Jae Yoon; Jagadeesh Meesala

Neural Networks - Image Recognition

Priniciples Of Machine Learning

2021

Abstract

Deep Learning uses different type of neural network architectures like object recognition, image classification and object detection etc.

The goal of this project is to apply those different neural network architectures on MNIST and POSE datasets.

1 Introduction

Deep learning project to build a hand written digit recognition app using MNIST dataset, using of CNN etc.

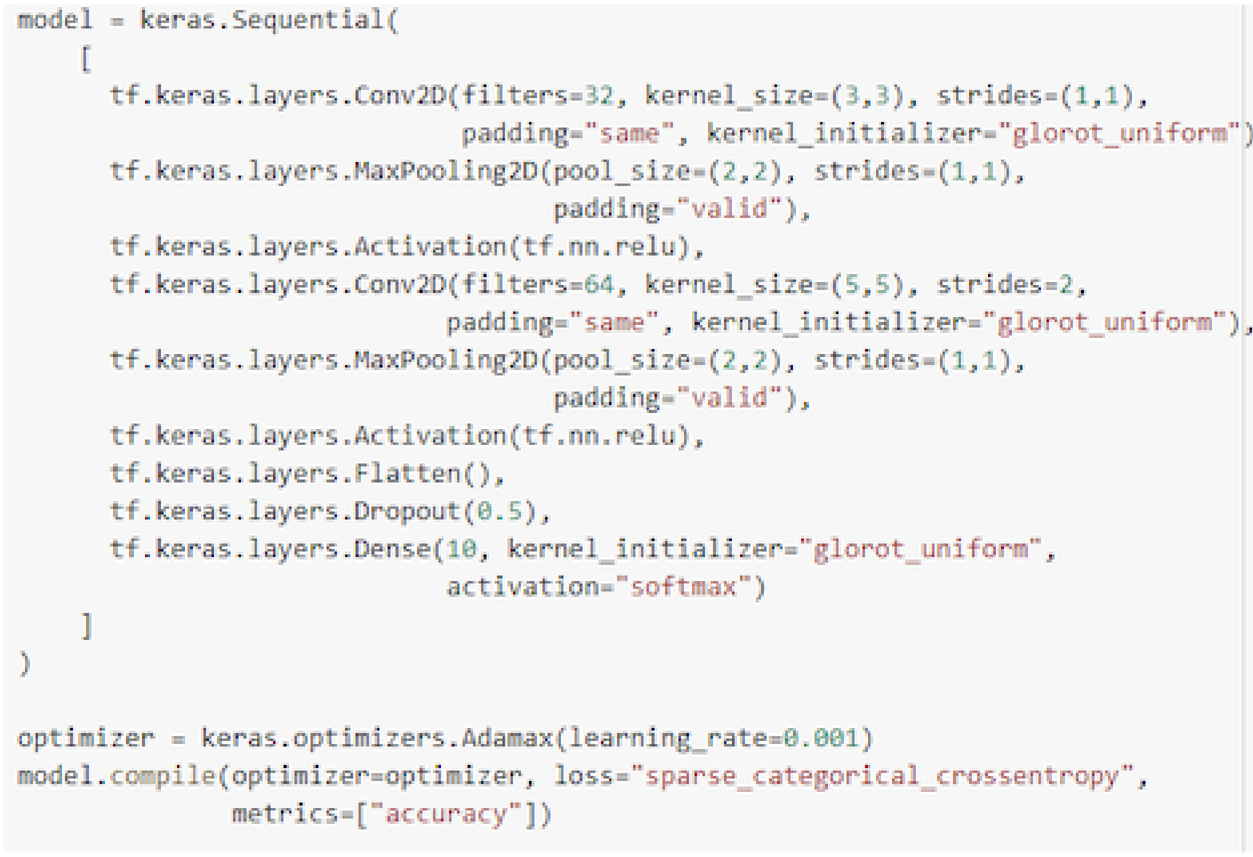
2 Results

2.1 Experiments with MNIST.mat

Preprocess

The dimension of the training data is (60000, 28, 28). As many machine learning algorithms cannot operate on label data directly, we have used one-hot code technique to convert the input and output variable to be numeric.

