Analysis of Machine Learning Models for Image Classification on Cifar-10

Mogage Nicolae – ICA 246/1

# Abstract

Computer vision is the ability of computers are able to identify, recognize and classify objects from images. This subject raises problems in the machine learning domain due to the high number of objects to exist, each one having a different version. For example, we may have a long pink car and a short blue one, but we identify them both as cars. In this manner, various research has been done to improve the databases and methods used in the industry. Some examples for the former case are ImageNet and Cifar datasets, which consists of millions, respectively thousands, images collected from real life. The Cifar one has 2 variations, one with 10 target classes, and the other with 100, where we opted for the first one. Next, the later one include various machine learning algorithms and combinations of them, but we chose only 5, which are: Convolutional Neuronal Networks (CNN), Deep Residual Learning Networks (ResNet), Densely Connected Convolutional Networks (DenseNet), Vision Transformers (ViT) and Convolutional Vision Transformers (CvT). All the models were trained from scratch, except for ViT where we used the pretrained model, and for the ResNet and DenseNet models, we also tested with pretrained on ImageNet dataset versions of them. With an accuracy of 86.17%, the Residual Network algorithm pretrained on ImageNet1K, which is a variation of the main dataset, showcased the most promising results and confirmed the high capabilities of ResNet. DenseNets and CNNs also showed performances closed to the ResNet ones, however the others score significatively lower due the not being pretrained at all or on a large enough dataset.

**Table of Contents**

[Abstract 1](#_Toc185289485)

[1. Introduction 2](#_Toc185289486)

[2. Cifar-10 Dataset 2](#_Toc185289487)

[3. Methods 3](#_Toc185289488)

[Convolutional Neuronal Networks 3](#_Toc185289489)

[Deep Residual Learning 4](#_Toc185289490)

[Densely Connected Convolutional Networks 5](#_Toc185289491)

[Vision Transformers 7](#_Toc185289492)

[Convolutional Vision Transformers 8](#_Toc185289493)

[4. Experiments 9](#_Toc185289494)

[Setup 9](#_Toc185289495)

[Results 10](#_Toc185289496)

[Discussion 12](#_Toc185289497)

[5. Conclusions 12](#_Toc185289498)

[6. Bibliography 13](#_Toc185289499)

# Introduction

A technology that is gaining and gaining momentum is computer vision, which allows computer to identify, recognize and classify objects from images. A challenge often encountered is the lack of training data on which the machine learning algorithm can learn from. Various research has been done in this domain, thus resulting in more reliable real-life examples, such as the ImageNet dataset, which consists of around 14 million pictures. Moreover, not only the data had improved, but also the methods that are being used, especially with the appearance of transformers model, which can be seen as a breakthrough due to their high capacity of learning and distinguishing complex patterns.

In this research we chose the cifar-10 dataset to properly evaluate the proposed methods. Furthermore, the models we selected to test are: Convolutional Neuronal Networks (CNN), Deep Residual Learning Networks (ResNet), Densely Connected Convolutional Networks (DenseNet), Vision Transformers (ViT), and a variance of the last mentioned, Convolutional Vision Transformers (CvT). These were chosen based on their high accuracies obtained on popular benchmarks. In addition, we also made a test for ResNet and DenseNet with their pretrained models on the ImageNet database.

This report details the used methodology, the experiments setup and their results, as well as a discussion of areas for future research. Understanding how these methods work, and their limitations, can help us to better contribute to the machine learning research domain. Moreover, the survey can be used as a starting point on what computer vision algorithms are the fit solution or not for one’s problem.

# Cifar-10 Dataset

This database was originally introduced by Krizhevsky et al. in Learning multiple layers of features from tiny images [1]. It consists of millions of tiny color images from the web, which were downloaded from the Internet based on 53,464 different nouns, directly copied from Wordnet [1]. The authors' initial objective was to use it for deep generative models' unsupervised learning. However, this showed as a problematic aspect for the object recognition experiments because there are not any reliable class labels. Therefore, the Cifar-10 and Cifar-100 [2] were created, where the former has 6000 examples of each of 10 classes and the later consists of 600 examples of each of 100 non-overlapping classes. In this manner, we chose the first dataset for complexity matters, as the models trained on this set requires fewer parameters and smaller memory overhead, making it accessible to a large audience. The motivation to go for this database instead of other existing possibilities was that it has enough complexity to properly evaluate the tested machine learning algorithms, and it has enough samples to create the possibility for the models to learn but also is small enough to be able to train on less performant systems. For example, the MNIST handwritten digit set was far too simple for the selected methods, as accuracies of over 95% were achieved with ease. On the other hand, datasets like ImageNet can be too large and too complex for a simpler network to train with limited resources.

