 for 5 epochs.

Cross-entropy loss is typical loss function for multi-class image classification. It measures the difference between the predicted probabilities and the one-hot encoded true labels, with the formula $L = -\sum_i y_i \log(p_i)$, where y_i is the true label and p_i is the predicted probability of the class i.

Weight decay, commonly used in training neural networks, is a regularization technique where a small penalty on the magnitude of the weights is added to the loss function to prevent overfitting. The updated loss function with weight decay is given by $L' = L + \frac{\lambda}{2}\|w\|^2$, where L is the original loss function, λ is the weight decay coefficient, and w represents the weights of the neural network. The decay coefficient can be passed to the optimizer.

We started with default hyper-parameters from tutorials referenced, and then manually searched for a good learning rate. When we observed overfitting, we added weight decay to the model and manually searched for good coefficients. We stopped training when there was no decrease in validation loss.

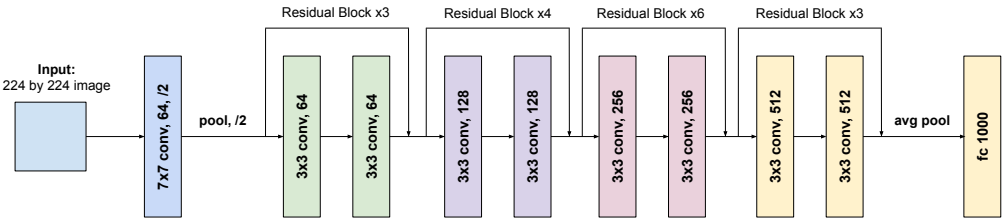


Figure 2: ResNet34 model architecture schematic.

3.2 Vision Transformer Architecture

We briefly explain the different layers in the vision transformer architecture that was proposed in the paper "An Image Is Worth: 16x16 Words" [Dosovitskiy et al., 2020]. First, the RGB image given as input to the neural network is split up into patches based on a patch size. Following this each patch is then flattened to convert it into a vector and then passed through a linear layer for each patch. The output from the linear layer is called the patch embedding and its size is determined by a hyperparameter that is set to 768 for most applications. The positional aspect of the images is maintained using a position embedding as well, which is the same size as the patch embedding. We then have the transformer encoder part of the architecture that has a multi-head self attention layer, a multilayer perceptron, and layer normalization as well as skip connections to help with the training. The attention layer allows the transformer to focus on different parts of the image for each patch as different parts of the image may have different levels of importance for the classification task. Since it is a multi-head attention layer the model is able to focus on different kinds of features or

relationships within the data. This is followed by the multi-layer perceptron layer that is applied for each patch independently and identically (with the same weights). The output is then passed through a final head Linear layer based on the number of classes in the dataset that we are trying to predict, which in our case is 10.

Since we attempt to make at least some comparisons between the fine-tuned ResNet34 and ViT, we kept the data subset and image processing the same. For training, we used stochastic gradient descent with a batch size of 32 and the Adam optimizer for 3 epochs. We were able to achieve satisfactory performance for our model fairly quickly since it was pretrained. We set a learning rate of 2e-5 and had a weight decay parameter of 0.01 for the Adam optimizer.

We adjusted the final head layer to have 10 neurons since the CIFAR-10 dataset has 10 layers. We also performed multiple transformations on the image to convert it into a pytorch tensor so that it could be passed as an input to the neural network. We first resize the image to 224 using torchvision's transforms feature and then convert it to a tensor after which the tensor is normalized based on the mean and standard deviation of the image. The number of layers and the neurons in each layer were taken from the pre-trained model and were not adjusted except for the head layer. The activation function used for the model is the GELU function. The rest of the hyperparameters were manually tuned until we got satisfactory performance from our model as measured by the validation accuracy.

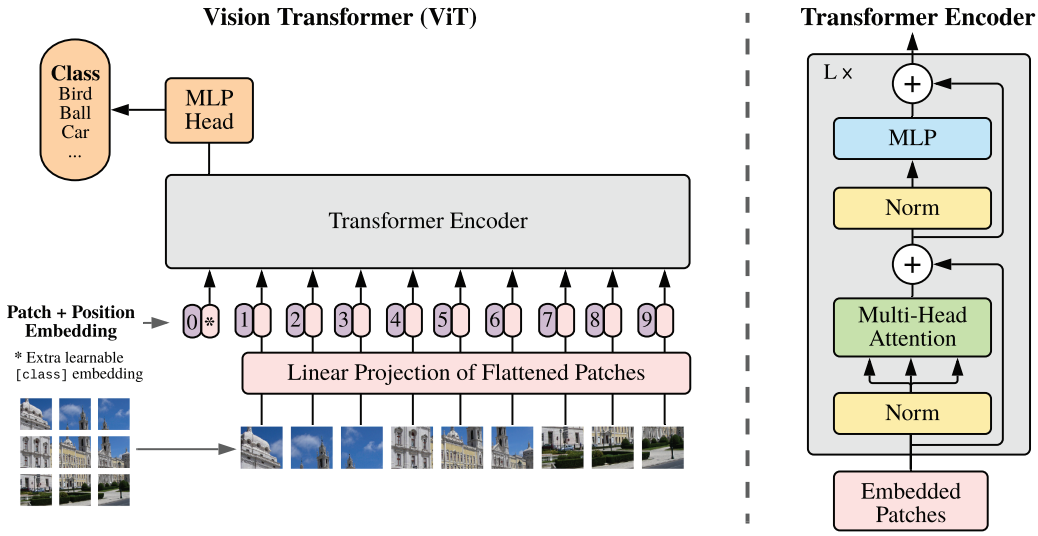


Figure 3: Vision Transformer (ViT) model architecture schematic from the original paper.

