CSCE 479/879 Homework 1: [FMNIST and CIFAR100]

[Izzat Adley, Ryan Bockmon, Quan Nguyen]

February 27, 2021

Abstract

We were tasked with two different image classification problems. One for Fashion-MNIST (F-MNIST) data set and another for the CIFAR-100 data set. We created and tested different deep neural networks models for each of the different data sets to experiment which architectures works better on each of them. In the end we created three different models for F-MNIST and two different models for CIFAR-100.

1 Introduction

We were tasked with two different image classification problems. One for F-MNIST data set and another for the CIFAR-100 data set. We created and tested different deep neural networks models for each of the different data sets to experiment which architectures works better on each of them. In the end we created three different models for F-MNIST and two different models for CIFAR-100.

We found similar results across each of the three model for the F-MNIST. There are a total of 10 labels consisted of integer values ranging from 0-9 each map to a different type of clothing. Each model had an overall validation accuracy above 85% and validation loss below 0.5. There and slightly different results between the two models created for the CIFAR-100 data. There are a total of 100 classes containing 6000 images for each class. The first model had an overall validation accuracy of 53.5% and a loss of 1.86 and the second model had an overall validation accuracy of 44.6% and a validation loss of 2.24. Each model for each data set all had signs of over fitting.

2 Problem Description

We were tasked with two different image classification problems. The first problem was based on a variation of the MNIST data set called Fashion-MNIST. The

Table 1: Mapping of integer values to F-MNIST labels.

Label Value	Meaning
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

data consisted of x amount of 28x28 image, each pixel an integer from 0–255. The class labels consisted of integer values ranging from 0-9 each map to a different type of clothing (Table 1). We had to design and implement at least two architectures for this problem with at least one hidden layer, a ReLU architecture, and softmax for the output layer. We were not able to use convolutional nodes. Measure of loss had to be done with cross-entropy after mapping the class labels to one-hot vectors.

The second problem was training a convolutional neural networks to the problem of image classification from the CIFAR 100 data set. The data consisted of 4-dimensional NHWC (number $\times$ height $\times$ width $\times$ channels) colored images. Each image has 3 color panes, each of size 32×32. There are a total of 100 classes containing 6000 images for each class. Classes consisted of a range of different images such as different animals, insects, household furniture, etc. We had to create at least two convolutional architectures for this problem and use at least two convolutional plus pooling layers. At least one layer must be a connected layer, followed by softmax for the output layer. Cross-entropy must be used as a measurement of loss.

3 Approaches

3.1 Fashion-MNIST

For the Fashion-MNIST data set, we decided to focus on 3 types of architectures (Figure 1). Each number in the diagram represents the number of nodes per hidden layer. In each hidden layer, there will be dropout with the dropout rate = 0.5 and epochs for training are set at 100 for each model.

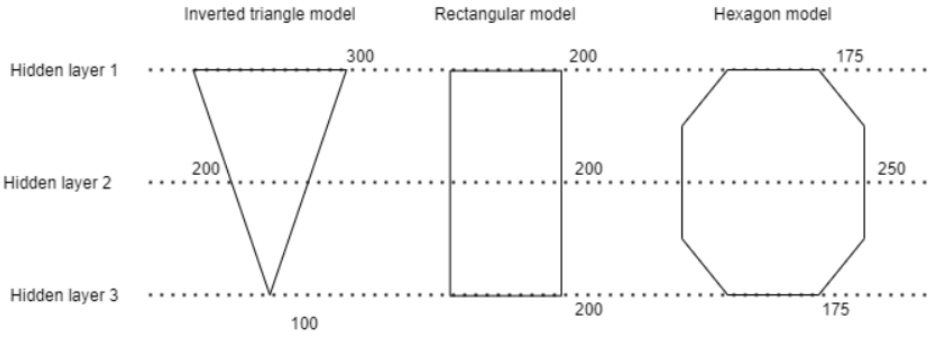

Figure 1: The three different model designs for the F-MNIST data.

We decided to choose the inverted triangle model because hypothetically, the model should allow for improvement of weights and a filter such that it improves the input information. Since the number of nodes as it get closer to the outputs, the information will concentrate as it gets from hidden layer 1 to hidden layer 3. Hypothetically, this will create higher quality information and thus, make better predictions.

