ist-handwritten-digits-recognition

January 31, 2024

0.1 Dataset Information

This dataset allows you to study, analyze and recognize elements in the images. That's exactly how your camera detects your face, using image recognition! It's a digit recognition problem. This data set has 49,000 images of 28 X 28 size, totalling 49 MB.

0.2 Import Modules

[ ]:

```
# !pip install tensorflow-gpu keras
```

[2]:

```
import pandas as pd
import numpy as np
from tqdm.notebook import tqdm
from keras.preprocessing.image import img_to_array, load_img
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
import warnings

warnings.filterwarnings('ignore')
```

0.3 Unzip the train data|

[1]:

```
# !unzip Train_UQcUa52.zip
```

0.4 Load the data

[4]:

```
df = pd.read_csv('train.csv')
df.head()
```

[4]:

```
  filename  label
0    0.png      4
1    1.png      9
2    2.png      1
3    3.png      7
4    4.png      3
```

[5]:

```
!pwd
```

```
/content
```

[6]:

```
image_path = 'Images/train/'
```

[8]:

```
X = np.array([img_to_array(load_img(image_path+df['filename'][i],␣
 ↪target_size=(28,28,1), grayscale=True))
              for i in tqdm(range(df.shape[0]))
              ]).astype('float32')
```

```
HBox(children=(FloatProgress(value=0.0, max=49000.0), HTML(value='')))
```

[9]:

```
y = df['label']
```

[10]:

```
print(X.shape, y.shape)
```

```
(49000, 28, 28, 1) (49000,)
```

0.5 Exploratory Data Analysis

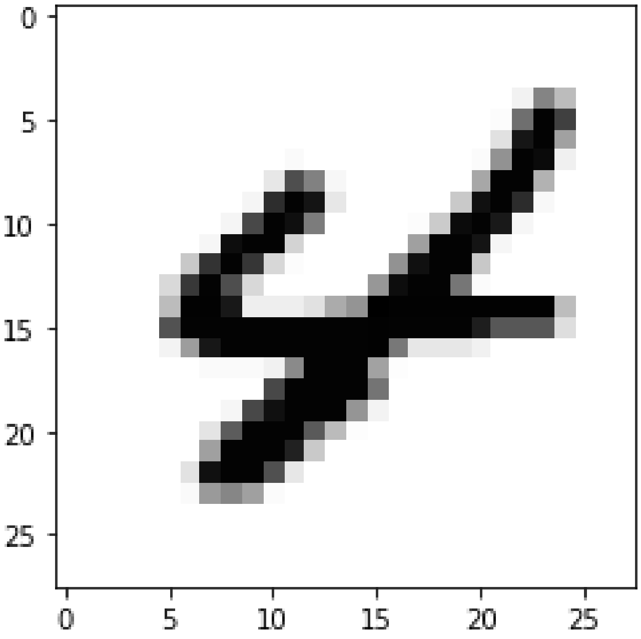
[11]:

```
image_index = 0
print(y[image_index])
plt.imshow(X[image_index].reshape(28,28), cmap='Greys')
```

```
4
```

[11]:

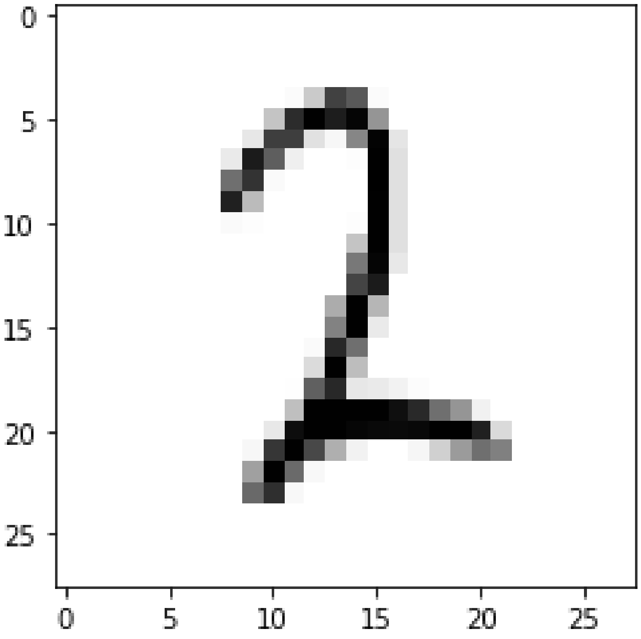
```
<matplotlib.image.AxesImage at 0x7f813cbf4e48>
```



```
[12]: image_index = 10
      print(y[image_index])
      plt.imshow(X[image_index].reshape(28,28), cmap='Greys')
```

```
2
```

```
[12]: <matplotlib.image.AxesImage at 0x7f813cb8e668>
```



