Machine Learning Technical Paper

A
Apoorva Chittimoori
AXC190091
axc190091@utdallas.edu
Computer Science

B
Elliana Suzanne Brown
ESB190002
esb190002@utdallas.edu
Computer Science

May 10, 2023

Abstract (Author A: 60%, Author B: 40%)

This report presents a comparative analysis of two popular Convolutional Neural Networks (CNNs) - AlexNet, VGG-16 and a newly proposed CNN on three benchmark datasets: MNIST, Fashion-MNIST, and CIFAR-10. The purpose of this study is to evaluate how well these CNNs perform in classifying images and to determine their advantages and disadvantages in various situations. The report offers comprehensive explanations of the networks and datasets, outlining their structures, training methods, and techniques for preprocessing data. In terms of test accuracy, the proposed CNN demonstrated superior performance compared to both AlexNet and VGG-16 on the CIFAR-10 dataset. Additionally, in terms of efficiency, it outperformed both models on the MNIST dataset. Overall, this study provides valuable insights into the suitability of different CNN architectures for image classification tasks and can guide the selection of appropriate models for specific applications.

1 Introduction (A: 40%, B: 60%)

Motivation
Convolutional neural networks (CNN) are a fundamental aspect of machine learning. They ultimately allow computers to learn and recognize patterns. This ability opens the door for many important machine learning functions, especially image classification. Today, successful image classification models can be constructed using any one of the countless CNNs that exist. However, the ready availability of these premade, and often specialized, models can be detrimental to gaining a complete understanding of convolutional neural networks. As such, with this research project, we hope to gain a more complete understanding of CNNs by constructing a new CNN from scratch [1].

Problem Statement
Many existing CNNs have obvious strengths and weaknesses. It is often the case that there is something of a tradeoff that must occur. For instance, a model with excellent accuracy and the ability to handle complex images might be more computationally expensive in comparison to a model with adequate accuracy for simple images. The goal of this research project is to develop a CNN in which all these factors are as balanced as possible, especially when compared to other existing CNNs i.e., VGG-16 and AlexNet [1].

Main Contributions

In this report, we investigate the performance of AlexNet, VGG-16, and a novel CNN that we developed, on three datasets: MNIST, Fashion-MNIST, and CIFAR-10. Our contributions include a comprehensive analysis of the network the networks and datasets, comparing accuracies, training/testing times, and other metrics. Additionally, we propose a new CNN architecture that outperforms the existing ones on all one out of three datasets. Overall, our report provides a deep insight into the workings of CNNs and their applications in image classification tasks.

Basic Concepts
To move forward with this project, it is first important to know the basics of CNNs. Convolutional neural networks are made up of three different types of layers: convolutional layers,

pooling layers, and fully connected layers. When an image is input into a CNN, it is passed in as a channel. A channel is a matrix of pixel values where the luminance of the pixel corresponds to an 8-bit number. In the case of grayscale images, only one channel is input. For colored RGB images, 3 channels must be input: one for red, one for green, and one for blue. Once the channel(s) is input, it is passed to a convolutional layer. This layer takes a kernel, a matrix much smaller than the input channel, and moves it over/across the input channel. As it moves, the convolutional layer takes the dot product of the two matrices to create a new matrix called a feature map. Based on the values in the kernel, this operation can be used to detect many different types of patterns in the image. These patterns can be as simple as just identifying horizontal lines. However, as the network gets deeper, these patterns get more complex. The results of this operation are then fed into an activation function like ReLU. This step is key in ensuring nonlinearity which is essential for having a multilayer CNN where each layer is built off its predecessor. Once an image has been properly convoluted, it is passed to a pooling layer. Pooling layers serve three important functions: they help prevent overfitting, they help keep computational costs down, and they speed up calculation time. These functions are implemented by cutting down or sampling the feature maps. Basically, pooling layers look at each feature map and discard anything deemed unimportant. This aids in preventing the model from outright memorizing images (overfitting) and reduces the amount of work the next layer will have to do. Finally, CNNs end with fully connected layers. These layers take the extracted features/patterns from the convolutional and pooling layers and classify them. This is most often accomplished using a softmax function to find the probability distribution over the different classes the images are being classified into. Using these layers, all sorts of CNNs can be constructed. Once all the layers are in place, CNN can begin training. Training a CNN primarily entails adjusting the weight of each neuron, backpropagating, and calculating gradient loss until the model starts consistently making accurate predictions (the predicted output matches the correct output). Once a CNN has been sufficiently trained, it can be put through testing and evaluation. An understanding of these basic concepts is the foundation for constructing a new CNN [1].

2 Related Work (A: 50%, B: 50%)

Summary Of Research Question

In working with the AlexNet and VGG-16 convolutional neural networks (CNN) for the purpose of image classification, 4 goals to strive for in creating a new CNN become obvious. Ideally, a newly created CNN should result in a model that is adequately accurate at classifying images, is capable of handling complex images, is relatively simple to learn/teach, and is minimally expensive computationally. As such, the research question for this project is how one might construct a CNN to best accomplish these goals without sacrificing any one goal for another.

How Research Was Done in The Past

Of course, there are countless examples of research that has been done on creating an ideal CNN. However, all research done on CNNs generally follow the same 6-step pattern. First, a problem must be defined. Problem definition for image classifying convolutional neural networks primarily entails outlining what potential inputs and outputs might look like. For instance, a CNN problem might be defined by the type of images to be input (grayscale, RBG, image size, etc.) and the number of classes the images are to be sorted into. Second, a dataset must be collected and prepared. It should be highlighted that CNN results can differ radically between different datasets. As such, the careful collection and preprocessing of a dataset is a vital part of CNN research. Third, the network architecture is designed. This step involves manipulating the number/types of layers present in the CNN and shaping the network's structure. As each layer performs a different function, a researcher will organize the network's layers to best suit their individual research goal. Fourth, the model must be trained. To train a newly developed CNN, training images need to be input to the model, loss needs to be calculated, and the CNNs weights need to be updated to minimize said loss. Fifth, the models hyperparameters need to be tuned. Two of the more popular examples of hyperparameters are the learning rate and the batch size. Hyperparameter tuning will be explored further below. Finally, the model must be tested and evaluated. Once a CNN model is trained, it must be tested using a separate testing dataset to evaluate its effectiveness. This evaluation most frequently includes calculating accuracy, precision, recall, and F1 score. However, CNNs can be evaluated based on many different other factors. For instance, a CNN's computational cost can be evaluated based on the number of number of parameters used and the number of operations needed.

Results/Progress Made in The Past
In the past, there has been significant research addressing each of the 4 goals listed above. To maximize accuracy, CNNs like ResNet have been created. Generally, a deeper CNN will be more accurate than a shallow CNN. This is because a deeper network is capable of learning more complex, non-linear functions. As such, one might assume that to maximize accuracy, one must maximize the number of layers in the model. However, this assumption does not account for the vanishing gradient problem. In the case of the vanishing gradient problem, the gradient becomes so small that the weights cannot be properly adjusted. In other words, if a CNN is made too deep, the model will overfit the data. Resnet is so noteworthy because it can create a very deep network without running the risk of overfitting. It does this by utilizing residual blocks. These blocks allow the model "to take activation from one layer and feed it to another layer," thus creating a very deep neural network without the risk of overfitting [2]. To handle complex images, researchers have developed models like YOLO. Practically, any useful CNN must be capable of some form of complex image classification. However, some CNNs are better at it than others. This is especially true when an image might contain more than one object. The YOLO architecture is capable of image detection. Basically, it can detect multiple objects in one image. This is accomplished by dividing the input image into a S x S grid and classifying the image(s) in each box with ImageNet. The scope of this project does not include image detection, but it is important to note how far the classification of complex images can go [3]. The previous two examples of past research focus on perfecting results. However, getting results are not the only goals to consider when creating a CNN. An effective CNN must also be simple enough to be easily taught/utilized and must be computationally affordable. For instance, CNNs like LeNet-5 are primarily known for their simplicity and how easy they are to learn. LeNet-5 is often considered the easiest CNN to learn/teach due to its straightforward architecture and minimal number of layers. LeNet-5 only has 7 layers: 2 convolutional layers, 2 max pooling layers, and 3 fully connected layers [4]. Ensuring a CNN is simple enough to learn/teach is important. The easier CNN is to fully understand, the better it can be practically utilized. Finally, to minimize computational expense, CNNs like MobileNet have been developed. When it was first developed, MobileNet models were designed with "mobile and embedded vision applications" in mind [5]. As such, it was imperative that the model be as computationally inexpensive as possible. This is accomplished via a width multiplier and depth wise separable convolutions. Of course, these methods are not the only way to minimize the cost of a CNN, but they are two of the most effective techniques.

Summary Of Past Work
In summary, all this past research seems to focus on optimizing any one of these goals individually. ResNet maximizes accuracy, YOLO can handle incredibly complex images, LeNet-5 is simple to learn, teach, and understand, and MobileNet is extremely efficient. However, because of this specialization, each of these CNNs will find a weakness in at least one of the other goals.

Comparison Of Our Work with Past
Unlike the research mentioned above, the aim of this project is to address and meet all four of these goals simultaneously to the best of our ability. Instead of specializing in any one area, the primary objective of this research is to develop a sort of jack-of-all-trades model that balances each of these conflicting goals. This will be accomplished by comparing the performance of 3 CNNs: VGG-16, AlexNet, and our own CNN.

3 Proposed Approach (A: 50%, B: 50%)

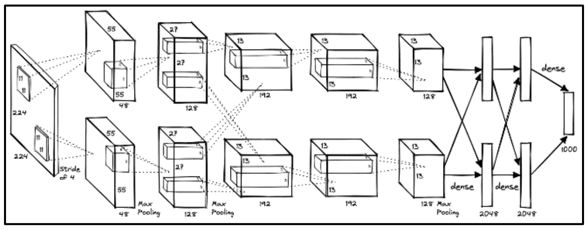
In this section, we will examine three Convolutional Neural Networks (CNNs): AlexNet, VGG-16, and our own CNN. CNNs have become increasingly popular due to their remarkable performance in various image processing tasks, such as object recognition and image classification. We will provide an overview of these three CNNs, especially their architecture. Additionally, we will discuss the advantages and limitations of each network, highlighting the unique features that make them suitable for specific applications.

AlexNet

AlexNet is the first of the three Convolutional Neural Networks (CNNs) currently under discussion. It was developed by Alex Krizhevsky in 2012, under the guidance of Ilya Sutskever and Geoffrey Hinton. AlexNet achieved significant recognition for its exceptional performance in the ImageNet Large Scale Visual Recognition Challenge, where it achieved a top-5 error rate of 15.3%, which was 10.8% lower than the runner-up [6].

The AlexNet architecture uses images sized 227X227X3 as input. It consists of eight layers: 5 convolutional layers and 3 fully connected layers. The first convolution layer applies 96 filters sized 11X11 with a stride of 4 and the resulting output feature map is 55X55X96. Following this, a max-pooling layer of size 3X3 and stride 2 is used, which yields a feature map of size 27X27X96. The second convolution operation uses 256 filters of size 5X5 with a stride of 1 and padding 2, and the resulting feature map is 27X27X256. This is followed by another max-pooling layer of size 3X3 and stride 2, which results in a feature map of size 13X13X256. The third and fourth convolution operations use 384 filters of size 3X3 with stride 1 and padding 1, both using ReLU activation function, resulting in output feature maps of size 13X13X384. The final convolution layer uses 256 filters of size 3X3 with stride 1 and padding 1, also using ReLU activation function, with an output feature map of size 13X13X256. Then, the third max-pooling layer is applied, with a size of 3X3 and stride 2, resulting in a feature map of size 6X6X256 [9].

After this, the first fully connected layer with a dropout rate of 0.5 is applied, followed by the first fully connected layer using ReLU activation function and an output size of 4096. The next dropout layer with the same rate of 0.5 is used, and then a second fully connected layer with 4096 neurons and ReLU activation function. Finally, the output layer, with 1000 neurons and softmax activation function, is applied. The AlexNet model's architecture has a total of 62.3 million learnable parameters, and as it goes deeper, the filter size decreases while the number of filters increases, allowing for the extraction of more features [9].