Architecture

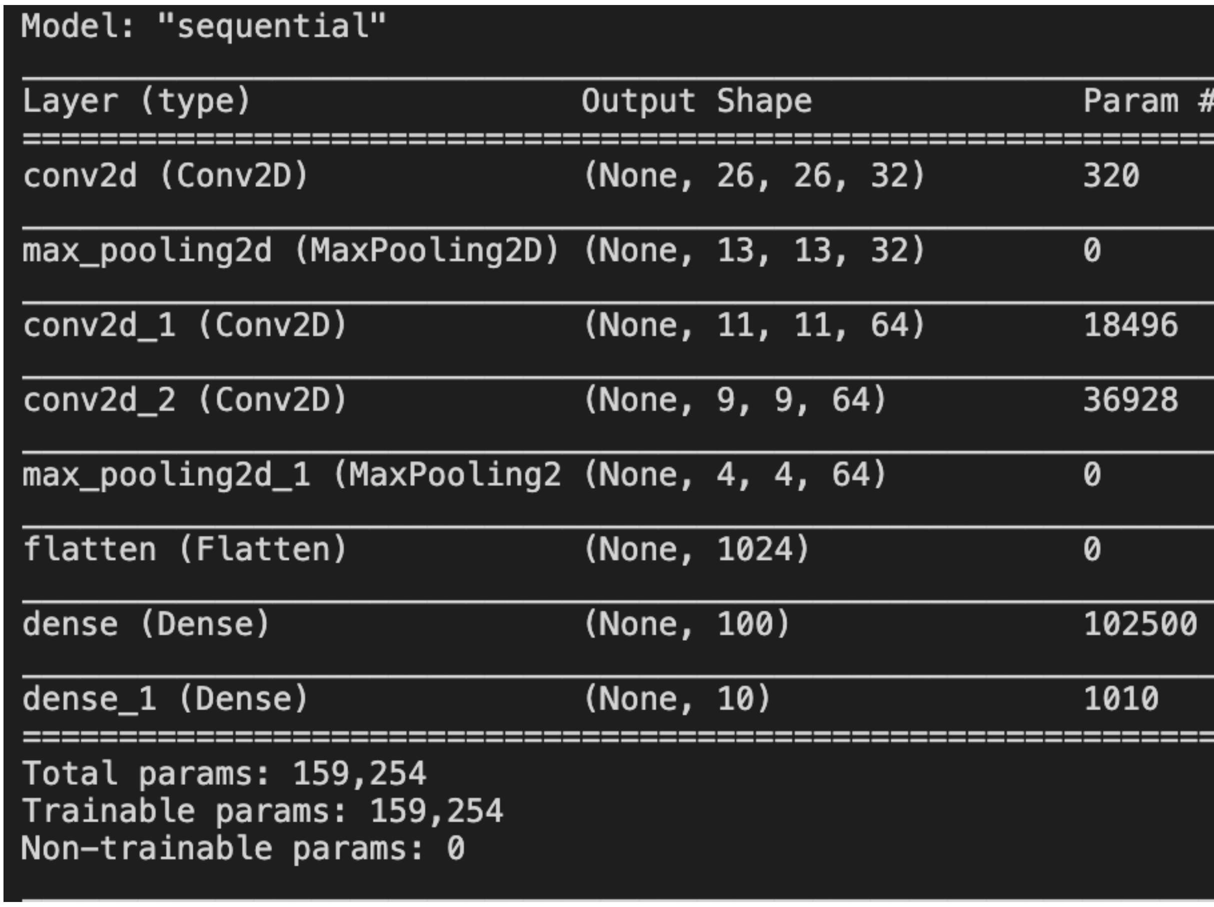


```
model = keras.Sequential(
    [
      tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), strides=(1,1),
                             padding="same", kernel_initializer="glorot_uniform")
      tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(1,1),
                                   padding="valid"),
      tf.keras.layers.Activation(tf.nn.relu),
      tf.keras.layers.Conv2D(filters=64, kernel_size=(5,5), strides=2,
                            padding="same", kernel_initializer="glorot_uniform"),
      tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(1,1),
                                   padding="valid"),
      tf.keras.layers.Activation(tf.nn.relu),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dropout(0.5),
      tf.keras.layers.Dense(10, kernel_initializer="glorot_uniform",
                            activation="softmax")
    ]
)

optimizer = keras.optimizers.Adamax(learning_rate=0.001)
model.compile(optimizer=optimizer, loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])
```

MNIST architecture

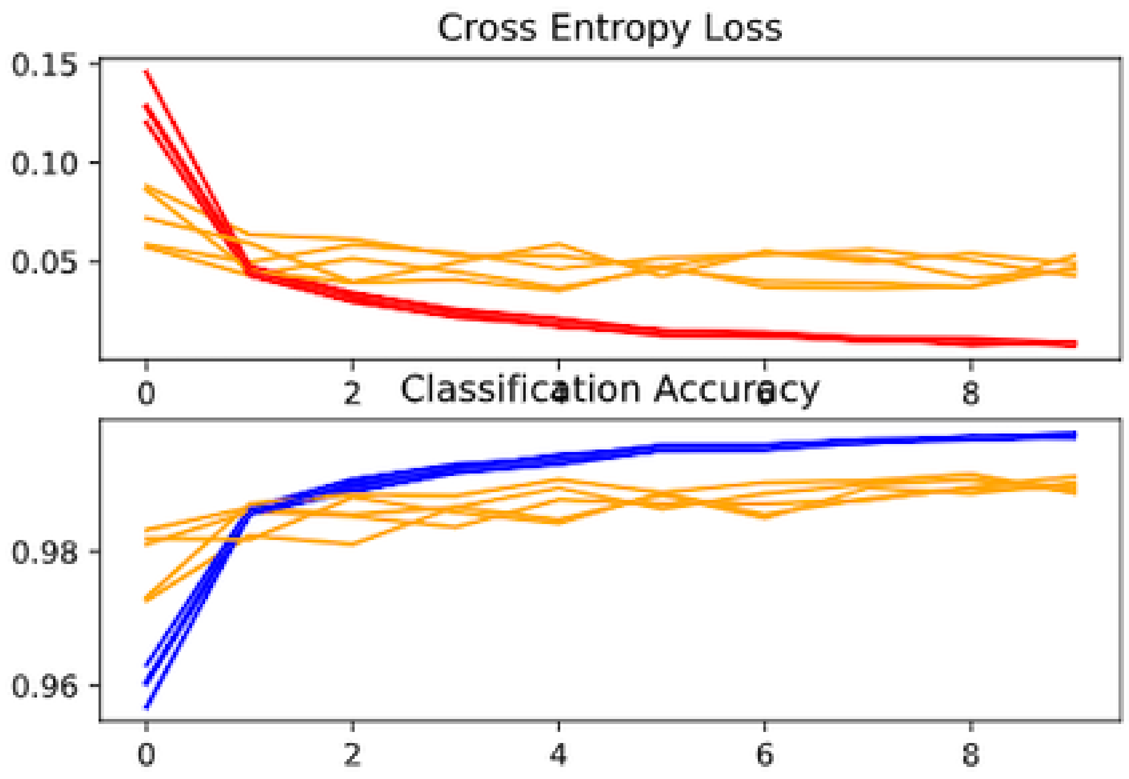
Model

We have tried and tested different combination of layers and found the following work the best.

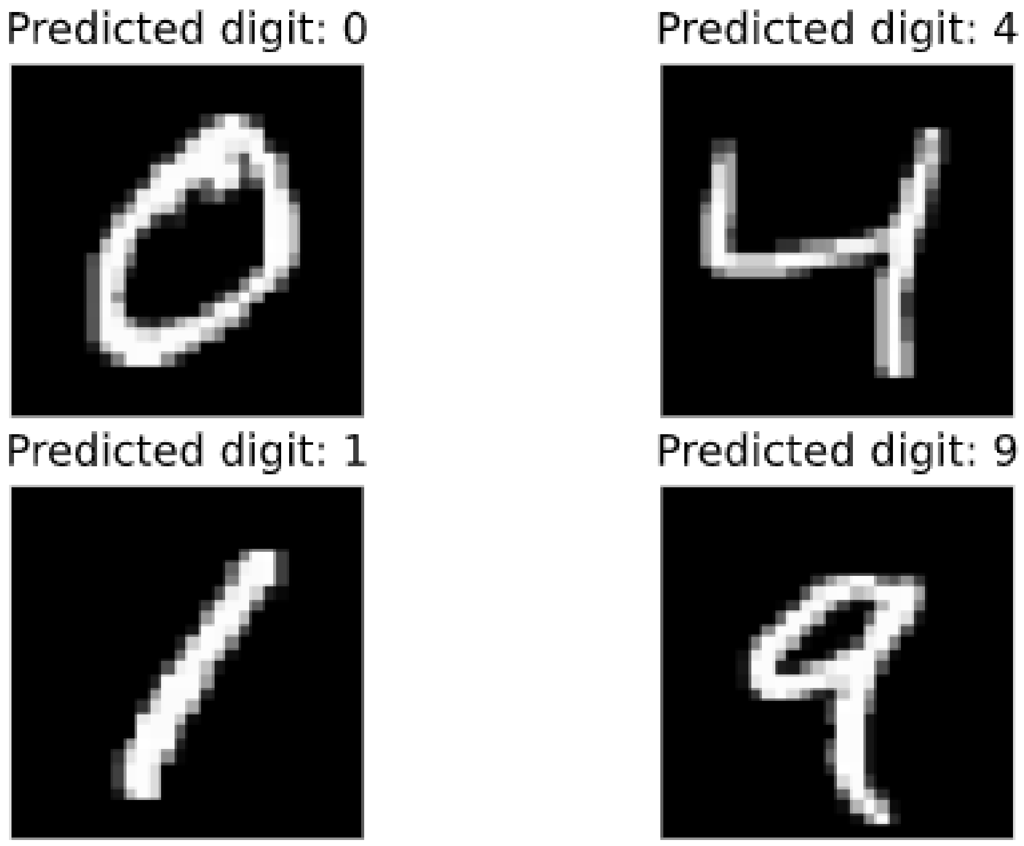


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 9, 9, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 100)               102500
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1010
=================================================================
Total params: 159,254
Trainable params: 159,254
Non-trainable params: 0
_________________________________________________________________
```

Model Summary



Cross Entropy Loss and Accuracy



Prediction of the dataset images

2.2 Experiments with pose.mat

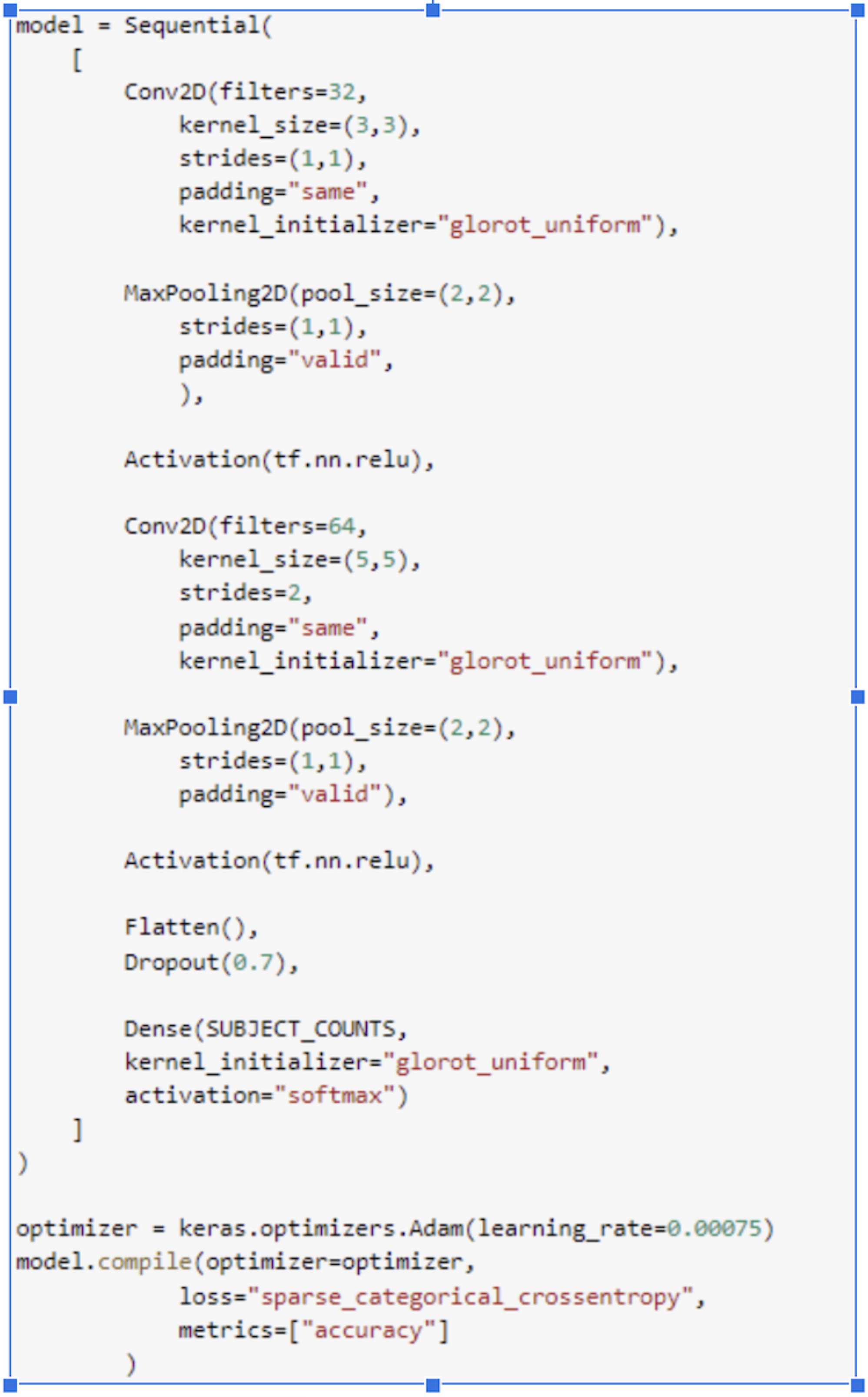
(First) 10 images per subject are used for training and (last) 3 for testing.

Architecture