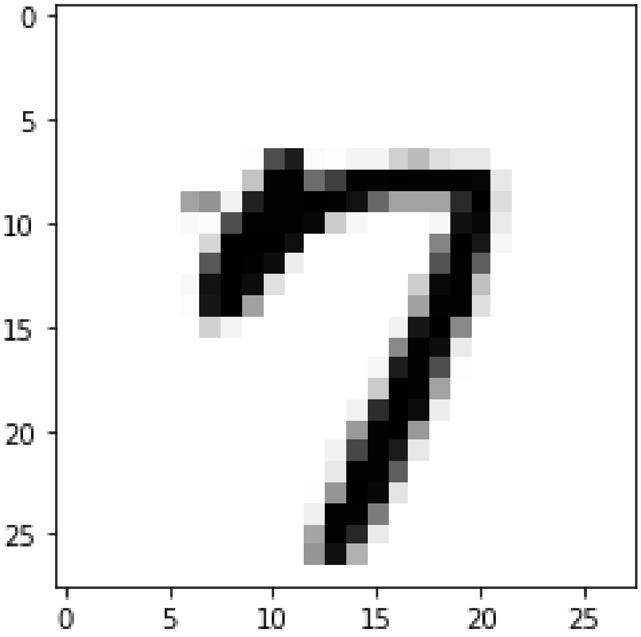
[13]:

```python
image_index = 100
print(y[image_index])
plt.imshow(X[image_index].reshape(28,28), cmap='Greys')
```

```
7
```

[13]:

```
<matplotlib.image.AxesImage at 0x7f813c629c50>
```



0.6 Train-Test Split

[14]:

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42, stratify=np.array(y))
```

0.7 Normalization

[16]:

```python
# x_train[0]
```

[17]:

```python
x_train /= 255
x_test /= 255
```

[19]:

```python
# x_train[0]
```

0.8 Model Creation

[20]:

```
input_shape = (28,28,1)
output_class = 10
```

[23]:

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

# define the model
model = Sequential()
model.add(Conv2D(28, kernel_size=(3,3), input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(128, activation=tf.nn.relu))
model.add(Dropout(0.3))
model.add(Dense(output_class, activation=tf.nn.softmax))

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',␣
  ↪metrics='accuracy')
```

[24]:

```
# train the model
model.fit(x=x_train, y=y_train, batch_size=32, epochs=30,␣
  ↪validation_data=(x_test, y_test))
```

```
Epoch 1/30
1149/1149 [==============================] - 10s 3ms/step - loss: 0.4816 -
accuracy: 0.8475 - val_loss: 0.1202 - val_accuracy: 0.9637
Epoch 2/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.1336 -
accuracy: 0.9605 - val_loss: 0.0848 - val_accuracy: 0.9743
Epoch 3/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0863 -
accuracy: 0.9732 - val_loss: 0.0807 - val_accuracy: 0.9742
Epoch 4/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0685 -
accuracy: 0.9783 - val_loss: 0.0734 - val_accuracy: 0.9788
Epoch 5/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0543 -
accuracy: 0.9825 - val_loss: 0.0690 - val_accuracy: 0.9809
Epoch 6/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0461 -
accuracy: 0.9844 - val_loss: 0.0684 - val_accuracy: 0.9808
Epoch 7/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0360 -
accuracy: 0.9873 - val_loss: 0.0743 - val_accuracy: 0.9798
Epoch 8/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0318 -
accuracy: 0.9884 - val_loss: 0.0733 - val_accuracy: 0.9811
```

```
Epoch 9/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0319 -
accuracy: 0.9891 - val_loss: 0.0658 - val_accuracy: 0.9838
Epoch 10/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0242 -
accuracy: 0.9919 - val_loss: 0.0728 - val_accuracy: 0.9827
Epoch 11/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0218 -
accuracy: 0.9926 - val_loss: 0.0815 - val_accuracy: 0.9818
Epoch 12/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0286 -
accuracy: 0.9895 - val_loss: 0.0766 - val_accuracy: 0.9829
Epoch 13/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0199 -
accuracy: 0.9928 - val_loss: 0.0762 - val_accuracy: 0.9820
Epoch 14/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0239 -
accuracy: 0.9918 - val_loss: 0.0754 - val_accuracy: 0.9836
Epoch 15/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0160 -
accuracy: 0.9938 - val_loss: 0.0865 - val_accuracy: 0.9820
Epoch 16/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0196 -
accuracy: 0.9935 - val_loss: 0.0842 - val_accuracy: 0.9822
Epoch 17/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0152 -
accuracy: 0.9951 - val_loss: 0.0825 - val_accuracy: 0.9828
Epoch 18/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0155 -
accuracy: 0.9943 - val_loss: 0.0889 - val_accuracy: 0.9817
Epoch 19/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0207 -
accuracy: 0.9930 - val_loss: 0.0886 - val_accuracy: 0.9822
Epoch 20/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0122 -
accuracy: 0.9955 - val_loss: 0.0958 - val_accuracy: 0.9822
Epoch 21/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0135 -
accuracy: 0.9957 - val_loss: 0.0986 - val_accuracy: 0.9824
Epoch 22/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0166 -
accuracy: 0.9949 - val_loss: 0.0987 - val_accuracy: 0.9824
Epoch 23/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0153 -
accuracy: 0.9949 - val_loss: 0.0917 - val_accuracy: 0.9832
Epoch 24/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0147 -
accuracy: 0.9950 - val_loss: 0.0967 - val_accuracy: 0.9838
```

```
Epoch 25/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0112 -
accuracy: 0.9957 - val_loss: 0.1057 - val_accuracy: 0.9816
Epoch 26/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0134 -
accuracy: 0.9959 - val_loss: 0.1024 - val_accuracy: 0.9830
Epoch 27/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0085 -
accuracy: 0.9968 - val_loss: 0.1256 - val_accuracy: 0.9795
Epoch 28/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0127 -
accuracy: 0.9958 - val_loss: 0.1099 - val_accuracy: 0.9832
Epoch 29/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0136 -
accuracy: 0.9952 - val_loss: 0.1043 - val_accuracy: 0.9824
Epoch 30/30
1149/1149 [==============================] - 4s 3ms/step - loss: 0.0132 -
accuracy: 0.9959 - val_loss: 0.1162 - val_accuracy: 0.9827
```

