Applied Machine Learning Mini Project

Student Name: Aditya Sanjay Mhaske

Topic: Object Detection

Problem Statement:

- The training set contains 100 categories with labels from 0 to 99. Each category has 5000 images. Each image is formatted as 28 ×28 pixels. There are 500K images in total. You can download the dataset from https://bit.ly/p556bonus.
- The test set covers the same categories in the training set without any unseen categories. The test set has 100K images. We didn't provide the labels of the test set.

1. Methodology (Formulation of the problem)

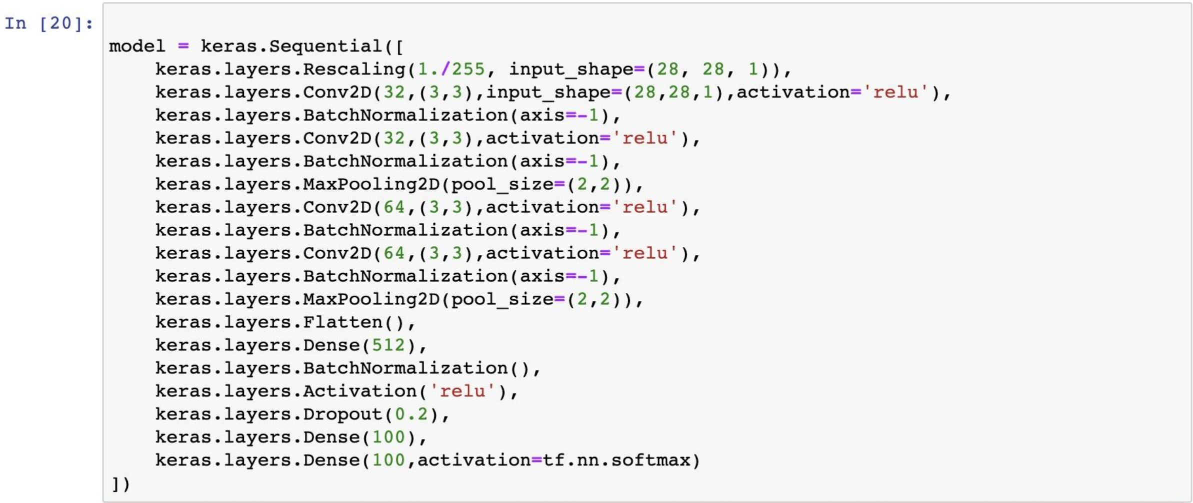
Technologies used:
Keras, Tensorflow, and implemented CNN

The Convolutional Neural Network:
(CNN or ConvNet) is a subtype of Neural Network that is mainly used for applications in image recognition. Its built-in convolutional layer reduces the high dimensionality of images without losing their information. That is why CNN's are especially suited for this use case.

2. Data Preprocessing

Rescaled images to 28 x 28 x 1
‘ ‘ ‘
keras.layers.Rescaling(1./255, input_shape=(28, 28, 1)),
‘ ‘ ‘