[11]

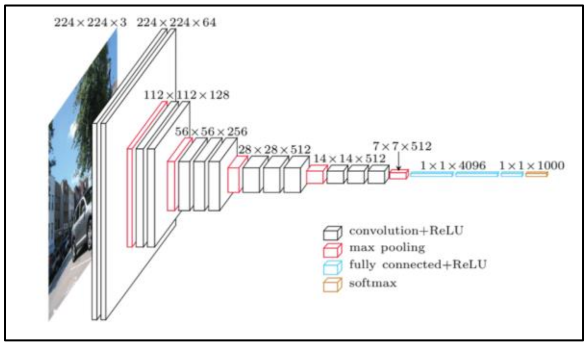
There are several benefits to implementing the AlexNet architecture. Firstly, it was the first convolutional neural network that utilized a GPU for training data, resulting in a significant reduction in training time. This feature enables AlexNet to be an easily trainable deep learning model. Secondly, the activation function utilized in this network, namely ReLU, has two benefits. It does not impose any upper bound on the output, unlike other activation functions, which minimizes the loss of features. Additionally, AlexNet has better performance than LeNet. As AlexNet is a deep neural network with 8 layers, it is much better at feature extraction than LeNet [7].

On the contrary, it is also important to note that implementing AlexNet has drawbacks. Its architecture, which comprises numerous convolutional and pooling layers, presents challenges in terms of interpretation. Additionally, the performance of AlexNet is impeded by the limited availability of training data [8]. In contrast to models with greater depth, AlexNet faces difficulties in extracting image features. Moreover, achieving higher accuracy with AlexNet is a time-consuming process as compared to other models [7].

One vital component of AlexNet's architecture is its use of a cross-entropy loss function in its training process. The cross-entropy loss function, also known as log function, evaluates how well a classification model performs when it produces a probability value between 0 and 1. If the anticipated probability diverges from the actual label, the cross- entropy value grows. By minimizing the cross-entropy loss, the model can learn to make more accurate predictions, which is essential for image classification tasks [10].

VGG-16

One existing network architecture implemented in this project is VGG-16. First proposed by Karen Simonyan and Andrew Zisserman for the 2014 ImageNet Challenge, the primary goal of VGG-16 is to utilize small convolution filters in a deeper network to improve performance [24]. More specifically, the architecture uses 3x3 channels and a network of 16 layers. As seen in the image below, this model takes in one 224x224x3 image as input at a time. In layman's terms, the VGG-16 model requires that all image inputs have 3 channels and are preprocessed so the image width and the image height are both 224. This can be done by cropping the center 224x224 pixels of the image. Once the input image meets the input requirements, it is passed to two convolution layers of size 224x224x64. These layers use 3x3 convolution filters and a hidden layer using the Rectified Linear Unit Activation Function [25]. These two convolution layers are followed by a max pooling layer to decrease the number of feature map parameters created in the previous layers, thus reducing dimensionality. The rest of the model continues in a similar pattern. The first max pooling is followed by two convolution layers of size 112x112x128, a max pool layer, three convolution layers of size 56x56x256, a max pool layer, three convolution layers of 28x28x512, a max pool layer, three convolution layers of size 14x14x512, and a max pool layer with an output of size 7x7x512. This output is then passed to three fully connected layers of sizes 25088, 4096, and 4096 [26]. These layers learn non-linear combinations of features from the convolutional layers to produce a final output.


$224 \times 224 \times 3$
$224 \times 224 \times 64$
$112 \times 112 \times 128$
$56 \times 56 \times 256$
$28 \times 28 \times 512$
$14 \times 14 \times 512$
$7 \times 7 \times 512$
$1 \times 1 \times 4096$
$1 \times 1 \times 1000$
convolution+ReLU
max pooling
fully connected+ReLU
softmax

[15]

There are several benefits to using the VGG-16 model. First, as the model is so straightforward, it is regarded as one of the simpler models to implement and teach. However, the largest advantages of using VGG-16 are a result of small convolution filters and many layers. This characteristic of the model increases non-linearity, improves accuracy, and improves speed [7]. Given that this model was first developed in 2014, it is considered highly impressive that it "achieves almost 92.7% top-5 test accuracy in ImageNet" [27].

The two major disadvantages of the VGG-16 model are a high computation cost and the vanishing gradients problem. As the VGG-16 model uses a large number of parameters, the model itself is very large. Consequently, the model requires more memory and time to run [7]. The model also requires many layers to run which can result in gradients too small to train the model effectively. This problem, also known as the vanishing gradients problem, can be remedied by using an activation function like ReLU [28].

This implementation of VGG-16 uses the cross-entropy loss function. This function measures the distance between the model's output class label probabilities and the true class values during training. The function will then penalize the model based on the accuracy of its prediction. Once the model is penalized, it can adjust until the function is minimized using an optimizer [29].

Newly Designed Architecture

For the final architecture under consideration, we have opted to develop a Convolutional Neural Network (CNN) comprising 10 layers, consisting of 7 convolutional layers and 3 fully connected layers.

The neural network consists of seven convolutional layers, each with a different number of filters. Specifically, the number of filters for the first to the seventh layers are 32, 64, 128, 256, 512, 256, and 128, respectively. All the convolutional layers have a kernel size of (3,3), a stride of (1,1), and utilize the ReLU activation function. Additionally, to ensure that the output size matches the input size, the padding is set to 'same' across all layers. There are two max-pooling layers in the model. Both the pooling layers have a pool size of (2,2) and a stride of (2,2). The first pooling layer is implemented after the first convolutional layer, and the second pooling layer is implemented after the second convolutional layer. It is then followed by the

The architecture includes three fully connected layers. The first fully connected layer comprises 1024 neurons and is subsequently followed by a dropout layer with a rate of 0.5. The second fully connected layer contains 512 neurons and is also followed by a dropout layer with a rate of 0.5. The last fully connected layer has ten neurons that utilize the softmax activation function to convert the previous layer's output into ten distinct probabilities.

By increasing the number of layers in a neural network, there are potential advantages and disadvantages. On the one hand, utilizing a new architecture has several advantages. For instance, the Rectified Linear Unit (ReLU) activation function is computationally efficient and can help to avoid the vanishing gradient problem [21]. Additionally, deep neural networks with multiple layers can learn complex features and patterns more effectively. In our implementation, we utilized 7 convolutional layers, which is two more than AlexNet [22].

On the other hand, there are also potential drawbacks. As with traditional machine learning methods, neural networks often require a substantial increase in labeled data, ranging from hundreds to millions of samples, which can be challenging to obtain. In some cases, alternative methods may be able to effectively handle machine learning problems with less data, and this may be true for the new architecture [23]. Furthermore, like AlexNet, the new architecture may be difficult to interpret and understand due to the presence of multiple layers.

We decided to utilize the cross-entropy loss function, as the other two architectures also implemented this loss function. This approach facilitated the comparison of all three architectures.

In the following section, we will delve into a detailed analysis of the performance of AlexNet, VGG-16, and our own CNN on various datasets. This analysis will provide a better understanding of the strengths and weaknesses of each CNN and help in selecting the most suitable one for specific applications. By comparing the accuracy, computational complexity, and other performance metrics, we will gain insights into which CNN performs better and why.

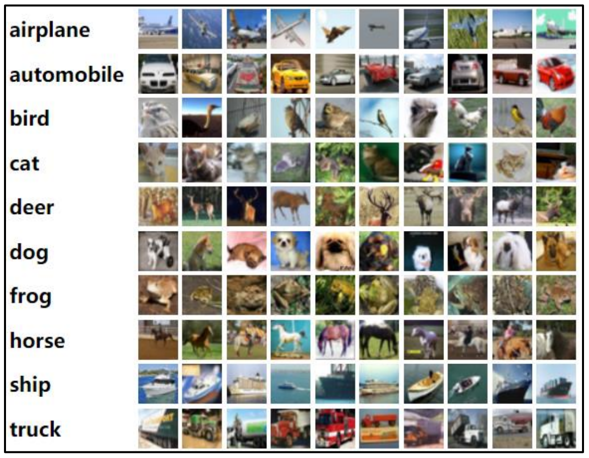
4 Experiments (A: 50%, B: 50%)

4.1 Empirical Settings

This subsection provides detailed information on the empirical settings utilized in this research report. It covers essential statistics related to the datasets, hyperparameter tuning techniques, and evaluation metrics used to assess the model's performance.

CIFAR-10 Statistics

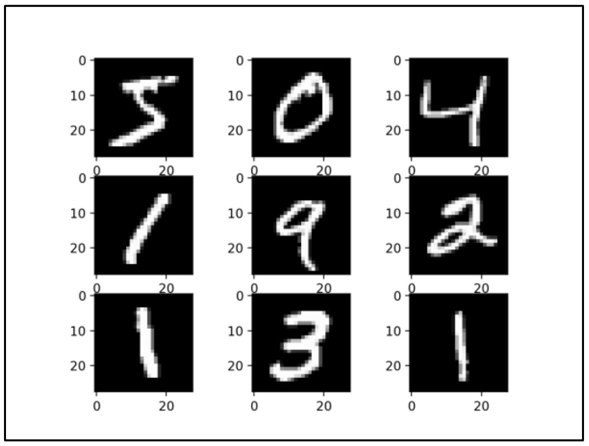
A portion of the Tiny Images collection is the CIFAR-10 [12]. It was gathered by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton [16]. The CIFAR-10 dataset consists of 60,000 32x32 color images that are split into 10 classes. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The dataset can be split into a training set, validation set, and testing set. There are 50,000 images in the training set. For our research, these 50,000 images are split into a training set (45,000) and a validation set (5,000). The testing set consists of 10,000 images [17].



[12]

MNIST Statistics

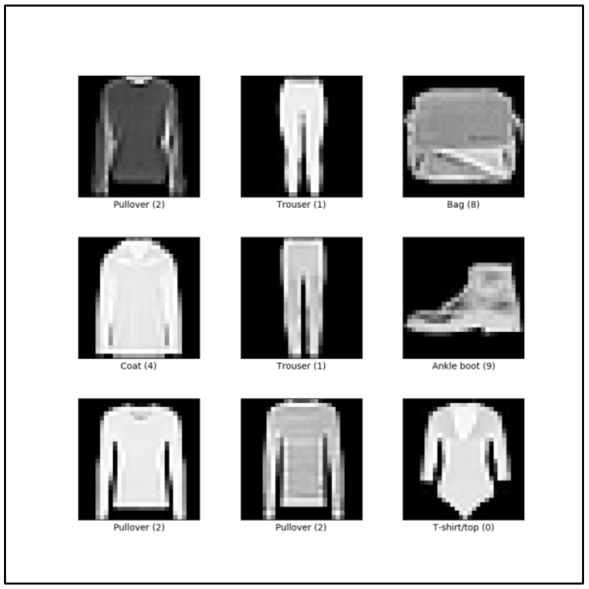
The MNIST dataset is a collection of 70,000 images depicting handwritten digits. These images can be split into 3 subsets: a training set, a validation set, and a testing set. The training set is much larger than the validation and testing sets at 54,000 images compared to the 6,000 images in the validation set and 10,000 in the testing set. The images in this dataset can also be split into 10 different classes. Since the MNIST dataset exclusively contains images of handwritten digits, it must have 10 classes – one for each digit [30].



[13]

Fashion-MNIST Statistics

The Fashion-MNIST dataset is very similar to the MNIST dataset. Like the MNIST dataset, it has a training set of size 54,000, a validation set of size 6,000, and a testing set of size 10,000. It also has 10 classes for the images to be sorted into. The only significant difference between the datasets regards what the images depict. In the Fashion-MNIST dataset, each image is a 28x28 grayscale picture of an item of clothing. As such, instead of each class corresponding to a digit, the Fashion-MNIST classes correspond to types of clothing: T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot [31].



