Task 1:

The MNIST dataset consists of handwritten digits from 0 to 9.

Each image in the dataset is a 28x28 grayscale image representing a single digit.

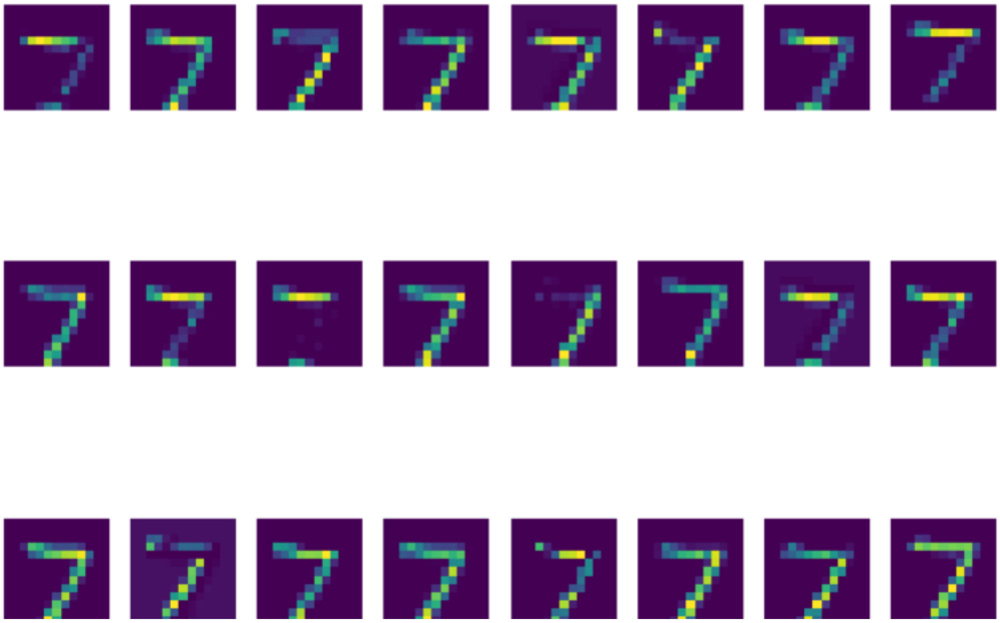
Here we defined a convolutional Neural Network on MNIST data set.

Features to visualize initial three layers are:

• Conv2D • AvgPooling2D • Conv2D

Tasks include:

- Import libraries
- Load the Data set
- Define CNN model
- Compile the Model
- Train the model
- Visualize the model



Report Analysis:

Model:

Input: 28x28 grayscale images
Convolutional Layers (CNN layers) with ReLU activation
Average Pooling Layer

Dense Layer with Softmax activation
Training Results:
Epochs:
Trained for 5 epochs
Accuracy:
Training Accuracy: ~99.78%
Validation Accuracy: ~98.96%
Feature Map Analysis:
Layer 1: Conv2D
Observations:
The first layer is used to capture simple patterns like edges and basic shapes.
Filters seem to respond to different strokes and orientations based on the basic shapes and edges.

Layer 2: AvgPooling2D

Observations:

- Helps to reduce load and enhance information flow. Decreases spatial dimension.

Layer 3: Conv2D
Observations:

- Deeper layers start capturing more complex patterns and higher-level representations.
- Filters might respond to combinations of strokes and more abstract shapes.

Normalization:

Feature maps enhance clarity and visual range. Normalizing features gives more clarity in progressive layers.

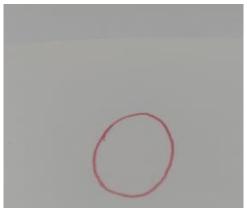
Hyperparameter Tuning:

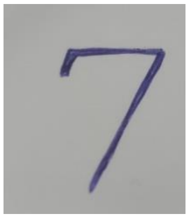
Consider experimenting with different architectures, filter sizes, or regularization techniques to improve performance of the model.

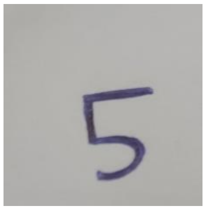
If we evaluate model we can see visualization provide insight of CNN transformation and higher level representation of input images from data set.

