Convolutional Deep Neural Network for Digit Classification

AIM

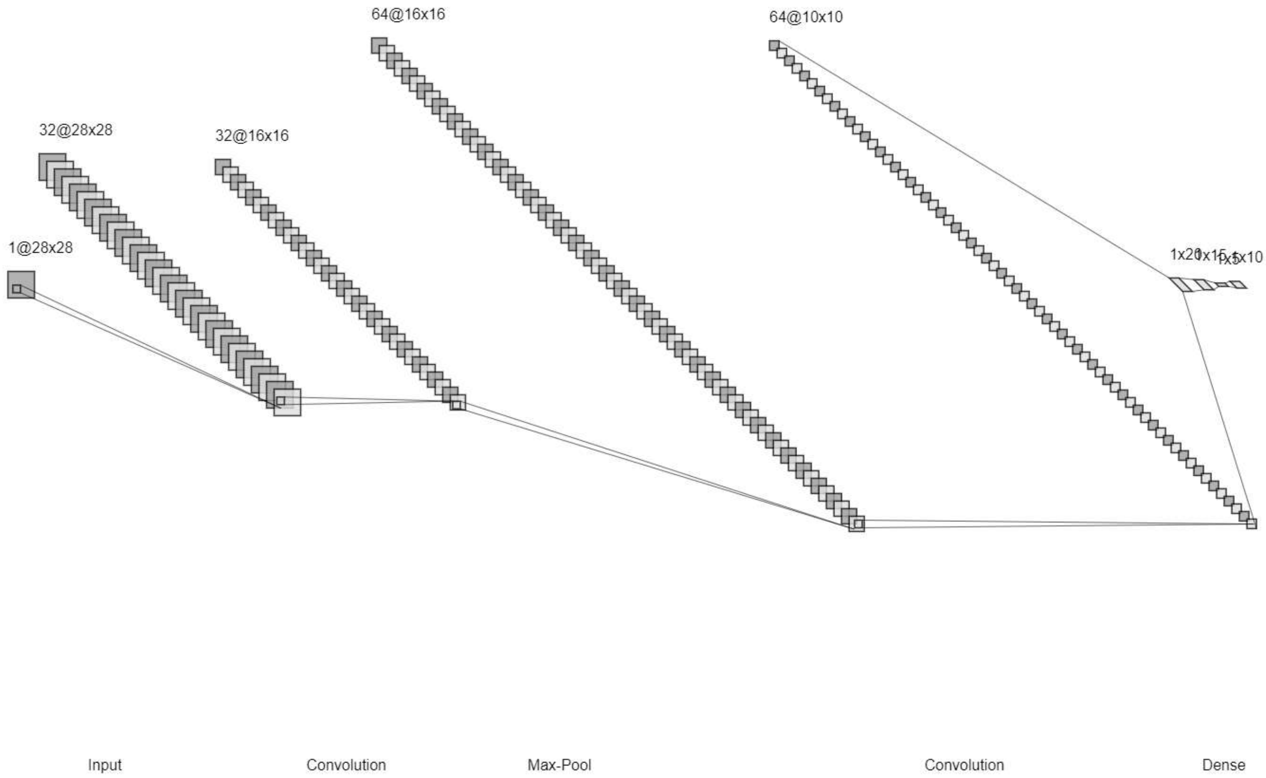
To Develop a convolutional deep neural network for digit classification and to verify the response for scanned handwritten images.

Problem Statement and Dataset

Digit classification and to verify the response for scanned handwritten images.

The MNIST dataset is a collection of handwritten digits. The task is to classify a given image of a handwritten digit into one of 10 classes representing integer values from 0 to 9, inclusively. The dataset has a collection of 60,000 handwrittend digits of size 28 X 28. Here we build a convolutional neural network model that is able to classify to it's appropriate numerical value.