[14]

Hyperparameter Tuning

This project required the tuning of 3 hyperparameters: the learning rate, the batch size, and the number of epochs. In our project, we set the learning rate to 0.001. For the VGG-16 mode, we changed the learning rate to 0.04 and switched the optimizer to ‘sgd’ from ‘adam’. The purpose of the learning rate in a CNN is to determine how much the model adjusts its weights during each iteration of training. As such, it was important to keep the learning rate small to help maximize precision. A larger learning rate might be faster, but it would be a detriment to the model’s accuracy. Another hyperparameter that needs to be tuned is the batch size. The role of the batch size in a CNN is to set the number of samples input for each training iteration. For our research, we used a batch size of 120. This size is small enough that it doesn’t result in overfitting but large enough to not make the model unreasonably computationally expensive. Finally, we also set the number of epochs used to 50. The project description specified that the number of epochs used had to fall in the range of 25-50. As such, we used as many epochs as we could to allow the model to learn more complex patterns.

Evaluation Metrics

For our research, we have chosen to implement the classification report as our evaluation metric. This report is composed of precision, recall, and f1-score, which are widely used performance measures for classification models.

Precision is the ratio of true positives to all the model's positive predictions. It evaluates the model's precision in identifying true cases. A high precision score indicates that the model classifies negative cases mistakenly as positive, or false-positive mistakes [18].

Precision = TP/ (TP + FP).

Recall is defined as the proportion of real positives to all real positive cases in the data. It gauges how well the model can find every successful example. A high recall score indicates that the model classifies positive cases mistakenly as negative, or false-negative mistakes [18].

Recall = TP / (TP + FN).

If a model has a higher precision value but low recall value; in such scenario, it can precisely identify true positive, but it may miss a considerable number of actual positives. On the other hand, a model that has a high recall value, but a low precision score can detect a lot of positives but also generate a lot of false positives.

The F1 score is a useful evaluation metric that considers both precision and recall. It gives a fair evaluation of a model's performance, ranging from 0 to 1, by employing a weighted harmonic mean. Although a higher F1 score suggests greater performance, it's vital to keep in mind that depending on the problem, precision or recall may be more crucial. The fact that F1 scores are often lower than accuracy measurements should also be considered. As a result, it is advised to compare classifier models using the weighted average of F1, since it provides a more thorough assessment of their performance [18].

F1 = 2 * (precision * recall) / (precision + recall)

The next subsection provides a detailed analysis of the three implemented neural networks. This is done by comparing the performance achieved by the three neural networks on the datasets.

4.2 Empirical Results

Overall, the analysis presented in this subsection aims to provide a comprehensive overview of the performance of the three neural networks in relation to the three datasets used in this study.

The early stopping technique was used to analyze the neural networks. Proper training is crucial for the success of a neural network model. Insufficient training may result in underfitting, while excessive training may lead to overfitting and poor performance on the test data. Early stopping is a popular approach to prevent overfitting while increasing performance. Early stopping strikes a balance between underfitting and overfitting. In this method, the model is trained on the training dataset while the performance is being tracked on the validation dataset. When the model's performance on the validation set begins to decline, training is terminated [19].

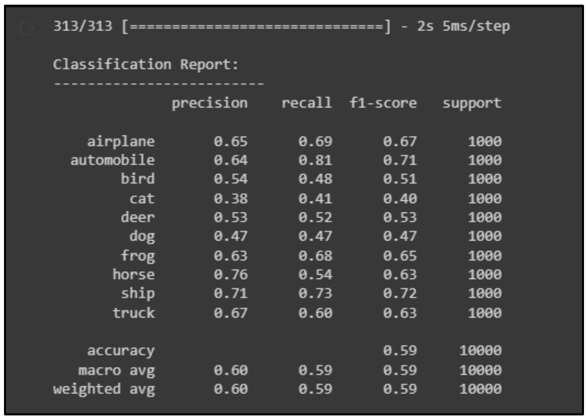
AlexNet (Early Sto*pping vs No early stopping)*

CIFAR-10

The classification report below is a representation of how well the AlexNet model worked on the CIFAR-10 testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.38 (cat) and 0.76 (horse). To explain, in this case, since ‘cat’ images have a lower precision score this basically means that the model is identifying images that are not cats as cats (false positives). Since, ‘horse’ images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren’t horses as horses (false positives). The recall value for all 10 classes lies in between 0.41 (cat) and 0.81(automobile). Since, ‘cat’ images have a low recall score, this means that the model is identifying a lot of cat images as not cat images (false negatives). As ‘automobile’ images have a high recall score, this means that the model is identifying a small amount of automobile images as not automobile images (false negatives). The F1-

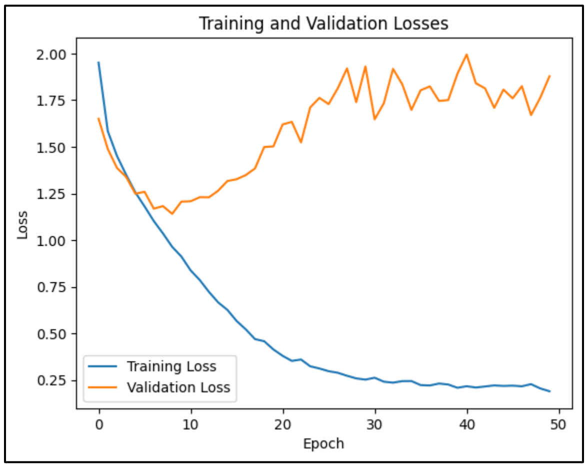
score for all 10 classes lies in between 0.40 (cat) and 0.72 (ship). The range of the F1-score indicates that the model has a decent performance in terms of precision and recall. The test accuracy is 0.59.

No Early Stopping – AlexNet on CIFAR-10



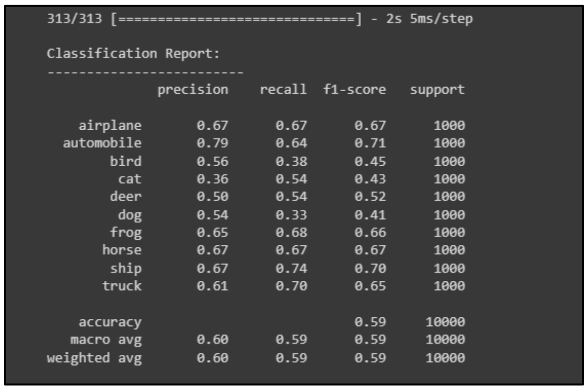
```
313/313 [==============================] - 2s 5ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

    airplane       0.65      0.69      0.67      1000
  automobile       0.64      0.81      0.71      1000
        bird       0.54      0.48      0.51      1000
         cat       0.38      0.41      0.40      1000
        deer       0.53      0.52      0.53      1000
         dog       0.47      0.47      0.47      1000
        frog       0.63      0.68      0.65      1000
       horse       0.76      0.54      0.63      1000
        ship       0.71      0.73      0.72      1000
       truck       0.67      0.60      0.63      1000

    accuracy                           0.59     10000
   macro avg       0.60      0.59      0.59     10000
weighted avg       0.60      0.59      0.59     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. The training loss is a measure of how well the model's predictions match the actual values in the training data. This means that the AlexNet model is improving in its ability to make predictions on the training data. Around 9 epochs, the validation curve starts turning upward. This basically means that the model started overfitting the training data. Overfitting happens when a machine learning model is excessively trained on the training data, resulting in the model memorizing the specific patterns in the training data instead of identifying the underlying patterns that can be generalized to new and unseen data [20].



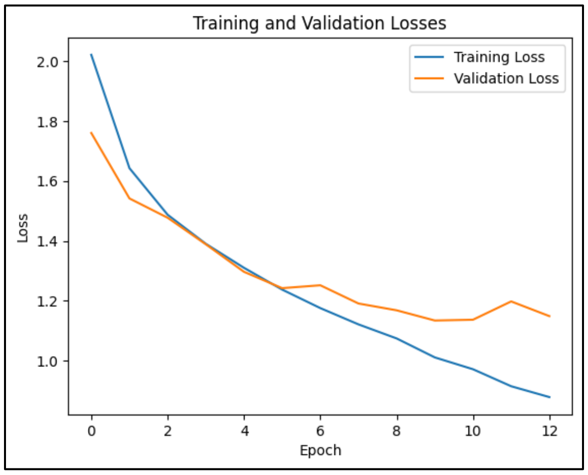
The classification report below is a representation of how well the AlexNet model worked on the CIFAR-10 testing set using early stopping. The precision value, for all 10 classes, lies in between 0.36 (cat) and 0.79 (automobile). To explain, in this case, since 'cat' images have a lower precision score this basically means that the model is identifying images that are not cats as cats (false positives). Since 'automobile' images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren't automobile as automobile images (false positives). The recall value for all 10 classes lies in between 0.38 (bird) and 0.74 (ship). Since, 'bird' images have a low recall score, this means that the model is identifying a lot of bird images as not bird images (false negatives). As 'ship' images have a high recall score, this means that the model is identifying a small amount of ship images as not ship images (false negatives). The F1- score for all 10 classes lies in between 0.43 (cat) and 0.71 (automobile). The range of the F1-score indicates that the model has an average performance in terms of precision and recall. The test accuracy is 0.59.

Early Stopping – AlexNet on CIFAR-10



```
313/313 [==============================] - 2s 5ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

    airplane       0.67      0.67      0.67      1000
  automobile       0.79      0.64      0.71      1000
        bird       0.56      0.38      0.45      1000
         cat       0.36      0.54      0.43      1000
        deer       0.50      0.54      0.52      1000
         dog       0.54      0.33      0.41      1000
        frog       0.65      0.68      0.66      1000
       horse       0.67      0.67      0.67      1000
        ship       0.67      0.74      0.70      1000
       truck       0.61      0.70      0.65      1000

    accuracy                           0.59     10000
   macro avg       0.60      0.59      0.59     10000
weighted avg       0.60      0.59      0.59     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. This means that the AlexNet model is improving in its ability to make predictions on the training data. Unlike the AlexNet implementation without early stopping, this implementation only runs for 12 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.

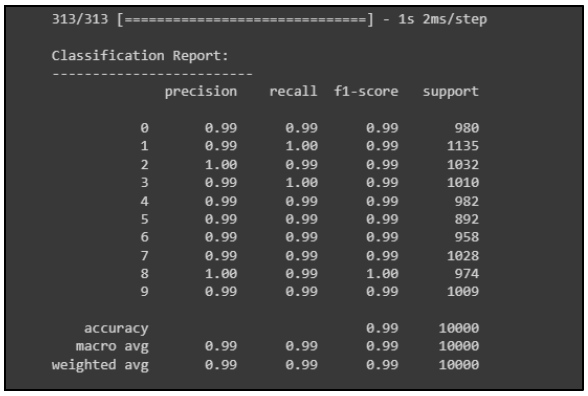


There is no difference in test accuracy between both implementations. There is also not a big difference between the precision, f1-score, and recall values. This basically states that there is no need to run the model for a larger number of epochs when you can achieve results for smaller number, and this also more efficient.

MNIST

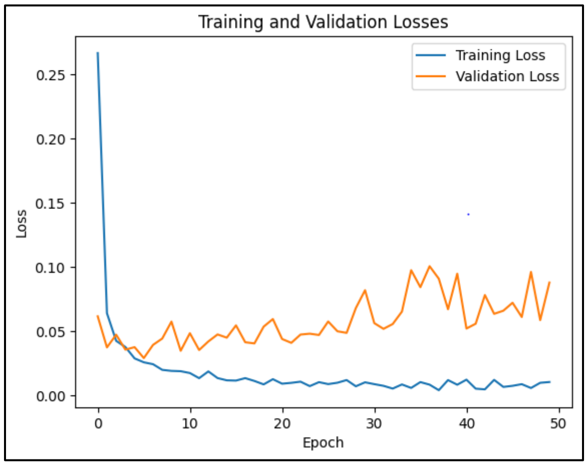
The classification report below is a representation of how well the AlexNet model worked on the MNIST testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.99 to 1.0. To explain, this range is extremely good as this means that the model hardly detects any false positives. The recall value for all 10 classes lies in between 0.99 to 1.0. To explain, this range is extremely good as this means that the model is hardly detecting any false negatives. The F1- score for all 10 classes lies in between 0.99 to 1.0. The range of the F1-score indicates that the model has an exceptional performance in terms of precision and recall. The test accuracy is 0.99.