3.3 Grad-CAM

To gain some insight into how our ResNet model was working, we decided to use the Gradient-weighted Class Activation Mapping (Grad-CAM). The method was introduced in "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" as a way to visualize what regions of an input image contribute the most to the predictions made by a neural network [Selvaraju et al., 2016]. We used the PyTorch implementation by Jacob Gildenblat [Gildenblat and contributors, 2021].

Such a technique is helpful for model interpretability and transparency. By revealing the specific regions of an image that the model focuses on when making predictions, Grad-CAM allows us to understand what parts of our input have a high impact on the decision-making process to validate the model's behavior and facilitate identifying biases or areas for improvement. Moreover, the visual explanations provided by Grad-CAM can enhance trust in the model's predictions for both research and real-world applications where a clear understanding of the model's inner workings is essential.

On the technical side, to use Grad-CAM we start off with a trained neural network (such as our modified Resnet34 model) and forward pass an image of interest through the network. We then take the gradient of the pre-softmax predicted class score y^c for class c with respect to the feature maps of the a convolutional layer A^k. We then apply a global average pool over the two dimensions of

the gradient matrix to obtain neuron importance weights α_k^c that indicate the the importance of each feature map in making the network's decision. We use these importance values to take a weighted average of our forward activation maps then apply a ReLU to ensure only positive contributions are considered. The result is a map of the relative importance of each region of the image that contributes twoards the network predicting the image belongs to class c.

3.4 Layer-wise Relevance Propagation

To investigate how our ViT model was working, we decided to use Layer-wise Relevance Propagation (LRP). The method was first introduced in 2016 as a method to quantify the relevance of every pixel in image classification [Bach et al., 2015].This method enhances the explainability of neural networks by highlighting the parts of the input that were the most important through the relevance scores. Originally proposed for multilayer neural networks and CNNs, it has recently been extended to transformers by Chefer et al., whose approach we used Chefer et al. [2020]. Most explainability methods for transformers focus on the attention layer and are based on visualization, but LRP provides a more holistic understanding of how the underlying neural network works.

LRP propagates the output probabilities for each class back through the network to understand the parts of the input that are more important for the classification task. A key feature of LRP to note is that during propagation, we have the conservation of relevance through the network. The sum of relevance scores in a layer is equal to the sum of relevance scores in the subsequent layer. The relevance propagation is done using: $R_i^{(l)} = \sum_j \frac{a_i^{(l)} w_{ij}^{(l,l+1)}}{\sum_i a_i^{(l)} w_{ij}^{(l,l+1)}} R_j^{(l+1)}$ where $R_i^{(l)}$ is the relevance of neuron i in layer l, $a_i^{(l)}$ is the activation of neuron i in layer l, and $w_{ij}^{(l,l+1)}$: Weight from neuron i in layer l to neuron j in layer $l+1$.

There are two key features of the ViT transformers that make LRP challenging. Firstly, transformers have skip connections which involve addition. Secondly, they have attention modules involve matrix multiplication. These operations can raise concerns about satisfying the relevance conservation property. Chefer et al. address this problem using normalization technique for the relevance scores that also ensures that the conservation of relevance property is satisfied. Another difference between conventional layer-wise relevance propagation and the technique described in the paper is that the authors also ignore negative weighted activations in their computation of the relevancy scores, which can happen if we have activation functions such as GELU.

We use this method to generate heatmaps on the input images to understand the parts of the image that were most important in predicting the class of the image. The code accompanying the paper Chefer et al. [2020] shows an example of loading a model that has been trained for classification on the ImageNet-21K dataset and how the modified LRP method for transformers can be used to generate relevancy scores for input images. We tweak the architecture of their model to be appropriate for our application to the CIFAR-10 dataset as described in 3.2 and then fine-tune the model using the CIFAR-10 dataset. We then use the modified LRP method for transformers on a variety of different input images from the CIFAR-10 dataset to gain more insight into the parts of the images the network is focusing on.

4 Results and experiments

4.1 CNN Transfer Learning

Due to computational constraints and considerations we chose a random subset of the dataset consisting of 7000 data points. This was split into a training which had 4050 data points, testing with 2500 data points, and validation dataset with 450 data points.

We evaluated the fine-tuned model using the metric of accuracy. In addition to computing overall accuracy, we computed the accuracy for each class and visualized classification accuracy using a confusion matrix. The accuracy of our final fine-tuned model on the testing dataset was 84.3%. For robustness, we also looked at the Area under the ROC curve (ROC-AUC score) to evaluate our model and the ROC-AUC score was 0.986, which is close to the maximum value of 1.