Besides that, for the Rectangule model, the hypothesis is that the architecture will allow for balanced weight changes in each hidden layer. This is because the number of nodes in each hidden layer are the same, and if implemented with dropout will allow for balanced training of all weights in the model. The balanced training will theoretically avoid over fitting and create better predictions on test data set.

Lastly, the Hexagon Model was chosen because of it's characteristic of a fat hidden layer. Hidden layer 1 and 3 has 175 nodes while Hidden layer 2 has 250 nodes. This means more weights would exist between hidden layer 1 and 2 and between hidden layer 2 and 3. These extra weights supposedly will allow for more flexibility when training and better prediction capabilities.

3.2 CIFAR-100

We decided on two types of architectures for this problem. The first one is a model with a lot of convolutional layers and dropouts back to back while the second model uses ResBlock, a type of ResBlock architecture. For the rest of the report, we will reference the first model as cifar100model1 and the second model as cifar100model2. In both networks, the convolutional layer was followed by a dropout layer in an attempt to avoid over fitting.

We chose those two models because we wanted to observe whether the ResBlock Bottleneck architecture exhibited some forms of advantages over the more

traditional model with just convolutional layers back to back. These advantages could be in the form of accuracy, computational expenses, and speed. The hypothesis was that both models would have similar accuracy while the second model would take less time. Our hypothesis bases on the assumption that ResBlocks's main advantage is that it allows certain convolutional layer to be skipped without loss of accuracy. Since the model had to go through fewer layer, it would train faster.

The reason behind why Bottleneck could achieve a faster training time is that it simplifies the number of filters at each layer by downgrading it and performing simpler but equivalent operations. In other words, instead of doing matrix multiplication of N x N, Bottleneck uses a 1x1 convolution to reduce the inputs, then perform the expensive matrix multiplication, and finally use 1x1 convolution again to project the size back to normal.

4 Experimental Setup

4.1 Fashion-MNIST

For the Fashion-MNIST data set, we partitioned the training data into train data of 50,000 and validation data of 10,000. Then, each of the three models were trained on 100 epochs with batch size = 32. The activation functions for each layer is ReLU and assign a dropout rate of 0.5 for each hidden layer in a model. Each architecture has 3 hidden layers with different number of nodes as showen in Table 2, and each model has an output layer with 10 nodes with a softmax activation function.

Table 2: Number of nodes in each layer for the F-MNIST models

Model:	Inverted Triangle	Rectangle	Hexagon
First layer	300	200	175
Second layer	200	200	250
Third layer	100	200	175

4.2 CIFAR-100

For the CIFAR-100 data set, we created two different models to test. Each model used an RMS optimizer with a learning rate = 0.0001, and a decay = 0.000001. A ReLu Activation function for most layers and a softmax for the output layers. Loss function was calculated with categorical-crossentropy. The two models where set up as followed:

4.2.1 CIFAR-100 model 1 (cifar100model1)

This model consisted of the following:

- A convolutional2D layer with 32 filters and kernel size is $3x3$.
- BatchNormalization layer
- A layer of MaxPooling2D with pool size = (2, 2)
- Dropout layers with rate = 0.25
- Convolutional2D layers with 64 filters and kernel size is $3x3$.
- BatchNormalization layer
- Convolutional2D layers with 64 filters and kernel size is $3x3$.
- BatchNormalization layer
- MaxPooling2D with pool size = (2, 2)
- Dropout layers with rate = 0.25
- Convolutional2D layers with 128 filters and kernel size is $3x3$.
- BatchNormalization layer
- Convolutional2D layers with 128 filters and kernel size is $3x3$.
- BatchNormalization layer
- MaxPooling2D with pool size = (2, 2)
- Dropout layers with rate = 0.25
- A Dense layer with 1024 features
- BatchNormalization layer
- Dropout layers with rate = 0.5
- Output layer with 100 features representing 100 classes using Softmax.