The Cifar-10 dataset is a popular benchmark in the machine learning field, especially on computer vision, being used for tasks such as image classification, generation or neuronal architecture search. The authors split the 60,000 color images into a training set of 50,000 images and a test set of 10,000 samples. We divided the data even further, so out from 50,000 records, a validation set with 10,000 pictures was created, and the training set was reduced to a number of 40,000. These images are of size 32x32 pixels and in RGB format, meaning that each picture has 3 channels, red, green, and blue. The 10 classes included in the research are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. Moreover, the labels of this dataset are completely mutually exclusive, ensuring that there is no overlap between them. For example, "Automobile" includes sedans, SUVs, etc., while "Truck" includes only big trucks, and neither includes pickup trucks. Some examples of each class can be seen in the Figure 2.1.

A collage of dogs

Description automatically generated

Figure 2.1: Cifar-10 samples example

# Methods

For a better comprehension of this research and improved results in the experiments phase, we made an analysis of each method and how they work on a deeper level. This helped us to properly process the dataset before feeding it to the network, slightly impacting the overall performance of the model. Moreover, it also contributed to the fine-tuning phase of the hyperparameters. Details of each method will be discussed in the following subsections to motivate the choose of each one and their contribution.

## Convolutional Neuronal Networks

CNNs are a specialized type of Artificial Neuronal Network (ANN), which were designed to work efficiently with 2-dimensional data, such as images and time-series signals. These networks are fundamental in the computer vision domain and have become useful for image recognition, object segmentation, image classification and so on. They outperform other AI models in standard learning tasks, particularly in image recognition. In addition, the architecture of CNNs was specially created to handle images efficiently, reducing the computational demands of pictures processing [3]. Moreover, by reducing the number of parameters required and using techniques like pooling, these networks minimize the risk of overfitting, which occurs when a model gets good results during training but performs poorly on unseen data [3]. Another use for convolutional neuronal networks is feature extraction, where they only focus on selectin important features from images and other employed model does the classification.

The base structure is constructed from different layers that work together to extract features, reduce dimensionality, and classify the input image data [3]. A simplified architecture includes:

1. The convolutionallayer, which is responsible for extracting features from the input image by applying a series of filters, called kernels, on small regions of the picture. Each kernel is a small filter that slides across the whole image, computing the dot product between its weights and the corresponding region. This generates activation maps that highlight specific features.
2. After each convolution, an activation function is applied, rectifiedlinearunit (ReLU) being among the most often used ones, in order to introduce non-linearity into the model. This function aims to apply an elementwise operation, in order to create the possibility for the model to use more complex characteristics.
3. Next, the poolinglayerare used to reduce spatial dimensionality of the convolutional layers, resulting in a fewer number of parameters and less computational complexity. The most common one is max pooling, which selects the maximum value from a small region (usually 2x2) of the feature map.
4. Finally, after multiple convolutional and pooling layers, the 2-dimensional data is transformed into linear vectors and fed into one or multiple fullyconnectedlayers to obtain the final output. These layers operate similarly to traditional ANNs.

A visualization of these layers can be seen in the Figure 3.1. The example makes use of the MNIST dataset, where the input are handwritten digits.

A diagram of different types of objects

Description automatically generated

Figure 3.1: CNN Architecture for image classification [3]

## Deep Residual Learning

Residual Networks were introduced originally by Kaiming He et al. [4] and are a revolutionary deep learning architecture that addresses the degradation problem in very deep convolutional neuronal networks. This problem happens when more layers beyond a certain depth are added to a CNN, resulting to higher training and validation errors. The main innovation brought by ResNet is the introduction of shortcut connections, which allow the information to skip one or more layers.

The network is created using residual blocks as the fundamental units, which can be seen in the Figure 3.2. These blocks consist of stacked convolutional layers with a shortcut connection that add the input information directly to its outputs [4]. The information transfer in a ResNet is done in two ways depending on whether the input and output dimension are the same. If they are equal, the network uses identity mappings, which means that the input is passed unchanged to the output. Otherwise, a linear projection is applied within the shortcut to match the output layer, usually through a 1x1 convolution [4]. Moreover, for the architectures with a higher number of layers (e.g. 101, 152), a bottleneck design is used to reduce computational complexity. In this case, each residual block is formed using three layers, a 1x1 convolution to reduce dimensionality, a 3x3 one as the bottleneck with smaller inputs, respectively outputs dimensions, and a 1x1 convolution to increase back the dimensionality to its original value. In the Figure 3.3 the difference between these two types of blocks can be visually observed [4]. Finally, ResNet architecture is formed by stacking sequentially multiple building blocks.

