EEE 443 – Final Project: Classification on CIFAR-10 Dataset

Group 2:

Kutay Kaplan – EEE – 22003434

Ahmet Bera Özbolat – EEE – 21902777

Efe Özdilek – EEE – 22003139

Yusuf Taha Sarı – EEE – 22003405

Muhammed Serkan Yıldırım – EEE – 22001869

Abstract

In this paper, classification on the CIFAR-10 dataset has been done using several types of convolutional neural networks (CNNs) such as AlexNet, ResNet, DenseNet, LeNet and VGGNet. In the introduction part, the main purpose and the main tools that were used throughout the project have been presented. In the methods part of the paper, CIFAR-10 dataset has been analyzed and the CNN types have been analyzed thoroughly. Also, several techniques of regularization have also been evaluated in the methods part. In the results part, training errors as a function of epochs, a list of misclassified patterns on training and test sets and test set accuracy have been portrayed. In the discussion part of the paper, the results have been analyzed and the optimal solution for classifying the images from the CIFAR-10 dataset has been chosen according to several metrics.

1. Introduction

The main purpose of this project is to classify the color images from the CIFAR-10 dataset using various types of CNN. AlexNet, ResNet, DenseNet, LeNet and VGGNet are used with the CIFAR-10 dataset, and training errors as a function of epochs, a list of misclassified patterns on training and test sets, and accuracy over the test set for each different type of CNN architectures were recorded. For each type of CNN, different cases were implemented. Regularization techniques such as dropout, max pooling, min pooling and momentum were used in order to observe the impact of these techniques on the accuracy of the results. The results deriving from all types of CNNs were compared with each other and the most efficient one was determined in this project.

It is important to tackle the question of how to classify these images since nowadays, many of the artificial intelligence and machine learning problems are reduced to clustering. One of the most important techniques for clustering is deep convolutional neural networks and in order to understand the working principle of CNNs, trying them on the CIFAR-10 dataset is beneficial. As mentioned earlier, several types of CNNs will be used in order to distinguish the images for each class. The methodology will be explained thoroughly in the following section.

2. Methodology

2.1. Information About CIFAR-10 Dataset

CIFAR-10 is a dataset that includes images widely used in the areas of machine learning and computer vision. The dataset was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Alex Krizhevsky later went on to make significant contributions to the field of deep learning. 1.1 Content: The CIFAR-10 dataset contains 10 different classes, such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6000 images per class, and a total of 60000 color images have a size of 32x32. 1.2 Training and Test Split: Typically, the dataset is split into a training set of 50,000 images and a test set of 10,000 images.

2.2. Used Neural Network Architectures On CIFAR-10

2.2.1 LeNet: LeNet is also usable for images with 32x32 pixels, as in the CIFAR-10. The network contains two convolutional layers in order to extract features from images. The first convolutional layer in LeNet typically has 6 output channels, while the second convolutional layer usually has 16 output channels. The final layer follows convolutional layers and corresponds to the classes. LeNet is quite effective for small-scale image recognition tasks, particularly hand-written digit recognition [3]. See Appendix A.

2.2.2 ResNet: ResNet, short for Residual Network, is one of the CNN architectures. ResNet shows remarkable performance on image classification tasks, especially in large datasets. It is also usable for object detection. There are several variants of ResNet, including ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, each with a different number of layers. ResNet architecture has two significant features that are different from other network structures. The first one is Residual Blocks. These blocks help to address the vanishing gradient problem and enable the training of intense networks by adding the input of a layer to its output after passing through a skip connection. This property allows a layer to learn a residual function. The

second feature is Skip Connections. These types of connections allow the gradient to be directly backpropagated to earlier layers. Skip connections do this by performing identity mapping, and their outputs are added to the outputs of stacked layers. Additionally, batch normalization should be used after every convolutional layer of ResNet in order to reduce the number of required training epochs [5]. See Appendix B.

2.2.3 DenseNet: DenseNet, short for Dense Convolutional Network. In DenseNet, each layer is connected to every other layer with a feed-forward connection. This connectivity pattern differs from traditional architectures, whose layers only link to the next layer. This type of layer connection improves the flow of information and gradients throughout the network, which makes the network more efficient and easier to train. Another fundamental point about DenseNet is that Each layer in DenseNet performs three operations: Batch Normalization, followed by a ReLU activation, and a 3x3 Convolution. DenseNet architecture also consists of a different parameter, which is the growth rate. This parameter mainly controls how much new information each layer contributes to the global state of information. In DenseNet structures, connections between each layer allow feature reuse in the network, which makes DenseNet quite parameter-efficient and reduces the risk of over-fitting [1]. See Appendix C.

2.2.4 AlexNet: AlexNet is another CNN architecture that effectively and substantially reduces the error rate compared to previous methods. It has a more complex structure than other described architectures. Original AlexNet consists of 8 learned layers. Five convolutional layers are followed by three fully connected layers, where all layers use the ReLu activation function. The output of the last layer is connected to a 1000-way softmax layer to produce a distribution over 1000 class labels. On the other hand, CIFAR-10 images are 32x32 pixels, much smaller than the 224x224 pixels AlexNet was designed for. At this point, we need to adjust the size of the kernels in the convolutional layers. Also, the number of layers needs to be reduced because CIFAR-10 is not as deep a dataset as AlexNet's original dataset. The softmax output layer also needs to be modified for ten classes [7]. See Appendix D.

2.2.5 VGGNet: Virtual Geometry Group Net (VGGNet) is a standardized CNN architecture type capable of achieving high results on the image classification tests such as Image Net. It is developed in order to create denser neural networks for specific tasks [8]. It has two regular designs called VGGNet-16 and VGGNet-19. It uses small 3x3 filters and able to fit more layers to the system without increasing the computational demand. VGGNet-16 has 12 convolution layers and 4 fully connected layers. A custom VGGNet is created rather than using the VGGNet-16 directly for some adjustments. Normally, VGGNet has 224x224 resolution RGB images as the input. However, in this project CIFAR-10 dataset only contains 32x32 images and input size is adjusted accordingly [9].