From the confusion matrix (4c), we see that the model has the most trouble distinguishing between cats and dogs, with 15 images of dogs being classified as cats and 21 images of cats being classified

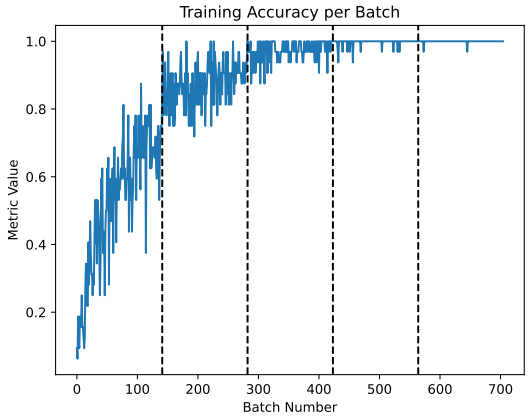
as dogs. The model struggled somewhat to distinguish between an automobile and a truck, which is reasonable given their similar characteristics. Lastly, the model also finds it slightly difficult to classify between birds and airplanes, often mistaking airplanes as birds. Based on the accuracy by class (4d), we are able to see that our model has the lowest accuracy for classifying cats and the highest accuracy for classifying ships. Overall, the model is fairly successful at classifying our images correctly.

Epoch	Training Loss	Training accuracy	Validation Loss	Validation Accuracy
1	1.42	52.44%	0.82	72.34%
2	0.44	87.55%	0.61	80.59%
3	0.19	97.32%	0.54	82.15%
4	0.08	99.71%	0.52	82.34%
5	0.04	99.96%	0.51	83.32%

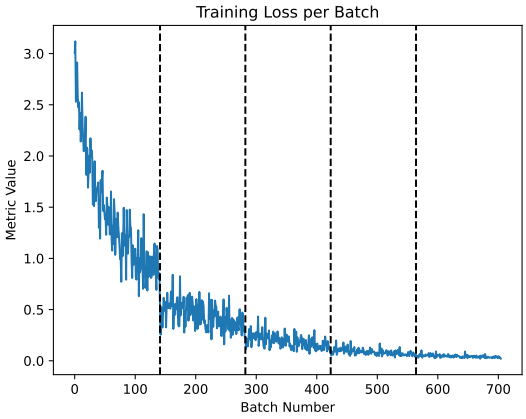
Table 1: Training Loss and Accuracy for ResNet34

As seen in 4a and 4b, the training loss seemed to plateau close to 1 so we decided to restrict the training to 5 epochs in order to prevent the model from overfitting. Based on the training accuracy of 1 towards the later epochs, it may seem like the model was overfitting, but it is important to note that this plot has the training accuracy for the batch of 32 that were randomly chosen from the dataset. Furthermore, the decreasing validation loss and increasing validation accuracy on our validation dataset (1) also shows that our model does not overfit to the training data.

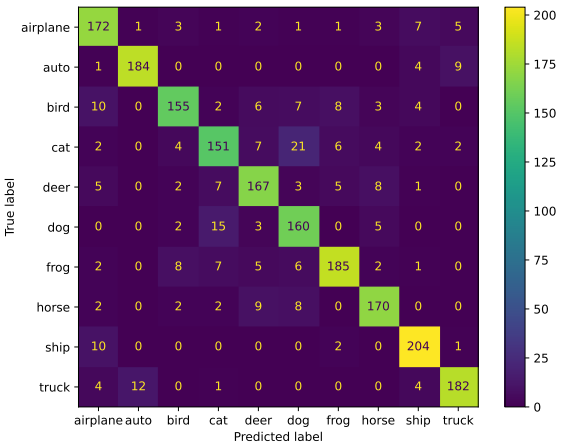
A related but deeper model (ResNet50) has achieved 98.3% accuracy with CIFAR-10 according to one benchmark [PapersWithCode]. We found a Kaggle competition result which was able to achieve 96% accuracy [Lorenzo, 2021] using ResNet34. We attribute this difference despite similar hyperparameter configurations to the smaller subset of CIFAR-10 used due to computational considerations.



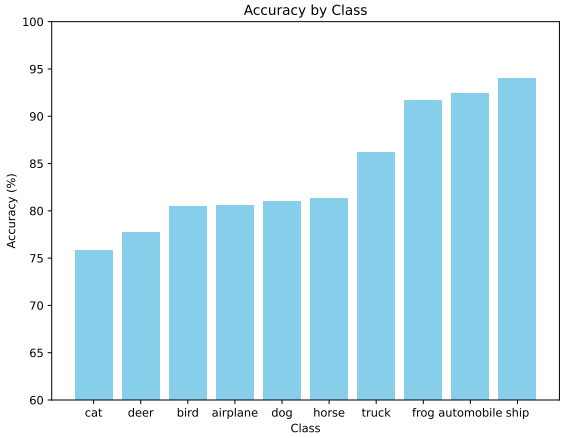
(a) Training accuracy for each batch.



(b) Training loss for each batch.



(c) Confusion matrix for the test set.



(d) Classes sorted by accuracy.

Figure 4: Training and evaluation for fine-tuning ResNet34 on CIFAR-10.

4.2 Vision Transformer Transfer Learning

We now analyze our results for the fine-tuning of the vision transformer in a similar way. We fine-tuned the vision transformer using the same subset and train-validation-test split of the data that we used for ResNet34.

The model was evaluated using its classification accuracy, which was 97.7% on the test set. We achieved a high ROC-AUC score of 0.9996. Both of these metrics are higher than ResNet34 and indicate that the vision transformer that we trained was very good at the classification task on our dataset. Finally, we look at the confusion matrix and the accuracy by class to better understanding the performance of our model for the different classes.

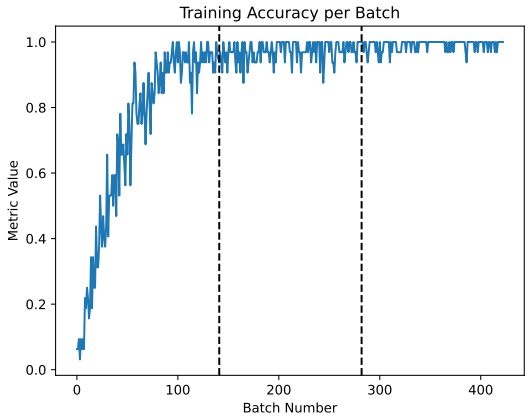
Based on the confusion matrix (5c), most of the values are concentrated along the diagonal, which reflects the high accuracy of the model. Similar to the ResNet34, the model struggles the most with classifying between dogs and cats, albeit less than the ResNet34 model. The least accurate class for the vision transformer is cat and the most accurate class is ship (5d), which is the same as the ResNet34 architecture. This indicates that low accuracy reflects some inherent structure or difficulties of images in the dataset rather than architectural challenges. Despite cats being the hardest to classify for the vision transformer, the model is able to achieve more than 95% accuracy for all classes in the dataset.

As seen in 5a, we are able to achieve high accuracy on our training dataset batches. To ensure that we are not overfitting, we look at the validation and test accuracy. As seen in 2, the validation accuracy is decreasing with epochs thus indicating that we are not overfitting. Furthermore, the 97.7% accuracy on the test dataset gives us greater confidence that we do not have an overfitting problem.

Epoch	Validation Accuracy
1	0.952%
2	0.966%
3	0.974%

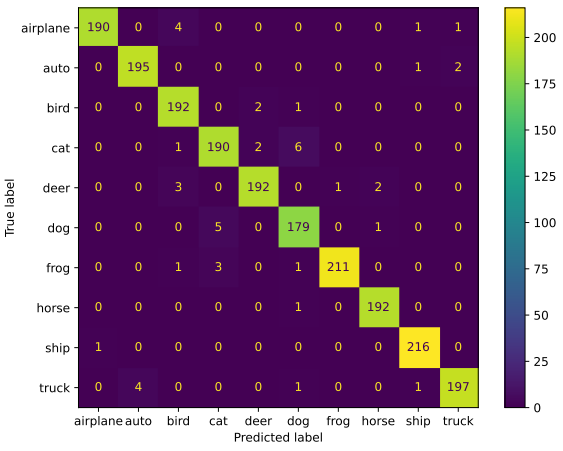
Table 2: Training validation accuracy for ViT

Currently, ViT is the state-of-the art-results for CIFAR-10, achieving an accuracy of 99.9% [PapersWithCode]. We achieved a similar but slightly lower accuracy, which can probably be explained by the smaller subset of the data for computational reasons. We also manually tuned our hyperparameters instead of using a automated technique for optimization largely due to time and computational constraints.

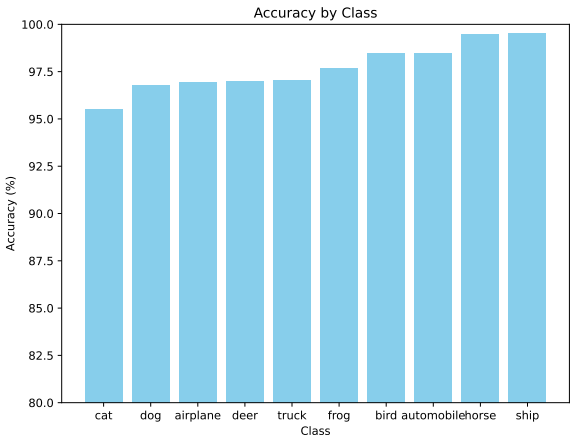


(a) Training accuracy for each batch.

(b) Training loss for each batch.



(c) Confusion matrix for the test set.



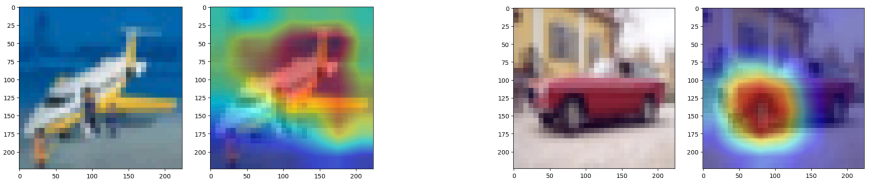
(d) Classes sorted by accuracy.