A diagram of a diagram

Description automatically generated

Figure 3.2: A building block [4]

A black and white image of a circle

Description automatically generated

Figure 3.3: Different residual blocks. Left: a building block for ResNet-34. Right: a bottleneck block for ResNet-50/101/152 [4]

Residual networks successfully address the vanishing gradient problem, which appeared often when training deeper architectures. This has allowed to increase the model’s depth, without the worry about degradation, which resulted in the ResNet achieving significantly higher accuracy on various image recognition tasks compared to shallower networks. Another benefit brought by ResNet is the excellent generalization ability, performing well on other tasks as well, such as object segmentation or image classification. These achievements had made this type of architecture to become a baseline model for benchmarking new algorithms in computer vision.

## Densely Connected Convolutional Networks

The research conducted by Gao Huang et al. [5] introduces a new type of architecture in the computer vision domain, the densely connected convolutional networks. This is a novel convolutional neuronal network that focuses on maximizing information flow between layers by directly interconnecting each layer in a feed-forward fashion. Therefore, while traditional CNNs with L layers have L connections, the DenseNet has direct connections. In the same way as for the ResNet where the main building blocks are the residual blocks, in this network the core components are the dense blocks, where the dense connectivity principle is applied. In this block, each layer receives the feature maps from all previous layers as input, and its own feature maps are used as input for all the following layers [5]. Moreover, in contrast to ResNets, in which the features are combined through summation before being passed into a layer, this network uses concatenation to accomplish the same task. In the Figure 3.4 is visually represented a 5-layered dense block that illustrates the layout schematically. To facilitate downsampling and handle the growing dimensionality of feature maps due to concatenation, transition layers are added between each dense block [5]. The most common techniques employed in this part are convolution and pooling operations, which reduce the spatial dimensions of the feature maps. Figure 3.5 illustrates a DenseNet with 3 dense blocks and transition layers between them. Another significantly important hyperparameter which was added in DenseNet is the growth rate (k). This variable is used to determine the number of new feature maps each layer are contributing with to the network’s overall knowledge. Experiments show that a relatively small growth rate (e.g. k=12) is enough for the model to obtain state-of-arts results [5]. In addition, similar to residual networks, dense networks can make use of the bottleneck layers in an efficient way to reduce computational complexity [5]. Referred as DenseNet-B, they introduce a 1x1 convolution before each 3x3 convolution to decrease the number of input feature maps. Furthermore, to better enhance the model compactness, the number of feature maps are reduced at transition layers [5]. A compression factor controls this reduction, where if a dense block contains feature maps, the transition layers generate output feature maps [5]. For the cases where , the networks are denoted as DenseNet-C, and if a bottleneck layer is also used, the authors refer to the model as DenseNet-BC.

By providing direct connections from layer to layer, dense convolutional neuronal networks also effectively address the vanishing gradient problem, which makes possible the training of even deeper architectures. Additionally, this architecture encourages feature reuse throughout the model and strengthen feature propagation [5]. Another important advantage of this algorithm is that it substantially reduces the number of parameters in the network, resulting in improved computational efficiency.

A diagram of a network

Description automatically generated

Figure 3.4: A 5-layered dense block with growth rate of k=4 [5]

A diagram of a block diagram

Description automatically generated

Figure 3.5: A DenseNet architecture with 3 dense blocks [5]

## Vision Transformers

In the work of Alexey Dosovitskiy et al. [6], the concept of vision transformer is firstly introduced, which is a novel deep learning approach that applies the transformers architecture, originally designed for natural language processing (NLP), directly to image recognition. The authors show that reliance on convolutional neuronal networks for computer vision tasks is not necessary, and a pure transformer can perform as well, if not better. Other attempts were done previous in order to incorporate transformers in computer vision, but they relied on hybrid architectures, which combined CNNs with self-attention, or used transformers to replace specific components while maintaining the convolutional structure. What makes ViT a significantly strong candidate in the image classification tasks is the power of pretraining, like other NLP models. When pretrained on large image datasets like ImageNet-21K or JFT-300M, ViT outperforms classical CNN approaches on various image recognition benchmarks. In addition, this method requires substantially fewer computational resources to train compared to state-of-art convolutional networks, while also keeping excellent results compared to them [6].

