SOURCE CODE

MODEL CREATION

```
#importing
libraries
import keras
from keras.datasets
import mnistimport
numpy as np
from keras.models import Sequential

from keras.layers.core import Dense,
Activation, Flatten, Dropoutfrom
keras.layers.convolutional import Conv2D
from keras.layers.pooling import
MaxPooling2Dfrom keras import
regularizers
from keras import metrics

from keras.utils.np_utils import
to_categoricalfrom keras import
optimizers
from scipy
import misc
import numpy
as np
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

Downloading data from
https://storage.googleapis.com/tensorflow/tfkeras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s
0us/step

print(X_train.shape,
Y_train.shape)(60000,
28, 28) (60000,)
# reshape the data so as to fit the format of (samples, height,
width, channels)X_train = X_train.reshape(60000, 28, 28,
1).astype('float32')
X_test = X_test.reshape(10000, 28, 28,
```

```
1).astype('float32') Y_train =
Y_train.reshape(60000)
 Y_test =
Y_test.reshape(10000)
Y_train =
to_categorical(Y_train,
10) Y_test =
to_categorical(Y_test,
10) # MODEL
DEFINITION
 model = Sequential()
model.add(Conv2D(filters=20,
kernel_size=(6,6),
```

```
kernel_regularizer=regularizers.l2(0.04),
strides=(1,1), padding='valid',
activation='relu',
data_format='channels_last',
input_shape=(28,28,1)))
model.add(Conv2D(filters=20,
kernel_size=(3,3),
kernel_regularizer=regularizers.l2(0.04),
strides=(1,1), padding='valid',
activation='relu'))
model.add(MaxPooling2D(pool_size=(4,4
), strides=(1,1)))
model.add(Dropout(rate=0.05,seed=3))
model.add(Conv2D(filters=10,
kernel_size=(6,6),
kernel_regularizer=regularizers.l2(0.04),
strides=(1,1), padding='valid',
activation='relu'))
model.add(Conv2D(filters=10,
kernel_size=(3,3),
kernel_regularizer=regularizers.l2(0.04),
strides=(1,1), padding='valid',
activation='relu'))
model.add(MaxPooling2D(pool_size=(4,4
), strides=(1,1)))
model.add(Dropout(rate=0.05,seed=8))
model.add(Flatten())
 model.add(Dense(units=30, activation='tanh',
kernel_regularizer=regularizers.l2(0.04)))
model.add(Dense(units=10,
activation='softmax',
kernel_regularizer=regularizers.l2(0.04)))
# MODEL COMPILATION

# reduce the learning rate if training accuracy suddenly drops
and keeps decreasing sgd = optimizers.SGD(lr=0.003) # lr by
default is 0.01 for SGD
model.compile(loss='categorical_crossentropy', optimizer=sgd,
metrics=[metrics.categorical_accuracy])

/usr/local/lib/python3.7/dist-packages/keras/optimizers/optimizer_v2/
gradient_descent.py:108: UserWarning: The `lr` argument is deprecated,
```

```
use `learning_rate` instead. super(SGD,
self).__init__(name, **kwargs)

# MODEL FIT model.fit(X_train, Y_train,
epochs=5, batch_size=50)model.save('mnist-
classifier-model.h5')
model.save_weights('mnist-classifier-
weights.h5')
```

```
Epoch 1/5
1200/1200 [==============================] - 16s 6ms/step - loss:
5.0627 - categorical_accuracy: 0.5675 Epoch 2/5
1200/1200 [==============================] - 5s 4ms/step - loss:
2.6873 - categorical_accuracy: 0.9143 Epoch 3/5
1200/1200 [==============================] - 5s 4ms/step - loss:
1.7160 - categorical_accuracy: 0.9505 Epoch 4/5
1200/1200 [==============================] - 5s 4ms/step - loss:
1.2230 - categorical_accuracy: 0.9613 Epoch 5/5
1200/1200 [==============================] - 5s 4ms/step - loss: 0.9486 - categorical_accurac
y: 0.9666 # MODEL EVALUATION
print("\nEvaluating the model on test data. This won't take long. Relax!") test_loss, test_accuracy = model.evaluate(X_test, Y_test, batch_size=10)
print("\nAccuracy on test data : ", test_accuracy*100)
print("\nLoss on test data : ", test_loss)
Evaluating the model on test data. This won't take long. Relax!
1000/1000 [==============================] - 3s 3ms/step - loss: 0.8243 - categorical_accuracy: 0.9770 Accuracy on test data : 97.69999980926514
Loss on test data : 0.8242666125297546
#TEST THE MODEL
prediction =
```

```
model.predict(X_test[:4])
print(prediction)
1/1 [==============================] - 0s 146ms/step

[[0.00552364 0.01395816 0.01411558 0.03165388 0.00503101
 0.01049465 0.00091657 0.8815433
0.009309 0.02745431]

[0.04030754 0.03318598 0.82771784 0.01982366 0.00735839
 0.00482082 0.02058108 0.01209613
0.02671191 0.00739667]
```

```
[0.00538969 0.9012829 0.0158494 0.00913932 0.01340986
 0.00374549 0.01759027 0.01883631
0.0076736 0.00708309]

 [0.8935922 0.00533478 0.01120698 0.00872513 0.00371715
 0.01620317 0.03634179 0.00775124
0.00643945 0.0106882 ]]

 print(np.argmax(prediction,
 axis=1)) print(Y_test[:4])
[7 2 1 0]

[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]

 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]

[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]

[1. 0. 0. 0. 0. 0. 0.
 0. 0. 0.]] #SAVE
 THE MODEL
 model.save('models/CNNmnist.h5')
```

