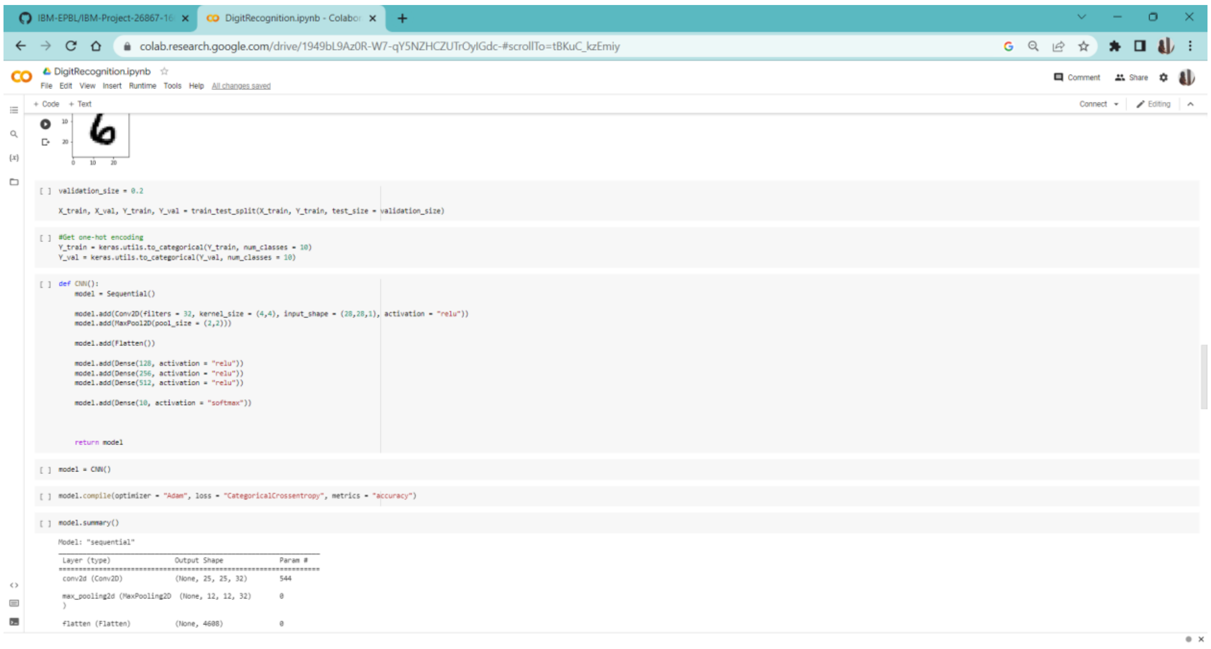
Model Training and evaluation:



```
validation_size = 0.2

X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, test_size = validation_size)
```

```
#Get one-hot encoding
Y_train = keras.utils.to_categorical(Y_train, num_classes = 10)
Y_val = keras.utils.to_categorical(Y_val, num_classes = 10)
```

```
def CNN():
    model = Sequential()

    model.add(Conv2D(filters = 32, kernel_size = (4,4), input_shape = (28,28,1), activation = "relu"))
    model.add(MaxPool2D(pool_size = (2,2)))

    model.add(Flatten())

    model.add(Dense(128, activation = "relu"))
    model.add(Dense(256, activation = "relu"))
    model.add(Dense(512, activation = "relu"))

    model.add(Dense(10, activation = "softmax"))


    return model
```

```
model = CNN()
```