The training accuracy was way higher than validation though, and we could not do much about it, with this much data models are bound to overfit, however, we did try adding Dropout. Another technique would be data augmentation which is cumbersome to implement.

Transfer learning did not improve the performance either. We had to manipulate the image array for that to make it 3-channeled but even then it was almost the same as our normal model.

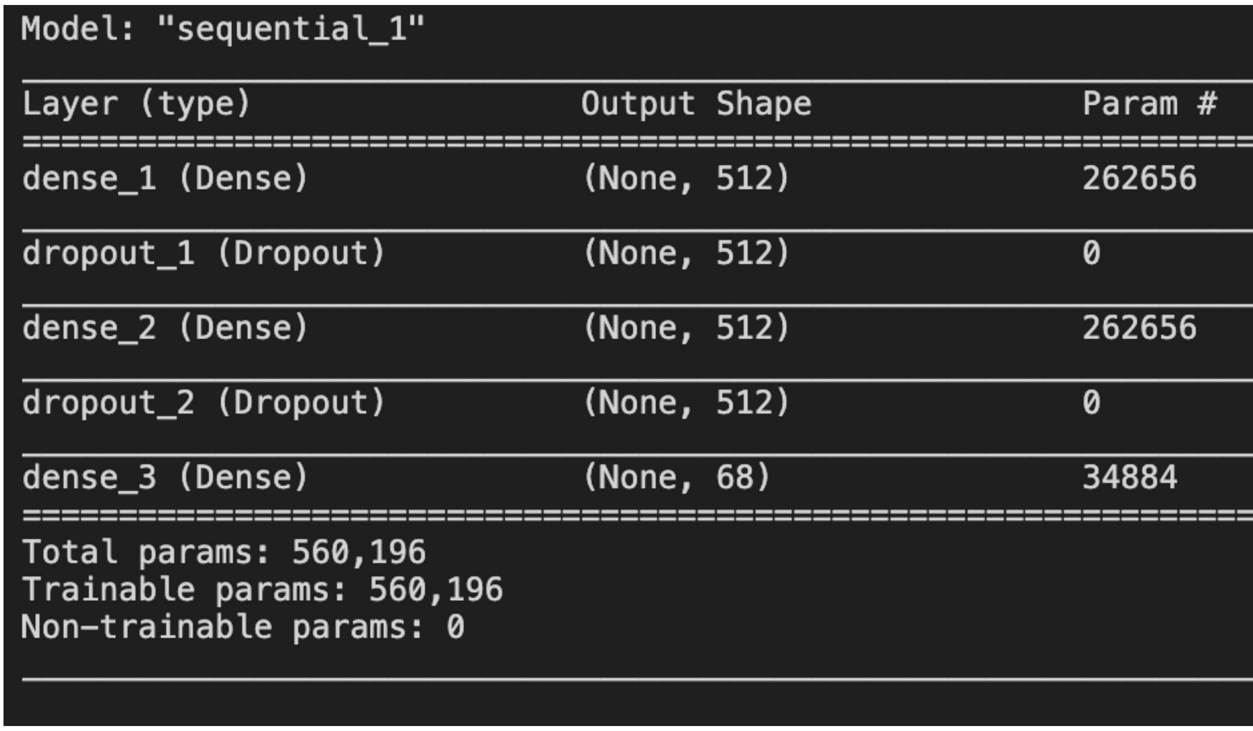
Due to the small dataset, the model overfits the data quite easily. To solve this problem and reduce it, we included Dropout and also reduced the deepness of the architecture. The most challenging hyperparameter to set was the learning rate. We believe 0.0007 to be the most optimized for our use.



```
model = Sequential(
    [
        Conv2D(filters=32,
            kernel_size=(3,3),
            strides=(1,1),
            padding="same",
            kernel_initializer="glorot_uniform"),

        MaxPooling2D(pool_size=(2,2),
            strides=(1,1),
            padding="valid",
            ),

        Activation(tf.nn.relu),

        Conv2D(filters=64,
            kernel_size=(5,5),
            strides=2,
            padding="same",
            kernel_initializer="glorot_uniform"),

        MaxPooling2D(pool_size=(2,2),
            strides=(1,1),
            padding="valid"),

        Activation(tf.nn.relu),

        Flatten(),
        Dropout(0.7),

        Dense(SUBJECT_COUNTS,
        kernel_initializer="glorot_uniform",
        activation="softmax")
    ]
)

optimizer = keras.optimizers.Adam(learning_rate=0.00075)
model.compile(optimizer=optimizer,
            loss="sparse_categorical_crossentropy",
            metrics=["accuracy"]
        )
```

Architecture

Model



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 68)                34884
=================================================================
Total params: 560,196
Trainable params: 560,196
Non-trainable params: 0
_________________________________________________________________
```

Model Summary

3 Code Files

The following notebook files are used for pre-processing and classification application:

- Jagadeesh Meesala
  - github link
- Jae Yoon
  - codelabs-code

4 References

[**tensorflow tutorial**] https://www.tensorflow.org/api_docs/python/tf/keras/layers/ [**keras documentation**] https://keras.io/guides/sequential_model/ [**CNN MNIST**] https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/ [**sci-kit Bayes**] https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html [**ResNet CaseStudies**] https://youtu.be/ZILIbUvp5lk