HOME(html)

```
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Handwritten Digit Recognition</title>
    <link rel="icon" type="image/svg" sizes="32x32" href="{{url_for('static'
,filename='images/icon.svg')}}" />
    <link rel="stylesheet" href="{{url_for('static', filename='css/main.css')}}" />
    <script src="https://unpkg.com/feather-icons"></script>
    <script defer src="{{url_for('static' ,filename="js/script.js")}}"></script>
</head>
<body>
<div class="container">
<div class="heading">
<h1 class="heading_main"> Handwritten Digit Recognizer</h1>
<h2 class="heading_sub">Detect handwritten digits</h2>
</div>
<div class="upload-container">
<div class="form-wrapper">
<form class="upload" action="predict.html" method="post" enctype="multipart/form-
data">
    <input type="file" id="file-uploader">
<label id="upload-image"><i data-feather="file-plus"></i>Select File</label>
    <input type="file" name="photo" id="upload-image" hidden />
<input type="submit" value = "submit" id="up_btn">


</form>
<img id="loading" src="{{url_for('static', filename='images/loading.gif')}}">
</div>
</div>
</div>
```

```
</body>

</html>
```

PREDICT(html)

```
<html>

<head>

<title>Predict | Handwritten Digit Recognition</title>

<link rel="stylesheet" href="{{url_for( 'static',filename='css/predict.css')}}" />

<link rel="icon" type="image/svg" sizes="32x32" href="{{url_for('static',
filename='images/icon.svg')}}" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

</head>

<body>

<div class="container">

<h1>Prediction</h1>

<div class="result-wrapper">

<div class="input-image-container">

<img src="{{url_for('static',filename='data/')}}{{img_name}}" />

</div>

<div class="result-container">

<div class="value">{{best.0}}</div>

<div class="accuracy">{{ best.1}}%</div>

</div>

</div>

<h1>Other Prediction </h1>

<div class="other_predictions">

{% for x in others %}

<div class="value">

<h2>{{x.0}}</h2>

<div class="accuracy">{{x.1}}%</div>
```

```
{% endfor %}

</div>

</div>

</body>


</html>
```

