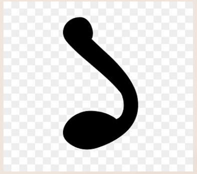
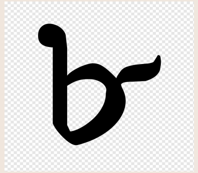
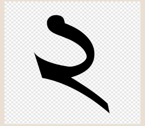
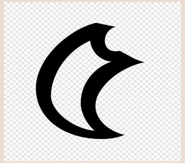
Brac University

Bangla Digit Recognition across Different Variations of CNN Models

By Atif Ronan (20201075), Nafiun Al Amin (20201069), Asif Ali (20201049), Ayan Haider (20201152), Nusaiba Zaman (20201104)

Introduction

This project focuses on using varying approaches to CNN Models to recognise Handwritten Bengali digits, which poses significant challenges for traditional recognition algorithms. Moreover, we explore the integration of ensemble techniques with CNN architectures to harness the collective intelligence of multiple models, thereby enhancing the classification accuracy, recall and precision to variations.

Literature Review

- Saha et al. (2019) had conducted research on Bangla language digit recognition using a seven layered D-CNN model. The dataset used was CMATERdb 3.1.1 contai 6000 images. The model had an accuracy of 97.60% for testing data and 99.90% for training data.
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8679309
- Rahman et al. (2018) has conducted a research based on Bangla Language digit recognition using D-CNN model. The dataset used is NumtaDB dataset, which is a large and unbiased dataset. The model had an accuracy of 92.72%.
  - https://ieeexplore.ieee.org/document/8554900
- Hasan et al. (2023) had conducted research to recognize and extract bangla words from digital images using different CNN models. The researchers used a custom dataset, collecting images of bengali words from books, posters, leaflets etc. The model with the highest accuracy was the VGG16 model, with 98.5% accuracy for training data and 97% for testing data.
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10117745
- Abujar et al.(2021) had conducted a research on developing a Handwritten text classification model with the capabilities of identify the author of a piece of handwr text in image form. The dataset they used consisted of 105 images from 3 persons. The model was based on CNN and performed strongly for this task, and could identify an author with 96.3% accuracy.
  - https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9579870&tag=1
- Akhand et al. (2016) conducted a research using three different CNNs with the same architecture on different training sets and combined their decisions for the Ba handwritten numeral recognition.
  - https://core.ac.uk/works/83923063

- Navya et al. (2023) conducted research on improving the existing OCR technology through proposing different combinations of models such as CNNs, HOG or SIFT for feature extraction and RNNs, HMMs and SVMs for handwriting recognition. The dataset used is the IAM dataset. The highest accuracy of 96% was produced by a combined model of CNN for feature extraction, RNN for handwriting recognition and CTC for word decoding.
  - https://ieeexplore.ieee.org/document/10420040

- Akhter and Ahmed (2023) uses an ensemble approach with pre-trained CNN models to enhance the accuracy of recognizing handwritten characters in the Bangla language. The proposed model consisted of an ensemble technique combining predictions from 4 pre-trained CNN architectures- ResNet50, DenseNet121, Xception, and EfficientNetB0. The model achieved accuracy of 97.83%, 97.78% and 97.01% on different datasets.
  - https://www.researchgate.net/publication/375276177_An_Ensemble_approach_of_Pre-trained_CNN_models_for_Recognition_of_Handwritten_Characte n_Bangla

References

Abujar, S., Badhon, S. S. I., & Duraisamy, P. (2021). Handwritten Text Recognition for Non-Latin Languages using Deep Learning - Bangla. *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. https://doi.org/10.1109/icccnt51525.2021.9579870

Akhand, M. a. H., Ahmed, M., & Rahman, M. H. (2016). Multiple Convolutional Neural Network Training for Bangla Handwritten Numeral Recognition. *6th International Conference on Computer and Communication Engineering (ICCCE)*. https://doi.org/10.1109/iccce.2016.73

Akhter, K. F. B., & Ahmed, T. (2023). An Ensemble approach of Pre-trained CNN models for Recognition of Handwritten Characters in Bangla. *ResearchGate*. https://www.researchgate.net/publication/375276177_An_Ensemble_approach_of_Pre-trained_CNN_models_for_Recognition_of_Handwritten_Characters_in_Bangla

Hasan, S. M. K., Nadif, M. A., Rahman, N., & Rana, M. (2023). A Bengali Word Identification and Verification Using Machine Learning Approach. *Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*. https://doi.org/10.1109/icaect57570.2023.10117745

Navya, L., Ali, M., Sai, K., Shyam, K., & Adepu, R. (2023). Handwritten Text Recognition Using Deep Learning Techniques. *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems (AICERA/ICIS)*. https://doi.org/10.1109/aicera/icis59538.2023.10420040

Saha, C., Faisal, R. H., & Rahman, M. M. (2019). Bangla Handwritten Digit Recognition Using an Improved Deep Convolutional Neural Network Architecture. *International Conference on Electrical, Computer and Communication Engineering (ECCE)*. https://doi.org/10.1109/ecace.2019.8679309

Shawon, A., Rahman, M. J., Mahmud, F., & Zaman, M. (2018). Bangla Handwritten Digit Recognition Using Deep CNN for Large and Unbiased Dataset. *International Conference on Bangla Speech and Language Processing (ICBSLP)*. https://doi.org/10.1109/icbslp.2018.8554900