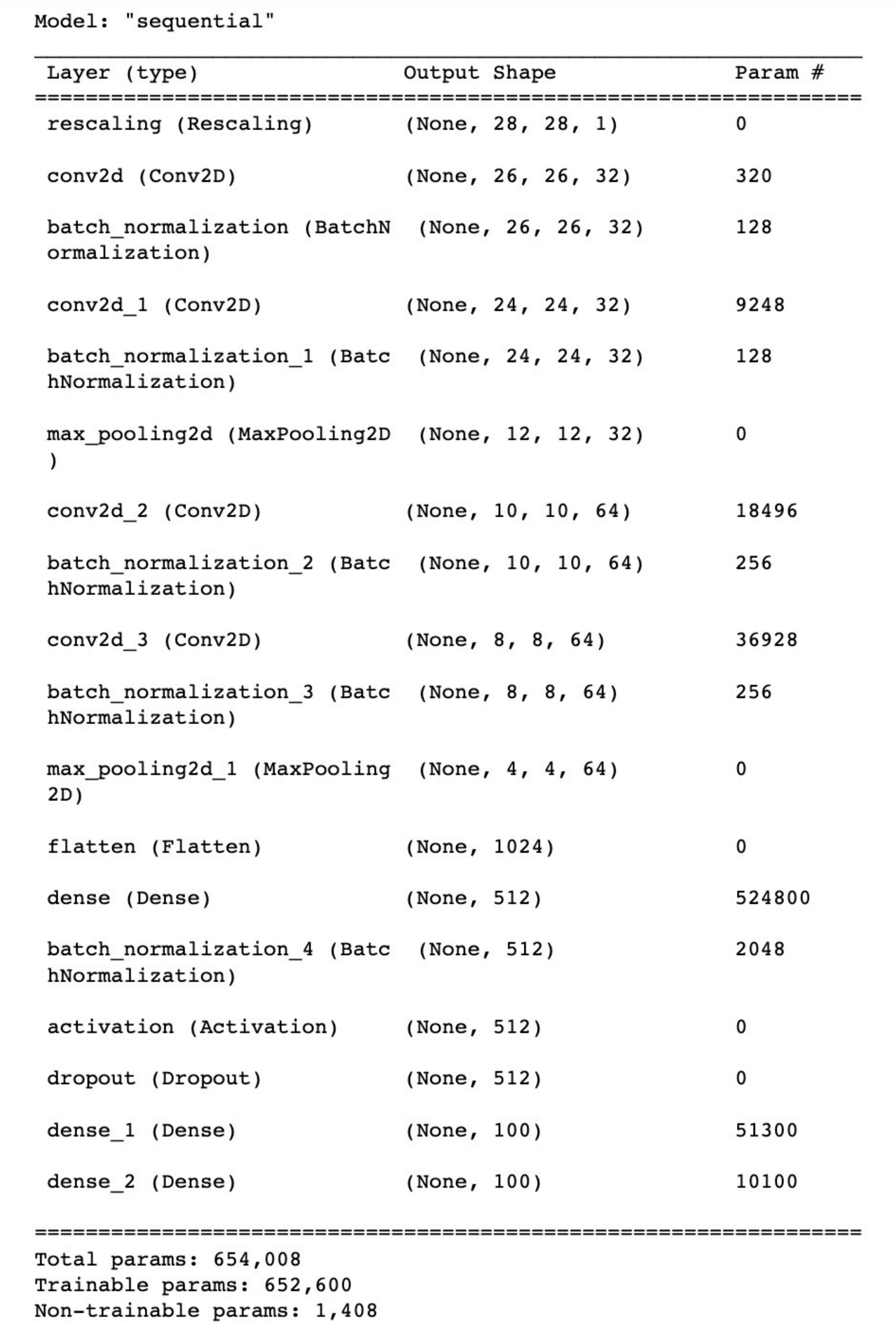
3. Machine Learning model Implementation



In [20]:

```
model = keras.Sequential([
    keras.layers.Rescaling(1./255, input_shape=(28, 28, 1)),
    keras.layers.Conv2D(32,(3,3),input_shape=(28,28,1),activation='relu'),
    keras.layers.BatchNormalization(axis=-1),
    keras.layers.Conv2D(32,(3,3),activation='relu'),
    keras.layers.BatchNormalization(axis=-1),
    keras.layers.MaxPooling2D(pool_size=(2,2)),
    keras.layers.Conv2D(64,(3,3),activation='relu'),
    keras.layers.BatchNormalization(axis=-1),
    keras.layers.Conv2D(64,(3,3),activation='relu'),
    keras.layers.BatchNormalization(axis=-1),
    keras.layers.MaxPooling2D(pool_size=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(512),
    keras.layers.BatchNormalization(),
    keras.layers.Activation('relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(100),
    keras.layers.Dense(100,activation=tf.nn.softmax)
])
```

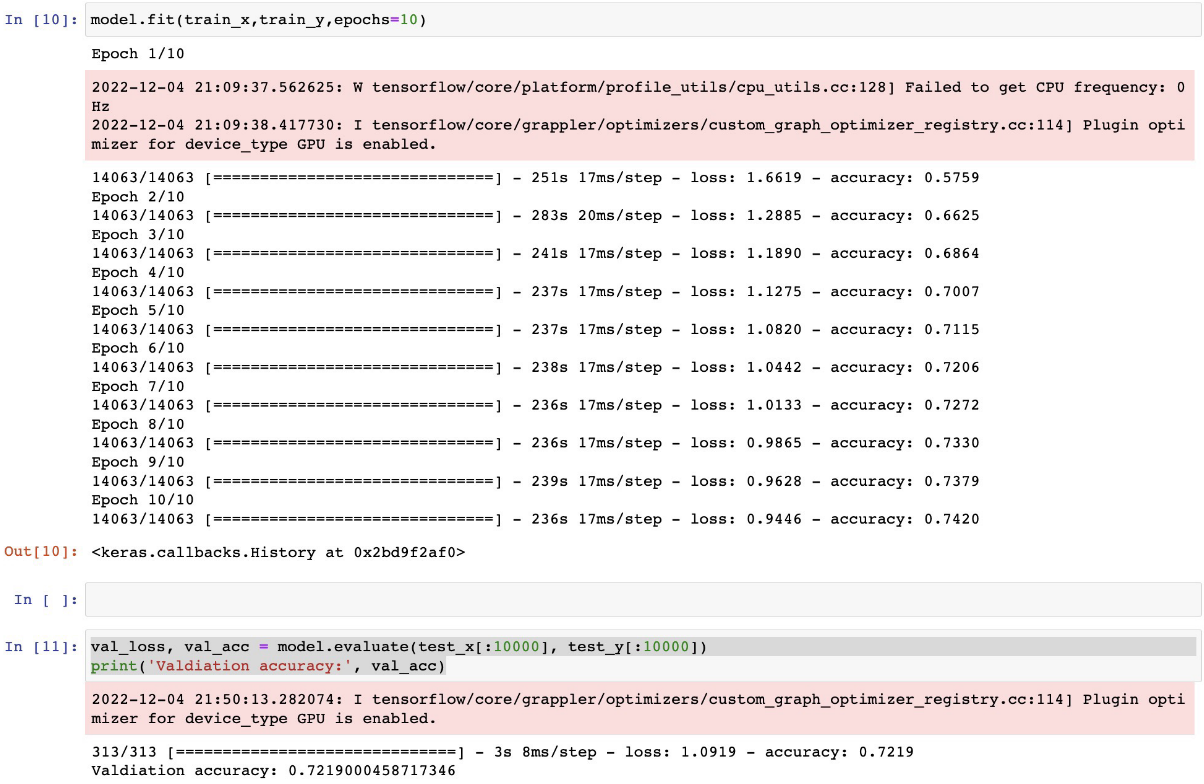
Model Summary :



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling (Rescaling)       (None, 28, 28, 1)         0         
                                                                 
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 batch_normalization (BatchN  (None, 26, 26, 32)       128       
 ormalization)                                                   
                                                                 
 conv2d_1 (Conv2D)           (None, 24, 24, 32)        9248      
                                                                 
 batch_normalization_1 (Batc  (None, 24, 24, 32)       128       
 hNormalization)                                                 
                                                                 
 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0         
 )                                                               
                                                                 
 conv2d_2 (Conv2D)           (None, 10, 10, 64)        18496     
                                                                 
 batch_normalization_2 (Batc  (None, 10, 10, 64)       256       
 hNormalization)                                                 
                                                                 
 conv2d_3 (Conv2D)           (None, 8, 8, 64)          36928     
                                                                 
 batch_normalization_3 (Batc  (None, 8, 8, 64)         256       
 hNormalization)                                                 
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 4, 4, 64)         0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 1024)              0         
                                                                 
 dense (Dense)               (None, 512)               524800    
                                                                 
 batch_normalization_4 (Batc  (None, 512)              2048      
 hNormalization)                                                 
                                                                 
 activation (Activation)     (None, 512)               0         
                                                                 
 dropout (Dropout)           (None, 512)               0         
                                                                 
 dense_1 (Dense)             (None, 100)               51300     
                                                                 
 dense_2 (Dense)             (None, 100)               10100     
                                                                 
=================================================================
Total params: 654,008
Trainable params: 652,600
Non-trainable params: 1,408
```

5. Results and Observations:

1. Training Accuracy: 74.2%
2. Validation accuracy: 72.2%



```
In [10]: model.fit(train_x,train_y,epochs=10)

         Epoch 1/10

         2022-12-04 21:09:37.562625: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0
         Hz
         2022-12-04 21:09:38.417730: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin opti
         mizer for device_type GPU is enabled.

         14063/14063 [==============================] - 251s 17ms/step - loss: 1.6619 - accuracy: 0.5759
         Epoch 2/10
         14063/14063 [==============================] - 283s 20ms/step - loss: 1.2885 - accuracy: 0.6625
         Epoch 3/10
         14063/14063 [==============================] - 241s 17ms/step - loss: 1.1890 - accuracy: 0.6864
         Epoch 4/10
         14063/14063 [==============================] - 237s 17ms/step - loss: 1.1275 - accuracy: 0.7007
         Epoch 5/10
         14063/14063 [==============================] - 237s 17ms/step - loss: 1.0820 - accuracy: 0.7115
         Epoch 6/10
         14063/14063 [==============================] - 238s 17ms/step - loss: 1.0442 - accuracy: 0.7206
         Epoch 7/10
         14063/14063 [==============================] - 236s 17ms/step - loss: 1.0133 - accuracy: 0.7272
         Epoch 8/10
         14063/14063 [==============================] - 236s 17ms/step - loss: 0.9865 - accuracy: 0.7330
         Epoch 9/10
         14063/14063 [==============================] - 239s 17ms/step - loss: 0.9628 - accuracy: 0.7379
         Epoch 10/10
         14063/14063 [==============================] - 236s 17ms/step - loss: 0.9446 - accuracy: 0.7420

Out[10]: <keras.callbacks.History at 0x2bd9f2af0>

In [ ]:

In [11]: val_loss, val_acc = model.evaluate(test_x[:10000], test_y[:10000])
         print('Valdiation accuracy:', val_acc)

         2022-12-04 21:50:13.282074: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plugin opti
         mizer for device_type GPU is enabled.

         313/313 [==============================] - 3s 8ms/step - loss: 1.0919 - accuracy: 0.7219
         Valdiation accuracy: 0.7219000458717346
```

6. How to improve accuracy

By using a pre-trained model like restnet can be implemented to improve further accuracy. Else adding more layers to the neural network can further lead to improve accuracy.