

## **Assignment 4: Deep Learning with Fruits-360**

Gavyn Gallagher

Northwestern University

MSDS 458: Artificial Intelligence and Deep Learning

Syamala Srinivasan

December 5, 2022

### Abstract

Company X is seeking to develop an application that uses neural networks to identify the number of calories in pictures of food. The data set Fruits-360, was used to test the classification of foods. With the hope being that if a neural network models works well on the data set then Alotghter 24 models were built using different numbers of convolutional layers as well as a variety of parameters. Ultimately the majority of the models performed well and using a model with a reduced dropout rate with three-convolutional layers produced the best scores. Recommendations for Company X is to further test the best model, model 22, on more food data sets that contain images of more food types. Once this occurs and proved successful then the model can be used on the app.

### Introduction and Problem Statement

Our Company X focuses on dietary health and wants to launch a new application that can track food calories. The app would work by users being able to take pictures of their meals and food through the app and then the app would recognize what food item it is and generate how many calories are contained in that food item. A database on the app would contain a list of all food items and their calories. The difficult part would be making the model to identify the food images based on images. We have been tasked with creating a successful model that can recognize food so then the app can implement the technology.

To test our model, the data set Fruits-360 from Kaggle.com will be used. Once the model can be run successfully and proven its accuracy then the model can be applied to the app. Fruits-360 contains images only of fruits and vegetables. Many of the foods included in the fruits-360 dataset users of the new app will use, thus Fruits-360 was deemed a good data set to use. In the training and testing data sets, there are 24 classes such as Pears, Zucchini, apples, eggplants, and apples. A successful model will be able to classify all 24 classes at a moderate to strong accuracy rate.

In this assignment, three experiments were conducted to build a successful model for Company X. All experiments focus on using the supervised learning Convolutional Neural Networks (CNN). Experiment A works with one convolutional layer, Experiment B with two convolutional layers, and lastly experiment C with three convolutional layers. 8 models will be tested in each experiment, each model tests a different parameter. First, a base model will be made then one parameter will be altered at a time. Parameters tested include padding, regularization, kernel size, stride size, dropout rate, and batch normalization. One model in each experiment will have all the changed parameters activated. All models' accuracy and loss will be

evaluated. The best model will be recognized and recommendations for what Company X should do next will be stated.

## **Literature Review**

### **A Review on Fruit Recognition and Feature Elevation Using CNN**

This article focuses on using convolutional neural networks to classify foods using the fruits-360 dataset. The goal was to help build a classification system for the fruits that could allow sellers to better differentiate their food.

First, the article depicts why fruits can be troublesome to classify due to the colors of the fruits themselves paired with the light and high occlusion. Next, the CNN architecture is explained of the model they created. The data is normalized using Relu and a pooling layer was used. CNN fully connected layers are then explained.

This is an important article as it details how CNN models work and why they are successful on classifying images. Also, the use of pooling layers reduces the size of the images after each convolutional layer. The use of pooling layers can prevent overfitting and should be used in our experiment.

### **Fruit Recognition From Images Using Deep Learning**

This article by Horea Mureasan and their team is about using the fruits-360 dataset to classify all the fruit and vegetable images. This 38 page article goes in depth about how CNNs work, the data preparation, as well as the python code of the CNN model. From this article we are hoping to learn more about how to build CNNs to the best ability with the Fruits-360 dataset.

First the article breaks down how a CNN works with the convolutional layers and pooling layer. Additionally loss layers and recurrent neural networks are explained. Next the data

preparation occurs where all 131 fruits and their counts are evaluated. The architecture of the CNN is explained through the tensorflow code. 10 models were created with the model with 4 convolutional layers, 2 fully connected layers, a kernel size of (5,5) being used.

This article is helpful in understanding the tensorflow code of a CNN model. The use of a different number of convolutional layers was interesting as well as the parameters they chose to alter. From this article I learned how to properly experiment with the Fruit-360 dataset.

### **Method(s)**

Data preparation was performed on the Fruit-360 data set. Image sizes were set to 100X100 pixels during image preprocessing. Also, the batch size was set to 128. Then images were rescaled by dividing them all by 255. In the training data set, there were 6,231 images belonging to 24 classes. While 3,110 images were in the test dataset.

In the three experiments, multiple parameters and the number of convolutional layers were tested. CNN (citation how it works). Parameters tested included: padding, regularization kernel size, stride size, dropout rate, and batch normalization. Padding helps add pixels or 0s to an image as the image size will decrease with each kernel scan (Chen, 2017). Regularization is used for simplifying the model and combatting overfitting (Haque et al., 2022). Strides are the number of pixels the filter moves from one position to the next (Chen, 2017). Dropout rates are used to drop a set amount of data with the goal of preventing overfitting. Batch normalization is where the input data uses the mean of the mini-batch to create normalization (Christopher, 2021). Through these parameter changes, we are hoping to see varying performance results and find what makes the best CNN model for the Fruits-360.

## Results

### Data Preparation

In the data preparation all of the classes in the Fruits-360 data set were looked at. Names of all the classes can be seen in figure 1. There was a total of 24 different classes of fruits and vegetables. The majority of the data is from various types of apples. One of the classes of apples is rotten apples. The only vegetable class is carrots. I hypothesize carrots will classify well as their orange color make them unique. Each class has about 300+ images.

*Figure 1*

*Fruit-360 24 Classes*

```
Class: apple_6  
Class: apple_braeburn_1  
Class: apple_crimson_snow_1  
Class: apple_golden_1  
Class: apple_golden_2  
Class: apple_golden_3  
Class: apple_granny_smith_1  
Class: apple_hit_1  
Class: apple_pink_lady_1  
Class: apple_red_1  
Class: apple_red_2  
Class: apple_red_3  
Class: apple_red_delicios_1  
Class: apple_red_yellow_1  
Class: apple_rotten_1  
Class: cabbage_white_1  
Class: carrot_1  
Class: cucumber_1  
Class: cucumber_3  
Class: eggplant_violet_1  
Class: pear_1  
Class: pear_3  
Class: zucchini_1  
Class: zucchini_dark_1
```

### Experiment A

One Convolutional layer was used in Experiment A models. Eight models were constructed where the first model is the baseline model, and models 2-7 each test one parameter, and model 8 has all parameters changed. We are hoping to learn how each parameter alters the

accuracy and loss scores of one convolutional layer CNN model. Eventually comparing the results of the one convolutional layer CNN models to two and three convolutional models.

When it comes to scores, the majority of the models performed well and similarity to each other. Accuracy and loss plots of models 1-8 can be seen in figure 2. One of the models, model\_2, had poor accuracy and loss and did not come away near the results of the others.

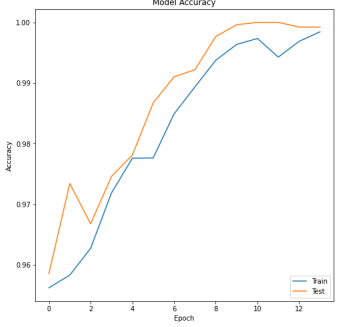
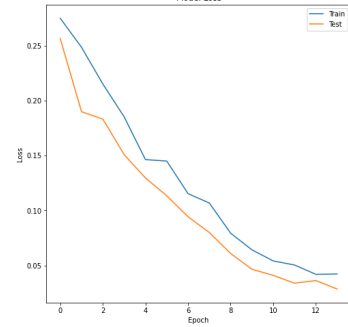
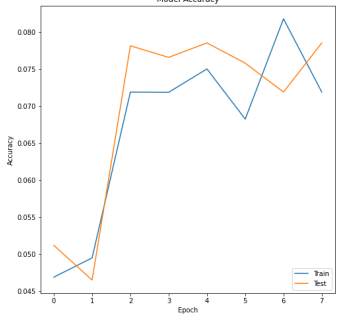
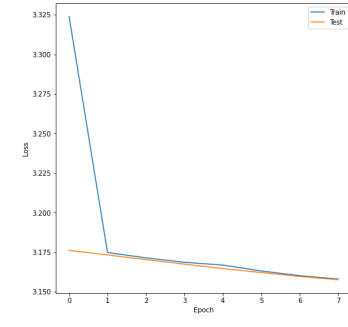
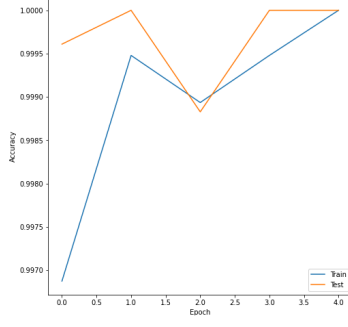
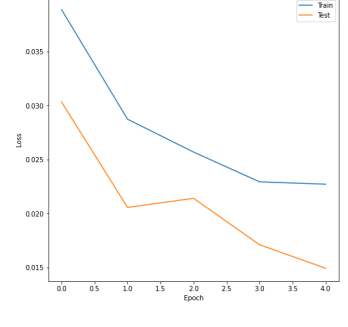
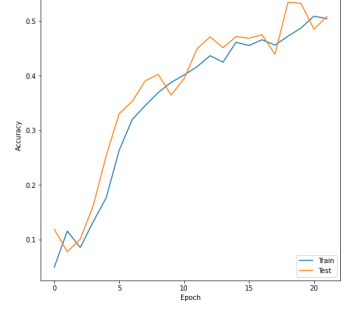
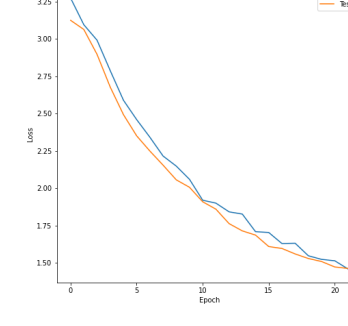
Padding was the parameter tested in model\_2. Another reason model\_2 did the worst is that it had a small number of iterations. Several models ran for over 40 iterations and generally performed better. A couple of the models hit perfect training and testing accuracy. No model achieved a perfect 0 in loss, overall the loss scores showed more variety. Increasing the kernel size to (5,5) dramatically decreased the results from the base model. Batch normalization interestingly did well on the training data and only okay on the testing. When all parameters' accuracy and loss scores were outstanding.

- Highest Training Accuracy: model\_3 and model\_8
- Highest Validation Accuracy: model\_3 and model\_8
- Lowest Training Loss: model\_1
- Lowest Validation Loss: model\_1

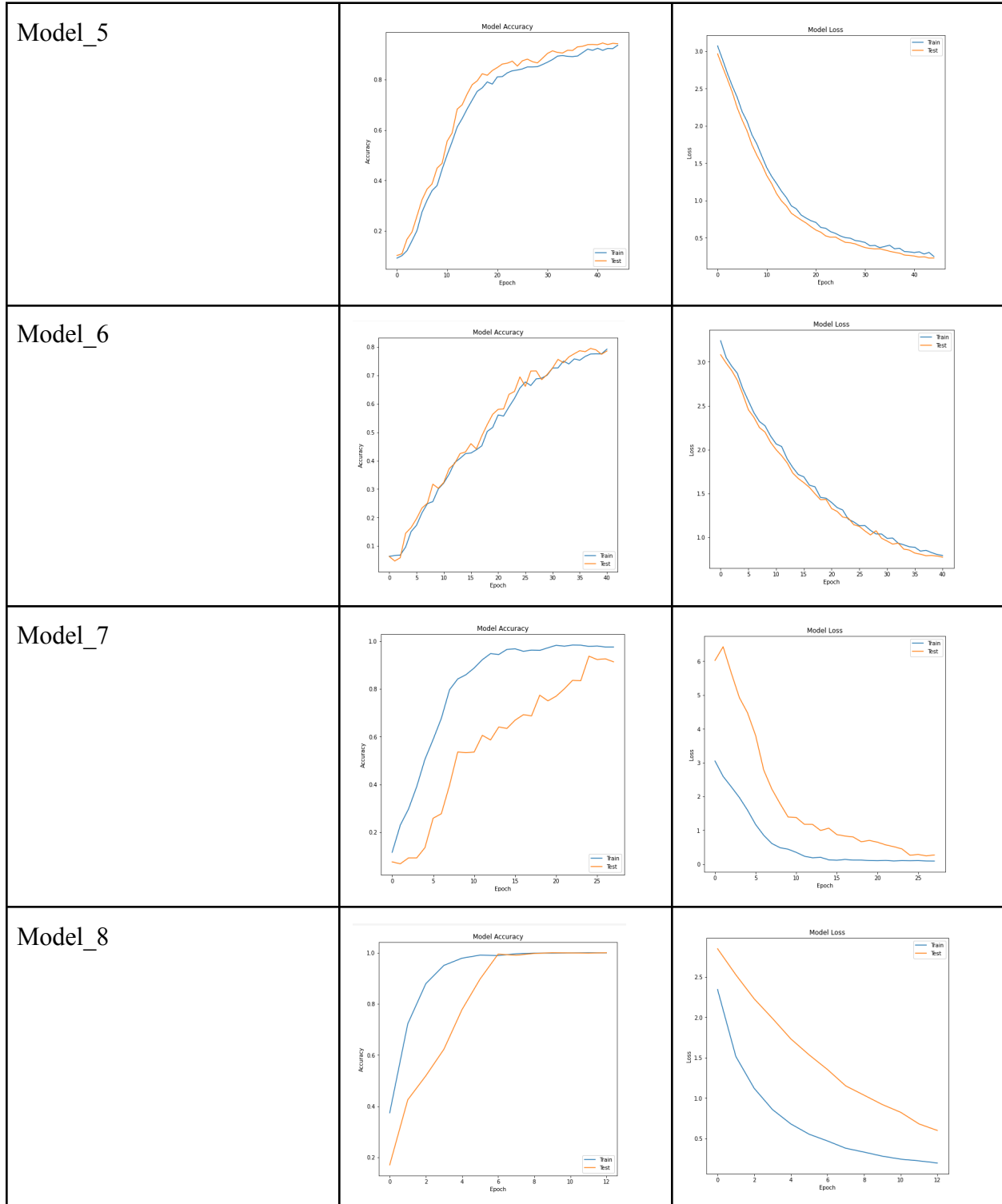
*Figure 2*

*1 convolutional layer Models' Accuracy and Loss Plots*

Model Name	Accuracy	Loss
------------	----------	------

Model_1	 <p>Model Accuracy plot for Model_1. The x-axis represents Epochs from 0 to 12, and the y-axis represents Accuracy from 0.96 to 1.00. The Train accuracy (blue line) starts at approximately 0.955 and increases to about 0.998. The Test accuracy (orange line) starts at approximately 0.958, fluctuates, and then increases to about 0.999.</p>	 <p>Model Loss plot for Model_1. The x-axis represents Epochs from 0 to 12, and the y-axis represents Loss from 0.05 to 0.25. The Train loss (blue line) starts at approximately 0.26 and decreases to about 0.04. The Test loss (orange line) starts at approximately 0.24 and decreases to about 0.03.</p>
Model_2	 <p>Model Accuracy plot for Model_2. The x-axis represents Epochs from 0 to 7, and the y-axis represents Accuracy from 0.045 to 0.080. The Train accuracy (blue line) starts at approximately 0.046 and increases to about 0.072. The Test accuracy (orange line) starts at approximately 0.051, drops to 0.044, and then increases to about 0.078.</p>	 <p>Model Loss plot for Model_2. The x-axis represents Epochs from 0 to 7, and the y-axis represents Loss from 3.150 to 3.325. The Train loss (blue line) starts at approximately 3.325 and decreases sharply to about 3.175. The Test loss (orange line) starts at approximately 3.178 and decreases slightly to about 3.165.</p>
Model_3	 <p>Model Accuracy plot for Model_3. The x-axis represents Epochs from 0.0 to 4.0, and the y-axis represents Accuracy from 0.9970 to 1.0000. The Train accuracy (blue line) starts at approximately 0.9968 and increases to about 0.9998. The Test accuracy (orange line) starts at approximately 0.9996, peaks at 0.9999, and then decreases to about 0.9988.</p>	 <p>Model Loss plot for Model_3. The x-axis represents Epochs from 0.0 to 4.0, and the y-axis represents Loss from 0.015 to 0.040. The Train loss (blue line) starts at approximately 0.038 and decreases to about 0.022. The Test loss (orange line) starts at approximately 0.030 and decreases to about 0.015.</p>
Model_4	 <p>Model Accuracy plot for Model_4. The x-axis represents Epochs from 0 to 20, and the y-axis represents Accuracy from 0.1 to 0.5. The Train accuracy (blue line) starts at approximately 0.05 and increases to about 0.5. The Test accuracy (orange line) starts at approximately 0.1, fluctuates, and then increases to about 0.5.</p>	 <p>Model Loss plot for Model_4. The x-axis represents Epochs from 0 to 20, and the y-axis represents Loss from 1.50 to 3.25. The Train loss (blue line) starts at approximately 3.25 and decreases to about 1.5. The Test loss (orange line) starts at approximately 3.1 and decreases to about 1.5.</p>





Moving on, the accuracy and loss scores of models 1-8 can be seen in figure 3. In the visual, it also shows each model is different by the parameter changes and processing time.

Changing all the parameters resulted in the longest processing time. The figure confirms that model\_3 and model\_8 did the best at accuracy. The best at loss was model\_3 and model\_1.

Overall Model\_3, where L2 regularization of 0.001 was used, was the best of one convolutional layer CNN model. Interestingly by decreasing the dropout rate in model\_6 the accuracy and loss performed worse than the baseline. The baseline had such great scores that it was difficult for other models to surpass it.