Neural Network Model



DESIGN STEPS

STEP 1:

Import tensorflow and preprocessing libraries

STEP 2:

Build a CNN model

STEP 3:

Compile and fit the model and then predict

PROGRAM

Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.metrics import classification_report,confusion_matrix

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
from tensorflow.keras import utils
from tensorflow.keras.preprocessing import image
```

One Hot Encoding Outputs

```
y_train_onehot = utils.to_categorical(y_train,10)
y_test_onehot = utils.to_categorical(y_test,10)
```

Reshape Inputs

```
X_train_scaled = X_train_scaled.reshape(-1,28,28,1)
X_test_scaled = X_test_scaled.reshape(-1,28,28,1)
```

Build CNN Model

```
model = keras.Sequential()
input = keras.Input(shape=(28,28,1))
model.add(input)

model.add(layers.Conv2D(filters=32,kernel_size=(5,5),
            strides=(1,1),padding='valid',activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=64,kernel_size=(5,5),
            strides=(1,1),padding='same',activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(20,activation='relu'))
model.add(layers.Dense(15,activation='relu'))
model.add(layers.Dense(5,activation='relu'))
model.add(layers.Dense(10,activation='softmax'))

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

model.fit(X_train_scaled ,y_train_onehot, epochs=5,batch_size=64,
          validation_data=(X_test_scaled,y_test_onehot))
```

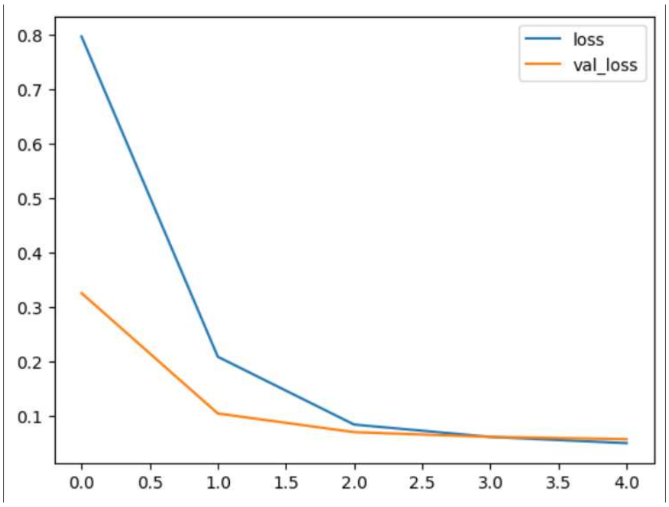
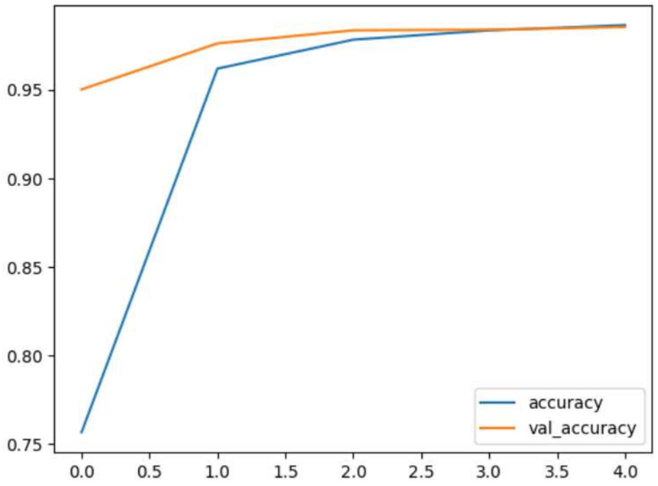
Metrics

```
metrics = pd.DataFrame(model.history.history)
metrics.head()
metrics[['loss','val_loss']].plot()
metrics[['accuracy','val_accuracy']].plot()

x_test_predictions = np.argmax(model.predict(X_test_scaled), axis=1)
print(confusion_matrix(y_test,x_test_predictions))
print(classification_report(y_test,x_test_predictions))
```

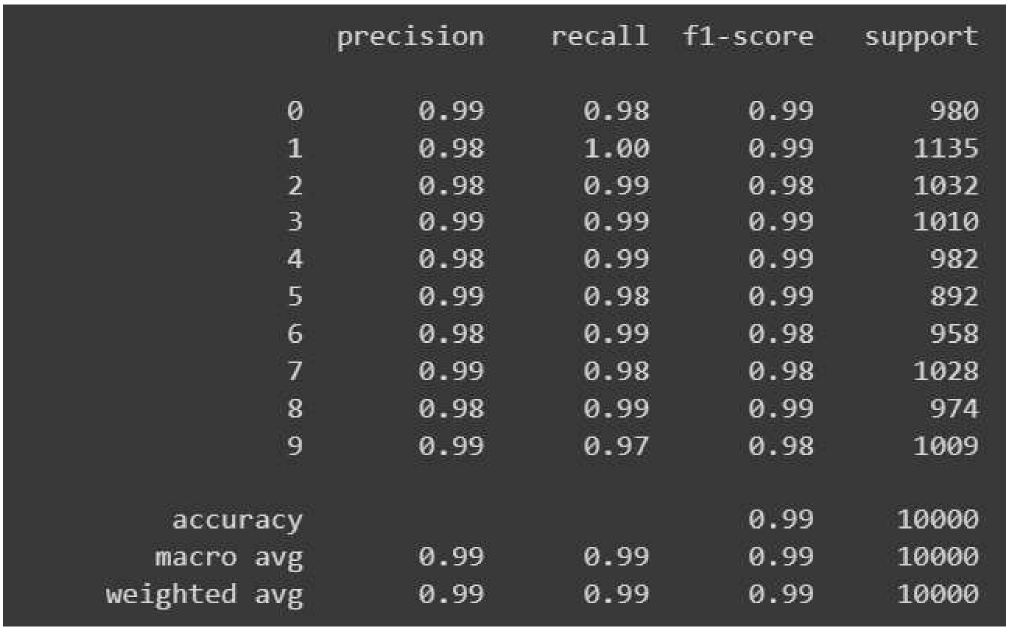
Predict for own handwriting

