IMAGE PROCESSING

USKUDAR UNIVERSITY

Table of Contents

**Table of Contents1**

**Introduction2**

**Comparison table2**

**Confusion matrixes3**

**Moments4**

code5

output6

output6

**Boundaries7**

code8

output8

confusion matrix9

**Moment + Boundary10**

code11

output11

confusion matrix12

**Introduction**

In this project, we compare two methods for classifying handwritten digits. The first method uses moment-based features, capturing spatial characteristics of digits. The second method relies on boundary-based features, focusing on extracting information like bounding rectangles from digit images.

Throughout this report, we analyze the outcomes of both methods, looking at accuracy, precision, and recall. By comparing results, we aim to understand which approach is more effective for digit classification and discuss potential areas for improvement in image processing techniques.

**Comparison Table:**

In the following table, we present a comparative analysis of two methodologies utilized for handwritten digit classification, focusing specifically on precision and accuracy metrics.

|  |  |  |
| --- | --- | --- |
| Approach | Precision | Accuracy |
| Moments - based | 0,74 | 0,72 |
| Boundaries - based | 0,53 | 0,50 |
| Moments + Boundaries | 0,80 | 0,78 |

**Confusion Matrixes:**

**moment**

A diagram of a graph

Description automatically generated with medium confidenceA blue and white chart with numbers and labels

Description automatically generatedA diagram of a number and a number

Description automatically generated with medium confidence

**Moment + boundry**

**Boundaries**

**Moment**

Code:

import pandas as pd

import numpy as np

import cv2

import tensorflow as tf

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_score, recall\_score, f1\_score, confusion\_matrix, accuracy\_score

digits = datasets.load\_digits()

labels = digits.target

images = digits.images

data = []

for idx, image in enumerate(images):

    image\_uint8 = np.uint8(image)

    ret, thresh = cv2.threshold(image\_uint8, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    img\_moment = []

    for cnt in contours:

        M = cv2.moments(cnt)

        img\_moment.extend(list(M.values()))

    data.append(img\_moment)

max\_length = max(len(item) for item in data)

data = [np.pad(seq, (0, max\_length - len(seq)), 'constant') for seq in data]

df\_data = pd.DataFrame(data)

df\_data = df\_data.astype('float32')

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    df\_data, digits.target, test\_size=0.2, shuffle=False

)

import tensorflow as tf

model = tf.keras.Sequential([

    tf.keras.layers.Flatten(input\_shape=(max\_length,)),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dense(10)

])

model.compile(optimizer='adam',

              loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

              metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=100)

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=2)

print('\nTest accuracy:', test\_acc)

output:

Test accuracy: 0.7222222089767456

confusion matrix:

code:

# Make predictions

y\_pred = np.argmax(model.predict(X\_test), axis=-1)

# Calculate metrics

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

# recall = recall\_score(y\_test, y\_pred, average='weighted')

# f1 = f1\_score(y\_test, y\_pred, average='weighted')

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nConfusion Matrix:\n', conf\_matrix)

print('\nPrecision:', precision)

# print('\nRecall:', recall)

# print('\nF1-Score:', f1)

print("\nAccuracy: ", accuracy)

A diagram of a graph

Description automatically generated with medium confidenceoutput:

Precision: 0.7424966042884498

Accuracy: 0.7222222222222222

**Boundaries**

code:

import pandas as pd

import numpy as np

import cv2

import tensorflow as tf

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_score, recall\_score, f1\_score, confusion\_matrix, accuracy\_score

digits = datasets.load\_digits()

labels = digits.target

images = digits.images

data = []

for idx, image in enumerate(images):

    image\_uint8 = np.uint8(image)

    ret, thresh = cv2.threshold(image\_uint8, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    img\_Rect = []

    for cnt in contours:

        x, y, w, h = cv2.boundingRect(cnt)

        img\_Rect.extend([x, y, w, h])

    data.append(img\_Rect)

max\_length = max(len(item) for item in data)

data = [np.pad(seq, (0, max\_length - len(seq)), 'constant') for seq in data]

df\_data = pd.DataFrame(data)

df\_data = df\_data.astype('float32')

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    df\_data, digits.target, test\_size=0.2, shuffle=False

)

import tensorflow as tf

model = tf.keras.Sequential([

    tf.keras.layers.Flatten(input\_shape=(max\_length,)),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dense(10)

])

model.compile(optimizer='adam',

              loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

              metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=100)

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=2)

print('\nTest accuracy:', test\_acc)

output:

Test accuracy: 0.5055555701255798

Confusion matrix:

Code:

# Make predictions

y\_pred = np.argmax(model.predict(X\_test), axis=-1)

# Calculate metrics

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

# recall = recall\_score(y\_test, y\_pred, average='weighted')

# f1 = f1\_score(y\_test, y\_pred, average='weighted')

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nConfusion Matrix:\n', conf\_matrix)

print('\nPrecision:', precision)

# print('\nRecall:', recall)

# print('\nF1-Score:', f1)

print("\nAccuracy: ", accuracy)

A blue and white chart with numbers and labels

Description automatically generatedoutput:

Precision: 0.530443212775874

Accuracy: 0.5055555555555555

**Moment + Boundary**

Code:

import pandas as pd

import numpy as np

import cv2

import tensorflow as tf

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, accuracy\_score

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_score, recall\_score, f1\_score, confusion\_matrix, accuracy\_score

digits = datasets.load\_digits()

labels = digits.target

images = digits.images

data = []

moment\_keys = [

    "m00", "m10", "m01", "m20", "m11", "m02", "m30", "m21", "m12", "m03",

    "mu20", "mu11", "mu02", "mu30", "mu21", "mu12", "mu03",

    "nu20", "nu11", "nu02", "nu30", "nu21", "nu12", "nu03"

]

for image in images:

    image\_uint8 = np.uint8(image)

    ret, thresh = cv2.threshold(image\_uint8, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    img\_moment = []

    for cnt in contours:

        M = cv2.moments(cnt)

        img\_moment.extend([M[key] for key in moment\_keys])

        x, y, w, h = cv2.boundingRect(cnt)

        img\_moment.extend([x, y, w, h])

    data.append(img\_moment)

max\_length = max(len(item) for item in data)

data = [np.pad(seq, (0, max\_length - len(seq)), 'constant') for seq in data]

df\_data = pd.DataFrame(data)

df\_data = df\_data.astype('float32')

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    df\_data, digits.target, test\_size=0.2, shuffle=False

)

import tensorflow as tf

model = tf.keras.Sequential([

    tf.keras.layers.Flatten(input\_shape=(max\_length,)),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dense(10)

])

model.compile(optimizer='adam',

              loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

              metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=100)

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=2)

print('\nTest accuracy:', test\_acc)

output:

Test accuracy: 0.7833333611488342

Confusion matrix:

Code:

# Make predictions

y\_pred = np.argmax(model.predict(X\_test), axis=-1)

# Calculate metrics

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

# recall = recall\_score(y\_test, y\_pred, average='weighted')

# f1 = f1\_score(y\_test, y\_pred, average='weighted')

accuracy = accuracy\_score(y\_test, y\_pred)

print('\nConfusion Matrix:\n', conf\_matrix)

print('\nPrecision:', precision)

# print('\nRecall:', recall)

# print('\nF1-Score:', f1)

print("\nAccuracy: ", accuracy)

A diagram of a number and a number

Description automatically generated with medium confidenceoutput:

Precision: 0.802579735079735

Accuracy: 0.7833333333333333