*Figure 3*

*1 convolutional layer Models' Scores*

Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_1	0.9987	0.02904	No	No	(3,3)	-	0.5	No	0:08:28
Model_2	0.0752	3.1573	Yes	No	(3,3)	-	0.5	No	0:05:12
Model_3	1	0.01533	No	Yes	(3,3)	-	0.5	No	0:02:58
Model_4	0.5035	1.4742	No	No	(5,5)	(1,1)	0.5	No	0:13:57
Model_5	0.944	0.2245	No	No	(3,3)	(2,2)	0.5	No	0:25:29
Model_6	0.7909	0.7681	No	No	(3,3)	-	0.3	No	0:25:46
Model_7	0.9128	0.2761	No	No	(3,3)	-	0.5	Yes	0:17:04
Model_8	1	0.6006	Yes	Yes	(5,5)	(2,2)	0.3	Yes	0:31:49

## Experiment B

Two convolutional layer CNN models were made in Experiment B. Models 9 - 16 are the two convolutional CNN models. The same parameters as in Experiment A were also tested.

Figure 4 displays model 9 - 16 accuracy and loss plots. All of the models proved to have high accuracy and low loss for training and validation data. The plots for each are so close that it is hard to distance which models did better over the others. In Experiment A, a few of the models performed poorly such as when the padding and kernel sizes were changed. While here in Experiment B, those same padding and kernel size models performed well. Models evaluation scores will have an indication of which models did perform better.

*Figure 4*

*2 convolutional layer Models' Accuracy and Loss Plots*

Model Name	Accuracy	Loss
Model_9		
Model_10		
Model_11		
Model_12		

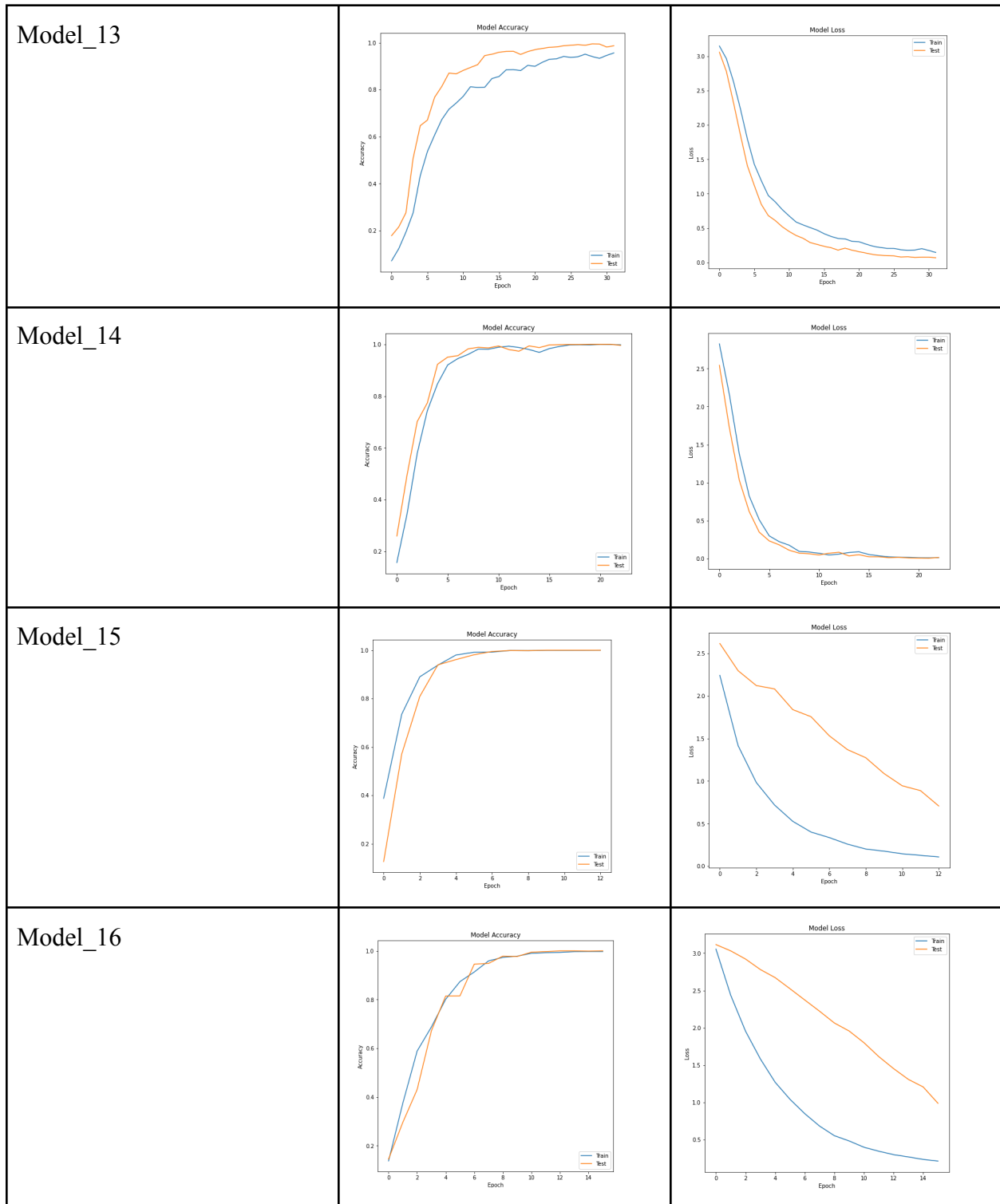


Figure 5 showcases the accuracy and loss scores for the two-layer convolutional models.

Here it is proven that there was no weak model in either score as there was in experiment 1. Four

of the 8 models achieved a perfect accuracy score. These models were where padding, regularization, batch normalization, and all parameters were changed. Regularization and all parameters being changed have produced perfect accuracy in both Experiments A and B. The lowest-performing model in accuracy was model 9 which was the base model with 2 convolutional layers. The best model by the loss was model 14 with a lower dropout rate. Interestingly model 16 had perfect accuracy but poor loss. One of the largest differences between one conventional and two convolutional models was that the two convolutional models had a longer processing time. Overall the models in experiment B did slightly better than those in experiment A. It must be decided if the extra processing time is worth the small gains.

*Figure 5*  
*2 convolutional layer Models' Scores*

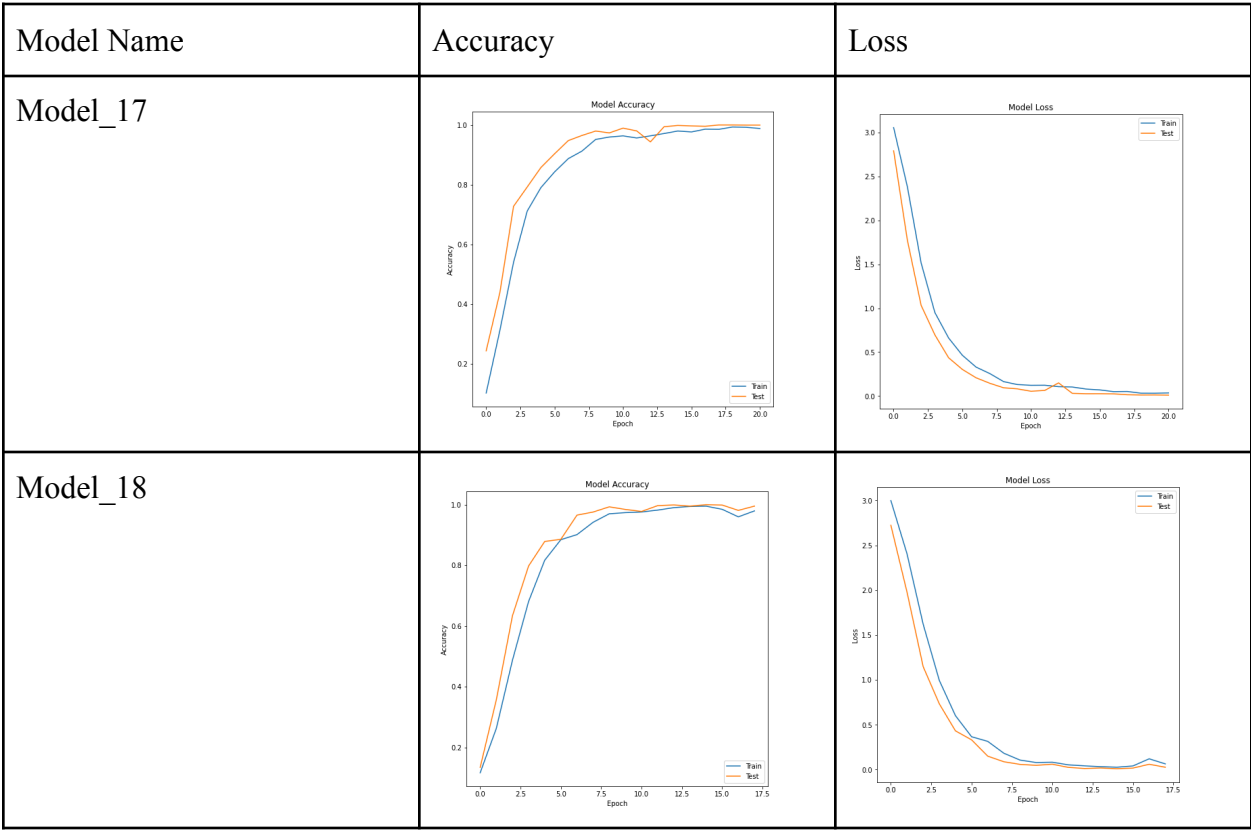
Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_9	0.9858	0.06069	No	No	(3,3)	-	0.5	No	0:08:47
Model_10	1	0.0282	Yes	No	(3,3)	-	0.5	No	0:13:49
Model_11	1	0.0282	No	Yes	(3,3)	-	0.5	No	0:11:03
Model_12	0.9952	0.0214	No	No	(5,5)	(1,1)	0.5	No	0:39:51
Model_13	0.9877	0.0673	No	No	(3,3)	(2,2)	0.5	No	0:20:24
Model_14	0.9961	0.0163	No	No	(3,3)	-	0.3	No	0:26:48
Model_15	1	0.7057	No	No	(3,3)	-	0.5	Yes	0:34:09
Model_16	1	0.9889	Yes	Yes	(5,5)	(2,2)	0.3	Yes	0:08:36

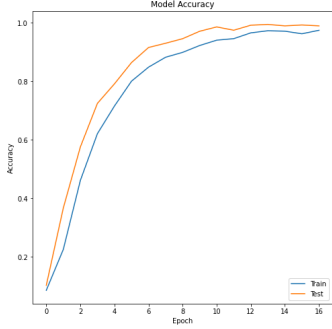
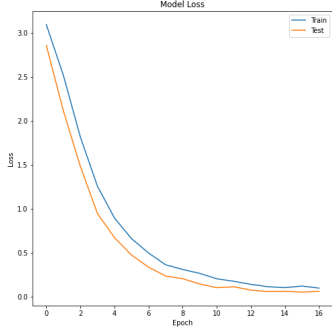
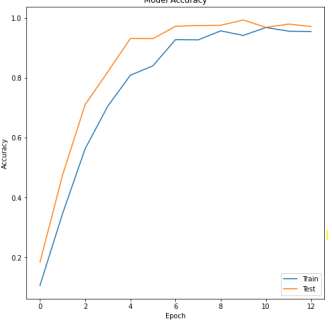
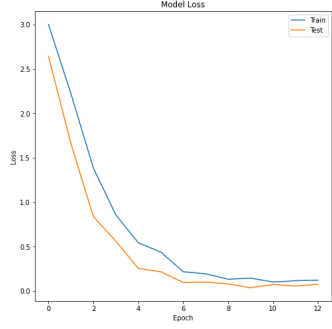
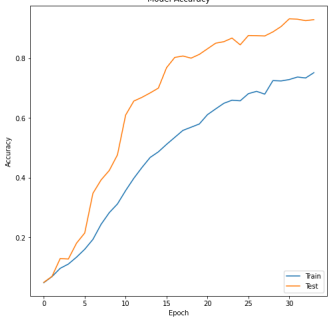
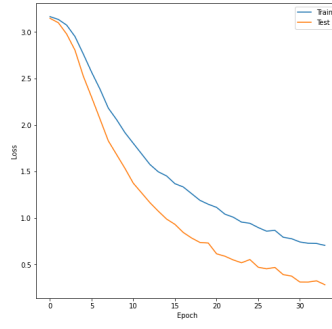
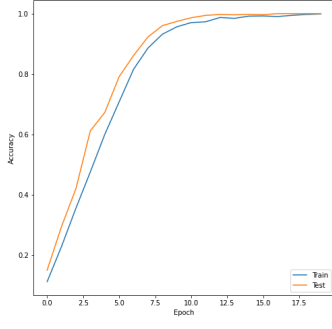
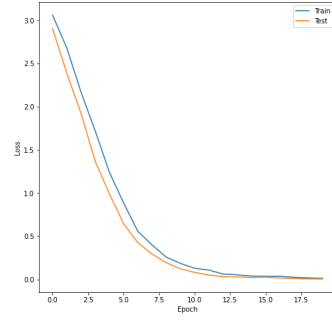
## Experiment C

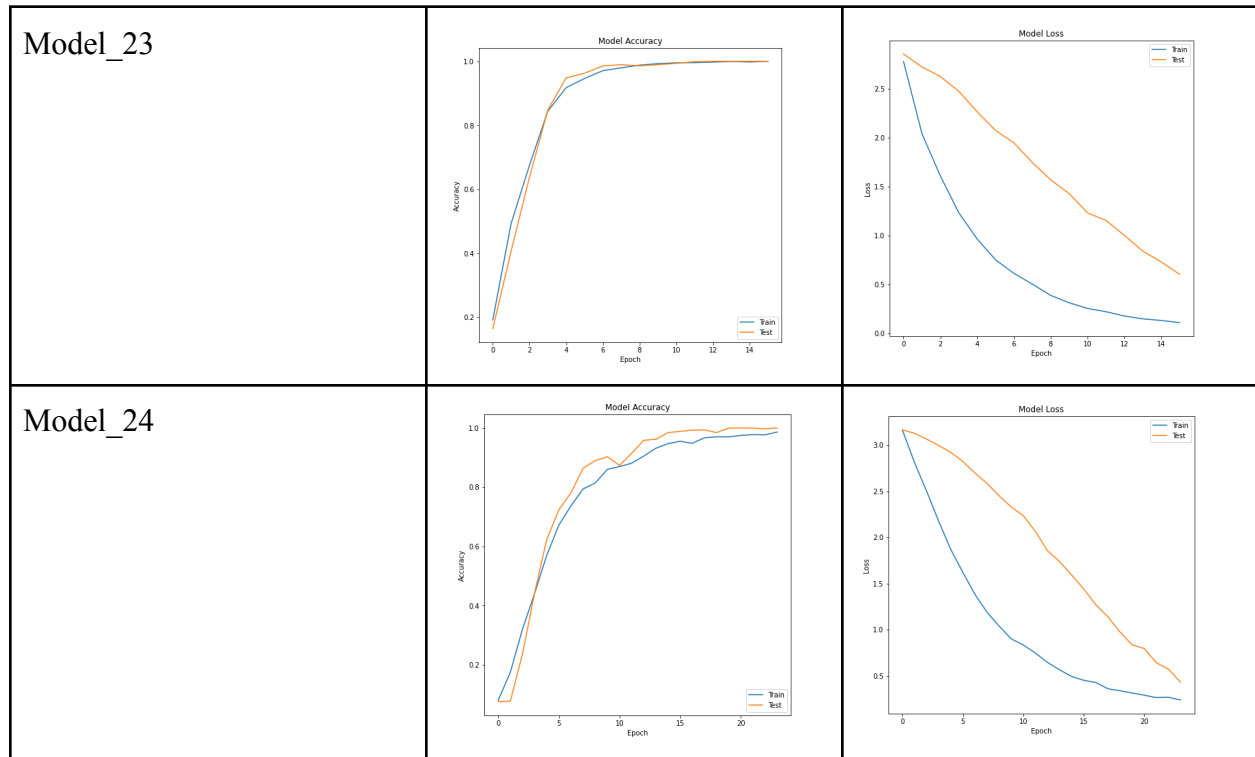
In this last experiment, 3 convolutional layers CNN models were used to test the same parameters as the previous experiments. The goal is to see how convolutional layers improve/worsen the accuracy scores compared to the one and two-convolutional CNN models. The accuracy and loss plots of models 17 - 24 are shown in figure 6. Only model 21, where the size of the stride increased to (2,2), did not have accuracy for training and validation data around 1.0. Model 21 performed similarly to how the stride change model 5 performed in experiment 1.

Meaning that two-convolutional works best for a (2,2) stride size on this dataset. It is interesting how padding and kernel size changes struggled with one-convolutional layer but did well with two and two convolutional layers. Model 23, batch normalization, had a big difference between its training and validation loss. Model 24, which had all the parameters changed also had a weaker loss, this may be due to the batch normalization. Overall three-convolutional performed very well and improved upon the one-convolutional models.

