Deep Learning (CS F425)
Assignment - 1

Group Members:

AnishKumar K- 2019A7PS0091H

Pranav Jibhakate - 2019A7PS0144H

Abhipal Sharma - 2019A7PS0161H

Under the guidance of :

Prof. NL Bhanumurthy

Table of Contents

    c. Nodes in hidden layers : 16, 64
3. Model 9
    a. Loss: Categorical cross-entropy
    b. Number of hidden layers : 4
    c. Nodes in hidden layers : 16, 32, 64, 128
4. Model 10
    a. Loss: KL divergence
    b. Number of hidden layers : 5
    c. Nodes in hidden layers : 16, 32, 64, 128, 64
5. Model 11
    a. Loss: Categorical cross-entropy
    b. Number of hidden layers : 6
    c. Nodes in hidden layers : 16, 32, 64, 128, 512, 128

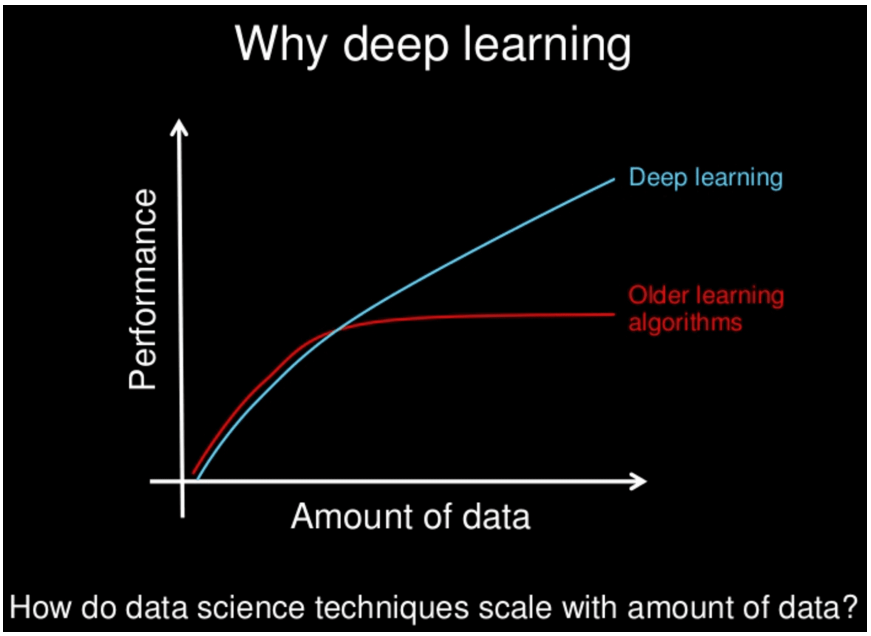
3. Description of models based on **ReLU Activation** Function
    1. Model 12
        a. Loss: Categorical cross-entropy
        b. Number of hidden layers : 1
        c. Nodes in hidden layers : 16
    2. Model 13
        a. Loss: KL divergence
        b. Number of hidden layers : 2
        c. Nodes in hidden layers : 16, 64
    3. Model 14
        a. Loss: Categorical cross-entropy
        b. Number of hidden layers : 4
        c. Nodes in hidden layers : 16, 32, 64, 128
    4. Model 15
        a. Loss: KL divergence
        b. Number of hidden layers : 5
        c. Nodes in hidden layers : 16, 32, 64, 128, 64
    5. Model 16
        a. Loss: Categorical cross-entropy
        b. Number of hidden layers : 6
        c. Nodes in hidden layers : 16, 32, 64, 128, 512, 128

Introduction

Deep Learning:

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier.

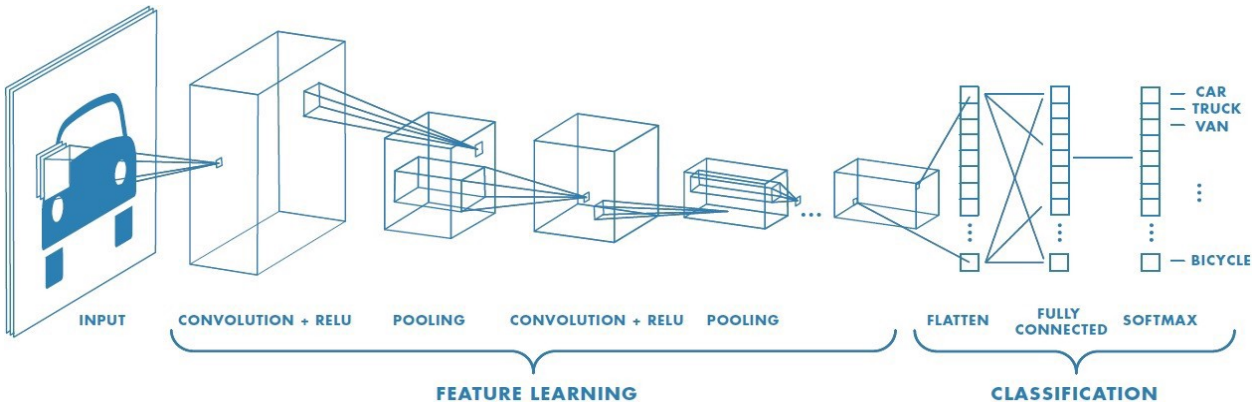
At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.



Each algorithm in the hierarchy applies a nonlinear transformation to its input and uses what it learns to create a statistical model as output. Iterations continue until the output has reached an acceptable level of accuracy. The number of processing layers through which data must pass is what inspired the label deep.

The advantage of deep learning is the program builds the feature set by itself without supervision. Unsupervised learning is not only faster, but it is usually more accurate.

Convolutional Neural Networks:



A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

ConvNets are preferred over Feed-Forward Neural Nets as a ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer

is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

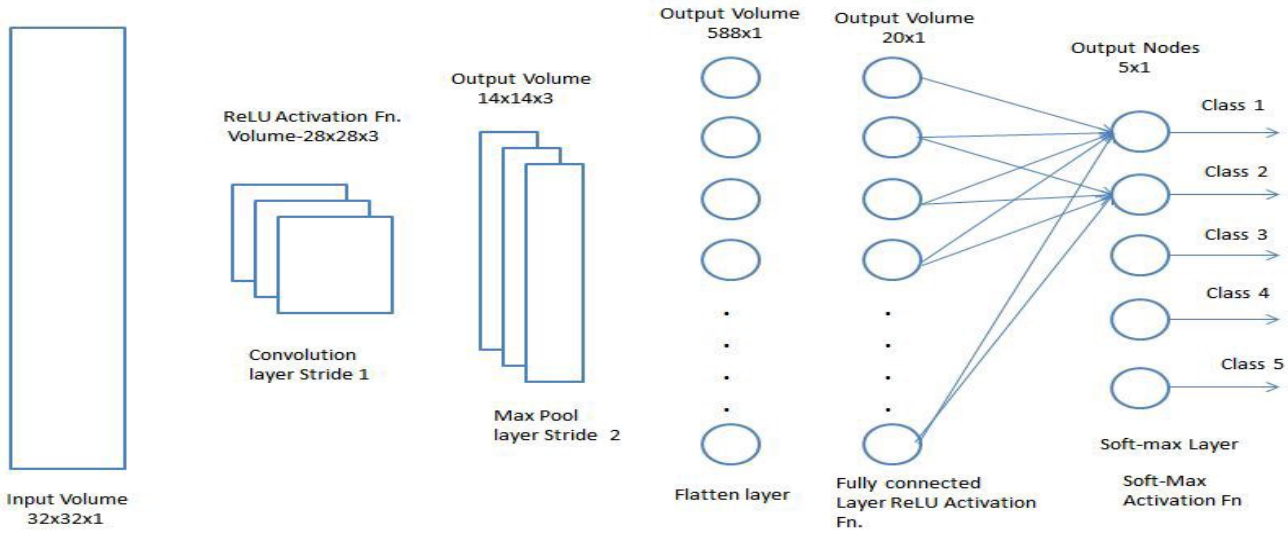
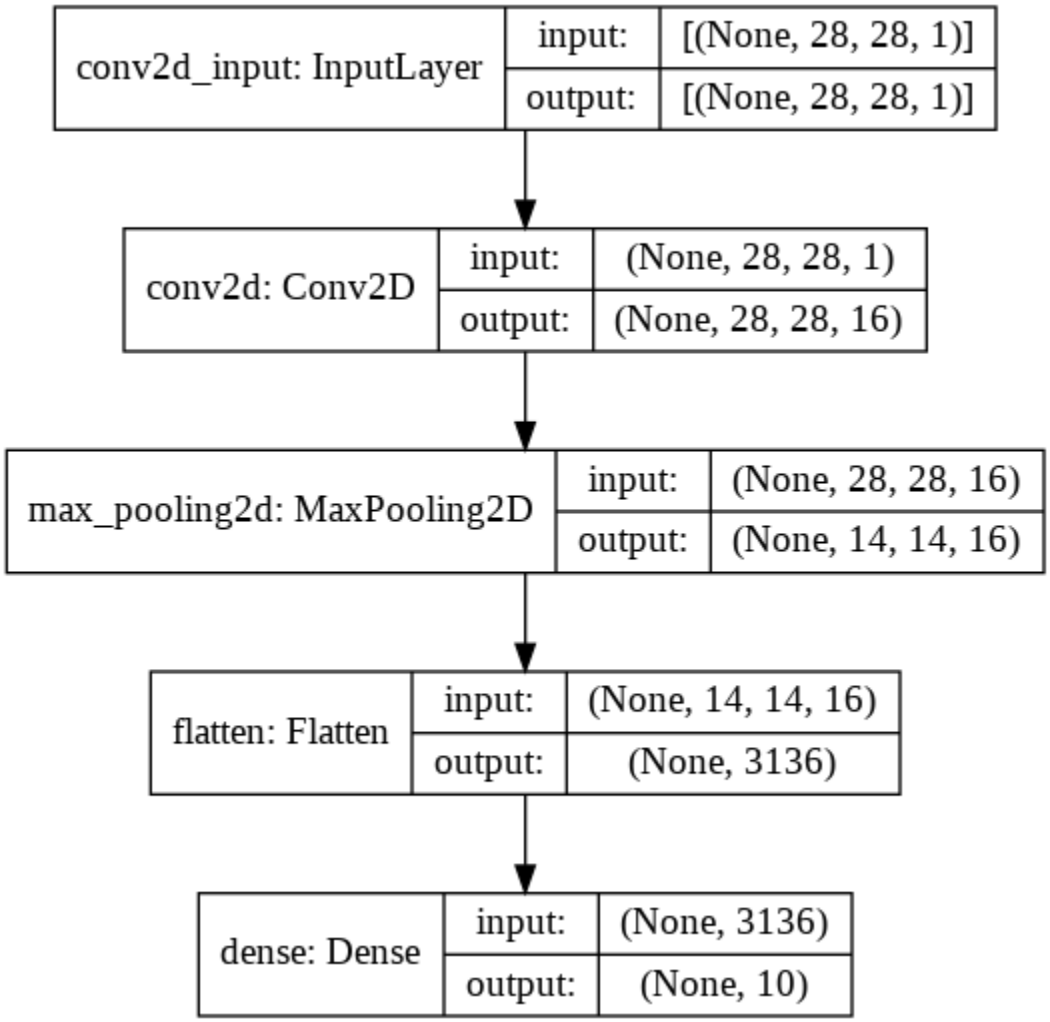
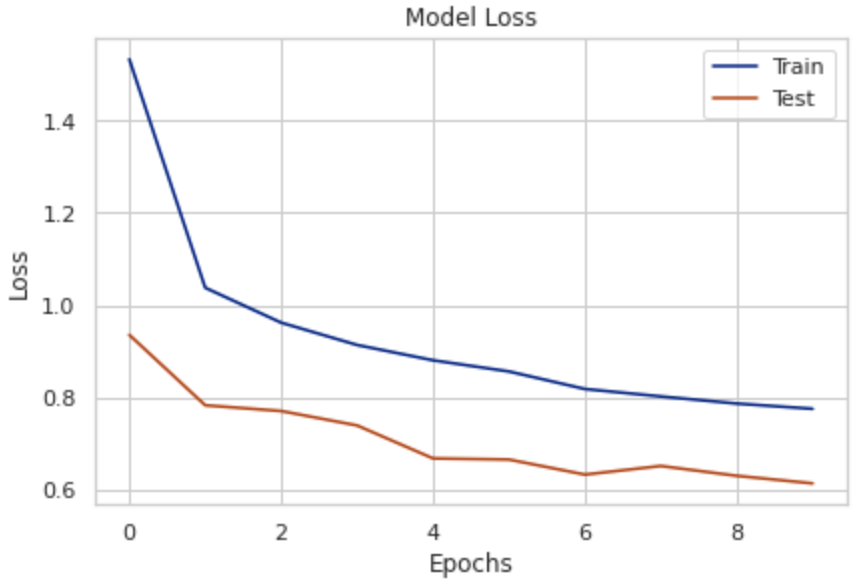


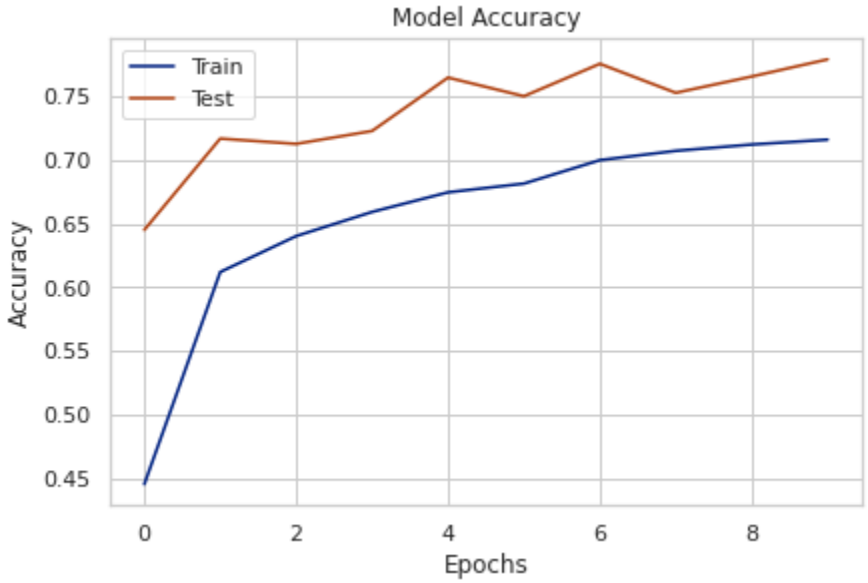
Fig : A CNN with one hidden Conv layer and one hidden fully connected layer

Description of Models based on Sigmoid Activation

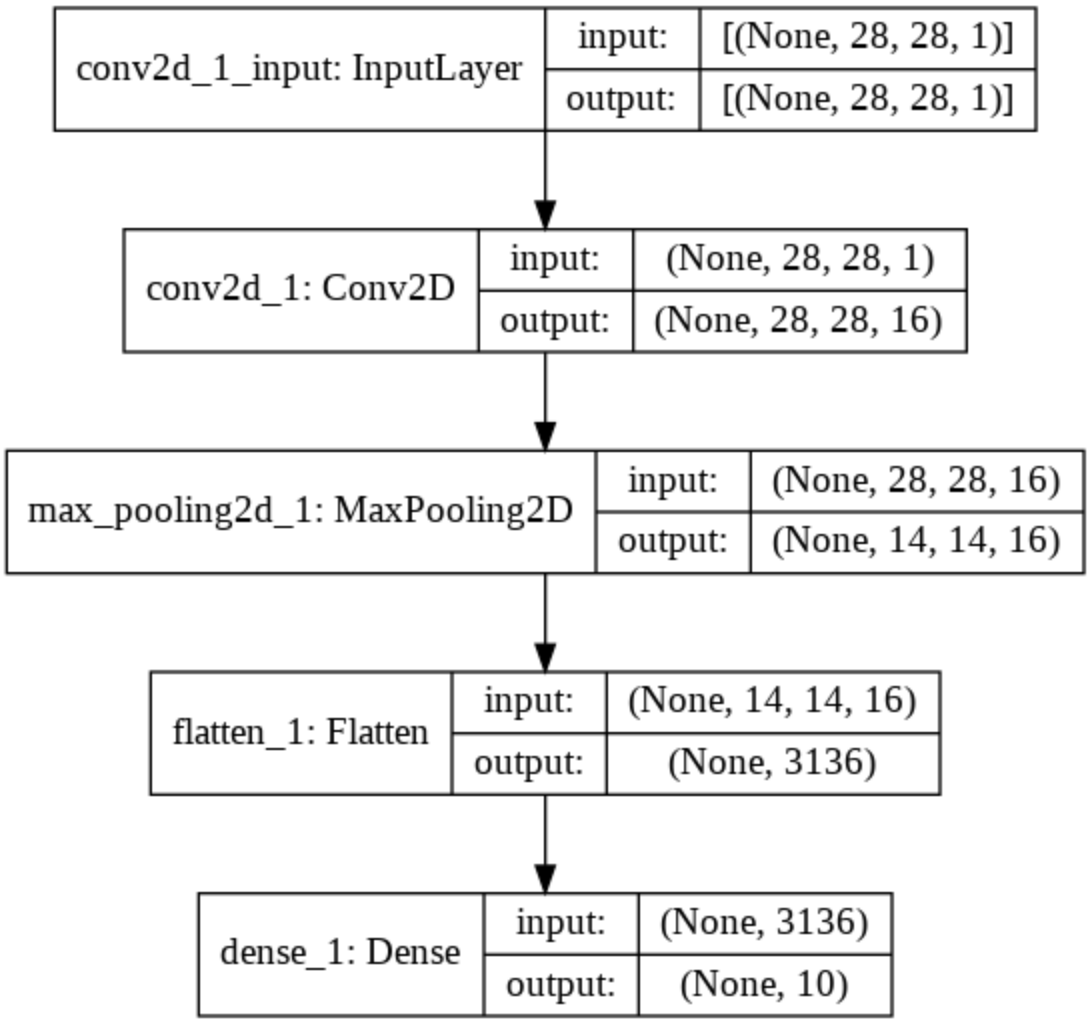
1. **Model 1**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 1
    - Nodes in hidden layers : 16
    - Validation loss : 0.6129
    - Validation accuracy : 0.7793