Dataset Description

NumtaDB: Bangla Handwritten Digits
Link : [NumtaDB](https://www.kaggle.com/datasets/BengaliAI/numta/data)

- Contains images from 6 different datasets
- Over 85000 images
- 180x180 pixels in dimensions
- Dataset labelled from a to f

NUMTADB: Bengali handwritten digits. (2018, August 14). Kaggle.

https://www.kaggle.com/datasets/BengaliAI/numta/data

Exploratory Data Analysis

Initial:

Before performing any data preprocessing and wrangling, the following factors were analysed

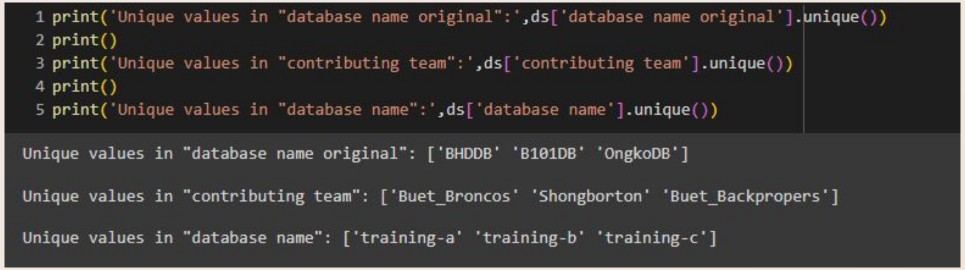
- ❖ Data diversity of select columns which were initially identified to be removed
    - “database name original”, “contributing team”, “ database name”

- ❖ Distribution of values for the target variable “digit”

- ❖ Correlation between digit and scanid columns

Code and Results

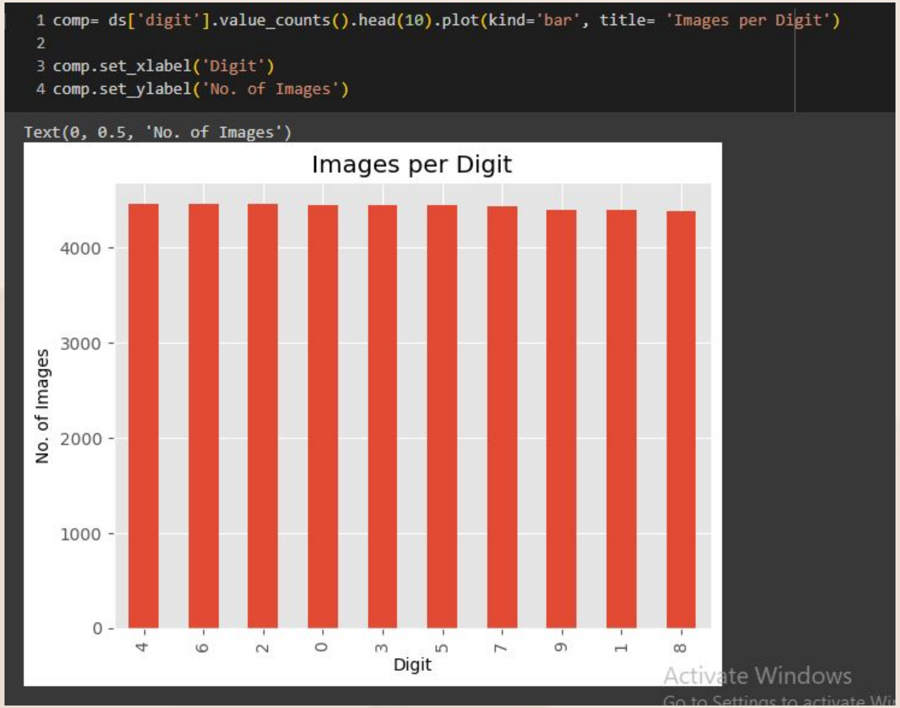
❖ Data diversity check



```
print('Unique values in "database name original":',ds['database name original'].unique())
print()
print('Unique values in "contributing team":',ds['contributing team'].unique())
print()
print('Unique values in "database name":',ds['database name'].unique())
```

```
Unique values in "database name original": ['BHDDB' 'B101DB' 'OngkoDB']

Unique values in "contributing team": ['Buet_Broncos' 'Shongborton' 'Buet_Backpropers']

Unique values in "database name": ['training-a' 'training-b' 'training-c']
```

