cnn-mnist

April 23, 2024

[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

0.0.1 Load Dataset

[2]:
```
train_df = pd.read_csv('fashion-mnist_train.csv')
test_df = pd.read_csv('fashion-mnist_test.csv')
```

[3]:
```
train_df.shape
```

[3]:
```
(60000, 785)
```

[4]:
```
test_df.shape
```

[4]:
```
(10000, 785)
```

[5]:
```
train_df.describe()
```

[5]:

	label	pixel1	pixel2	pixel3	pixel4	\
count	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	
mean	4.500000	0.000900	0.006150	0.035333	0.101933	
std	2.872305	0.094689	0.271011	1.222324	2.452871	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	0.000000	0.000000	0.000000	0.000000	
50%	4.500000	0.000000	0.000000	0.000000	0.000000	
75%	7.000000	0.000000	0.000000	0.000000	0.000000	
max	9.000000	16.000000	36.000000	226.000000	164.000000	

	pixel5	pixel6	pixel7	pixel8	pixel9	\
count	60000.000000	60000.000000	60000.000000	60000.000000	60000.000000	
mean	0.247967	0.411467	0.805767	2.198283	5.682000	
std	4.306912	5.836188	8.215169	14.093378	23.819481	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.000000	0.000000	0.000000	

```
max      227.000000    230.000000    224.000000    255.000000    254.000000

       …      pixel775      pixel776      pixel777      pixel778  \
count  …  60000.000000  60000.000000  60000.000000  60000.000000
mean   …     34.625400     23.300683     16.588267     17.869433
std    …     57.545242     48.854427     41.979611     43.966032
min    …      0.000000      0.000000      0.000000      0.000000
25%    …      0.000000      0.000000      0.000000      0.000000
50%    …      0.000000      0.000000      0.000000      0.000000
75%    …     58.000000      9.000000      0.000000      0.000000
max    …    255.000000    255.000000    255.000000    255.000000

           pixel779      pixel780      pixel781      pixel782      pixel783  \
count  60000.000000  60000.000000  60000.000000  60000.000000  60000.000000
mean      22.814817     17.911483      8.520633      2.753300      0.855517
std       51.830477     45.149388     29.614859     17.397652      9.356960
min        0.000000      0.000000      0.000000      0.000000      0.000000
25%        0.000000      0.000000      0.000000      0.000000      0.000000
50%        0.000000      0.000000      0.000000      0.000000      0.000000
75%        0.000000      0.000000      0.000000      0.000000      0.000000
max      255.000000    255.000000    255.000000    255.000000    255.000000

          pixel784
count  60000.00000
mean       0.07025
std        2.12587
min        0.00000
25%        0.00000
50%        0.00000
75%        0.00000
max      170.00000

[8 rows x 785 columns]
```

[6]:
```
train_df.label.unique()
```

[6]:
```
array([2, 9, 6, 0, 3, 4, 5, 8, 7, 1], dtype=int64)
```

Each row represents an grayscale image containing 784 pixels and each pixel having values in range from 0-255

The column label is a discrete value in the range 0 to 9 each value representing a specific category

[7]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat','Sandal',␣
 ↪'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

0.0.2 Preprocess Data

Convert each image of 784 into (28x28x1)(height x width x color_channels). Divide values by 255 to scale the values.

[8]:
```
x_train = train_df.iloc[:,1:].to_numpy()
x_train = x_train.reshape([-1,28,28,1])
x_train = x_train / 255
```

[9]:
```
y_train = train_df.iloc[:,0].to_numpy()
```

[10]:
```
x_test = test_df.iloc[:,1:].to_numpy()
x_test = x_test.reshape([-1,28,28,1])
x_test = x_test / 255
```

[11]:
```
y_test = test_df.iloc[:,0].to_numpy()
```

0.0.3 Visualization

[12]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

0.0.4 Model Building

[15]:
```
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten,MaxPooling2D,Dropout
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[15], line 1
----> 1 from keras.models import Sequential
      2 from keras.layers import Dense,Conv2D,Flatten,MaxPooling2D,Dropout
```

```
ModuleNotFoundError: No module named 'keras'
```

[16]:
```
model = Sequential()

model.
 ↪add(Conv2D(filters=64,kernel_size=(3,3),input_shape=(28,28,1),activation='relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(rate=0.3))
model.add(Flatten())
model.add(Dense(units=32, activation='relu'))
model.add(Dense(units=10, activation='sigmoid'))
model.
 ↪compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 model = Sequential()
      3 model.
 ↪add(Conv2D(filters=64,kernel_size=(3,3),input_shape=(28,28,1),activation='rel
 ↪u'))
      4 model.add(MaxPooling2D(pool_size = (2,2)))

NameError: name 'Sequential' is not defined
```

[17]:
```
model.fit(x_train,y_train,epochs=50,batch_size=1200,validation_split=0.05)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 model.fit(x_train,y_train,epochs=50,batch_size=1200,validation_split=0.
 ↪05)

NameError: name 'model' is not defined
```

0.0.5 Evaluation

[18]:
```
evaluation = model.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.2265 -
accuracy: 0.9252
```

[30]:
```
print(f"Accuracy: {evaluation[1]}")
```

```
Accuracy: 0.9251999855041504
```

[26]:
```
y_probas = model.predict(x_test)
```

```
313/313 [==============================] - 1s 3ms/step
```

[28]:
```
y_pred = y_probas.argmax(axis=-1)
```

[29]:
```
y_pred
```

[29]:
```
array([0, 1, 2, …, 8, 8, 1])
```

[38]:
```
plt.figure(figsize=(10,10),)
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_test[i], cmap=plt.cm.binary)
#     plt.xlabel(f"True Class:{y_test[i]}")
    plt.title(f"Pred:{class_names[y_pred[i]]}")
plt.show()
```

[39]:
```
from sklearn.metrics import classification_report
```

[40]:
```
num_classes = 10
class_names = ["class {}".format(i) for i in range(num_classes)]
cr = classification_report(y_test, y_pred, target_names=class_names)
print(cr)
```

```
              precision    recall  f1-score   support

     class 0       0.88      0.87      0.87      1000
     class 1       0.99      0.99      0.99      1000
     class 2       0.90      0.88      0.89      1000
     class 3       0.92      0.94      0.93      1000
```

```
      class 4       0.88      0.90      0.89      1000
      class 5       0.99      0.97      0.98      1000
      class 6       0.79      0.78      0.78      1000
      class 7       0.96      0.96      0.96      1000
      class 8       0.99      0.98      0.98      1000
      class 9       0.96      0.97      0.97      1000

     accuracy                           0.93     10000
    macro avg       0.93      0.93      0.93     10000
 weighted avg       0.93      0.93      0.93     10000
```

[ ]: