CS F425 – DEEP LEARNING

Assignment 2

Sathvik Bhaskarpandit[1], Ruban S[2], and Deepti Kumar[3]

[1]2019A7PS1200H
[2]2019A7PS0097H
[3]2018B5A70790H

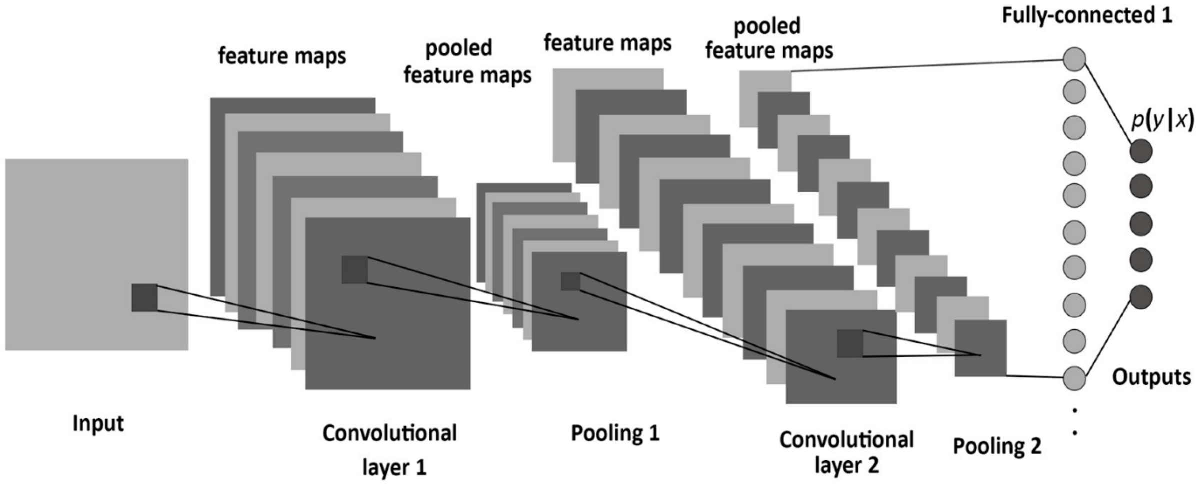
Table of Contents

Convolutional Neural Network Models

We have implemented a base convolutional neural network model for this assignment in Keras. We have also implemented 21 other models to study the effect of change in the various hyperparameters such as number of convolution blocks, number of filters, filter size, stride, activation function, dropout, pool size and pool type.

We used Glorot uniform initialization, Cross-Entropy Loss, and Adam optimizer for training our models for all the below-mentioned networks. Batch Normalization has been used after every convolution. Early stopping is used to reduce training time.

Dataset: The Fashion MNIST dataset provided consists of 60,000 28×28 images of clothing items. To feed the images into our neural network, we flatten each image into a 784-dimensional vector.



Typical architecture of a Convolutional Neural Network

The initial network (base model) has the following hyperparameters

Block 1

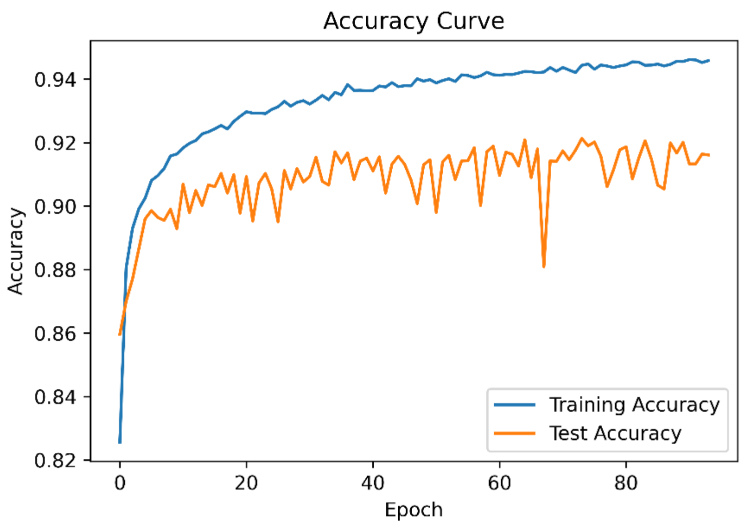
- Number of convolution filters = 64
- Filter size = 3×3
- Stride = 1×1
- Activation function: ReLU
- Dropout = 0.2
- Pool size = 2
- Pool type: `max`

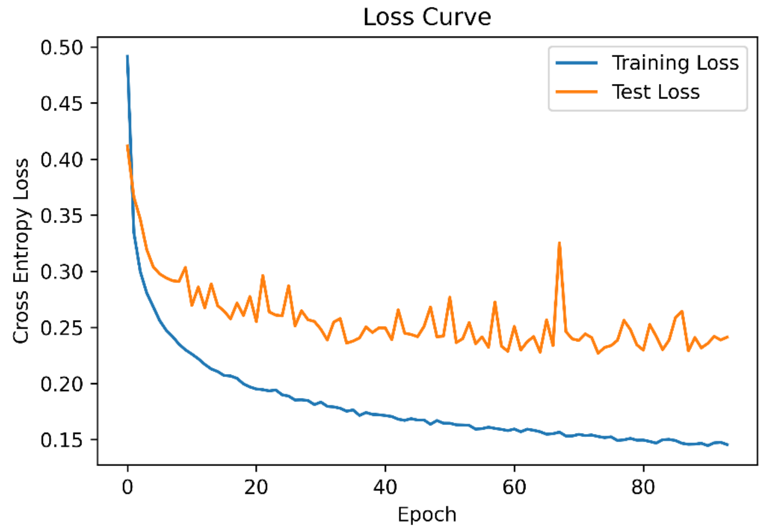
Block 2

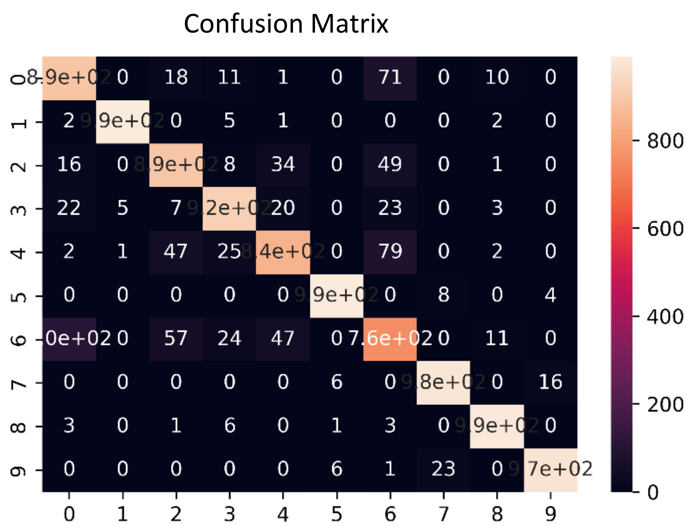
- Number of convolution filters = 32
- Filter size = 2×2
- Stride = 1×1
- Activation function: ReLU
- Dropout = 0.2
- Pool size = 2
- Pool type: `max`

To do a comparative study with various models, we have kept one block the same as the base model, and varied one hyperparameter per model in the other block.