[24]:

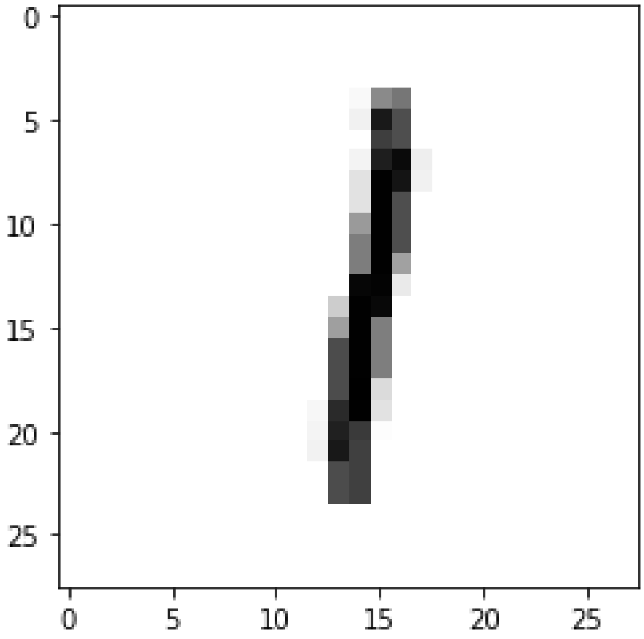
```
<tensorflow.python.keras.callbacks.History at 0x7f80dc057278>
```

0.9 Testing the model

[27]:

```python
image_index = 10
# print("Original output:",y_test[image_index])
plt.imshow(x_test[image_index].reshape(28,28), cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1,28,28,1))
print("Predicted output:", pred.argmax())
```

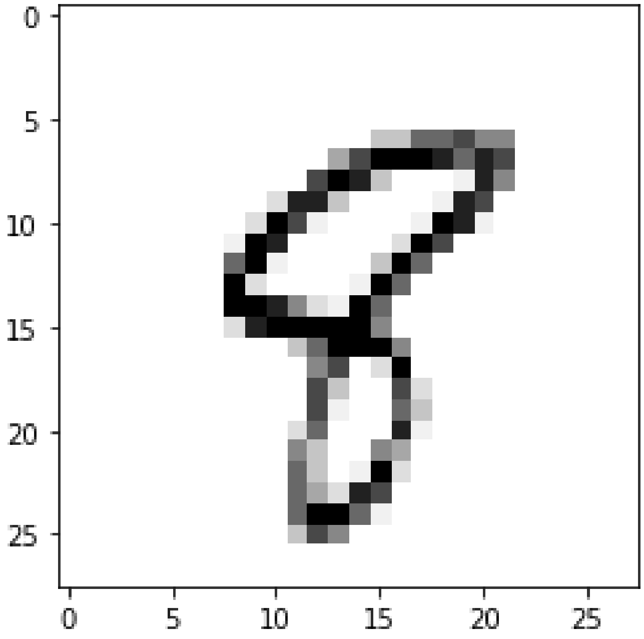
```
Predicted output: 1
```



[28]:

```
image_index = 100
# print("Original output:",y_test[image_index])
plt.imshow(x_test[image_index].reshape(28,28), cmap='Greys')
pred = model.predict(x_test[image_index].reshape(1,28,28,1))
print("Predicted output:", pred.argmax())
```

```
Predicted output: 8
```



9

[ ]: