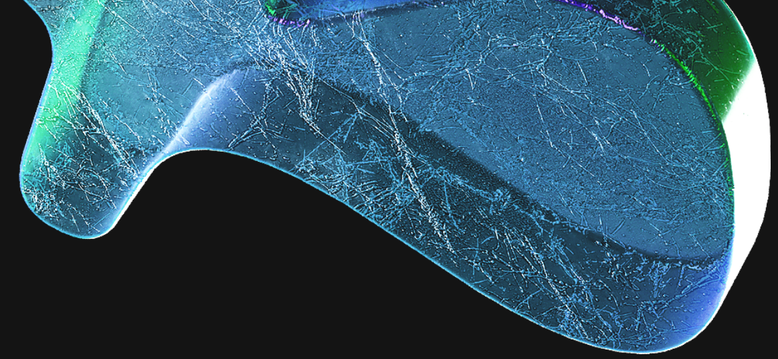
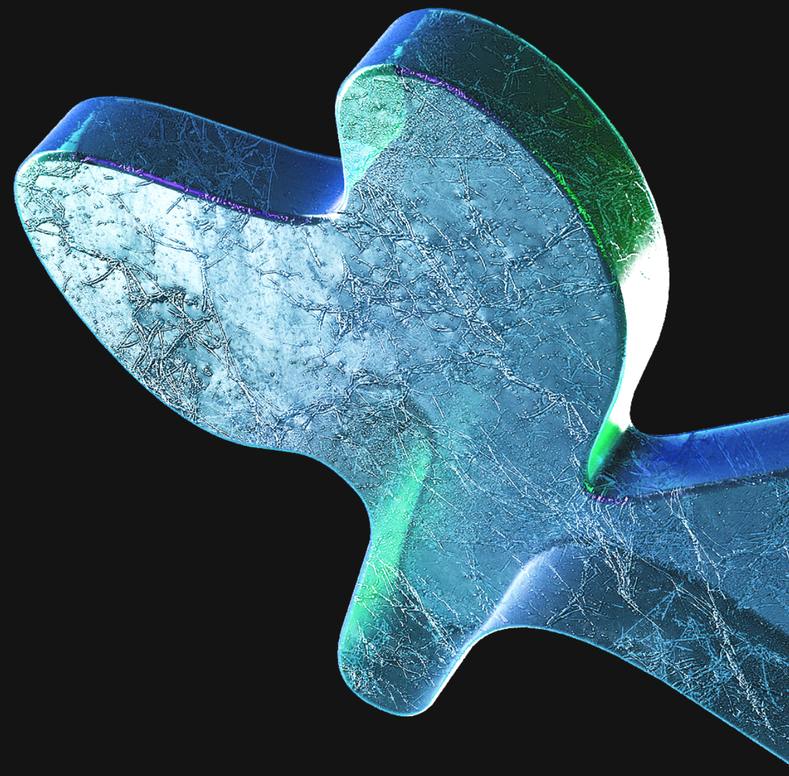
Computer Vision Final Project

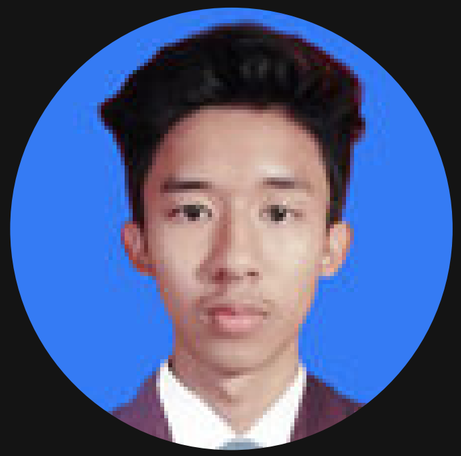
Drawable Digit Recognition



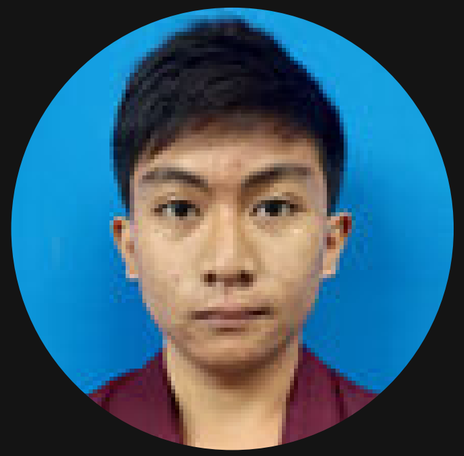


COMP7116001 - LB01 - LEC

Team



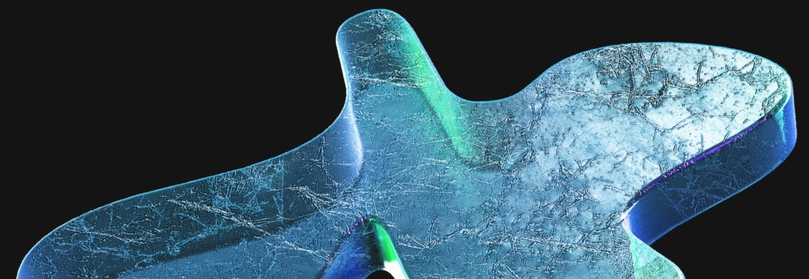
2440070642
FADLY HAIKAL FASYA



2440018854
GERRY WILLIAM NANLOHY

What you need to know

about Digit Recognition



- What is Digit Recognition?
- Main Background
- Challenges
- Libraries
- Framework & API
- Modules
- Dataset
- Data Collecting
- Pre-Processing
- Feature Extraction
- Classification
- Result

What is Digit Recognition?

Digit Recognition is the ability of computers to recognize digits/numbers from human handwriting

What is the background?

Handwritten character recognition is one of the practically important issues in pattern recognition applications. The applications of digit recognition include postal mail sorting, bank check processing, data entry forms, etc.

What are the challenges?

DATASET

Handmade Dataset
(0-1-2-3-4-5-6-7-8-9)

METHOD/ALGORITHM

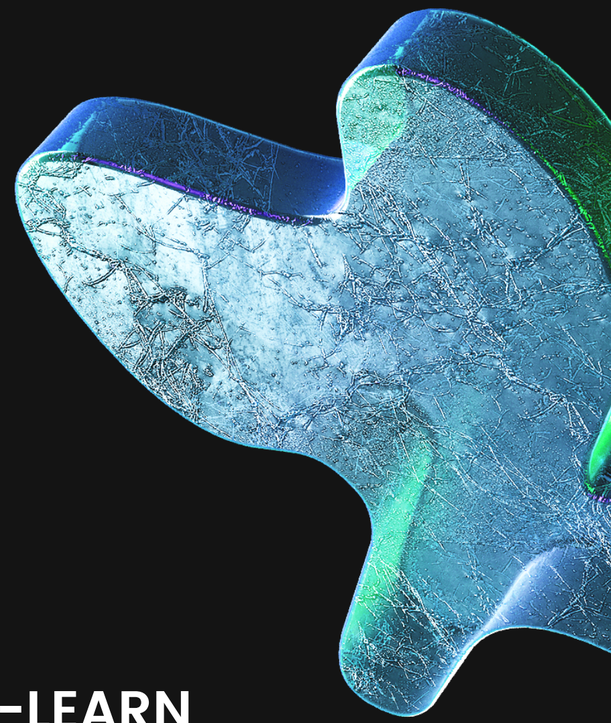
KNN - SVM - RANDOM FOREST -
Tensorflow Keras

ACCURACY

DIFFERENCES IN THE ACCURACY OF
EACH METHOD/ALGORITHM

What are the libraries?

Python Libraries



NUMPY

MATPLOTLIB

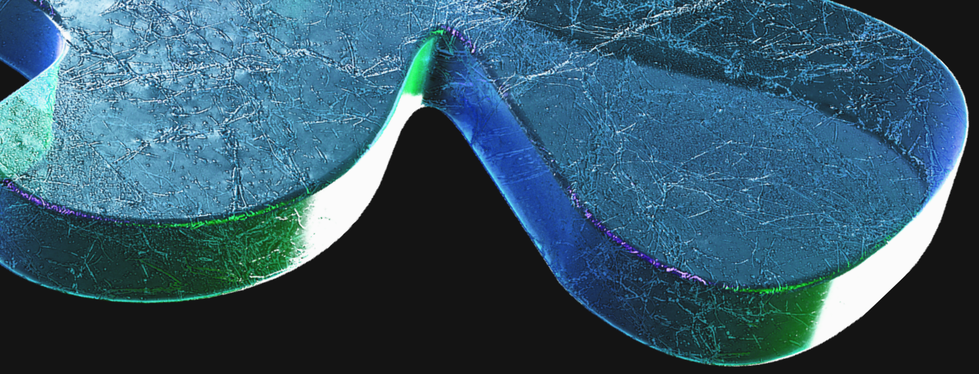
SCIKIT-LEARN

OPENCV

PILLOW

TENSORFLOW

KERAS



What are the framework & API?