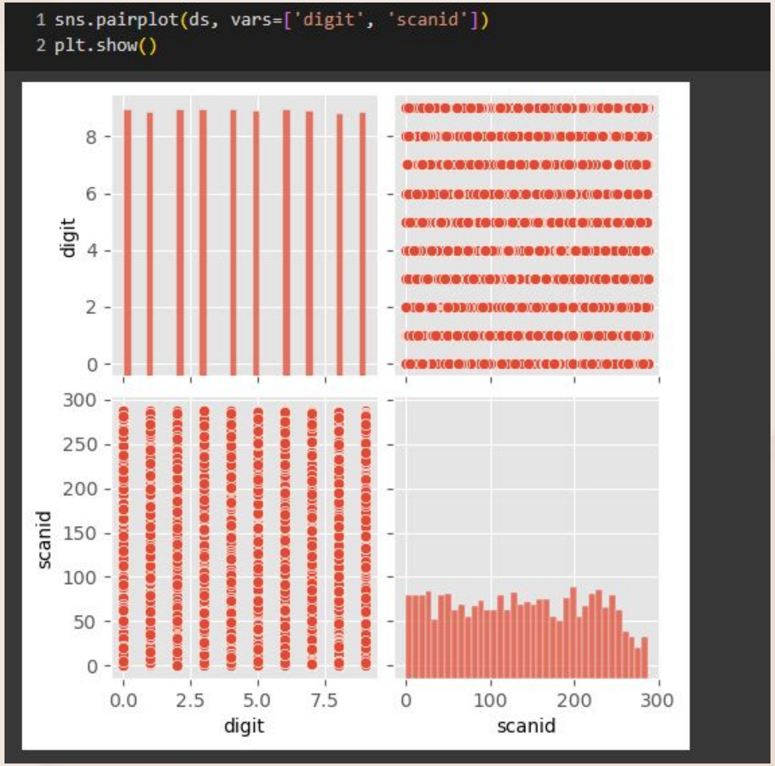
❖ Distribution of values check



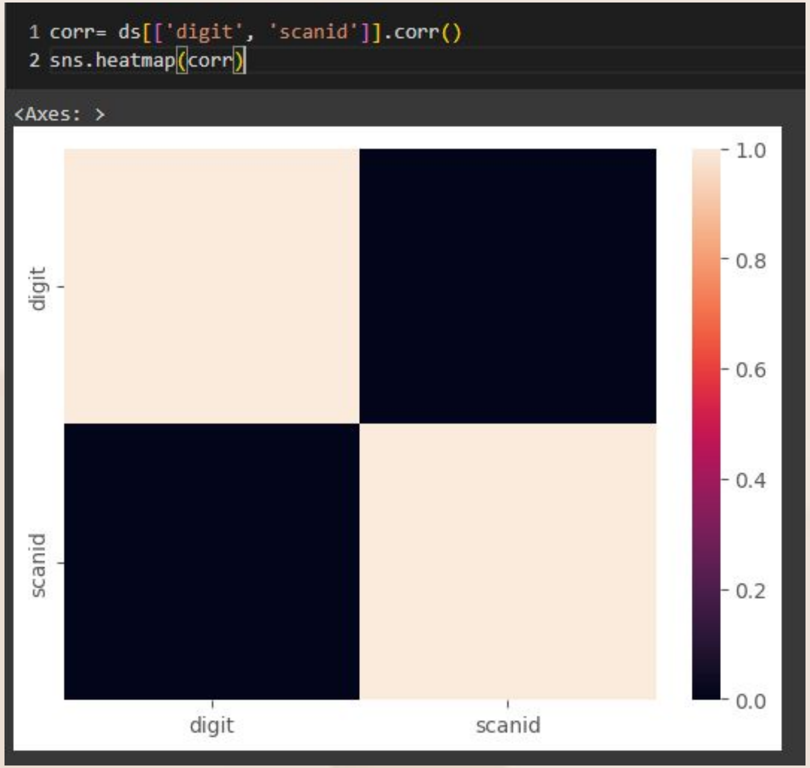
```
comp= ds['digit'].value_counts().head(10).plot(kind='bar', title= 'Images per Digit')

comp.set_xlabel('Digit')
comp.set_ylabel('No. of Images')
```

```
Text(0, 0.5, 'No. of Images')
```

❖ Correlation check



```
sns.pairplot(ds, vars=['digit', 'scanid'])
plt.show()
```



```
corr= ds[['digit', 'scanid']].corr()
sns.heatmap(corr)
```

```
<Axes: >
```

Findings

- Data diversity of select columns which were initially identified to be removed
  - Columns “database name original” and “contributing team” were considered redundant as our end goal is to be able to check images from any source, not just the handful present in the database

- Distribution of values for the target variable “digit”
  - There was no imbalance in our target variable, meaning there would be no bias towards any digit

- Correlation between digit and scanid columns
  - Column “scanid” was considered redundant as it is only an identity column and has no correlation with our target variable

Exploratory Data Analysis

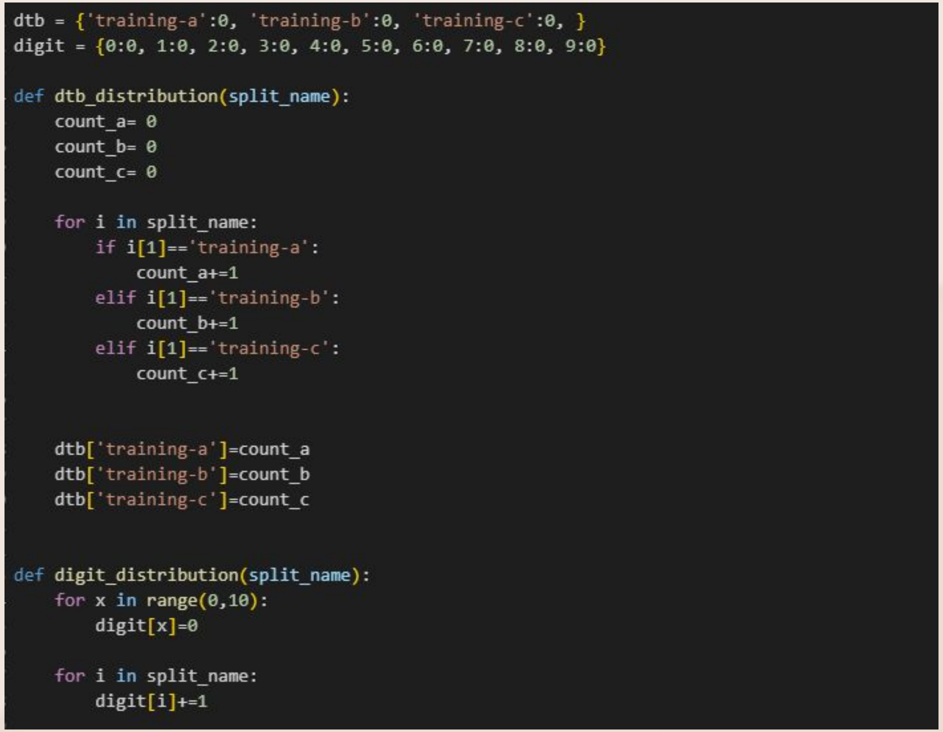
After Data Wrangling:

Based on the initial EDA, and prior knowledge of the dataset, data pre-processing and wrangling were performed to remove redundant columns, and the dataset was split into train and test sets, after which further analyses were performed on the following factors

- ❖ Distribution of values from the 3 databases in X_train and X_test
    - “training-a”, “training-b”, “training-c”
- ❖ Distribution of values for the target variable “digit” in y_train and y_test

Code and Results

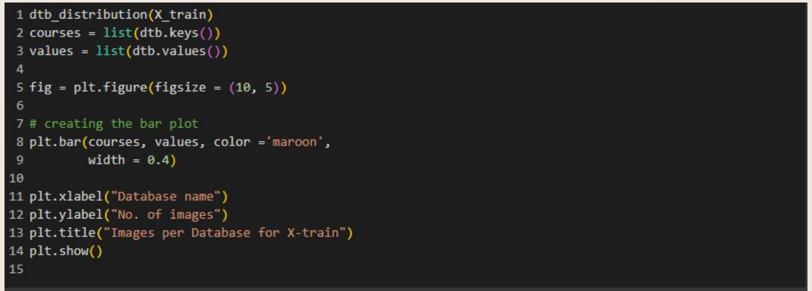
❖ Distribution calculator for X_ and y_ sets



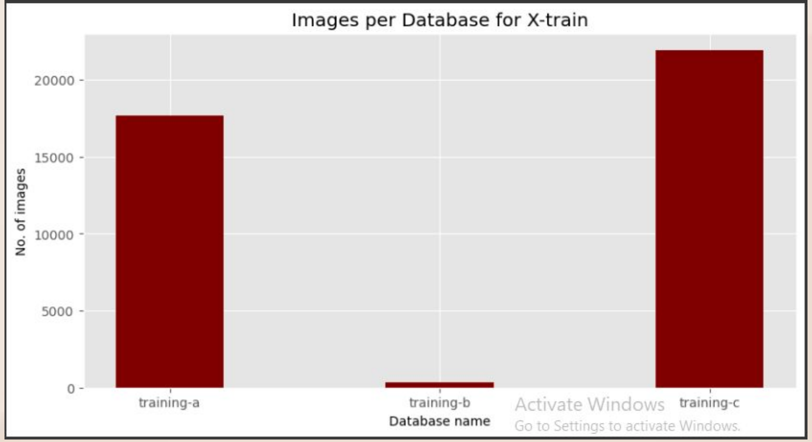
```
dtb = {'training-a':0, 'training-b':0, 'training-c':0, }
digit = {0:0, 1:0, 2:0, 3:0, 4:0, 5:0, 6:0, 7:0, 8:0, 9:0}

def dtb_distribution(split_name):
    count_a= 0
    count_b= 0
    count_c= 0

    for i in split_name:
        if i[1]=='training-a':
            count_a+=1
        elif i[1]=='training-b':
            count_b+=1
        elif i[1]=='training-c':
            count_c+=1


    dtb['training-a']=count_a
    dtb['training-b']=count_b
    dtb['training-c']=count_c


def digit_distribution(split_name):
    for x in range(0,10):
        digit[x]=0

    for i in split_name:
        digit[i]+=1
```