Figure 5: Training and evaluation for fine-tuning ViT on CIFAR-10.

4.3 CNN Explanability with Grad-CAM

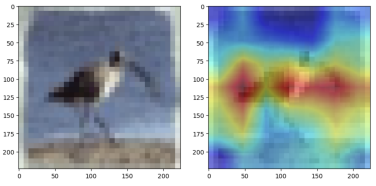
Some intriguing observations emerge when applying Grad-CAM to our model. In 6, we present a correctly classified example image from each class, alongside the image overlaid with Grad-CAM value heatmaps. Notably, the model seems to emphasize the deer's ears in certain classifications (6e). Additionally, it's worth noting that the network considers humans when classifying an image as a horse (6h). Upon reviewing our dataset, we have noticed a substantial number of images containing both horses and humans. This suggests a potential pattern where the network uses humans as a proxy for horse detection. These insights underscore the value of Grad-CAM in unveiling hidden model behaviors. If we aim to deploy this model for a robust application, the insights gained from Grad-CAM may prompt us to scrutinize our dataset for potential biases and clean it to foster more accurate learning, particularly in distinguishing horses from other objects.

Exploring Grad-CAM on misclassified images (7) provides valuable insights into our model's behavior and helps identify patterns in its mistakes. In one instance (7d), our network misclassifies a picture of a horse as a deer, directing its attention to the region where antlers might be. Despite the error, Grad-CAM reveals the model is actively looking at the animal, signaling a nuanced misinterpretation rather than a gross mistake. This highlights the inherent complexity of our dataset, acknowledging that perfect accuracy may be an unrealistic expectation. Examining the cat image (7b) reveals an intriguing pattern where the network overlooks the black cat, focusing instead on the vibrant orange background. This observation prompts us to question why the model disregards certain features. Perhaps it has learned to associate darker colors with shadowy backgrounds, or maybe it is accustomed to using the color orange to identify frogs due to its less common appearance in other animals. Similarly, when classifying the truck (7a), the model confuses the focus of the image, identifying the non-truck element. This recurring pattern suggests a challenge for the model, where it tends to focus on empty or unimportant regions of the image leading to misclassifications. This is opposed to the horse and deer example (7c) where our network seems to accurately identify the subject of the image, but incorrectly assesses the patterns and draws the wrong conclusions. Looking at these detailed analyses through Grad-CAM not only help us understand the model's thought process but also provide crucial cues for refining our dataset and enhancing the model's overall performance.

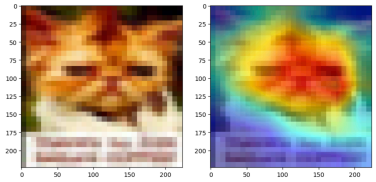


(a) airplane

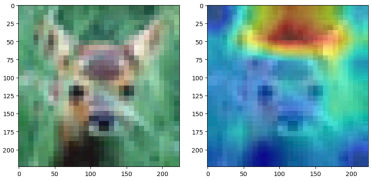
(b) automobile



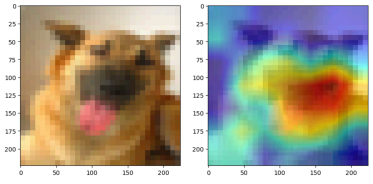
(c) bird



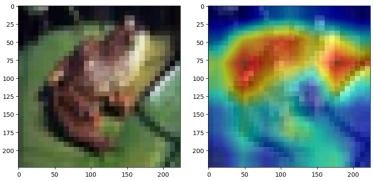
(d) cat



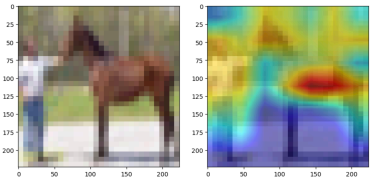
(e) deer



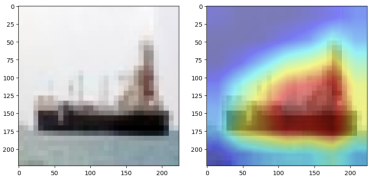
(f) dog



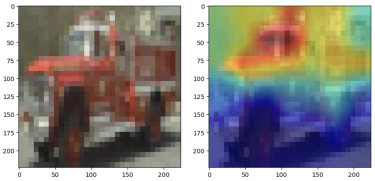
(g) frog



(h) horse

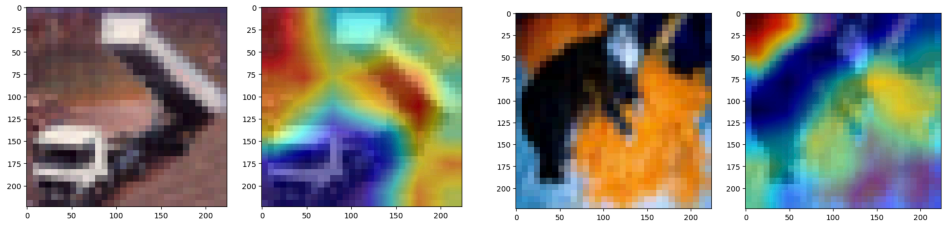


(i) ship



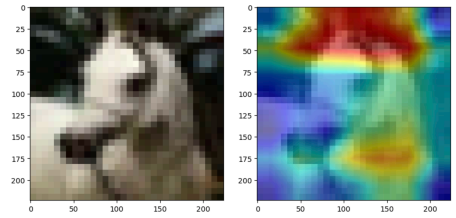
(j) truck

Figure 6: Original CIFAR-10 images correctly classified by ResNet34 alongside image with Grad-CAM value heatmap.

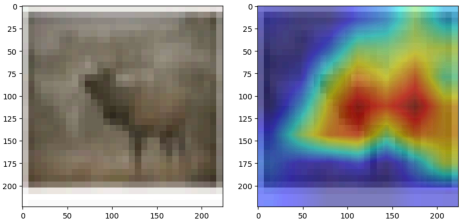


(a) True Class: truck; Pred. Class: plane

(b) True Class: cat; Pred. Class: frog



(c) True Class: horse; Pred. Class: deer



(d) True Class: deer; Pred. Class: horse

Figure 7: Incorrectly classified examples by ResNet34 with Grad-CAM heatmaps.

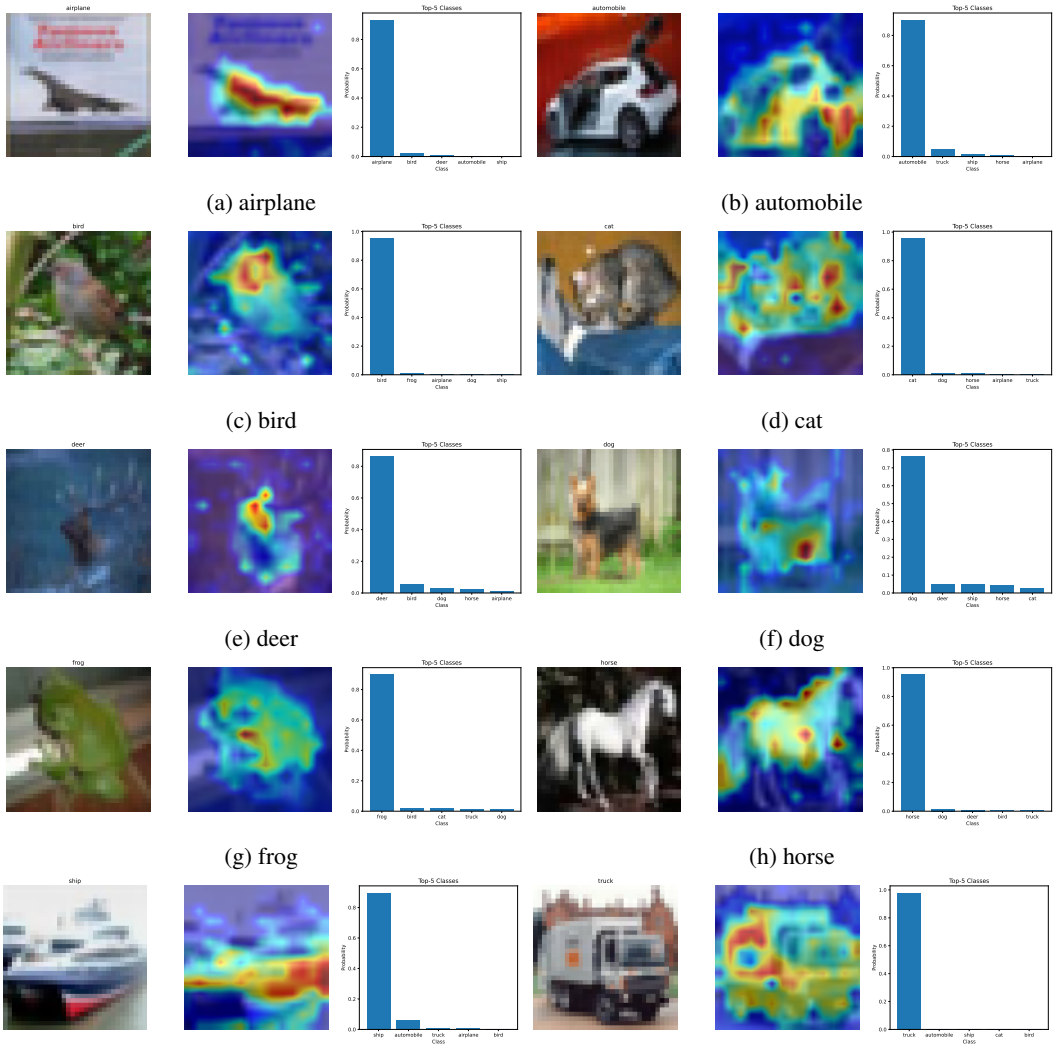
4.4 Transformer Explainability with LRP

Through LRP, we are able to visualize the different parts of the image that the model focuses on. In 8, we are able to see the different patterns in the images that the model has learned through the heatmaps of the relevance scores. For the correctly classified images, the model is successfully able to focus on the foreground and the main object in the image while ignoring the background. This can be seen through the relevance scores largely being concentrated on the object to be classified in the foreground of the image. In addition, the areas of greatest relevance often correspond to salient features that humans also associate with the predicted class. Similar to Grad-CAM for ResNet34, the relevance scores for the deer (8e) seem to focus on the head with the antlers, which are a salient feature of deer. In other cases such as the horse (8h), the highest relevance is along the edges, indicating the importance of the object boundary.