The vision transformer processes images by converting them into a series of patches, the same as words in a sentence. The ViT structure consists of multiple steps. Firstly, the input image is divided into fixed-size patches, usually 16x16 pixels. Afterwards, each picture patch is flattened and linearly projected into a lower dimensional space using a trainable linear projection, resulting in a sequence of patch embeddings. Similar to BERT’s class token, the authors prepend a learnable embedding to the sequence of patches, whose state at the output of the transformer servers as the image representation [6]. Moreover, position embeddings are added to retain positional information. The resulted sequence of embedding vector is fed into a standard transformer encoder as input. This encoder consists of multiple alternating layers of multi-head attention (MHA) and multi-layer perceptron (MLP) blocks, with batch normalizations between them. The model is able to capture connections between different patches throughout the self-attention mechanism, enabling global information communication. Non-linear transformations and feature extractions are provided by the MLP blocks. Finally, a MLP Head is added to perform classification, which receives the output of the encoder, more specific the state of the class token. Below in the Figure 3.6 the model’s overview can be observed.

A diagram of a transformer

Description automatically generated

Figure 3.6: Architecture of a Vision Transformer [6]

## Convolutional Vision Transformers

Lastly, with the purpose to combine the strengths of both convolutional neuronal networks and vision transformers for image recognition tasks, Haiping Wu et al. introduce the convolutional vision transformer architecture [7]. Through this addition, the model is able to capture local spatial context while maintaining the global context and generalization capabilities of transformers. As visually illustrated in the Figure 3.7 (a), CvT’s structure consists of the following components:

1. A hierarchical multi-stage structure inspired by the CNN architectures, with a total number of 3 stages. This structure facilitates a progressive downsampling of feature maps and an increase in token feature dimensions.
2. In the begging of each stage, a convolutional token embedding layer performs a convolution with overlapping patches of tokens reshaped to the 2D spatial grid as input [7]. Another normalization layer is applied to the tokens. This operation reduces the number of tokens while increasing the token feature dimension, achieving spatial downsampling and richer feature representation [7].
3. At each stage, a stack of convolutional transformer blocks is used, which replaces the position wise linear projection with a convolutional projection layer. Implemented using a depth-wise separable convolution, this module enhances the model’s ability to capture local spatial context. Moreover, this level also improves computational efficiency by reducing the number of parameters and FLOPs required compared to standard convolutions.
4. Similar to the vision transformer algorithm, a classification token is used, which is added only in the last stage, and a MLP Head is applied to the final classification token output.

By incorporating convolutions, CvT effectively the combines the benefits of each concept. Therefore, convolutional vision transformer consistently outperforms traditional methods for both CNN-based models, such as ResNets or DenseNets, and previous transformer approaches, like ViT or DeiT, on the ImageNet benchmarks. In addition, introducing desirable properties of CNN to the ViT architecture, such as shift, scale, and distortion invariance, it makes the CvT more robust to variations in images. Another benefit is the elimination of positional embeddings, simplification that allows this architecture to be more flexible and adaptable to tasks with different input resolutions.

A diagram of a diagram

Description automatically generated

Figure 3.7: The pipeline of CvT architecture [7]

# Experiments

In the following sections we will cover the setup made for the problems, the results we got during and after training, as well as a discussion on the performance of each model and future research.

## Setup

Firstly, to prepare the data we tried multiple techniques, such as flipping the image, adding a padding on each size of 4, or normalization. After multiple tries and combination, we concluded that only adding a normalization function for each channel leads to the best performance of the model. Thus, we applied it with the mean values of 0.4914, 0.4822, 0.4465 and with the standard deviation values of 0.2023, 0.1994, 0.2010 for the red, blue and green channels. The mathematical formula applied maps the data to be centered around 0, which is given by the equation .

The loss function used for the experiments is the categorial cross entropy [8], where bigger discrepancies in outputs results in greater losses. The mathematical formula is given by the equation 4.2, where indicates whether example i belongs to class j, is the prediction for class j for example i, N is the number of examples, and C is the number of classes. Moreover, adaptive moment estimation (Adam) [9], mathematically represented by the formula 4.3, is used as the optimization function in all cases. This chose was done because of the convergence speed of the function. Therefore, a fewer number of epochs is necessary to achieve almost the same result than using other methods like SGD or AdaGrad. However, the cost of this speed is that the result we obtain is not the best, this choice being done based on the available resources. To avoid the case where the model does not converge to a solution, we also introduced a scheduler to decrease the learning rate with 0.3 on the 6th and 10th epochs.

