Datasets:

1. Image dataset: GTSRB - German Traffic Sign Recognition Benchmark (From [Kaggle](https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign?datasetId=82373))

2. Non-image dataset: Mobile Price Classification (From previous course)

3. Self-made dataset: 飲料辨識

Algorithms:

1. KNN

2. Decision Tree

3. Random Forest

4. SVM (SVC and NuSVC)

5. Lenet

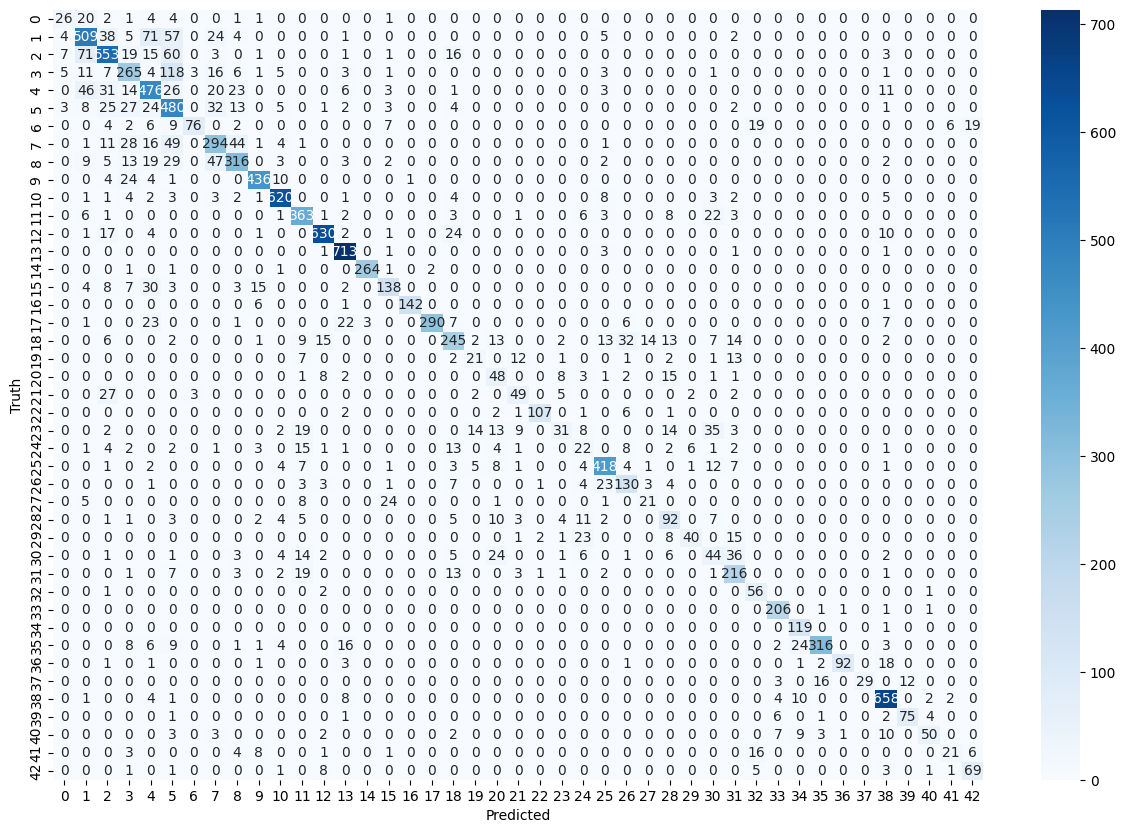
Analysis:

1. GTSRB:

GTSRB是German Traffic Sign Recognition Benchmark的縮寫，要辨識出德國的各種交通號誌，是IJCNN 2011所使用的題目，而這個資料庫是一個單照片多分類(single-image, multi-class)的分類問題，裡面有超過50,000張的照片和43種類別，而且分類的內容是一個現實世界可能會遇到的問題。