```
img = image.load_img('/drive/MyDrive/Colab Notebooks/Deep Learning/Lab/Exp 3/eight.png')
img_tensor = tf.convert_to_tensor(np.asarray(img))
img_28 = tf.image.resize(img_tensor,(28,28))
img_28_gray = tf.image.rgb_to_grayscale(img_28)
img_28_gray_inverted = 255.0-img_28_gray
img_28_gray_inverted_scaled = img_28_gray_inverted.numpy()/255.0
x_single_prediction = np.argmax(
   model.predict(img_28_gray_inverted_scaled.reshape(1,28,28,1)),axis=1)
plt.imshow(img_28_gray_inverted_scaled.reshape(28,28),cmap='gray')
print(x_single_prediction)
```

OUTPUT

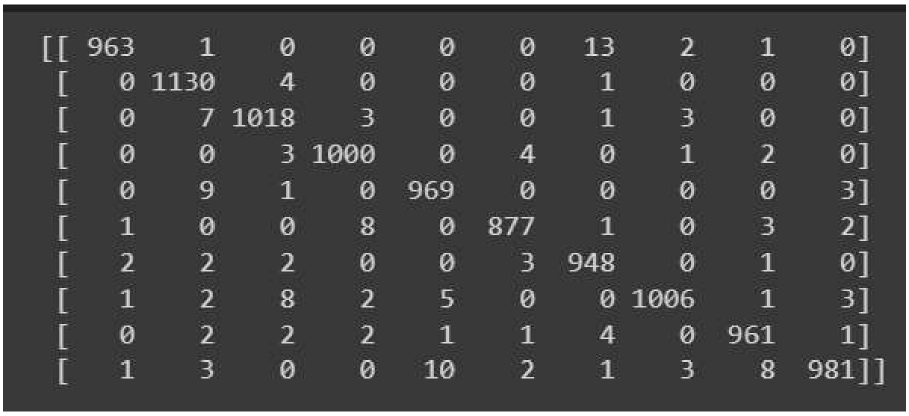
Training Loss, Validation Loss Vs Iteration	Accuracy, Validation Accuracy Vs Iteration
	

Classification Report



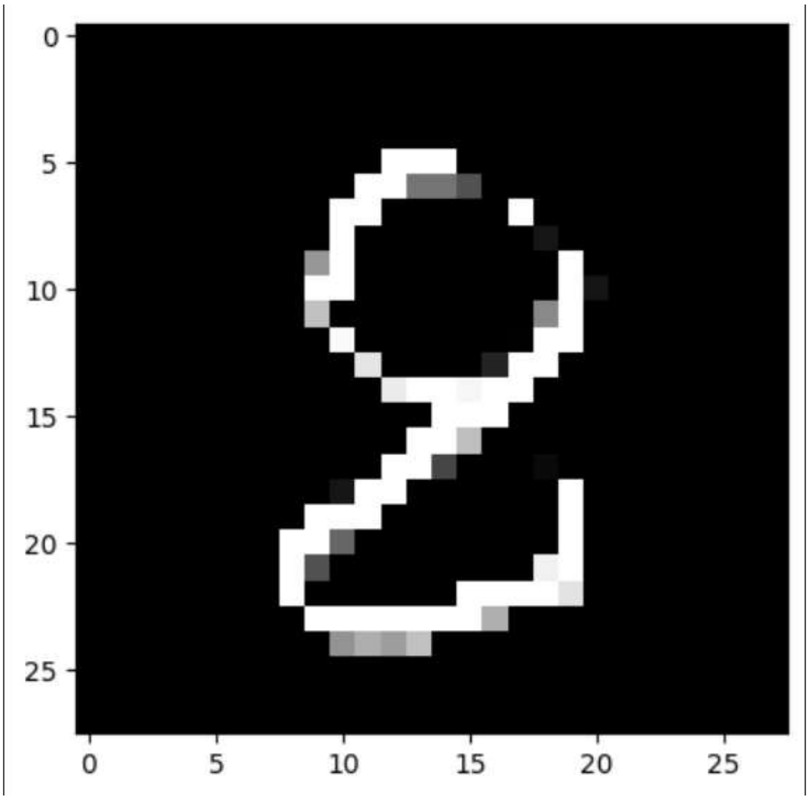
```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       980
           1       0.98      1.00      0.99      1135
           2       0.98      0.99      0.98      1032
           3       0.99      0.99      0.99      1010
           4       0.98      0.99      0.99       982
           5       0.99      0.98      0.99       892
           6       0.98      0.99      0.98       958
           7       0.99      0.98      0.98      1028
           8       0.98      0.99      0.99       974
           9       0.99      0.97      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

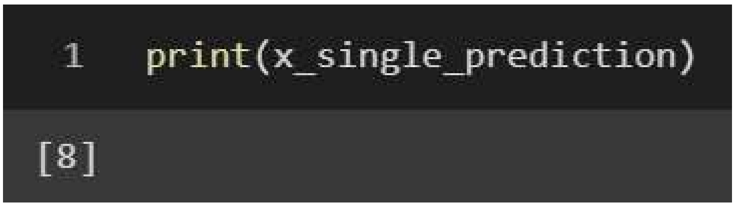
Confusion Matrix



```
[[ 963    1    0    0    0    0   13    2    1    0]
 [   0 1130    4    0    0    0    1    0    0    0]
 [   0    7 1018    3    0    0    1    3    0    0]
 [   0    0    3 1000    0    4    0    1    2    0]
 [   0    9    1    0  969    0    0    0    0    3]
 [   1    0    0    8    0  877    1    0    3    2]
 [   2    2    2    0    0    3  948    0    1    0]
 [   1    2    8    2    5    0    0 1006    1    3]
 [   0    2    2    2    1    1    4    0  961    1]
 [   1    3    0    0   10    2    1    3    8  981]]
```

New Sample Data Prediction





```
print(x_single_prediction)
```

```
[8]
```

RESULT

A convolutional deep neural network for digit classification and to verify the response for scanned handwritten images is developed sucessfully.