Task 2:

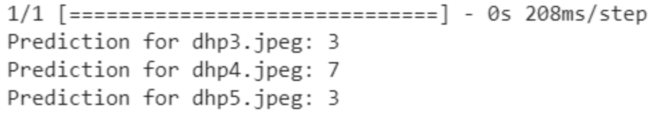
Captured Images:







Evaluation Result:



```
1/1 [==============================] - 0s 208ms/step
Prediction for dhp3.jpeg: 3
Prediction for dhp4.jpeg: 7
Prediction for dhp5.jpeg: 3
```

Steps:

- Upload Filed in collab
- Specify the path of handwritten images
- Load train model of the previous task
- Display the prediction

Analysis:
Here we can see based on the trained model this evaluation gives accurate results for hand written digit recognition.

Task 3:

Let’s experiment :

In the first experiment we have taken 10 epochs with learning rate 0.01 and number of features map as 64

Experiment 1 (10 epochs, learning rate=0.01, num_feature_maps=64):
Training:
Accuracy: ~99.04%
Loss: ~0.0376
Validation:
Accuracy: ~97.88%
Loss: ~0.1221

Experiment 2 we have taken 5 epochs, learning rate 0.001 and feature map 32

Experiment 2 (5 epochs, learning rate=0.001, num_feature_maps=32):
Training:
Accuracy: ~99.31%
Loss: ~0.0211
Validation:
Accuracy: ~98.88%
Loss: ~0.0355

Analysis:

Learning Rate:
Experiment 1 where the learning rate is higher (0.01). It is converged faster but may have overshot the optimal weights and the accuracy is low due to the overshot.

Experiment 2 with a lower learning rate (0.001) converged more steadily . Experiment 2 shows slightly higher accuracy.

Epochs:
Experimenting with 10 epochs might lower the accuracy where experiment 2 with 5 epochs shows higher balance and higher accuracy and validation.

Number of Feature Maps:

Experiment 1 with more feature maps (number of feature maps 64) have increased the model's capacity to capture complex patterns but it can be resulted in overfitting

Experiment 2 with fewer feature maps (number of feature maps 32) still achieved high accuracy which is more robust and provides higher accuracy.

Comment:
Experiment 2 provides better convergence as we used lower learning rate.
Experiment 2 mitigates the overfitting issue in experiment 1 using lower learning rate and higher epochs.

Task 4:
Here we modified the CNN model with CIFAR-100 with BatchNorm, Different Filter Sizes, and Residual Connections.

For this experiment I have used 10 epochs and added convolutional layers for model modification.

```
Epoch 10/10
      1563/1563 [==============================] - 19s 12ms/step - loss:
      1.7626 - accuracy: 0.5108 - val_loss: 1.9403 - val_accuracy: 0.4740

      40%+ accuracy is achieved.
```

The modified CIFAR-100 model has been trained for 10 epochs. Here's a summary of the training results:

Training Accuracy: 51.08%
Validation Accuracy: 47.40%

Plot:

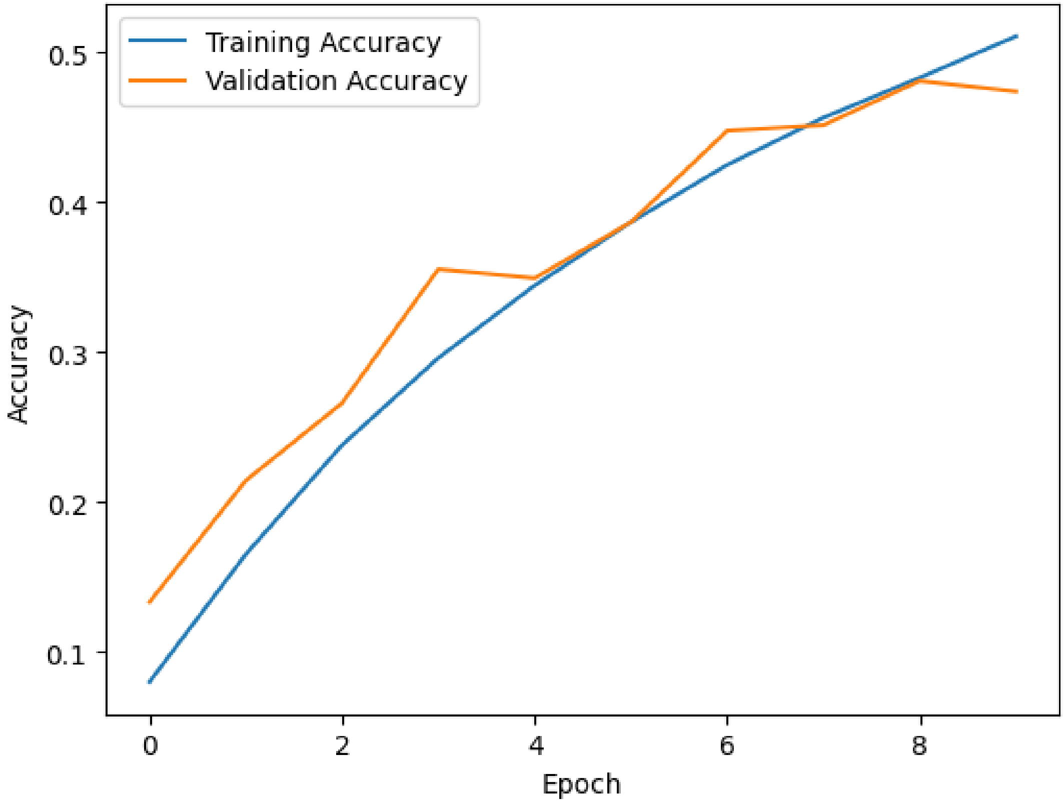
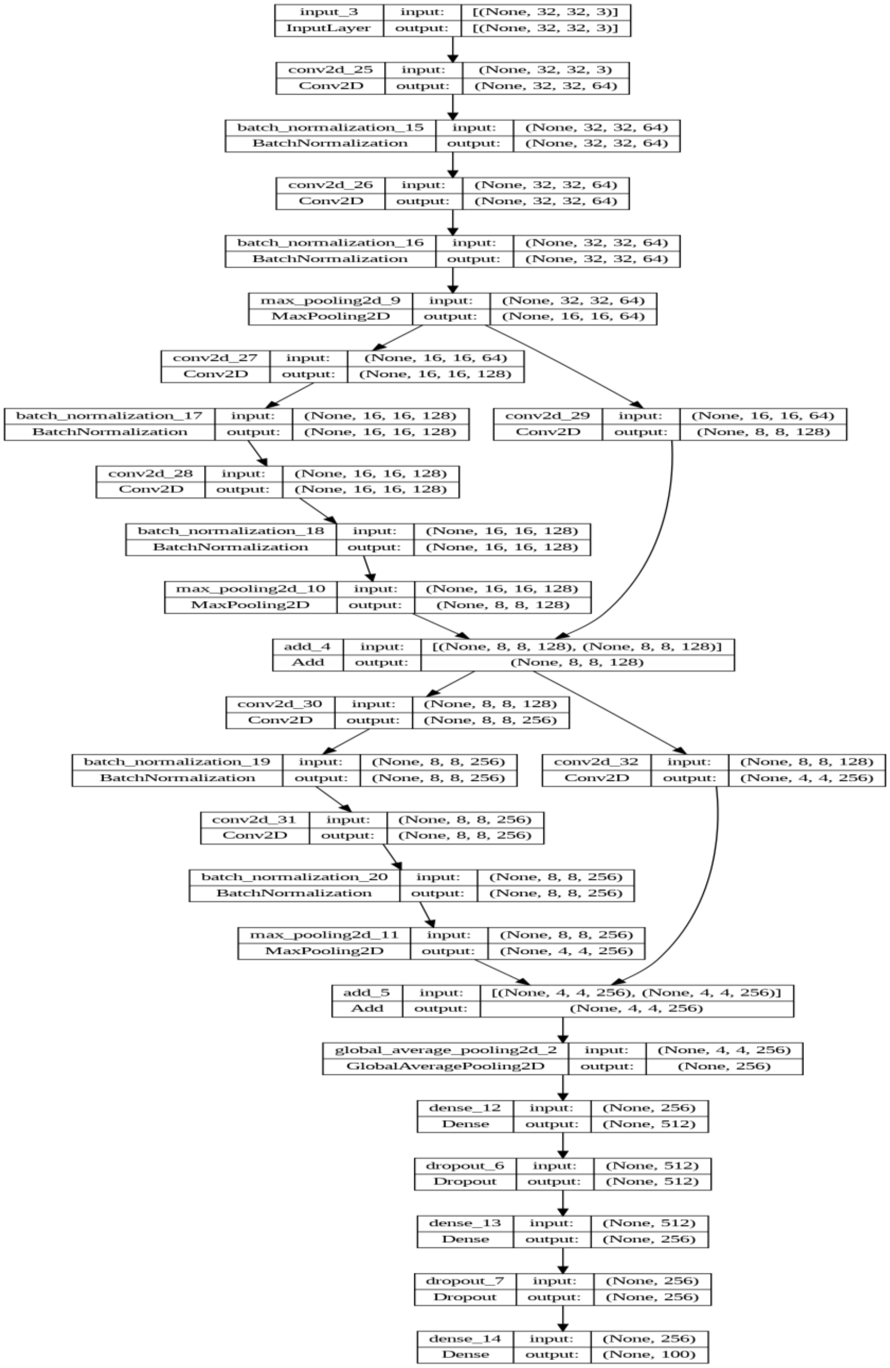