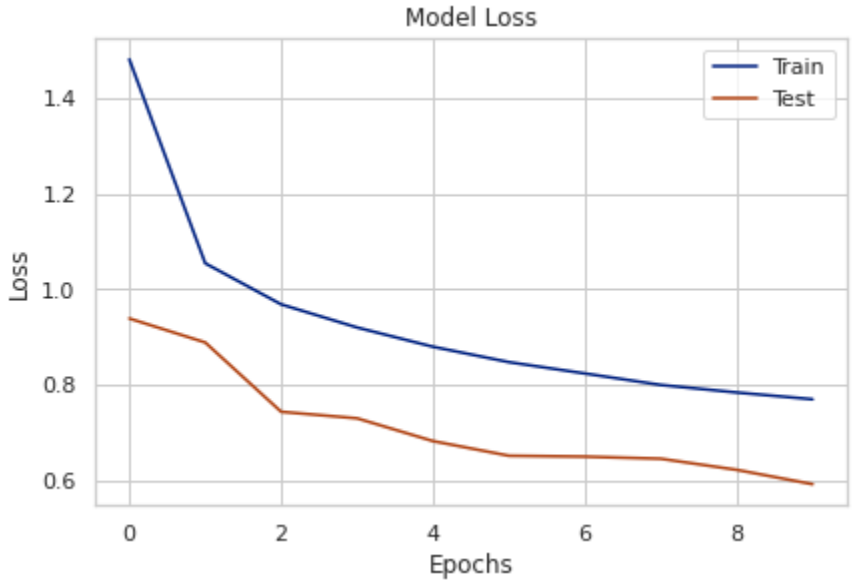


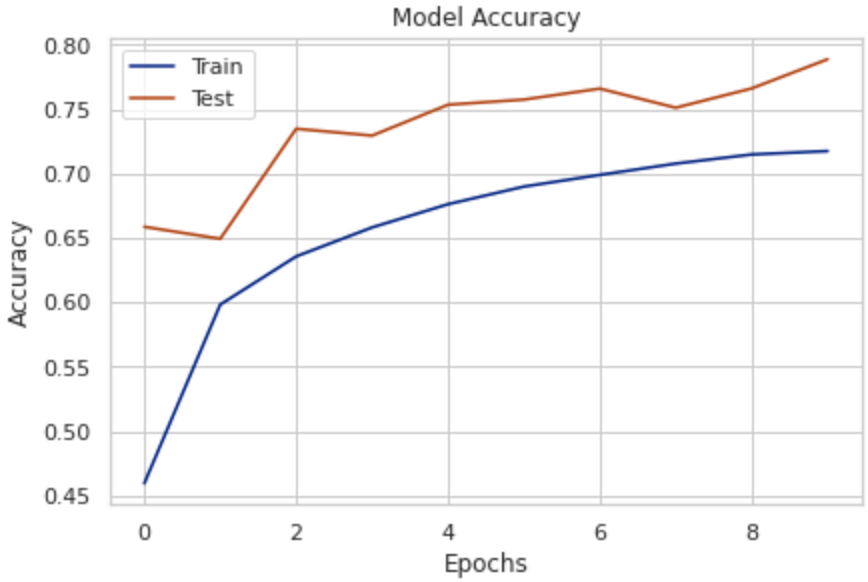




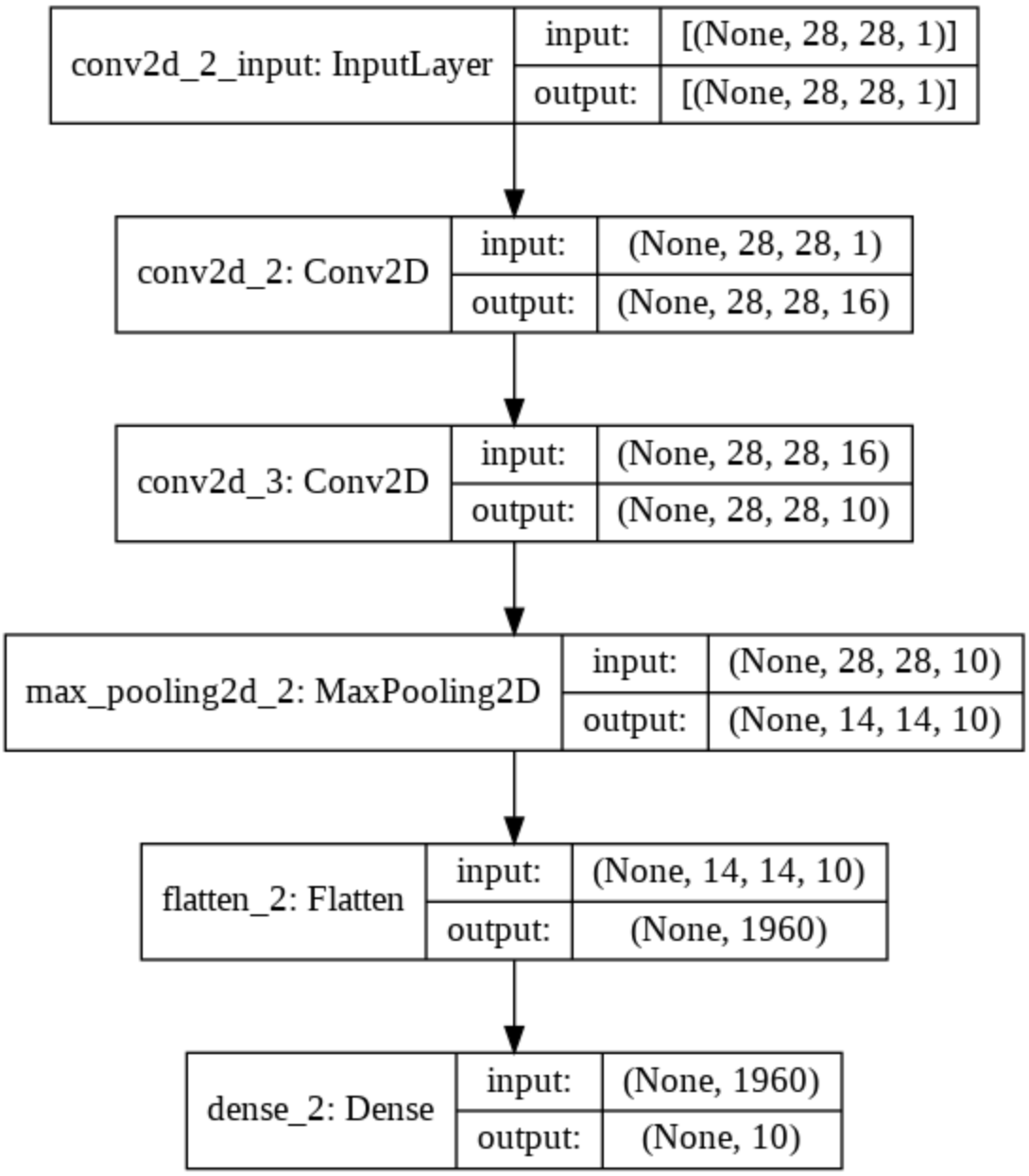
2. **Model 2**
    - Loss: KL Divergence
    - Number of hidden layers : 1
    - Nodes in hidden layers : 16
    - Validation loss : 0.5912
    - Validation accuracy : 0.7888

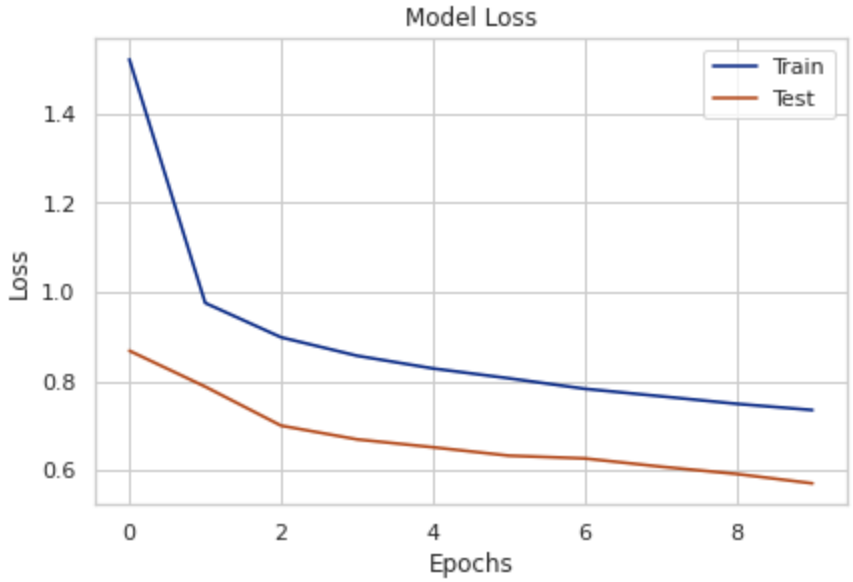

conv2d_1_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_1: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
max_pooling2d_1: MaxPooling2D
input: (None, 28, 28, 16)
output: (None, 14, 14, 16)
flatten_1: Flatten
input: (None, 14, 14, 16)
output: (None, 3136)
dense_1: Dense
input: (None, 3136)
output: (None, 10)

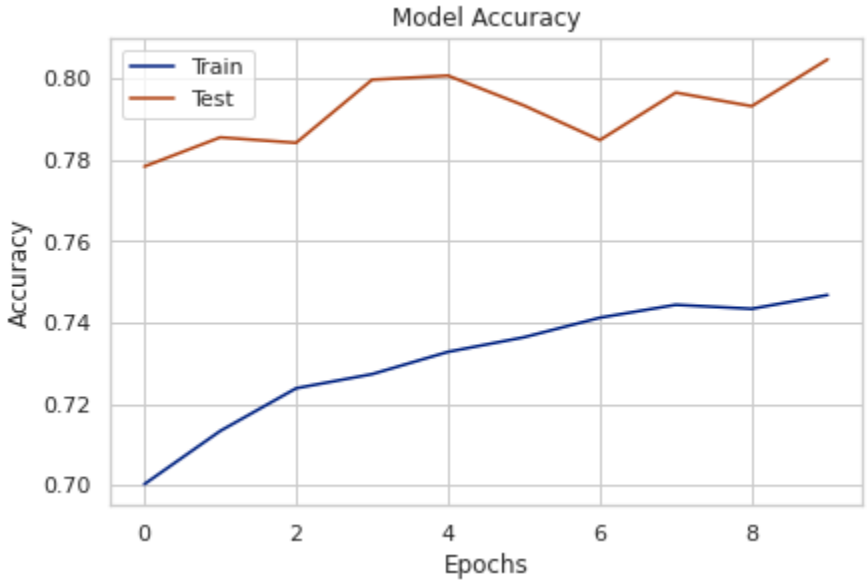

Model Loss
Train
Test
Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8



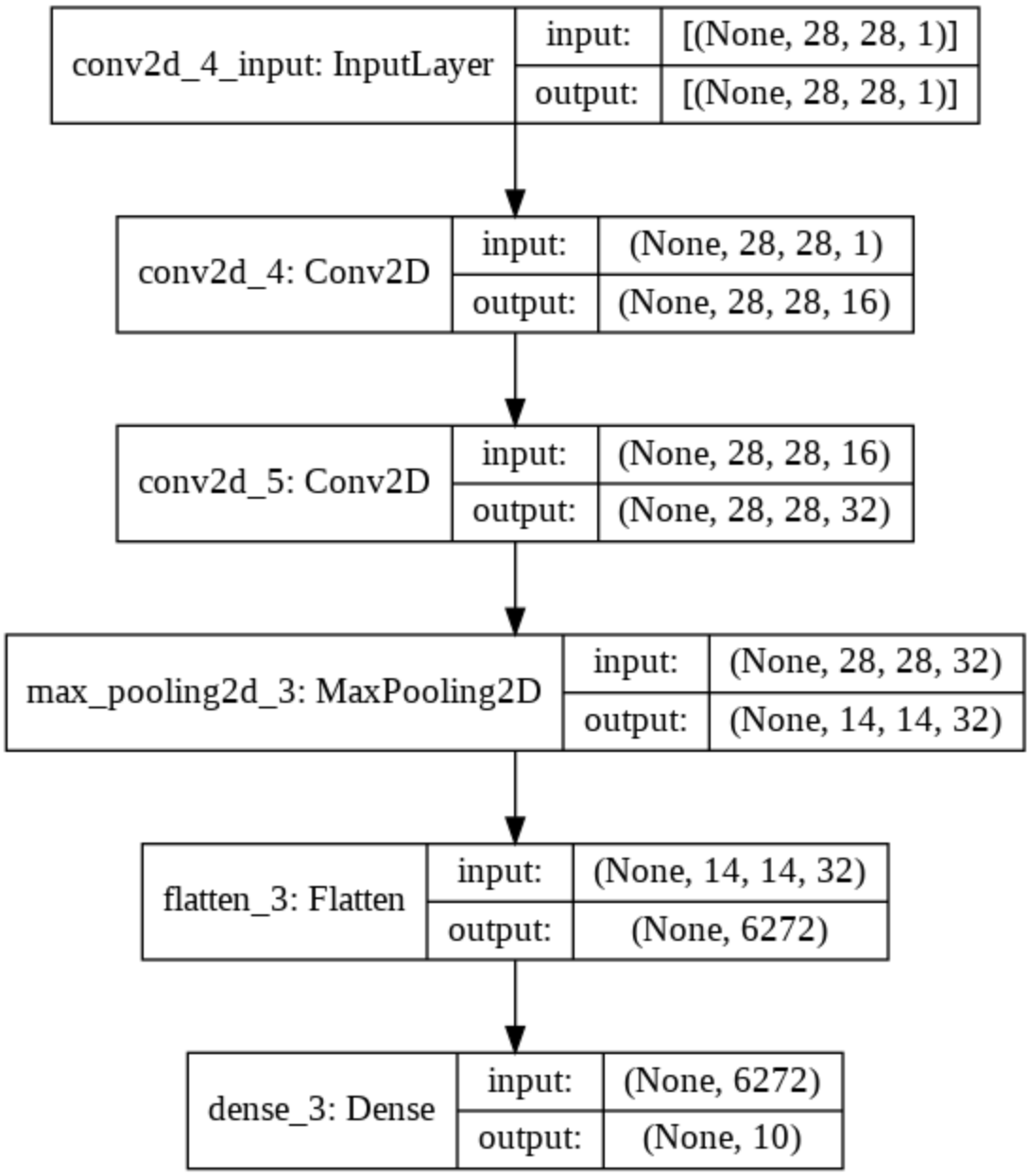
3. **Model 3**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 2
    - Nodes in hidden layers : 16,10
    - Validation loss : 0.5714
    - Validation accuracy : 0.7925

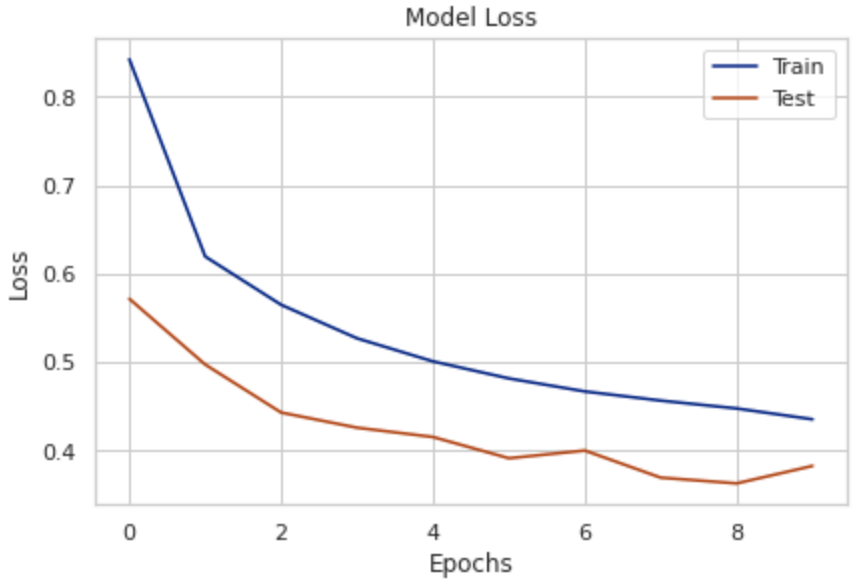

conv2d_2_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_2: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_3: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 10)
max_pooling2d_2: MaxPooling2D
input: (None, 28, 28, 10)
output: (None, 14, 14, 10)
flatten_2: Flatten
input: (None, 14, 14, 10)
output: (None, 1960)
dense_2: Dense
input: (None, 1960)
output: (None, 10)

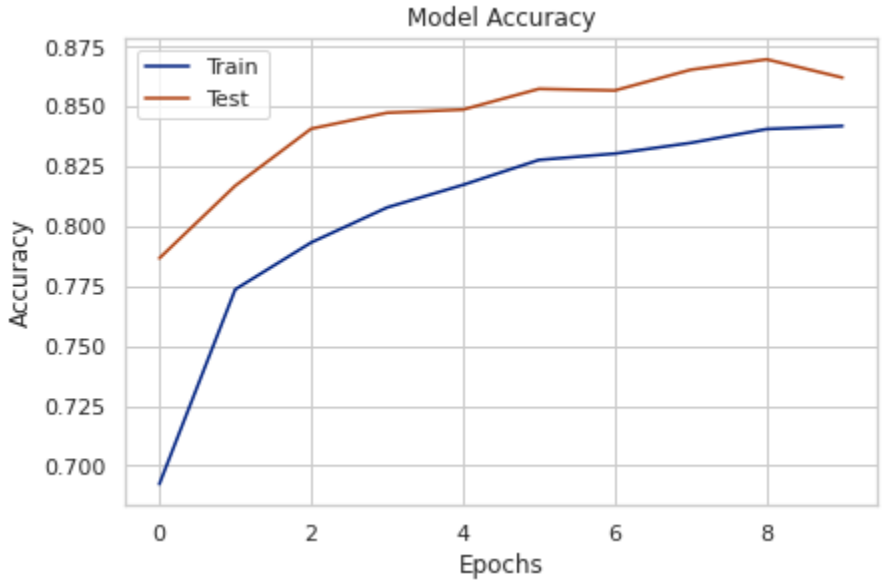




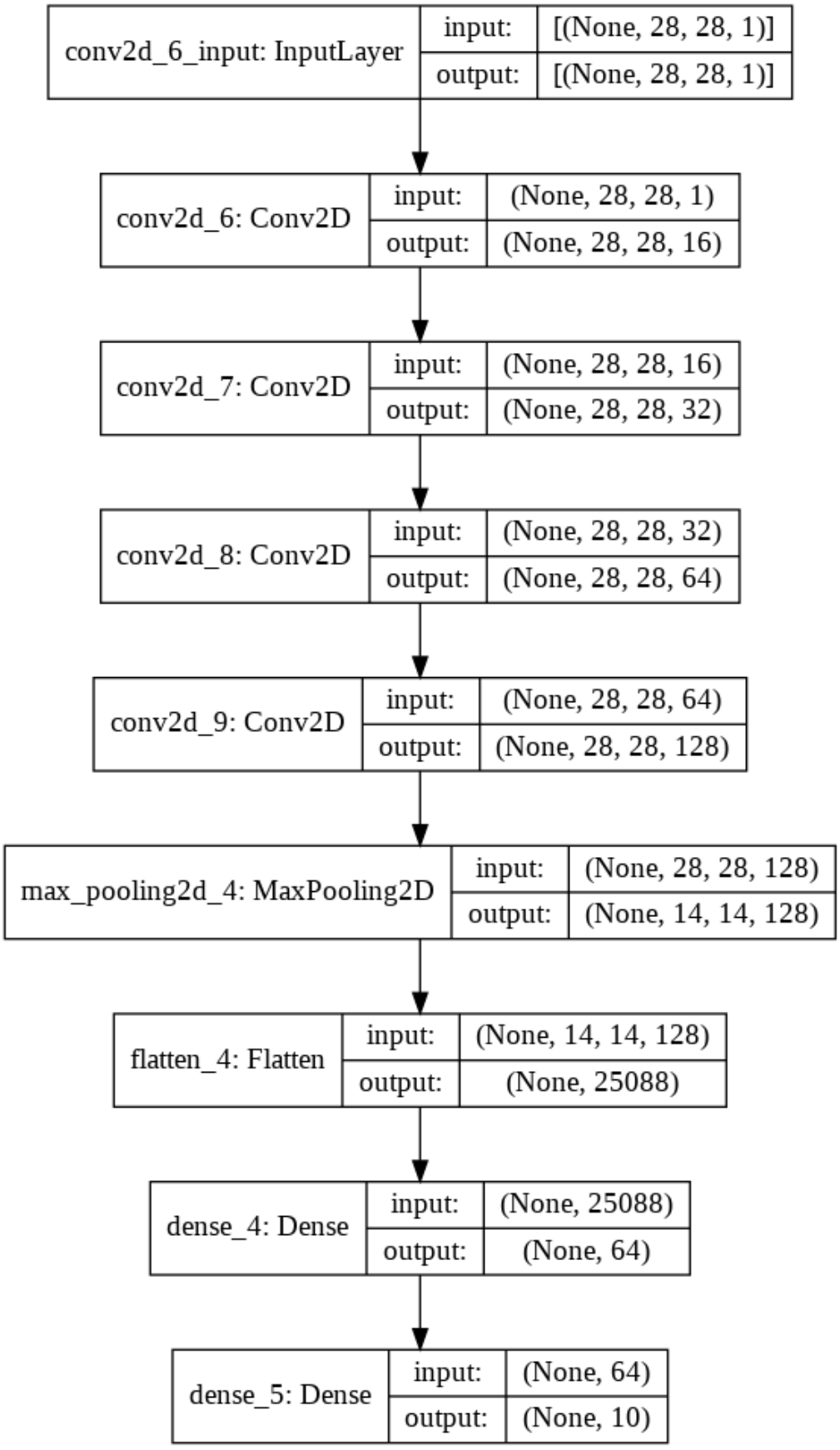
4. **Model 4**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 2
    - Nodes in hidden layers : 16,32
    - Validation loss : 0.3830
    - Validation accuracy : 0.8622

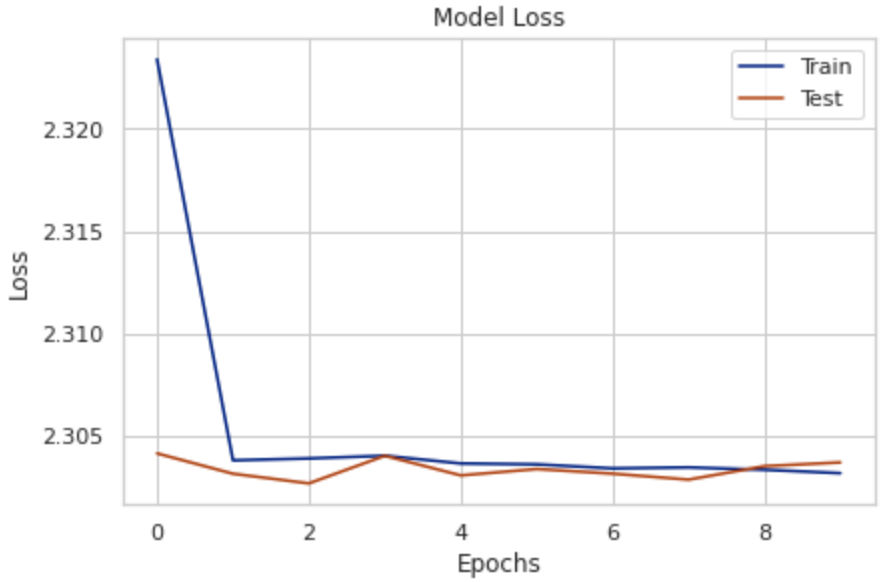

conv2d_4_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_4: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_5: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 32)
max_pooling2d_3: MaxPooling2D
input: (None, 28, 28, 32)
output: (None, 14, 14, 32)
flatten_3: Flatten
input: (None, 14, 14, 32)
output: (None, 6272)
dense_3: Dense
input: (None, 6272)
output: (None, 10)

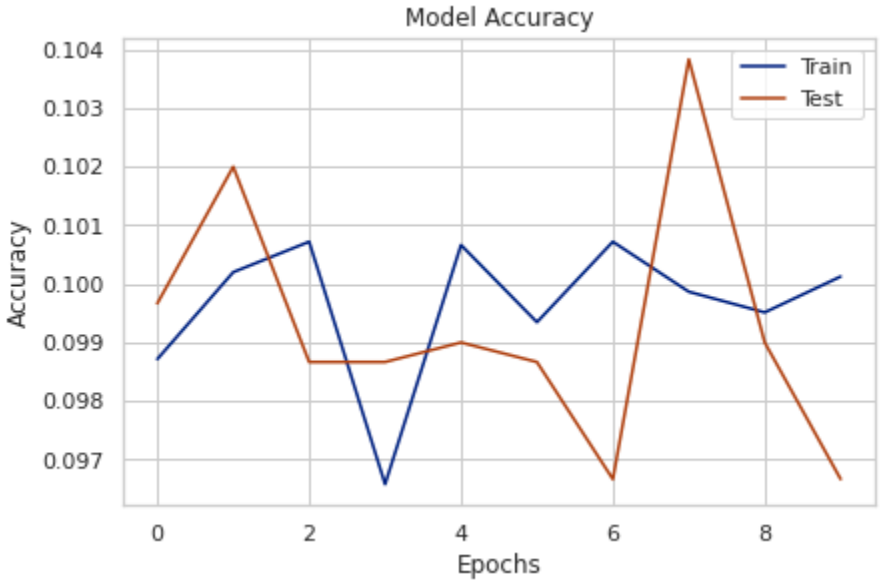




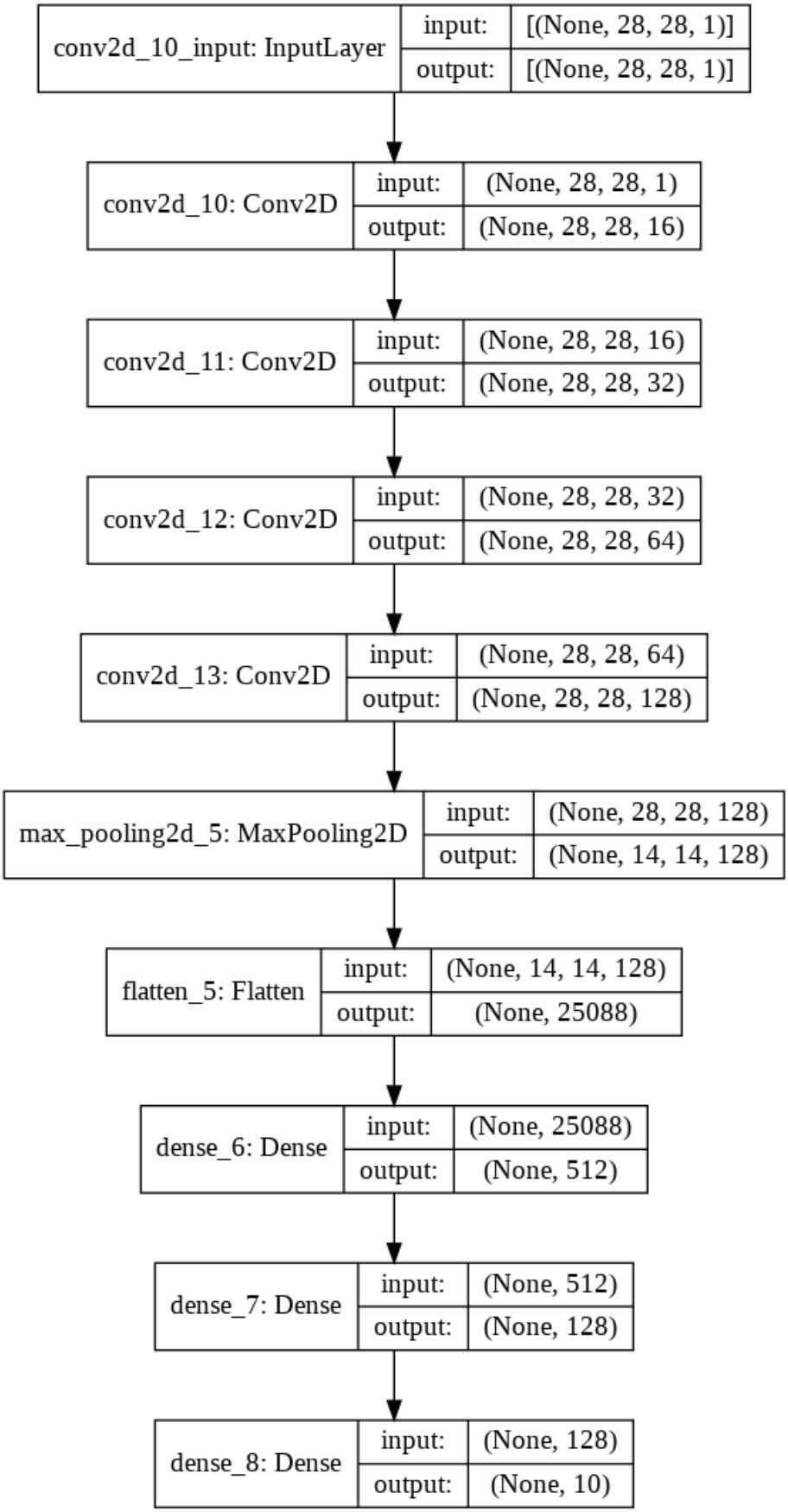
5. **Model 5**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 5
    - Nodes in hidden layers : 16,32,64,128,64
    - Validation loss : 0.2304
    - Validation accuracy : 0.8735

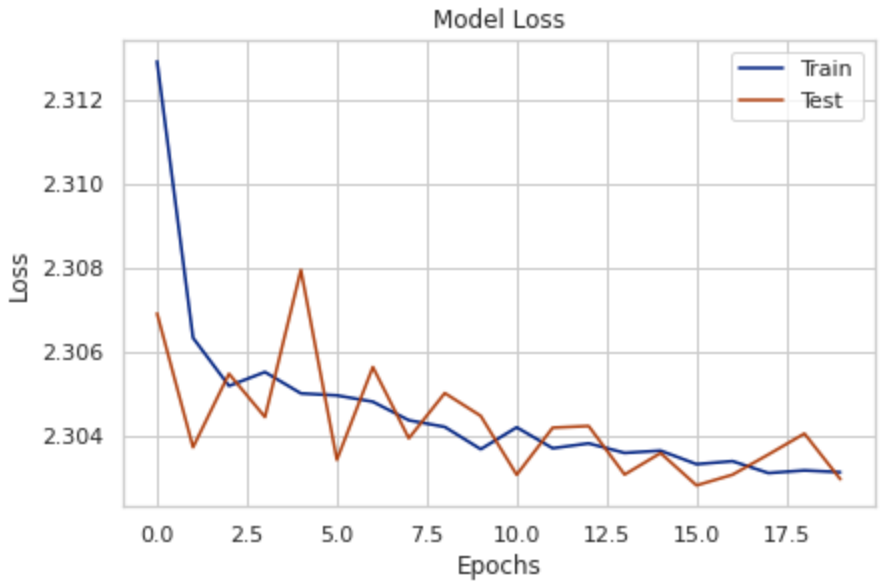

conv2d_6_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_6: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_7: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 32)
conv2d_8: Conv2D
input: (None, 28, 28, 32)
output: (None, 28, 28, 64)
conv2d_9: Conv2D
input: (None, 28, 28, 64)
output: (None, 28, 28, 128)
max_pooling2d_4: MaxPooling2D
input: (None, 28, 28, 128)
output: (None, 14, 14, 128)
flatten_4: Flatten
input: (None, 14, 14, 128)
output: (None, 25088)
dense_4: Dense
input: (None, 25088)
output: (None, 64)
dense_5: Dense
input: (None, 64)
output: (None, 10)