Figure 6  
3 convolutional layer Models' Accuracy and Loss Plots



Model_19	 <p>Model Accuracy plot for Model_19. The x-axis is 'Epoch' (0 to 16) and the y-axis is 'Accuracy' (0.0 to 1.0). The 'Train' (blue) and 'Test' (orange) curves both rise sharply from epoch 0 to 4, then more gradually towards epoch 16, reaching near-perfect accuracy (above 0.95).</p>	 <p>Model Loss plot for Model_19. The x-axis is 'Epoch' (0 to 16) and the y-axis is 'Loss' (0.0 to 3.0). Both 'Train' (blue) and 'Test' (orange) curves decrease rapidly from epoch 0 to 4, then level off, reaching near-zero loss by epoch 16.</p>
Model_20	 <p>Model Accuracy plot for Model_20. The x-axis is 'Epoch' (0 to 12) and the y-axis is 'Accuracy' (0.0 to 1.0). The 'Train' (blue) and 'Test' (orange) curves rise from epoch 0 to 4, then plateau, with the Test accuracy slightly higher than Train accuracy after epoch 6.</p>	 <p>Model Loss plot for Model_20. The x-axis is 'Epoch' (0 to 12) and the y-axis is 'Loss' (0.0 to 3.0). Both 'Train' (blue) and 'Test' (orange) curves decrease from epoch 0 to 4, then plateau at a low loss value.</p>
Model_21	 <p>Model Accuracy plot for Model_21. The x-axis is 'Epoch' (0 to 30) and the y-axis is 'Accuracy' (0.0 to 1.0). The 'Train' (blue) and 'Test' (orange) curves show a slower, more fluctuating increase in accuracy over 30 epochs, with Test accuracy reaching about 0.9 and Train accuracy about 0.75.</p>	 <p>Model Loss plot for Model_21. The x-axis is 'Epoch' (0 to 30) and the y-axis is 'Loss' (0.0 to 3.0). Both 'Train' (blue) and 'Test' (orange) curves decrease slowly over 30 epochs, with Train loss reaching about 0.7 and Test loss about 0.3.</p>
Model_22	 <p>Model Accuracy plot for Model_22. The x-axis is 'Epoch' (0.0 to 17.5) and the y-axis is 'Accuracy' (0.0 to 1.0). The 'Train' (blue) and 'Test' (orange) curves rise sharply from epoch 0 to 7.5, then plateau near 1.0 accuracy.</p>	 <p>Model Loss plot for Model_22. The x-axis is 'Epoch' (0.0 to 17.5) and the y-axis is 'Loss' (0.0 to 3.0). Both 'Train' (blue) and 'Test' (orange) curves decrease sharply from epoch 0 to 7.5, then plateau near 0.0 loss.</p>



Further evaluating models 17-24 figure 7 displays the accuracy and loss scores. Only two of the eight models had perfect accuracy, while experiment B had four models do that. Batch normalization has been consistently amongst the best models of the three experiments. Compared to other experiments the base model did best with three convolutional layers. Processing times continued to get longer with the additional conventional model. Accuracy was not much higher than previous models in the other experiments, but the time was much higher. OF the total 24 models the two best loss scores belong to model 22 and model 17. Lowering the dropout to .3 accounted for two of the three best loss. The best overall score is model 22 with a 56-minute processing time.

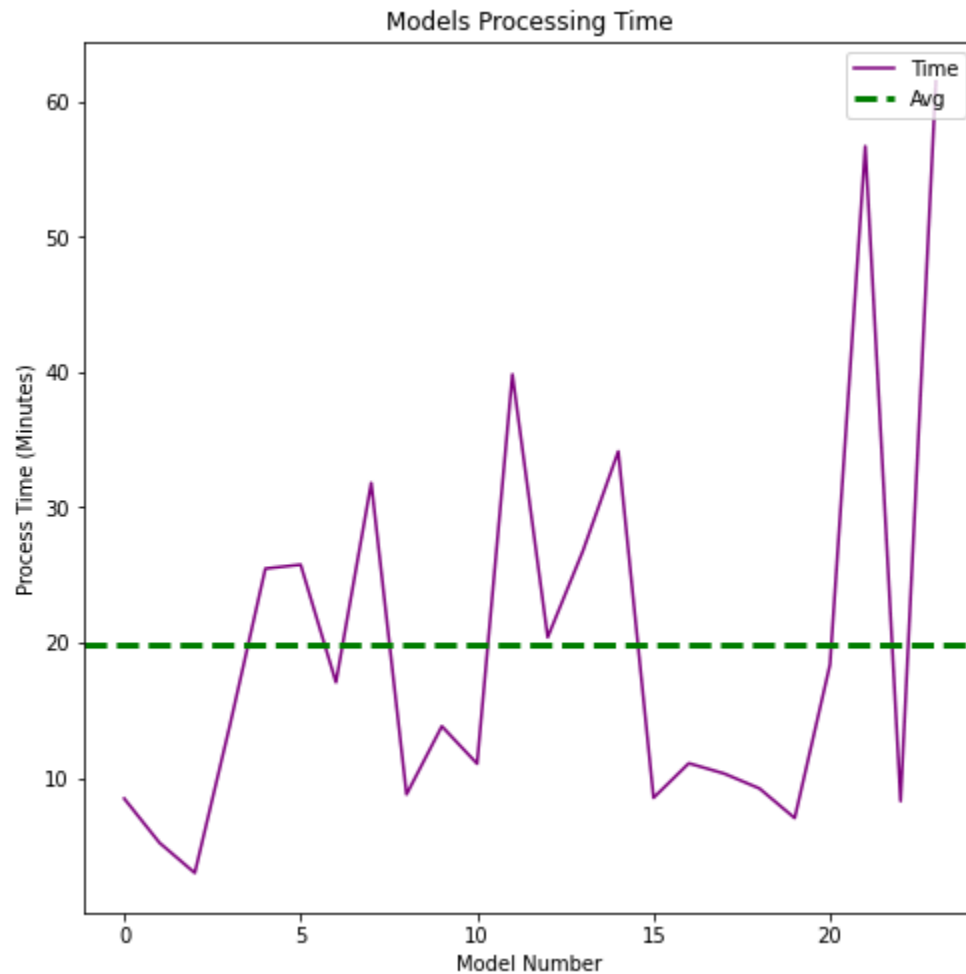
*Figure 7*  
*3 convolutional layer Models' Scores*



Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_17	0.9996	0.0097	No	No	(3,3)	-	0.5	No	0:11:05
Model_18	0.9954	0.0257	Yes	No	(3,3)	-	0.5	No	0:10:19
Model_19	0.9884	0.0625	No	Yes	(3,3)	-	0.5	No	0:09:13
Model_20	0.9733	0.0762	No	No	(5,5)	(1,1)	0.5	No	0:07:02
Model_21	0.9321	0.2808	No	No	(3,3)	(2,2)	0.5	No	0:18:23
Model_22	1	0.0056	No	No	(3,3)	-	0.3	No	0:56:43
Model_23	1	0.6057	No	No	(3,3)	-	0.5	Yes	0:08:17
Model_24	0.9996	0.4285	Yes	Yes	(5,5)	(2,2)	0.3	Yes	1:01:31

Lastly to evaluate all the models, a line chart in figure 8, shows the processing times of all the models. The dashed green line is the mean, which is about 20 minutes long. Models 1- 8 had the one conventional layer processed less time than the two and three conventional CNN models. A third of the models took longer than the 20 minute mean. Also, kernel size and strides apartments changing models consitsintly took longer to run. Processing time is a good tie breaking in diecidng which is the best model when they multiple models have the same accuracy and loss scores.

*Figure 8*  
*All Models' Processing Times*



### Conclusion

In conclusion, Company X is seeking to build an app that can take pictures of food and record how many calories the food has. This allows the users to keep track of calory intake. The technology behind the app requires a model that can identify and classify each food item. Company X has asked us to build a model capable of this using neaural networks. Our plan to build CNN models using the Fruits-360 dataset which contains thousands of images of different fruits and vegetables. If a model is proven to be accurate enough then then the model will be implemented into the new app.

Three experiments were held and 24 models were ultimately created. Experiment A used one-convolutional layer models, B used two-convolutional layers, and C with three-convolutional layer. Various parameters were tested including regularization, padding, kernel size, stride size, drop out rate, and batch normalization. Many of the models performed high on accuracy and decent on loss. The best overall model was model 22 with three-convolutional layers and a reduced dropout rate. This model had a 56 minute processing time, much longer than other models for minimal improvements. Many of the models had perfect accuracy leading to overfitting.

There are a few recommendations for what Company X should do. Models should be tested on other food datasets to include others foods beyond fruits and vegetables. This will make sure the app can work on a large variety of food. Also, to go with a slightly faster model that performs slightly worse and users will need the app to process their food images quickly. Once after the other datasets and the model running well on those then implement them into the app. Through this assignment we learned how CNN models can be used to classify food images.

### Reference

Chen, T.-H. (2017, November 24). *What is “padding” in Convolutional Neural Network?*

Machine Learning Notes.

<https://medium.com/machine-learning-algorithms/what-is-padding-in-convolutional-neural-network-c120077469cc>

Christopher, A. (2021, June 17). *Batch Normalization*. Nerd for Tech.

<https://medium.com/nerd-for-tech/batch-normalization-51e32053f20>

Haque, R. U., Khan, R. H., Shihavuddin, A. S. M., Syeed, M. M. M., & Uddin, M. F. (2022).

Lightweight and Parameter-Optimized Real-Time Food Calorie Estimation from Images

Using CNN-Based Approach. *Applied Sciences*, 12(19), 9733.

<https://doi.org/10.3390/app12199733>

Indira, D. N. V. S. L. S., Goddu, J., Indraj, B., Challa, V. M. L., & Manasa, B. (2021). A review on fruit recognition and feature evaluation using CNN. *Materials Today: Proceedings*.

<https://doi.org/10.1016/j.matpr.2021.07.267>

Mureşan, H., & Oltean, M. (2018). Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10(1), 26–42.

<https://doi.org/10.2478/ausi-2018-0002>

## Appendix

*Figure 1**Fruit-360 24 Classes*

```
Class: apple_6  
Class: apple_braeburn_1  
Class: apple_crimson_snow_1  
Class: apple_golden_1  
Class: apple_golden_2  
Class: apple_golden_3  
Class: apple_granny_smith_1  
Class: apple_hit_1  
Class: apple_pink_lady_1  
Class: apple_red_1  
Class: apple_red_2  
Class: apple_red_3  
Class: apple_red_delicios_1  
Class: apple_red_yellow_1  
Class: apple_rotten_1  
Class: cabbage_white_1  
Class: carrot_1  
Class: cucumber_1  
Class: cucumber_3  
Class: eggplant_violet_1  
Class: pear_1  
Class: pear_3  
Class: zucchini_1  
Class: zucchini_dark_1
```

*Figure 2**Model\_1 Accuracy Plot*

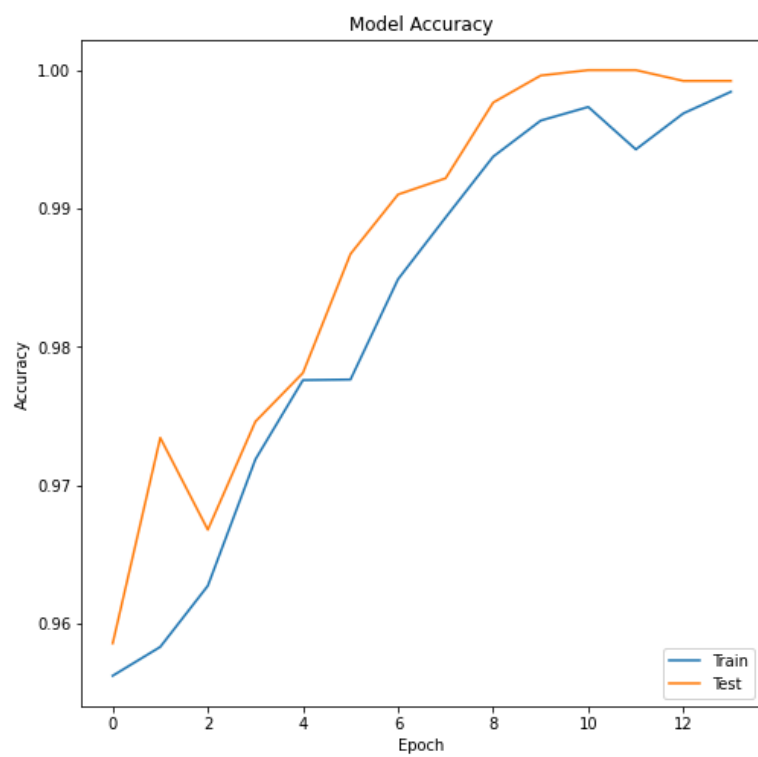
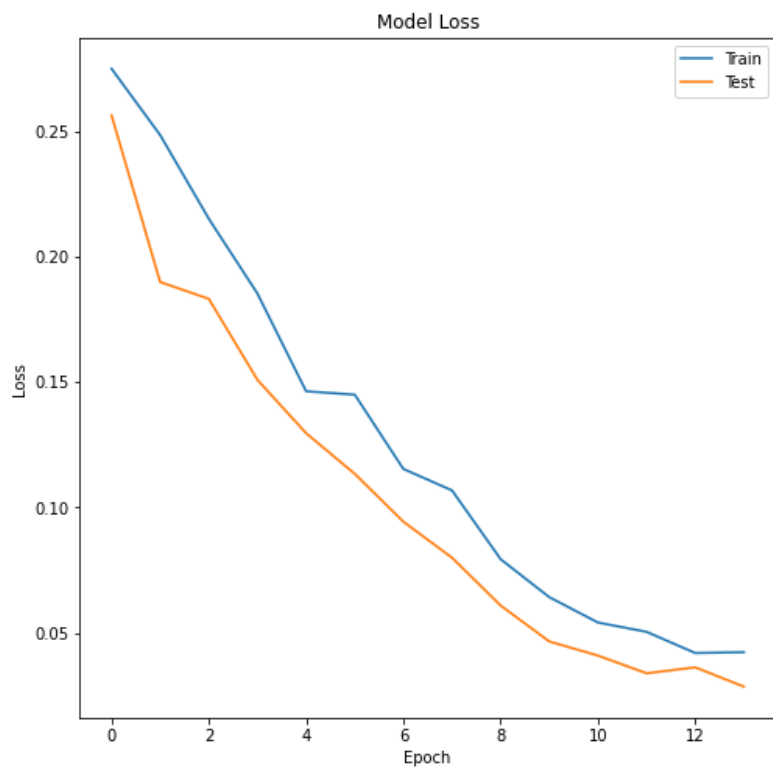