4.3 CIFAR-100 model 2 (cifar100model2)

This model used the Bottleneck ResBlock provided in Hackathon4. The model consisted of the following:

- A convolutional2D layers with 64 filters and kernel size is 7.
- A layer with MaxPool2D of pool size=3, strides=2 and padding="SAME"
- 3 Bottleneck ResBlock with 64 filters
- 4 Bottleneck ResBlock with 128 filters
- 6 Bottleneck ResBlock with 256 filters

- 3 Bottleneck ResBlock with 512 filters
- A with GlobalAvgPool2D
- Output layer with 100 features representing 100 classes using Softmax.

5 Experimental Results

Table 3: Results of the three models on the F-MNIST data.

Model:	Inverted Triangle	Rectangle	Hexagon
Validation accuracy	0.88	0.882	0.8803
Validation Loss	0.4108	0.3571	0.3705
Sign of over fitting	Yes	Yes	Yes
95% confidence interval	(0.113, 0.126)	(0.113, 0.126)	(0.113, 0.126)

5.1 Fashion-MNIST

Table 3 shows the summary of results for each of the three models that was used on the F-MNIST data set. Each of the three models had similar results. The Rectangle architecture had the highest accuracy (0.8715) and lowest loss (0.4428) followed by Hexagon (accuracy = 0.8653, loss = 0.4682) and Inverted Triangle (accuracy = 0.864, loss = 0.485).

Each of the three models showed that there is signs of over fitting. As depicted in Figures 2, 3, and 4 these graphs shows each of the three different models depicting each architectures training accuracy, training loss, validation accuracy and validation loss after each epoch. Over fitting started happening after about 50 epochs where the training loss (train) keeps decreasing while validation loss (val) increases or remains the same.

Confusion matrices with a heat map where used to show how well each model did at predicting each of the individual labels (Figures 5, 6, and 7). The x axis is what the model predicted a given image to be and the y axis is what the actual image was. So the diagonal on the graph is marked if the model correctly labels and image and the off diagonals are marked if the model predicted a wrongly. Each model's confusion matrix showed to have relatively high accuracy at predicting labels correctly across each of the 10 labels.

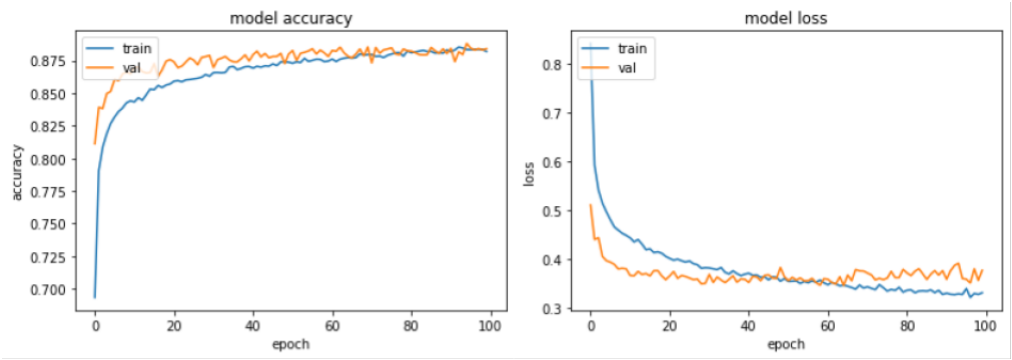

Figure 2: Inverted triangle model's accuracy (left) and loss (right) results after each epoch

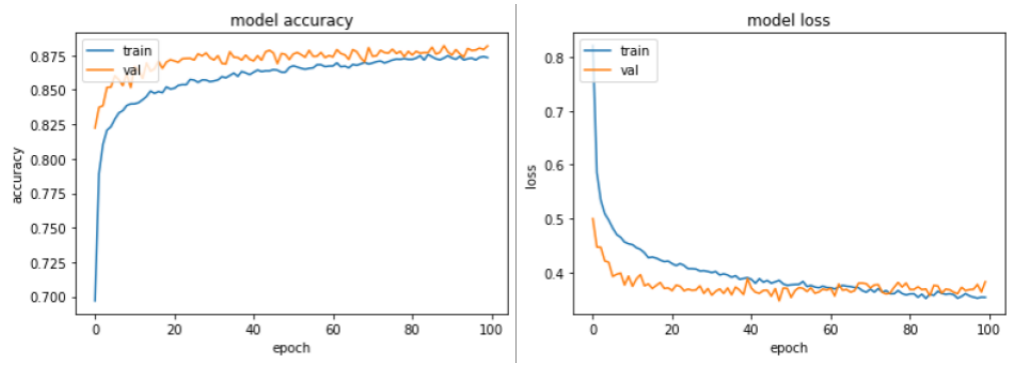

Figure 3: Rectangle model's accuracy (left) and loss (right) results after each epoch