APP(py)

```
from flask import Flask, render_template, request
from recognizer import recognize
app = Flask(_name_)
@app.route('/')
def main():
    return render_template("home.html")



@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        image = request.files.get('photo', '')
        best, others, img_name = recognize(image)
        return render_template("predict.html", best=best, others=others,
img_name=img_name)



if__name__== "__main__":
    app.run()
```

RECOGNIZER(py)

```
import os
import random
import string
from pathlib import Path
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image,ImageOps
def random_name_generator(n):

    return ''.join(random.choices(string.ascii_uppercase + string.digits, k=n))
def recognize(image):
    model = load_model(Path("./model/CNNmnist.h5"))
    img = Image.open(image).convert("L")
    img_name = random_name_generator(10) + '.jpg'
    if not os.path.exists(f"./static/data/"):
        os.mkdir(os.path.join('./static/', 'data'))
    img.save(Path(f"./static/data/{img_name}"))
    img = ImageOps.grayscale(img)
    img = ImageOps.invert(img)
    img = img.resize((28, 28))
    img2arr = np.array(img)
    img2arr = img2arr / 255.0
    img2arr = img2arr.reshape(1, 28, 28, 1)
    results = model.predict(img2arr)
    best = np.argmax(results, axis=1)[0]

    pred = list(map(lambda x: round(x * 100, 2), results[0]))
    values = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    others = list(zip(values, pred))
    best = others.pop(best)
    return best, others, img_name
```

HOME PAGE(CSS)

```
clear_button{
margin-left: 15px;
font-weight: bold;
color: blue;
}

#confidence{
font-family: 'Josefin Sans', sans-serif; margin-top: 7.5%;
}

#content{
margin: 0 auto;
padding: 2% 15%;
padding-bottom: 0;
}

.welcome{
text-align:       center;
position:       relative;
color: honeydew;
background-color: greenyellow;
padding-top: 1%;
padding-bottom: 1%;
font-weight: bold;
font-family: 'Prompt', sans-serif;
}

#team_id{
text-align: right;font-
size: 25px; padding-
right: 3%;
}

#predict_button{
margin-right: 15px;
color: blue;
font-weight: bold;
}

#prediction_heading{
font-family: 'Josefin Sans', sans-serif; margin-top: 7.5%;
```

```
}

#result{
font-size: 5rem;
}

#title{
padding: 1.5% 15%;
margin: 0 auto; text-
align: center;
}

.btn {
font-size: 15px;
padding: 10px;
-webkit-appearance: none;
```

```
background: #eee; border:
1px solid #888;margin-top:
20px; margin-bottom:
20px;
 }

 .buttons_div{ margin-
 bottom: 30px;margin-
 right: 80px;
 }

 .heading{
 font-family: 'Varela Round', sans-serif; font-weight: 700;font-size:
 2rem;
 display: inline;
 }

 .leftside{
 text-align: center;
 margin: 0 auto;
 margin-top: 2%;
 /* padding-left: 10%; */
 }

 #frame{
 margin-right: 10%;
 }

 .predicted_answer{
 text-align: center;
 margin: 0 auto;
 padding: 3% 5%;
 padding-top: 0;
 /* padding-left: 10%; */
 }

 p{
 font-family: 'Source Code Pro', monospace,sans-serif; margin-top: 1%;
 }

 @media (min-width: 720px) {
 .leftside{ padding-
 left: 10%;
 }
```

```
}
```

HOME PAGE(JS)

```
// feather.replace();

//              form = document.querySelector('.upload')

//              loading = document.querySelector("#loading")

//              select = document .querySelector( "#upload-image");

//              select.addEventListener( "change",(e) => {

//                e.preventDefault();

//                form.submit()

//                form.style.visibility = "hidden";

//                loading.style.display = 'flex';

//

//              });
```

```
const fileUploader = document.getElementById('file-
uploader');fileUploader.addEventListener('change',
(event) => {
 const files = event.target.files;
 window.location.href =
 "/predict.html";console.log('files',
 files);
});
```