Figure 3  
Model\_1 Loss Plot



*Figure 4*  
*Model\_2 Accuracy Plot*

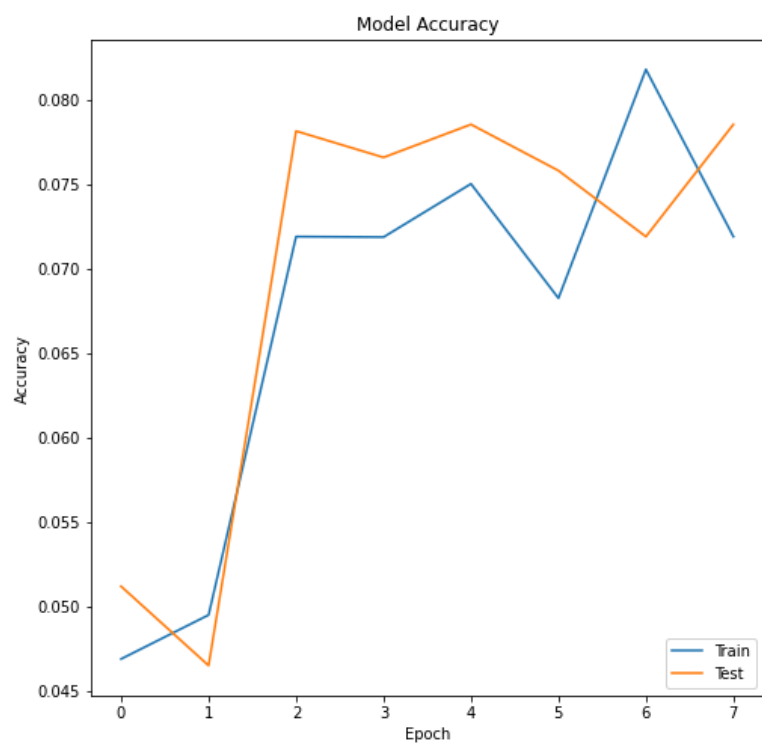


Figure 5  
Model\_2 Loss Plot



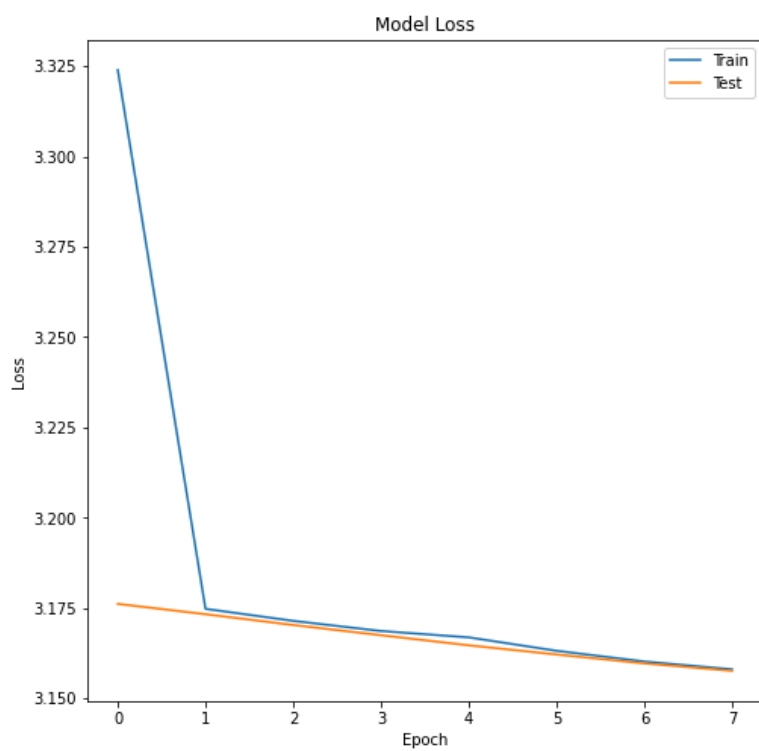


Figure 6  
Model\_3 Accuracy Plot

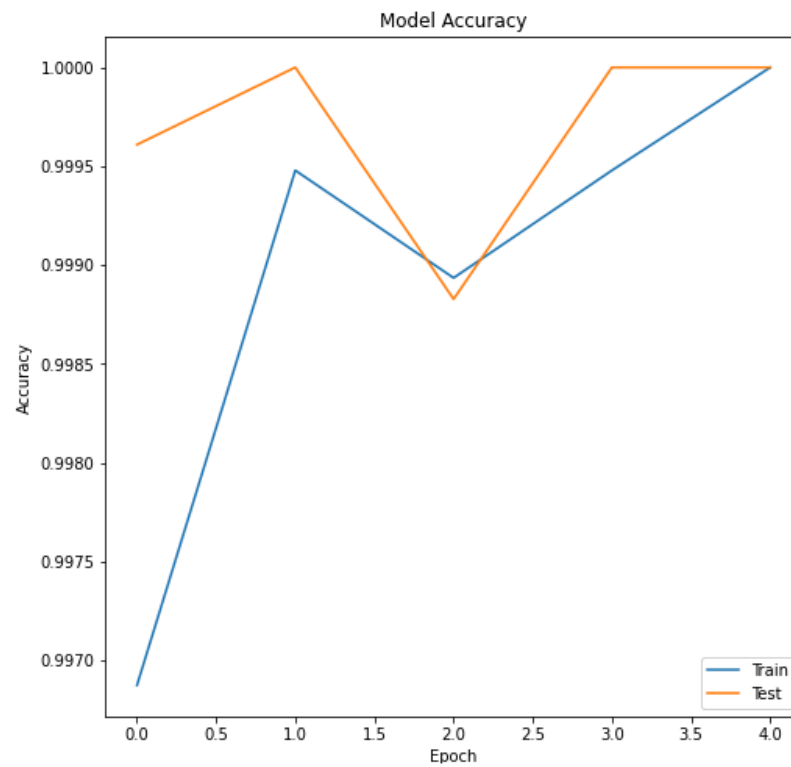
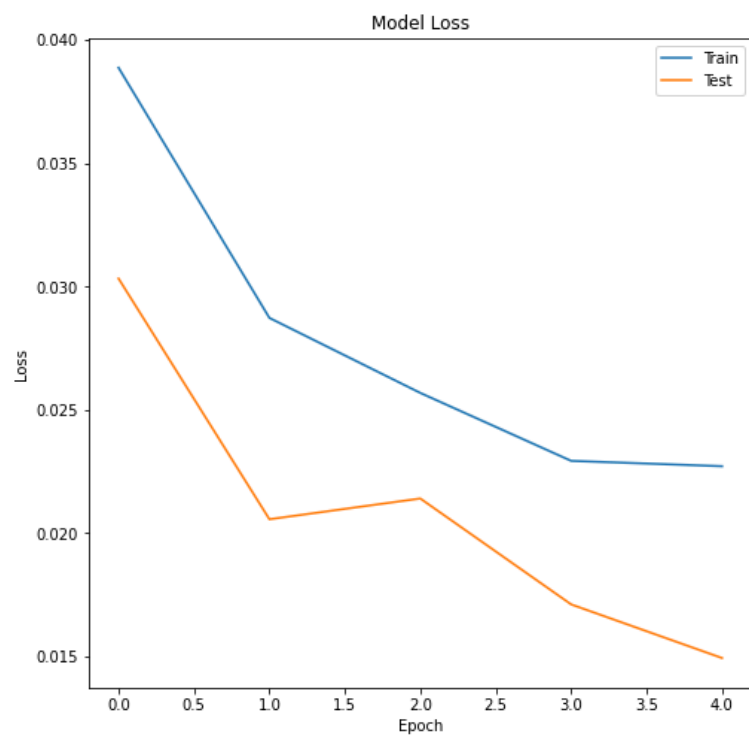


Figure 7  
Model\_3 Loss Plot



*Figure 8*  
*Model\_4 Accuracy Plot*

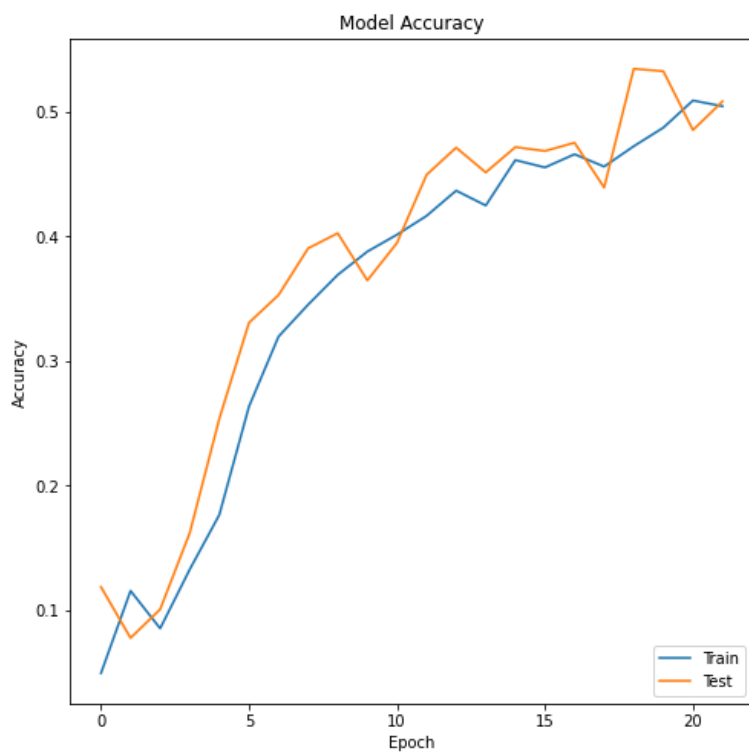


Figure 9  
Model\_4 Loss Plot

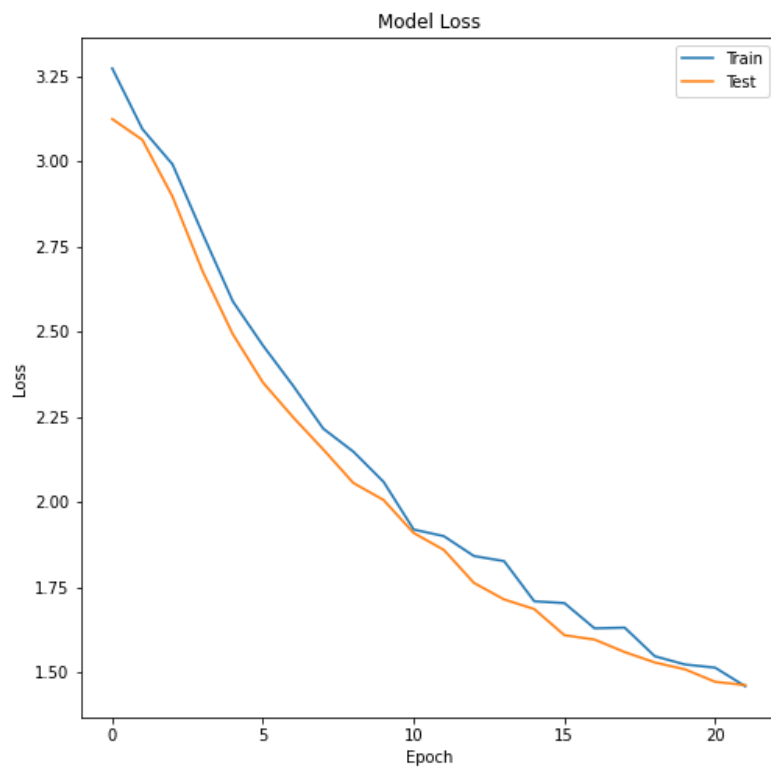


Figure 10  
Model\_5 Accuracy Plot

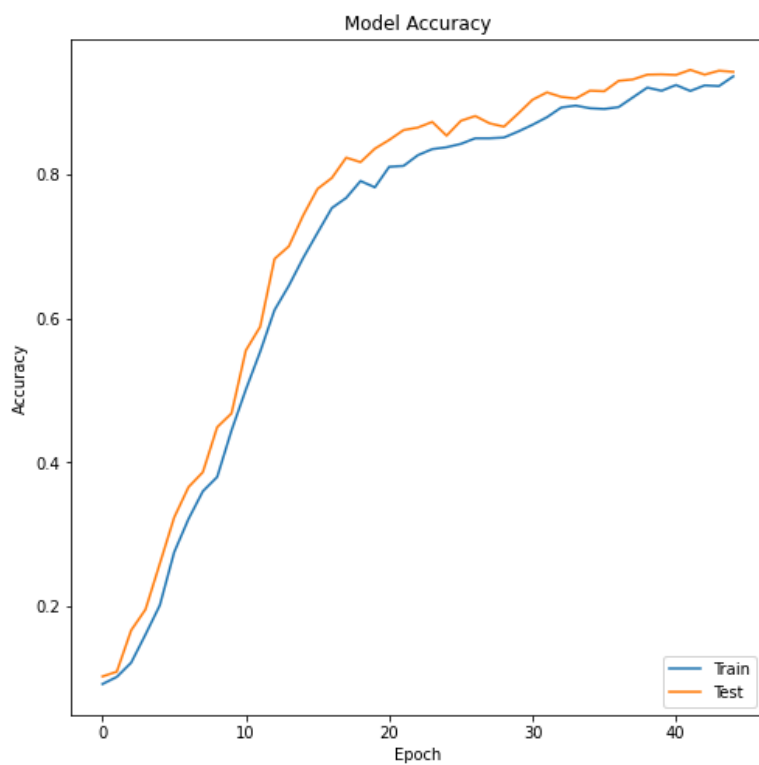
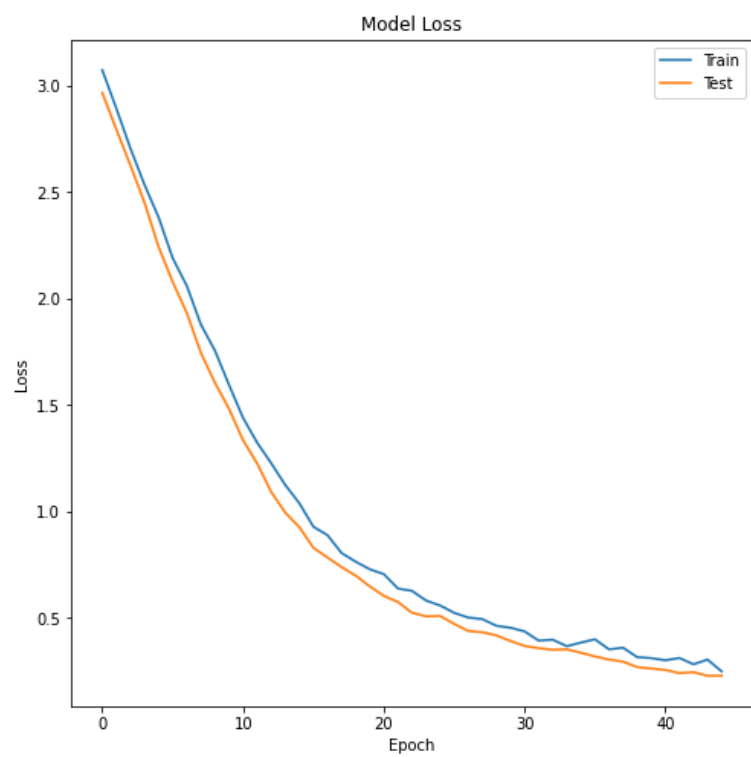