Python Framework & API

STREAMLIT

INTERACTIVE WEB - INTERACTIVE VISULIZATION DATA

PYNGROK

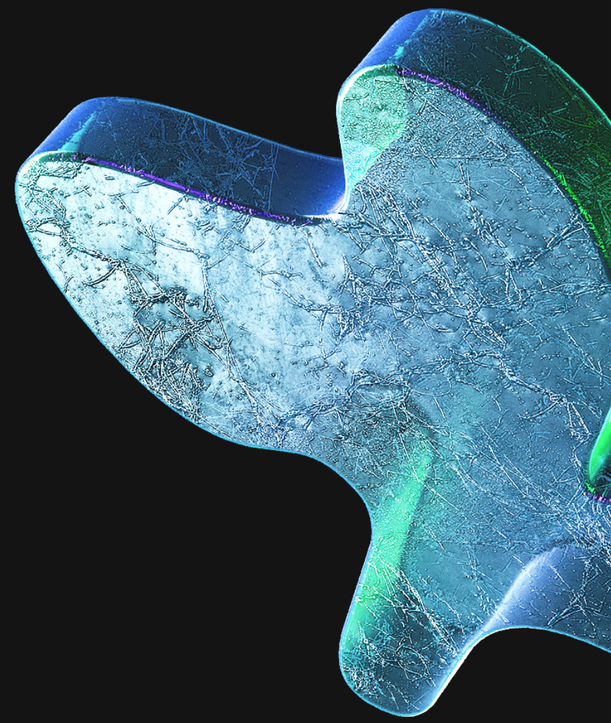
Local Web Server - Building Webhook Integrations - Enabling SSH Access - Testing Chatbots - Demo App - Integration

LOCALTUNNEL

Web Local Services - Unique Accessible URL

What are the module?

Python Modules



UUID

Unique Random ID - 128 BIT ID
Converter - Hardware ID - MAC ID
- Generate Random
Documents/Address

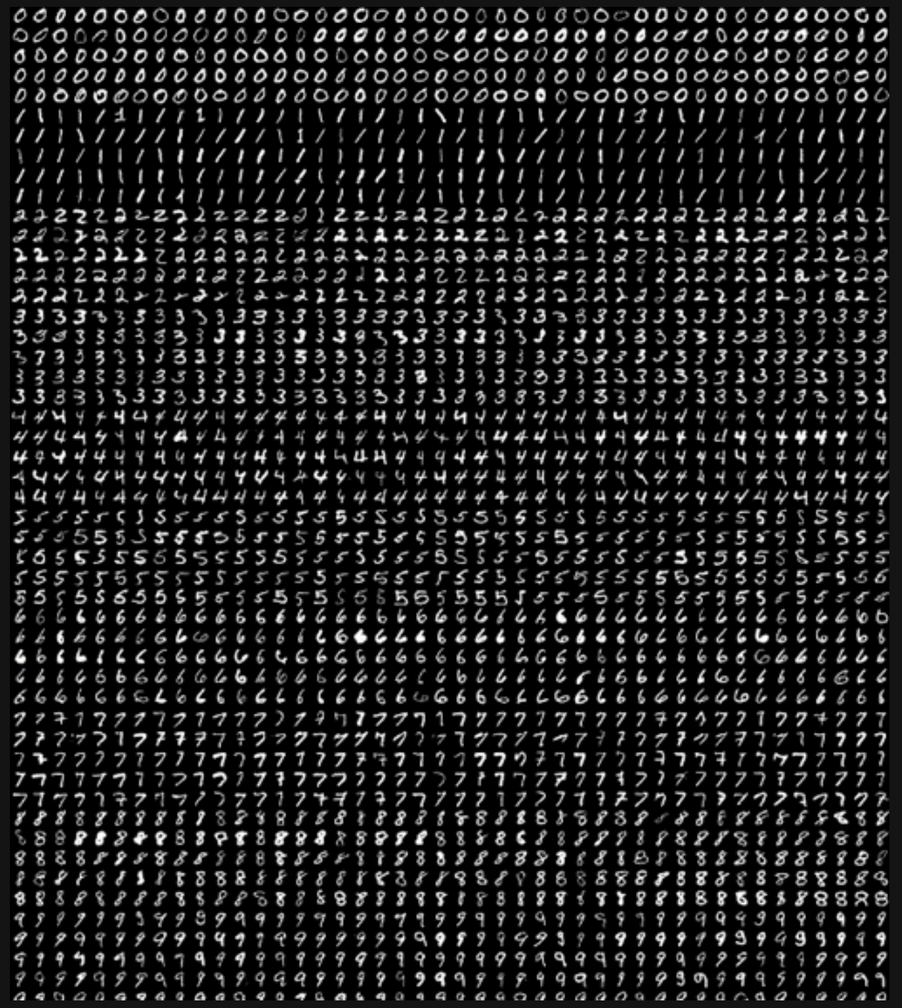
IO PYTHON

Input & Ouput Operation -
UnNICODE Data

Classes: BytesIO & StringIO

Dataset

MNIST



Handwritten Digit Dataset

Handmade

Data Collecting

```
pip install pillow
```

run the code “pip install pillow” to import the pillow library into the program

```
!pip install streamlit==1.13.0
!npm install -g localtunnel
```

run the code “!pip install streamlit==1.13.0” to import the streamlit framework and “!npm install -g localtunnel” to call the localtunnel module into the program

```
!pip install streamlit-drawable-canvas
```

run the code “!pip install streamlit drawable-canvas” to import the streamlit framework

Data Collecting

Streamlit App

```
%%writefile app.py
import numpy as np
import streamlit as st
from streamlit_drawable_canvas import st_canvas
import cv2
import joblib

st.title("Drawable Digit Recognition using Support Vector Machine")
st.markdown("> This app demonstrates Digit Recognition using SVM. The app allows
st.markdown(">Disclaimer : The performance of our SVM Model on Drawable Digit Re
st.header("🎲 How to Use the Application")
st.markdown(
    """
    * Draw Digits freely as you want!
    * You can Undo, Redo or Delete your drawing with button in the bottom left o
    * Please Don't forget to Press ⬇️ Send Button in the bottom left of the canv
    * Press Predict Button if you're done with your drawing and the result will
    """
    )
st.write(" ")
st.write(" ")
st.write(" ")

st.markdown("Draw Digits [0-9]")
# Create a canvas component
canvas_result = st_canvas(
    fill_color="rgba(255, 165, 0, 0.3)",  # Fixed fill color with some opacity
    stroke_width=9,
    stroke_color="black",
```

Streamlit Application

Streamlit Create Dataset

```
[ ] %%writefile app.py
    import numpy as np
    import streamlit as st
    from streamlit_drawable_canvas import st_canvas
    import cv2
    import uuid
    from io import BytesIO, BufferedReader


    st.title("Drawable Canvas for Digit Recognition Dataset")
    st.markdown(">The app lets users draw digit on to a canvas and input label of th
    st.header("🎲 How to Use the Application")
    st.markdown(
        """
        * Draw Digit as your desire
        * You can Undo, Redo or Delete your drawing with button in the bottom left o
        * Please Don't forget to Press ⬇️ Send Button in the bottom left of the canv
        * Input the Drawed Digit label in the Label Section!
        * Press Input Button if you're done with your drawing and the inputted data
        * Now download Button will be appear!
        * Press download Button and the drawed digit image will be downloaded for yo
        """
        )
    st.write(" ")
    st.write(" ")
    st.write(" ")

    st.markdown("Draw Digit [0-9]")
    # Create a canvas component
```