2.3 Regularization Techniques

2.3.1 Momentum: With momentum, weight updates depend on the previous updates rather than being dependent only on the current gradient. Momentum can help to classify the images in CIFAR-10 by reducing the convergence time of a model.

2.3.2 Dropout: The dropout technique removes neurons from the neural network according to the dropout rate. This implementation helps to achieve more accurate weights by reducing the probability of overfitting.

2.3.3 Max Pooling: Pooling layers in CNNs reduce the input size for following convolutional layers. This feature helps to avoid overfitting. In max pooling, images are sliced into windows (often 2x2 pixels) and the maximum value within the windows are selected. This operation reduces the dimensions of inputs, while saving the maxima and discarding the rest.

2.3.4 Min Pooling: Min pooling is used for the same purpose with max pooling but in min pooling, minima of each window is stored opposite to max pooling.

3. Results

3.1. Results for LeNet

Table 1: Different Cases Test Accuracy Results Using LeNet

	Test Accuracy (%)	Training Accuracy (%)	CPU Time (s)
Min Pooling, No Dropout	51.92	90.85	1110
Max Pooling, No Dropout	56.63	94.9	1172
Min Pooling, Dropout = 0.5	49.98	42.45	1534
Max Pooling, Dropout = 0.5	52.98	48.29	1597

The table above refers to the test/training accuracies for different cases and also CPU Times for different cases have been recorded in the table. The maximum test and training accuracies have been obtained when max pooling is used without dropout. Also, it can be clearly seen that removing dropout decreases the CPU Time significantly and dropout did not help increasing the accuracies in this case. LeNet has a significant success at clustering Class 1, Class 7 and Class 8 images in most of the cases yet LeNet could not cluster Class 3 and Class 4 perfectly. See Appendix E. As can be seen from the training loss and validation loss as a function of epochs, without dropout, validation loss starts to increase after some point, i.e. the network reaches its capacity and overfitting starts. So, in order to prevent overfitting, early stopping technique can be used in the best performing case. In other cases validation loss decreases mostly. See Appendix L.

Table 2: For Early Stopping Case of LeNet

For Early Stopping,	Test Accuracy (%)	Training Accuracy (%)	CPU Time (s)
Max Pooling Without Dropout	62.39	78.74	114

From Table 2, it can be seen that test accuracy has increased by approximately %6 when early stopping is used, yet, training accuracy has decreased since the network has been trained with less number of epochs than previous. It can be seen that early stopping was perfect done when validation loss starts to increase. Similar to the previous results, the network again has difficulty clustering Class 3 and Class 4, at this time, also, Class 5. Again, Class 1 and Class 8 have been almost perfectly clustered by the LeNet. See Appendix F.

3.2 Results for ResNet

In the ResNet model, the primary hyperparameters under investigation for their impact on accuracy were the number of layers and the type of pooling used. In addition, Momentum

with rate 0.9 and dropout with rate 0.5 were integrated to the network as the regularization techniques since they proved to increase the accuracy of the model. The results are as follows.

Table 3: Different Cases Test Accuracy Results Using ResNet

Cases	Train Accuracy (%)	Test Accuracy (%)	Epochs	CPU Time (s)
1) ResNet18 with Min Pooling	95.08	86.88	28	929.47
2)ResNet18 with Max Pooling	91.67	80.31	24	812.27
3)ResNet34 with Min Pooling	95.54	87.43	31	1603
4)ResNet34 with Max Pooling	95.42	87.00	30	1544.04
5)ResNet50 with Min Pooling	82.02	72.44	38	3065.45
6)ResNet50 with Max Pooling	83.90	76.02	60	4782.13

It appears that the accuracy is the highest when ResNet length is 34 with Min Pooling. See Appendix G for loss as a function of epochs. Accordingly, the misclassification graph of that case is in Appendix H.

3.3 Results for DenseNet

In our DenseNet model, we investigated 3 different block configurations and 2 different growth rates to optimize network depth and width for the CIFAR-10 dataset. Additionally, we employed dropout (0.5) for regularization, aiming to balance accuracy with computational efficiency. Results are shown below.

Table 4: Different Cases Test Accuracy Results Using DenseNet

Cases (Block Configuration / Growth Rate)	**Train loss**	**Test Loss**	**Test Accuracy (%)**	**CPU Time (s)**
[6, 12, 24, 16]/32	**0.0692**	**0.6895**	**85.05**	**7976.92**
[6, 12, 24, 16]/64	**0.0402**	**0.7505**	**85.60**	**8924.27**
[6, 12, 32, 32]/32	**0.0660**	**0.7490**	**84.56**	**9980.98**
[6, 12, 32, 32]/64	**0.0428**	**0.7763**	**85.59**	**11760.11**
[6, 12, 12, 8]/32	**0.0689**	**0.6762**	**85.30**	**6515.36**
[6, 12, 12, 8]/64	**0.0407**	**0.7430**	**85.67**	**6879.84**

The overall best case, considering both accuracy and computational efficiency, appears to be the configuration with a block size of [6, 12, 12, 8] and a growth rate of 64. This setup not only achieved the highest accuracy but also did so with a reasonable computational time. See Appendix I. Also, for misclassifications for part, see Appendix M.

3.4 Results for AlexNet

Table 5: Different Cases Accuracy Results Using AlexNet with Full Layer For 100 Epoch

	Train Accuracy (%)	Test Accuracy (%)	CPU Time (s)
Min Pooling, No Dropout (a)	96.51	68.75	13822
Max Pooling, No Dropout (b)	98.36	74.48	13905
Min Pooling, Dropout = 0.5 (c)	73.58	70.37	14214
Max Pooling, Dropout = 0.5 (d)	91.17	75.05	14058

In the first part of AlexNet implementation, a complete layer built with five convolutional layers, one flattened layer, and two fully connected layers for AlexNet is used. As one can observe from the table, this implementation takes quite a long time because of the large number of layers. Therefore, there may be underfitting. At this point, one can understand that dropout helps improve accuracy because it deletes half of the layers in fully connected layers and reduces the chance of underfitting. In the case of full size, AlexNet Max Pooling performed better, as seen from the graphs and table. Additionally, most of the misclassifications are in classes 3,4,5,6. See Appendix J. For misclassifications part, see Appendix N.

Table 6: Different Cases Test Accuracy Results Using AlexNet with Full Layer For 100 Epoch

	Train Accuracy (%)	Test Accuracy (%)	CPU Time (s)
Min Pooling, No Dropout (a)	99.16	67.09	9815
Max Pooling, No Dropout (b)	99.39	75.85	9491
Min Pooling, Dropout = 0.5 (c)	97.69	69.35	9871
Max Pooling, Dropout = 0.5 (d)	98.41	77.46	9848

In the second part of the AlexNet implementation, a more straightforward AlexNet structure was used with three convolutional layers, one flattened layer, and one output layer in order to prevent underfitting. The structure with fewer layers takes less training time than full-size AlexNet. Similar to the complete structure, Max pooling also performs better in smaller structures and, as expected, accuracy is higher with dropout compared to a whole network of AlexNet because when the number of layers is reduced, the chance of underfitting also decreases.

3.5. Results for VGGNet:

Table 7: VGGNet Result Table

Cases	Features	Test Accuracy	Minimum Training Error
Case 1:	{'batch_size': 5, 'lr': 0.001, 'momentum': 0.9},	% 73.60	0.131
Case 2:	{'batch_size': 5, 'lr': 0.001, 'momentum': 0.5},	% 73.59	0.128
Case 3:	{'batch_size': 5, 'lr': 0.005, 'momentum': 0.9},	% 71.57	0.473
Case 4:	{'batch_size': 5, 'lr': 0.005, 'momentum': 0.5},	% 73.37	0.135
Case 5:	{'batch_size': 50, 'lr': 0.001, 'momentum': 0.9},	% 72.94	0.215
Case 6:	{'batch_size': 50, 'lr': 0.001, 'momentum': 0.5},	% 70.58	0.482
Case 7:	{'batch_size': 50, 'lr': 0.005, 'momentum': 0.9},	% 74.27	0.105
Case 8:	{'batch_size': 50, 'lr': 0.005, 'momentum': 0.5}	% 73.25	0.196

In the VGGNet part, we investigated 8 different cases for a custom VGGNet setup. Mini batch size, learning rate and the momentum parameters are changed to find best results. All of the results exceeded the %70 Top-1 accuracy limit. The results were very close varying between %70.58 to %74.27 and the best result is achieved at the Case 7 with the mini batch size 50, learning rate 0.005, and the momentum 0.9. However, results showed that the effect of these changes were minimal. The graphs about the VGGNet part showed that in all of the cases the maximum accuracy is achieved earlier than the 30th epoch. And the training loss is decreased gradually regardless of the test accuracy. See Appendix K.

4. Discussion

The main purpose of this project was to analyze a range of CNN types and implement them on the CIFAR-10 dataset. Various types of CNNs such as AlexNet, LeNet, VGGNet, ResNet and DenseNet were used.

In LeNet implementation, test and training accuracies of all cases were not as large as the other CNN types. The main reason for this situation is that LeNet is an old-fashioned CNN type that was even found before the foundation of the CIFAR-10 dataset. Overfitting occurred during the training of LeNet, therefore, early stopping really helped increase the accuracy. Max pooling increased the accuracy dramatically yet min pooling increased the accuracy less than expected. Dropout between convolutional layers had adverse impacts on the accuracy in general. So, overall, LeNet implementation was not as successful as the others, since the capacity and complexity of LeNet is much smaller with respect to other CNN models.

In ResNet implementation, the model that performed the best was the ResNet34 with Min pooling integrated in the convolutional blocks. Additionally, the momentum rate and dropout rates were 0.9 and 0.5 respectively which saved the model from overfitting. Since shortcut connections allow the gradient to flow through the network, training deeper models becomes easier with this ResNet. Accordingly, accuracy of the model was tested under models with different lengths and it appears that ResNet34 seems to be the best approach among all CNN types for the CIFAR-10 image recognition case. Moreover, even though max pooling was expected to be better, min pooling seems to be a better choice in the ResNet case. The reason for that could be that CIFAR-10 images are relatively small, and the objects in these images

can be quite subtle and min pooling can help in emphasizing finer details that max pooling might overlook.

In AlexNet implementation, the best accuracy was observed with a reduced number of convolutional and fully connected layers with a dropout rate of 0.5 and Max pooling. This result is reasonable because the original structure of AlexNet was designed for 224x224 pixel images, but the dataset we used includes 32x32 pixel images. Therefore, underfitting occurs while training. At this point, reducing the number of layers by hand or by using dropout decreases the chance of underfitting. On the other hand, higher accuracy rates than the best case of AlexNet, with a test accuracy of 77.46%, may be reached with other CNN models because underfitting still occurs with 100 epochs of training.