Figure 11  
Model\_5 Loss Plot



*Figure 12*  
*Model\_6 Accuracy Plot*

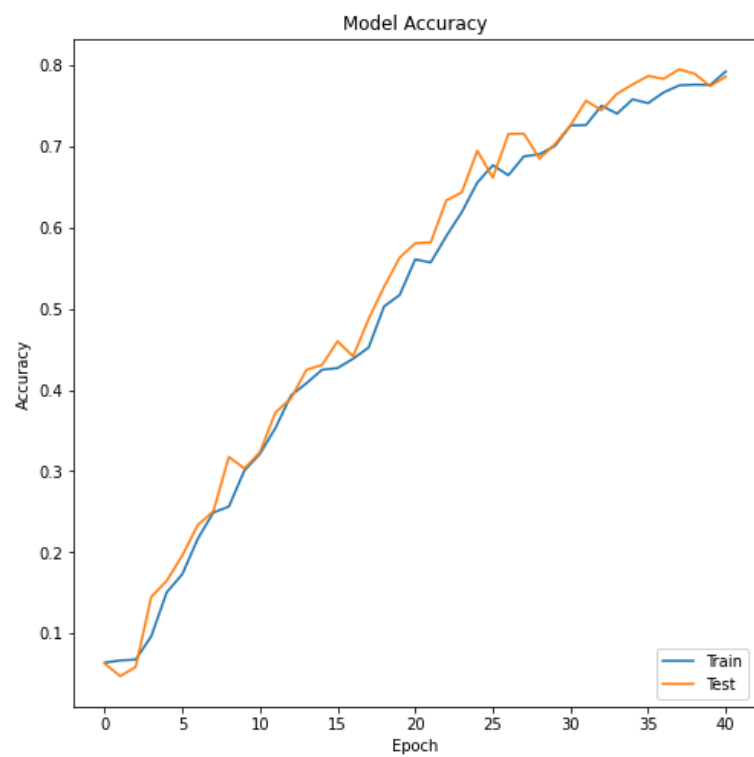


Figure 13  
Model\_6 Loss Plot



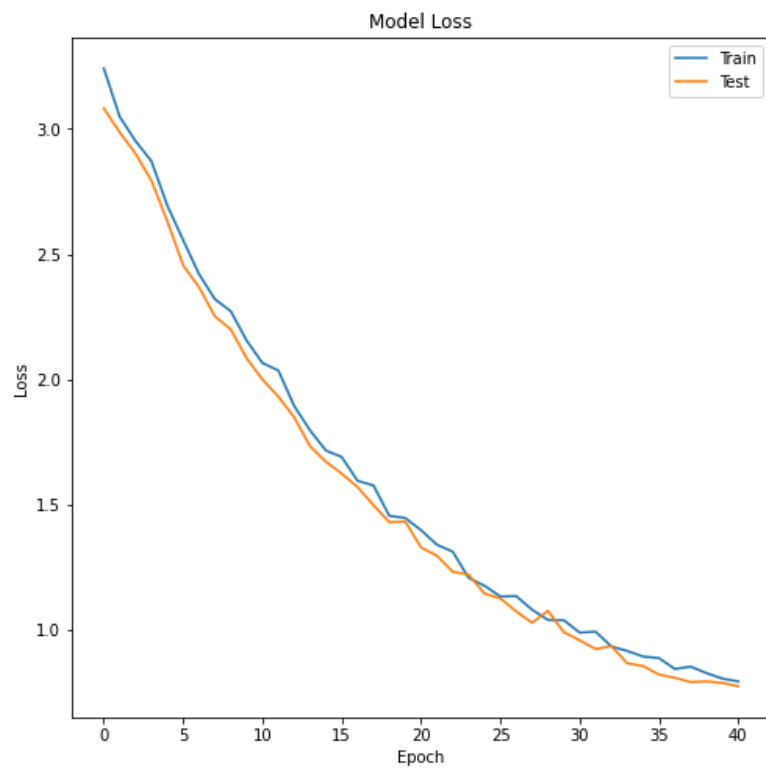


Figure 14  
Model\_7 Accuracy Plot

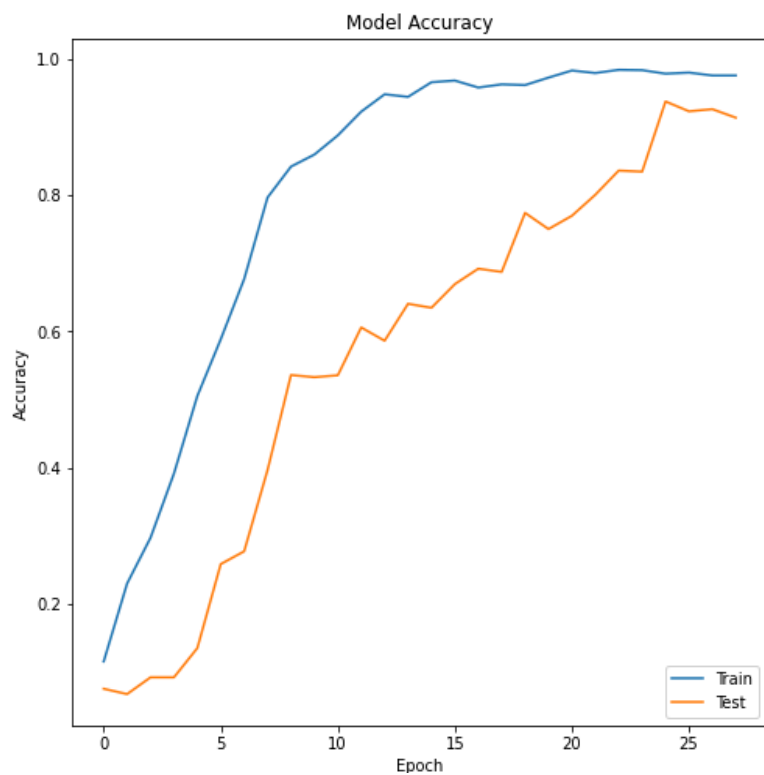


Figure 15  
Model\_7 Loss Plot

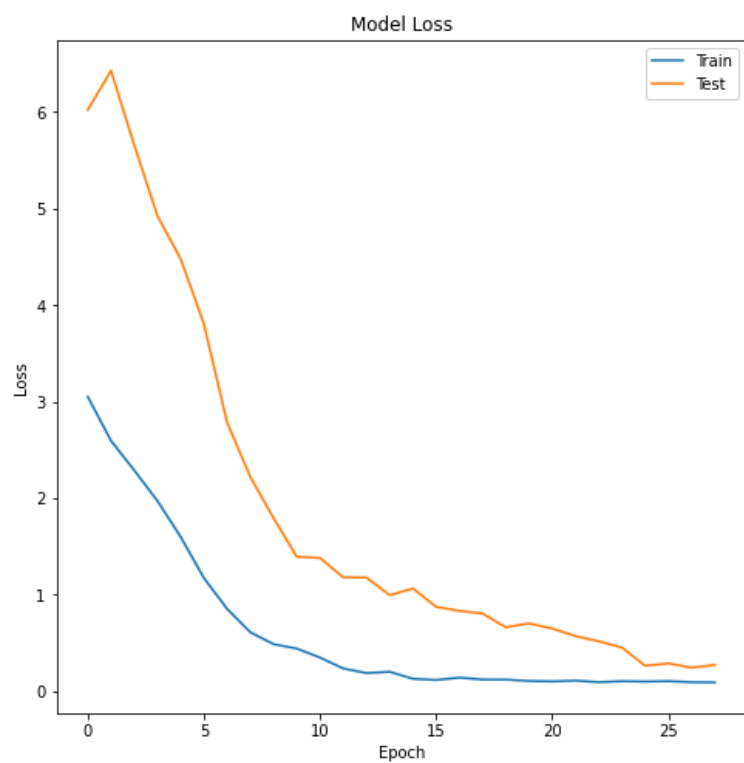


Figure 16  
Model\_8 Accuracy Plot

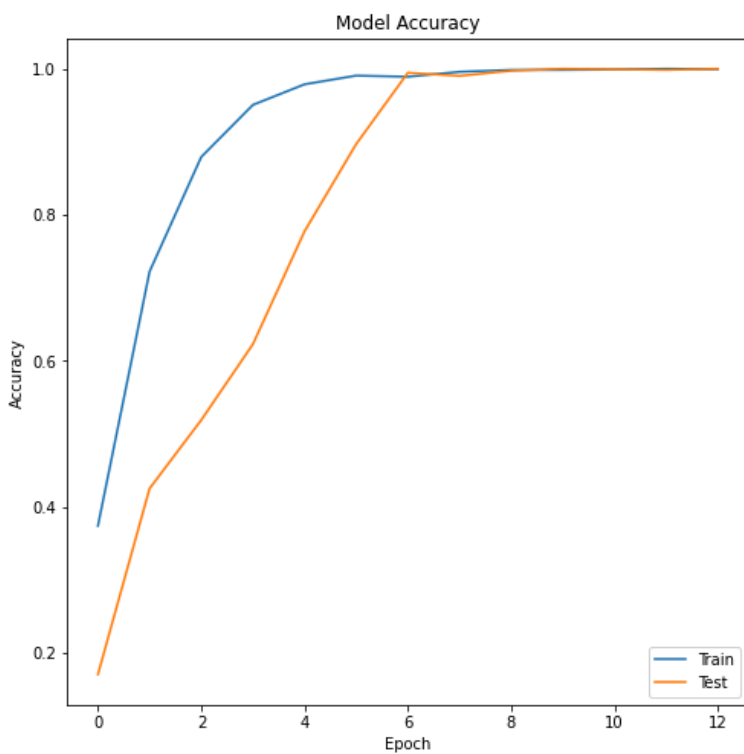
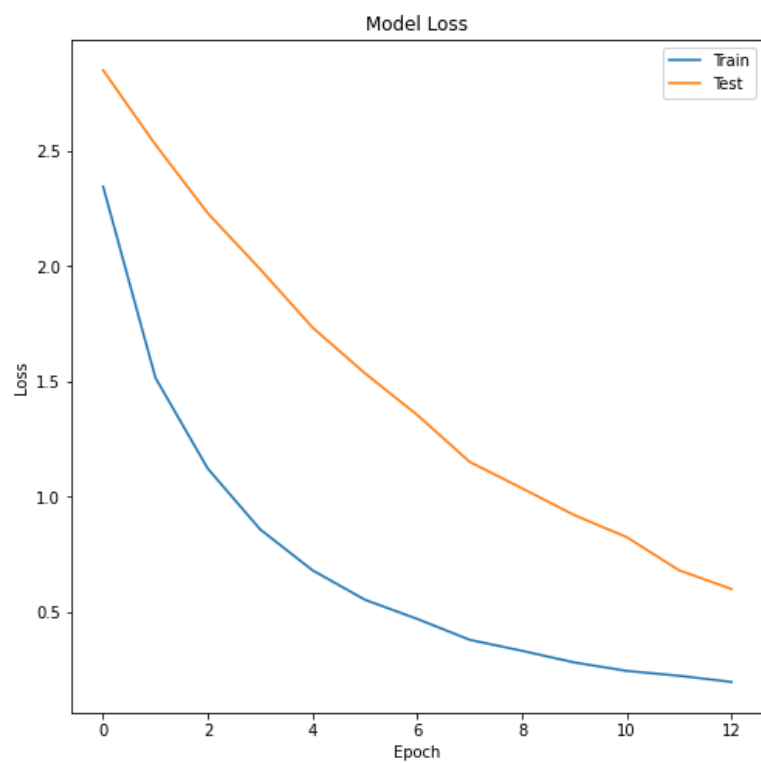


Figure 17  
Model\_8 Loss Plot



*Figure 18*  
*1 convolutional layer Models' Scores*

Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_1	0.9987	0.02904	No	No	(3,3)	-	0.5	No	0:08:28
Model_2	0.0752	3.1573	Yes	No	(3,3)	-	0.5	No	0:05:12
Model_3	1	0.01533	No	Yes	(3,3)	-	0.5	No	0:02:58
Model_4	0.5035	1.4742	No	No	(5,5)	(1,1)	0.5	No	0:13:57
Model_5	0.944	0.2245	No	No	(3,3)	(2,2)	0.5	No	0:25:29
Model_6	0.7909	0.7681	No	No	(3,3)	-	0.3	No	0:25:46
Model_7	0.9128	0.2761	No	No	(3,3)	-	0.5	Yes	0:17:04
Model_8	1	0.6006	Yes	Yes	(5,5)	(2,2)	0.3	Yes	0:31:49

*Figure 19*  
*Model\_9 Accuracy Plot*

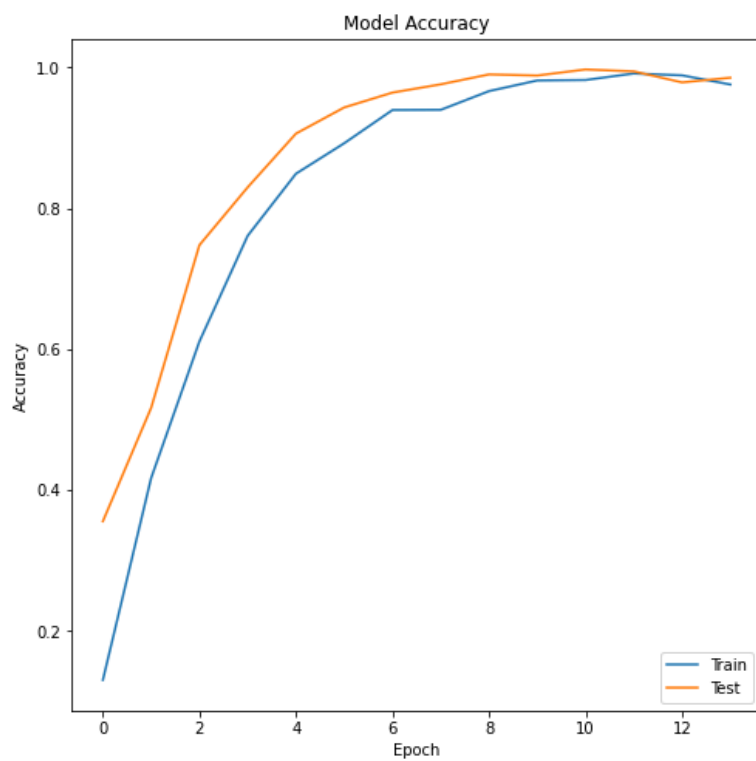
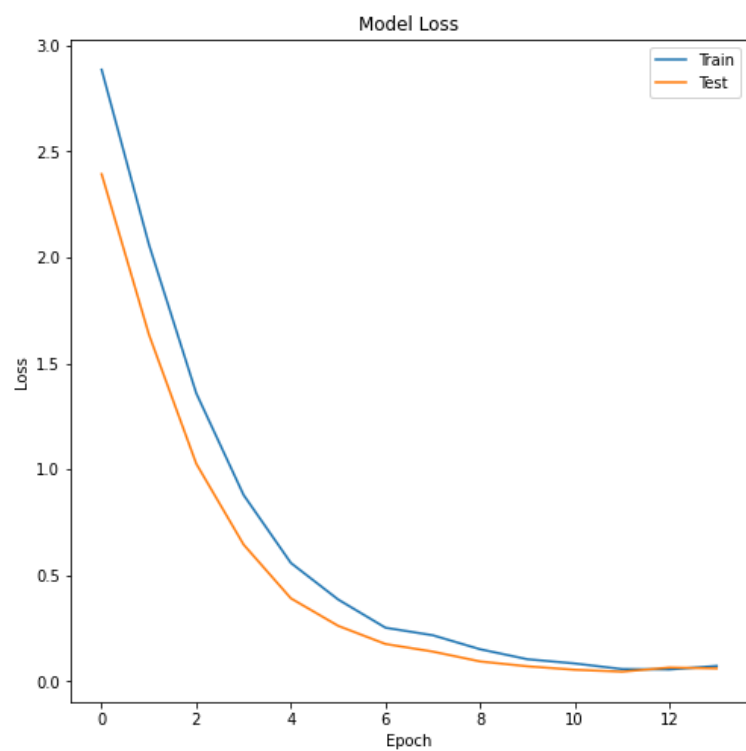


Figure 20  
Model\_9 Loss Plot



*Figure 21*  
*Model\_10 Accuracy Plot*



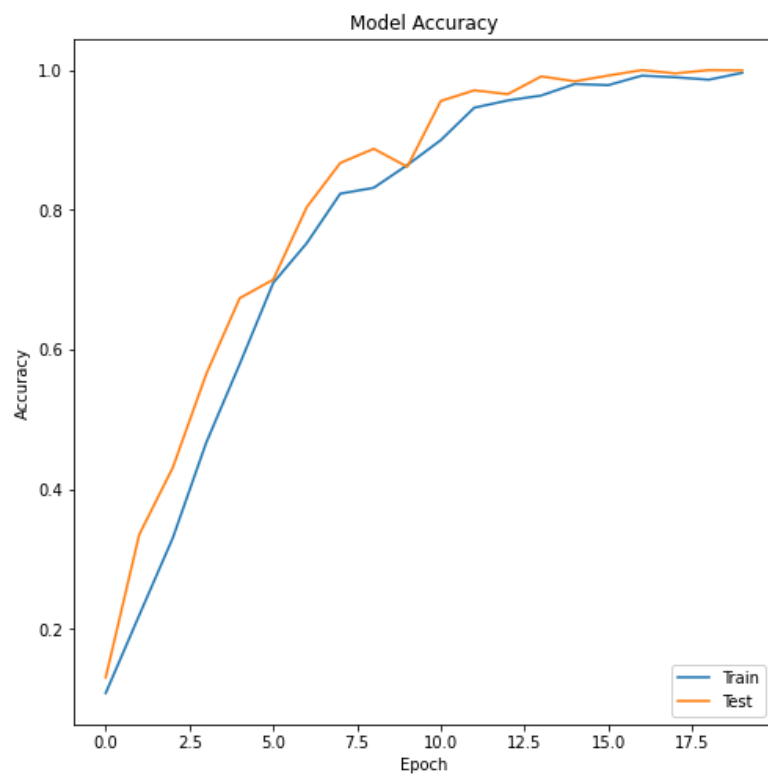
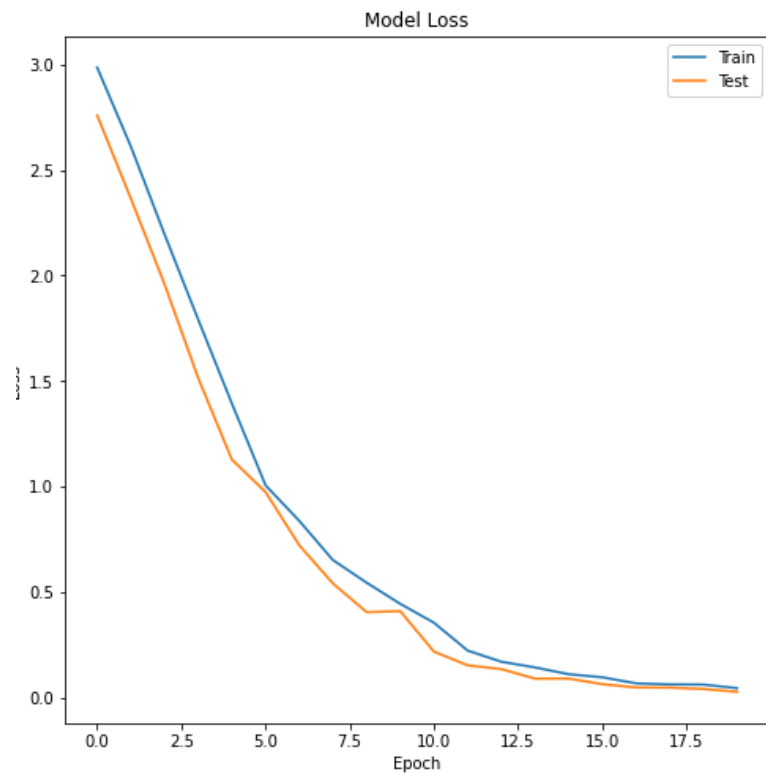