Statistics for the base model







Classification Report

	Precision	Recall	F1-score	Support
0	0.86	0.89	0.87	1000
1	0.99	0.99	0.99	1000
2	0.87	0.89	0.88	1000
3	0.92	0.92	0.92	1000
4	0.89	0.84	0.87	1000
5	0.99	0.99	0.99	1000
6	0.77	0.76	0.76	1000
7	0.97	0.98	0.97	1000
8	0.97	0.99	0.98	1000
9	0.98	0.97	0.97	1000
accuracy	0.92	0.92	0.92	0.92
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

Accuracy and Log Loss of the tested CNN Models

Hyperparameters for a layer are in the order: `(number of filters, filter size, stride length, activation function, dropout, pool size, pool type)`

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	(64, (3,3), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.961	0.921	0.134	0.227
2	(32, (3,3), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.953	0.922	0.155	0.225
3	(128, (3,3), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.96	0.923	0.129	0.226
4	(256, (3,3), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.956	0.922	0.14	0.232
5	(64, (2,2), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.95	0.915	0.155	0.242
6	(64, (5,5), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.955	0.909	0.143	0.252
7	(64, (7,7), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.947	0.915	0.165	0.241
8	(64, (3,3), (1,1), 'sigmoid', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.878	0.865	0.326	0.367
9	(64, (3,3), (1,1), 'tanh', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.945	0.918	0.165	0.232
10	(64, (3,3), (2,2), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.906	0.88	0.278	0.342
11	(64, (3,3), (5,5), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.856	0.837	0.418	0.461
12	(64, (3,3), (1,1), 'relu', 0, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.956	0.917	0.132	0.239
13	(64, (3,3), (1,1), 'relu', 0.5, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.926	0.904	0.25	0.295
14	(64, (3,3), (1,1), 'relu', 0.1, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.956	0.917	0.132	0.235
15	(64, (3,3), (1,1), 'relu', 0.8, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.873	0.861	0.395	0.422
16	(64, (3,3), (1,1), 'relu', 0.2, 1, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.958	0.916	0.132	0.241
17	(64, (3,3), (1,1), 'relu', 0.2, 5, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.926	0.895	0.218	0.287
18	(64, (3,3), (1,1), 'relu', 0.2, 2, 'average') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.96	0.918	0.124	0.233
19	(64, (3,3), (1,1), 'relu', 0.2, 2, 'min') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.964	0.919	0.11	0.231
20	(64, (3,3), (1,1), 'relu', 0.2, 2, 'max') (16, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.943	0.918	0.177	0.239
21	(64, (3,3), (1,1), 'relu', 0.2, 2, 'max') (16, (2,2), (1,1), 'relu', 0.2, 2, 'average')	0.958	0.923	0.125	0.229
22	(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')	0.962	0.91	0.114	0.271

Observations and Explanations

Number of Epochs

The average number of epochs for convergence with early stopping with a patience of 20, was around 90 epochs.

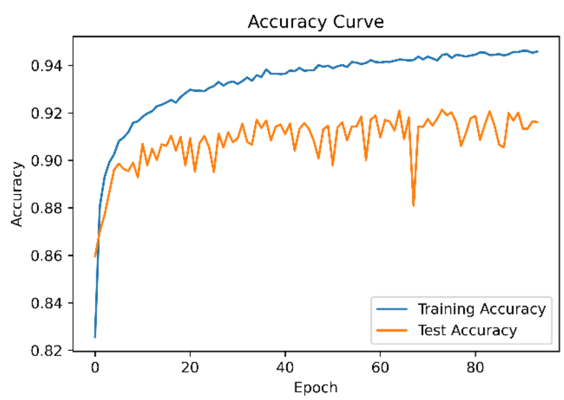
Effect of Number of Convolution Filters

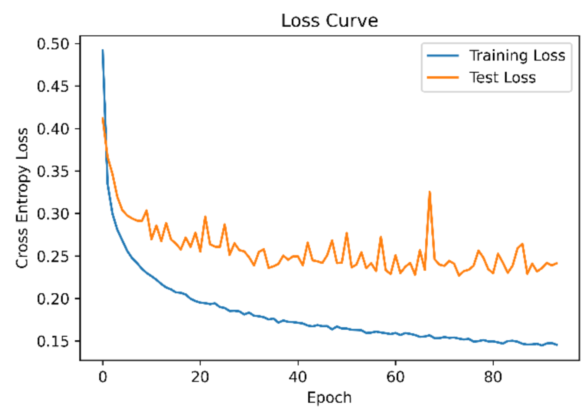
Each filter generates a feature map. Feature maps allow the network to learn the explanatory factors within the image, so the more the number filters mean the more the network learns. This may not necessarily be good all the time - saturation and convergence matter the most. In changing the number of filters in the first block, we observe a minor increase in test accuracy.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
2	`(32, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.953	0.922	0.155	0.225
3	`(128, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.96	0.923	0.129	0.226
4	`(256, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.956	0.922	0.14	0.232