Streamlit Create Dataset

Data Collecting

Local Tunnel

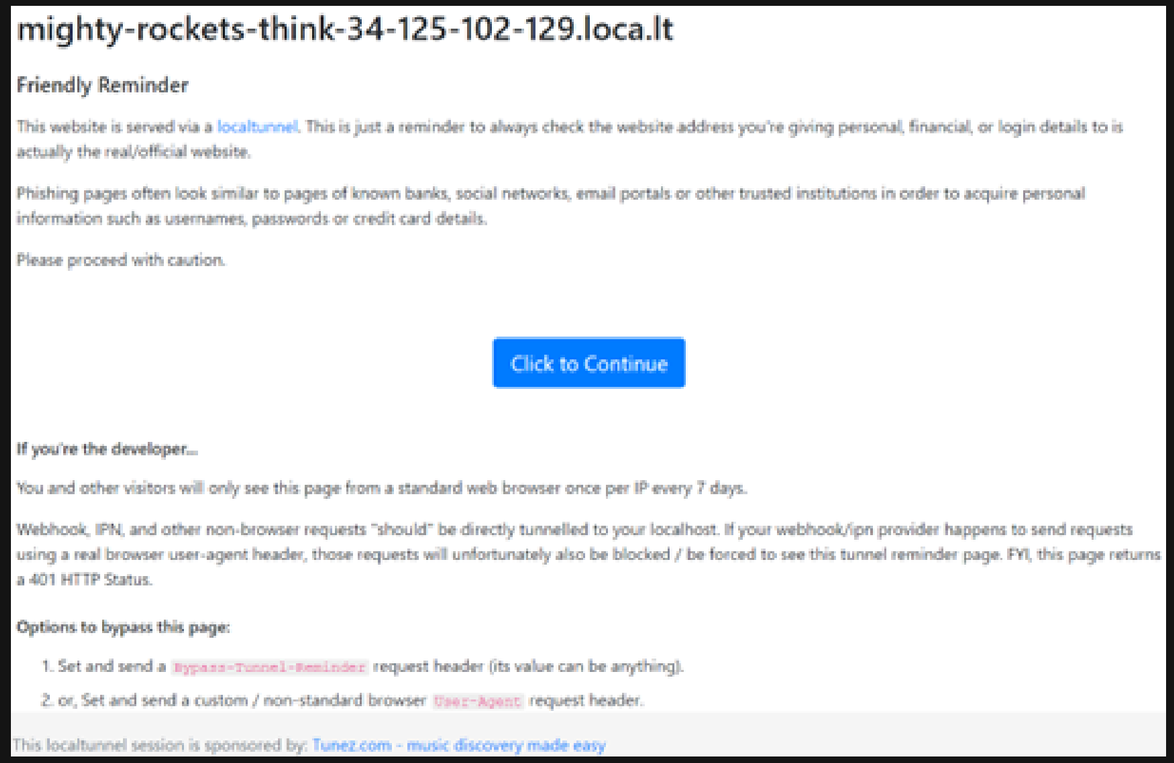
```
!streamlit run app.py & npx localtunnel --port 8501
```

run the code “!streamlit run app.py & npx localtunnel --port 8501” in the "Local Tunnel" column to run the streamlit application on the local web in the browser

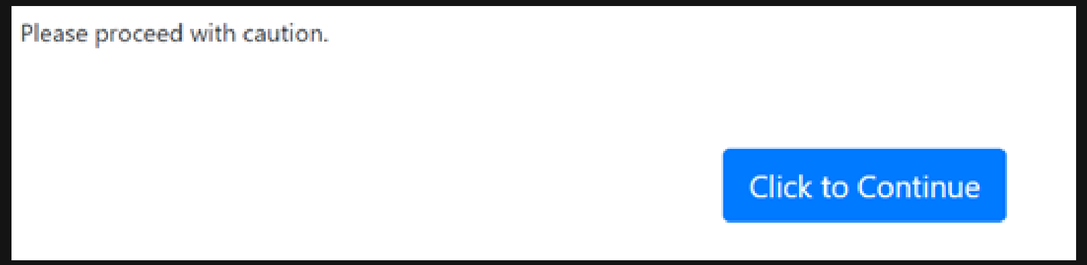
```
2023-01-15 15:45:23.278 INFO    numexpr.utils: NumExpr defaulting to 2 threads.
[#######...........] / extract:localtunnel: verb lock using /root/.npm/_locks/s
Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


  You can now view your Streamlit app in your browser.

  Network URL: http://172.28.0.2:8501
  External URL: http://34.125.102.129:8501

npx: installed 22 in 3.254s
your url is: https://mighty-rockets-think-34-125-102-129.loca.lt
```

Localtunnel URL

Data Collecting



Localtunnel Local Website



Continue Column

Data Collecting

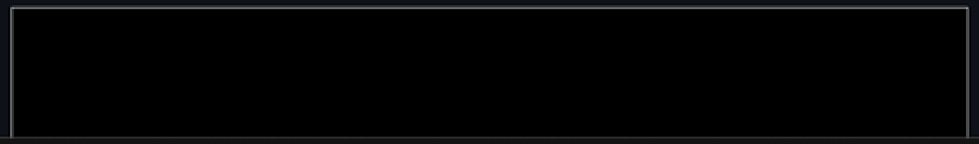
Drawable Canvas for Digit Recognition Dataset

> The app lets users draw digit on to a canvas and input label of the drawed digit. Then it will show the inputted data and user can download drawed digit image for Digit Recognition Dataset

How to Use the Application

- Draw Digit as your desire
- You can Undo, Redo or Delete your drawing with button in the bottom left of the canvas
- Please Don't forget to Press ⬇️ Send Button in the bottom left of the canvas!
- Input the Drawed Digit label in the Label Section!
- Press Input Button if you're done with your drawing and the inputted data will be shown bellow
- Now download Button will be appear!
- Press download Button and the drawed digit image will be downloaded for your dataset!

Draw Digit [0-9]



Drawable Canvas on Streamlit Application

Draw Digit [0-9]

Draw The Digit

Inputted Data

Label : 0123

Download the Image

Download Digit Image for Dataset

Pre-Processing

```python
def recognize_digit(image):
  gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
  blur = cv2.GaussianBlur(gray, (5,5), 0)

  canny = cv2.Canny(blur, 30, 150)

  contours, _ = cv2.findContours(canny.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  boundingBoxes = [cv2.boundingRect(c) for c in contours]

  display = []
  image_numbers = []

  for contour in contours:
    (x, y, w, h) = cv2.boundingRect(contour)

    if w>= 5 and h>=20:
      area = blur[y:y+h, x:x+w]
      ret, area = cv2.threshold(area, 127, 255, cv2.THRESH_BINARY_INV)

      new_square = drawSquare(area)
      number = resize(new_square, 100)
      result = number.reshape((1, 10000));
      # number = resize(new_square, 20)
      # result = number.reshape((1, 400));
      result = result.astype(np.float32);
      image_numbers.append(number)
```

Pre-Processing Image

```python
      #knn opencv
      # ret, res, neighbours, distance = model_cv_knn.findNearest(result, k=1)
      # n = str(int(float(res[0]))); print(res.shape)
      # display.append(n)

      #svm
      # res = model_svm_mnist_new.predict(result)
      res = model_svm_manual_new.predict(result)
      n = str(int(float(res)))
      display.append(n)

      #tensor keras
      # result = np.array(result).astype(np.float32)
      # res = model_tensor.predict(result)
      # n = str(int(float(res[0][0])))
      # display.append(n)

      # draw rectangle around individual digit
      cv2.rectangle(image, (x,y), (x+w, y+h), (255,0,0), 1)
      cv2.putText(image, n, (x,y-10), cv2.FONT_ITALIC, 2, (0,255,0), 2)

  return image, display, image_numbers
```

Create Model

Feature Extraction

```python
# Loading the digits data
data = cv2.imread('./drive/MyDrive/DigitRecognition_Dataset/digits.png')
gray = cv2.cvtColor(data, cv2.COLOR_BGR2GRAY)

# Resizing each digit from 20x20 to 10x10
resized = cv2.pyrDown(gray)

# Splitting image original image into 5000 different arrays of size 20x20
# Resulting array: 50 * 100 * 20 * 20
arr = [np.hsplit(i, 100) for i in np.vsplit(gray, 50)]
arr = np.array(arr)

# Spliting into training and test set
# Total: 5000, Train: 3500 images, Test: 1500
X_train_mnist = arr[:, :70].reshape(-1, 400).astype(np.float32)
X_test_mnist = arr[:, 70:100].reshape(-1, 400).astype(np.float32)

## Targets for each image
y = [0,1,2,3,4,5,6,7,8,9]

y_train_mnist = np.repeat(y, 350)[:, np.newaxis]
y_test_mnist = np.repeat(y, 150)[:, np.newaxis]
```

MNIST Dataset

```python
train_root_path = './drive/MyDrive/DigitRecognition_Dataset/Train'
train_names = get_path_list(train_root_path)
train_image_list,
image_classes_list,
image_blur_list = get_class_id(train_root_path, train_names)

(X_train, X_test, y_train, y_test) = train_test_split(train_image_list,
                                                      image_classes_list,
                                                      test_size=0.25,
                                                      random_state=42)
```

DigitRecognition_Dataset

Classification

KNN with OpenCV Algorithm

```
# Using K-NN(k- nearest neighbors) as the ML algorithm
classifier_knn = cv2.ml.KNearest_create()
classifier_knn.train(X_train_mnist, cv2.ml.ROW_SAMPLE, y_train_mnist)
response, result, neighbours, distance = classifier_knn.findNearest(X_test_mnist, k=3)

# Testing and calculating the accuracy of knn classifier
correct = result == y_test_mnist
correct = np.count_nonzero(correct)
accuracy = correct * (100.0/result.size)
print ("MNIST Dataset : ", accuracy)
```

```
MNIST Dataset :  93.46666666666667
```