Figure 22  
Model\_10 Loss Plot



*Figure 23*  
*Model\_11 Accuracy Plot*

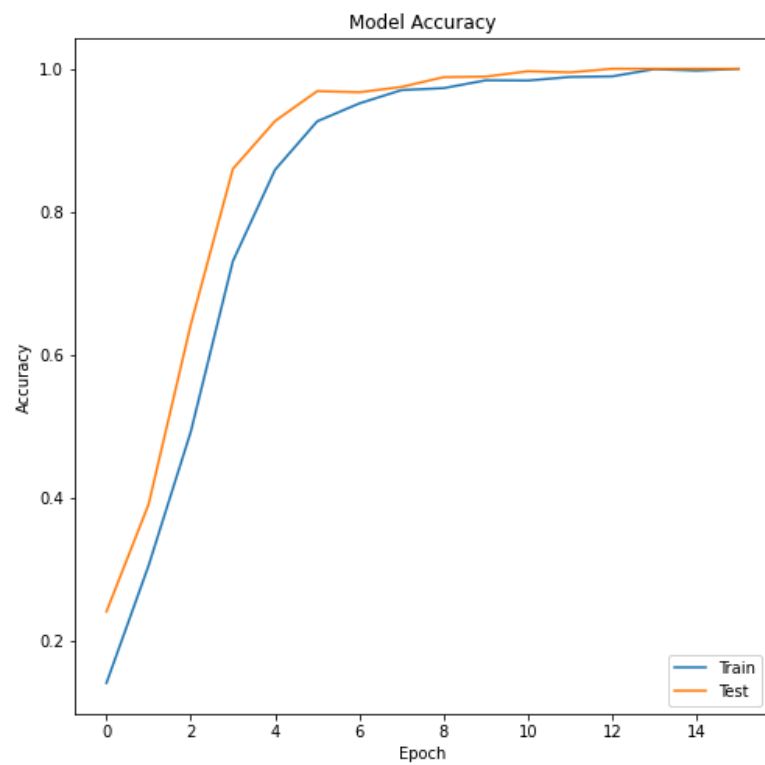


Figure 24  
Model\_11 Loss Plot

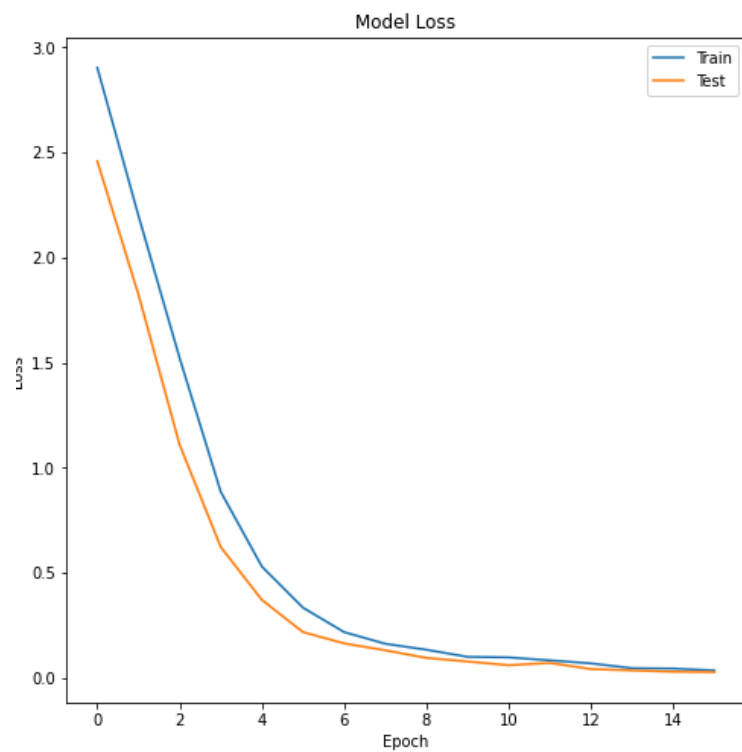


Figure 25  
Model\_12 Accuracy Plot

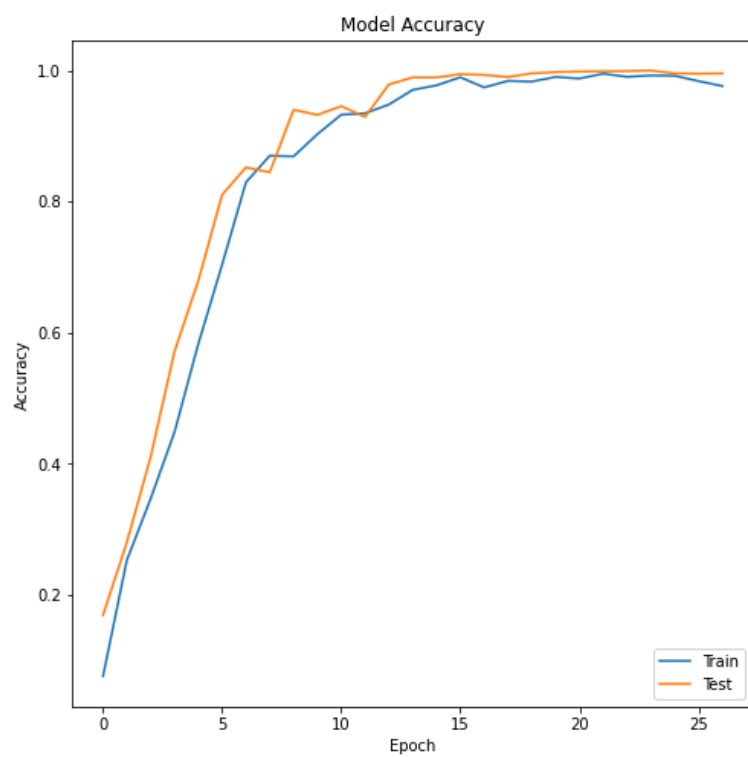


Figure 26  
Model\_12 Loss Plot

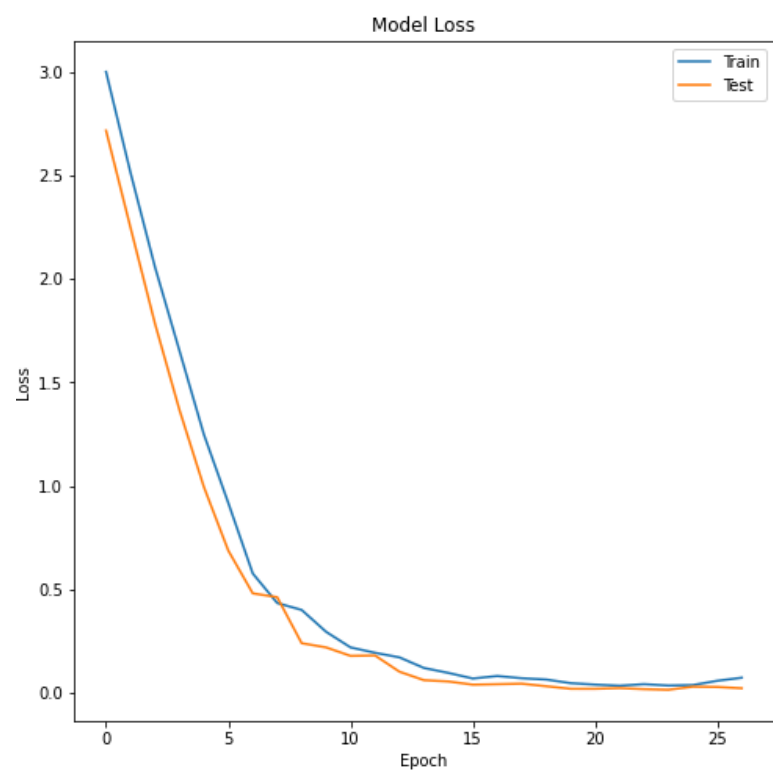


Figure 27  
Model\_13 Accuracy Plot

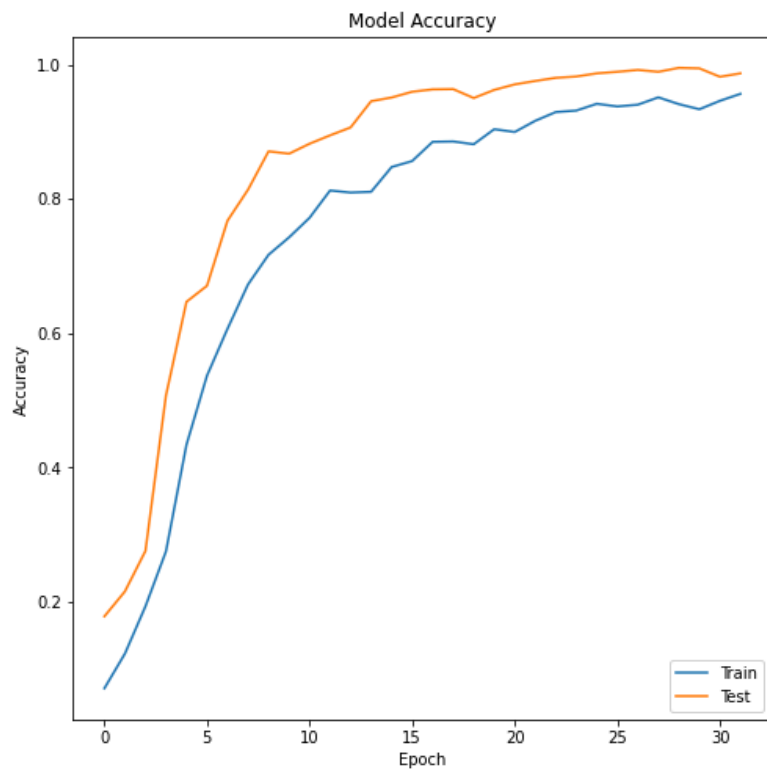


Figure 28  
Model\_13 Loss Plot

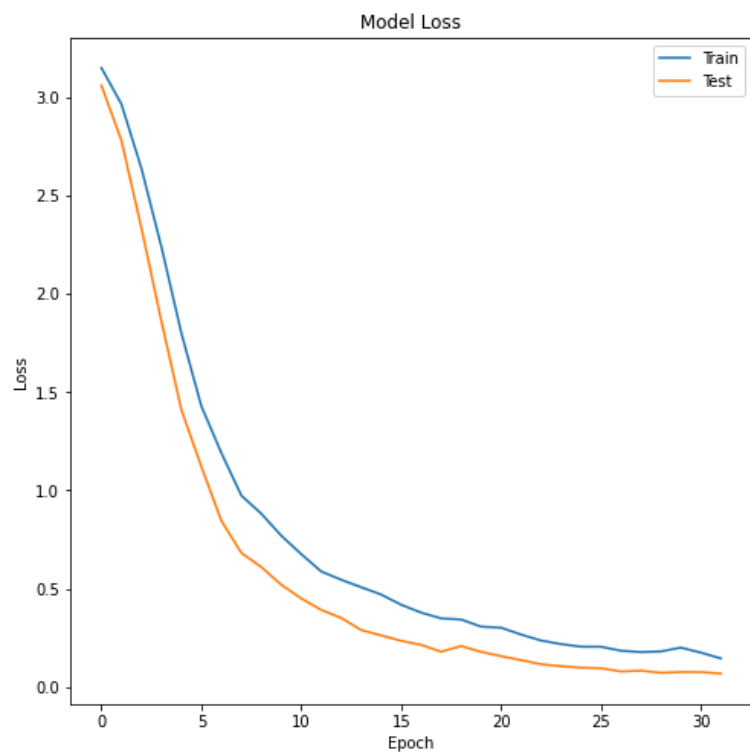


Figure 29  
Model\_14 Accuracy Plot



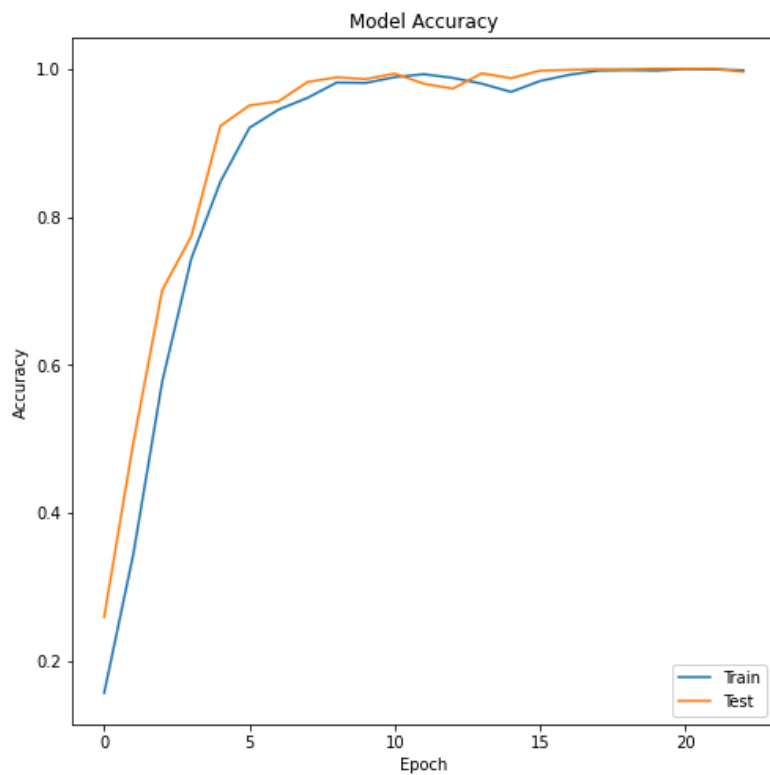


Figure 30  
Model\_14 Loss Plot

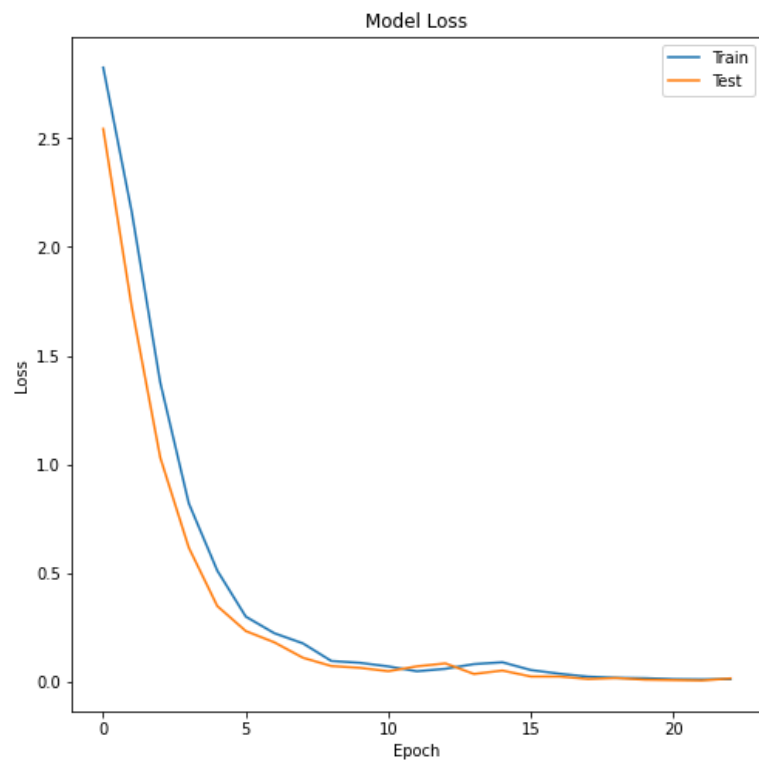


Figure 31  
Model\_15 Accuracy Plot

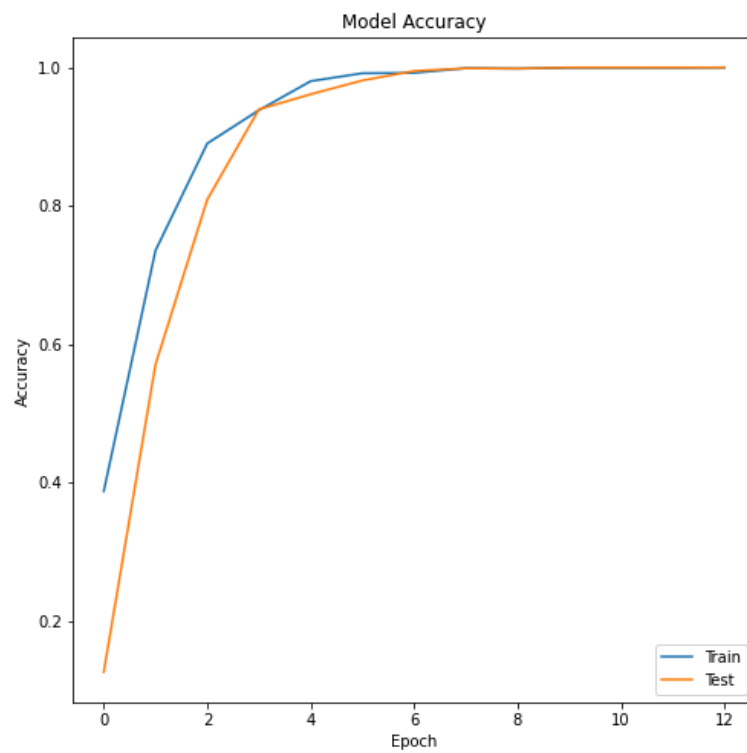
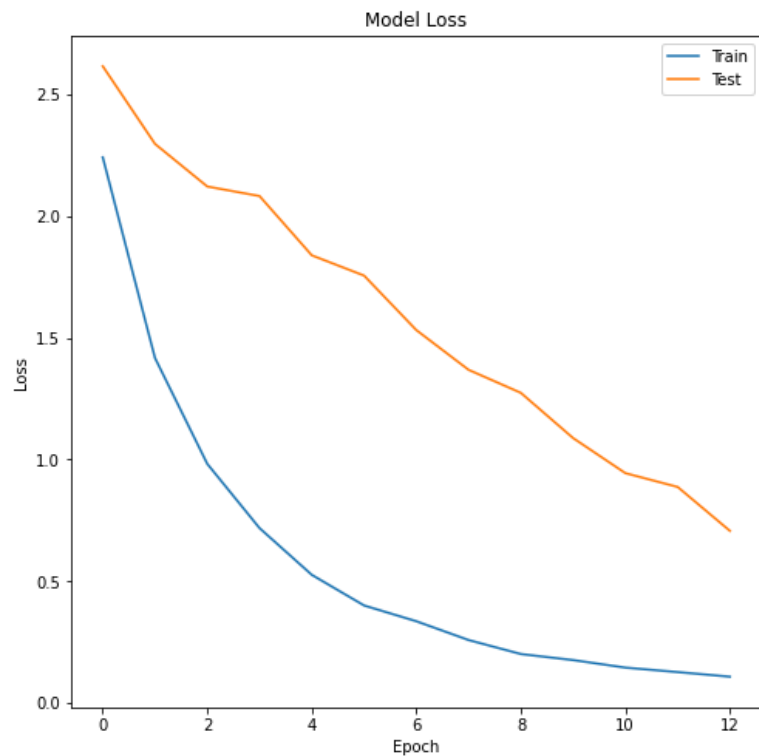