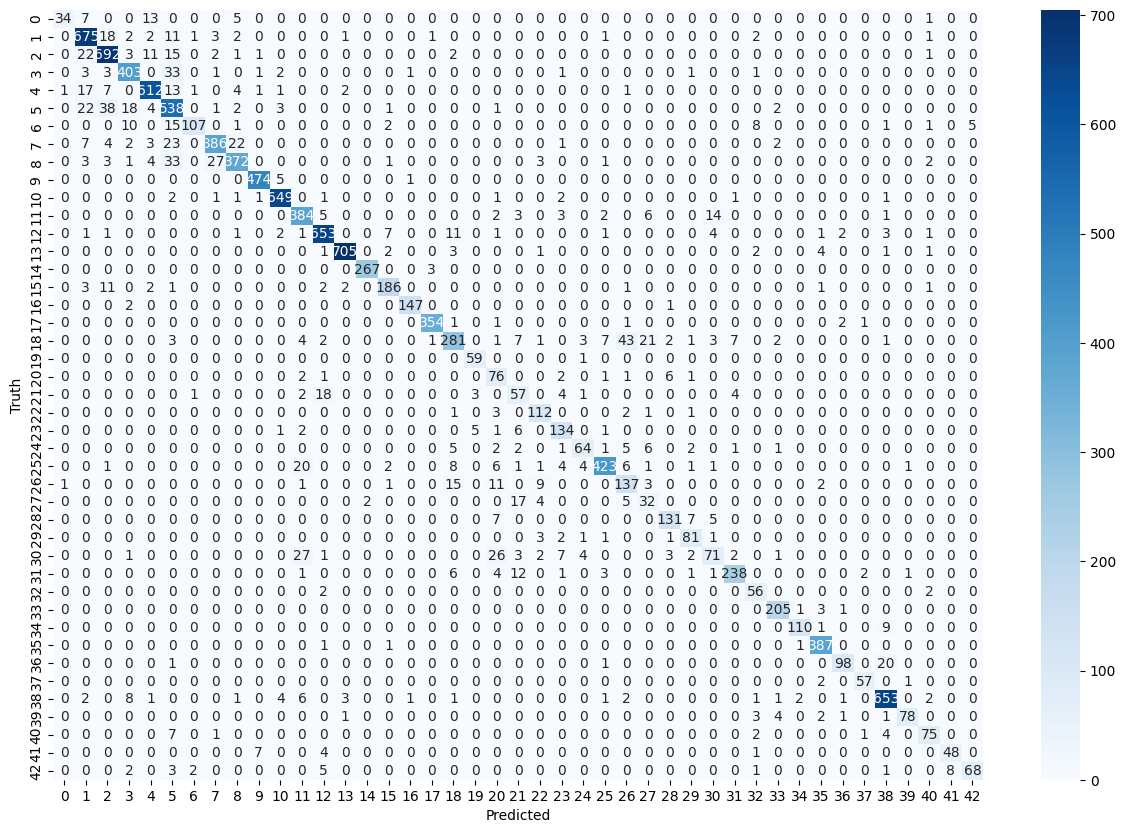
a. NuSVC:

NuSVC是SVM再加上一個限制，限制訓練誤差和分數的下界，這邊我的nu設定為0.1，kernel為rbf，gamma=0.001。訓練一次需要，訓練出來的準確率有0.773，但訓練一次就需要18分鐘，非常沒有效率。



b. Lenet:

Lenet是Yann LeCun 團隊於1998年提出的一個CNN架構，有兩個卷積層，兩個pooling層和池化層、全連接卷積層、全連接層所組成，訓練時間比NuSVC還要快上許多，訓練20個epochs也只需要約1分鐘，得到的準確率維0.862，比NuSVC來的好很多，也難怪後來在處理影像上，CNN會越來越成為主流。



c. Discussion:

根據這兩個分類器來說，可以預期CNN based的方法一定比SVM based的方法performance來的更好，但我沒有想到會差距這麼大，我猜是因為SVM要一次看3072維的資料(32\*32\*3)，要想辦法去找hyperplane來切割出這40多種的分類需要花很多的時間，如果有時間應該可以試試加上PCA去降維度，加入的同時就可以額外去做Grid Search，讓performance可以更好。

2. Mobile Price Classification:

Mobile Price Classification在Kaggle上也有類似的資料庫，但Kaggle的沒有提供Test set的正確解答，我的dataset是從先前的課程所提供的dataset，在Mobile Price Classification的地方會做Grid Search，找出最佳parameter時順便做cross validation (cv=5)，也有做training data數量的測試，分類器選Decision Tree和SVM，從sample rate 0.1, 0.2, …, 1，每個測試3次取平均。

a. Compare the results when using different classifiers.