4.2

4.3

Next, a description for the architecture of each model is given. Firstly, in the case of convolutional neuronal networks, we use 4 convolutional layers, each one has the corresponding (input, output) channels pair: (3, 64), (64, 128), (128, 256), (256, 512). On top of these is added a batch normalization layer, which proven effective in improving the model’s performance. As activation function, we opted for ReLU as recommended in [3]. The pooling module selected is the Max Pooling with a size of 2x2. In the end, 4 fully connected layers are added to make the classification. Next, in the case of ResNet, DenseNet, and ViT we used the implemented method from PyTorch, a popular python library used in machine learning. We selected ResNet-50, which for the tests was pretrained on the ImageNet1K of the second version dataset, DenseNet-161 and ViT-B-32, these being pretrained on the first version of ImageNet1K. The motivation for these options were based on the limitations of the hardware used in training, these being the most performant version of the original architectures. Lastly, the CvT model is implemented in the same manner as presented in the research [7]. The configuration is provided by the next list depending on each stage: embedded dimension=[64, 128, 256], transformer heads=[1, 2, 4], transformer blocks=[1, 2, 3], patch sizes=[3, 3, 3], strides=[2, 2, 2], and paddings=[1, 1, 1].

Finally, the metrics used are accuracy, f1 score, and ROC AUC score. These measurements allow us to properly evaluate the method and draw the correct conclusions. For the third metric, a score of 0.5 indicates random guessing, and a score of 1 indicates perfect performance. During the training, the datasets were split into batches of 32 and the initial learning rate set was 0.001.

## Results

The training process’s results can be seen in the Figure 4.1. We trained the models for 15 epochs in order to test what tendencies each model has for overfitting, and observer that around 10 epochs are enough in order to obtain the best performance. The ViT architecture is the most affected, which may be the caused by using the base model with 32 patches, and also not being pretrained on a large enough dataset.

A graph of different colored lines

Description automatically generated

Figure 4.1: Training metrics

Based on the chosen metrics, we obtained the results shown in the Table 4.1. The pretrained ResNet50 outperforms all the other models, with pretrained DenseNet and CNN coming close to achieve over 80% accuracy. Having ROC-AUC scores close to 1 for all methods demonstrated the high capabilities these methods have in the computer vision domain.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Accuracy** | **F1-Score** | **ROC-AUC Score** |
| **CNN** | 83.71% | 0.8372 | 0.95 |
| **ResNet50** | 68.18% | 0.6797 | 0.9151 |
| **Pretrained ResNet50** | **86.17%** | **0.8618** | **0.9551** |
| **DenseNet161** | 79.12% | 0.7921 | 0.9434 |
| **Pretrained DenseNet161** | 84.87% | 0.8484 | 0.9541 |
| **CvT** | 68.70% | 0.6859 | 0.9213 |
| **ViT-B-32** | 56.59% | 0.5648 | 0.8741 |

Table 4.1: Experiments results

Moreover, we constructed the confusion matrices for each one of the models to better visualize the obtained results. As it can be seen, the most mistakes were for the dog-cat and the automobile-truck classes, which occurs in all methods, proving the difficulty of the dataset with 32x32 images.

|  |  |
| --- | --- |
| A graph with blue squares and black text  Description automatically generated   1. CNN | A graph with blue squares and white text  Description automatically generated   1. ResNet50 |
| A graph with numbers and a bar chart  Description automatically generated with medium confidence   1. Pretrained ResNet50 | A graph with numbers and a number in the center  Description automatically generated with medium confidence   1. DenseNet161 |
| A graph with blue squares and black text  Description automatically generated   1. Pretrained DenseNet161 | A graph with numbers and a triangle  Description automatically generated with medium confidence   1. CvT |
| A graph with blue squares and black text  Description automatically generated   1. ViT | |

Figure 4.2: Confusion Matrices after test results

## Discussion

The results we got showcase the difficulties which the Cifar-10 dataset presents, making it an excellent benchmark for newly developed architectures. The performance of transfer learning via the pretrained models was also highlighted. For example, the ViT network points the importance of requiring a training on a large dataset before for it to obtain acceptable results, having the lowest performance in this research. Moreover, the CvT model, which was not fed previous with a larger dataset, have obtained better metrics, proving the capacity of CNNs in the computer vision domain. Another example is between the ResNet50 and DenseNet161, where if we compare the trained from scratch versions, we observe an outstanding improvement, of over 10% accuracy, showing the importance of adopting a feed-forward approach. However, because the former method is pretrained on a better dataset than the later one, it has succeeded to surpass it only with a minor difference.