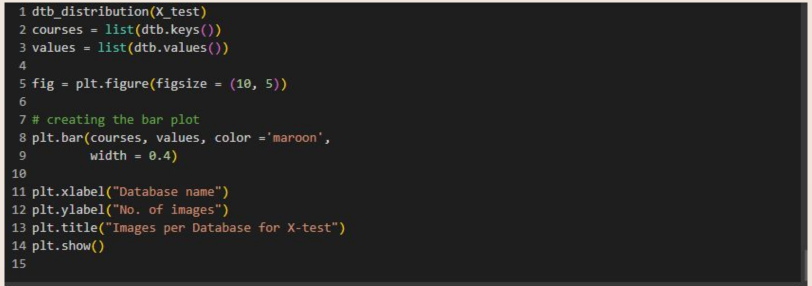
Code and Results

- Distributions for X_train and X_test

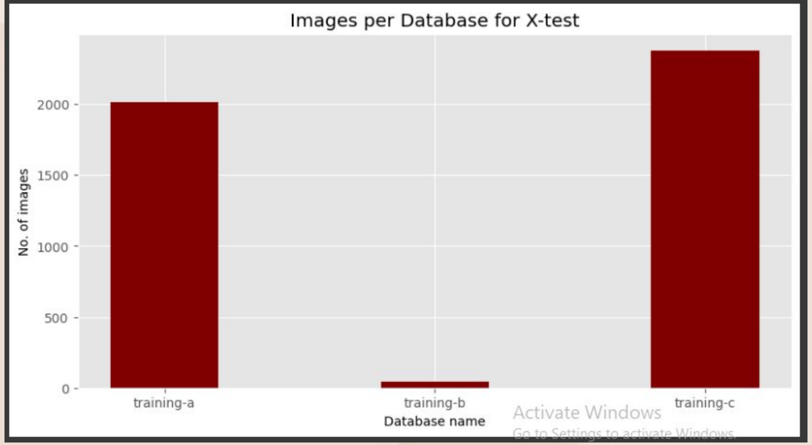


```
dtb_distribution(X_train)
courses = list(dtb.keys())
values = list(dtb.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Database name")
plt.ylabel("No. of images")
plt.title("Images per Database for X-train")
plt.show()
```



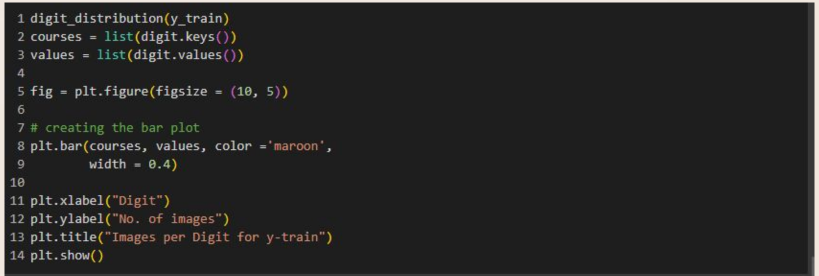


```
dtb_distribution(X_test)
courses = list(dtb.keys())
values = list(dtb.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Database name")
plt.ylabel("No. of images")
plt.title("Images per Database for X-test")
plt.show()
```

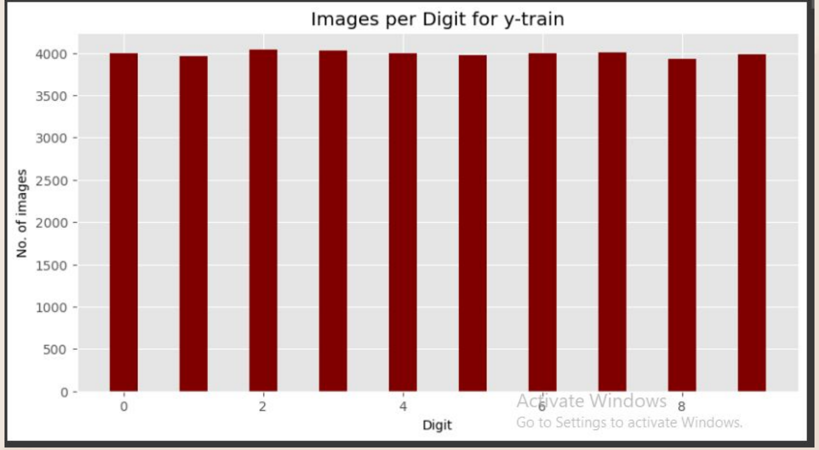


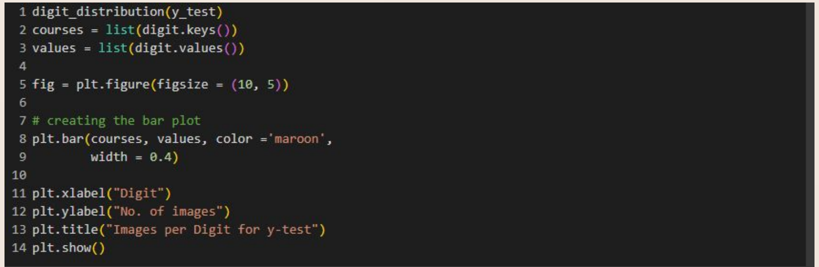
Code and Results

❖ Distributions for y_train and y_test

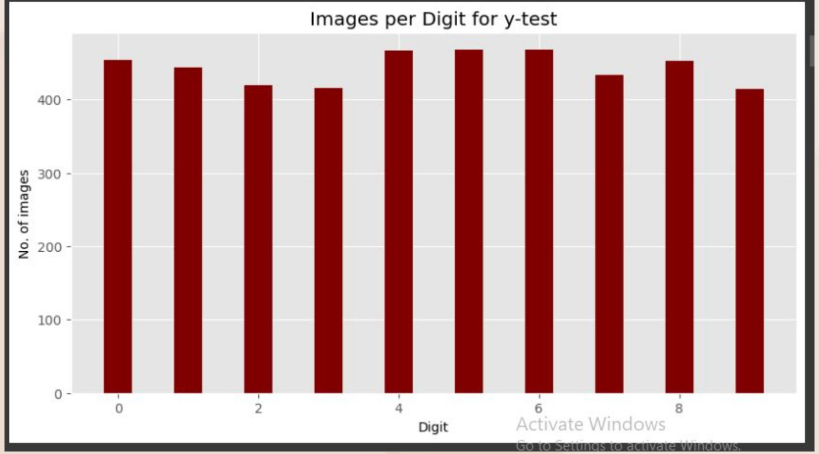


```python
digit_distribution(y_train)
courses = list(digit.keys())
values = list(digit.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Digit")
plt.ylabel("No. of images")
plt.title("Images per Digit for y-train")
plt.show()
```





```python
digit_distribution(y_test)
courses = list(digit.keys())
values = list(digit.values())

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(courses, values, color ='maroon',
        width = 0.4)

plt.xlabel("Digit")
plt.ylabel("No. of images")
plt.title("Images per Digit for y-test")
plt.show()
```