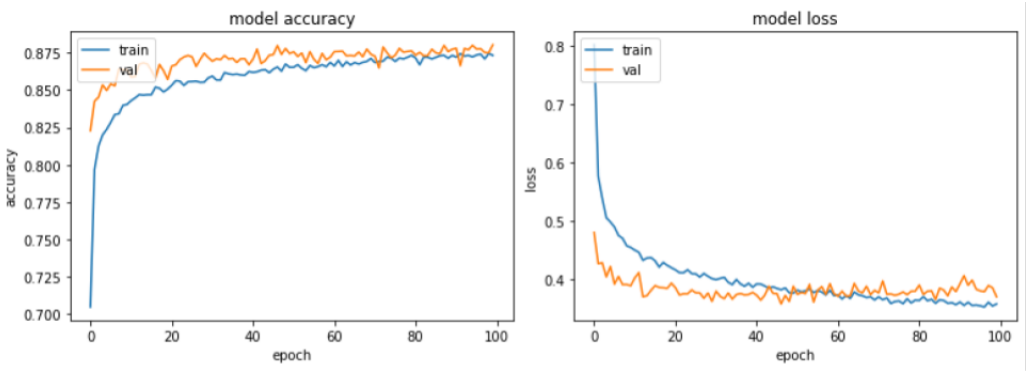

Figure 4: Hexagon model's accuracy (left) and loss (right) results after each epoch

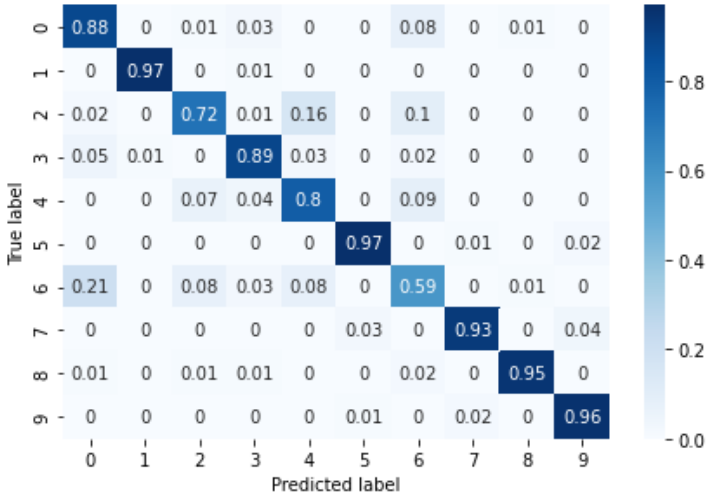

Figure 5: Inverted triangle model's confusion matrix

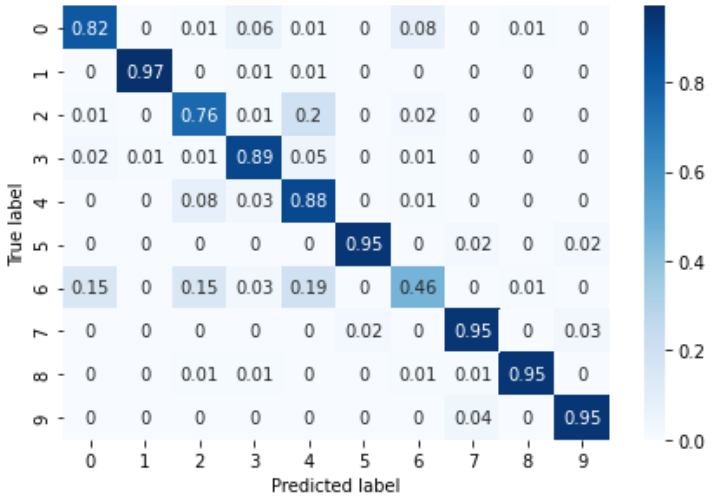

Figure 6: Rectangle confusion matrix

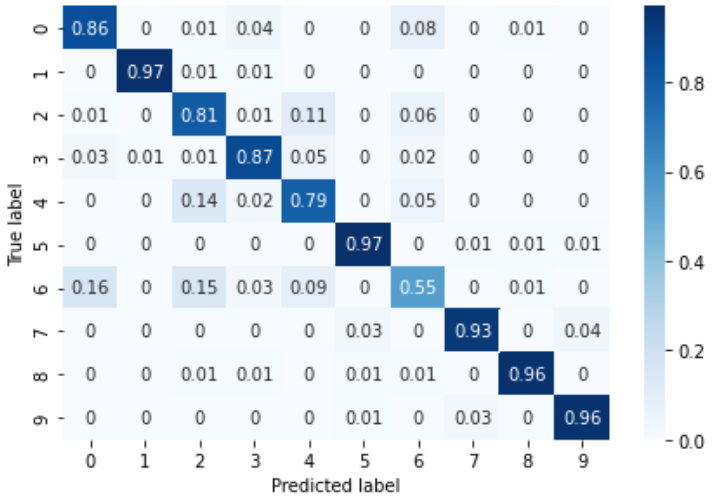

Figure 7: Hexagon model confusion matrix

5.2 CIFAR-100

The total run time for cifar100model1 took 30 minuets and cifar100model2 took 116 minutes. With a total of 645,412 parameters in cifar100model1 and 1,557,060 parameters in cifar100model2. Table 4 shows the summery of results

Table 4: Results of the two architectures on the CIFAR-100 data.

Model:	Model 1	Model 2
Validation accuracy	0.5352	0.4460
Validation Loss	1.8522	2.2410
Sign of over fitting	Yes	Yes
95% confidence interval	(0.5254, 0.5450)	(0.4363, 0.4557)

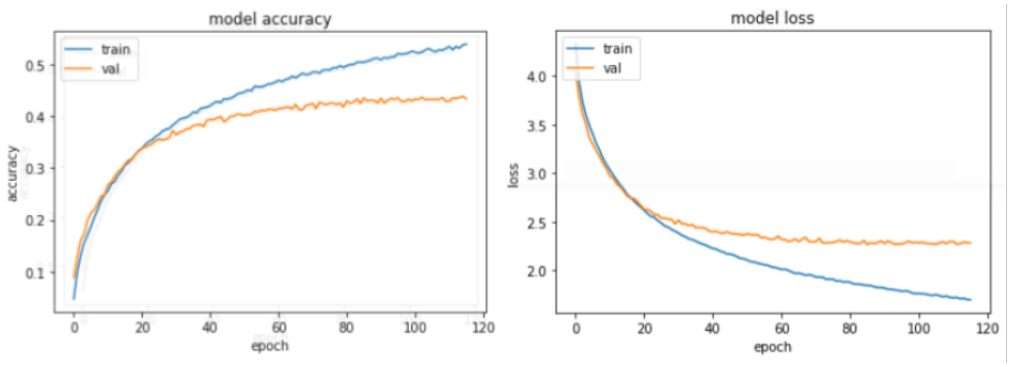

Figure 8: Cifar100model1's accuracy (left) and loss (right) results after each epoch

for the two different models that was used on the CIFAR-100 data set. Cifar100model1 had the highest overall validation accuracy at 0.5352 and lowest loss at 1.8522. Cifar100model2 had the lowest over all validation accuracy at 0.446 and highest loss at 2.241.

Both of the architectures showed that there is signs of over fitting. As depicted in Figures 8 and 9 these graphs shows both the different architectures depicting each architectures training accuracy, training loss, validation accuracy and validation loss after each epoch. Over fitting started happening after about 50 epochs for model 1 and around 20 epochs for model 2.

Confusion matrices with a heat map where used to show how well each model did at predicting each of the individual labels (Figures 10 and 11). Each model's confusion matrix showed to have relatively high accuracy at predicting labels correctly across each of the 100 labels.

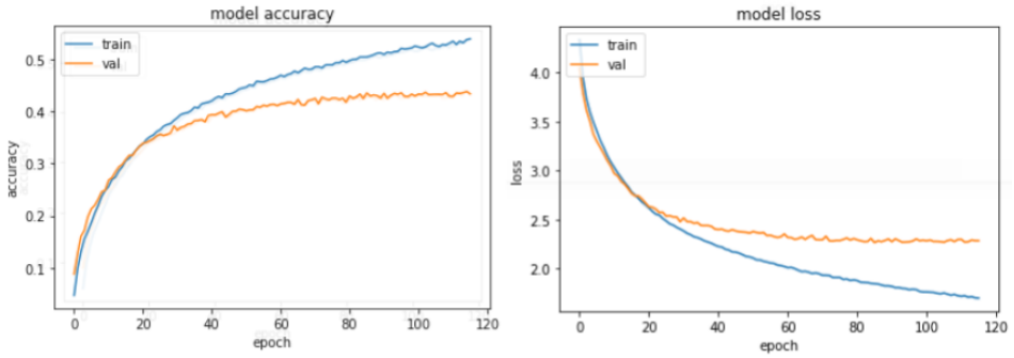

Figure 9: Cifar100model2's accuracy (left) and loss (right) results after each epoch

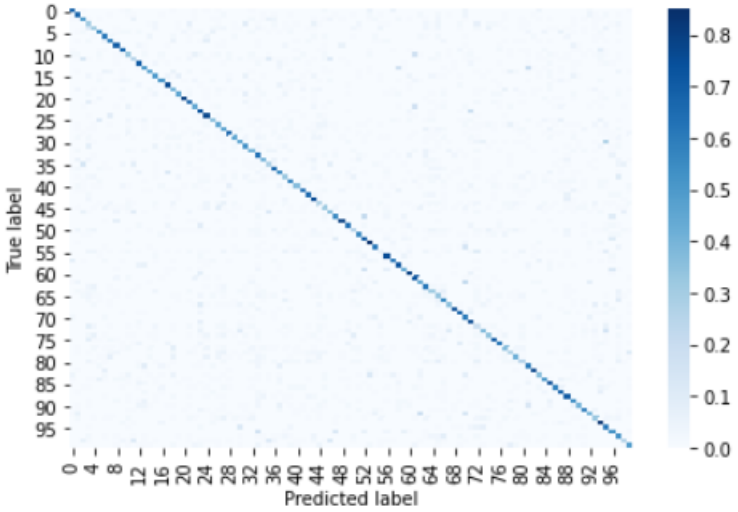

Figure 10: Cifar100model1's confusion matrix

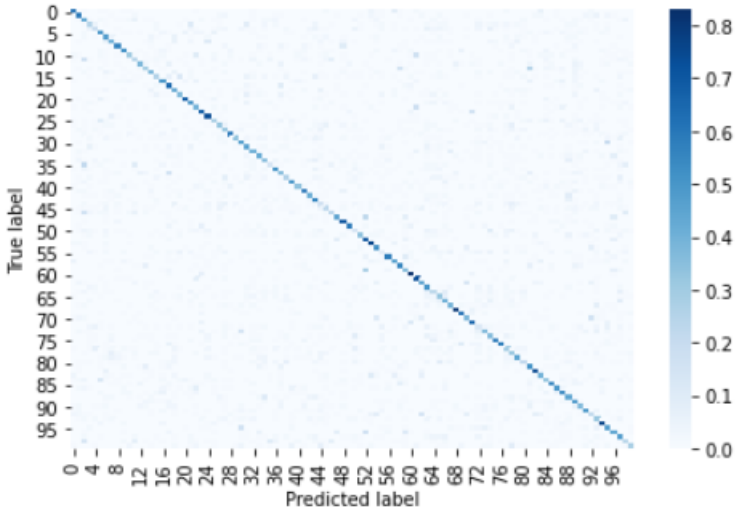

Figure 11: cifar100model2's confusion matrix

6 Discussion

6.1 Fashion-MNIST

Based on the table (Table 3), the rectangular model achieved the best validation accuracy with the lowest validation loss and error confidence interval. This meant that a balanced number of nodes in the hidden layers is important for a good generalization. Both the inverted triangle and hexagon performed around the same on the validation data set. It can also be said that due to partitioning 10,000 images for the validation data, we can get a very small and accurate 95% confidence interval for the misclassifications of the models.

Besides that, the confusion matrix for each model displays where a misclassification occurs the most. For the inverted triangle model, most labels were predicted correctly, except for label 6 (shirts). The model tends to make the mistake of labeling the image with label 0 (t-shirts) around 20% of the time. This can also be seen with how for images with true label 6, the model managed to predict it correctly only 60% of the time. For the rectangular model the performance was even worse, with only correctly predicting images with true label 6 only 46% of the time. This can also be seen with the Hexagon model, getting the prediction correctly only 55% of the time. This implies that images with label 6 is tricky to understand and thus more difficult to predict.

The graph results displays how all three models had massive over fitting. This was possibly due to the combination of the ReLU activation function and high rates of dropout. This caused all three models to suffer from many inactive

nodes which leads to over fitting and thus lowers the effectiveness of each model's prediction capabilities for unknown data. The over fitting of the model could have been one of the major reasons for the misclassification of label 6 images.

6.2 CIFAR-100

From the results, we concluded that the first model had higher accuracy compared to the second model (52.52% to 44.60%). Both model exhibited signs of over fitting because the training accuracy is about $10\% - 20\%$ higher than validation accuracy. The models learned much slower after epoch 75 for the first model and after epoch 50 for the second one. One surprising result was that the run time for the first model is much quicker compared to that of the second model. This could be due that the total parameters of the first model were only two-fifth of those of the second model. However, if we took into account the whole model, the second model actually ran quicker as it had to process almost 50 convolutional layers with much more filters compared to 5 in the first model. While we could have added as many convolutional layers to the first model to observe how much speed advantage the second model has, we concluded that it was impossible since a model with that much layers without the projection block of Resblock would take too long to train and too costly computationally, rendering it impractical in reality.

The result contradicted our hypothesis in that the first model has a higher accuracy than the second model. Since this came as a surprise, we took some time to discuss possible reasons behind it. We proposed some possible explanations for why there is such discrepancy:

- The amount of data available for training was only $40,000$, but the second model has too many parameters to train.
- The addition of dropout layers after the convolutional layer in Bottleneck may be responsible for the difference in accuracy because dropout and Batch Normalization are both regularizers. Since we used both of them, it might have an adverse impact. After some tests, we concluded that it might not be the reason in our model as removing the dropout layer drops the accuracy down significantly (44% to 27%).
- Other optimizers could be used to better fit the second model.
- We did not implement Bottleneck appropriately. As we needed to modify the given code to make it suitable for our usage, we might have changed some structures that impacted on the final accuracy. We did attempt to limit such risks by checking that the model still performs as the concepts of Bottleneck Resblock architecture described.

7 Conclusions

7.1 Fashion-MNIST

All three models achieved good results, but the trend in increasing validation loss when training shows that over fitting is a major issue for the models. To improve these models and achieve better results, implementation of more hidden layers combined with other activation functions is a good place to begin. By using activation functions like leaky relu, gelu or elu, it might reduce the amount of dead nodes in the model. This could allow the models to generalize better the images patterns and make better predictions. Besides that, lowering the rate of dropout could also help the learning process. Since it was displayed that high rates of dropouts led to over fitting and thus weaker prediction capabilities, by reducing the dropout rates it could allow more nodes to train and learn the information.

7.2 CIFAR-100

The two models for CIFAR-100 achieve fairly good validation accuracy. Nevertheless, there were still many rooms for improvement as the models still exhibited overfitting and the accuracy could be further improved as evidenced by the leaderboard showing that there were teams that had achieved $> 80\%$ validation accuracy. If we were to continue this project, we would refine the Bottleneck architecture to increase accuracy. We would do more research into Bottleneck to find out more ways to leverage the benefits of such architecture. We would also use different activation and optimizer as well. As noted in the FMNIST problem, using dropout after ReLU is risky as it increased the average of all data instead of keeping it the same (hackathon 3 also discusses this). We would also do data augmentation to diversify the data and allows the model to not only have more data to train but also avoid overfitting.