Diagram:


input_3 | input: | [(None, 32, 32, 3)]
InputLayer | output: | [(None, 32, 32, 3)]
conv2d_25 | input: | (None, 32, 32, 3)
Conv2D | output: | (None, 32, 32, 64)
batch_normalization_15 | input: | (None, 32, 32, 64)
BatchNormalization | output: | (None, 32, 32, 64)
conv2d_26 | input: | (None, 32, 32, 64)
Conv2D | output: | (None, 32, 32, 64)
batch_normalization_16 | input: | (None, 32, 32, 64)
BatchNormalization | output: | (None, 32, 32, 64)
max_pooling2d_9 | input: | (None, 32, 32, 64)
MaxPooling2D | output: | (None, 16, 16, 64)
conv2d_27 | input: | (None, 16, 16, 64)
Conv2D | output: | (None, 16, 16, 128)
batch_normalization_17 | input: | (None, 16, 16, 128)
BatchNormalization | output: | (None, 16, 16, 128)
conv2d_29 | input: | (None, 16, 16, 64)
Conv2D | output: | (None, 8, 8, 128)
conv2d_28 | input: | (None, 16, 16, 128)
Conv2D | output: | (None, 16, 16, 128)
batch_normalization_18 | input: | (None, 16, 16, 128)
BatchNormalization | output: | (None, 16, 16, 128)
max_pooling2d_10 | input: | (None, 16, 16, 128)
MaxPooling2D | output: | (None, 8, 8, 128)
add_4 | input: | [(None, 8, 8, 128), (None, 8, 8, 128)]
Add | output: | (None, 8, 8, 128)
conv2d_30 | input: | (None, 8, 8, 128)
Conv2D | output: | (None, 8, 8, 256)
batch_normalization_19 | input: | (None, 8, 8, 256)
BatchNormalization | output: | (None, 8, 8, 256)
conv2d_32 | input: | (None, 8, 8, 128)
Conv2D | output: | (None, 4, 4, 256)
conv2d_31 | input: | (None, 8, 8, 256)
Conv2D | output: | (None, 8, 8, 256)
batch_normalization_20 | input: | (None, 8, 8, 256)
BatchNormalization | output: | (None, 8, 8, 256)
max_pooling2d_11 | input: | (None, 8, 8, 256)
MaxPooling2D | output: | (None, 4, 4, 256)
add_5 | input: | [(None, 4, 4, 256), (None, 4, 4, 256)]
Add | output: | (None, 4, 4, 256)
global_average_pooling2d_2 | input: | (None, 4, 4, 256)
GlobalAveragePooling2D | output: | (None, 256)
dense_12 | input: | (None, 256)
Dense | output: | (None, 512)
dropout_6 | input: | (None, 512)
Dropout | output: | (None, 512)
dense_13 | input: | (None, 512)
Dense | output: | (None, 256)
dropout_7 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_14 | input: | (None, 256)
Dense | output: | (None, 100)

As the accuracy of the test is more than 40% thus the model is successfully done.