Findings:

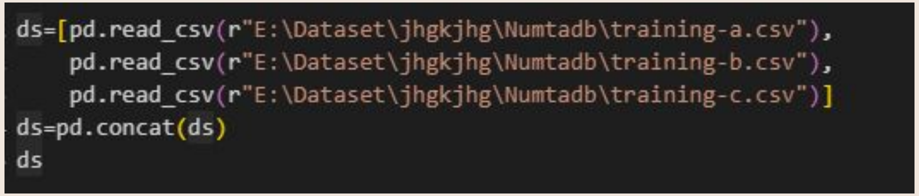
- Distribution of values from the 3 databases in X_train and X_test
  - Despite the seemingly large imbalance, balancing was not required as having prior knowledge of and examining the images of training-a, training-b, and training-c shows that there aren't any images with significant uniqueness to that database
  - Additionally, due to the size imbalance of 3 databases, balancing would require the loss of an extreme amount of data for there to be equal distribution

- Distribution of values for the target variable "digit" in y_train and y_test
  - There was no imbalance in our target variable in y_train
  - The minor imbalance in y_test was negligible compared to volume of images present for each digit

Pre-Processing and Wrangling

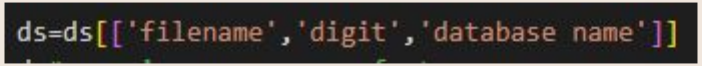
<u>Summary of steps:</u>

- ❖ Merging training-a, training-b, training-c



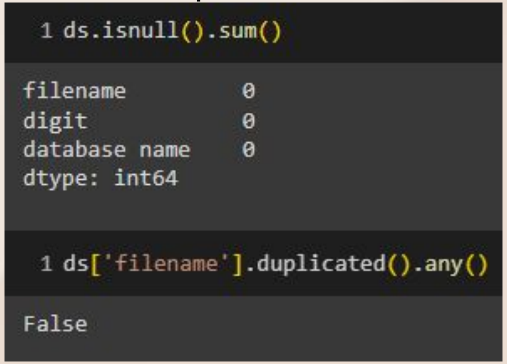
```
ds=[pd.read_csv(r"E:\Dataset\jhgkjhg\Numtadb\training-a.csv"),
    pd.read_csv(r"E:\Dataset\jhgkjhg\Numtadb\training-b.csv"),
    pd.read_csv(r"E:\Dataset\jhgkjhg\Numtadb\training-c.csv")]
ds=pd.concat(ds)
ds
```

- ❖ Redundant column removal
  - ▪ “database name original”, “contributing team”, “scanid”



```
ds=ds[['filename','digit','database name']]
```

- ❖ Null and Duplicate value checking



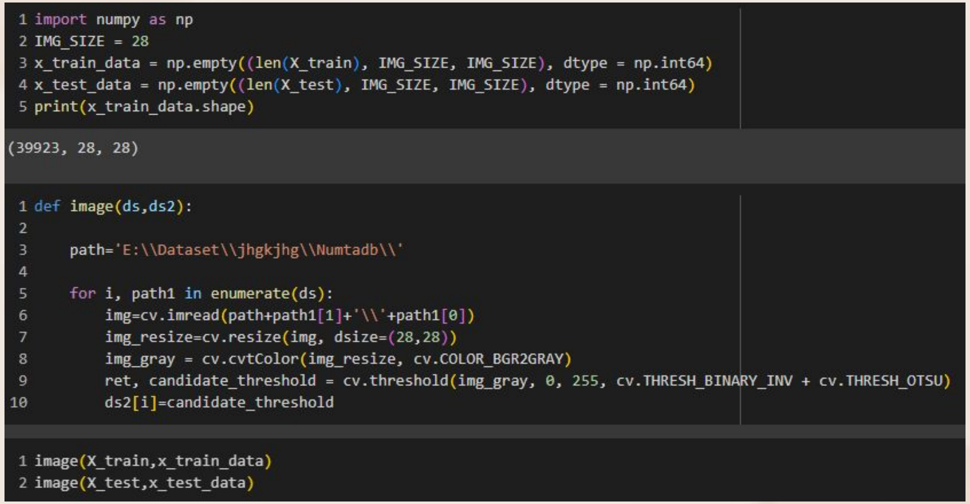
```
ds.isnull().sum()
```

```
filename         0
digit            0
database name    0
dtype: int64
```

```
ds['filename'].duplicated().any()
```

```
False
```

❖ Image Resizing



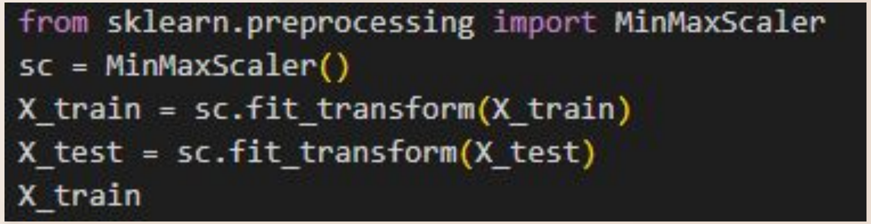
```
import numpy as np
IMG_SIZE = 28
x_train_data = np.empty((len(X_train), IMG_SIZE, IMG_SIZE), dtype = np.int64)
x_test_data = np.empty((len(X_test), IMG_SIZE, IMG_SIZE), dtype = np.int64)
print(x_train_data.shape)
```

```
(39923, 28, 28)
```

```
def image(ds,ds2):

    path='E:\\Dataset\\jhgkjhg\\Numtadb\\'

    for i, path1 in enumerate(ds):
        img=cv.imread(path+path1[1]+'\\'+path1[0])
        img_resize=cv.resize(img, dsize=(28,28))
        img_gray = cv.cvtColor(img_resize, cv.COLOR_BGR2GRAY)
        ret, candidate_threshold = cv.threshold(img_gray, 0, 255, cv.THRESH_BINARY_INV + cv.THRESH_OTSU)
        ds2[i]=candidate_threshold
```

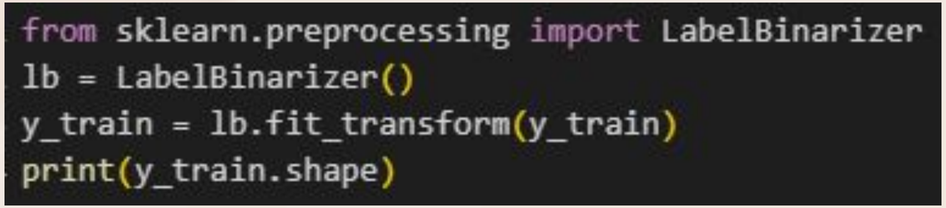
```
image(X_train,x_train_data)
image(X_test,x_test_data)
```

❖ Image Scaling



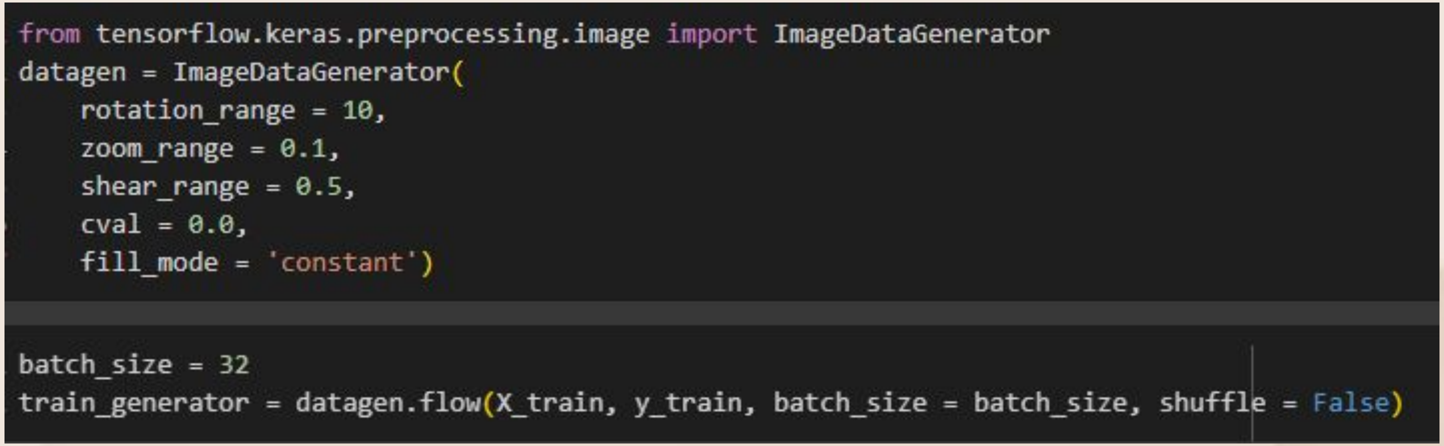
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
X_train
```

- Binary Labeling



```python
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
y_train = lb.fit_transform(y_train)
print(y_train.shape)
```

- Image Augmentation



```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
    rotation_range = 10,
    zoom_range = 0.1,
    shear_range = 0.5,
    cval = 0.0,
    fill_mode = 'constant')
```