No Early Stopping – AlexNet on MNIST



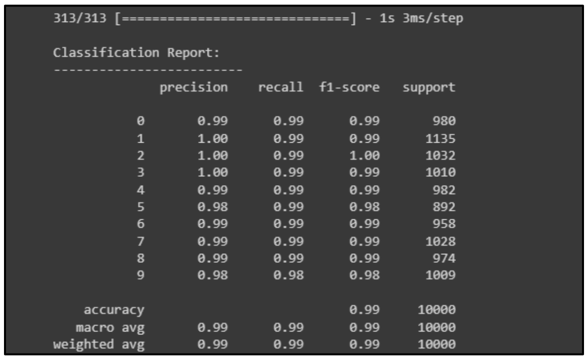
```
313/313 [==============================] - 1s 2ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      1.00      0.99      1135
           2       1.00      0.99      0.99      1032
           3       0.99      1.00      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.99      0.99      0.99       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       1.00      0.99      1.00       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the validation loss curve for the CIFAR-10 dataset, the validation loss curve for the MNIST dataset fluctuates much more frequently. This basically indicates that the model is overfitting the data and that the dataset could be too small for AlexNet, which is a complex model.



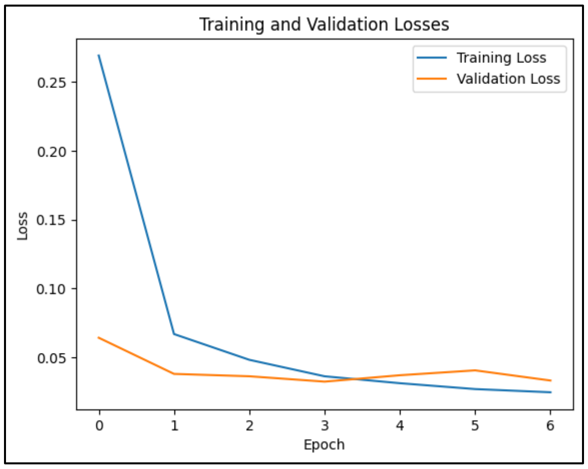
The classification report below is a representation of how well the AlexNet model worked on the MNIST testing set using early stopping. The precision value, for all 10 classes, lies in between 0.98 to 1.0. To explain, this range is extremely good as this means that the model hardly detects any false positives. The recall value for all 10 classes lies in between 0.98 to 0.99. To explain, this range is extremely good as this means that the model is hardly detecting any false negatives. The F1- score for all 10 classes lies in between 0.98 to 1.0. The range of the F1-score indicates that the model has an exceptional performance in terms of precision and recall. The test accuracy is 0.99.

Early Stopping – AlexNet on MNIST



```
313/313 [==============================] - 1s 3ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       1.00      0.99      0.99      1135
           2       1.00      0.99      1.00      1032
           3       1.00      0.99      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.98      0.99      0.98       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.98      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the AlexNet implementation without early stopping, this implementation only runs for 6 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.

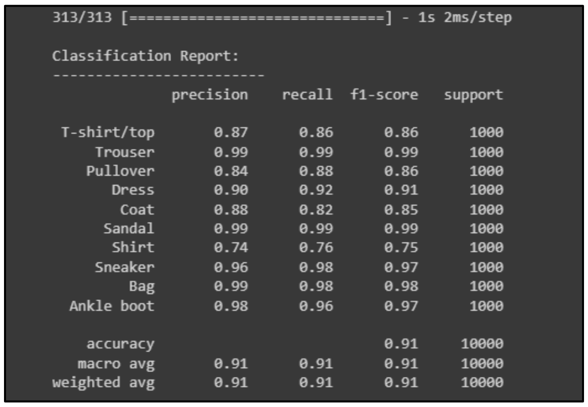


There is no difference in test accuracy between both implementations. There is also not a big difference between the precision, f1-score, and recall values. Like the AlexNet implementation for the CIFAR-10 dataset, it is unnecessary to run the model for 50 epochs for the MNIST dataset. Especially when the result can be achieved much earlier, saving a lot of time.

Fashion-MNIST

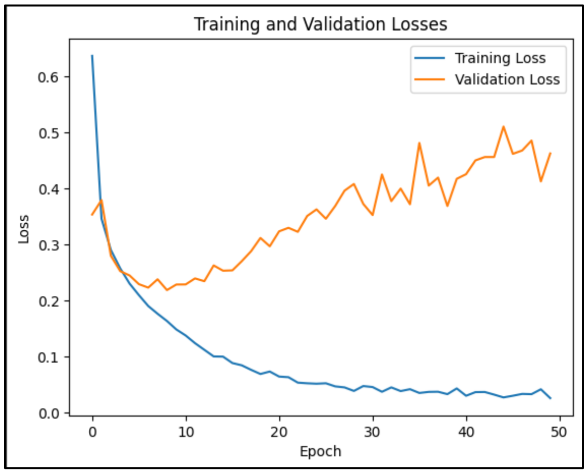
The classification report below is a representation of how well the AlexNet model worked on the Fashion-MNIST testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.74 (shirt) and 0.99 (trouser, sandal, bag). To explain, in this case, since 'shirt' images have a lower precision score this basically means that the model is identifying images that are not shirt as shirt (false positives). Since, 'bag' images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren't bag as bags (false positives). The recall value for all 10 classes lies in between 0.76 (shirt) and 0.99 (trouser, sandal). Since, 'shirt' images have a low recall score, this means that the model is identifying a lot of shirt images as not shirt images (false negatives). As 'sandal' images have a high recall score, this means that the model is identifying a small amount of sandal images as not sandal images (false negatives). The F1- score for all 10 classes lies in between 0.75 (shirt) and 0.99 (trouser, sandal). The range of the F1-score indicates that the model has a good performance in terms of precision and recall. The test accuracy is 0.91.

No Early Stopping – AlexNet on Fashion-MNIST



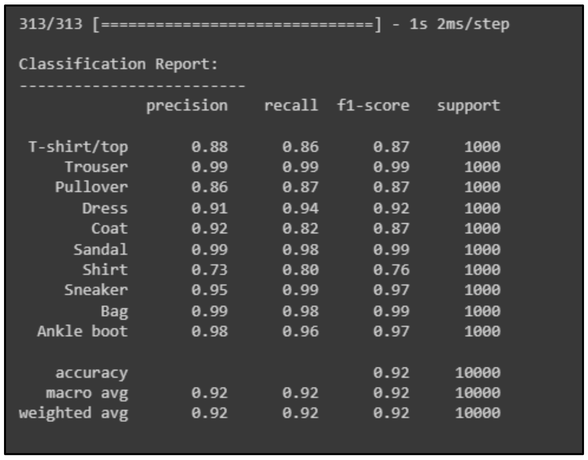
```
313/313 [==============================] - 1s 2ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.87      0.86      0.86      1000
     Trouser       0.99      0.99      0.99      1000
    Pullover       0.84      0.88      0.86      1000
       Dress       0.90      0.92      0.91      1000
        Coat       0.88      0.82      0.85      1000
      Sandal       0.99      0.99      0.99      1000
       Shirt       0.74      0.76      0.75      1000
     Sneaker       0.96      0.98      0.97      1000
         Bag       0.99      0.98      0.98      1000
  Ankle boot       0.98      0.96      0.97      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. The training loss is a measure of how well the model's predictions match the actual values in the training data. Around 8 epochs, the validation curve starts turning upward. And then like the validation loss curve for the MNIST dataset fluctuates much more frequently. This basically indicates that the model is overfitting the data and that the dataset could be too small for AlexNet, which is a complex model.



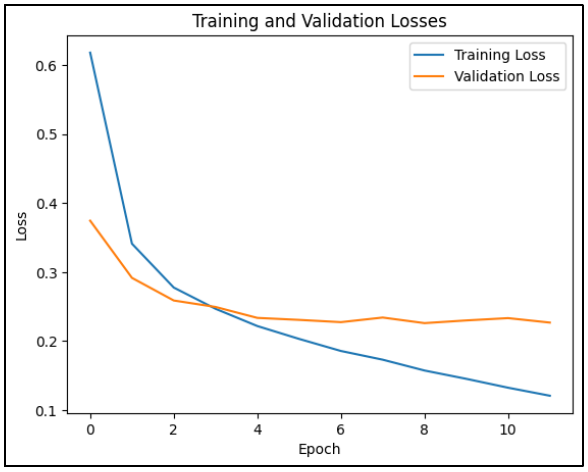
The classification report below is a representation of how well the AlexNet model worked on the Fashion-MNIST testing set using early stopping. The precision value, for all 10 classes, lies in between 0.73 (shirt) and 0.99 (trouser, sandal, bag). To explain, in this case, since 'shirt' images have a lower precision score this basically means that the model is identifying images that are not shirt as shirt (false positives). Since, 'sandal' images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren't sandal as sandals (false positives). The recall value for all 10 classes lies in between 0.80 (shirt) and 0.99 (trouser, sneaker). Since, 'shirt' images have a low recall score, this means that the model is identifying a lot of shirt images as not shirt images (false negatives). As 'sneaker' images have a high recall score, this means that the model is identifying a small amount of sneaker images as not sneaker images (false negatives). The F1- score for all 10 classes lies in between 0.76 (dog) and 0.99 (trouser, sandal, bag). The range of the F1-score indicates that the model has a good performance in terms of precision and recall. The test accuracy is 0.92.

Early Stopping – AlexNet on Fashion-MNIST



```
313/313 [==============================] - 1s 2ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.88      0.86      0.87      1000
     Trouser       0.99      0.99      0.99      1000
    Pullover       0.86      0.87      0.87      1000
       Dress       0.91      0.94      0.92      1000
        Coat       0.92      0.82      0.87      1000
      Sandal       0.99      0.98      0.99      1000
       Shirt       0.73      0.80      0.76      1000
     Sneaker       0.95      0.99      0.97      1000
         Bag       0.99      0.98      0.99      1000
  Ankle boot       0.98      0.96      0.97      1000

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the AlexNet implementation without early stopping, this implementation only runs for 11 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.

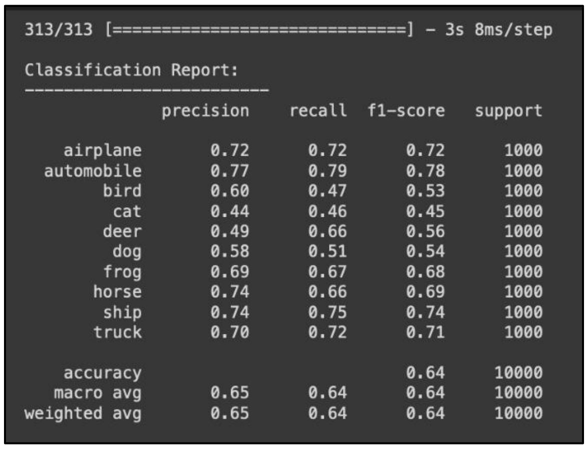


There is a trivial difference in the test accuracy between both implementations. There is also not a big difference between the precision, f1-score, and recall values. Like other datasets, it unnecessary to run the model for more epochs, in this case, as a better test accuracy can be achieved with a smaller number of epochs.

VGG-16 *(Early Stopping vs No early stopping)*
CIFAR-10

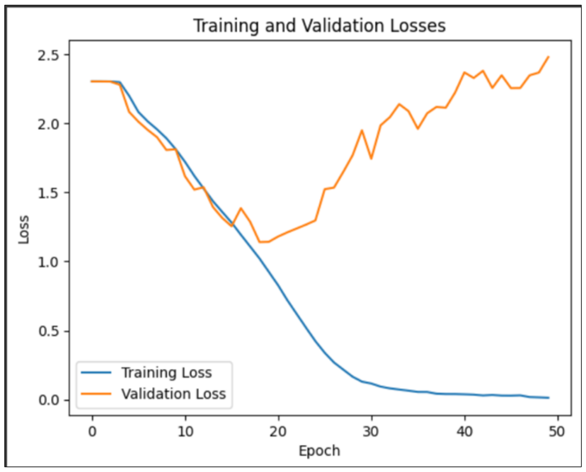
As seen with AlexNet, our experiments with VGG-16 include trials with early stopping and trials without early stopping. The classification report below documents the accuracy, precision, recall, f1-score, and support of the VGG-16 convolutional neural network on the CIFAR-10 dataset without early stopping. This combination of CNN, dataset, and stopping mechanism did not perform as well as many of the other combinations. The model's precision ranges from 0.44 (cat) to 0.77 (automobile). This means that when the model outputs a false positive, it most frequently incorrectly assumed said image was that of a cat and most infrequently incorrectly assumed said image was that of an automobile. The model's recall ranged from 0.46 (cat) to 0.79 (automobile). This means that when the model outputs a false negative, it most frequently incorrectly labelled cat images and most infrequently incorrectly labelled automobile images. The model's F1-score ranged from 0.45 (cat) to 0.78 (automobile). This means that the model struggled the most with the cat class and handled the automobile class the best. Overall, the model's test accuracy was found to be 0.64.

No Early Stopping – VGG16 on CIFAR-10



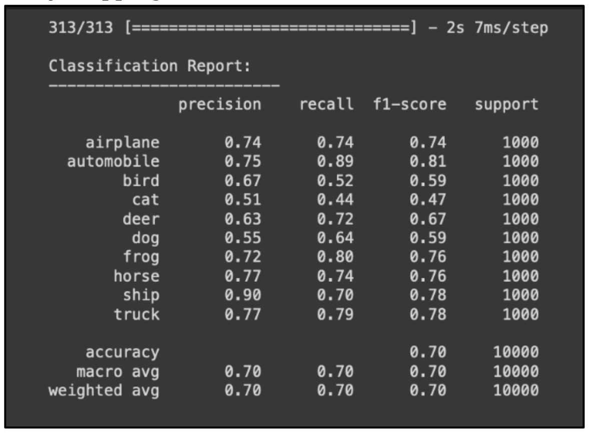
```
313/313 [==============================] - 3s 8ms/step

Classification Report:
--------------------------
              precision    recall  f1-score   support

    airplane       0.72      0.72      0.72      1000
  automobile       0.77      0.79      0.78      1000
        bird       0.60      0.47      0.53      1000
         cat       0.44      0.46      0.45      1000
        deer       0.49      0.66      0.56      1000
         dog       0.58      0.51      0.54      1000
        frog       0.69      0.67      0.68      1000
       horse       0.74      0.66      0.69      1000
        ship       0.74      0.75      0.74      1000
       truck       0.70      0.72      0.71      1000

    accuracy                           0.64     10000
   macro avg       0.65      0.64      0.64     10000
weighted avg       0.65      0.64      0.64     10000
```

The graph below displays the training loss and validation loss curves over the number of epochs for the no early stopping VGG-16 model on the CIFAR-10 dataset. Like the AlexNet model graphs, the training loss indicates how well the predictions match the actual training labels. As the graph displays a decreasing training loss as the number of epochs increases, it can be determined that the model's prediction capabilities are increasing. However, as the training loss continues to decrease, the validation loss begins to rise around epoch 18. This change might suggest overfitting is starting to occur.



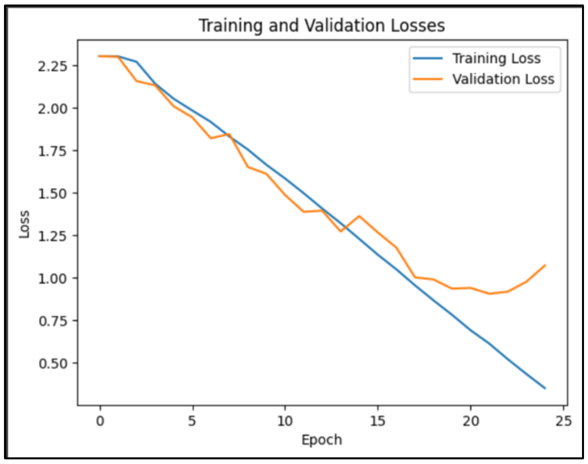
In comparison to VGG-16 without early stopping, early stopping seems to make a significant impact on the results. The precision of the early stopping VGG-16 models on CIFAR-10 ranges from 0.51 (cat) to 0.90 (ship). Recall ranges from 0.44 (cat) to 0.89 (automobile), f1-score ranges from 0.47 (cat) to 0.81(automobile), and the overall accuracy of the model was found to be 0.70. It should be noted that each of these results is better than those output from VGG-16 without stopping. Also, the classes that minimized and maximized the individual evaluation metrics primarily stayed constant between the two stopping methods. The model consistently handled the cat class the worst and the automobile class the best. The only deviation from this is that the ship class had a greater precision than automobile with early stopping.

Early Stopping – VGG16 on CIFAR-10



```
313/313 [==============================] - 2s 7ms/step

Classification Report:
--------------------------
              precision    recall  f1-score   support

    airplane       0.74      0.74      0.74      1000
  automobile       0.75      0.89      0.81      1000
        bird       0.67      0.52      0.59      1000
         cat       0.51      0.44      0.47      1000
        deer       0.63      0.72      0.67      1000
         dog       0.55      0.64      0.59      1000
        frog       0.72      0.80      0.76      1000
       horse       0.77      0.74      0.76      1000
        ship       0.90      0.70      0.78      1000
       truck       0.77      0.79      0.78      1000

    accuracy                           0.70     10000
   macro avg       0.70      0.70      0.70     10000
weighted avg       0.70      0.70      0.70     10000
```

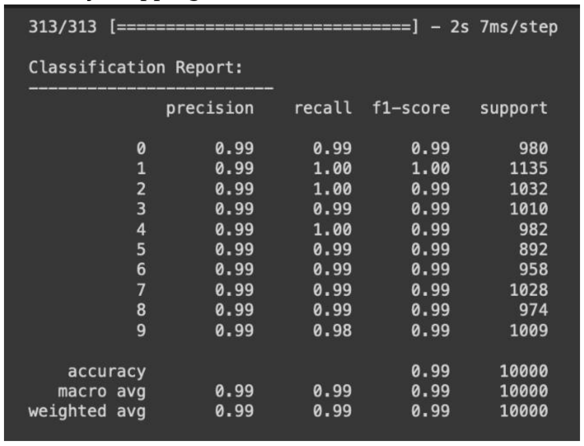
The graphed losses for VGG-16 with early stopping tell a similar story to that of VGG-16 without stopping. The graph displays a steady improvement until epoch 21 when the model stopped early. The validation loss began to climb at this time, thus suggesting oncoming overfitting.



MNIST

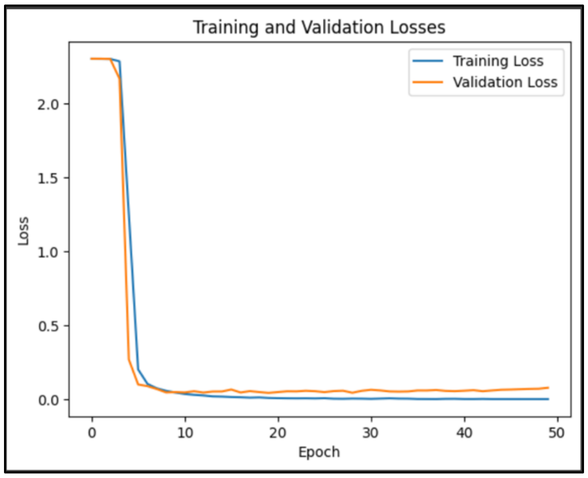
The no early stopping VGG-16 model on the MNIST dataset saw a massive improvement in performance over its performance with the CIFAR-10 database. All classes have a precision score of 0.99. Recall has a range of 0.98 (9) to 1.00 (1, 2, and 4). Every class apart from class 1 had an f1-score of 0.99 while the 1 class had a score of 1.00. Overall, the test accuracy was found to be 0.99. These scores indicate that VGG-16 is excellent at classifying MNIST images without stopping early.

No Early Stopping – VGG16 on MNIST



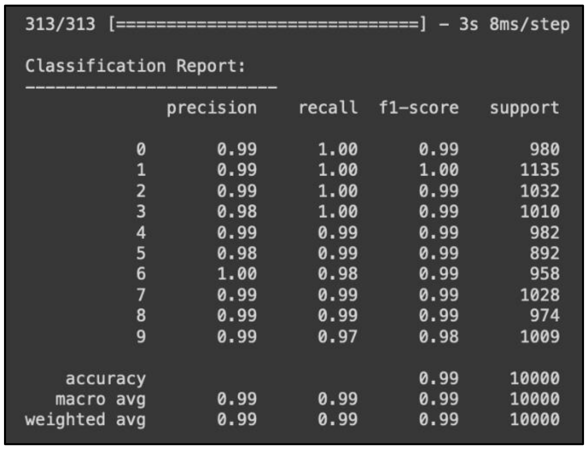
```
313/313 [==============================] - 2s 7ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      1.00      1.00      1135
           2       0.99      1.00      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.99      1.00      0.99       982
           5       0.99      0.99      0.99       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.98      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

The corresponding graph for this combination of CNN, stopping method, and dataset is similarly stellar. The training loss and validation loss decreased significantly from around epoch 3 to epoch 6. After epoch 6, both losses stay mostly consistent for the rest of the model's execution. This indicates that the model learned how to classify MNIST images quickly and did not need to improve further after epoch 6.



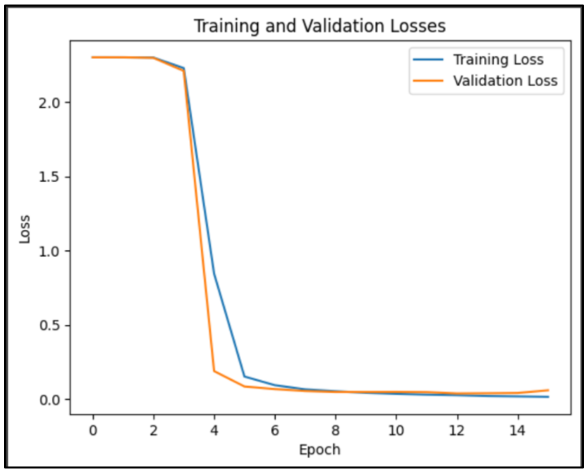
The classification report below displays similar results to those found with no early stopping. Precision ranges from 0.98 (3 and 5) to 1.00 (6), recall ranges from 0.97 (9) to 1.00 (1, 2, 3, and 4), f1-scores range from 0.98 (9) to 1.00 (1), and overall test accuracy was found to be 0.99. As the results of these reports are so like the results from the model without early stopping, it can be concluded that the stopping method used does not make much of an impact on a VGG-16 model classifying MNIST images.

Early Stopping – VGG16 on MNIST



```
313/313 [==============================] - 3s 8ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       0.99      1.00      1.00      1135
           2       0.99      1.00      0.99      1032
           3       0.98      1.00      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.98      0.99      0.99       892
           6       1.00      0.98      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.97      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

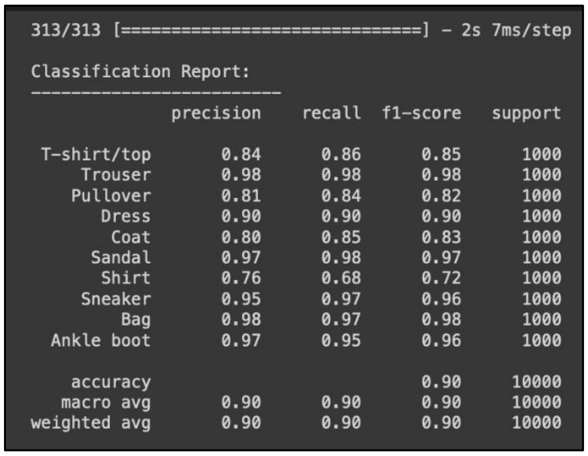
Due to the inconsequential nature of the stopping method for the VGG-16 model on the MNIST dataset, the graph for early stopping losses is nearly identical to that for no early stopping. The only distinction is that the model stops executing after 14 epochs under early stopping.



Fashion-MNIST

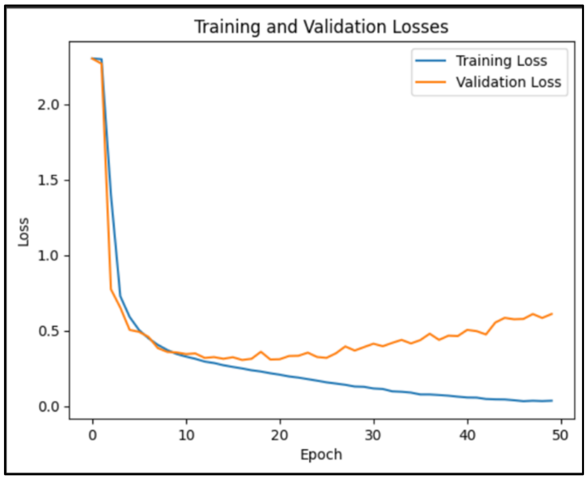
While VGG-16 has proven to be best at classifying MNIST images and worst at classifying CIFAR-10 images, the Fashion-MNIST has proven to be something of a middle ground. Precision ranges from 0.76 (shirt) to 0.98 (trouser and bag). Recall ranges from 0.68 (shirt) to 0.98 (trouser and sandal). F1-scores range from 0.72 (shirt) to 0.98 (trouser and bag). Finally, the total testing accuracy was found to be 0.90. These scores are not quite as impressive as the scores from the model on the MNIST dataset, but they are a marked improvement on the scores from the model on the CIFAR-10 dataset. This makes sense as the Fashion-MNIST dataset is more complex than the MNIST dataset and less complex than the CIFAR-10 dataset.

No Early Stopping – VGG16 on Fashion-MNIST



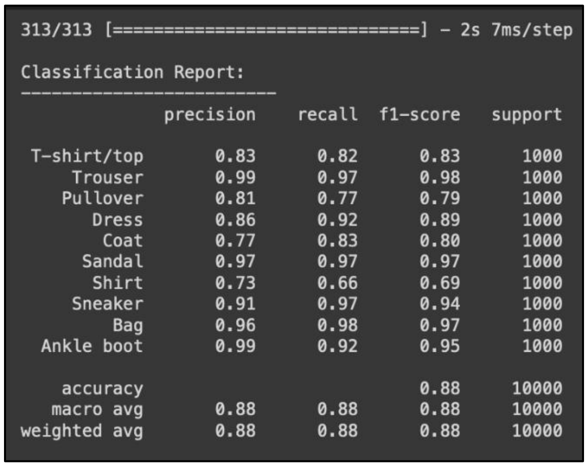
```
313/313 [==============================] - 2s 7ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.84      0.86      0.85      1000
     Trouser       0.98      0.98      0.98      1000
    Pullover       0.81      0.84      0.82      1000
       Dress       0.90      0.90      0.90      1000
        Coat       0.80      0.85      0.83      1000
      Sandal       0.97      0.98      0.97      1000
       Shirt       0.76      0.68      0.72      1000
     Sneaker       0.95      0.97      0.96      1000
         Bag       0.98      0.97      0.98      1000
  Ankle boot       0.97      0.95      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

The graph of training and validation loss for the VGG-16 model on the Fashion-MNIST dataset without early stopping depicts a model that shows every quick improvement from epoch 0 to epoch 5. After epoch 5, improvement of the model slows, and the validation loss starts slowly increasing.



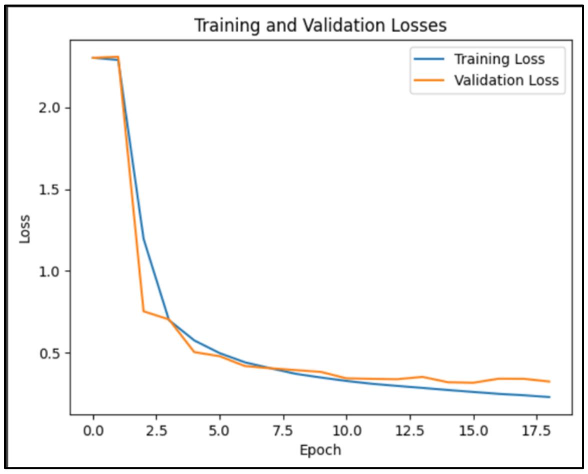
Although early stopping is generally known to increase the accuracy of a model, it slightly decreased the accuracy of this model. Precision ranges from 0.73 (shirt) to 0.99 (ankle boot). Recall ranges from 0.66 (shirt) to 0.98 (bag). F1-scores range from 0.69 (shirt) to 0.98 (trouser) and overall test accuracy was found to be 0.88. Not only are these scores slightly worse than those found with the no early stopping model, but they also represent different classes. While the model was consistently the worst at handling the shirt class, the best handled class varied based on evaluation metric and stopping method.

Early Stopping – VGG16 on Fashion-MNIST



```
313/313 [==============================] - 2s 7ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.83      0.82      0.83      1000
     Trouser       0.99      0.97      0.98      1000
    Pullover       0.81      0.77      0.79      1000
       Dress       0.86      0.92      0.89      1000
        Coat       0.77      0.83      0.80      1000
      Sandal       0.97      0.97      0.97      1000
       Shirt       0.73      0.66      0.69      1000
     Sneaker       0.91      0.97      0.94      1000
         Bag       0.96      0.98      0.97      1000
  Ankle boot       0.99      0.92      0.95      1000

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

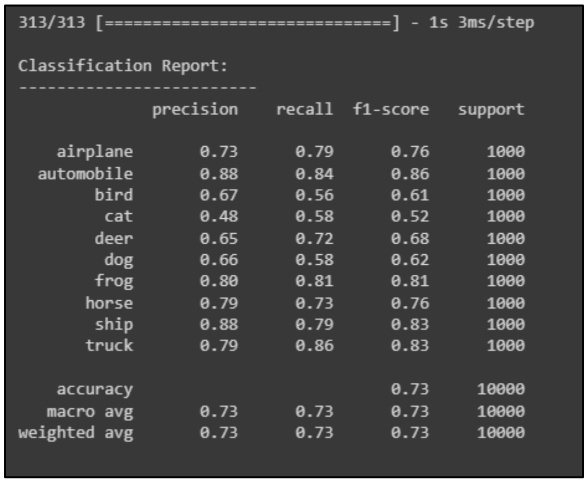
The graph of this model with early stopping sheds some light on the question of why early stopping yields worse results. As seen below, the early stopping graph ends at 17.5 epochs while the no early stopping graph ends at 50 epochs. While the training loss stops radically dropping around epoch 3, there is still a slight decline all the way to epoch 50. While the training loss decline from epoch 18 to epoch 50 is small, it is still enough to make stopping early less effective than the model without early stopping.



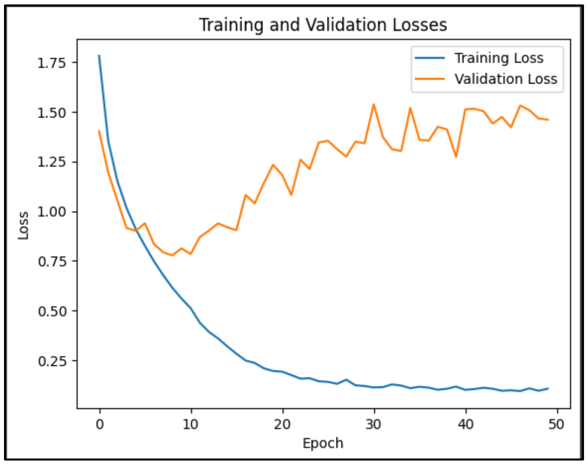
New Architecture (*Early Stopping vs No early stopping)*
CIFAR-10

The classification report below is a representation of how well the New Architecture model worked on the CIFAR-10 testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.48 (cat) and 0.88 (automobile, ship). To explain, in this case, since 'cat' images have a lower precision score this basically means that the model is identifying images that are not cats as cats (false positives). Since 'ship' images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren't ships as ships (false positives). The recall value for all 10 classes lies in between 0.56 (cat) and 0.86 (truck). Since, 'cat' images have a low recall score, this means that the model is identifying a lot of cat images as not cat images (false negatives). As 'truck' images have a high recall score, this means that the model is identifying a small amount of truck images as not truck images (false negatives). The F1- score for all 10 classes lies in between 0.52 (cat) and 0.86 (automobile). The range of the F1-score indicates that the model has a good performance in terms of precision and recall. The test accuracy is 0.73.

No Early Stopping – New Architecture on CIFAR-10


```
313/313 [==============================] - 1s 3ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

    airplane       0.73      0.79      0.76      1000
  automobile       0.88      0.84      0.86      1000
        bird       0.67      0.56      0.61      1000
         cat       0.48      0.58      0.52      1000
        deer       0.65      0.72      0.68      1000
         dog       0.66      0.58      0.62      1000
        frog       0.80      0.81      0.81      1000
       horse       0.79      0.73      0.76      1000
        ship       0.88      0.79      0.83      1000
       truck       0.79      0.86      0.83      1000

    accuracy                           0.73     10000
   macro avg       0.73      0.73      0.73     10000
weighted avg       0.73      0.73      0.73     10000
```

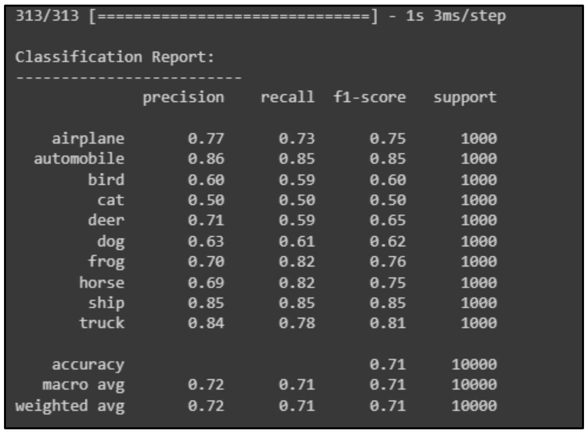
This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. This means that the New Architecture model is improving in its ability to make predictions on the training data. Around 10 epochs, the validation curve starts turning upward. This basically means that the model started overfitting the training data.



The classification report below is a representation of how well the New Architecture model worked on the CIFAR-10 testing set using early stopping. The precision value, for all 10 classes, lies in between 0.50 (cat) and 0.86 (automobile). To explain, in this case, since 'cat' images have a lower precision score this basically means that the model is identifying images that are not cats as cats (false positives).

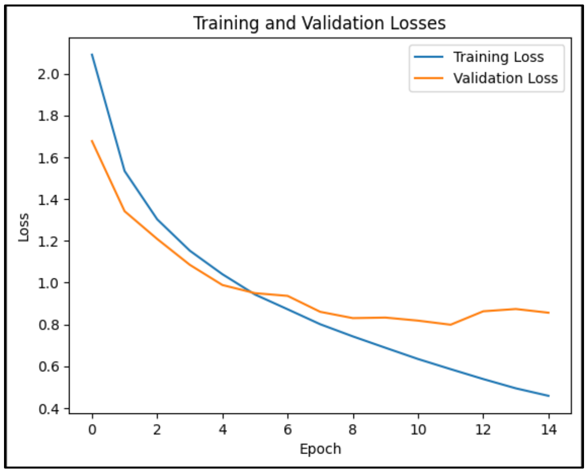
Since ‘automobile’ images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren’t automobile as automobile (false positives). The recall value for all 10 classes lies in between 0.50 (cat) and 0.85 (automobile, truck). Since, ‘cat’ images have a low recall score, this means that the model is identifying a lot of cat images as not cat images (false negatives). As ‘automobile’ images have a high recall score, this means that the model is identifying a small amount of automobile images as not automobile images (false negatives). The F1-score for all 10 classes lies in between 0.50 (cat) and 0.85 (automobile, ship). The range of the F1-score indicates that the model has an average performance in terms of precision and recall. The test accuracy is 0.71.

Early Stopping – New Architecture on CIFAR-10



```
313/313 [==============================] - 1s 3ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

    airplane       0.77      0.73      0.75      1000
  automobile       0.86      0.85      0.85      1000
        bird       0.60      0.59      0.60      1000
         cat       0.50      0.50      0.50      1000
        deer       0.71      0.59      0.65      1000
         dog       0.63      0.61      0.62      1000
        frog       0.70      0.82      0.76      1000
       horse       0.69      0.82      0.75      1000
        ship       0.85      0.85      0.85      1000
       truck       0.84      0.78      0.81      1000

    accuracy                           0.71     10000
   macro avg       0.72      0.71      0.71     10000
weighted avg       0.72      0.71      0.71     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the New Architecture implementation without early stopping, this implementation only runs for 14 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.

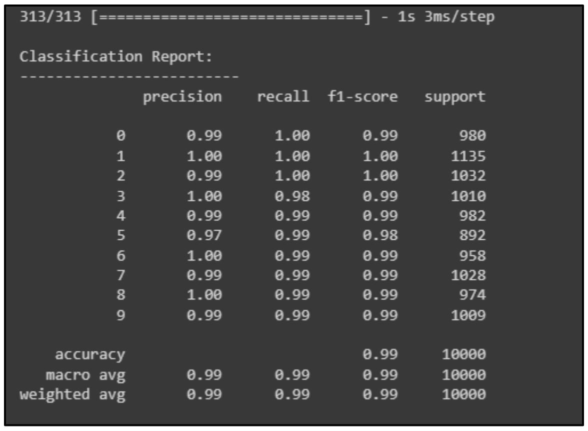


There is a trivial difference in the test accuracy between both implementations. There is also not a big difference between the precision, f1-score, and recall values. It is unnecessary to run the model for 50 epochs for the CIFAR-10 dataset. Especially when the result can be achieved much earlier, saving a lot of time.

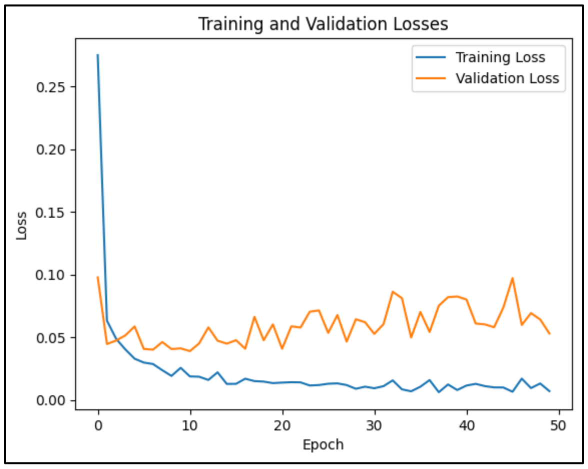
MNIST

The classification report below is a representation of how well the New Architecture model worked on the MNIST testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.97 to 1.0. To explain, this range is extremely good as this means that the model hardly detects any false positives. The recall value for all 10 classes lies in between 0.98 to 1.0. To explain, this range is extremely good as this means that the model is hardly detecting any false negatives. The F1- score for all 10 classes lies in between 0.98 to 1.0. The range of the F1-score indicates that the model has an exceptional performance in terms of precision and recall. The test accuracy is 0.99.

No Early Stopping – New Architecture on MNIST

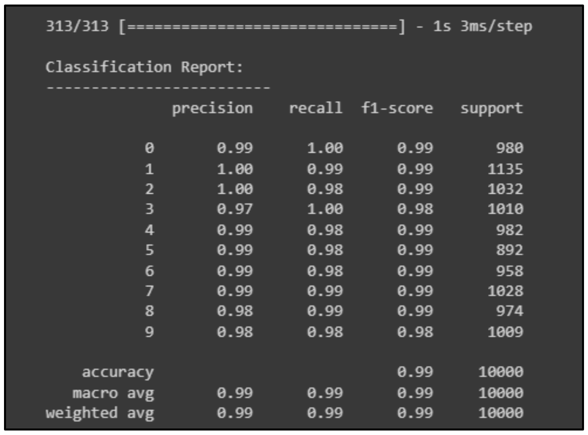

```
313/313 [==============================] - 1s 3ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       1.00      1.00      1.00      1135
           2       0.99      1.00      1.00      1032
           3       1.00      0.98      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.97      0.99      0.98       892
           6       1.00      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       1.00      0.99      0.99       974
           9       0.99      0.99      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the validation loss curve for the CIFAR-10 dataset, the validation loss curve for the MNIST dataset fluctuates much more frequently. This basically indicates that the model is overfitting the data and that the dataset could be too small for New Architecture, which is a complex model.

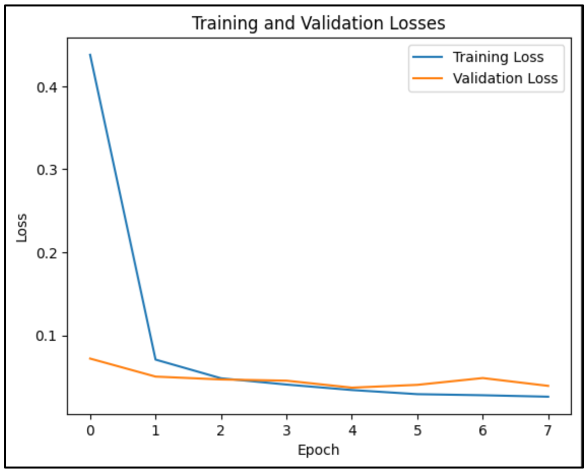


The classification report below is a representation of how well the New Architecture model worked on the MNIST testing set using early stopping. The precision value, for all 10 classes, lies in between 0.97 to 1.0. To explain, this range is extremely good as this means that the model hardly detects any false positives. The recall value for all 10 classes lies in between 0.98 to 1.0. To explain, this range is extremely good as this means that the model is hardly detecting any false negatives. The F1- score for all 10 classes lies in between 0.98 to 1.0. The range of the F1-score indicates that the model has an exceptional performance in terms of precision and recall. The test accuracy is 0.99.

Early Stopping – New Architecture on MNIST


```
313/313 [==============================] - 1s 3ms/step

Classification Report:
-------------------------
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       980
           1       1.00      0.99      0.99      1135
           2       1.00      0.98      0.99      1032
           3       0.97      1.00      0.98      1010
           4       0.99      0.98      0.99       982
           5       0.99      0.98      0.99       892
           6       0.99      0.98      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.98      0.99      0.99       974
           9       0.98      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

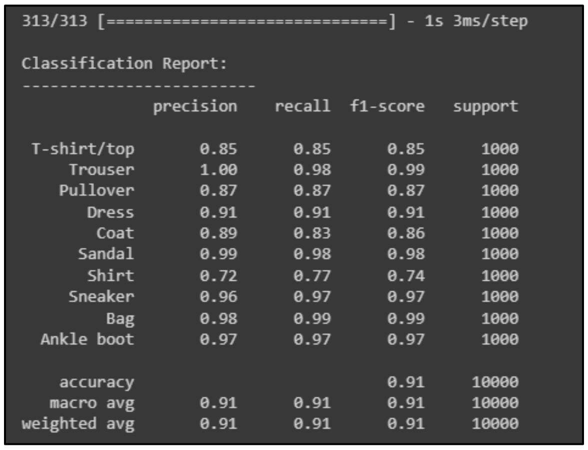
This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the New Architecture implementation without early stopping, this implementation only runs for 7 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.



Fashion-MNIST

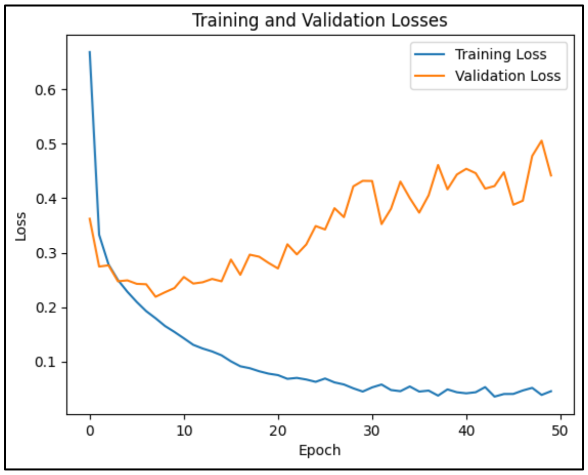
The classification report below is a representation of how well the New Architecture model worked on the Fashion-MNIST testing set without using early stopping. The precision value, for all 10 classes, lies in between 0.72 (shirt) and 1.0 (trouser). To explain, in this case, since ‘shirt’ images have a lower precision score this basically means that the model is identifying images that are not shirt as shirt (false positives). Since, ‘trouser’ images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren’t trouser as trousers (false positives). The recall value for all 10 classes lies in between 0.77 (shirt) and 0.99 (bag). Since, ‘shirt’ images have a low recall score, this means that the model is identifying a lot of shirt images as not shirt images (false negatives). As ‘bag’ images have a high recall score, this means that the model is identifying a small amount of bag images as not bag images (false negatives). The F1- score for all 10 classes lies in between 0.74 (shirt) and 0.99 (trouser, bag). The range of the F1-score indicates that the model has a good performance in terms of precision and recall. The test accuracy is 0.91.

No Early Stopping – New Architecture on Fashion-MNIST



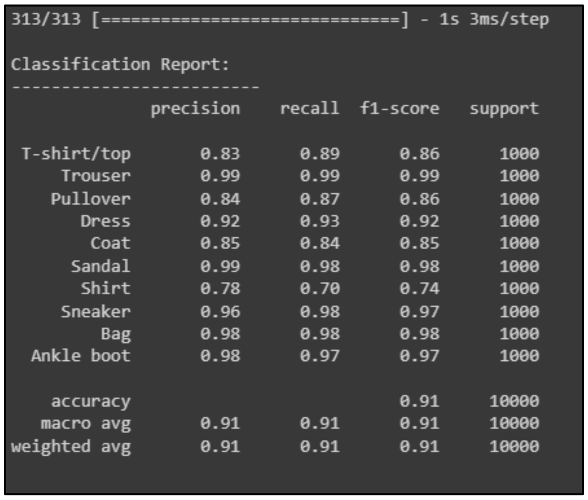
```
313/313 [==============================] - 1s 3ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.85      0.85      0.85      1000
     Trouser       1.00      0.98      0.99      1000
    Pullover       0.87      0.87      0.87      1000
       Dress       0.91      0.91      0.91      1000
        Coat       0.89      0.83      0.86      1000
      Sandal       0.99      0.98      0.98      1000
       Shirt       0.72      0.77      0.74      1000
     Sneaker       0.96      0.97      0.97      1000
         Bag       0.98      0.99      0.99      1000
  Ankle boot       0.97      0.97      0.97      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Around 7 epochs, the validation curve starts turning upward. And then, like the validation loss curve for the AlexNet implementation of Fashion-MNIST dataset, the curve fluctuates much more frequently. This basically indicates that the model is overfitting the data and that the dataset could be too small for New Architecture, which is a complex model.



