Batiquin, Cherry Mae IT4R5

[127]:
```
import pandas as pd

train = pd.read_csv("/kaggle/input/train-and-test/CNNtrain.csv")
print(train.shape)
train.head()
```

```
(42000, 785)
```

[127…

	label	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

[128]:
```
# read test
test= pd.read_csv("/kaggle/input/train-and-test/CNNtest.csv")
print(test.shape)
test.head()
```

```
(28000, 784)
```

[128…

	pixel0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

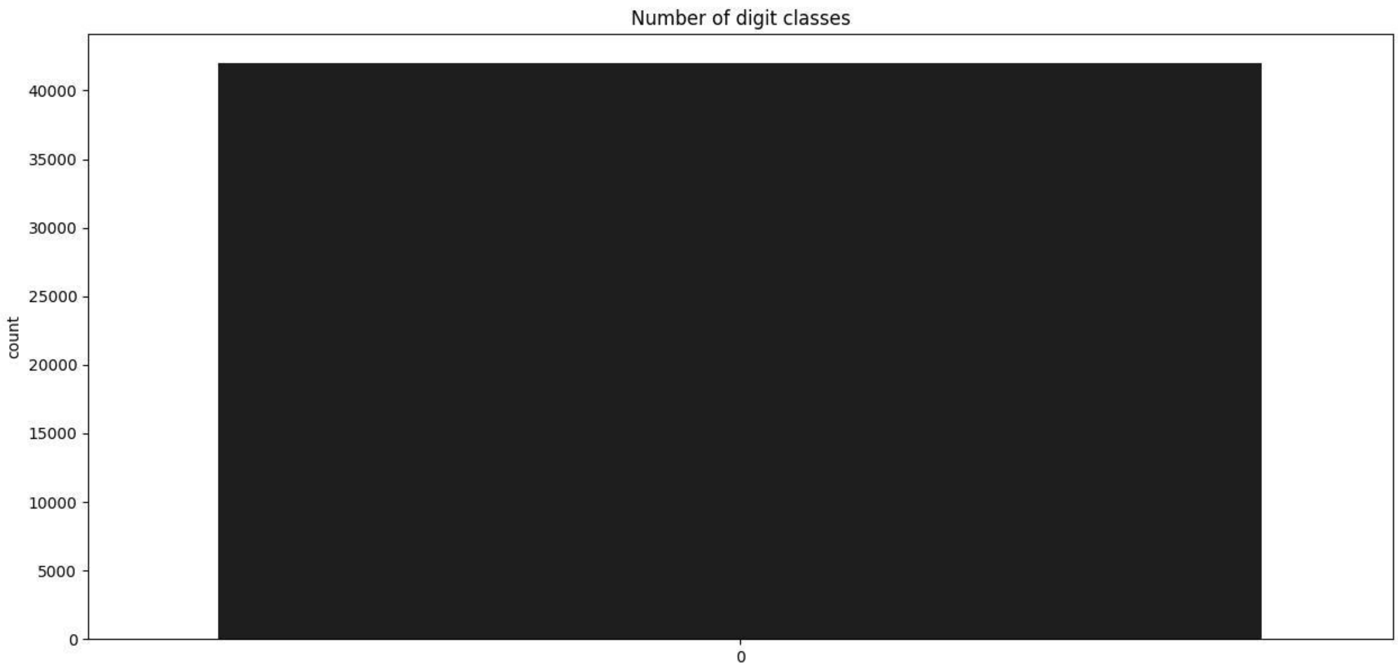
5 rows × 784 columns

[129]:
```
# put labels into y_train variable
Y_train = train["label"]
# Drop 'label' column
X_train = train.drop(labels = ["label"],axis = 1)
```