```python
batch_size = 32
train_generator = datagen.flow(X_train, y_train, batch_size = batch_size, shuffle = False)
```

Model Description

The Proposed Models are:

- CNN model
    - 3 Convolution Layers and 1 Dense Layer
- CNN with Logistic Regression
    - 3 Convolution Layers and Logistic Regression as Classifier
- CNN with Random Forest Classifier
    - 3 Convolution Layers and Random Forest Classifier as Classifier
- Ensemble CNN model
    - CNN model with 3 Convolution Layers and CNN model with 4 Convolution Layers

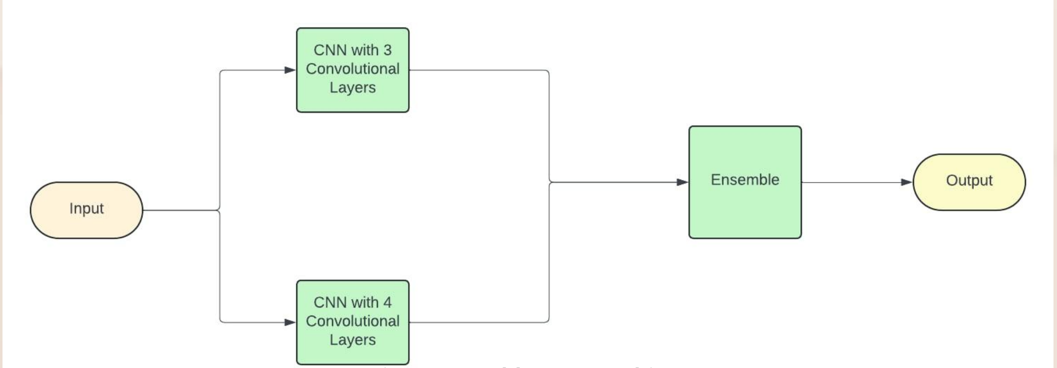
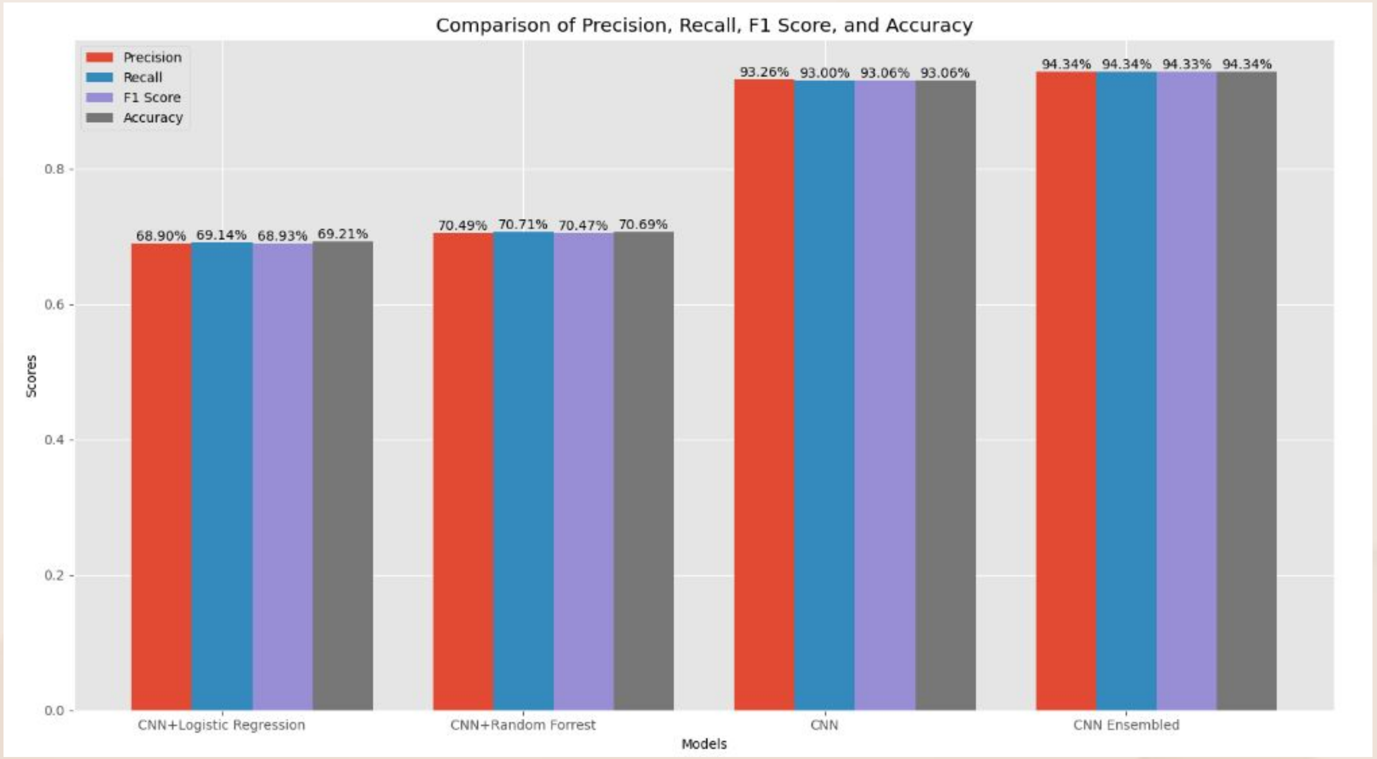

Input
CNN with 3 Convolutional Layers
CNN with 4 Convolutional Layers
Ensemble
Output

Fig 1: Ensemble CNN Architecture

Results and Analysis



Limitations

The limitations encountered in this study are:

- ❏ Low Computational Complexity
- ❏ Dataset with Lower Augmentation
- ❏ Not enough research papers on Ensemble models

Conclusion & Future Research

Overall this study has attempted to implement multiple different Convolutional Neural Network models to find the model that gives the best results for accurately recognizing Bangla handwritten digits, utilising images from the NumtaDB database. From our findings, the best accuracy achieved in this study was from an **Ensemble model of two variations of Simple CNN (94.34%)**, followed by the **Simple CNN model (93.06%)**, **CNN with Random Forest Classification model (70.69%**), and **CNN with Logistic Regression Classification model (69.21%)**.

The classification capability of the model can be improved further by

- ❖ Including more augmented photos from other datasets
- ❖ Using different ensemble techniques to merge the different CNN models
- ❖ Utilising different CNN models as well as models other than CNN

Thank You