|  |  |  |
| --- | --- | --- |
| Classifier | KNN | Decision Tree |
| Accuracy | 0.973 | 0.94 |
| Confusion  Matrix |  |  |
| Classifier | Random Forest | SVM |
| Accuracy | 0.907 | 0.987 |
| Confusion  Matrix |  |  |

b. Compare the results when using different amounts of training data.

c. Discussion

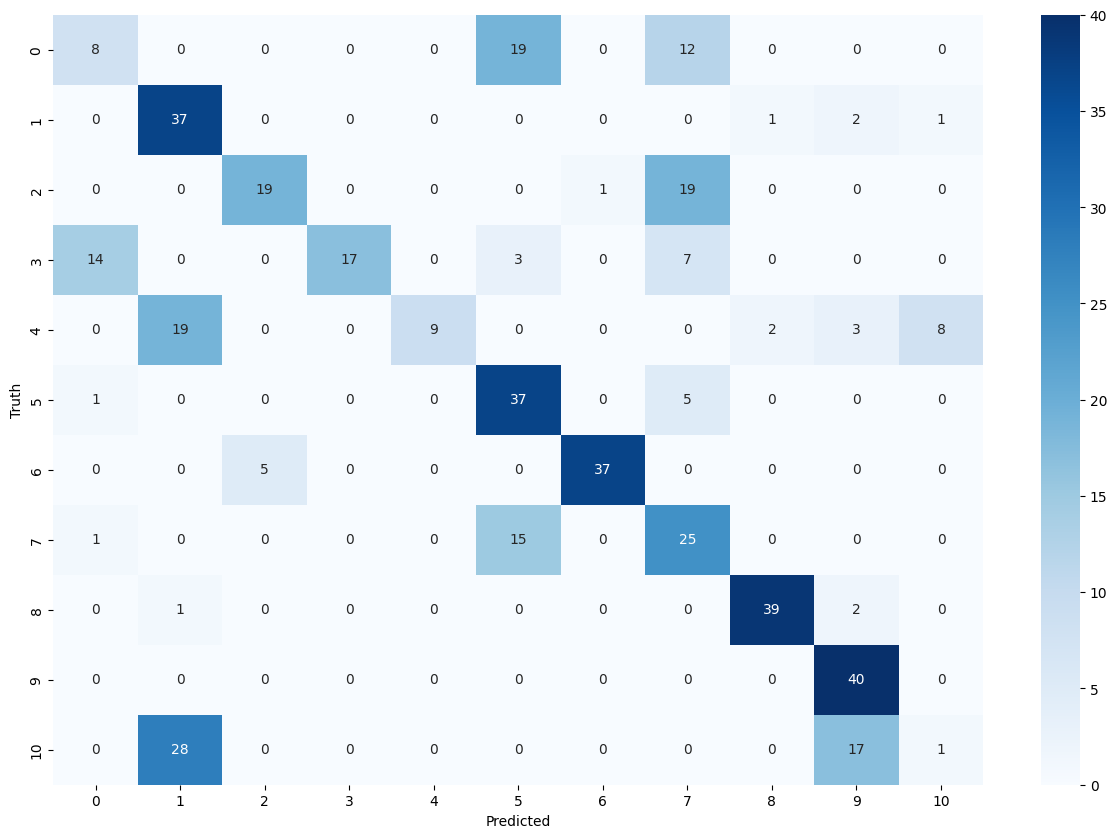
可以看到a部分的準確率排名是SVM > KNN > Decision Tree > Random Forest，而訓練時間基本上都是瞬間完成，所以不好比較，而這個dataset應該是比較單純，所以光是用KNN就可以拿到非常好的成績了，SVM也能很好的去分類，至於Decision Tree和Random Forest的performance不好，我猜是over fitting的關係，導致太貼近訓練資料，這應該也能解釋Random Forest比Decision Tree還要更不準的原因。再來看不同訓練數量的測試，可以看到SVM一直都在Random Forest的上方，而且浮動的幅度也比Random Forest還要小，可以說SVM對訓練資料量的要求小很多。因為時間的關係我只做了Decision Tree和Random Forest，有時間的話還可以補上SVM和KNN，或許會有一些有趣的發現和結果。

3. 飲料辨識:

這個Dataset是我和高鈺鴻(學號109550040)一起製作的，透過拍各種飲料瓶的四周來當作資料庫，我們總共拍了11種罐狀物品，每個物品大約有50張的照片，所以總共有大約550張照片做成我們的資料庫，裡面的類別有: Bar、日式綠茶、淡麗、麥香、綠茶、藍莓酒、寶礦力、蘋果酒、鐵觀音、水壺和刮鬍泡，有些是同品牌的飲料所以我們認為某些辨識上可能會比較複雜。而我使用了和第一個Image Dataset同樣的兩種classifier，NuSVC和Lenet。

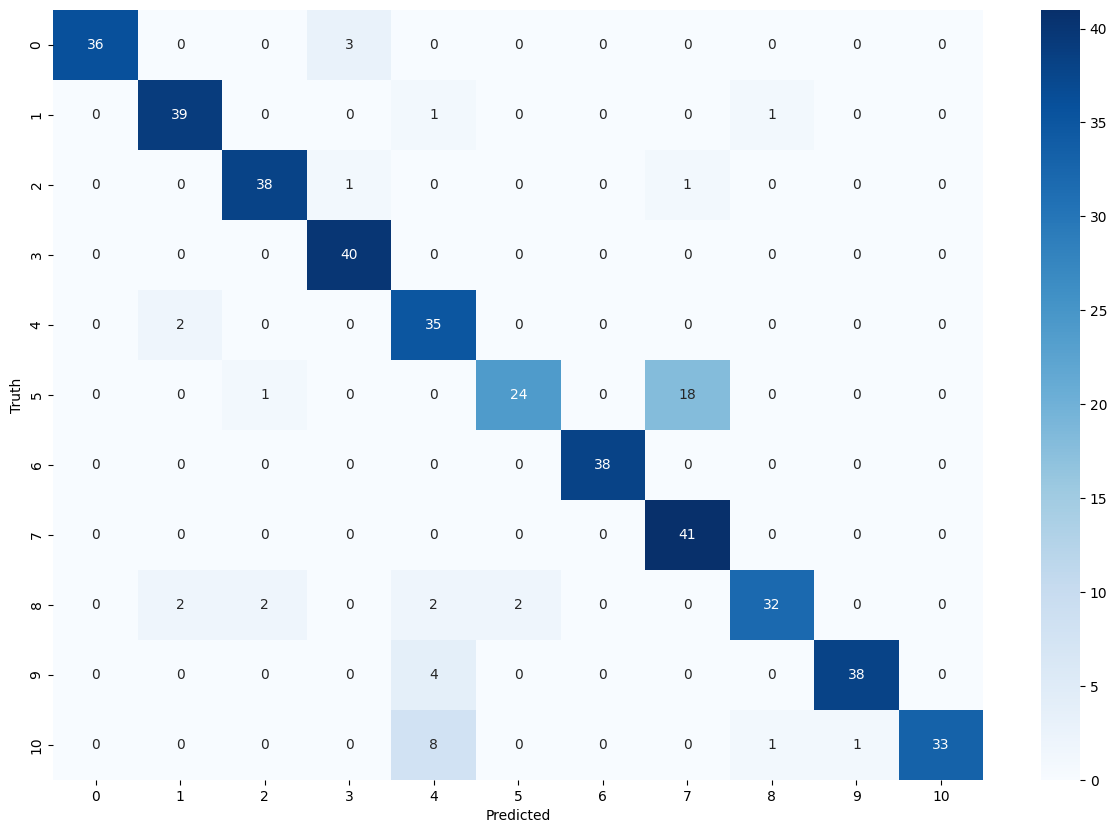
a. NuSVC:

下圖是NuSVC的Confusion Matrix，我把資料庫的20%當成validation set，3次得到的準確率是0.591、0.594、0.673，平均下來是0.619。



b. Lenet:

下圖是Lenet的Confusion Matrix，同樣把資料庫的20%當成validation set，3次得到的準確率是0.887、0.910、0.872，平均下來是0.890。



c. Discussion

可以從兩張圖和得到的準確率來知道CNN在一次的贏過了NuSVC許多，NuSVC只要顏色像或是輪廓像就會被誤判，但CNN只有在同品牌且只有些微文字和顏色不同的鋁罐才有出現比較大量的錯誤。如果還有更多時間的話，我應該會再繼續加大資料庫的種類和張數，甚至可以拍更多奇怪角度的照片，讓整個辨識的能力能夠更好。下面有資料庫每個類別的範例:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Code:

1. GTSRB – NuSVC:

import numpy as np

import pandas as pd

from PIL import Image

from matplotlib import pyplot as plt

import seaborn as sns

from skimage.io import imread

import cv2

from sklearn import preprocessing

from sklearn import svm

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

from sklearn.model\_selection import GridSearchCV

# load train and test data

train = pd.read\_csv('Train.csv')

train\_x = []

for i in train['Path']:

    img = Image.fromarray(cv2.imread(i), 'RGB')

    train\_x.append(np.array(img.resize((32,32))))

train\_x = np.array(train\_x)

train\_y = np.array(train['ClassId'].values)

print(train\_x.shape)

test = pd.read\_csv('Test.csv')

test\_x = []

for i in test['Path']:

    img = Image.fromarray(cv2.imread(i), 'RGB')

    test\_x.append(np.array(img.resize((32,32))))

test\_x = np.array(test\_x)

test\_y = np.array(test['ClassId'].values)

print(test\_x.shape)

# normalize data

train\_x.resize(train\_x.shape[0], 32 \* 32 \* 3)

test\_x.resize(test\_x.shape[0], 32 \* 32 \* 3)

train\_x = preprocessing.normalize(train\_x)

test\_x = preprocessing.normalize(test\_x)

# train model

svm\_clf = svm.NuSVC(nu=0.1, kernel='rbf', gamma=0.001, random\_state=8, verbose=10)

svm\_clf.fit(train\_x, train\_y)

svm\_clf.score(test\_x, test\_y)

# predict

svm\_pred = svm\_clf.predict(test\_x)

# plot confusion matrix

plt.figure(figsize=(15, 10))

sns.heatmap(confusion\_matrix(test\_y, svm\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

2. GTSRB – CNN:

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import cv2

import tensorflow as tf

from PIL import Image

import os

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

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

# prepare the data

data = []

labels = []

classes = 43

cur\_path = os.getcwd()

print(cur\_path)

# Load the images and lables from the dataset

for i in range(classes):

    path = os.path.join(cur\_path,'train',str(i))

    images = os.listdir(path)

    for a in images:

        try:

            image = Image.open(path + '\\'+ a)

            image = image.resize((32,32))

            image = np.array(image)

            data.append(image)

            labels.append(i)

        except:

            print("Error loading image")

# Convert the data and labels into numpy arrays

data = np.array(data)

labels = np.array(labels)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.25, random\_state=8)

y\_train = to\_categorical(y\_train, 43)

y\_test = to\_categorical(y\_test, 43)

# Building the model

model = Sequential()

model.add(Conv2D(filters=6, kernel\_size=(5, 5), input\_shape=(32, 32, 3), activation='tanh'))

model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(filters=16, kernel\_size=(5, 5), activation='tanh'))

model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))

model.add(Flatten())

model.add(Dense(120, activation='tanh'))

model.add(Dense(84, activation='tanh'))

model.add(Dense(43, activation='softmax'))

# Compile and train the model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

epochs = 20

history = model.fit(X\_train, y\_train, batch\_size=32, epochs=epochs, validation\_data=(X\_test, y\_test))

# Plot the loss and accuracy curves for training and validation

plt.figure(0)

plt.plot(history.history['accuracy'], label='training accuracy')

plt.plot(history.history['val\_accuracy'], label='val accuracy')

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

plt.figure(1)

plt.plot(history.history['loss'], label='training loss')

plt.plot(history.history['val\_loss'], label='val loss')

plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()

# Load the test dataset

test = pd.read\_csv('Test.csv')

labels = test["ClassId"].values

imgs = test["Path"].values

data=[]

for img in imgs:

    image = Image.open(img)

    image = image.resize((32,32))

    data.append(np.array(image))

X\_test=np.array(data)

pred = model.predict(X\_test)

pred = np.argmax(pred, axis=1)

print(accuracy\_score(labels, pred))

# Plot the confusion matrix

plt.figure(figsize=(15, 10))

sns.heatmap(confusion\_matrix(labels, pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

3. MPC – All classifier: (步驟都相同，指註解了KNN的部分)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn import metrics

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

# load the data

data = pd.read\_csv("train.csv")

# split the data into X and y

X = data.drop('price\_range', axis=1)

y = data['price\_range']

test = pd.read\_csv("test.csv")

test\_data = test.drop('price\_range', axis=1)

test\_ans = pd.DataFrame(test['price\_range'])

# KNN

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

# use GridSearchCV to find the best k of KNN

param = {'n\_neighbors': np.arange(1, 25)}

knn = KNeighborsClassifier()

knn\_cv = GridSearchCV(knn, param, cv=5)

knn\_cv.fit(X, y)

# use the best k to train the model

knn\_clf = KNeighborsClassifier(n\_neighbors=knn\_cv.best\_estimator\_.n\_neighbors)

knn\_clf.fit(X, y)

# predict the test data

knn\_pred = knn\_clf.predict(test\_data)

print("Accuracy:",metrics.accuracy\_score(test\_ans, knn\_pred))

# plot the confusion matrix

confusion\_matrix(test\_ans, knn\_pred)

plt.figure(figsize=(5, 3.5))

sns.heatmap(confusion\_matrix(test\_ans, knn\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

# Decision Tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

param = {'criterion':['gini', 'entropy'], 'max\_depth':range(1, 5, 1), 'min\_samples\_leaf':range(1, 26, 5)}

clf = GridSearchCV(DecisionTreeClassifier(), param, cv=5, scoring='accuracy', n\_jobs=-1, verbose=10)

clf.fit(X, y)

dt\_clf = DecisionTreeClassifier(criterion=clf.best\_params\_['criterion'], max\_depth=clf.best\_params\_['max\_depth'], min\_samples\_leaf=clf.best\_params\_['min\_samples\_leaf'])

dt\_clf.fit(X, y)

dt\_pred = dt\_clf.predict(test\_data)

print("Accuracy:", accuracy\_score(test\_ans, dt\_pred))

confusion\_matrix(test\_ans, dt\_pred)

plt.figure(figsize=(5, 3.5))

sns.heatmap(confusion\_matrix(test\_ans, dt\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

# Random Forest:

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import cross\_val\_score

param = {'n\_estimators':range(1, 100, 10), 'criterion':['gini', 'entropy'], 'max\_depth':range(1, 5, 1), 'min\_samples\_leaf':range(1, 26, 5)}

clf = GridSearchCV(RandomForestClassifier(), param, cv=5, scoring='accuracy', n\_jobs=-1, verbose=10)

clf.fit(X, y)

rf\_clf = RandomForestClassifier(n\_estimators=clf.best\_params\_['n\_estimators'], criterion=clf.best\_params\_['criterion'], max\_depth=clf.best\_params\_['max\_depth'], min\_samples\_leaf=clf.best\_params\_['min\_samples\_leaf'])

rf\_clf.fit(X, y)

rf\_pred = rf\_clf.predict(test\_data)

print("Accuracy:", accuracy\_score(test\_ans, rf\_pred))

confusion\_matrix(test\_ans, rf\_pred)

plt.figure(figsize=(5, 3.5))

sns.heatmap(confusion\_matrix(test\_ans, rf\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

# SVM

from sklearn.svm import SVC

from sklearn.model\_selection import GridSearchCV

param = {'C':[0.001, 0.01, 0.1, 1, 10], 'gamma':[100, 10, 1, 0.1, 0.01], 'kernel':['rbf', 'linear']}

clf = GridSearchCV(SVC(), param, cv=5, scoring='accuracy', n\_jobs=-1, verbose=10)

clf.fit(X, y)

SVM\_clf = SVC(C=clf.best\_params\_['C'], gamma=clf.best\_params\_['gamma'], kernel=clf.best\_params\_['kernel'])

SVM\_clf.fit(X, y)

svm\_pred = SVM\_clf.predict(test\_data)

print("Accuracy:", accuracy\_score(test\_ans, svm\_pred))

confusion\_matrix(test\_ans, svm\_pred)

plt.figure(figsize=(5, 3.5))

sns.heatmap(confusion\_matrix(test\_ans, svm\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

4. Self-made dataset – NuSVC:

import numpy as np

import pandas as pd

import os

from PIL import Image

from matplotlib import pyplot as plt

import seaborn as sns

from skimage.io import imread

import cv2

from sklearn import preprocessing

from sklearn import svm

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

from sklearn.model\_selection import GridSearchCV

# Loading the data

train\_x = []

train\_y = []

test\_x = []

test\_y = []

num\_of\_classes = 11

for i in range(0, num\_of\_classes):

    path = './train/' + chr(65 + i) + '/'

    files = os.listdir(path)

    print(len(files))

    for file in files:

        img = Image.fromarray(cv2.imread(path + file), 'RGB')

# Split data into train set and val set

        tmp = np.random.randint(0, 100)

        if tmp < 20:

            train\_x.append(np.array(img.resize((32,32))))

            train\_y.append(i)

        else:

            test\_x.append(np.array(img.resize((32,32))))

            test\_y.append(i)

train\_x = np.array(train\_x)

train\_y = np.array(train\_y)

test\_x = np.array(test\_x)

test\_y = np.array(test\_y)

# Normalizing the data

train\_x = train\_x.reshape(train\_x.shape[0], 32\*32\*3)

test\_x = test\_x.reshape(test\_x.shape[0], 32\*32\*3)

train\_x = preprocessing.normalize(train\_x)

test\_x = preprocessing.normalize(test\_x)

# Training the model

svm\_clf = svm.NuSVC(nu=0.1, kernel='rbf', gamma=0.001, random\_state=8, verbose=10)

svm\_clf.fit(train\_x, train\_y)

svm\_clf.score(test\_x, test\_y)

# Predicting the model

svm\_pred = svm\_clf.predict(test\_x)

plt.figure(figsize=(15, 10))

sns.heatmap(confusion\_matrix(test\_y, svm\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')

5. Self-made dataset – CNN:

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

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import cv2

import tensorflow as tf

from PIL import Image

import os

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

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

# Load the data

data = []

labels = []

test = []

test\_labels = []

num\_of\_classes = 11

for i in range(0, num\_of\_classes):

    path = './train/' + chr(65 + i) + '/'

    files = os.listdir(path)

    print(len(files))

    for file in files:

        image = Image.open(path + file)

        image = image.resize((32,32))

        image = np.array(image)

        # split the data into train and test

        tmp = np.random.randint(0, 100)

        if tmp < 20:

            data.append(image)

            labels.append(i)

        else:

            test.append(image)

            test\_labels.append(i)

# Convert the data into numpy array

data = np.array(data)

labels = np.array(labels)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.25, random\_state=8)

y\_train = to\_categorical(y\_train, num\_of\_classes)

y\_test = to\_categorical(y\_test, num\_of\_classes)

# Building the model

model = Sequential()

model.add(Conv2D(filters=6, kernel\_size=(5, 5), input\_shape=(32, 32, 3), activation='tanh'))

model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(filters=16, kernel\_size=(5, 5), activation='tanh'))

model.add(MaxPool2D(pool\_size=(2, 2), strides=(2, 2)))

model.add(Flatten())

model.add(Dense(120, activation='tanh'))

model.add(Dense(84, activation='tanh'))

model.add(Dense(num\_of\_classes, activation='softmax'))

# Compile the model and train it

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

epochs = 20

history = model.fit(X\_train, y\_train, batch\_size=32, epochs=epochs, validation\_data=(X\_test, y\_test))

# Plot the accuracy and loss

plt.figure(0)

plt.plot(history.history['accuracy'], label='training accuracy')

plt.plot(history.history['val\_accuracy'], label='val accuracy')

plt.title('Accuracy')

plt.xlabel('epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

plt.figure(1)

plt.plot(history.history['loss'], label='training loss')

plt.plot(history.history['val\_loss'], label='val loss')

plt.title('Loss')

plt.xlabel('epochs')

plt.ylabel('loss')

plt.legend()

plt.show()

# Testing the model

val = np.array(test)

val\_labels = np.array(test\_labels)

cnn\_pred = model.predict(val)

cnn\_pred = np.argmax(cnn\_pred, axis=1)

print("CNN accuracy: ", accuracy\_score(val\_labels, cnn\_pred))

# plot the confusion matrix

plt.figure(figsize=(15, 10))

sns.heatmap(confusion\_matrix(val\_labels, cnn\_pred), annot=True, cmap='Blues', fmt='g')

plt.xlabel('Predicted')

plt.ylabel('Truth')