Figure 32  
Model\_15 Loss Plot



*Figure 33*  
*Model\_16 Accuracy Plot*

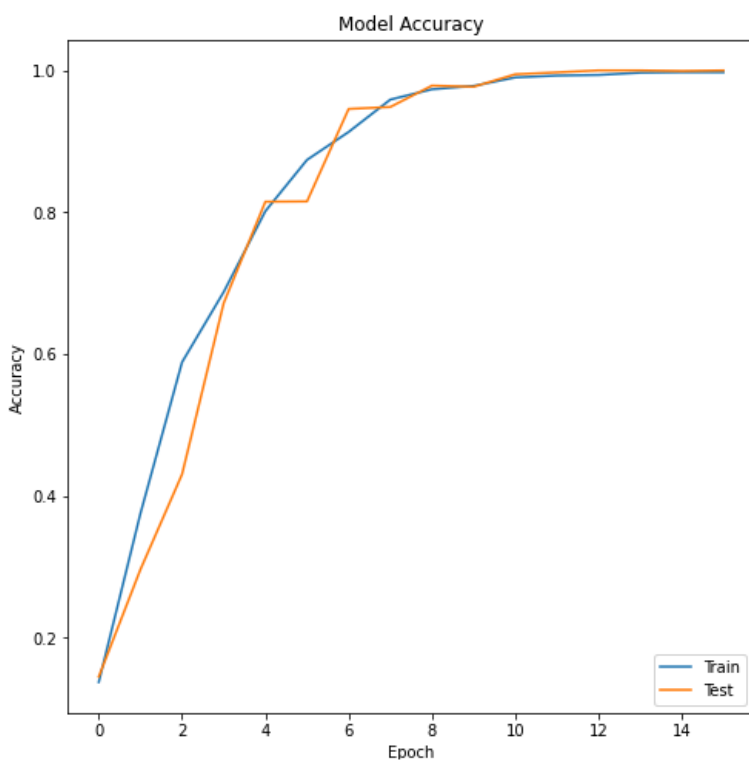
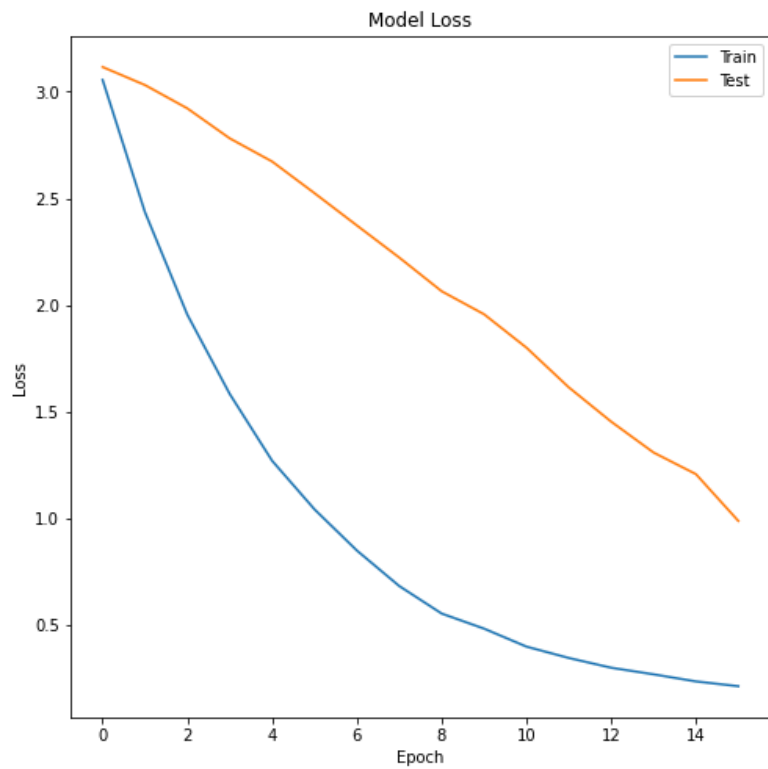


Figure 34  
Model\_16 Loss Plot



*Figure 35*  
*2 convolutional layer Models' Scores*

Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_9	0.9858	0.06069	No	No	(3,3)	-	0.5	No	0:08:47
Model_10	1	0.0282	Yes	No	(3,3)	-	0.5	No	0:13:49
Model_11	1	0.0282	No	Yes	(3,3)	-	0.5	No	0:11:03
Model_12	0.9952	0.0214	No	No	(5,5)	(1,1)	0.5	No	0:39:51
Model_13	0.9877	0.0673	No	No	(3,3)	(2,2)	0.5	No	0:20:24
Model_14	0.9961	0.0163	No	No	(3,3)	-	0.3	No	0:26:48
Model_15	1	0.7057	No	No	(3,3)	-	0.5	Yes	0:34:09
Model_16	1	0.9889	Yes	Yes	(5,5)	(2,2)	0.3	Yes	0:08:36