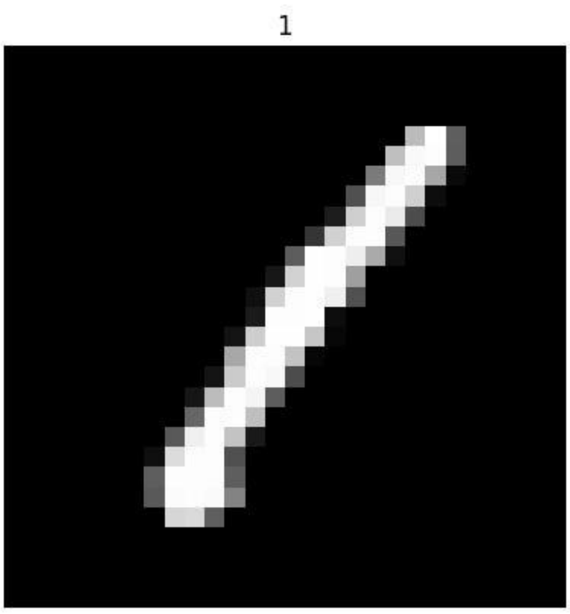
[130]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(15,7))
g = sns.countplot(Y_train, palette="icefire")
plt.title("Number of digit classes")
Y_train.value_counts()
```

[130…
```
label
1    4684
7    4401
3    4351
9    4188
2    4177
6    4137
0    4132
4    4072
8    4063
5    3795
Name: count, dtype: int64
```

Batiquin, Cherry Mae IT4R5




```
# plot some samples
img = X_train.iloc[0].to_numpy()
img = img.reshape((28,28))
plt.imshow(img,cmap='gray')
plt.title(train.iloc[0,0])
plt.axis("off")
plt.show()
```



Batiquin, Cherry Mae IT4R5

```python
# Normalize the data
X_train = X_train / 255.0
test = test / 255.0
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)
```

```
x_train shape:  (42000, 784)
test shape:  (28000, 784)
```

```python
# Reshape
X_train = X_train.values.reshape(-1,28,28,1)
test = test.values.reshape(-1,28,28,1)
print("x_train shape: ",X_train.shape)
print("test shape: ",test.shape)
```

```
x_train shape:  (42000, 28, 28, 1)
test shape:  (28000, 28, 28, 1)
```

```python
# Label Encoding
from tensorflow.keras.utils import to_categorical # convert to one-hot-encoding
Y_train = to_categorical(Y_train, num_classes = 10)
```

```python
# Split the train and the validation set for the fitting
from sklearn.model_selection import train_test_split
X_train, X_val, Y_train, Y_val = train_test_split(X_train,
                                                  Y_train,
                                                  test_size = 0.1,
                                                  random_state=2)
print("x_train shape",X_train.shape)
print("x_test shape",X_val.shape)
print("y_train shape",Y_train.shape)
print("y_test shape",Y_val.shape)
```

```
x_train shape (37800, 28, 28, 1)
x_test shape (4200, 28, 28, 1)
y_train shape (37800, 10)
y_test shape (4200, 10)
```

```python
from sklearn.metrics import confusion_matrix
import itertools

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import Flatten, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import RMSprop,Adam
```

Batiquin, Cherry Mae IT4R5

```
model = Sequential()
# input layer
model.add(Conv2D(filters = 8, kernel_size = (7,7),padding = 'Same',
                 activation ='relu', input_shape = (28,28,1)))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))
# hidden layer 1
model.add(Conv2D(filters = 16, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

# hidden layer 2
model.add(Conv2D(filters = 8, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.25))

# fully connected
model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation = "softmax"))
```

```
# Define the optimizer
optimizer = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999)
```

```
# Compile the model
model.compile(optimizer = optimizer ,
              loss = "categorical_crossentropy",
              metrics=["accuracy"])
```

Batiquin, Cherry Mae
IT4R5