MNIST Dataset

Handmade Dataset

```
# Using K-NN(k- nearest neighbors) as the ML algorithm
model_cv_knn = cv2.ml.KNearest_create()
model_cv_knn.train(X_train, cv2.ml.ROW_SAMPLE, y_train)
response, result, neighbours, distance = model_cv_knn.findNearest(X_test, k=3)

# Testing and calculating the accuracy of knn classifier
res_flatten = result.flatten()
correct = res_flatten == y_test
correct = np.count_nonzero(correct)
accuracy = correct * (100/res_flatten.size)
print ("Manual Dataset : ", accuracy)
```

```
Manual Dataset :  77.35849056603773
```

Classification

KNN with Scikit-Learn Algorithm

```
model_knn_sklearn = KNeighborsClassifier(n_neighbors=5,algorithm='auto',n_jobs=10)
model_knn_sklearn.fit(X_train_mnist,y_train_mnist)
confidence = model_knn_sklearn.score(X_test_mnist,y_test_mnist)
y_pred_mnist = model_knn_sklearn.predict(X_test_mnist)
accuracy = accuracy_score(y_test_mnist, y_pred_mnist)
print ("MNIST Dataset : ", accuracy*100)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/neighbors/_classification.py:198: Da
  return self._fit(X, y)
MNIST Dataset :  93.13333333333334
```

MNIST Dataset

Handmade Dataset

```
model_knn_sklearn = KNeighborsClassifier(n_neighbors=5,algorithm='auto',n_jobs=10)
model_knn_sklearn.fit(X_train,y_train)
confidence = model_knn_sklearn.score(X_test,y_test)
y_pred = model_knn_sklearn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print ("Manual Dataset : ", accuracy*100)
```

```
Manual Dataset :  78.30188679245283
```

Classification

CNN with TensorFlow Algorithm

```
model_rf = RandomForestClassifier(n_estimators=100, n_jobs=10)
model_rf.fit(X_train_mnist,y_train_mnist)
confidence = model_rf.score(X_test_mnist,y_test_mnist)
y_pred_mnist = model_rf.predict(X_test_mnist)
accuracy = accuracy_score(y_test_mnist, y_pred_mnist)
print ("MNIST Dataset : ", accuracy*100)
```

```
<ipython-input-16-d210f56e470b>:2: DataConversionWarning: A col
  model_rf.fit(X_train_mnist,y_train_mnist)
MNIST Dataset :  93.06666666666666
```

MNIST Dataset

Handmade Dataset

```
model_rf = RandomForestClassifier(n_estimators=100, n_jobs=10)
model_rf.fit(X_train,y_train)
confidence = model_rf.score(X_test,y_test)
y_pred = model_rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print ("Manual Dataset : ", accuracy*100)
```

```
Manual Dataset :  81.13207547169812
```

Classification

```
# Sequential create a stack of layers
model_tensor = keras.Sequential([
    keras.layers.Dense(10, input_shape=(400,), activation='sigmoid')
])

# Optimizer will help in backproagation to reach better global optima
model_tensor.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Does the training
model_tensor.fit(X_train_mnist, y_train_mnist, epochs=50)
model_tensor.evaluate(X_test_mnist, y_test_mnist)
```

CNN with TensorFlow Algorithm

Classification

SVM Algorithm

```
model_svm_mnist = svm.SVC(gamma=0.001, kernel='poly')
model_svm_mnist.fit(X_train_mnist,y_train_mnist)
y_pred_mnist = model_svm_mnist.predict(X_test_mnist)
accuracy = accuracy_score(y_test_mnist, y_pred_mnist)
print("MNIST Dataset : ", accuracy*100)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/utils/va
  y = column_or_1d(y, warn=True)
MNIST Dataset :  93.53333333333333
```

MNIST Dataset

Handmade Dataset

```
model_svm_manual = svm.SVC(gamma=0.001, kernel='poly')
model_svm_manual.fit(X_train,y_train)
y_pred = model_svm_manual.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Manual Dataset : ", accuracy*100)
```

```
Manual Dataset :  83.9622641509434
```

Result

Comparison Result From All Method

Algorithm	Dataset	
	MNIST	DigitRecognition
KNN with OpenCV (k = 3)	93.46%	77.35%
KNN with Scikit-Learn (k=3)	93.13%	78.30%
Random Forest	93.06%	81.13%
CNN with TensorFlow (50 epochs)	86,59%	73,58%
SVM	**93.53%**	**83.96%**