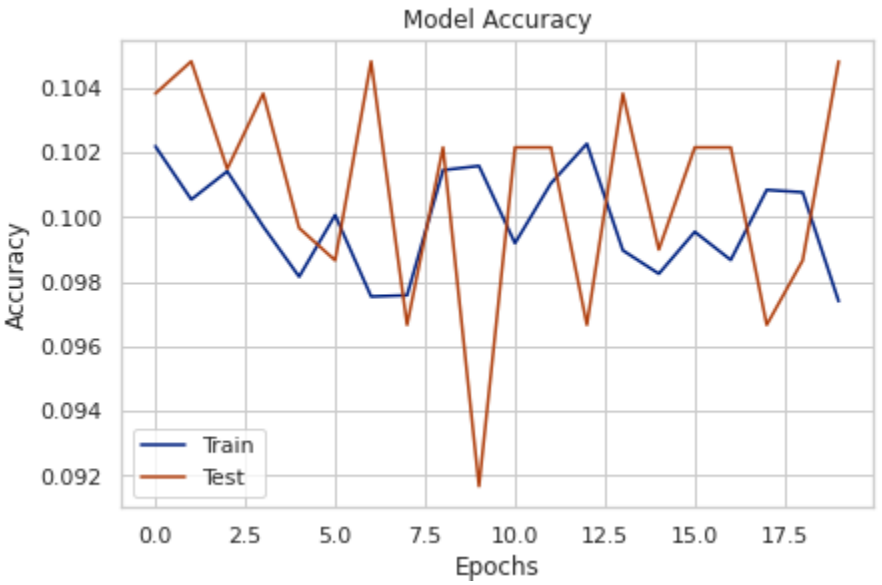




6. **Model 6**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 6
    - Nodes in hidden layers : 16, 32, 64, 128, 512, 128
    - Validation loss : 0.2304
    - Validation accuracy : 0.8901

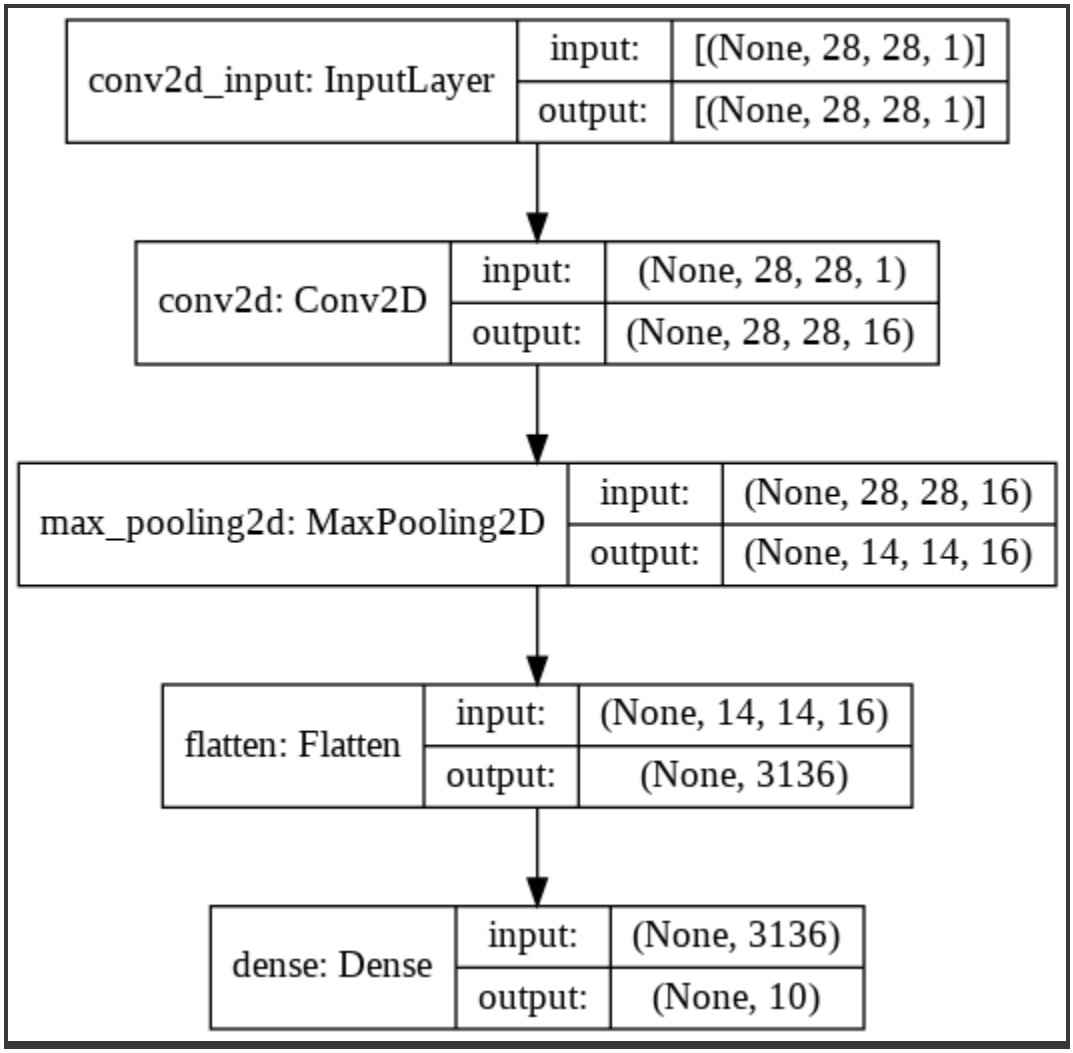

conv2d_10_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_10: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_11: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 32)
conv2d_12: Conv2D
input: (None, 28, 28, 32)
output: (None, 28, 28, 64)
conv2d_13: Conv2D
input: (None, 28, 28, 64)
output: (None, 28, 28, 128)
max_pooling2d_5: MaxPooling2D
input: (None, 28, 28, 128)
output: (None, 14, 14, 128)
flatten_5: Flatten
input: (None, 14, 14, 128)
output: (None, 25088)
dense_6: Dense
input: (None, 25088)
output: (None, 512)
dense_7: Dense
input: (None, 512)
output: (None, 128)
dense_8: Dense
input: (None, 128)
output: (None, 10)

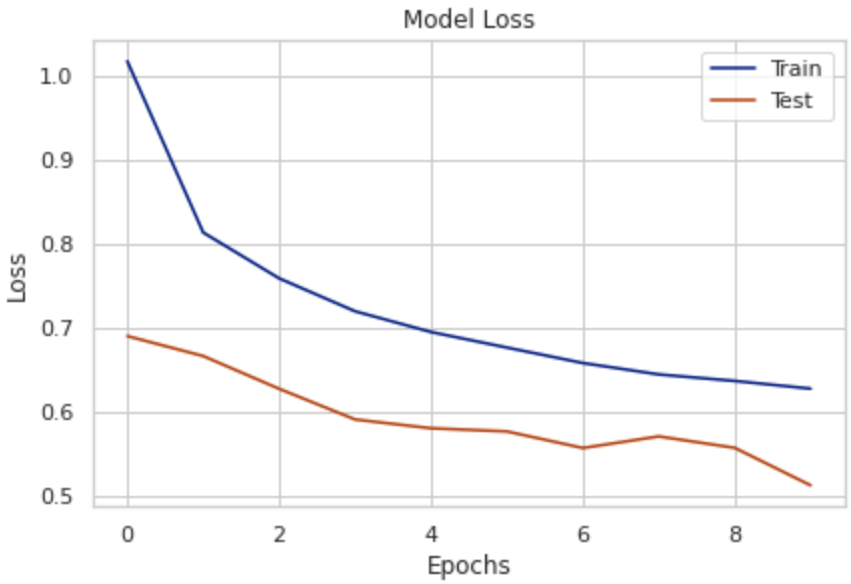

Model Loss
Train
Test
2.312
2.310
2.308
2.306
2.304
Loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

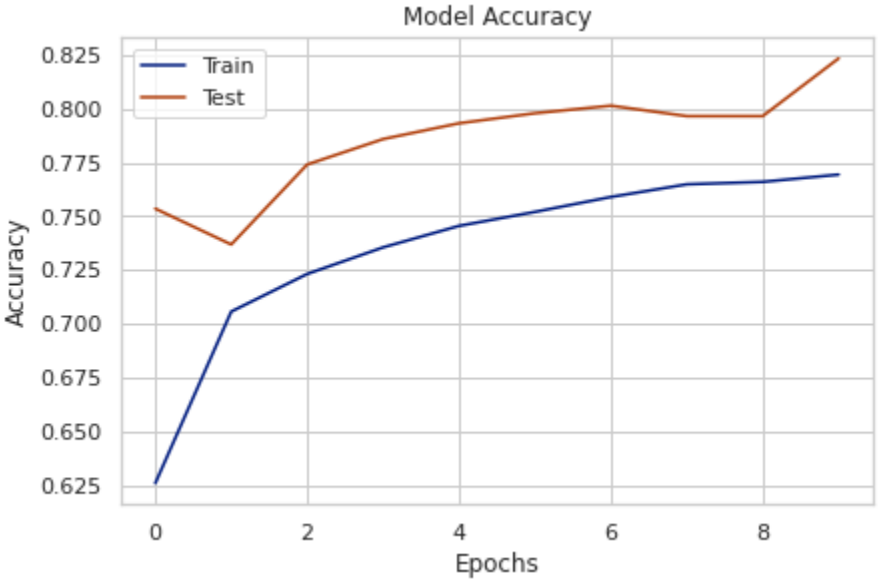

Model Accuracy
0.104
0.102
0.100
0.098
0.096
0.094
0.092
Accuracy
Train
Test
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

Description of models based on Tanh Activation function

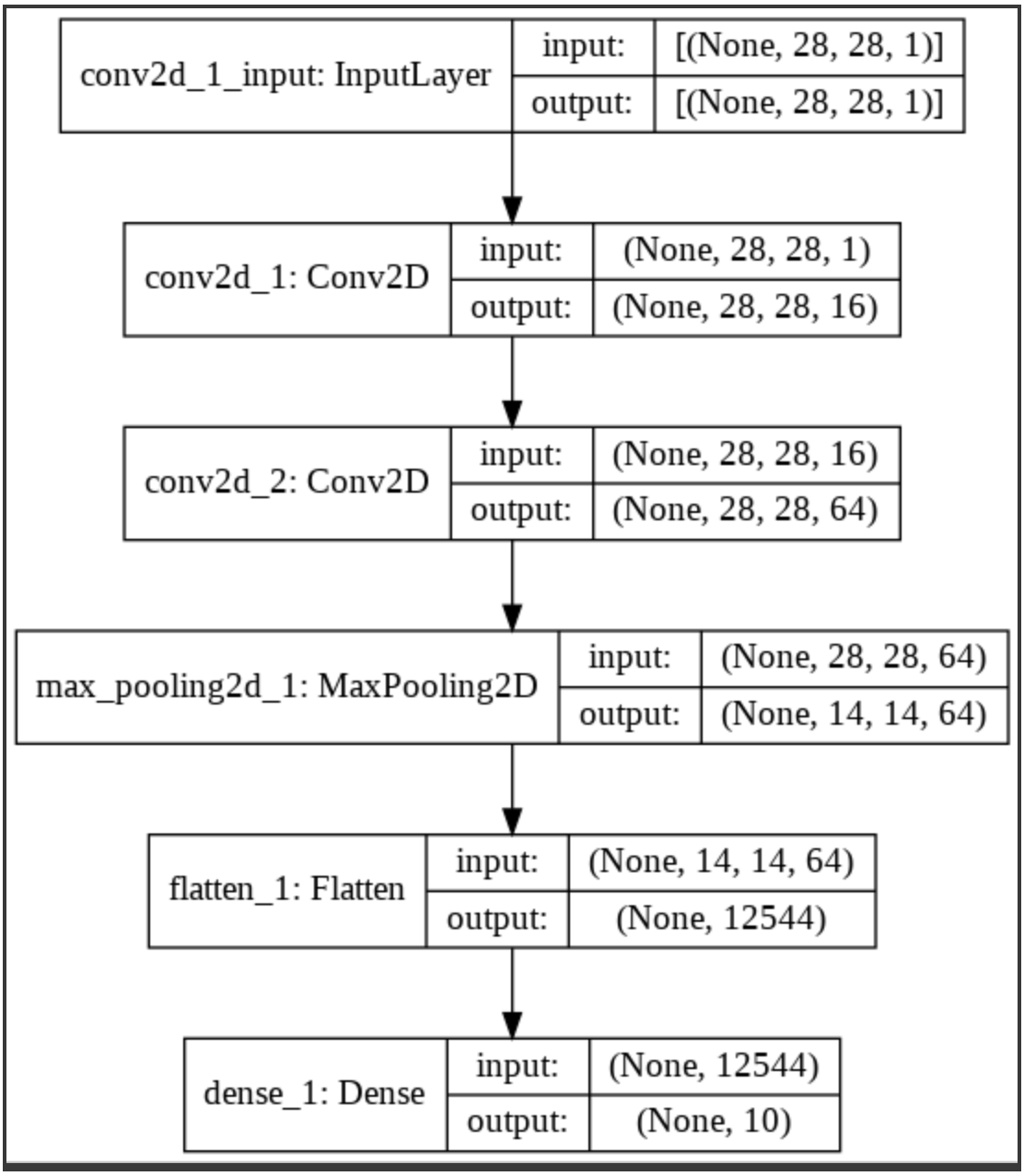
1. **MODEL 7**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 1
    - Nodes in hidden layers : 16
    - Validation Loss : `0.5116`
    - Validation Accuracy : `0.8343`