```python
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 28, 28, 8)         400

 max_pooling2d_20 (MaxPooli  (None, 14, 14, 8)         0
 ng2D)

 dropout_31 (Dropout)        (None, 14, 14, 8)         0

 conv2d_21 (Conv2D)          (None, 14, 14, 16)        3216

 max_pooling2d_21 (MaxPooli  (None, 7, 7, 16)          0
 ng2D)

 dropout_32 (Dropout)        (None, 7, 7, 16)          0

 conv2d_22 (Conv2D)          (None, 7, 7, 8)           1160

 max_pooling2d_22 (MaxPooli  (None, 3, 3, 8)           0
 ng2D)

 dropout_33 (Dropout)        (None, 3, 3, 8)           0

 flatten_11 (Flatten)        (None, 72)                0

 dense_28 (Dense)            (None, 256)               18688

 dropout_34 (Dropout)        (None, 256)               0

 dense_29 (Dense)            (None, 10)                2570

=================================================================
Total params: 26034 (101.70 KB)
Trainable params: 26034 (101.70 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```python
# data augmentation
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # dimesion reduction
        rotation_range=5,  # randomly rotate images in the range 5 degrees
        zoom_range = 0.1, # Randomly zoom image 10%
        width_shift_range=0.1,  # randomly shift images horizontally 10%
        height_shift_range=0.1,  # randomly shift images vertically 10%
        horizontal_flip=False,  # randomly flip images
        vertical_flip=False)  # randomly flip images

datagen.fit(X_train)
```

Batiquin, Cherry Mae IT4R5

```python
epochs = 15  # for better result increase the epochs
batch_size = 100
```

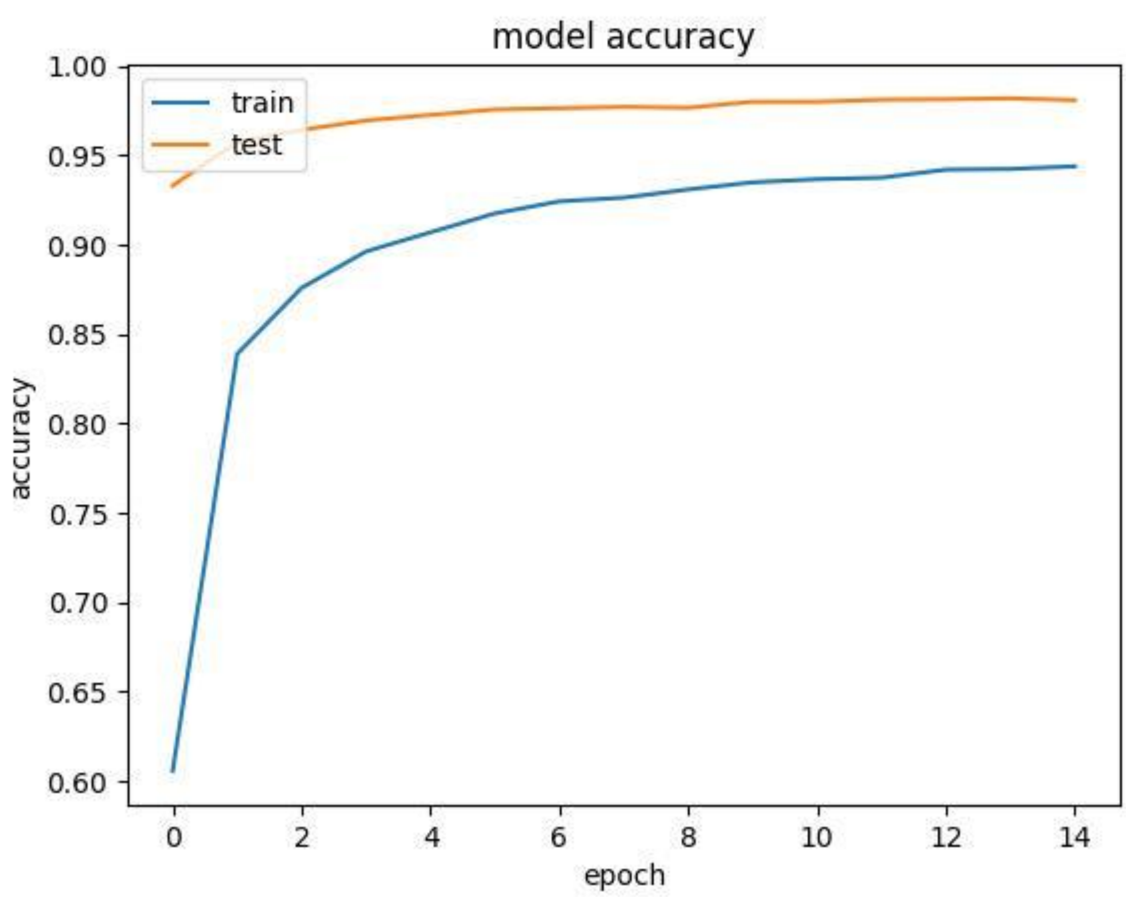
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

```python
# Fit the model
history = model.fit(datagen.flow(X_train,Y_train, batch_size=batch_size),
                    epochs = epochs,
                    validation_data = (X_val,Y_val),
                    steps_per_epoch=X_train.shape[0] // batch_size)
```

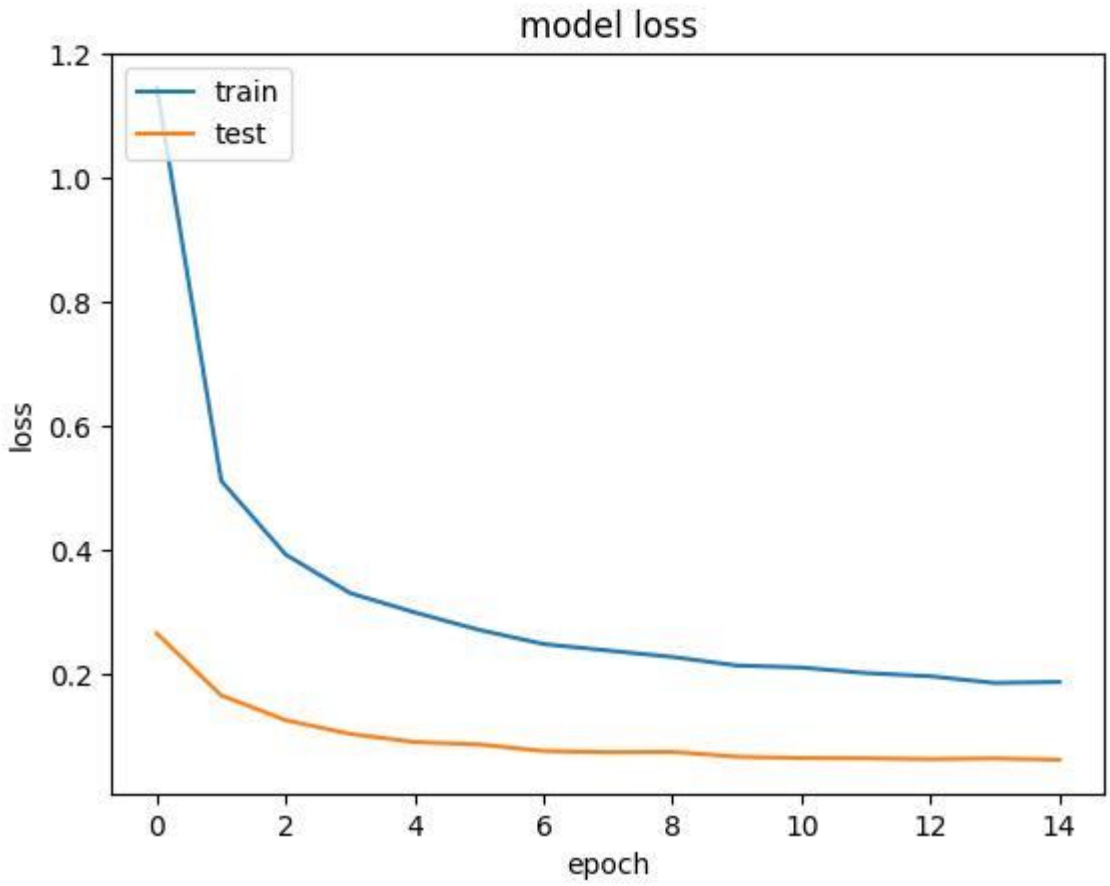
```
Epoch 1/15
378/378 [==============================] - 26s 64ms/step - loss: 1.1463 - accuracy: 0.6057 - val_loss: 0.2646 - val_accuracy: 0.9331
Epoch 2/15
378/378 [==============================] - 24s 64ms/step - loss: 0.5110 - accuracy: 0.8389 - val_loss: 0.1645 - val_accuracy: 0.9574
Epoch 3/15
378/378 [==============================] - 24s 62ms/step - loss: 0.3919 - accuracy: 0.8759 - val_loss: 0.1245 - val_accuracy: 0.9643
Epoch 4/15
378/378 [==============================] - 24s 64ms/step - loss: 0.3297 - accuracy: 0.8963 - val_loss: 0.1023 - val_accuracy: 0.9695
Epoch 5/15
378/378 [==============================] - 24s 63ms/step - loss: 0.2986 - accuracy: 0.9070 - val_loss: 0.0892 - val_accuracy: 0.9726
Epoch 6/15
378/378 [==============================] - 24s 63ms/step - loss: 0.2703 - accuracy: 0.9175 - val_loss: 0.0854 - val_accuracy: 0.9757
Epoch 7/15
378/378 [==============================] - 24s 63ms/step - loss: 0.2474 - accuracy: 0.9243 - val_loss: 0.0747 - val_accuracy: 0.9764
Epoch 8/15
378/378 [==============================] - 24s 62ms/step - loss: 0.2369 - accuracy: 0.9263 - val_loss: 0.0728 - val_accuracy: 0.9771
Epoch 9/15
378/378 [==============================] - 24s 63ms/step - loss: 0.2266 - accuracy: 0.9310 - val_loss: 0.0734 - val_accuracy: 0.9767
Epoch 10/15
378/378 [==============================] - 24s 63ms/step - loss: 0.2129 - accuracy: 0.9349 - val_loss: 0.0656 - val_accuracy: 0.9800
Epoch 11/15
378/378 [==============================] - 24s 64ms/step - loss: 0.2095 - accuracy: 0.9367 - val_loss: 0.0633 - val_accuracy: 0.9800
Epoch 12/15
378/378 [==============================] - 24s 62ms/step - loss: 0.2006 - accuracy: 0.9375 - val_loss: 0.0629 - val_accuracy: 0.9812
Epoch 13/15
378/378 [==============================] - 24s 63ms/step - loss: 0.1954 - accuracy: 0.9420 - val_loss: 0.0617 - val_accuracy: 0.9814
Epoch 14/15
378/378 [==============================] - 24s 63ms/step - loss: 0.1846 - accuracy: 0.9424 - val_loss: 0.0627 - val_accuracy: 0.9819
Epoch 15/15
378/378 [==============================] - 23s 62ms/step - loss: 0.1862 - accuracy: 0.9439 - val_loss: 0.0606 - val_accuracy: 0.9810
```

Batiquin, Cherry Mae IT4R5

```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Batiquin, Cherry Mae IT4R5




```
# confusion matrix
import seaborn as sns
# Predict the values from the validation dataset
Y_pred = model.predict(X_val)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred,axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(Y_val,axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
f,ax = plt.subplots(figsize=(8, 8))
sns.heatmap(confusion_mtx, annot=True, linewidths=0.01,
            cmap="Greens",linecolor="gray", fmt= '.1f',ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

```
132/132 [==============================] - 1s 7ms/step
```

Batiquin, Cherry Mae IT4R5