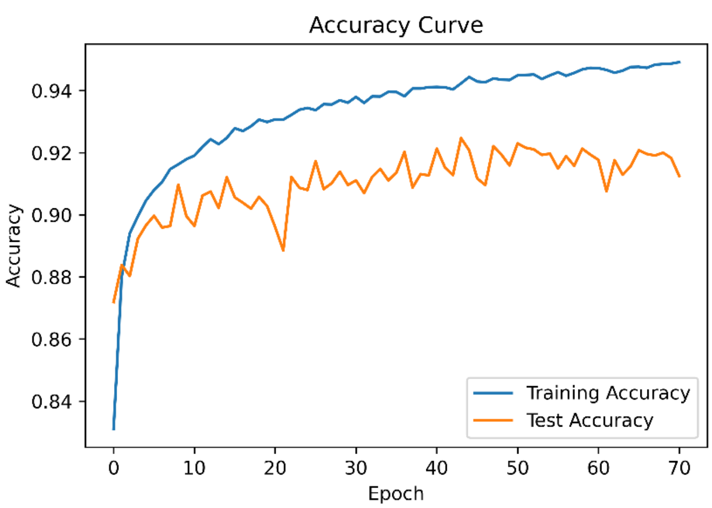
Plots

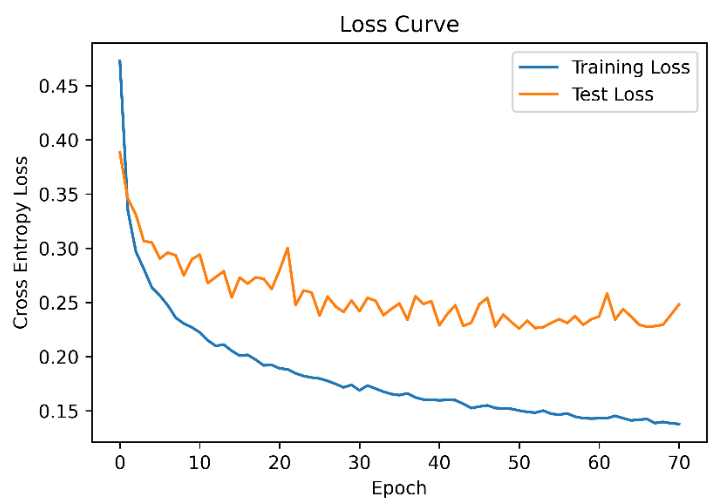
Model 1





Model 3





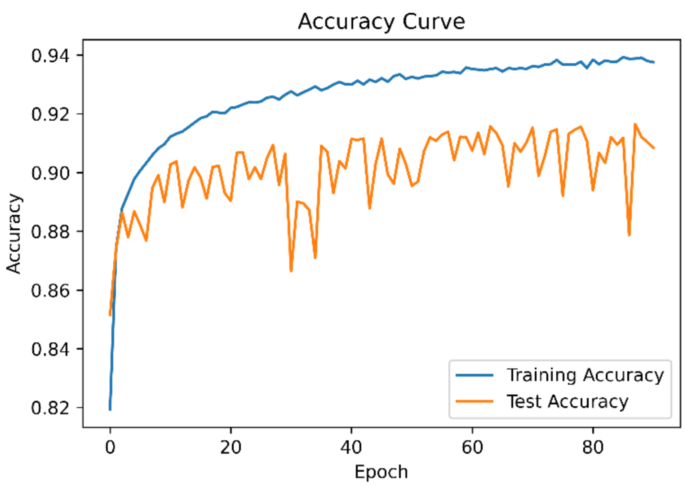
Effect of Filter Size

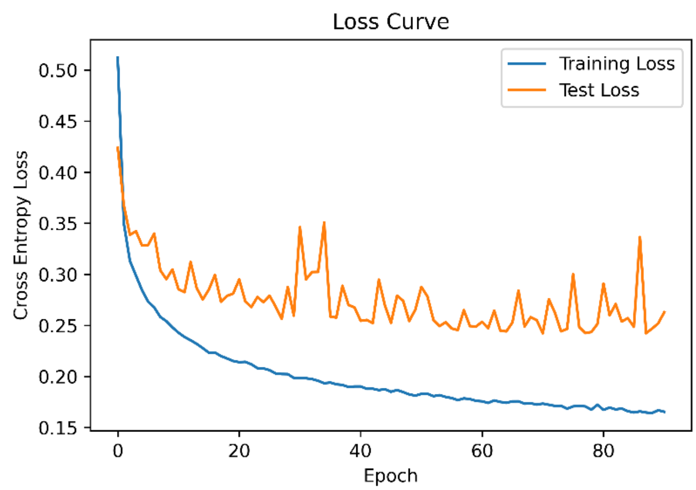
Filter size can be thought of as the size of the region from which we need to extract features or more informally, how much neighbor information we can see when processing a layer. Optimal filter size highly depends on the dataset and input images.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
5	`(64, (2,2), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.95	0.915	0.155	0.242
6	`(64, (5,5), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.955	0.909	0.143	0.252
7	`(64, (7,7), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.947	0.915	0.165	0.241