Future work could explore even more preprocessing methods to better enhance the datasets used in the training phase. Moreover, other benchmarks like Cifar-100 or STL-10 can be employed to further test the performances of the models. Another important research direction is pretraining all the models with larger datasets, such as ImageNet-21K or JFT-300M, and compare the results on various smaller sets. Even though this requires a lot more computational power, it can provide valuable insights on how each method works better and which should one choose to use in real life scenarios.

# Conclusions

The study was made to attempt, assess and compare various machine learning approaches in terms of performance over the popularly known Cifar-10 image dataset benchmark in the area of computer vision. The models chosen for this study include CNN, ResNet, DenseNet, ViT, and CvT, which represent both classical and modern approaches to image classification tasks. Our experiments included training models from scratch as well as pretrained versions of ResNet, DenseNet and ViT to investigate the impact of transfer learning on performance. The results provide valuable insights into the strengths, limitations, and applicability of these techniques for small-scale yet challenging datasets like Cifar-10.

Therefore, the improvements show that transfer learning greatly increases the performance of the models. The pretrained ResNet50 model attained the highest accuracy of 86.17%, clearly showing how well it generalizes to smaller datasets after pretraining on a larger image dataset like ImageNet. The finding emphasizes the power of residual learning to address the degradation problem in training deeper networks effectively. DenseNet also performed outstanding with a pretrained version of 84.87%, confirming its strength behind attracting the reuse of features and efficient gradient propagation. The traditional CNN achieved a strong performance among models trained from scratch with an accuracy of 83.71%. The CNN, which is simple in architecture compared to ResNet and DenseNet, managed to demonstrate effective usage of convolution and pooling layers to capture relevant features in the Cifar-10 dataset. The performance shows that even today, CNNs are still relevant to image classification tasks, especially in cases when resources are limited.

On the other hand, the Vision Transformer (ViT) struggled to match convolution-based architecture performance and could only achieve an accuracy of 56.59% when pretrained with a smaller dataset. This accentuates the importance of pretraining for transformer-based models. Whereas CNNs are fit to learn space hierarchies from small datasets, large-scale data are required for discovering significant patterns in transformers through the self-attention mechanism. The relatively poor performance of ViT indicates very clearly how the patch approach fails with lower resolution and smaller data samples datasets. Furthermore, the Convolutional Vision Transformer (CvT) provided a better performance compared to ViT and achieved an accuracy of 68.70%. With the combined convolutional operations and transformer architecture, CvT efficiently captures both local spatial context and interdependencies among data globally. Moreover, CvT shows better performance when compared to hybrid alternatives that utilize both convolutional networks and transformers.

In addition, the experiments revealed some of the unique challenges that the Cifar-10 dataset poses. The pixel resolution of 32x32 and the availability of visually similar classes such as dogs and cats, or automobiles and trucks, remained significantly difficult across all models, which is also reflected in their confusion matrices. Even for models like ResNet and DenseNet that outperformed the others, these errors were persistent, meaning a need for better preprocessing methods.

# Bibliography

|  |  |
| --- | --- |
| [1] | A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," University of Toronto, 2009. |
| [2] | A. Krizhevsky, V. Nair and G. Hinton, "Cifar-100 and cifar-10 (canadian institute for advanced research)," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed 15 December 2024]. |
| [3] | K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks,* 2015. |
| [4] | K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. |
| [5] | G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *2017 IEEE Conference on Computer Vision And Pattern Recognition (CVPR)*, 2017. |
| [6] | A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houldsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2021. |
| [7] | H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan and L. Zhang, "CvT: Introducing Convolutions to Vision Transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. |
| [8] | A. Mao, M. Mohri and Y. Zhong, *Cross-Entropy Loss Functions: Theoretical Analysis and Applications,* 2023. |
| [9] | D. P. Kingma and J. L. Ba, *Adam: A Method for Stochastic Optimization,* 2017. |

This work is the result of my own activity, and I confirm I have neither given, nor received unauthorized assistance for this work.

I declare that I did not use generative AI or automated tools in the creation of content or drafting of this document.