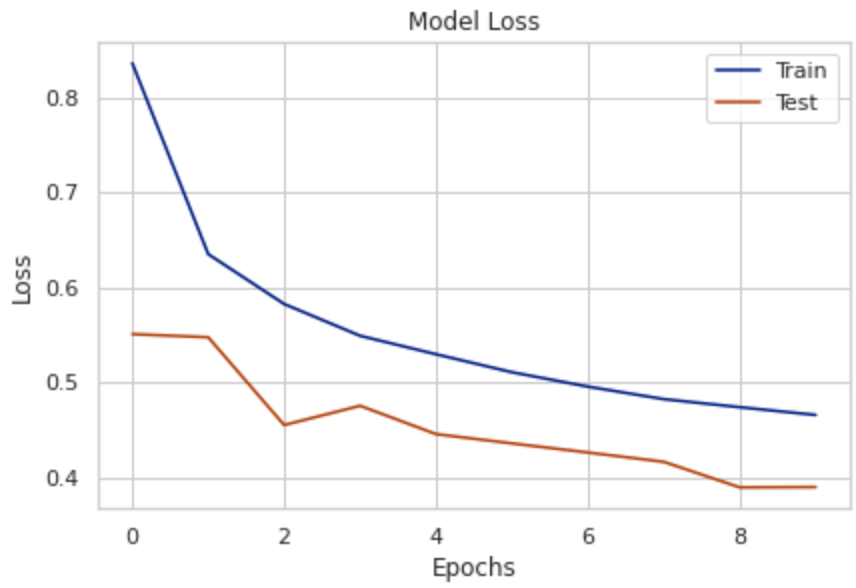


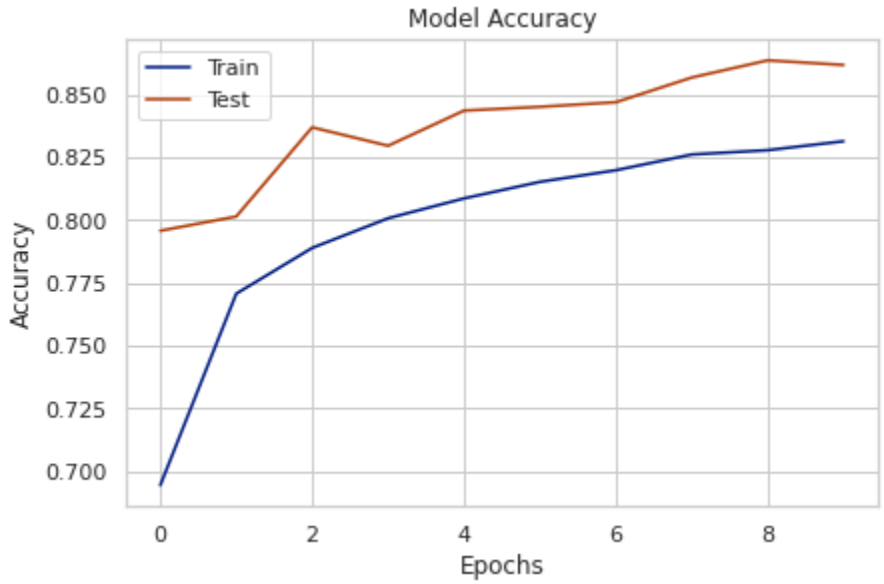




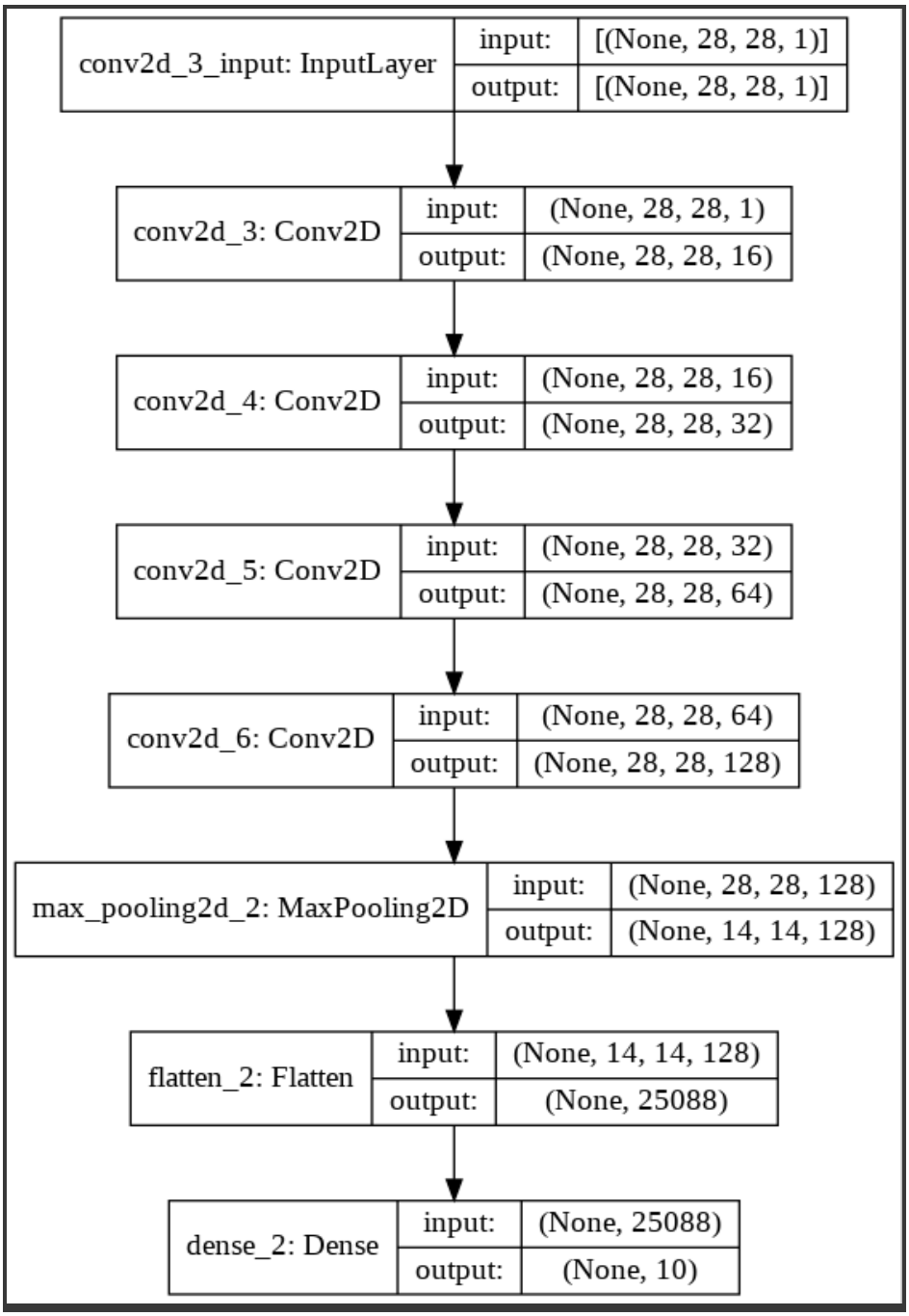
2. **MODEL 8**
    - Loss: KL divergence
    - Number of hidden layers : 2
    - Nodes in hidden layers : 16, 64
    - Validation Loss : `0.3900`
    - Validation Accuracy : `0.8618`

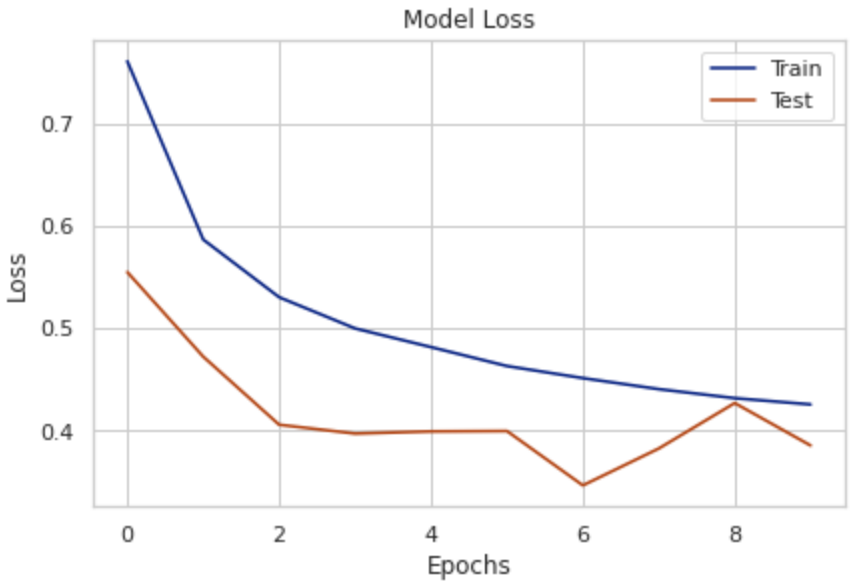

conv2d_1_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_1: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_2: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 28, 28, 64)
output: (None, 14, 14, 64)
flatten_1: Flatten
input: (None, 14, 14, 64)
output: (None, 12544)
dense_1: Dense
input: (None, 12544)
output: (None, 10)

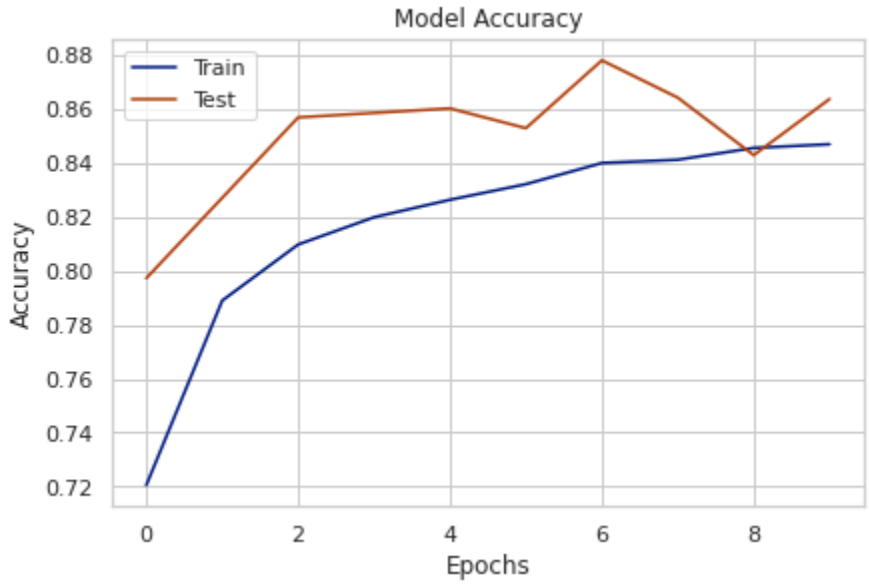




3. **MODEL 9**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 4
    - Nodes in hidden layers : 16, 32, 64, 128
    - Validation Loss : `0.3856`
    - Validation Accuracy : `0.8638`


conv2d_3_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_3: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_4: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 32)
conv2d_5: Conv2D
input: (None, 28, 28, 32)
output: (None, 28, 28, 64)
conv2d_6: Conv2D
input: (None, 28, 28, 64)
output: (None, 28, 28, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 28, 28, 128)
output: (None, 14, 14, 128)
flatten_2: Flatten
input: (None, 14, 14, 128)
output: (None, 25088)
dense_2: Dense
input: (None, 25088)
output: (None, 10)

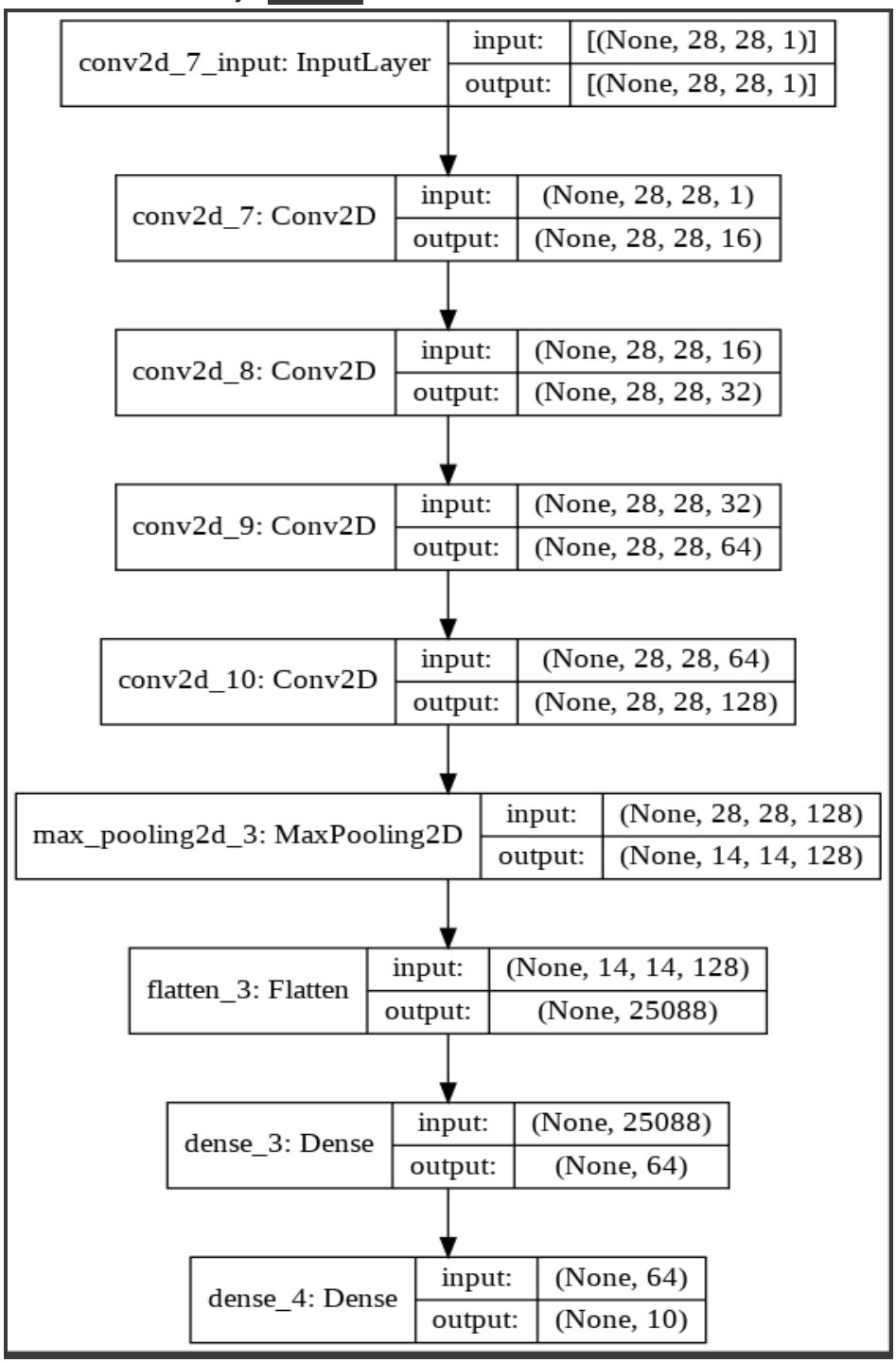


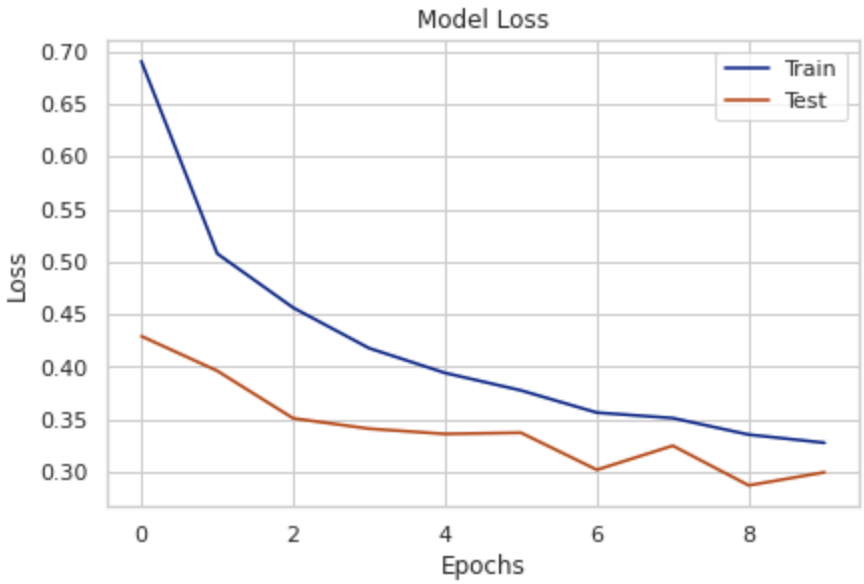


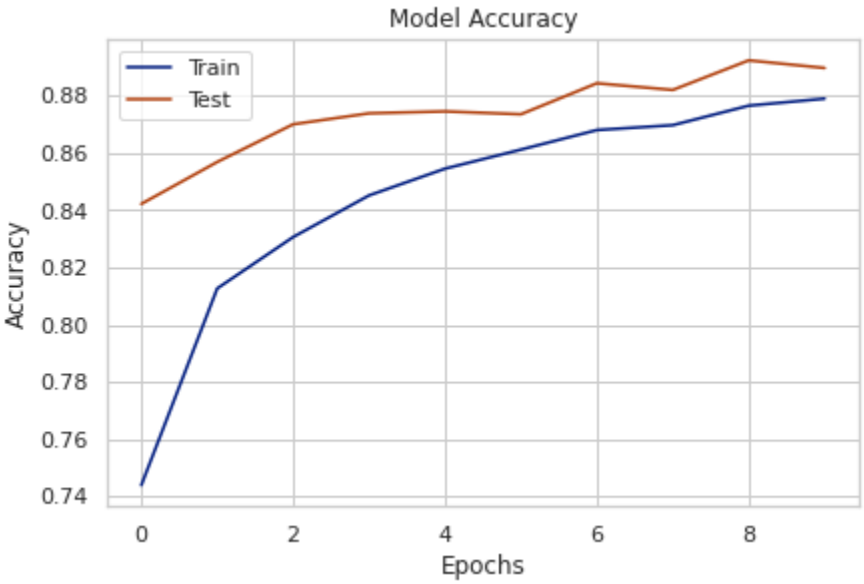
4. MODEL 10

- Loss: KL divergence
- Number of hidden layers : 5
- Nodes in hidden layers : 16, 32, 64, 128, 64
- Validation Loss : `0.2995`

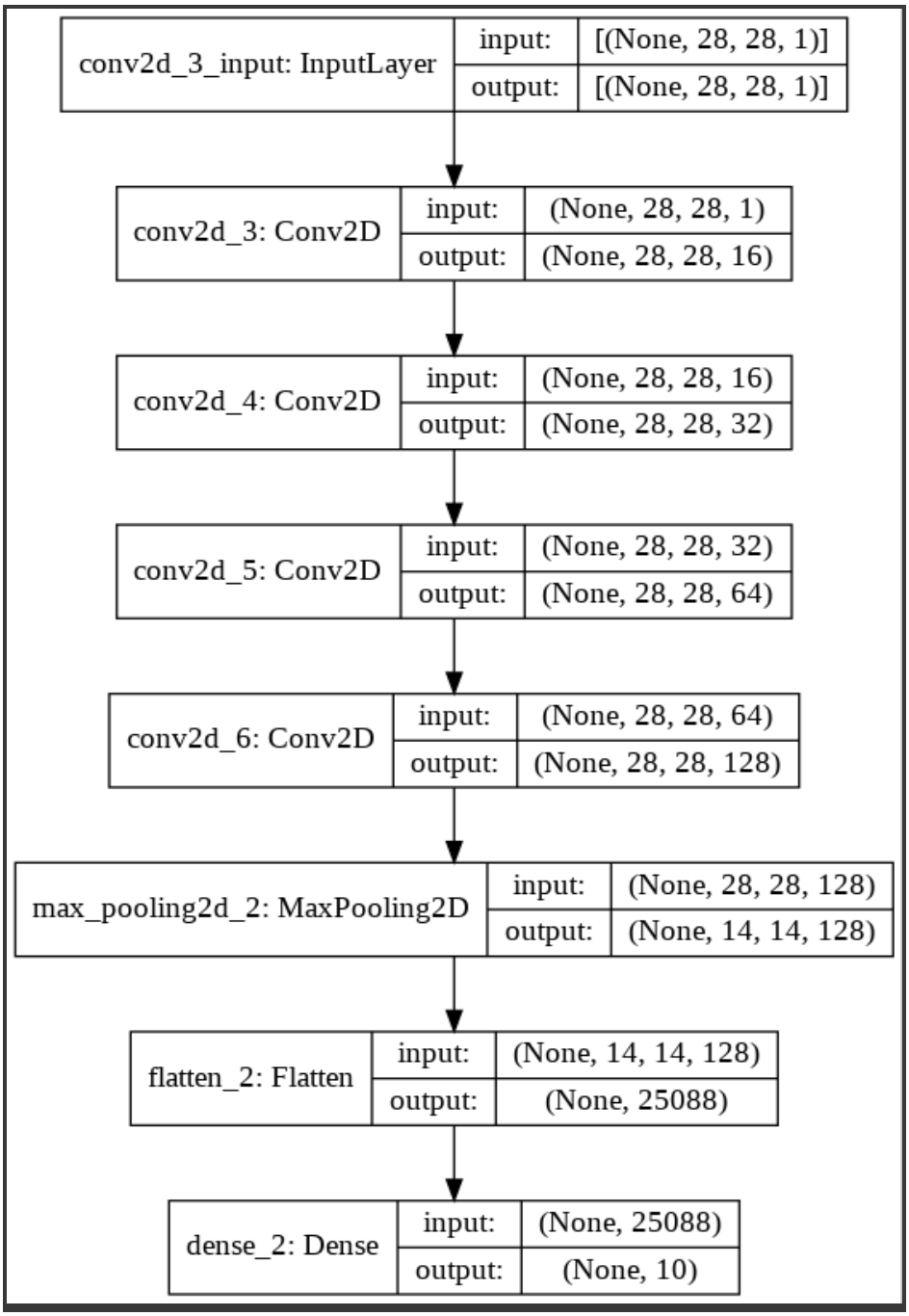
- Validation Accuracy : `0.8897`

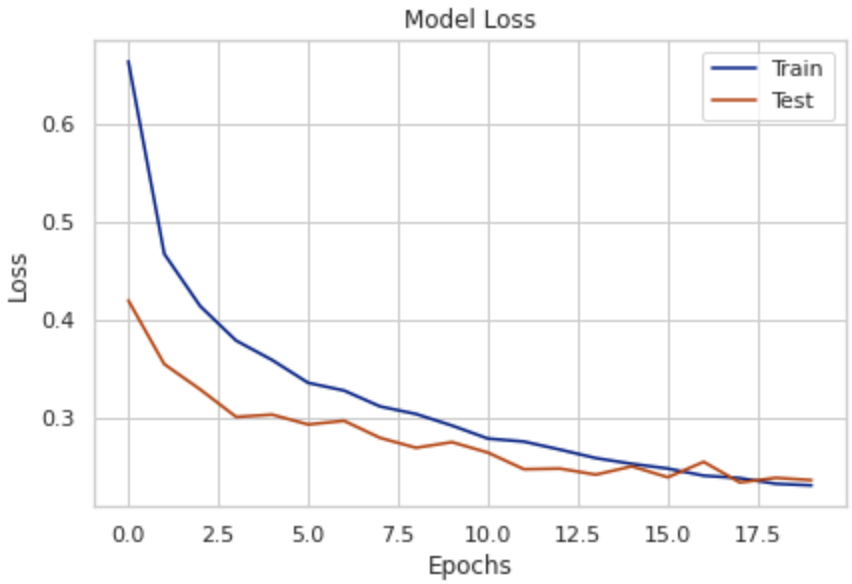


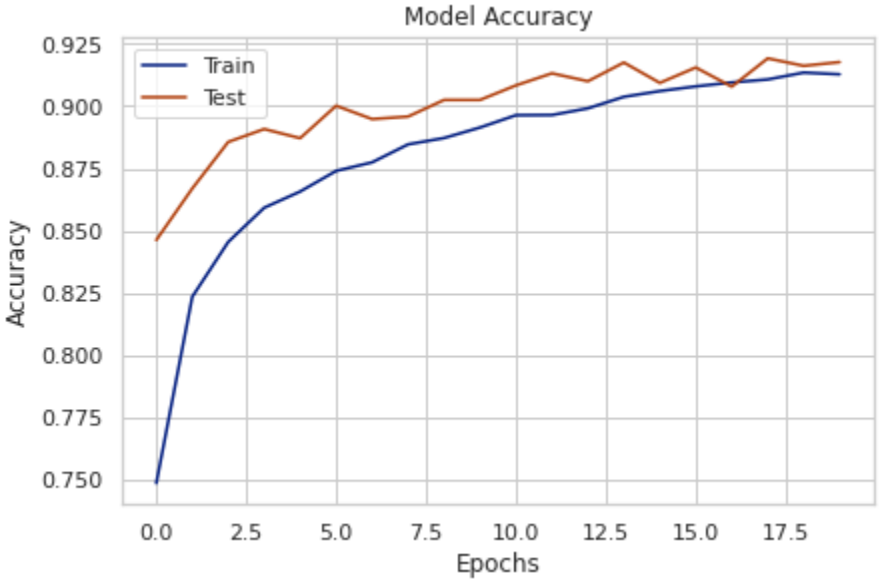




5. **MODEL 11**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 6
    - Nodes in hidden layers : 16, 32, 64, 128, 512, 128
    - Validation Loss : `0.2364`
    - Validation Accuracy : `0.9177`

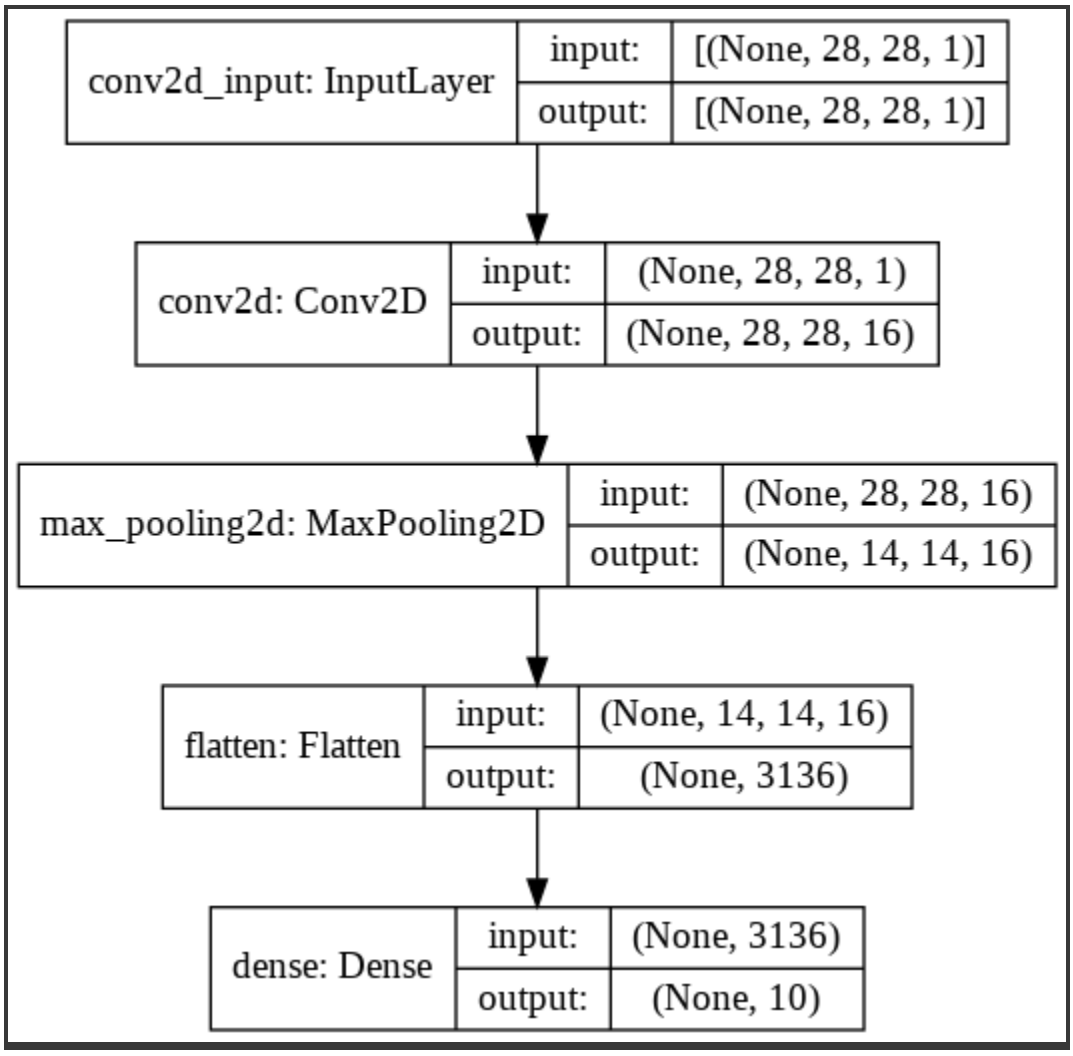

conv2d_3_input: InputLayer
input: [(None, 28, 28, 1)]
output: [(None, 28, 28, 1)]
conv2d_3: Conv2D
input: (None, 28, 28, 1)
output: (None, 28, 28, 16)
conv2d_4: Conv2D
input: (None, 28, 28, 16)
output: (None, 28, 28, 32)
conv2d_5: Conv2D
input: (None, 28, 28, 32)
output: (None, 28, 28, 64)
conv2d_6: Conv2D
input: (None, 28, 28, 64)
output: (None, 28, 28, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 28, 28, 128)
output: (None, 14, 14, 128)
flatten_2: Flatten
input: (None, 14, 14, 128)
output: (None, 25088)
dense_2: Dense
input: (None, 25088)
output: (None, 10)

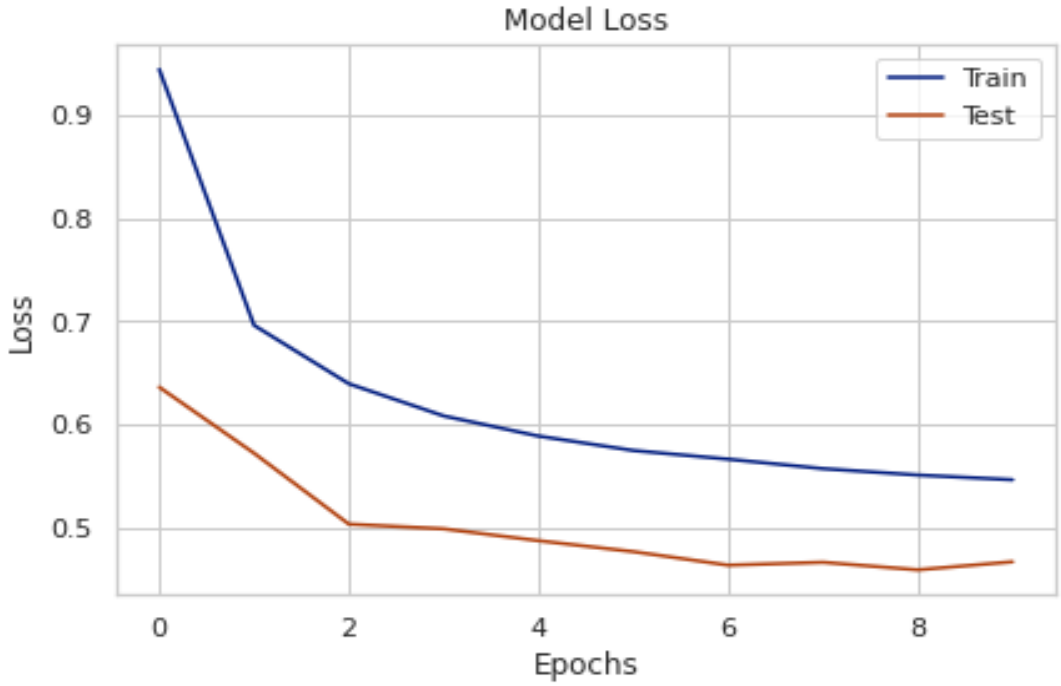

Model Loss
Train
Test
Loss
0.6
0.5
0.4
0.3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

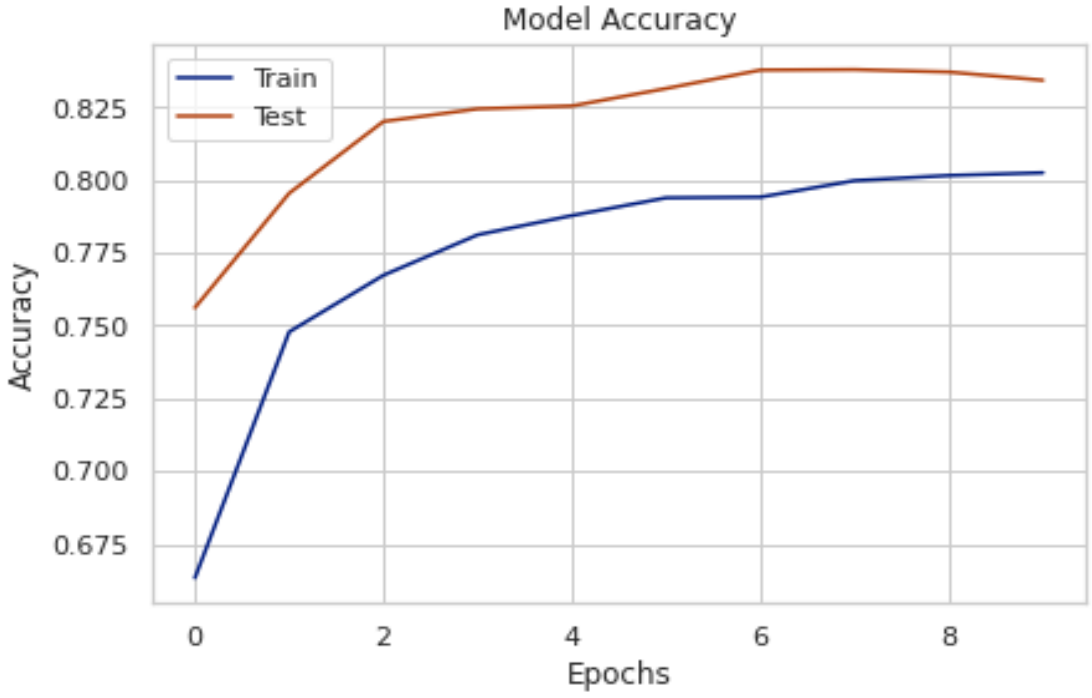

Model Accuracy
Train
Test
Accuracy
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

Description of models based on ReLU Activation function

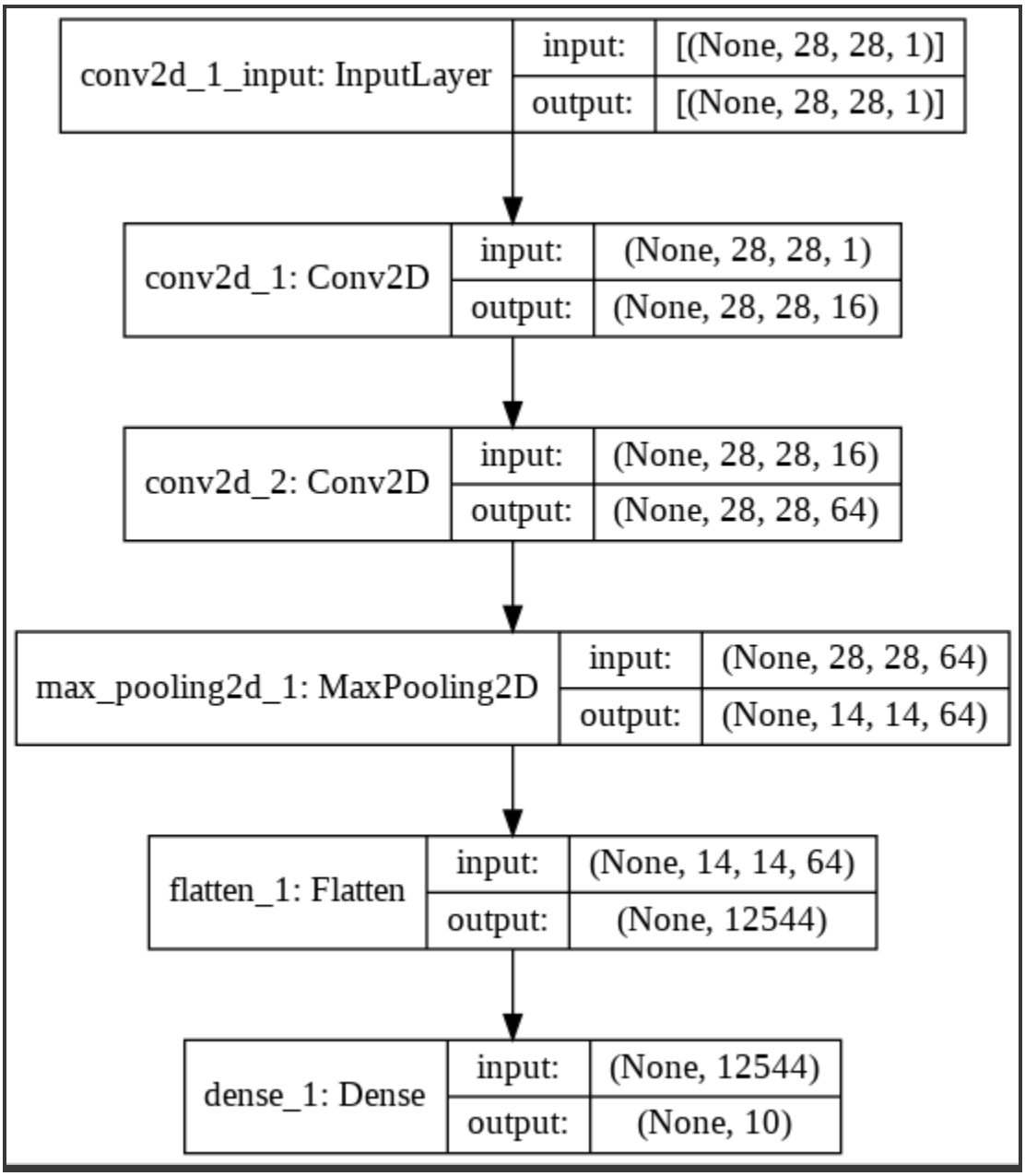
1. **MODEL 12**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 1
    - Nodes in hidden layers : 16
    - Validation Loss : `0.4661`
    - Validation Accuracy : `0.8343`