In DenseNet implementation, we achieved accuracies below 85%, despite using a dropout rate of 0.5 to prevent overfitting and average pooling to condense features. The CIFAR-10 dataset's low-resolution images present a challenge for DenseNet's complex architecture to effectively capture and utilize intricate features. While DenseNet is state-of-the-art, it appears that in this instance, other CNN models may achieve better results, suggesting that for specific datasets like CIFAR-10, certain architectures may be more adept at handling image classification tasks, possibly due to varying approaches to feature extraction and learning

In VGGNet implementation, the best accuracy is achieved by the Case 7 with 74.3% accuracy. This accuracy is achieved after some adjustments for the network. However, it is considered low compared to other structures used in this project. The reason for this is that VGGNet was created for more general use with larger datasets and the Cifar-10 dataset does not fit these characteristics. Therefore, results showed that the VGGNet structure is not the most suitable CNN structure to be used in training with the CIFAR-10 dataset.

From the above results, it can be seen that ResNet implementation is the optimal solution for clustering the images from the CIFAR-10 dataset. Originally designed for image classification tasks, ResNet's effectiveness in accurately categorizing images into classes aligns with these results. Its deep layered architecture, which addresses issues like the vanishing gradient problem, makes it particularly suitable for handling complex image datasets like CIFAR-10. Additionally, with the help of DenseNet, an accuracy of 85% was obtained since DenseNet is one of the most complex networks among these networks. LeNet got the lowest accuracy since LeNet is the oldest network among all these networks, therefore, it is the simplest one. LeNet was used for datasets like MNIST, therefore, LeNet was not expected to obtain high accuracies in CIFAR-10 dataset. Overall, for all CNN implementations, results are pretty much as expected.

References

[1] A. Goyal, "Understanding and Visualizing DenseNets," Towards Data Science, [Online]. Available: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

[2] A. Goyal, "Creating DenseNet-121 with TensorFlow," Towards Data Science, [Online]. Available: https://towardsdatascience.com/creating-densenet-121-with-tensorflow-edbc08a956d8

[3] "LeNet-5: A Classic CNN Architecture," Data Science Central, [Online]. Available: https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/

[4] N. Shahriar, "What is Convolutional Neural Network (CNN) - Deep Learning," Medium, [Online]. Available: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5

[5] A. Goyal, "Understanding and Visualizing ResNets," Towards Data Science, [Online]. Available: https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8

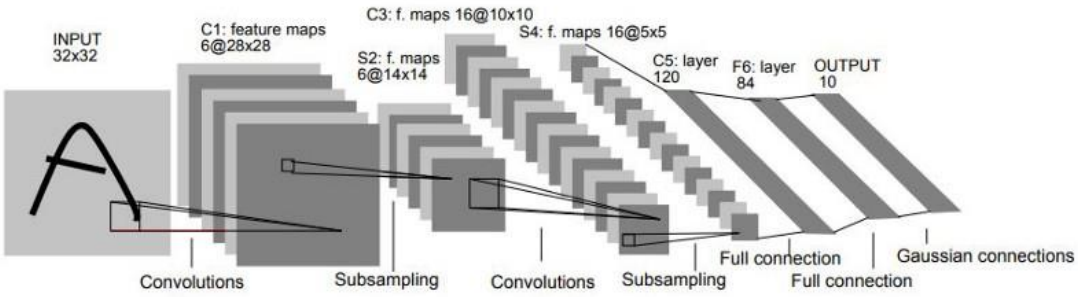
[6] "The architecture of ResNet-50," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/The-architecture-of-ResNet-50-vd-a-Stem-block-b-Stage1-Block1-c-Stage1-Block2_fig4_349646156

[7] "Introduction to DenseNets - Dense CNN," Analytics Vidhya, [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/introduction-to-densenets-dense-cnn/

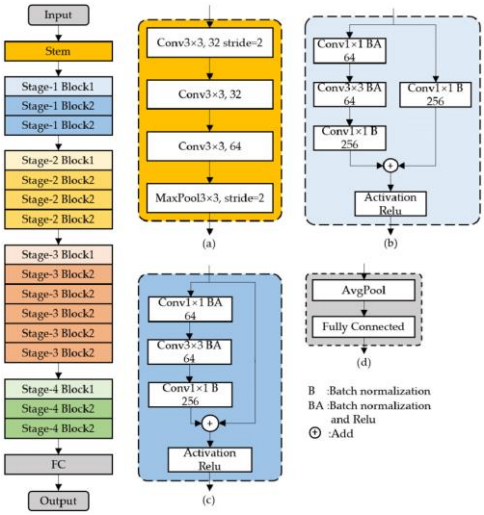
[8] G. Boesch, “VGG very deep convolutional networks (vggnet) - what you need to know,” viso.ai, https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/ (accessed Jan. 1, 2024).

[9] “What is vggnet,” Deepchecks, https://deepchecks.com/glossary/vggnet/ (accessed Jan. 1, 2024).