Figure 36  
Model\_17 Accuracy Plot

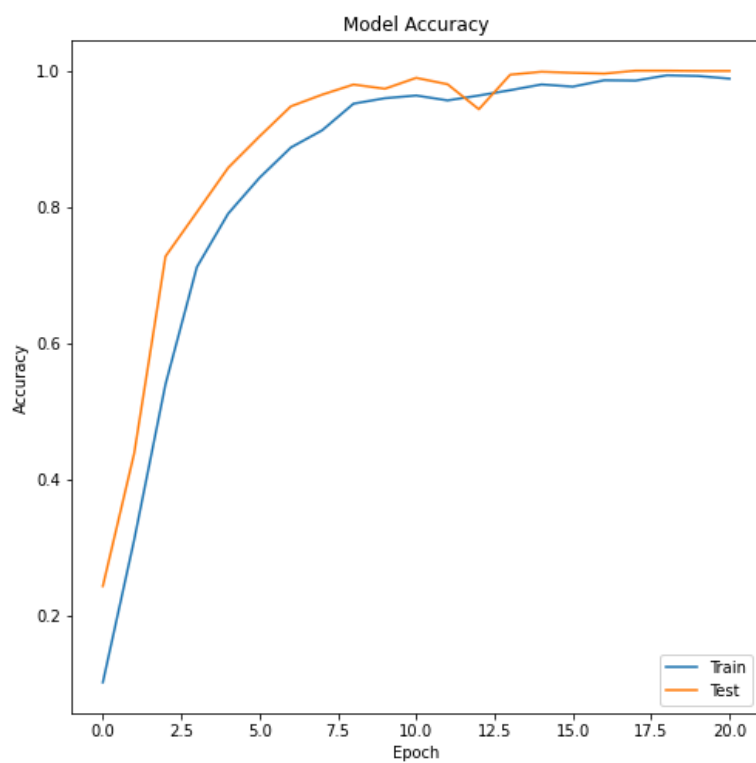


Figure 37  
Model\_17 Loss Plot



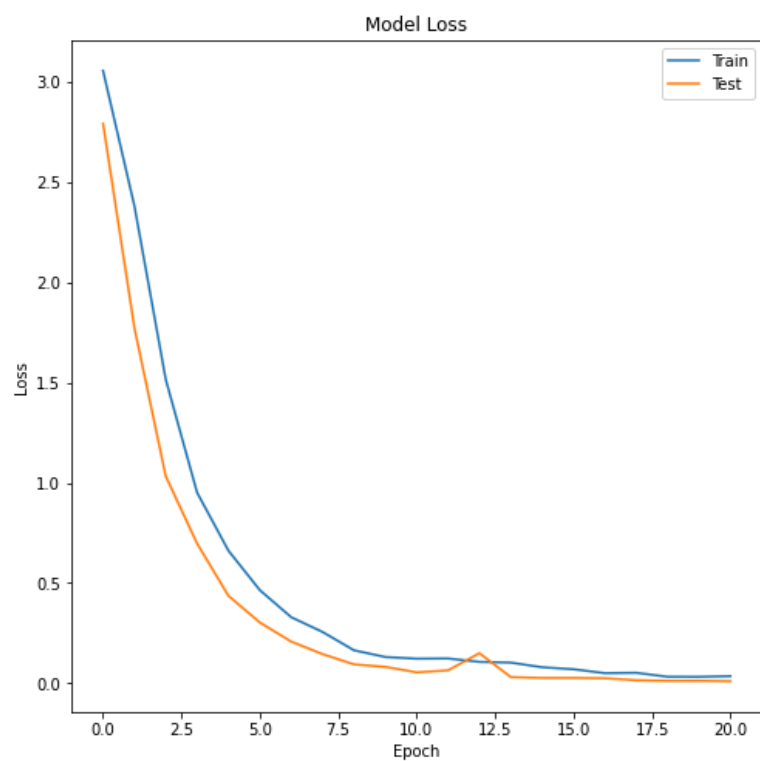


Figure 38  
Model\_18 Accuracy Plot

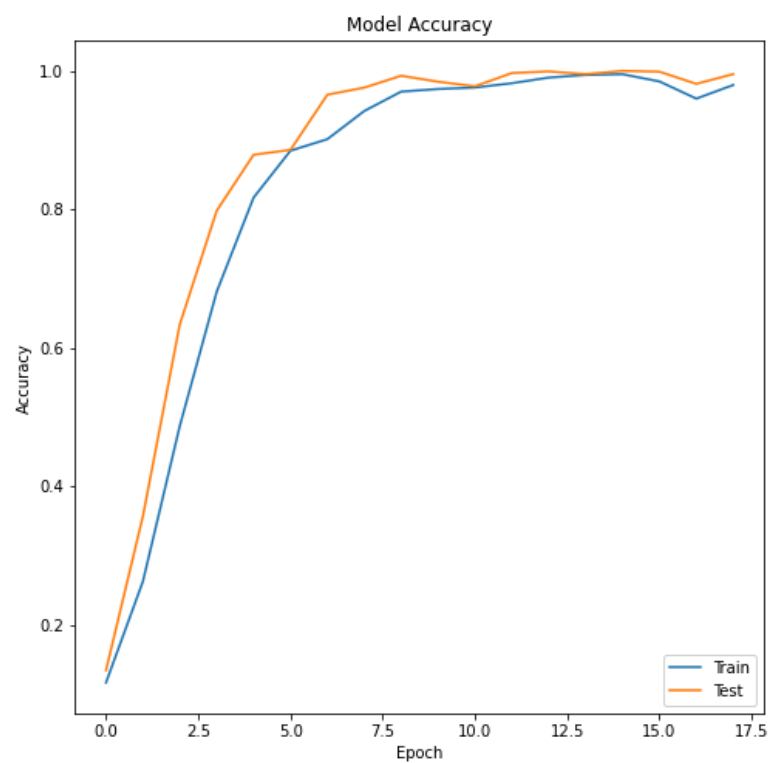


Figure 39  
Model\_18 Loss Plot

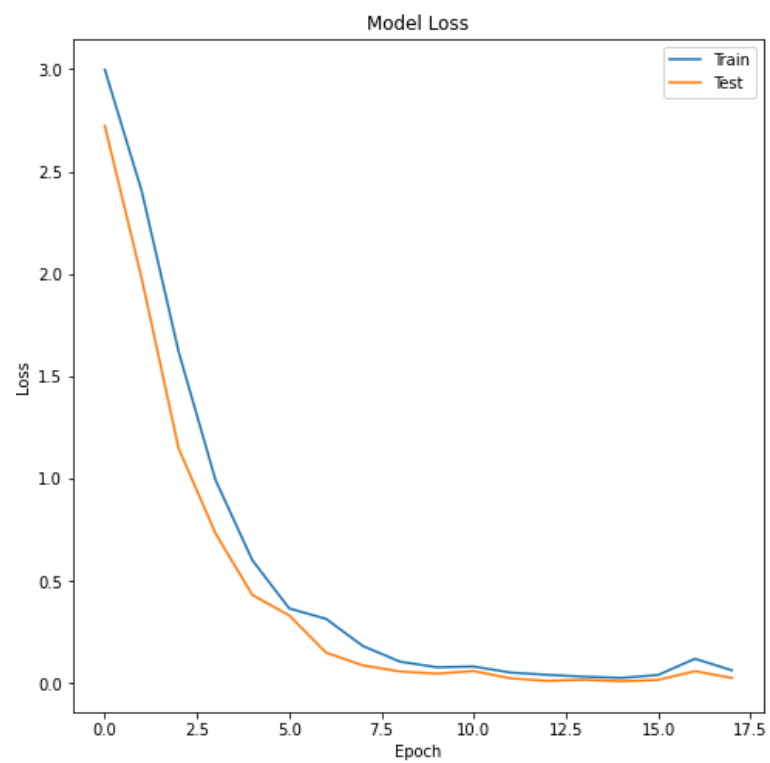


Figure 40  
Model\_19 Accuracy Plot

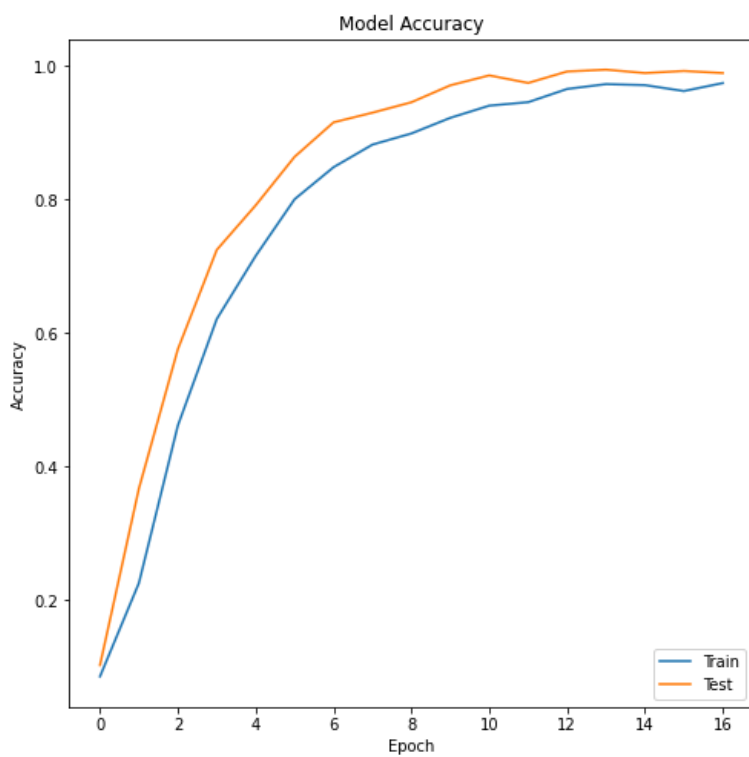


Figure 41  
Model\_19 Loss Plot

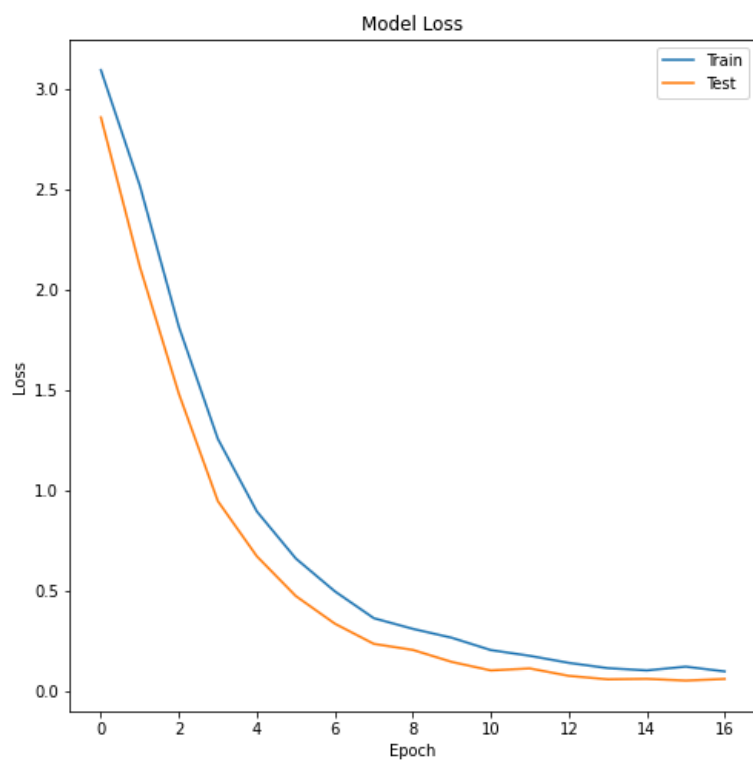


Figure 42  
Model\_20 Accuracy Plot

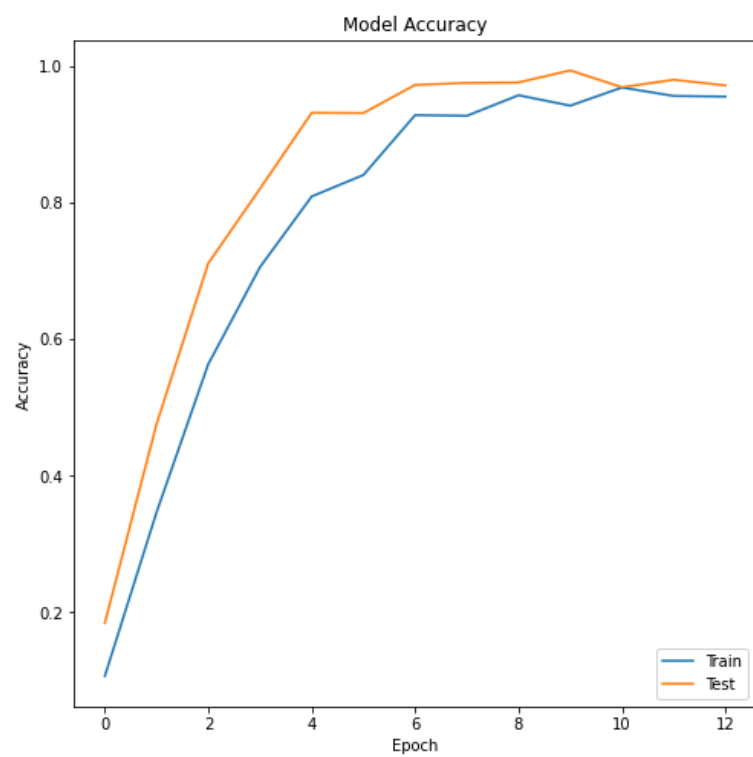


Figure 43  
Model\_20 Loss Plot

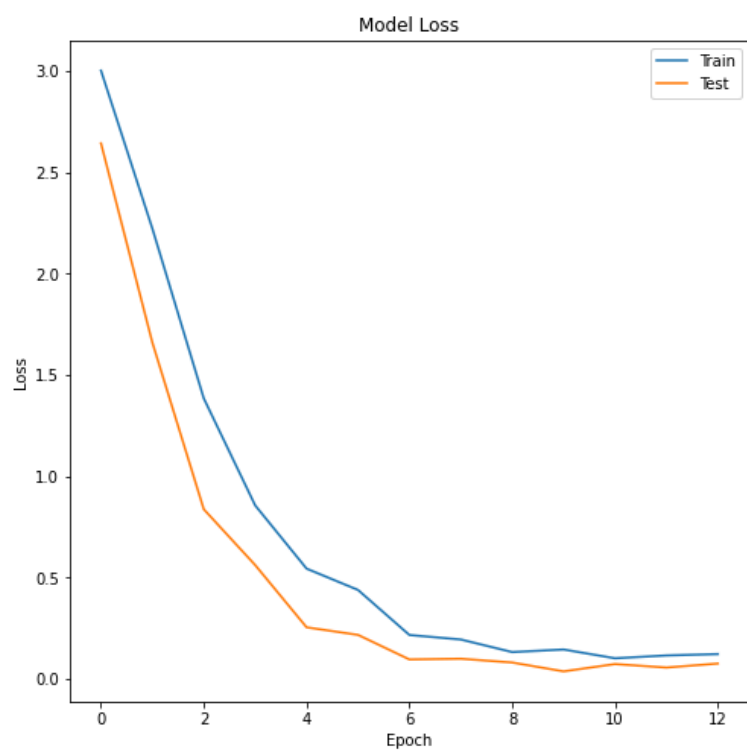


Figure 44  
Model\_21 Accuracy Plot

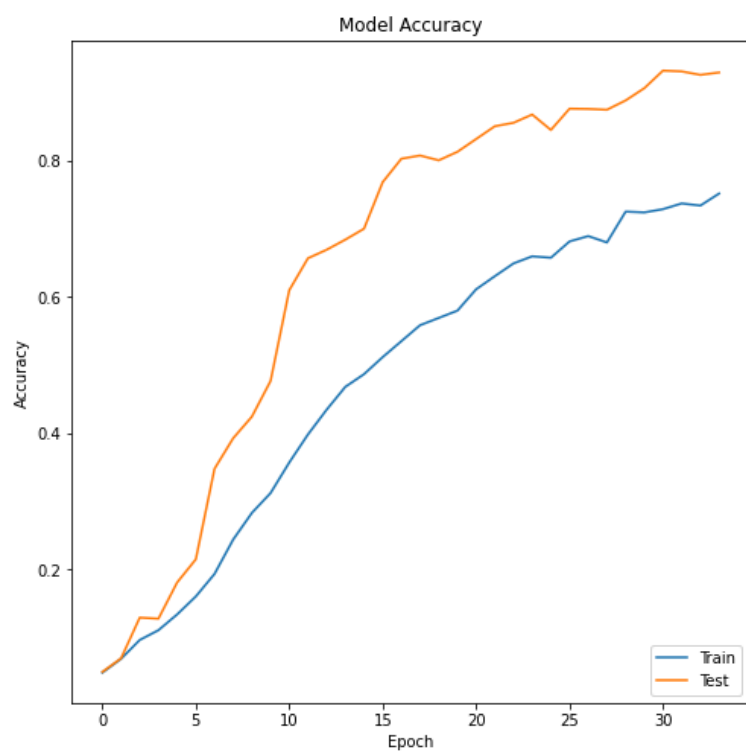


Figure 45  
Model\_21 Loss Plot



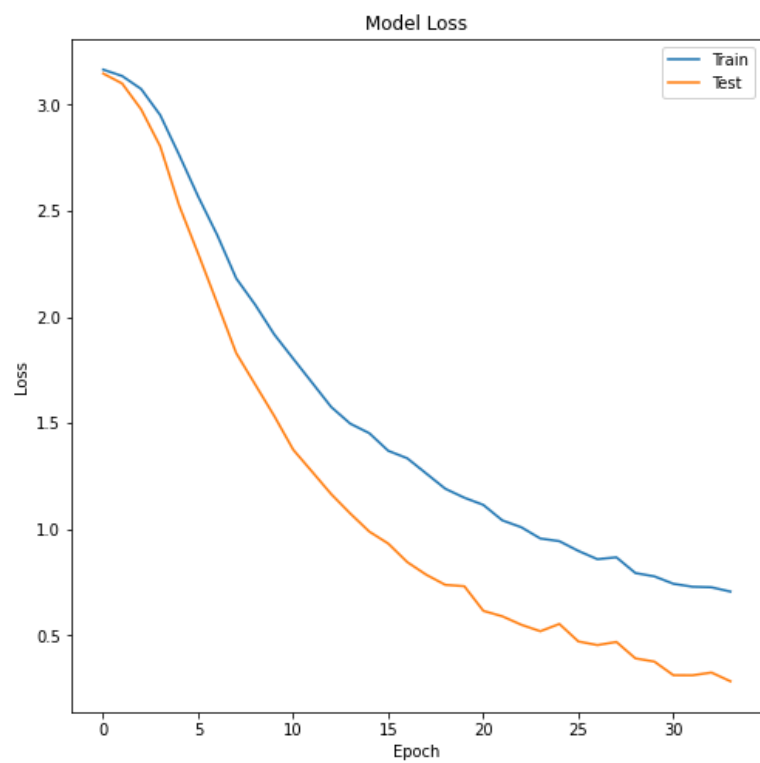


Figure 46  
Model\_22 Accuracy Plot

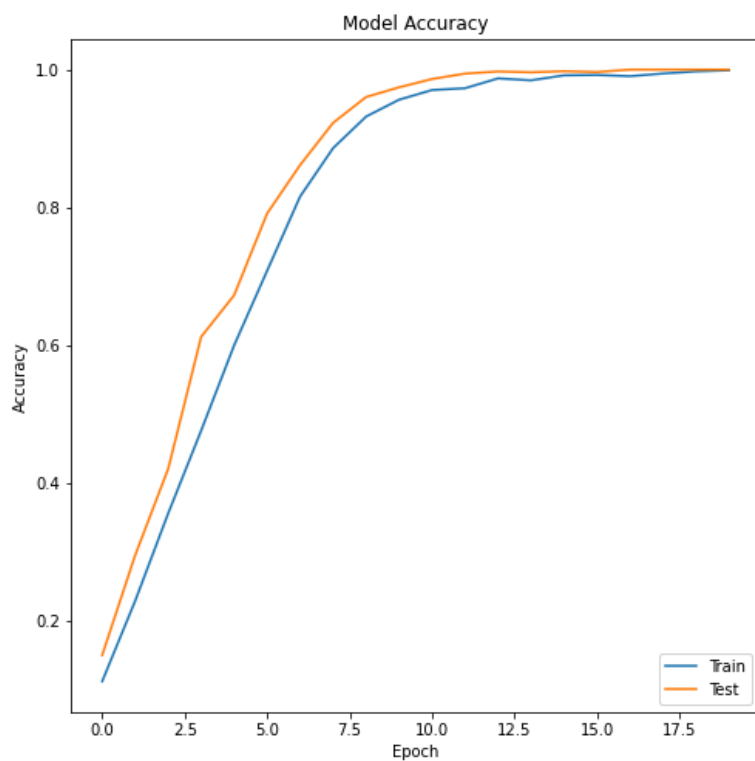


Figure 47  
Model\_22 Loss Plot

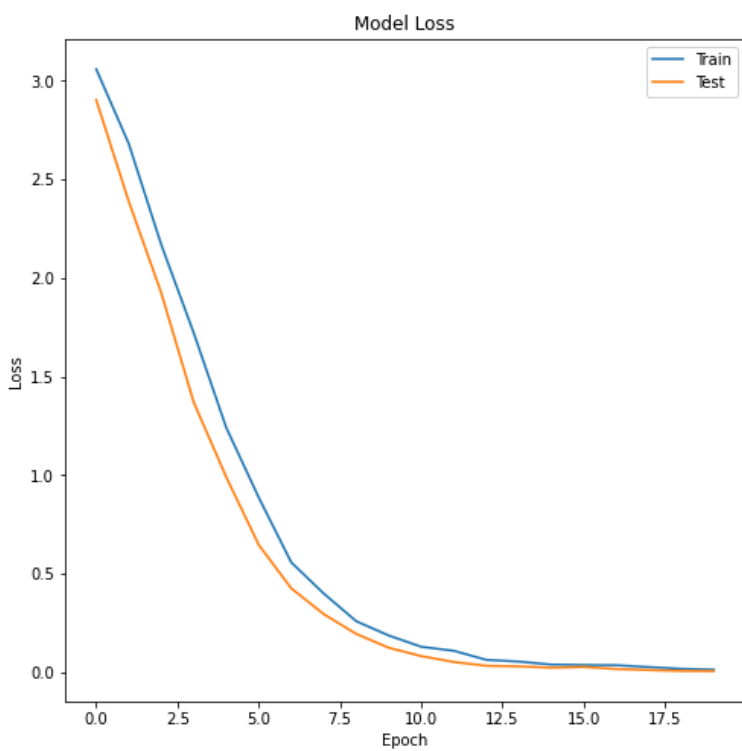


Figure 48  
Model\_23 Accuracy Plot

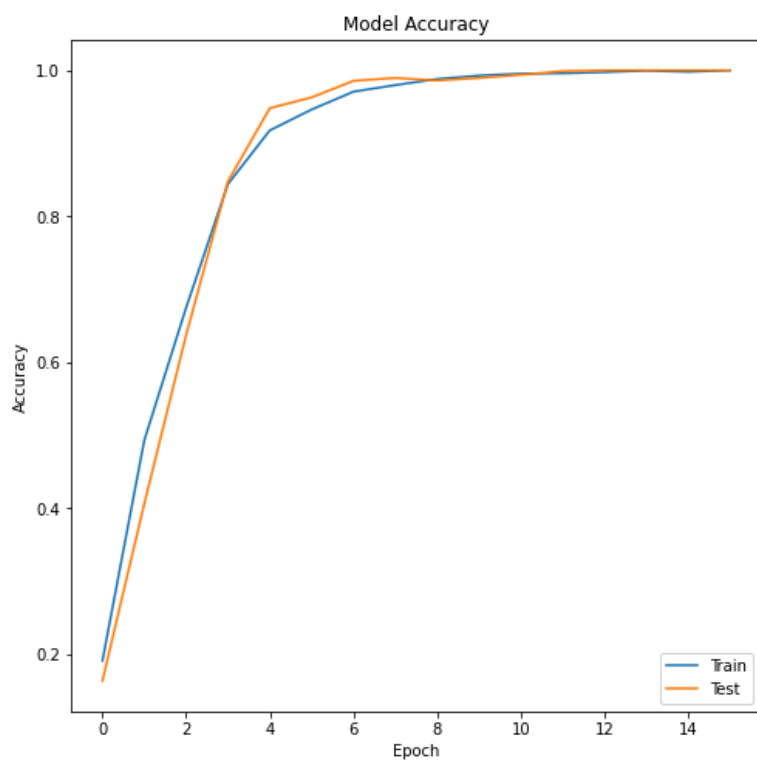


Figure 49  
Model\_23 Loss Plot

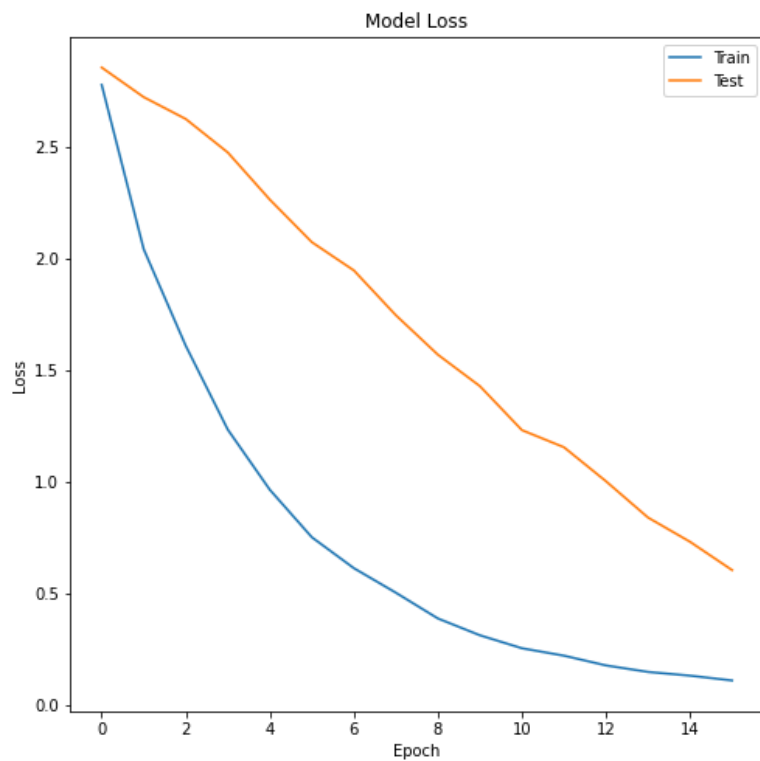


Figure 50  
Model\_24 Accuracy Plot

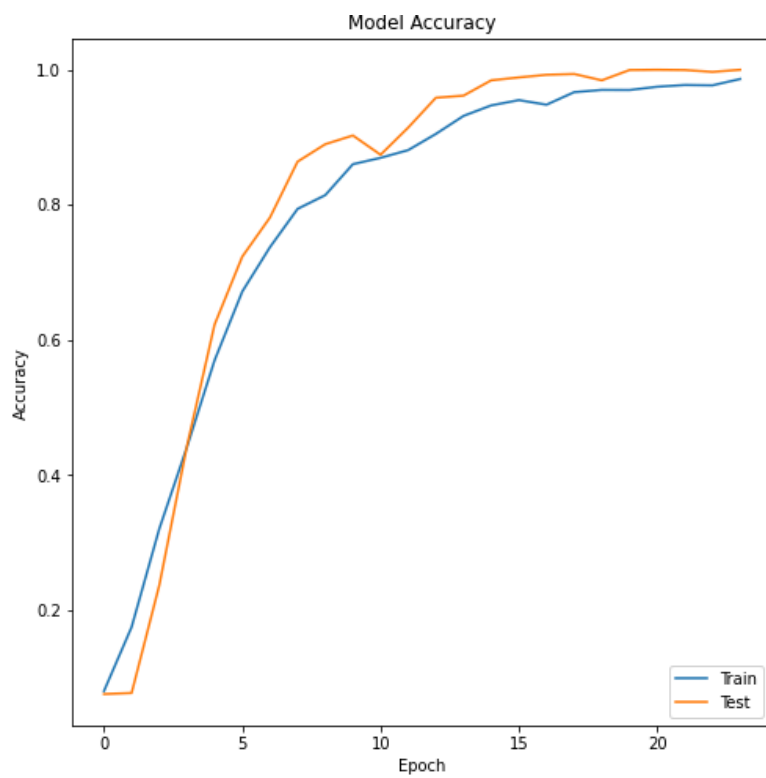
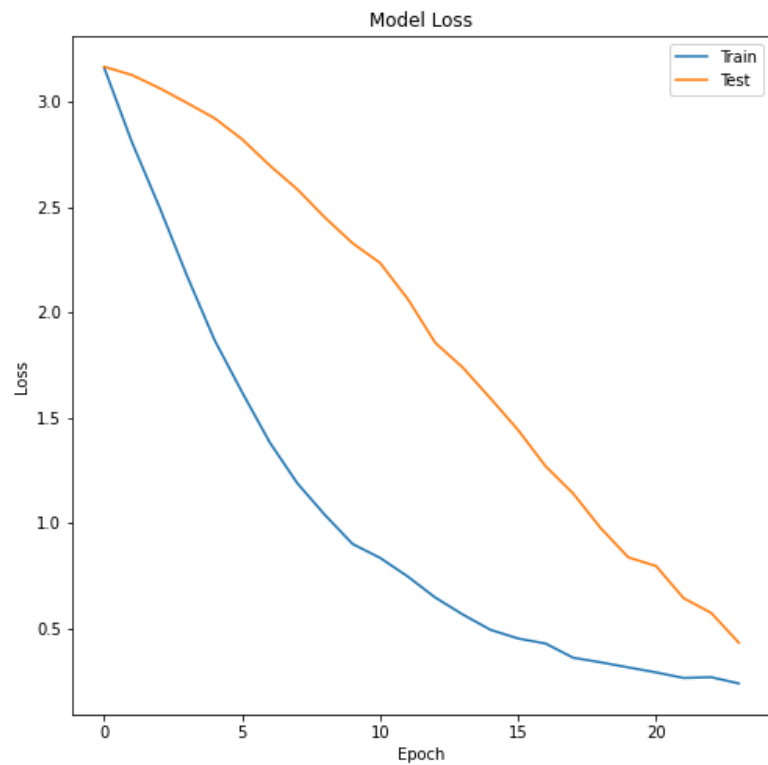


Figure 51  
Model\_24 Loss Plot



*Figure 52*  
*3 convolutional layer Models' Scores*

Model Name	Accuracy	Loss	Padding	Regularization	Kernel Size	Stride Size	Dropout Rate	Batch Normalization	Process Time
Model_17	0.9996	0.0097	No	No	(3,3)	-	0.5	No	0:11:05
Model_18	0.9954	0.0257	Yes	No	(3,3)	-	0.5	No	0:10:19
Model_19	0.9884	0.0625	No	Yes	(3,3)	-	0.5	No	0:09:13
Model_20	0.9733	0.0762	No	No	(5,5)	(1,1)	0.5	No	0:07:02
Model_21	0.9321	0.2808	No	No	(3,3)	(2,2)	0.5	No	0:18:23
Model_22	1	0.0056	No	No	(3,3)	-	0.3	No	0:56:43
Model_23	1	0.6057	No	No	(3,3)	-	0.5	Yes	0:08:17
Model_24	0.9996	0.4285	Yes	Yes	(5,5)	(2,2)	0.3	Yes	1:01:31

*Figure 53*  
*All Models' Processing Times*