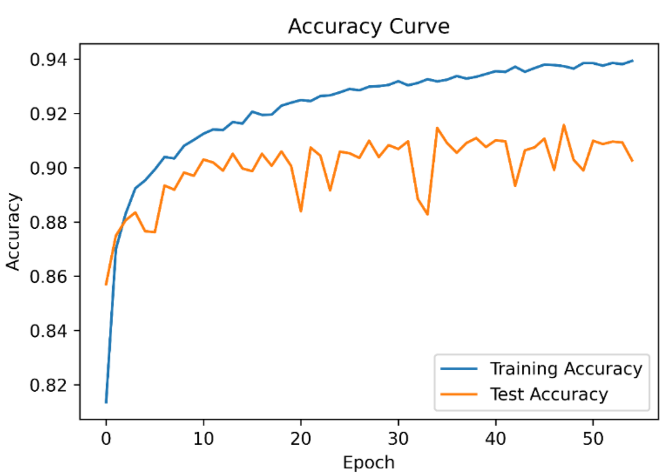
Plots

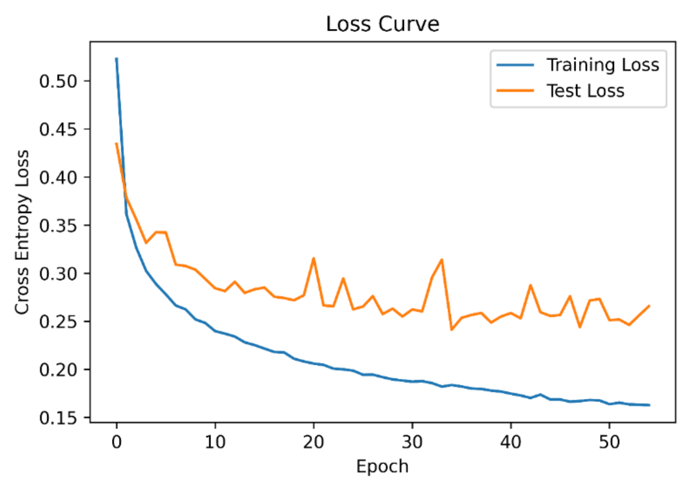
Model 5





Model 7





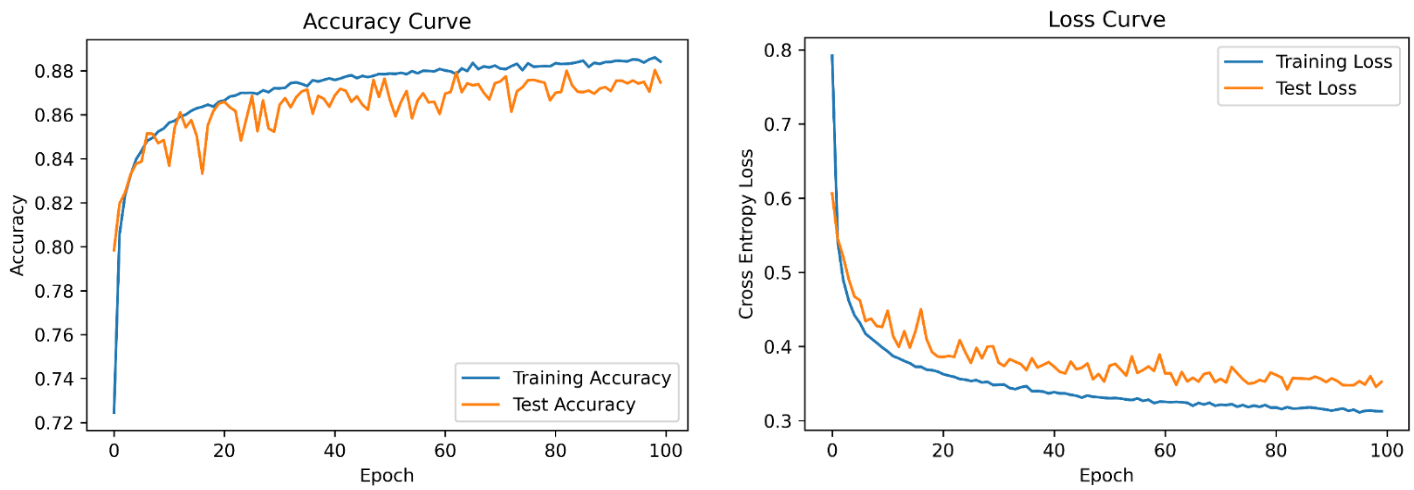
Effect of Stride

Stride is the distance between two convolutional window start points. A stride of 2 implies that the filter should slide by 2 pixels then generate a feature map. Smaller strides capture features at a smaller scale, however larger strides can be used when images are sparser.
For our dataset, a stride value of greater than one showed a major decrease in accuracy.

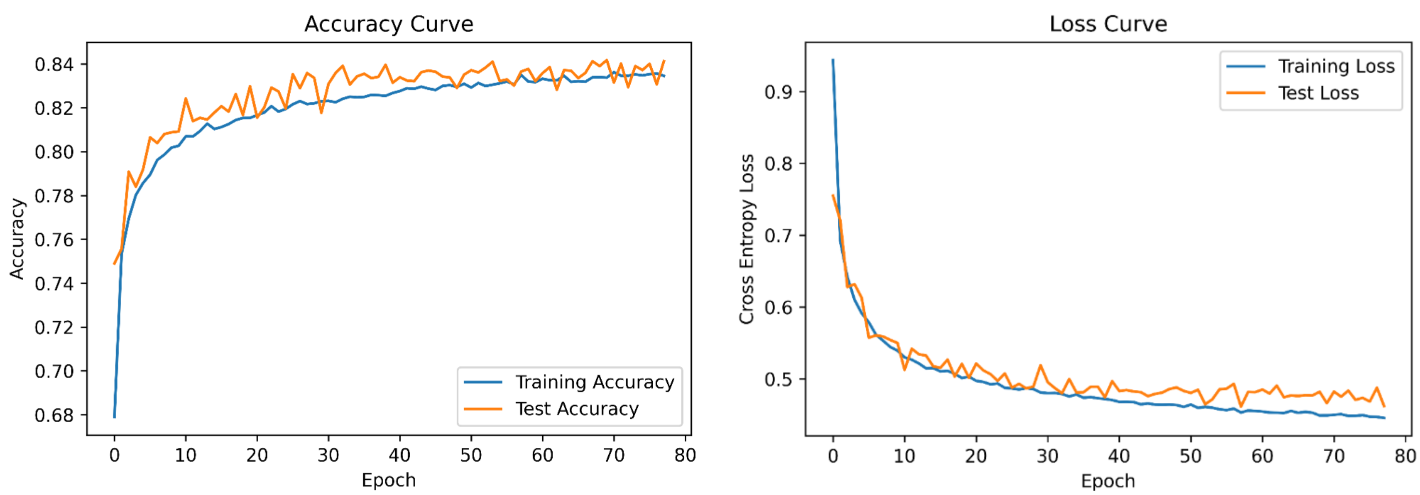
Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	(64, (3,3), (1,1), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.961	0.921	0.134	0.227
10	(64, (3,3), (2,2), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.906	0.88	0.278	0.342
11	(64, (3,3), (5,5), 'relu', 0.2, 2, 'max') (32, (2,2), (1,1), 'relu', 0.2, 2, 'max')	0.856	0.837	0.418	0.461