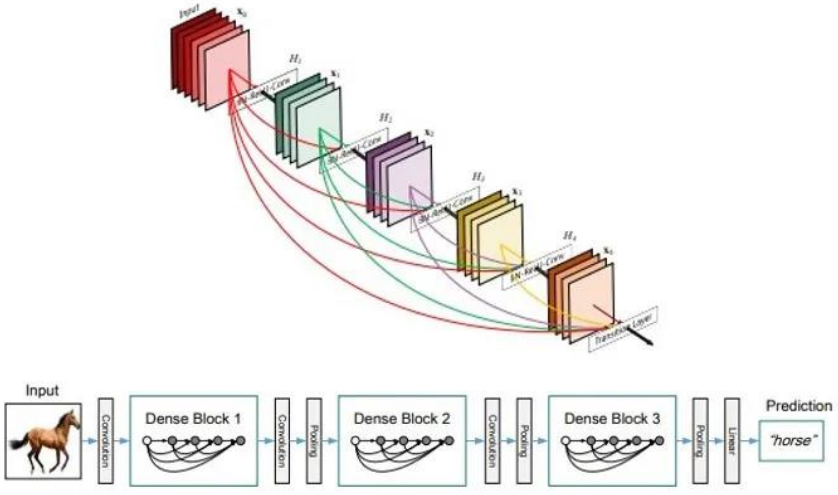
Appendix



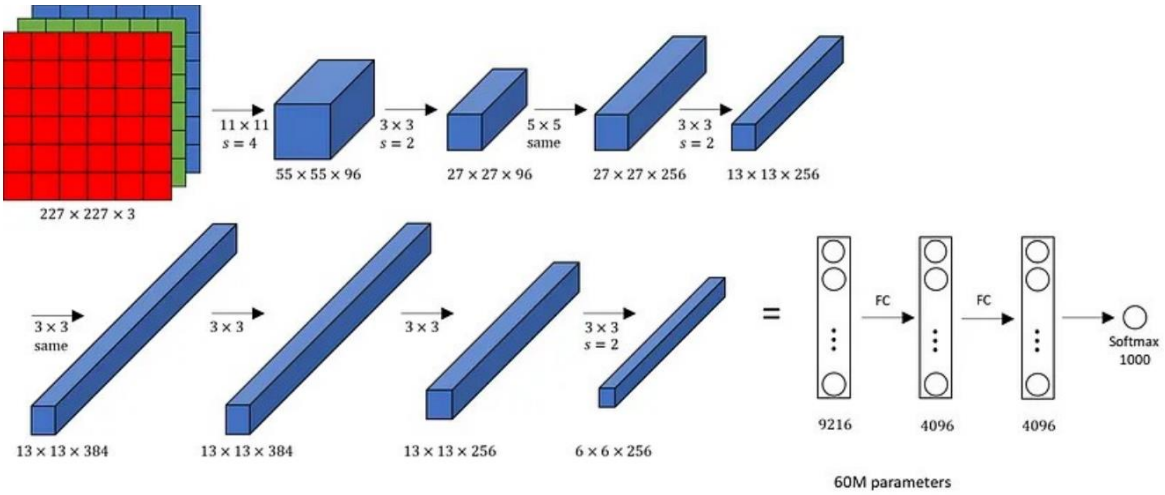
Appendix A: LeNet Structure [3]



Appendix B: ResNet50 architecture [6]



Appendix C: DenseNet Structure [2]



Appendix D: Structure of AlexNet [7]

Min Pooling Without Dropout

```
Class 0: Training misclassifications = 297, Test misclassifications = 404
Class 1: Training misclassifications = 219, Test misclassifications = 380
Class 2: Training misclassifications = 541, Test misclassifications = 597
Class 3: Training misclassifications = 633, Test misclassifications = 660
Class 4: Training misclassifications = 499, Test misclassifications = 556
Class 5: Training misclassifications = 301, Test misclassifications = 564
Class 6: Training misclassifications = 288, Test misclassifications = 424
Class 7: Training misclassifications = 179, Test misclassifications = 423
Class 8: Training misclassifications = 194, Test misclassifications = 350
Class 9: Training misclassifications = 214, Test misclassifications = 450
```

Max Pooling Without Dropout

```
Class 0: Training misclassifications = 241, Test misclassifications = 395
Class 1: Training misclassifications = 120, Test misclassifications = 294
Class 2: Training misclassifications = 271, Test misclassifications = 524
Class 3: Training misclassifications = 505, Test misclassifications = 664
Class 4: Training misclassifications = 235, Test misclassifications = 493
Class 5: Training misclassifications = 360, Test misclassifications = 519
Class 6: Training misclassifications = 225, Test misclassifications = 353
Class 7: Training misclassifications = 246, Test misclassifications = 404
Class 8: Training misclassifications = 89, Test misclassifications = 293
Class 9: Training misclassifications = 259, Test misclassifications = 398
```

Min Pooling With Dropout= 0.5

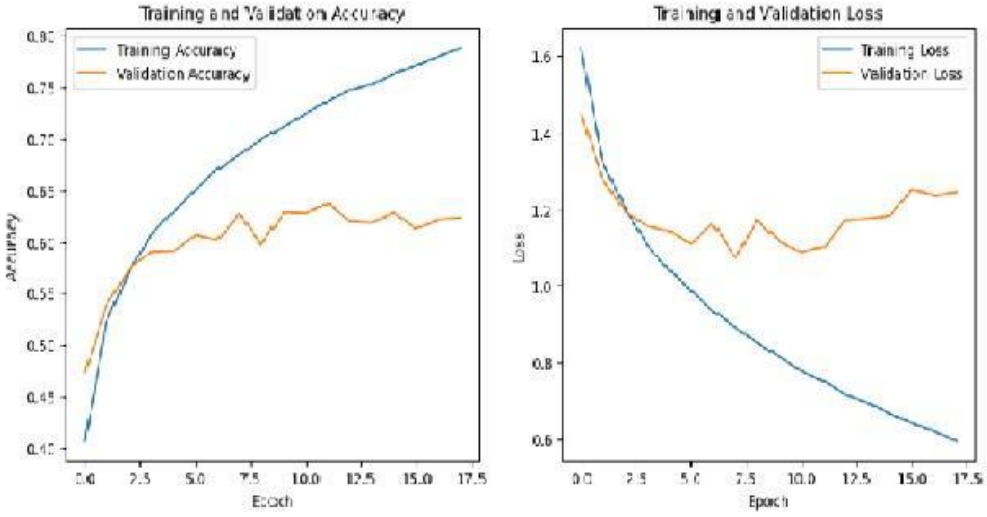
```
Class 0: Training misclassifications = 2335, Test misclassifications = 476
Class 1: Training misclassifications = 1851, Test misclassifications = 375
Class 2: Training misclassifications = 3198, Test misclassifications = 671
Class 3: Training misclassifications = 3242, Test misclassifications = 672
Class 4: Training misclassifications = 3334, Test misclassifications = 691
Class 5: Training misclassifications = 2531, Test misclassifications = 495
Class 6: Training misclassifications = 2108, Test misclassifications = 417
Class 7: Training misclassifications = 1988, Test misclassifications = 432
Class 8: Training misclassifications = 1771, Test misclassifications = 380
Class 9: Training misclassifications = 1875, Test misclassifications = 393
```

Max Pooling With Dropout = 0.5

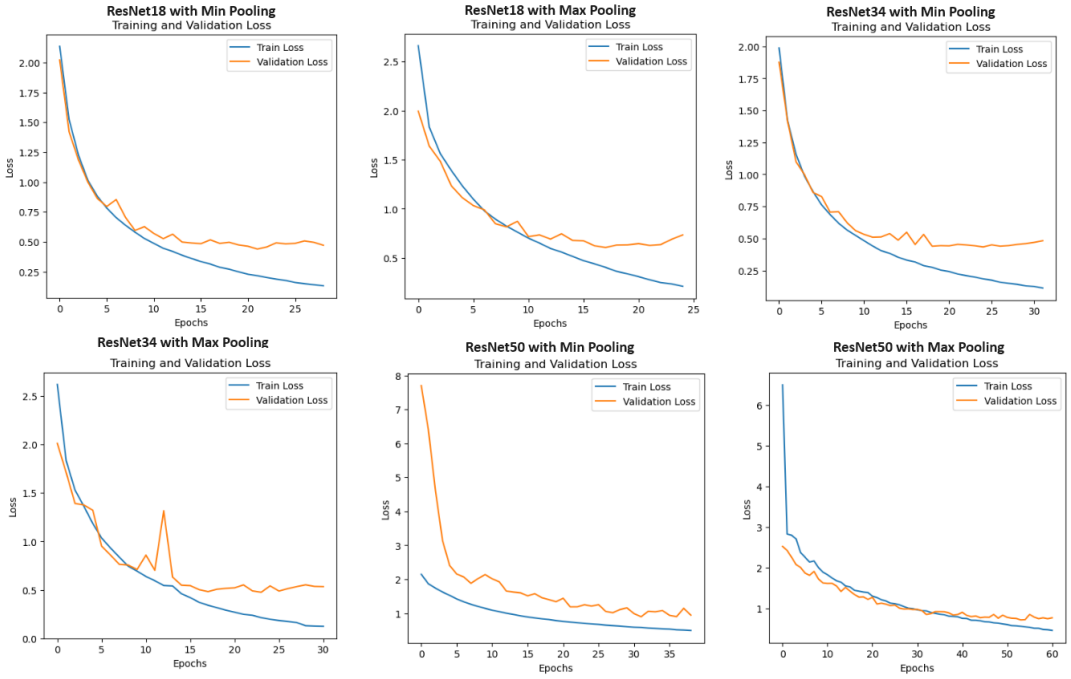
```
Class 0: Training misclassifications = 3169, Test misclassifications = 628
Class 1: Training misclassifications = 1242, Test misclassifications = 268
Class 2: Training misclassifications = 3419, Test misclassifications = 721
Class 3: Training misclassifications = 3831, Test misclassifications = 756
Class 4: Training misclassifications = 2806, Test misclassifications = 602
Class 5: Training misclassifications = 2292, Test misclassifications = 453
Class 6: Training misclassifications = 1082, Test misclassifications = 207
Class 7: Training misclassifications = 2312, Test misclassifications = 474
Class 8: Training misclassifications = 735, Test misclassifications = 166
Class 9: Training misclassifications = 1956, Test misclassifications = 427
```

Appendix E: Misclassifications of LeNet

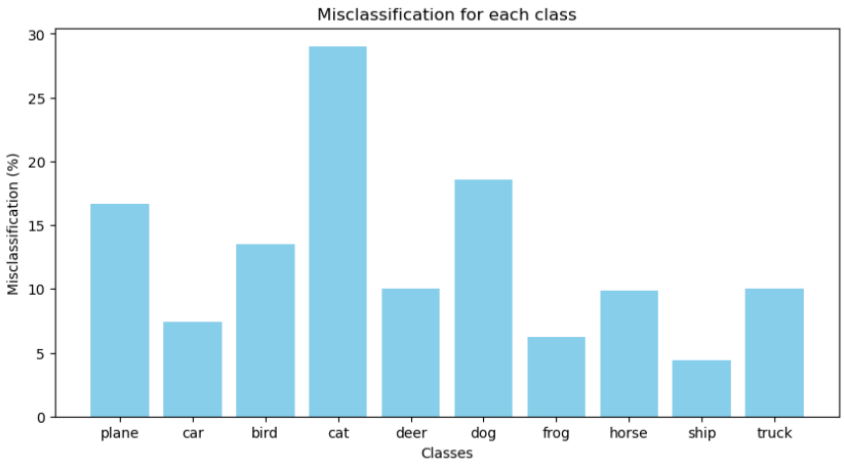
```
Class 0: Training misclassifications = 703, Test misclassifications = 287
Class 1: Training misclassifications = 401, Test misclassifications = 222
Class 2: Training misclassifications = 1134, Test misclassifications = 449
Class 3: Training misclassifications = 1345, Test misclassifications = 515
Class 4: Training misclassifications = 1185, Test misclassifications = 455
Class 5: Training misclassifications = 1990, Test misclassifications = 591
Class 6: Training misclassifications = 775, Test misclassifications = 310
Class 7: Training misclassifications = 897, Test misclassifications = 367
Class 8: Training misclassifications = 314, Test misclassifications = 221
Class 9: Training misclassifications = 866, Test misclassifications = 344
```