Analyzing LRP output for commonly confused classes lends more insight into which features uses for its decision. Airplanes and birds were previously shown to be the second most confused class. For the airplane (8a), the model seems to focus on the wings, but focuses largely on the beak and head of the bird (8c), which might be a difference exploited by the model to differentiate between the two in this case. The model also showed difficulty distinguishing automobiles and trucks. For the automobile (8b), the model focuses on the entirety of the car body, ignoring the open trunk, which might actually be more effective for correct classification. For the truck image (8j), the relevance scores focus on the wheels and storage container of the truck. Finally, cats are the least accurate class. However, the relevance scores for the cat image are not the most interpretable and the aspect of the image that the model is focusing on is unclear.

The misclassified images (9) lend greater insight into the weaknesses of the model and patterns of they occur. In 9a and 9b, a large amount of relevance concentrated on a very small point in the image. In this case, it appears that the model is unable to correctly distinguish the object from the background as its shape. This could be due to the unconventional top-view angle of both images, an edge case that future improvements could target. Similarly, the dog misclassified as a cat in 9c is shown from an angle that obscures its definitive shape. Another instance where images seem to be misclassified are when multiple labels have similar features. In 9d, the relevance is concentrated on the ears of a cat misclassified as a deer, which could be mistaken for the antlers of a deer. This behavior is shared with ResNet34 and is an inherent difficulty with the dataset.

Overall, our detailed analysis of the LRP heatmaps class allows us to look 'under the hoood' to understand the general patterns of salient features and common errors. This provides specific cases to focus on when improving the model and features of the dataset that are intrinsically complex.



(a) airplane (b) automobile

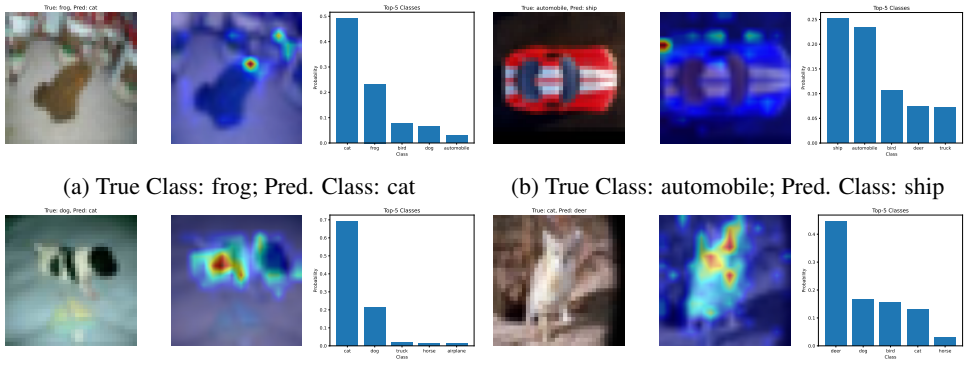
(c) bird (d) cat

(e) deer (f) dog

(g) frog (h) horse

(i) ship (j) truck

Figure 8: Original image correctly classified by ViT, image with LRP value heatmap and bar plot of predicted class probabilities for each image.



(a) True Class: frog; Pred. Class: cat (b) True Class: automobile; Pred. Class: ship

(c) True Class: dog; Pred. Class: cat (d) True Class: cat; Pred. Class: deer

Figure 9: Incorrectly classified examples by ViT with LRP heatmaps.

5 Conclusion and future work

Overall, we were successful in our original aim to apply transfer learning to ResNet34 and ViT for CIFAR-10. We found that the vision transformer model performs very well on CIFAR-10 image classification, achieving an accuracy of 97.7%. This was much better than the 84.3% accuracy from ResNet34. This reaffirms ViT as the state-of-the-art in image classification. Many explanations for its superior performance have been proposed, such as the ability of attention layers to integrate long-range dependencies in images [Dosovitskiy et al., 2020].

We also successfully leveraged transfer learning with pre-training on large datasets such as ImageNet21K. We were surprised by how few epochs (3-5) were required to effectively train both models, indicating that the pre-trained weights conserved useful information from related tasks. In addition, the training time took just 1-2 minutes for both despite limited computational resources.

We then delved deeper into the accuracy and explainability for different classes. Both ViT and ResNet34 struggled to distinguish between the same classes, most of all between dogs and cats. The vision transformer was much better at these task and had >95% accuracy for all classes. These patterns in class accuracy reflect the inherent complexity of comparing certain classes in the dataset as opposed to architectural challenges.

We also tried to better understand what was going on under the hood of each of our models and explored explainability techniques such as Grad-CAM and LRP. Through the explainability methods, we were able to better understand the aspects of the images that our classifiers were focusing on to come up with the predicted labels. This was especially useful to explain patterns in misclassifications. Both ResNet34 and ViT are good at identifying where the subject of the image is and focusing on the subject rather than the background. Both models focused on salient features of each class such as deer antlers. However, Grad-CAM and LRP showed that both models struggled to distinguish classes with similar salient features, such as cat ears and deer antlers.