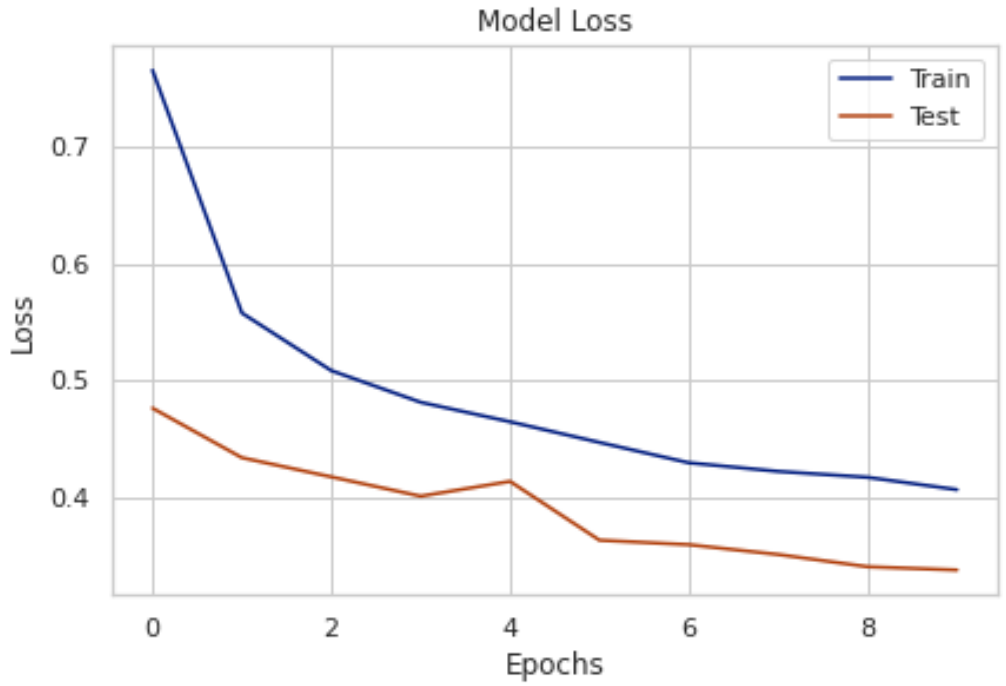


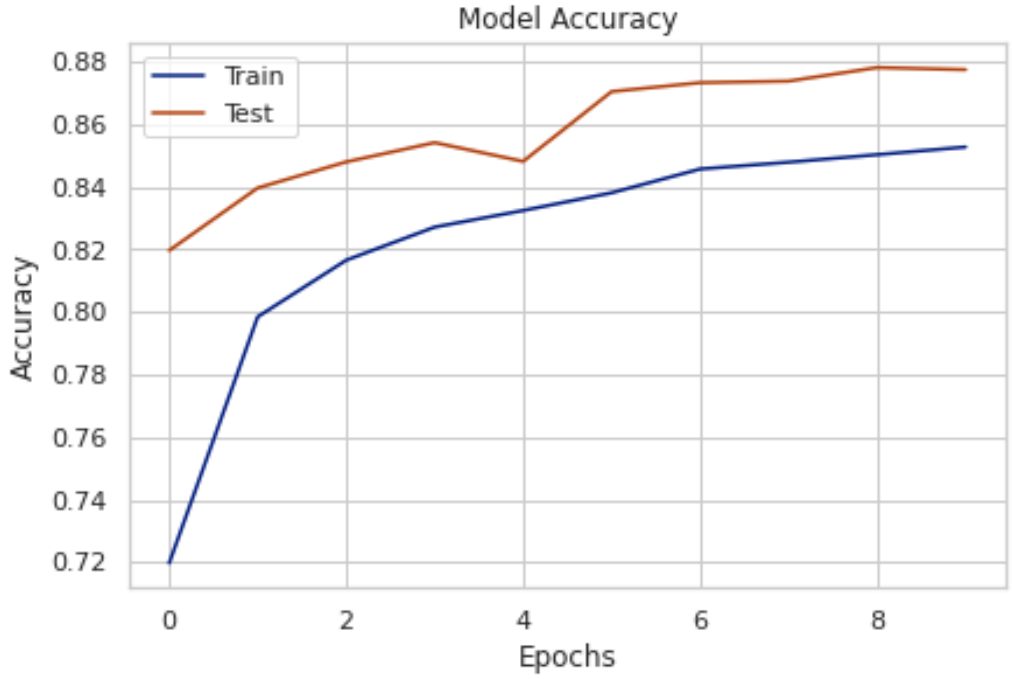

Model Loss
Train
Test
Loss
0.9
0.8
0.7
0.6
0.5
0
2
4
6
8
Epochs


Model Accuracy
Train
Test
Accuracy
0.825
0.800
0.775
0.750
0.725
0.700
0.675
0
2
4
6
8
Epochs

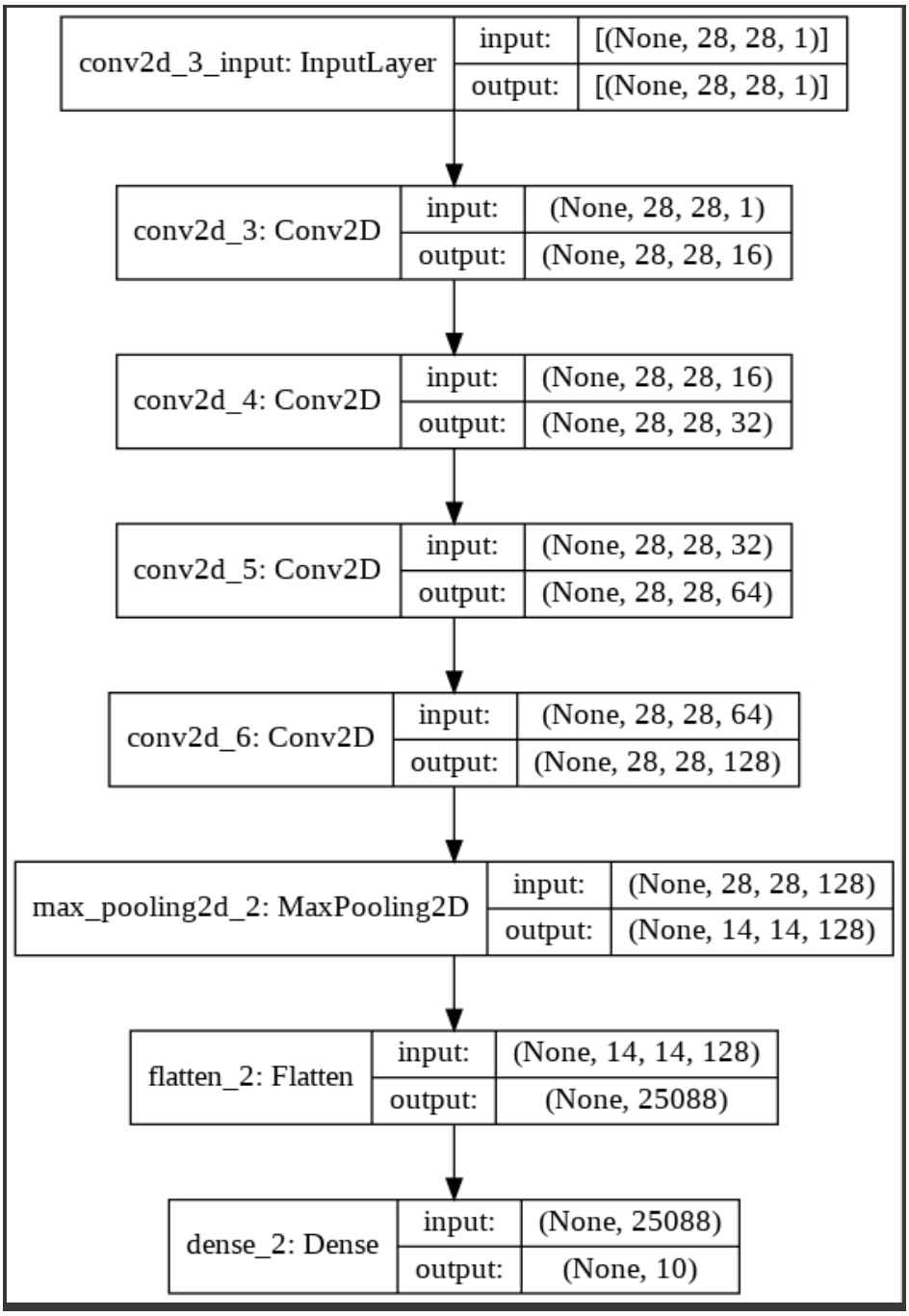
2. **MODEL 13**
    - Loss: KL divergence
    - Number of hidden layers : 2
    - Nodes in hidden layers : 16, 64
    - Validation Loss : `0.3373`
    - Validation Accuracy : `0.8775`

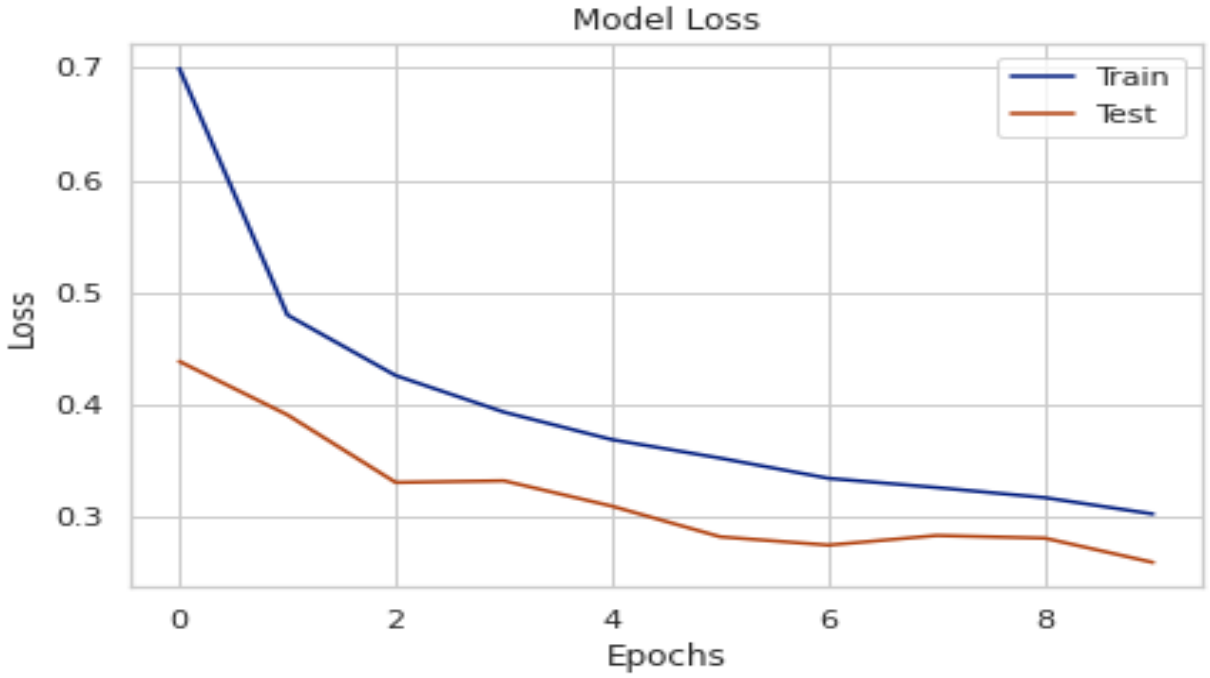


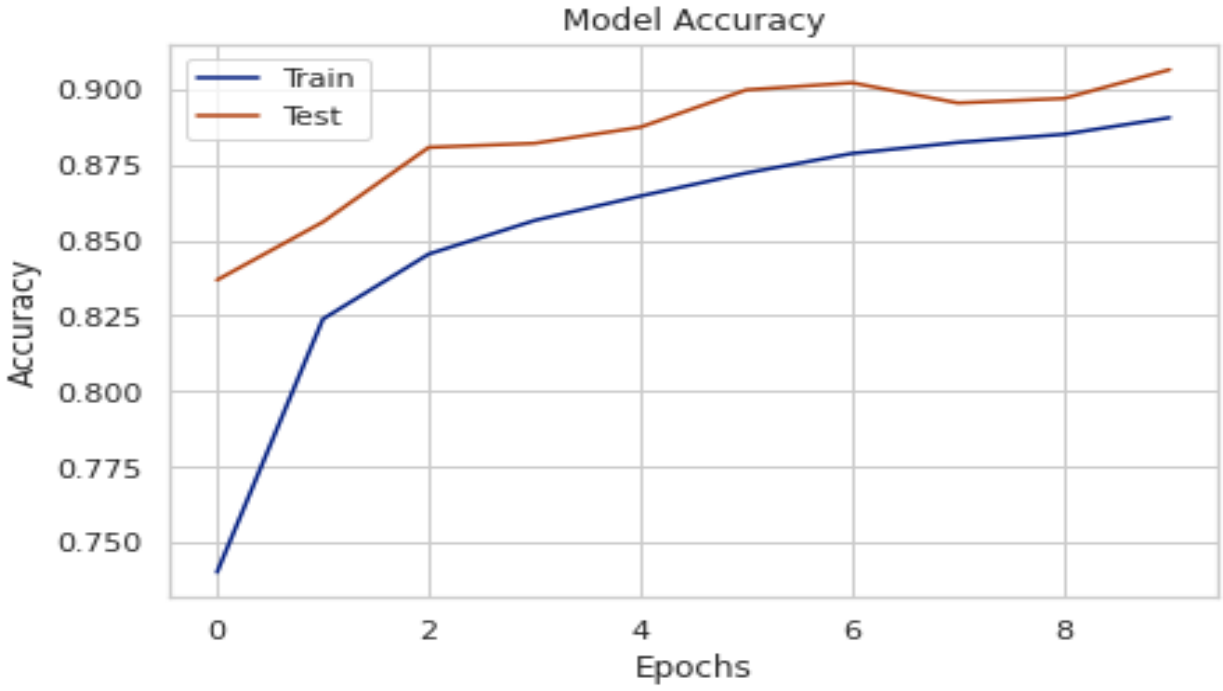

Model Loss
Train
Test
Loss
0.7
0.6
0.5
0.4
0
2
4
6
8
Epochs


Model Accuracy
Train
Test
Accuracy
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0
2
4
6
8
Epochs

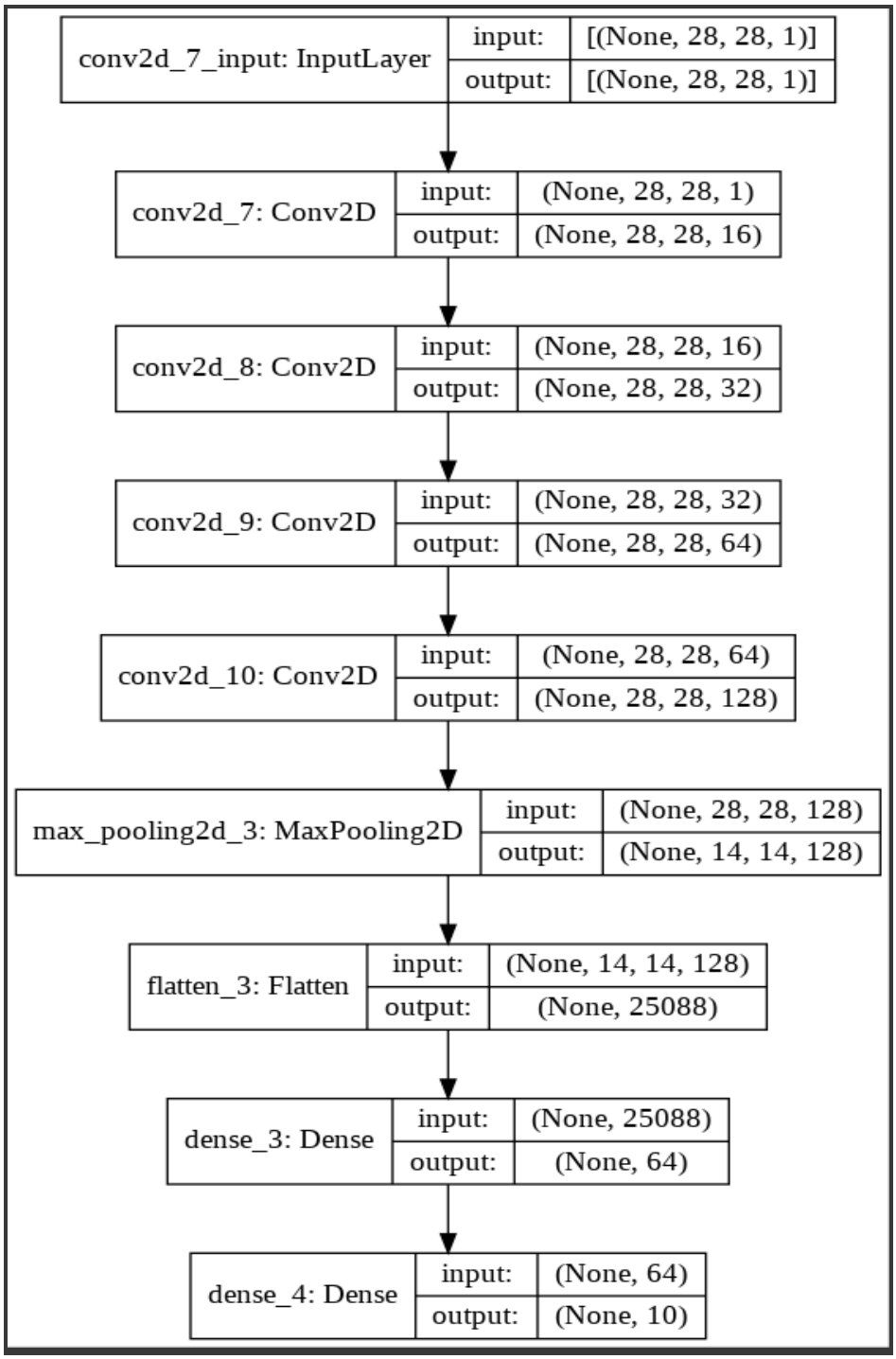
3. **MODEL 14**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 4
    - Nodes in hidden layers : 16, 32, 64, 128
    - Validation Loss : `0.2598`
    - Validation Accuracy : `0.9063`