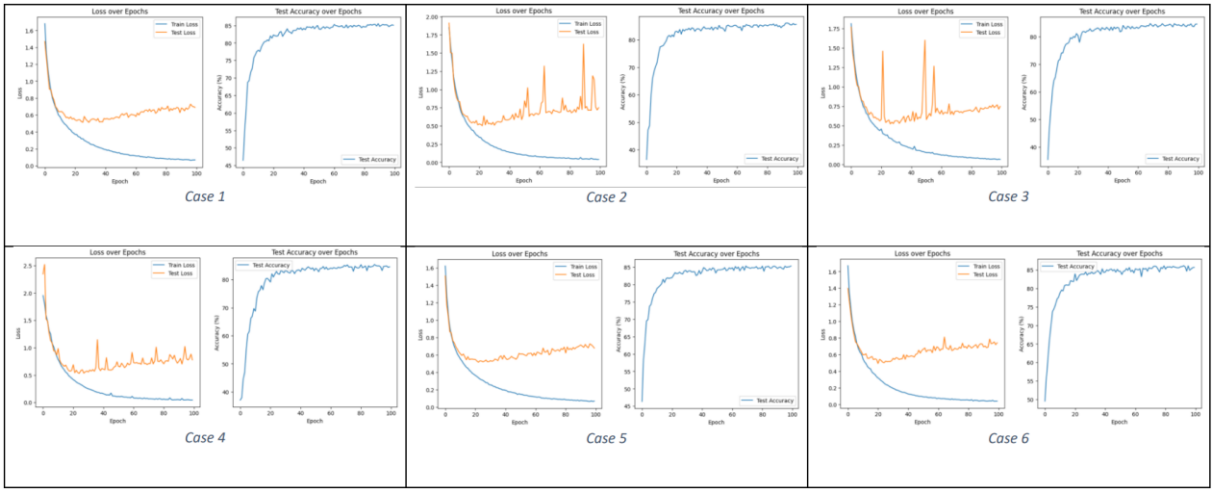
Appendix F: Results For Best Performing Case with Early Stopping of LeNet



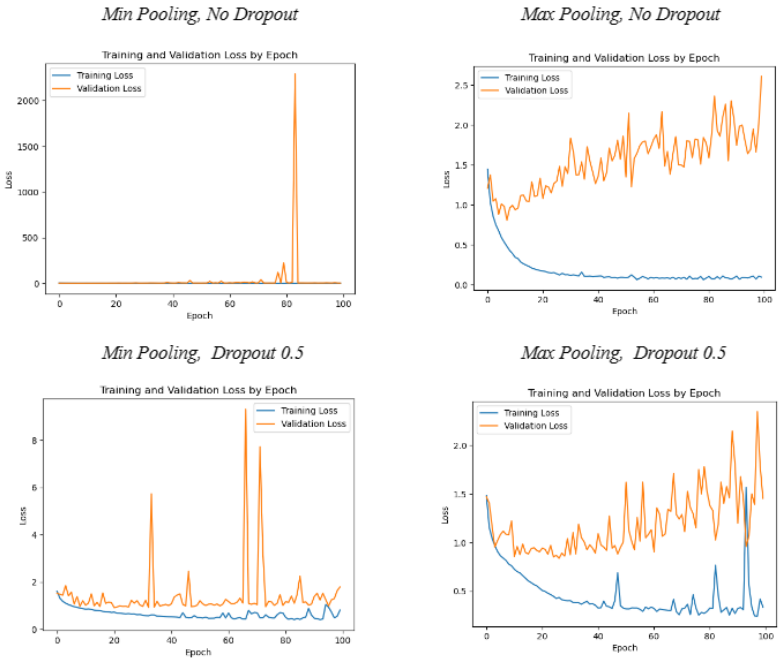
Appendix G: Loss As A Function of Epochs For ResNet



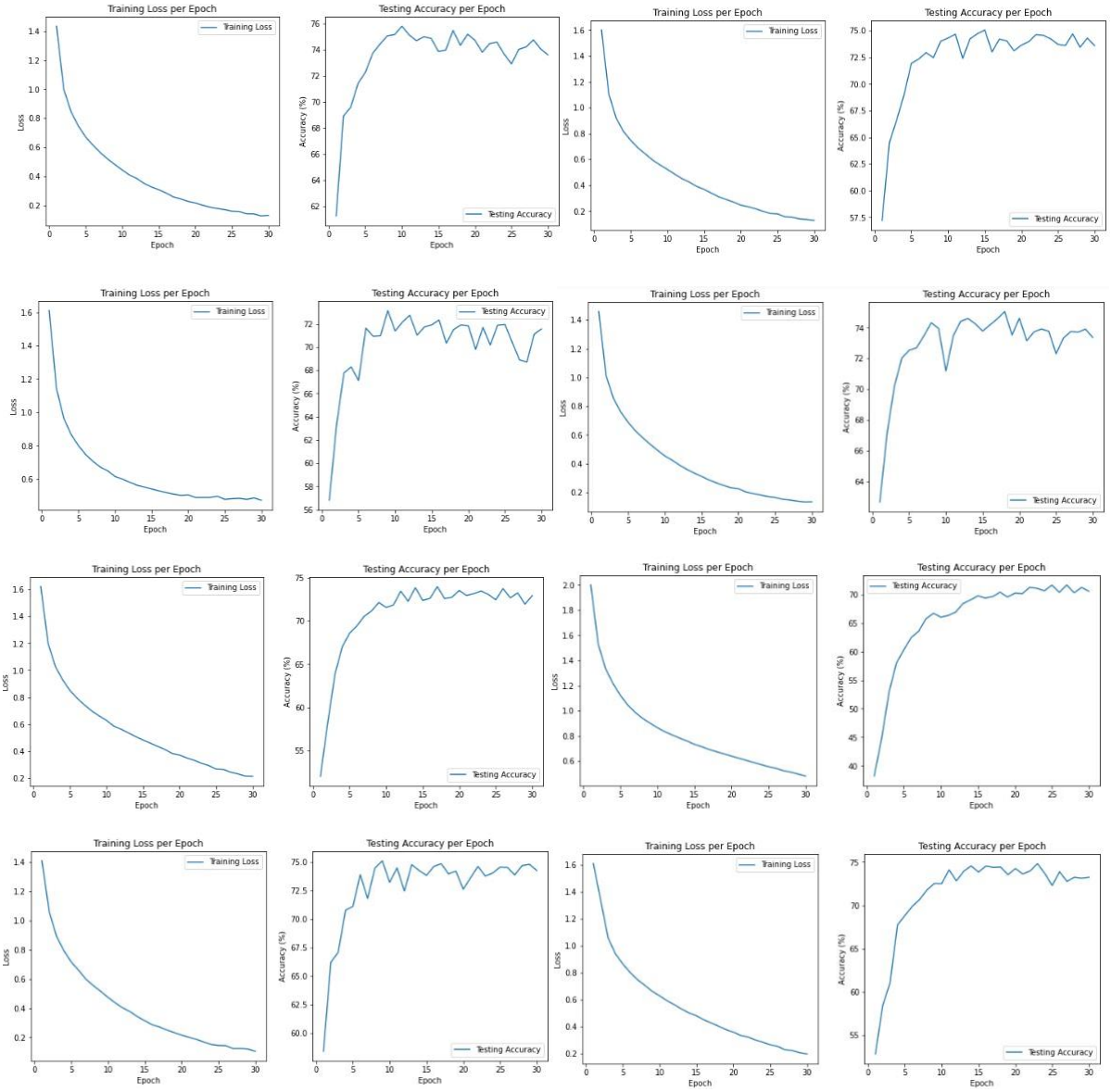
Appendix H: Misclassifications Percentage for ResNet



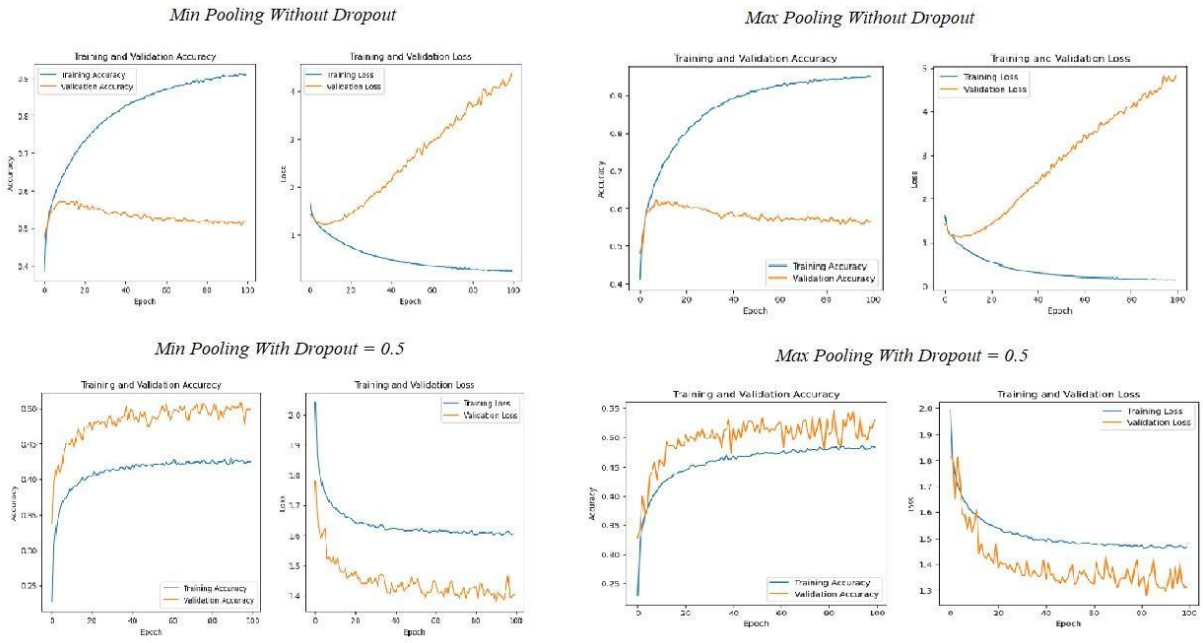
Appendix I: Training/Test Losses as a Function of Epochs with Different Cases For DenseNet



Appendix J: Training/Validation Losses as a Function of Epochs with Different Cases For AlexNet



Appendix K: VGGNet Result Graphs



Appendix L: Training/Validation Losses as a Function of Epochs with Different Cases For LeNet

Cases	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
[6, 12, 24, 16] /32	135	77	216	269	153	230	94	115	106	91
[6, 12, 24, 16]/64	152	76	223	219	141	246	100	148	74	103
[6, 12, 32, 32]/32	163	88	156	282	166	259	90	151	82	98
[6, 12, 32, 32]/64	130	69	179	308	161	220	90	131	94	136
[6, 12, 12, 8]/32	113	93	170	300	158	239	120	136	74	109
[6, 12, 12, 8]/64	131	83	188	326	164	190	86	122	48	105

Appendix M: Misclassifications For DenseNet

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
Min Pooling, No Dropout (a)	272	280	425	526	325	372	236	315	137	237
Max Pooling, No Dropout (b)	212	124	302	319	310	438	106	219	220	245
Min Pooling, Dropout = 0.5 (c)	155	131	479	359	385	466	319	287	157	225
Max Pooling, Dropout = 0.5 (d)	195	179	396	453	339	277	252	185	132	111

Appendix N: Misclassifications For AlexNet