Plots

Model 10



Model 11



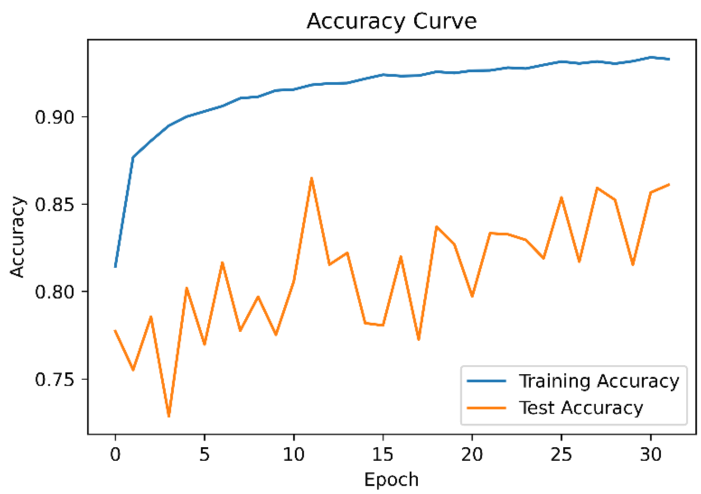
Effect of Activation Function

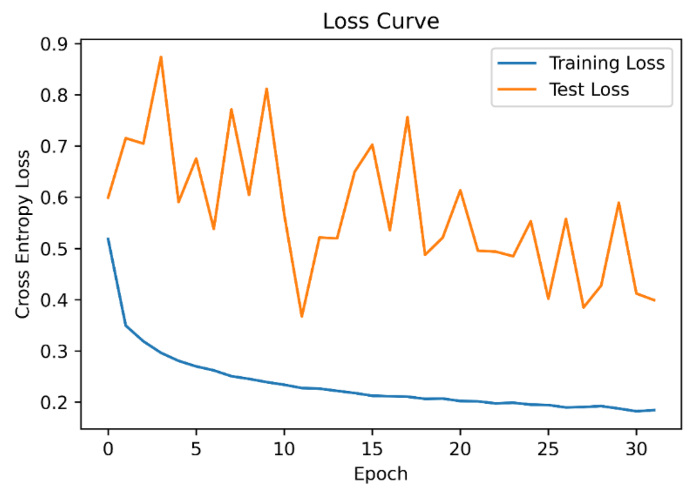
ReLU activation performs the best of all three, while tanh performs the worst of them all. This is because the ReLU function does not saturate for larger weights while both tanh and sigmoid do. Tanh saturates fastest among all three, which might be the reason for lower accuracy.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
8	`(64, (3,3), (1,1), 'sigmoid', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.878	0.865	0.326	0.367
9	`(64, (3,3), (1,1), 'tanh', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.945	0.918	0.165	0.232

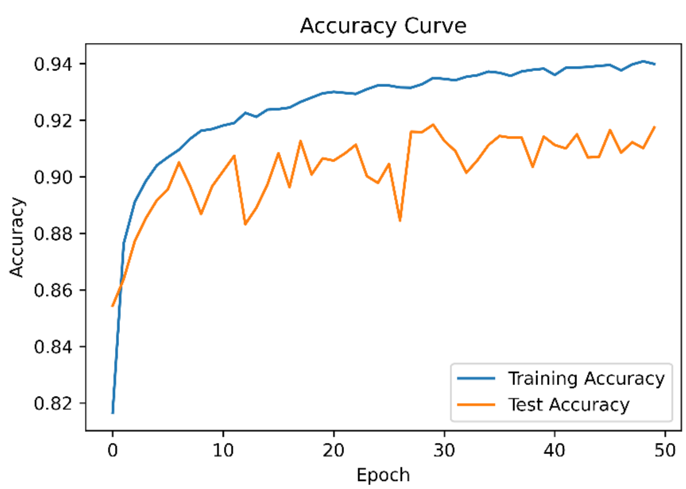
Plots

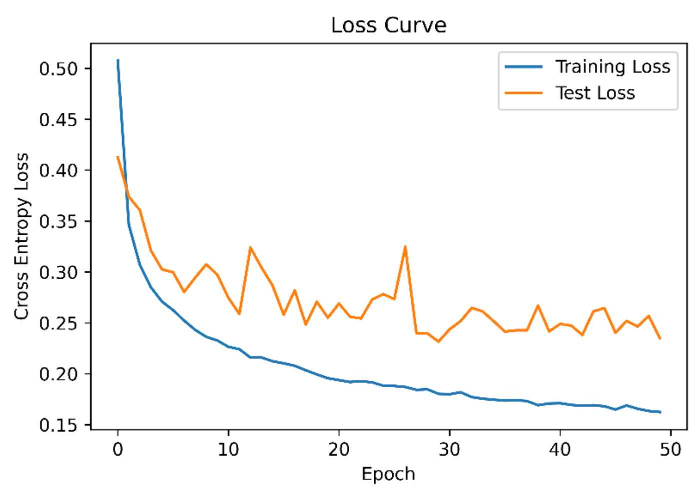
Model 8





Model 9





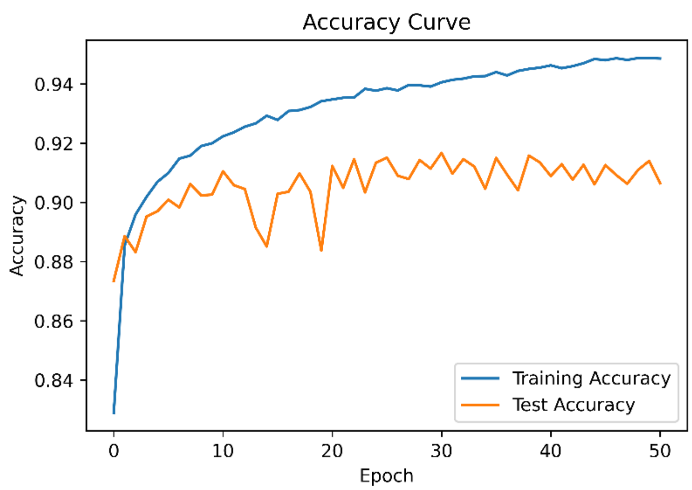
Effect of Dropout

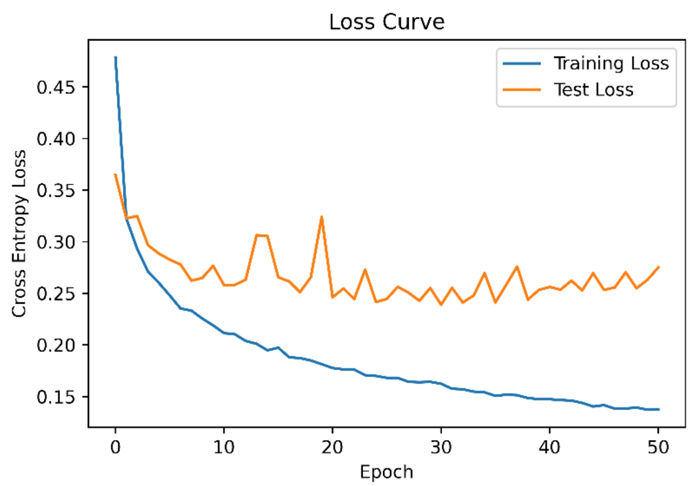
Dropout effectively nullifies a fraction of neurons while training. This decreases the chance of overfitting. However, if the dropout rate is too high, too many neurons are nullified and the model starts to underfit.
Dropout has a noticeable effect on the accuracy of the mode. A dropout rate of 0.2 seems to work well for our model.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
12	`(64, (3,3), (1,1), 'relu', 0, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.956	0.917	0.132	0.239
13	`(64, (3,3), (1,1), 'relu', 0.5, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.926	0.904	0.25	0.295
14	`(64, (3,3), (1,1), 'relu', 0.1, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.956	0.917	0.132	0.235
15	`(64, (3,3), (1,1), 'relu', 0.8, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.873	0.861	0.395	0.422

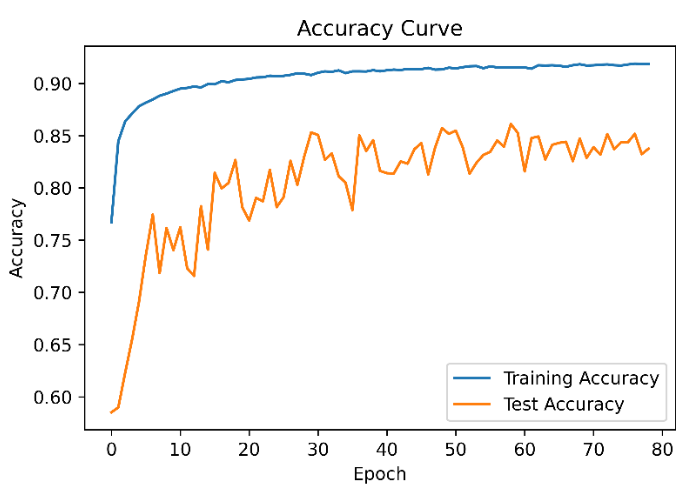
Plots

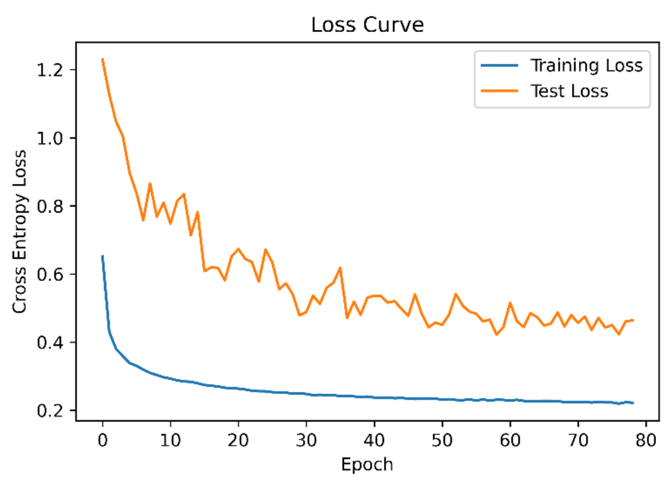
Model 12





Model 15





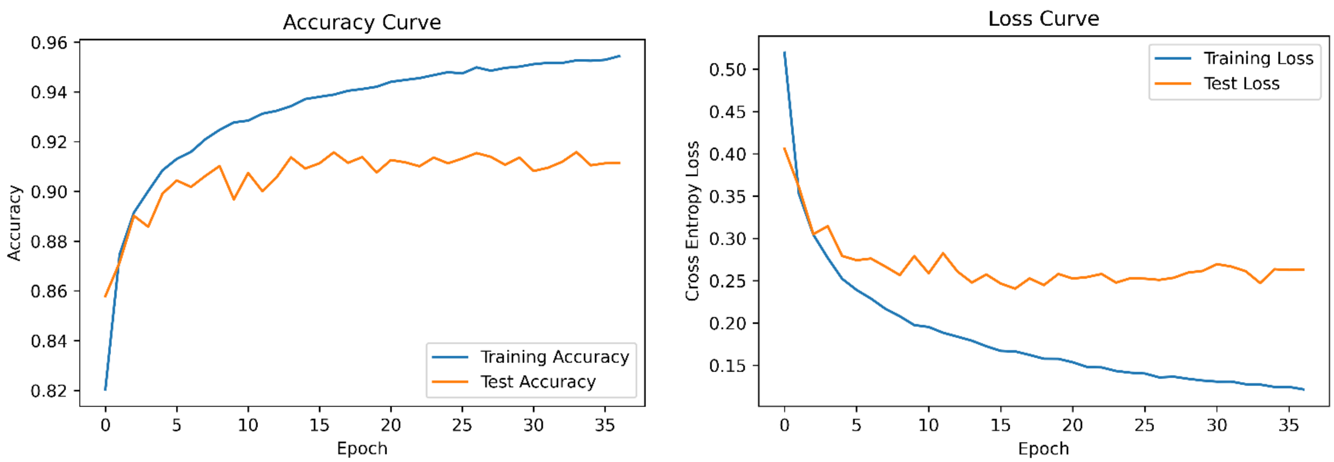
Effect of Pool Size

Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarizes the features present in a region of the feature map generated by a convolution layer. Smaller pool sizes tend to work better, as seen in our model. Too large the pool size and valuable information present in a feature map is lost.

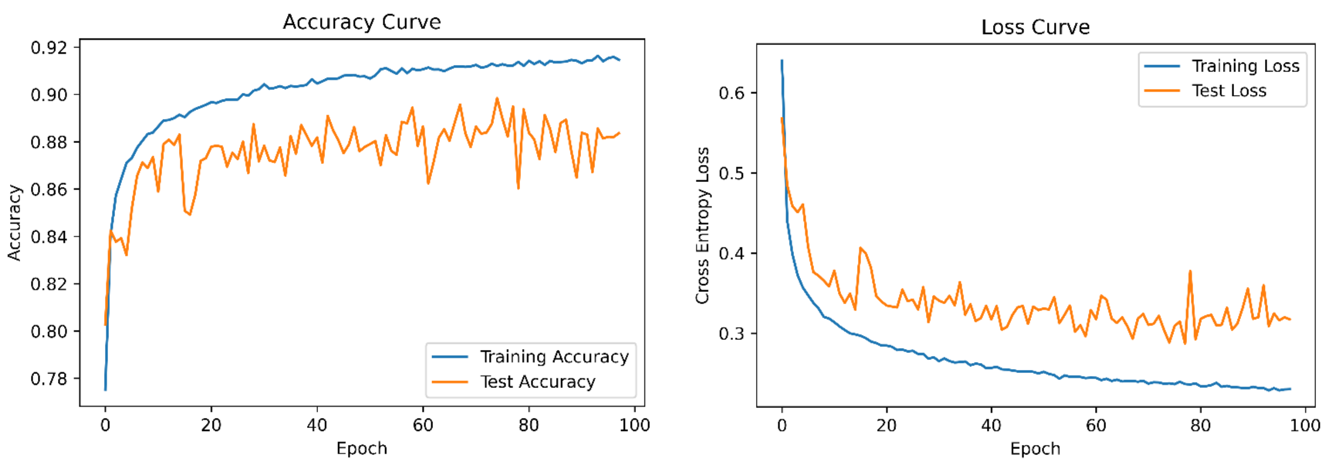
Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
16	`(64, (3,3), (1,1), 'relu', 0.2, 1, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.958	0.916	0.132	0.241
17	`(64, (3,3), (1,1), 'relu', 0.2, 5, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.926	0.895	0.218	0.287

Plots

Model 16



Model 17



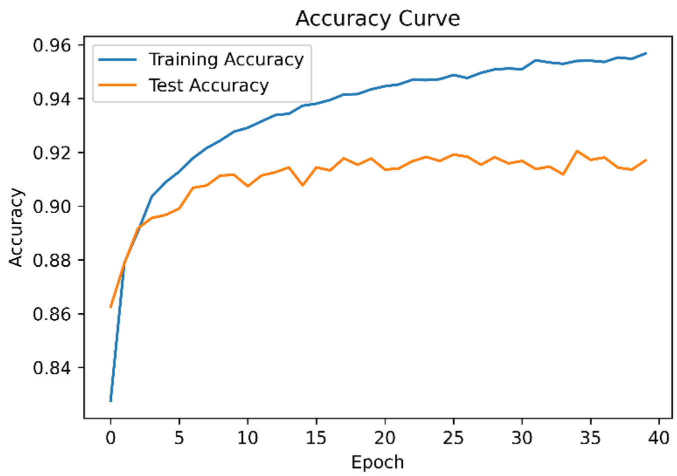
Effect of Pool Type

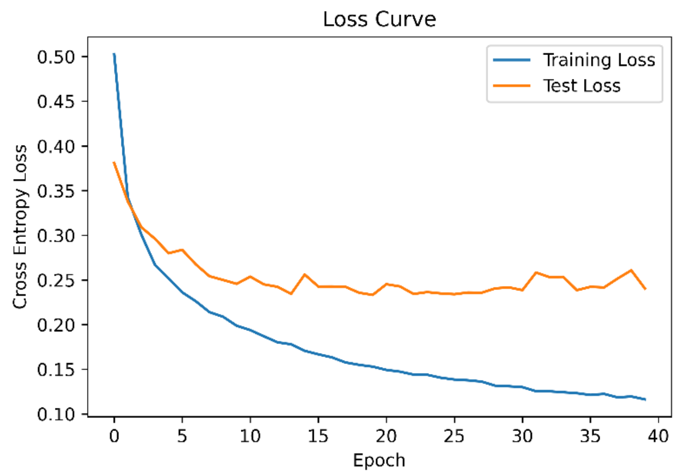
`max` and `min` pooling give us the most prominent features present in a feature map, while `average` pooling gives us a rough idea and average of features in a feature map. Although by a marginal difference, we notice that `max` pooling performs the best among the three.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
18	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'average')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.96	0.918	0.124	0.233
19	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'min')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.964	0.919	0.11	0.231

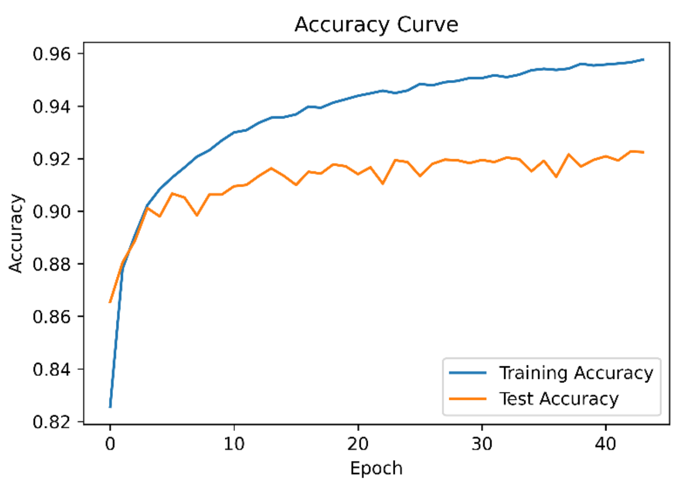
Plots

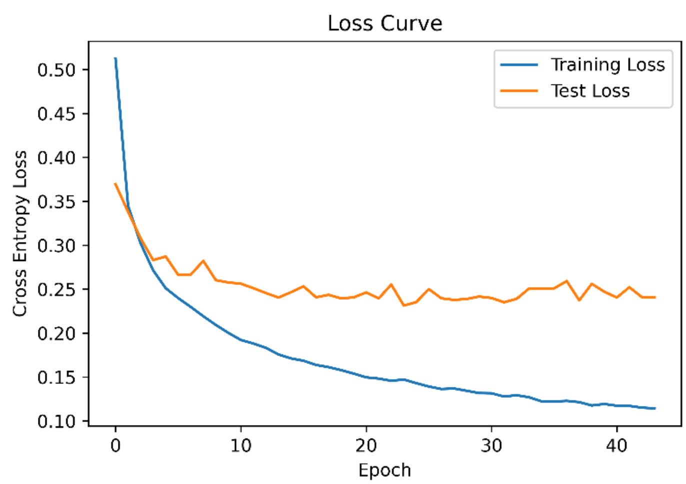
Model 18





Model 19





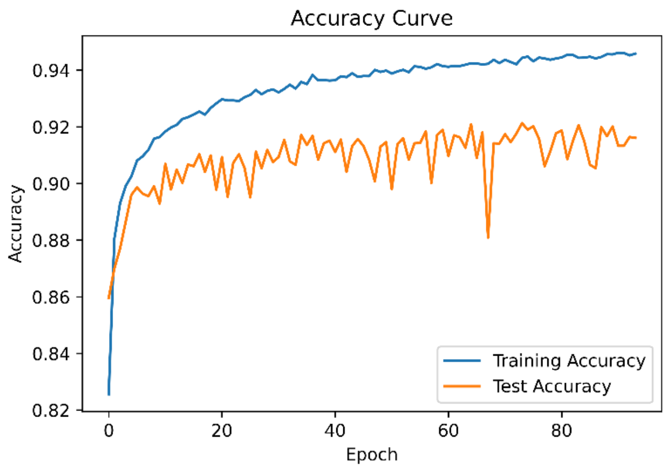
Effect of Number of Convolutional Blocks