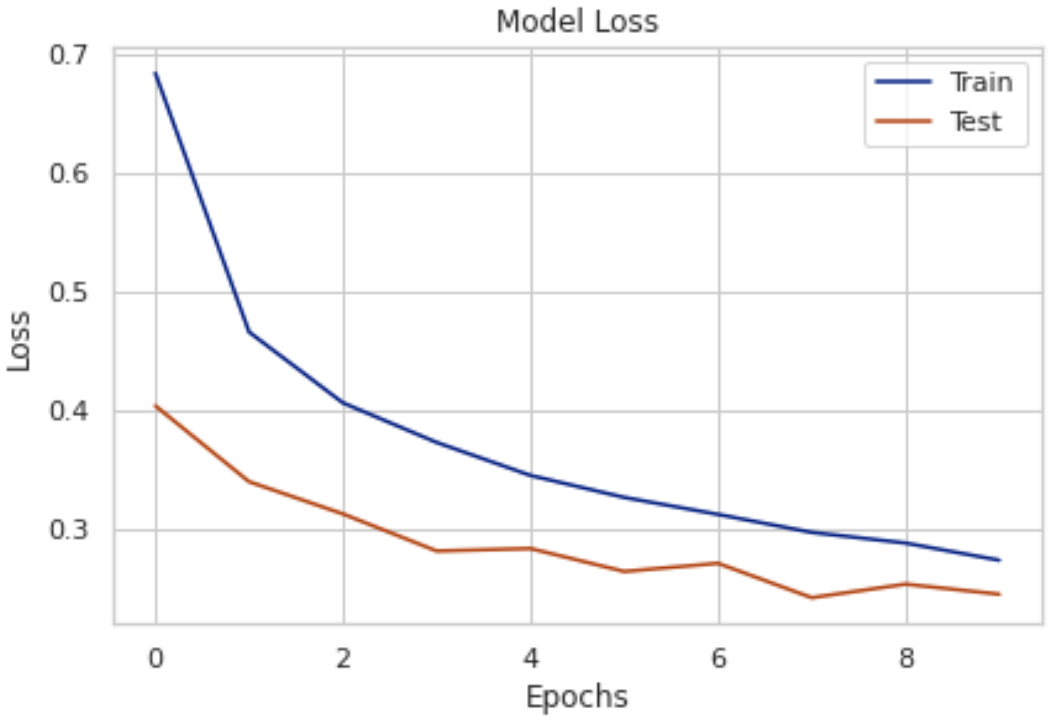


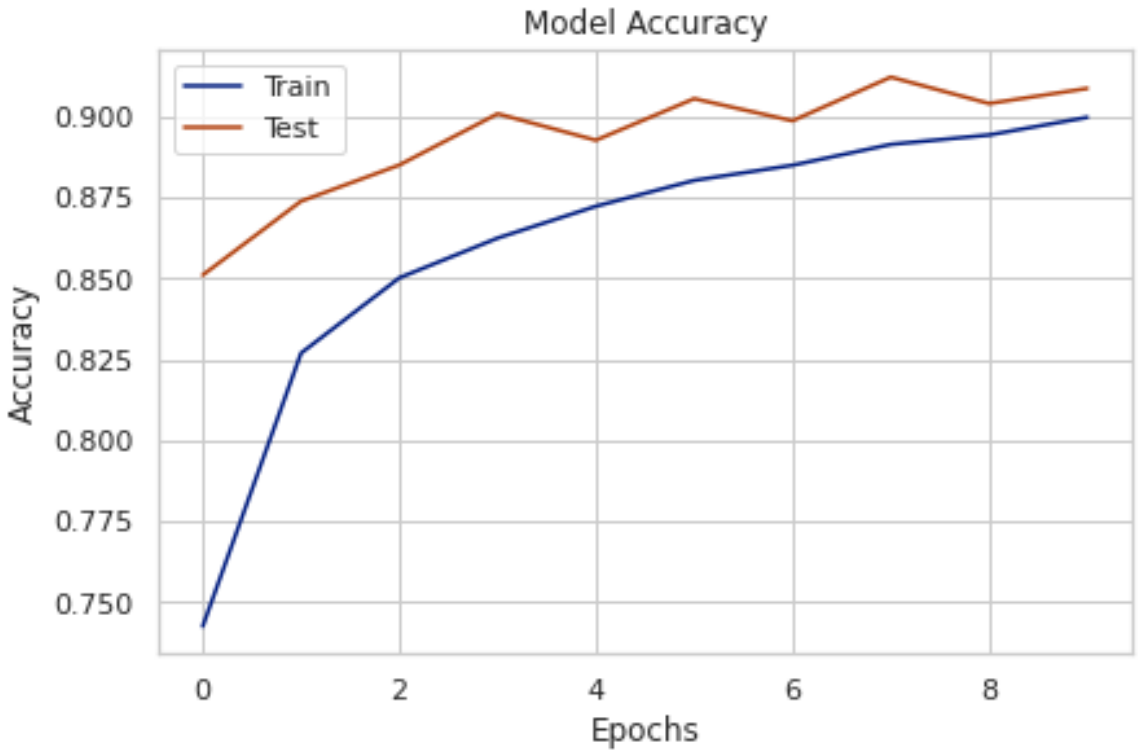

Model Loss
Train
Test
Loss
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
Epochs


Model Accuracy
Train
Test
Accuracy
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
2
4
6
8
Epochs

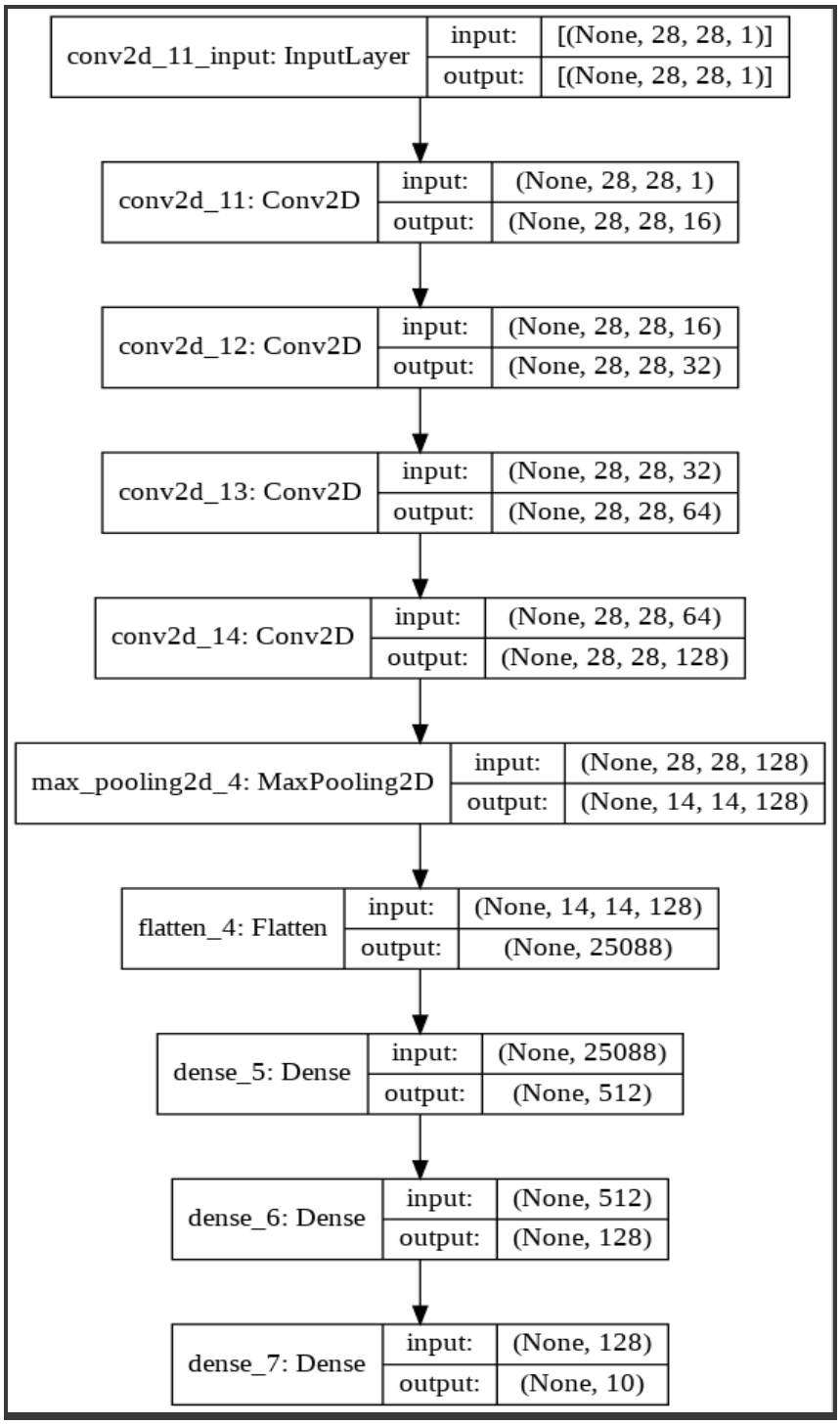
4. **MODEL 15**
    - Loss: KL divergence
    - Number of hidden layers : 5
    - Nodes in hidden layers : 16, 32, 64, 128, 64
    - Validation Loss : `0.2459`
    - Validation Accuracy : `0.9087`

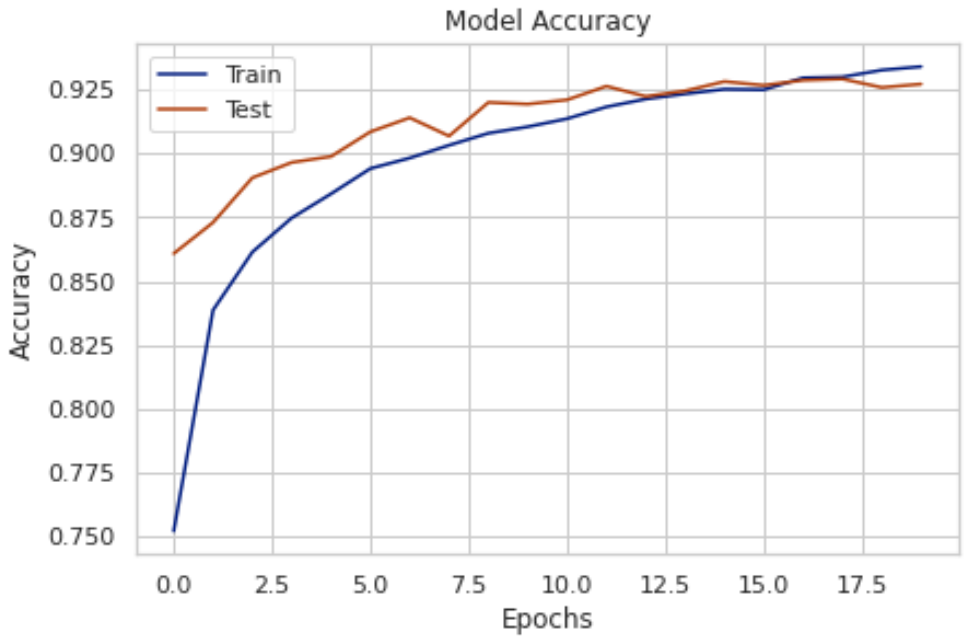


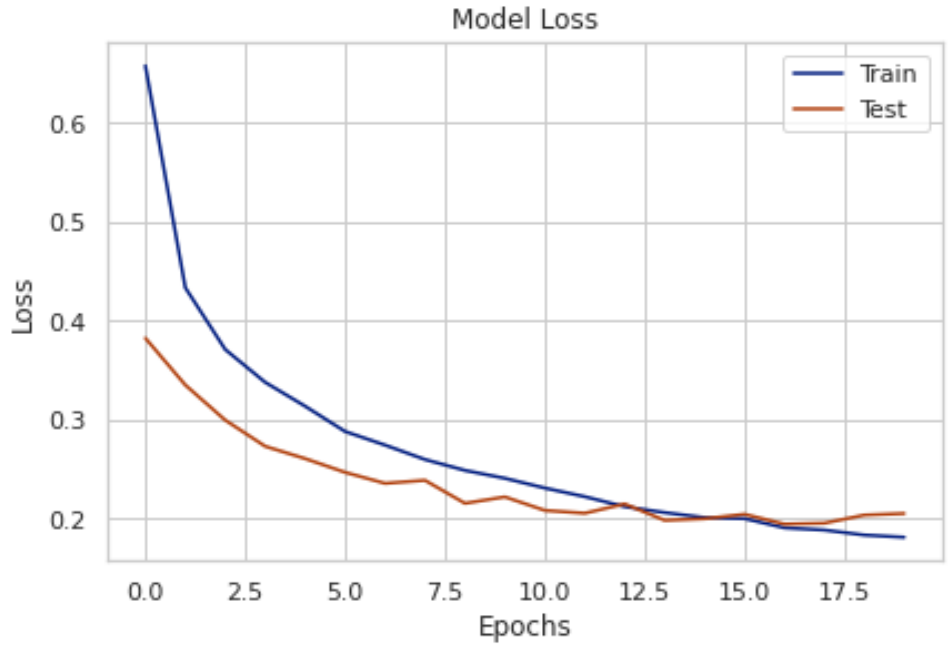

Model Loss
Train
Test
Loss
0.7
0.6
0.5
0.4
0.3
0
2
4
6
8
Epochs


Model Accuracy
Train
Test
Accuracy
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
2
4
6
8
Epochs

5. **MODEL 16**
    - Loss: Categorical cross-entropy
    - Number of hidden layers : 6
    - Nodes in hidden layers : 16, 32, 64, 128, 512, 128
    - Validation Loss : `0.2056`
    - Validation Accuracy : `0.9272`




Model Accuracy
Train
Test
Accuracy
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs


Model Loss
Train
Test
Loss
0.6
0.5
0.4
0.3
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs

Comparison between the various models based on cross-val loss and accuracy

Model	**Activation**	**Loss**	**Accuracy**
Model 1	Sigmoid	0.6129	0.7793
Model 2	Sigmoid	0.5912	0.7888
Model 3	Sigmoid	0.5714	0.7925
Model 4	Sigmoid	0.3830	0.8622
Model 5	Sigmoid	0.2304	0.8735
Model 6	Sigmoid	0.2303	0.8901
Model 7	Tanh	0.5116	0.8343
Model 8	Tanh	0.3900	0.8618
Model 9	Tanh	0.3856	0.8638
Model 10	Tanh	0.2995	0.8897
Model 11	Tanh	0.2364	0.9177
Model 12	ReLU	0.4661	0.8343
Model 13	ReLU	0.3373	0.8775
Model 14	ReLU	0.2598	0.9087
Model 15	ReLU	0.2459	0.9087
Model 16	ReLU	0.2056	0.9272

- As can be seen from the table above, keeping the Activation function constant, the cross-val loss decreases and the cross-val accuracy increases as the number of hidden layers is increased.

- For similar architectures, the ReLU activation function provides the best results as can be seen for model 16
- For similar architectures, the Sigmoid activation function provides the worst results as can be seen for model 6
- For similar architectures, the Tanh activation function provides the mediocre results as can be seen for model 11