Grad-CAM also allowed us to infer potentially problematic properties of our dataset. For example, many images with horses also had humans, and Grad-CAM showed that ResNet34 often used the humans in the images to classify those same images as horses. Hence, explainability allowed us to recognize that the model may associate humans and the horse label together. In other applications where this distinction is particularly relevant, this could be a serious concern and would require adjustments to the dataset. This might include deleting such images or adding more images to balance the dataset.

LRP shed additional light on errors made by the vision transformer. Instances where ViT misclassifies images largely stem from a large amount of weight being placed on a very small part of the image and incorrectly identifying where the object is in the image. This was especially evident in images where the object was at a rare angle such as top view. The different shape may have hindered the model from distinguishing the image from the background. Identifying these edge cases could inform future data adjustments or training strategies to improve accuracy.

This project can be extended through a more comprehensive survey and comparison of explainability techniques. Each technique has its own limitations that have not been fully explored in this paper and utilizing a variety of techniques in conjunction with each other to understand each model might be a useful extension. More specifically, we could look into game theory based methods such as Shapley additive explanations and saliency maps for the ResNet34 model. It would also be interesting to compare LRP for both CNNs and transformers. Similarly, we could also present an attention rollout based analysis of the Vision transformer model as an additional explainability method for Vision transformers in conjunction with LRP. We could also use another dataset that has larger images or more datalabels such as the CIFAR-100 dataset or the ImageNet-21K dataset. Finally, we noticed that there is little library support for transformer explainability and future efforts should attempt this implementation.

6 Broader impacts

While computer vision models have reached a high accuracy, often outperforming humans, their application to sensitive tasks that involve human life has been slow due to the lack of explainability and trust. This project has several applications in the field of healthcare, with recent attempts to work

on image classification for a variety of diseases. Notable previous applications have been explaining ML-based tuberculosis diagnosis from lung scans and tumors from MRI data [Chen et al., 2022]. In this domain, the models must be explainable because model decisions have real human impact that we need to be able to trust. Medical experts could validate the robustness by looking at the parts of the image that the model focuses on and validating how reasonable the model's decision making process seems. Thus, our work could have broader impacts where Vision Transformers are used for such classification tasks and LRP can be used to gain a better understanding of the trends that our model has learned. Feature importance in these models could also guide research into explaining the mechanism in these diseases, as models are sometimes able to infer the importance of areas overlooked by humans.

Another important application of our work on interpretability is in identifying model bias. Grad-CAM values were able to show that the model was learning to associate humans to predict horses. In other applications, this could be a serious issue with harmful biases where the model makes hidden assumptions or false correlations between certain classes that we do not want to consider. In this case, we are using a pre-trained model so it is unclear whether the bias orginates from pre-training or fine-tuning. With higher impact models, we might put more focus on using our interpretability techniques to identify bias and its origins.

We trained all models using the CS department teapot server, which is equipped with two NVIDIA GeForce RTX 4090 GPUs. We wanted to be mindful of the shared computational resources that we were using so we used pre-trained models and fine-tuned them on a subset of the CIFAR-10 dataset. This took about 1-2 minutes to run on the teapot server and did not take a significant amount of energy allowing us to tune our hyperparameters more effectively.

7 Code

We include all the PyTorch code to train the models and reproduce the results and figures in a GitHub repository. We also provide our saved fine-tuned models that can be loaded ready for use. Additional code to fine-tune AlexNet and train transformers using Hugging Face libraries is included as helpful references but not used directly for this paper.

`https://github.com/SahilRane/NeuralNetworkExplainability.git`

8 Video

A video presenting our work can be found as an unlisted YouTube video at the link: `https://youtu.be/zvQ4JWh6tQk`.

References

Samira Abnar and Willem H. Zuidema. Quantifying attention flow in transformers. *CoRR*, abs/2005.00928, 2020. URL `http://dblp.uni-trier.de/db/journals/corr/corr2005.html#abs-2005-00928`.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10, 2015. URL `https://api.semanticscholar.org/CorpusID:9327892`.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. *CoRR*, abs/2012.09838, 2020. URL `https://arxiv.org/abs/2012.09838`.

Haomin Chen, Catalina Gomez, Chien-Ming Huang, and Mathias Unberath. Explainable medical imaging ai needs human-centered design: Guidelines and evidence from a systematic review. *npj Digital Medicine*, 5(1), 2022. doi: 10.1038/s41746-022-00699-2.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020. URL `http://arxiv.org/abs/2010.11929`.

Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

Francesco Lorenzo. Fine-tuning resnet34 with pytorch, Aug 2021. URL https://www.kaggle.com/code/francescolorenzo/96-fine-tuning-resnet34-with-pytorch.

PapersWithCode. Cifar-10 image classification benchmark. URL https://paperswithcode.com/sota/image-classification-on-cifar-10?tag_filter=3.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization, 2016. URL http://arxiv.org/abs/1610.02391.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. URL http://arxiv.org/abs/1311.2901. cite arxiv:1311.2901.