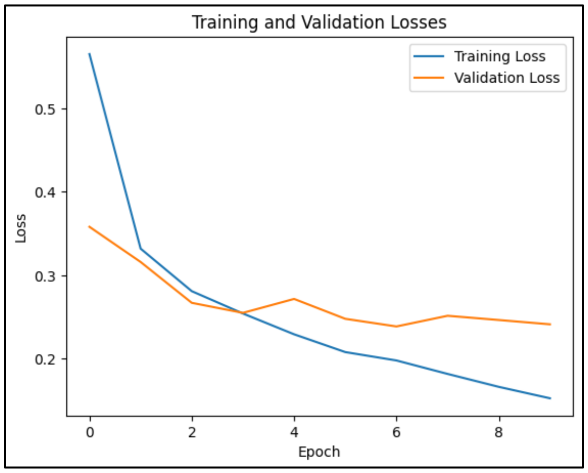
The classification report below is a representation of how well the New Architecture model worked on the Fashion-MNIST testing set using early stopping. The precision value, for all 10 classes, lies in between 0.78 (shirt) and 0.99 (trouser, sandal). To explain, in this case, since ‘shirt’ images have a lower precision score this basically means that the model is identifying images that are not shirt as shirt (false positives). Since, ‘trouser’ images have a higher precision score, this basically means that the model is identifying a lesser number of images that aren’t trouser as trousers (false positives). The recall value for all 10 classes lies in between 0.74 (shirt) and 0.99 (trouser). Since, ‘shirt’ images have a low recall score, this means that the model is identifying a lot of shirt images as not shirt images (false negatives). As ‘trouser’ images have a high recall score, this means that the model is identifying a small number of trouser images as not trouser images (false negatives). The F1- score for all 10 classes lies in between 0.74 (shirt) and 0.99 (trouser). The range of the F1-score indicates that the model has an excellent performance in terms of precision and recall. The test accuracy is 0.91.

Early Stopping – New Architecture on Fashion-MNIST



```
313/313 [==============================] - 1s 3ms/step

Classification Report:
------------------------
              precision    recall  f1-score   support

 T-shirt/top       0.83      0.89      0.86      1000
     Trouser       0.99      0.99      0.99      1000
    Pullover       0.84      0.87      0.86      1000
       Dress       0.92      0.93      0.92      1000
        Coat       0.85      0.84      0.85      1000
      Sandal       0.99      0.98      0.98      1000
       Shirt       0.78      0.70      0.74      1000
     Sneaker       0.96      0.98      0.97      1000
         Bag       0.98      0.98      0.98      1000
  Ankle boot       0.98      0.97      0.97      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

This graph consists of both the training loss and validation loss curves. As the epochs progress, the training loss is decreasing. Unlike the New Architecture implementation without early stopping, this implementation only runs for 9 epochs. This is because the model recognized that it would start overfitting the data if it ran more epochs.



There is no difference in test accuracy between both implementations. There is also not a big difference between the precision, f1-score, and recall values. This basically states that there is no need to run the model for a larger number of epochs when you can achieve results for smaller number, and this also more efficient.

AlexNet vs VGG-16 vs New Architecture *(Early Stopping)*

CIFAR-10

The results of this study regarding the CIFAR-10 dataset are as follows:

AlexNet: Achieved a total test accuracy of 0.59 within 12 epochs.

VGG-16: Achieved a total test accuracy of 0.70 within 21 epochs.

New Architecture: Achieved a total test accuracy of 0.71 within 14 epochs.

As such, because the new architecture was found to have the best test accuracy at 0.71, it can be concluded that it is the most effective model out of the three on the CIFAR-10 dataset. Likewise, because the AlexNet model was able to stop early after only 12 epochs, it can be concluded that it is the most efficient model out of the three. VGG-16 was nearly as effective as the new architecture but took about 7 epochs more than the new architecture to terminate.

MNIST

The results of this study regarding the MNIST dataset are as follows:

AlexNet: Achieved a total test accuracy of 0.99 within 6 epochs.

VGG-16: Achieved a total test accuracy of 0.99 within 14 epochs.

New Architecture: Achieved a total test accuracy of 0.99 within 7 epochs.

All three CNNs were able to achieve a total test accuracy of 0.99 on the MNIST dataset. As such, the comparison between the two is solely reliant on the number of epochs it took to terminate. The AlexNet model was able to stop early after 6 epochs, thus making it the most efficient model. However, it should be noted that the new architecture only took one epoch more while VGG-16 terminated after 14 epochs.

Fashion-MNIST

The results of this study regarding the CIFAR-10 dataset are as follows:

AlexNet: Achieved a total test accuracy of 0.92 within 11 epochs.

VGG-16: Achieved a total test accuracy of 0.88 within 18 epochs.

New Architecture: Achieved a total test accuracy of 0.91 within 9 epochs.

With these results, it can be concluded that AlexNet is the most effective and efficient model on the Fashion-MNIST dataset. The AlexNet has the greatest test accuracy at 0.92 (.01 greater than the new architecture's accuracy and 0.04 greater than VGG-16's accuracy). The new architecture took the fewest number of epochs to terminate (2 fewer than AlexNet and 9 fewer than VGG-16).

5 Conclusion (A: 50%, B: 50%)

In this study, we set out to create a balanced CNN on par with or greater than existing CNNs like AlexNet and VGG-16. We wanted to create a CNN that was able to accurately classify images, able to handle complex images, is relatively simple to learn/teach, and is not computationally expensive. When compared with AlexNet and VGG-16, this new model is not only on par, but surpasses them, in most cases. This new architecture achieved the highest test accuracy out of all three CNNs on one out of three datasets thus ensuring its ability to accurately classify images. And it nearly had the same test accuracy for the Fashion-MNIST dataset compared to the AlexNet (only a difference of 0.01 and was equal to the AlexNet and VGG-16 accuracies for the MNIST dataset). The model was not only able to classify simple, grayscale images like those in MNIST. It was also able to handle slightly more complex, colored images like those in CIFAR-10. As outlined in the proposed approach section above, the model only has 10 layers (7 convolutional and 3 fully connected) and is very straightforward. This straightforward design qualifies it as a simple CNN that is easy to teach, learn, and fully understand. Finally, the new architecture is not computationally expensive. Analyzing how many epochs it takes for the model to finish under early stopping is one way to determine how computationally expensive a model is. While AlexNet did take fewer epochs for both the CIFAR-10 and MNIST datasets, the margin between AlexNet and the new architecture is small. With the CIFAR-10 dataset, the new architecture only took 2 more epochs than AlexNet and it only took 1 more epoch than AlexNet on the MNIST dataset. Also, on the Fashion-MNIST dataset, the new architecture took fewer epochs than both AlexNet and VGG-16. Between the three CNNs, the new architecture proved to be roughly as computationally expensive as AlexNet and much less costly than VGG-16. Overall, in comparison to the other two CNNs, our new CNN has demonstrated a successful balance between objectives (an ability to meet all outlined goals without trading one for another) and has outperformed both existing CNNs we tested, in most cases.

6 References (A: 50%, B: 50%)

[1] C. C. Chatterjee, "Basics of the Classic CNN," *Medium*, Jul. 31, 2019. https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add

[2] V. Sharma, "ResNets: Why do they perform better than Classic ConvNets? (Conceptual Analysis)," *Medium*, Nov. 23, 2021. https://towardsdatascience.com/resnets-why-do-they-perform-better-than-classic-convnets-conceptual-analysis-6a9c82e06e53

[3] "YOLO: Real-Time Object Detection Explained," *www.v7labs.com*. https://www.v7labs.com/blog/yolo-object-detection

[4] J. Jeong, "The Most Intuitive and Easiest Guide for CNN," *Medium*, Jul. 17, 2019. https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

[5] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv.org*, 2017. https://arxiv.org/abs/1704.04861

[6] G. L. Team, "AlexNet: The First CNN to win Image Net," *Great Learning Blog: Free Resources what Matters to shape your Career!*, Jun. 24, 2020. https://www.mygreatlearning.com/blog/alexnet-the-first-cnn-to-win-image-net/#:s~:text=AlexNet%20was%20primarily

%20designed%20by (accessed May 10, 2023).
[7] T. M. Ayyar, “A practical experiment for comparing LeNet, AlexNet, VGG and ResNet models with their advantages…,” *Medium*, Nov. 06, 2020. https://tejasmohanayyar.medium.com/a-practical-experiment-for-comparing-lenet-alexnet-vgg-and-resnet-models-with-their-advantages-d932fb7c7d17
[8] “AlexNet of convolutional neural network - Alibaba Cloud,” *www.alibabacloud.com*. https://www.alibabacloud.com/tech-news/technology/giqt3nak69-alexnet-of-convolutional-neural-network#:~:text=Advantages%20of%20Alexnet&text=and%20pooling%20layers.- (accessed May 10, 2023).
[9] “AlexNet Architecture | Introduction to Architecture of AlexNet,” *Analytics Vidhya*, Mar. 19, 2021. https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/
[10]
“Loss Functions — ML Glossary documentation,” *ml-cheatsheet.readthedocs.io*. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy
[11] “AlexNet and ImageNet: The Birth of Deep Learning,” *Pinecone*. https://www.pinecone.io/learn/imagenet/
[12] “Papers with Code - CIFAR-10 Dataset,” *paperswithcode.com*. https://paperswithcode.com/dataset/cifar-10
[13] J. Brownlee, “How to Develop a CNN for MNIST Handwritten Digit Classification,” *Machine Learning Mastery*, May 07, 2019. https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/
[14] https://www.facebook.com/jason.brownlee.39, “How to Develop a Deep CNN for Fashion-MNIST Clothing Classification,” *Machine Learning Mastery*, Jul. 05, 2019. https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/
[15] “Papers with Code - VGG Explained,” *paperswithcode.com*. https://paperswithcode.com/method/vgg

[16] “CIFAR-10 - Object Recognition in Images,” *kaggle.com*. https://www.kaggle.com/c/cifar-10
[17] A. Krizhevsky, “CIFAR-10 and CIFAR-100 datasets,” *Toronto.edu*, 2009. https://www.cs.toronto.edu/~kriz/cifar.html
[18] M. Krishnan, “Understanding the Classification report through sklearn,” *Muthukrishnan*, Jul. 07, 2018. https://muthu.co/understanding-the-classification-report-in-sklearn/
[19] J. Brownlee, “A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks,” *Machine Learning Mastery*, Dec. 06, 2018. https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/
[20] IBM, “What is Overfitting? | IBM,” *www.ibm.com*. https://www.ibm.com/topics/overfitting
[21] Jason Brownlee, “A Gentle Introduction to the Rectified Linear Unit (ReLU) for Deep Learning Neural Networks,” *Machine Learning Mastery*, Apr. 20, 2019. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
[22] “Advantages of Deep Learning, Plus Use Cases and Examples | Width.ai,” *www.width.ai*. https://www.width.ai/post/advantages-of-deep-learning#:~:text=The%20multiple%20layers%20in%20deep (accessed May 10, 2023).
[23] N. Donges, “4 Reasons Why Deep Learning and Neural Networks Aren’t Always the Right Choice,” *Built In*, 2019. https://builtin.com/data-science/disadvantages-neural-networks
[24] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv.org*, 2014. https://arxiv.org/abs/1409.1556
[25] “Understanding VGG16: Concepts, Architecture, and Performance,” *Datagen*. https://datagen.tech/guides/computer-vision/vgg16/
[26] “VGG16: A Brief Summary | Data Science and Machine Learning,” *www.kaggle.com*. https://www.kaggle.com/getting-started/178568
[27] G. Boesch, “VGG Very Deep Convolutional Networks (VGGNet) - What you need to know,” *viso.ai*, Oct. 06, 2021. https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/
[28] Chi-Feng Wang, “The Vanishing Gradient Problem,” *Medium*, Jan. 08, 2019. https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

[29] “CrossEntropyLoss — PyTorch 1.6.0 documentation,” *pytorch.org*. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[30] samuel100, “MNIST database of handwritten digits - Azure Open Datasets,” *learn.microsoft.com*. https://learn.microsoft.com/en-us/azure/open-datasets/dataset-mnist?tabs=azureml-opendatasets (accessed May 10, 2023).

[31] “Fashion MNIST,” *www.kaggle.com*. https://www.kaggle.com/datasets/zalando-research/fashionmnist