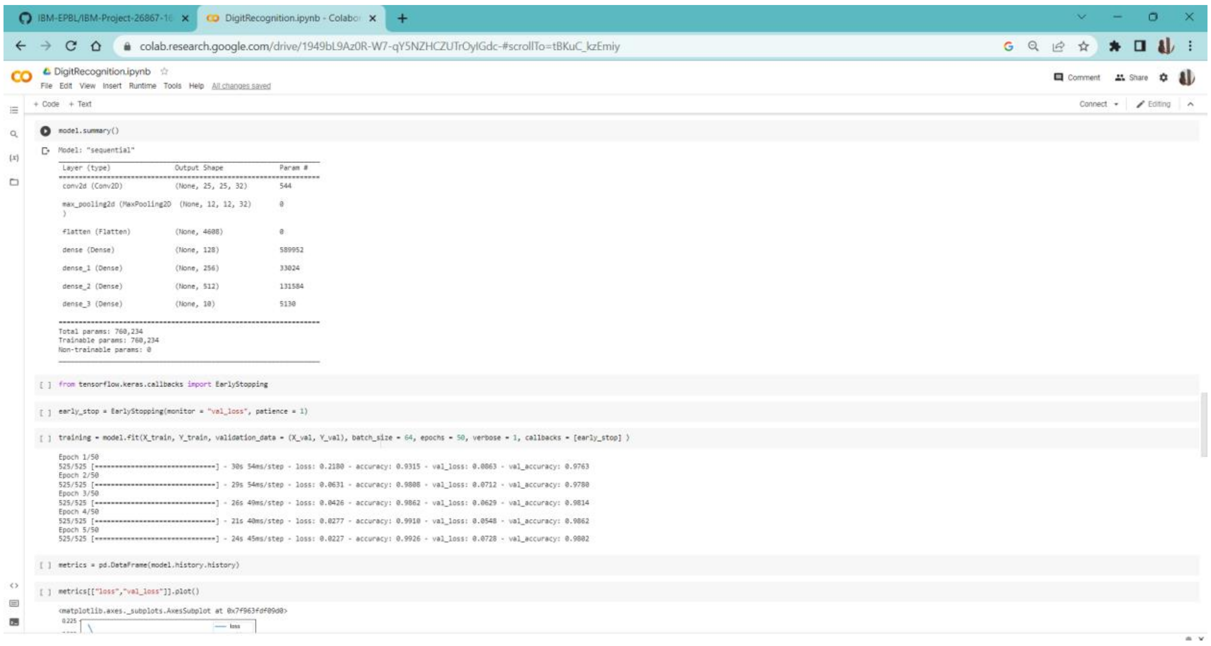
```
model.compile(optimizer = "Adam", loss = "CategoricalCrossentropy", metrics = "accuracy")
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 25, 25, 32)        544

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0
 )

 flatten (Flatten)           (None, 4608)              0
```



```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 25, 25, 32)        544

 max_pooling2d (MaxPooling2D  (None, 12, 12, 32)       0
 )

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 128)               589952

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 512)               131584

 dense_3 (Dense)             (None, 10)                5130

=================================================================
Total params: 760,234
Trainable params: 760,234
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.callbacks import EarlyStopping
```

```
early_stop = EarlyStopping(monitor = "val_loss", patience = 1)
```

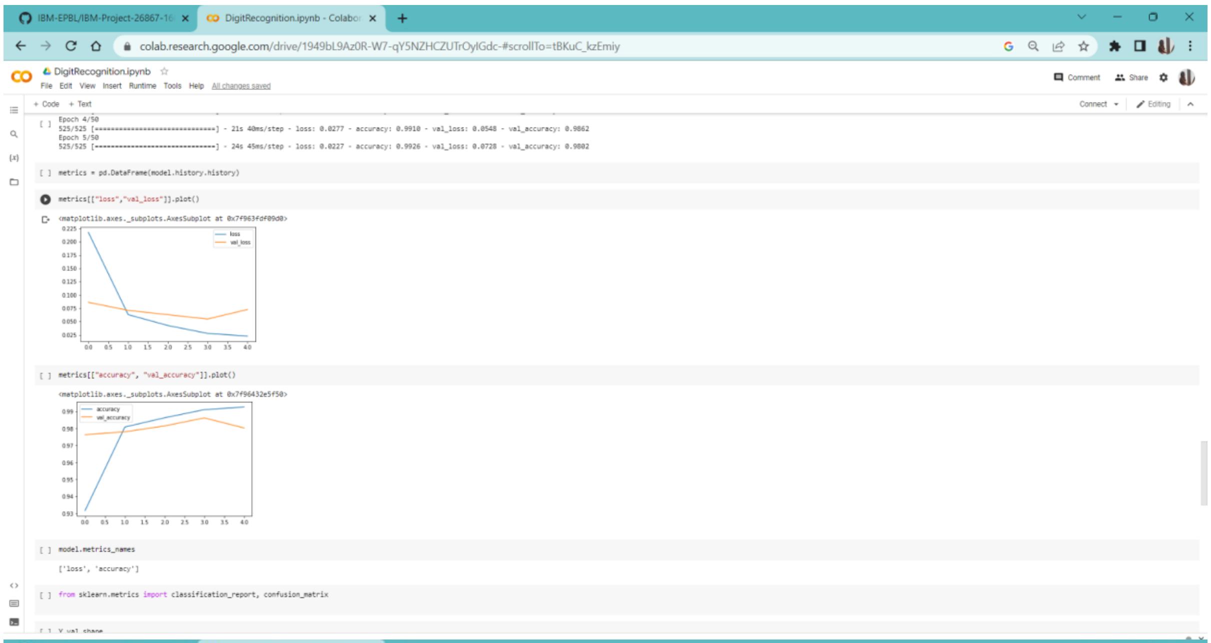
```
training = model.fit(X_train, Y_train, validation_data = (X_val, Y_val), batch_size = 64, epochs = 50, verbose = 1, callbacks = [early_stop] )
```

```
Epoch 1/50
525/525 [==============================] - 30s 54ms/step - loss: 0.2180 - accuracy: 0.9315 - val_loss: 0.0863 - val_accuracy: 0.9763
Epoch 2/50
525/525 [==============================] - 29s 54ms/step - loss: 0.0631 - accuracy: 0.9808 - val_loss: 0.0712 - val_accuracy: 0.9780
Epoch 3/50
525/525 [==============================] - 26s 49ms/step - loss: 0.0426 - accuracy: 0.9862 - val_loss: 0.0629 - val_accuracy: 0.9814
Epoch 4/50
525/525 [==============================] - 21s 40ms/step - loss: 0.0277 - accuracy: 0.9910 - val_loss: 0.0548 - val_accuracy: 0.9862
Epoch 5/50
525/525 [==============================] - 24s 45ms/step - loss: 0.0227 - accuracy: 0.9926 - val_loss: 0.0728 - val_accuracy: 0.9802
```

```
metrics = pd.DataFrame(model.history.history)
```

```
metrics[["loss","val_loss"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f963fdf09d0>
```



```
Epoch 4/50
525/525 [==============================] - 21s 40ms/step - loss: 0.0277 - accuracy: 0.9910 - val_loss: 0.0548 - val_accuracy: 0.9862
Epoch 5/50
525/525 [==============================] - 24s 45ms/step - loss: 0.0227 - accuracy: 0.9926 - val_loss: 0.0728 - val_accuracy: 0.9802
```

```
metrics = pd.DataFrame(model.history.history)
```

```
metrics[["loss","val_loss"]].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f963fdf09d0>
```

```
metrics[["accuracy", "val_accuracy"]].plot()
```

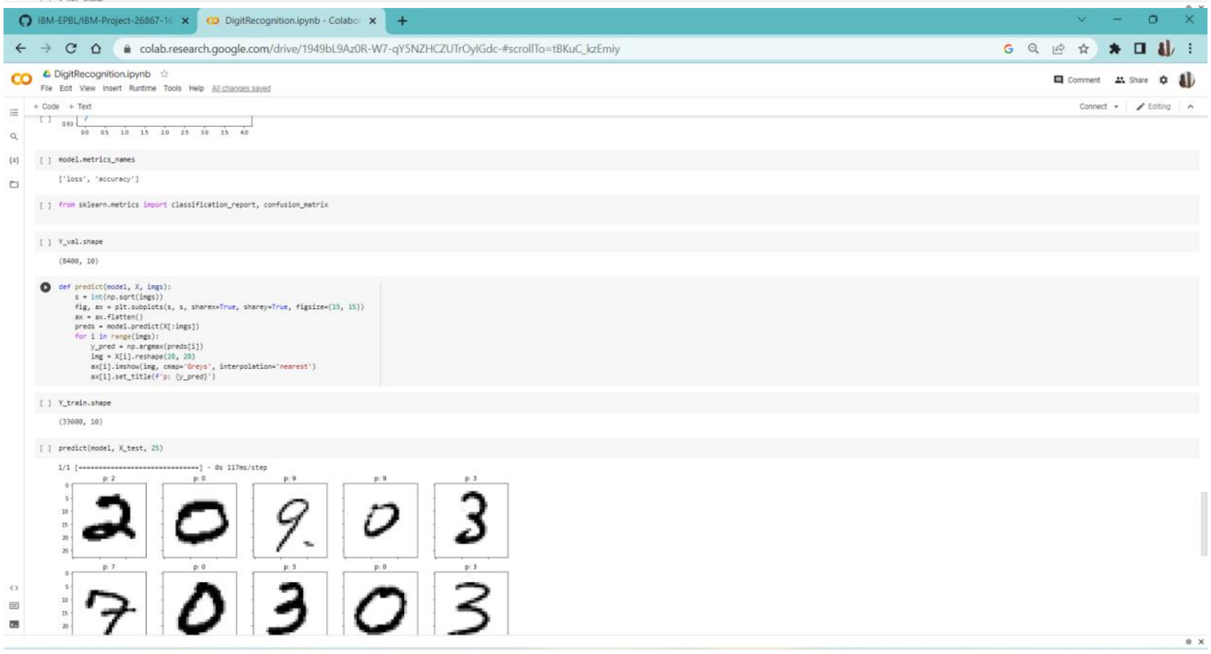
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f96432e5f50>
```

```
model.metrics_names
```

```
['loss', 'accuracy']
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

IBM-EPBL/IBM-Project-26867-16
DigitRecognition.ipynb - Colabor
colab.research.google.com/drive/1949bL9Az0R-W7-qY5NZHCZUTrOyIGdc-#scrollTo=tBKuC_kzEmiy
DigitRecognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
loss
val_loss
accuracy
val_accuracy



```
model.metrics_names
```

```
['loss', 'accuracy']
```

```
from sklearn.metrics import classification_report, confusion_matrix
```

```
Y_val.shape
```

```
(8400, 10)
```

```
def predict(model, X, imgs):
    s = int(np.sqrt(imgs))
    fig, ax = plt.subplots(s, s, sharex=True, sharey=True, figsize=(15, 15))
    ax = ax.flatten()
    preds = model.predict(X[:imgs])
    for i in range(imgs):
        y_pred = np.argmax(preds[i])
        img = X[i].reshape(28, 28)
        ax[i].imshow(img, cmap='Greys', interpolation='nearest')
        ax[i].set_title(f'p: {y_pred}')
```

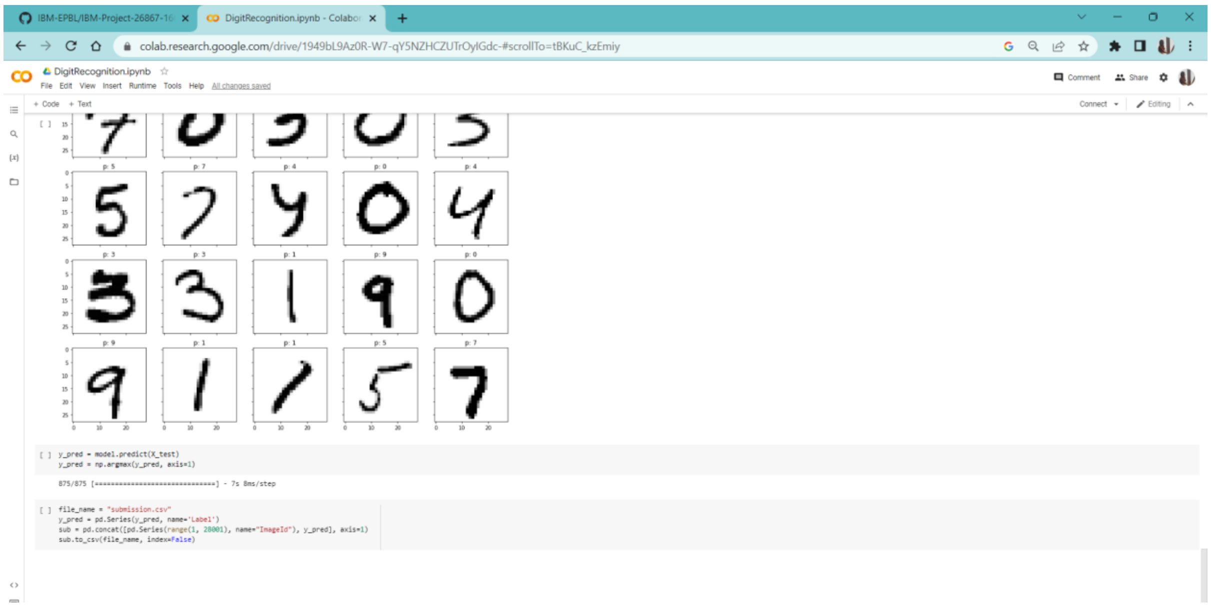
```
Y_train.shape
```

```
(33600, 10)
```

```
predict(model, X_test, 25)
```

```
1/1 [==============================] - 0s 117ms/step
```

IBM-EPBL/IBM-Project-26867-16
DigitRecognition.ipynb - Colabor
colab.research.google.com/drive/1949bL9Az0R-W7-qY5NZHCZUTrOyIGdc-#scrollTo=tBKuC_kzEmiy
DigitRecognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
p: 2
p: 0
p: 9
p: 9
p: 3
p: 7
p: 0
p: 3
p: 0
p: 3


IBM-EPBL/IBM-Project-26867-16
DigitRecognition.ipynb - Colabor
colab.research.google.com/drive/1949bL9Az0R-W7-qY5NZHCZUTrOyIGdc-#scrollTo=tBKuC_kzEmiy
DigitRecognition.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connect
Editing
p: 5
p: 7
p: 4
p: 0
p: 4
p: 3
p: 3
p: 1
p: 9
p: 0
p: 9
p: 1
p: 1
p: 5
p: 7

```
y_pred = model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
```

```
875/875 [==============================] - 7s 8ms/step
```

```
file_name = "submission.csv"
y_pred = pd.Series(y_pred, name='Label')
sub = pd.concat([pd.Series(range(1, 28001), name="ImageId"), y_pred], axis=1)
sub.to_csv(file_name, index=False)
```