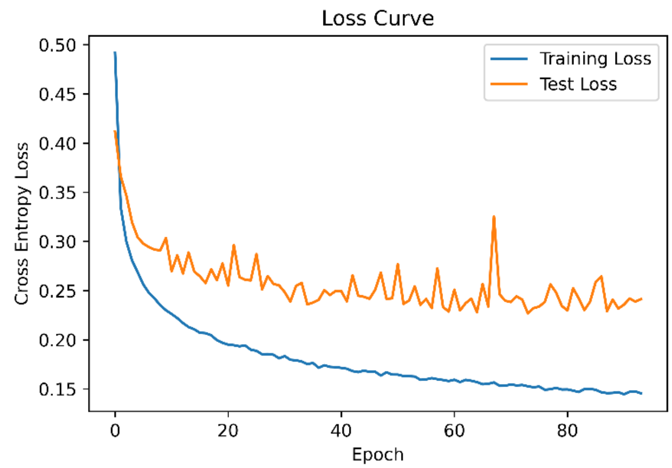
As we increase the number of convolutional blocks, the filters learn more complex features in the images. For example, the first convolutional block can extract darker edges while subsequent blocks learn lines, lighter portions, and so on. When we use one convolutional block our model cannot learn enough regions that highlight an image. As we increase the number of blocks to 2, the model learns more complex features and the accuracy increases by 1.3%.

Model Number	Tuple(s) of Hyperparameter of each layer(s)	Train Accuracy	Test Accuracy	Train Log Loss	Test Log Loss
1	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')` `(32, (2,2), (1,1), 'relu', 0.2, 2, 'max')`	0.961	0.921	0.134	0.227
22	`(64, (3,3), (1,1), 'relu', 0.2, 2, 'max')`	0.962	0.91	0.114	0.271

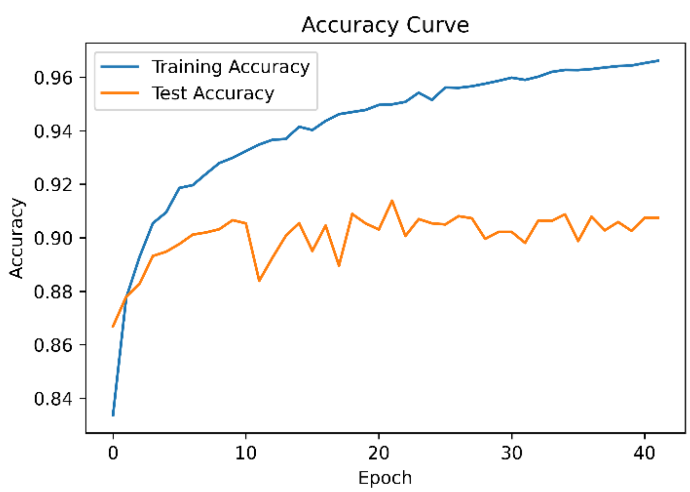
Plots

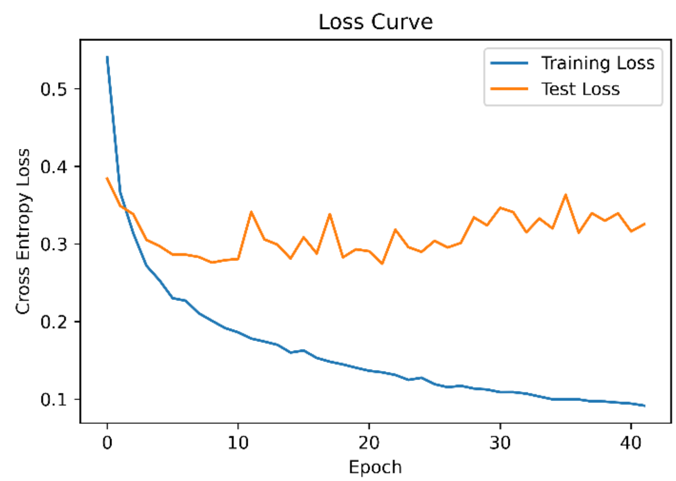
Model 1





Model 22





Conclusion

After running various models and comparing them with the base model, we have come to the conclusion that each of the hyperparameters that were taken into comparison does affect the accuracy and loss of the model. Some hyperparameters have more effect on the result, while some have a comparatively lesser effect.

References

- Dataset - Fashion MNIST | Kaggle
- Keras Documentation - Module: tf.keras | TensorFlow Core v2.6.0

Glossary

Name	Description
Activation Function	The activation function defines the output of that node given an input or set of inputs.
ADAM Optimizer	Adam optimization is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments.
Confusion Matrix	A confusion matrix is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one.
Convolutional Neural Network	A convolutional neural network (CNN) is a class of artificial neural network, most commonly applied to analyze visual imagery.
Cross-Entropy Loss	Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1.
Dropout	Dropout is a technique where randomly selected neurons are ignored during training.
Early Stopping	Early stopping is a form of regularization used to avoid overfitting when training a learner with an iterative method
Epoch	Epoch indicates the number of passes of the entire training dataset the machine learning algorithm has completed.
Fashion MNIST	Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples.
Feature Map	The feature map is the output of one filter applied to the previous layer.
Filter	In Convolutional Neural Networks, Filters detect spatial patterns such as edges in an image by detecting the changes in intensity values of the image.
Glorot Uniform Initialization	The goal of Glorot Initialization is to initialize the weights such that the variance of the activations is the same across every layer.
Keras	Keras is an open-source software library that provides a Python interface for artificial neural networks.
Overfitting	Overfitting occurs when the model performs well on the train data but poor on the test data.
Pooling	Pooling is used to reduce the spatial dimension of the activations.
Stride	Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video.
Underfitting	A model is said to be underfitting when it's not able to classify training data.