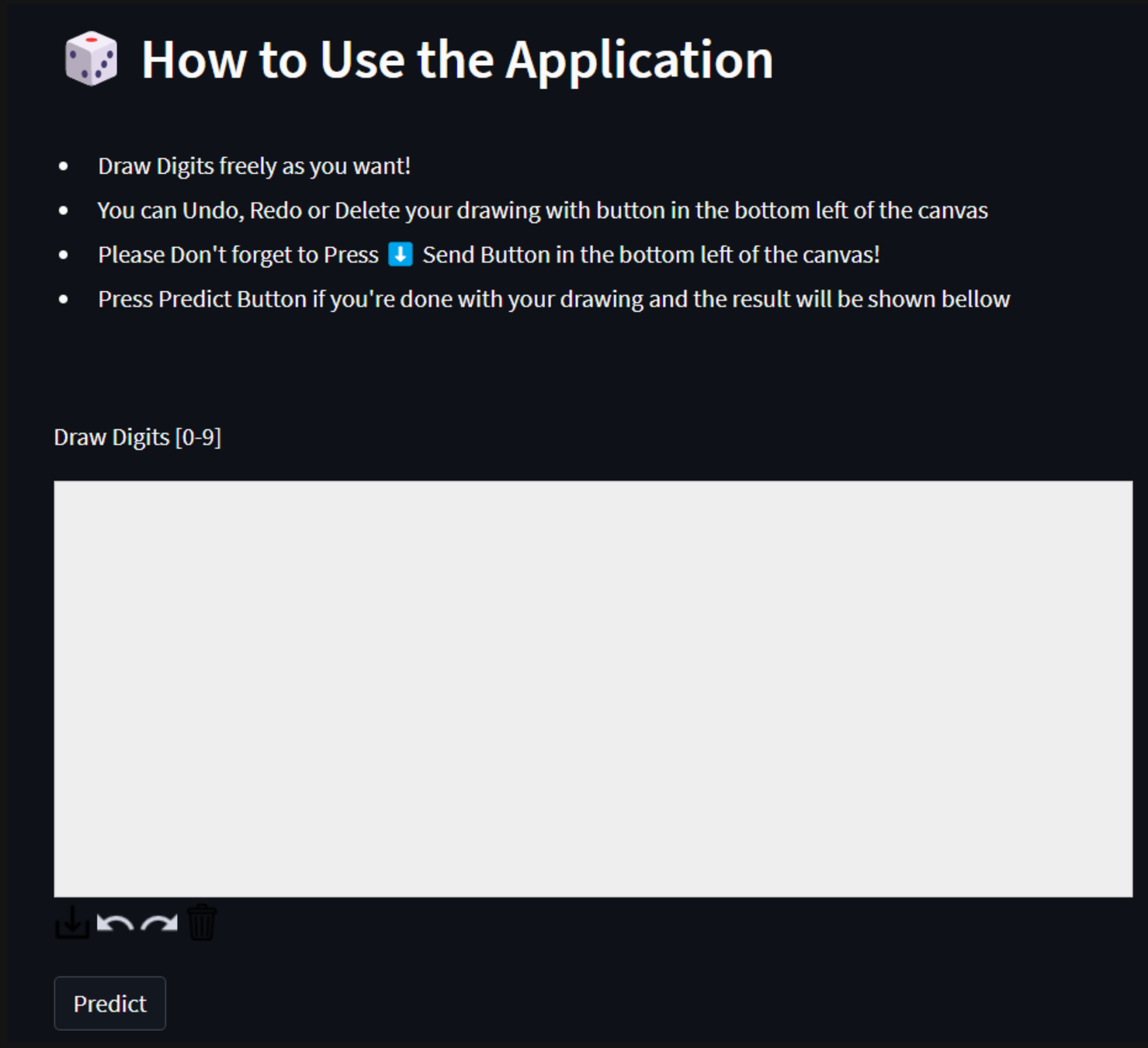
Table 1. *Comparison Table Result of First Trial*

Algorithm	Dataset	
	MNIST	DigitRecognition
KNN with OpenCV (k = 3)	93.46%	83.33%
KNN with Scikit-Learn (k=3)	93.13%	84.12%
Random Forest	93.26%	86.11%
CNN with TensorFlow (50 epochs)	86,40%	82,53%
SVM	**93.53%**	**88.09%**

Table 2. *Results Table Comparison of the Second Trial*

Result

The Drawable Canvas for Digit Recognition based on Web Application



Thank You

Do you have any questions?

https://github.com/FadlyHaikal/DigitRecognition