HANDWRITING RECOGNITION AND WRITER IDENTIFICATION ON TEXT INDEPENDENT DATA

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BCSCCS708: **MINI PROJECT**

Submitted by

NAG ASHISH S V
(Reg. No.: 221003064, B. Tech, CSE B)
RANGA SATYA VISWA PAVAN
(Reg. No.: 221003080, B. Tech, CSE B)
PRAMODH SAIRAM P V
(Reg. No.: 221003117, B. Tech, CSE B)

December 2020

SRINIVASA RAMANUJAN CENTRE
KUMBAKONAM, TAMIL NADU, INDIA – 612 001

SRINIVASA RAMANUJAN CENTRE

KUMBAKONAM – 612 001

Bonafide Certificate

This is to certify that the report titled "**Handwriting Recognition and Writer Identification on Text Independent Data**" submitted as a requirement for the course, BCSCCS708**: MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. Nag Ashish S V (**Reg. No.: **221003064,** B. Tech CSE B**), Mr. Ranga Satya Viswa Pavan(**Reg. No.: **221003080,** B. Tech CSE B**), Mr. Pramodh Sairam P V(**Reg. No.: **221003117,** B. Tech CSE B**)** during the academic year 2020-20, in the Srinivasa Ramanujan Centre, under my supervision.

Signature of Project Supervisor **:**

Name with Affiliation **:** Dr. Durga Karthik/AP III/CSE/SRC/SASTRA

Date **:** 28.12.2020

Mini Project *Viva voc*e held on _____31.12.2020_____

Examiner 1 **Examiner 2**

Acknowledgements

We pay our sincere obeisance to God ALMIGHTY for his grace and infinite mercy and for showing on us his choicest blessings.

We would like to express our sincere thanks to our honorable chancellor **Prof. R. Sethuraman**, Vice Chancellor **Dr. S. Vaidhyasubramaniam** and Registrar **Dr. R. Chandramouli** for giving us an opportunity to be a student of this esteemed institution.

We express our deepest thanks to our revered, **Dr. V. Ramaswamy**, Dean of Engineering, and **Dr. A. AlliRani**, Associate Dean, Srinivasa Ramanujan Centre for all their moral support and suggestions when required without any reservations.

We exhibit our pleasure in expressing our thanks to **Dr. S. Sivagurunathan**, Head of the Department, Computer Science and Engineering, Srinivasa Ramanujan Centre for his encouragement during our project work.

We also take this opportunity to express our deep sense of gratitude to our guide **Smt**. **Dr. Durga Karthik**, Assistant Professor-III, Department of CSE, our internal guide, for her cordial support, valuable information and meticulous guidance which enabled us to complete this project successfully.

We would like to place on record the benevolent approach and pain taking efforts of the project coordinators **Dr. J. Sangeetha**/Assistant Professor-III and **Ms. R. Bhavani**/Assistant Professor-II. Our sincere thanks to all faculties in the department, who have directly or indirectly helped us in completing this project.

Without the support of our parents and friends this project would never have become a reality. We owe our sincere thanks to them. We dedicate this work to all our well-wishers, with love and affection

Table of Contents

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network
ANN	Artificial Neural Network
SIFT	Scale Invariant Feature Transform
LBP	Local Border Patterns
SGD	Stochastic Gradient Descent
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol

ABSTRACT

Identification of individuals using their handwriting is one of the stimulating problems. It can be used in various number of fields like security and criminological analysis, antique documents and literature analysis, forgery and signature analysis etc. Handwriting has a crucial character in demonstration of identity and cultured traits of an individual. It is the key distinctiveness of an individual. Methodologies which are based on deep learning are verified as the versatile techniques for extracting features from huge volumes of diverse data and gives solid and accurate predictions of the underlying patterns than traditional methodologies. Automatic handwriting recognition system aids in identifying whether the stated handwriting is accurately matched and allocated to the original writer of that handwriting.

There are two modes of data capturing in any kind of writer identification, they are online and offline. Spatial coordinate values are considered mainly in the former case whereas the temporal details are considered in latter case. With respect to the textual content, offline writer identification can be categorized as text dependent and text independent. Text-dependent procedures highly emphasize on context and semantics in the handwritten image, so it needs an image having fixed text content and evaluates the resemblance of the input image with already listed prototypes for this purpose. Contrarily, text independent handwriting recognition emphasizes on the images that do not hang on the text content which is fixed.

Since the convolutional neural networks and their state of art architectures are very powerful in extracting deep features and their huge success in the fields of computer vision made us to implement the proposed model of writer identification system. It is done in three stages. The first stage is Image pre-processing where various filters are applied on images to remove noise. The second stage is feature extraction using AlexNet based CNN architecture where the patterns are learnt and deep features are extracted in order to accurately classify different handwriting styles and the final stage is writer classification using Artificial Neural Networks (ANN) followed by evaluation of the model.

KEYWORDS: Handwriting Recognition, Writer Identification, AlexNet, Convolutional Neural Networks, Artificial Neural Networks, Image Augmentation.

CHAPTER 1

SUMMARY OF BASE PAPER

Base paper title: Automatic Visual Features for Writer Identification: A Deep Learning Approach
by: Arshia Rehman- Higher Education Department, GGPGC, Pakistan, Saeeda Naz-The University of Sydney, Sydney, NSW 2006, Australia, Muhammad Imran Razzak and Ibrahim A Hameed-Department of ICT and Natural Sciences, Norwegian University of Science and Technology, 6009 Alesund, Norway
Journal Name: Deep learning,
Published on: 21 January 2019.
Published in: IEEE Access (Volume 7)

1.1 Introduction:

Automatic handwriting recognition and writer identification system helps in determining the valid writer of a particular handwritten text image among a large number of writers whose handwriting has been already trained. Many research scholars had shown interest in this domain because of its large number of applications like security and criminological analysis, antique documents and literature analysis, etc. The traditional approaches extract features from handwritings using hand designed and manual calculations. They used techniques like scale invariant feature transforms, local binary patterns, wavelet transforms etc, where the final feature vectors are extracted from local patches of handwriting images and constructed models like bag of words etc. With the advent of deep learning algorithms like CNNs, ANNs etc, the feature extraction work has become much easier. Since the convolutional neural networks and their state of art architectures like AlexNet, VGG, GoogleNet etc are very powerful in extracting deep features and their huge success in the fields of computer vision made several research scholars to employ CNN techniques for writer identification.

This paper presents a deep learning based automatic feature extraction model using AlexNet CNN architecture, which is a state of art architecture model trained on ImageNet database. In this project the pre-processed handwriting images of 10 different writers are trained on CNN architecture inspired from AlexNet and classification is done using Artificial Neural Networks.

Convolutional Neural Networks: A Convolutional Neural Network is a popular deep learning algorithm that takes an input image, process pixel data, undergoes learning by assigning weights and biases through optimizers and large number of back propagations, to extract features from the image and becomes capable of distinguishing each another. The CNN architecture is very similar to the arrangement pattern of neurons present in the brains of human beings and got inspired from organization of the visual cortex. Receptive field is the visual area where discrete neurons are restricted to receive stimuli and responds to them. This visual area is a collection of such fields that gets overlapped.

1.2 Architecture Diagram:

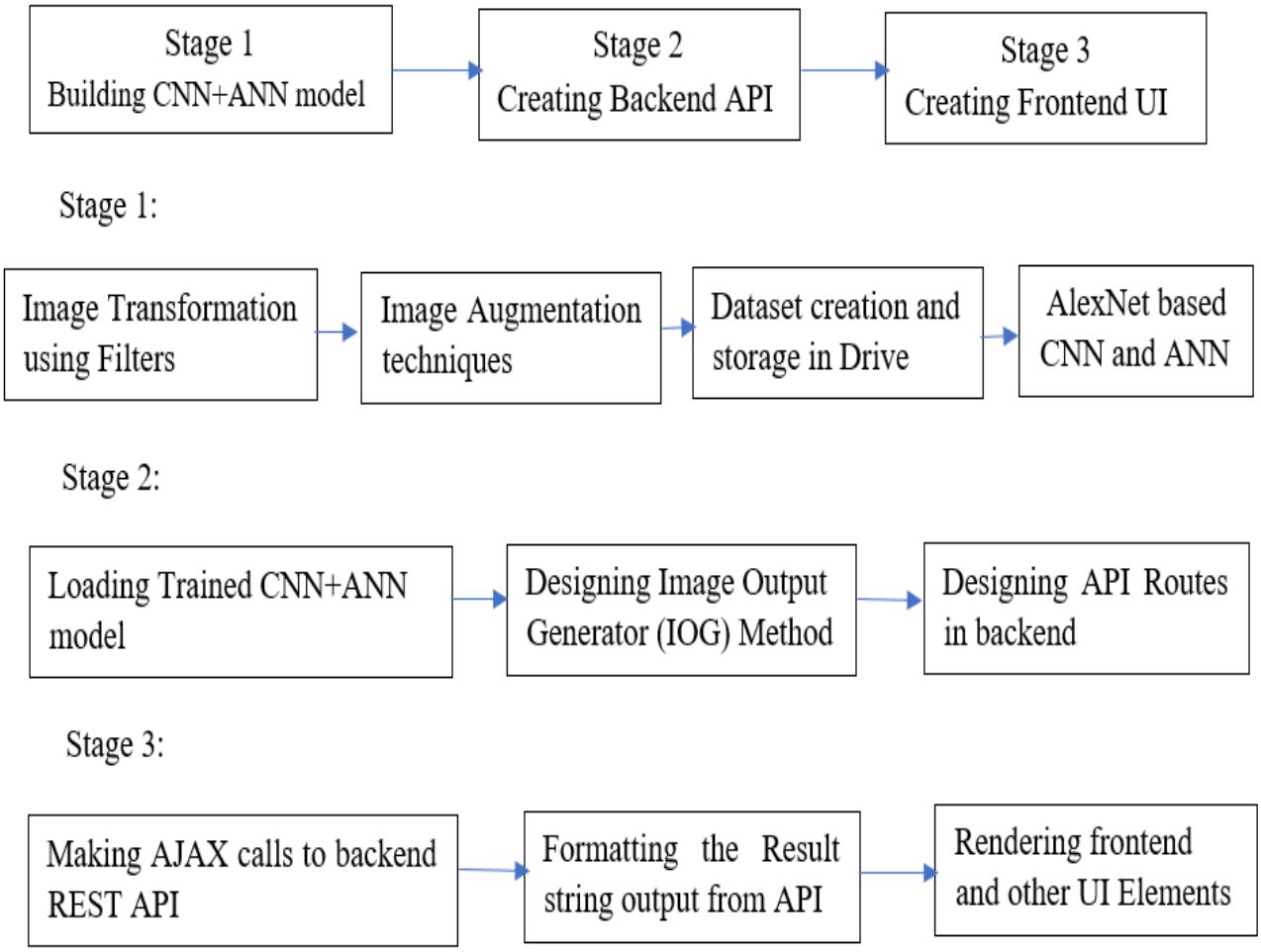


Fig 1.1 Architecture of the project

The above image fig 1.1 describes about the architecture of the project. The entire architecture is divided into three stages. Each stage is implemented by a sequence of substages referred as pipelines. Stage 1 is all about constructing the classification model, where the images are pre-processed, augmented and trained on AlexNet based CNN architecture. It comprises of 4 pipelines. The next two stages are for constructing a web application using this trained model. Stage 2 is for designing the backend API of the web application, which is a sequence of 3 pipelines and Stage 3 is for designing the frontend UI of the web application. It is executed as a sequence of 3 pipelines

AlexNet Based CNN architecture implemented for training the Handwriting Images

Layers	Inputs	Kernel	Filters	Strides	Output
Conv1	224*224*3	7*7	32	2*2	109*109*32
Pooling	109*109*32	3*3	32	2*2	54*54*32
Conv2	54*54*32	5*5	64	2*2	25*25*64
Pooling	25*25*64	3*3	64	2*2	12*12*64
Conv3	12*12*64	3*3	96	1*1	10*10*96
Pooling	10*10*96	2*2	96	2*2	5*5*96
Conv4	5*5*96	1*1	128	1*1	5*5*128
Pooling	5*5*128	3*3	128	2*2	2*2*128
Flatten	2*2*128	-	-	-	512
ANN-FC1	512	-	-	-	256
ANN-FC2	256	-	-	-	256
Output	256	-	-	-	10

Table 1.1 CNN Architecture

The above table 1.1 shows the architecture of the convolutional neural network. There are 4 convolution layers, each layer followed by a pooling layer. A flatten layer is used to extract the feature vector and it is sent to two sequential fully connected layers. Finally, there is an output layer having 10 nodes. Each layer has its own configurations of various parameters like kernel size, no of filters, strides size, padding used etc. The formula for the output dimension is shown below.

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in}: number of input features
n_{out}: number of output features
k: convolution kernel size
p: convolution padding size
s: convolution stride size

Formula for calculating output dimension

In every convolution layer, the kernel is convolved over the input matrix, where the matrix values get multiplied with the values of filter and gets added. There can be any number of such filters. Stride is the shift length that is made by the kernel. In the entire architecture, we kept valid padding i.e. we didn't add extra padding to any matrix in the network. Hence p=0 in the entire network. The handwriting image is the input for the network, it is a matrix of size 224*224*3. As per the above formula the output matrix dimension should be 109*109*32 i.e. ((224+0-7)/2) +1= (217/2) +1= floor (108.5) +1= 108+1 =109. There are 32 filters used, hence output is 109*109*32. Pooling layer is used to downscale the image, by extracting the maximum value from each submatrix of kernel size. Flatten layer converts 2d matrix into a 1d feature vector which is sent to a dense network called as Fully connected layers. Finally, the last FC layer is connected to output layer having 10 nodes which represents 10 different writers.

1.3 PROPOSED METHODOLOGY:

This paper proposed a model for writer identification, using Convolutional Neural Networks. CNNs are mainly used for their abilities of features extraction. The main feature that the CNN algorithm uses in this writer identification system is 'Raw Pixel'. Apart from the pixel value, the features like i) Spacing between letters, words, sentences and lines. For ex: cursive handwriting letter space is zero ii) Thickness of letter edges, since pressure applied by every writer while writing will be different, this leads to different thickness in letter edges. iii) Curvature along with depth and height of the letters etc.

The extracted feature vector is sent to Artificial Neural Networks or Fully Connected layers for performing handwriting classification. The obtained training and testing accuracy are considered as the evaluation parameter for the model. The entire workflow can be briefly divided into 3 stages, each stage has 3-4 sub stages called as pipelines.

Stage 1: Development of the Classification Model

This stage is mainly for building the classification model and its evaluation. It is broadly divided into 4 pipelines. Each pipeline has its own well-defined task.

Pipeline 1: In this stage the main objective is to remove the noise and variations in background lighting, since the images are differently captured by every writer. This is done using Gaussian blur filter and finally extracting the edges alone from the handwriting image. This is done using Canny Edge Detector.
Gaussian Blur filtering is an operation in which a Gaussian filter instead of a box filter is convolved over an image. It is a low-pass filter which eliminates the high-frequency components and reduces the unwanted noise in the image.
The Canny edge detector is an edge detection operation that uses multi-stage algorithm to detect a extensive range of edges in images.

Pipeline 2: The main objective of this stage is Image Augmentation. Image augmentation is a procedure which is used to theatrically enlarge the size of a training and testing dataset by creating altered variants of images in the dataset. There are various augmentation techniques like i) Rotation ii) Scaling iii) Random distortion iv) Flipping v) Zooming vi) Horizontal and Vertical shifting, etc
To increase the image samples, for training the model efficiently, the paper proposed two augmentation techniques, rotation and random distortion followed by random patching of images to obtain a size of 224*224. The following are the operations performed on data.
1)50 images by rotation with 12 degrees clockwise and 14 degrees anti clockwise.
2)50 images by rotation with 10 degrees clockwise and 8 degrees anti clockwise.
3)50 images by distortion with grid width 10 units, grid height 10 units and magnitude 12 units.
4)50 images by distortion with grid width 4 units, grid height 4 units and magnitude 8 units.
5)random slices of 224*224 are extracted for every image.

Pipeline 3: Creation of appropriate training dataset and testing datasets is the main objective.
In this project, we used 10 writers handwriting, i.e. a total of 10000 images, 1000 images for every writer, of size 224*224. For every writer 800 images are used for training the model and 200 images are used for testing or validating. So, a total of 8000 images are used for training and a total of 2000 images are used for testing. The created datasets are uploaded in Google Drive, which is considered as a database for this project. Uploading in google drive is efficient for accessing data while working in google colab.

Pipeline 4: Constructing a Convolutional Neural Network model based on AlexNet architecture is essential for training the handwriting images for classification. The entire neural network is designed using TensorFlow and keras library. The network architecture is shown in the figure 1.2 The network is constructed by referencing AlexNet State of Art CNN architecture. It has 4 convolution layers followed by 4 pooling layers, each after every convolution layer. Feature vector is obtained from flatten layer and sent to two fully connected layers. Other configurations of the network are described below.

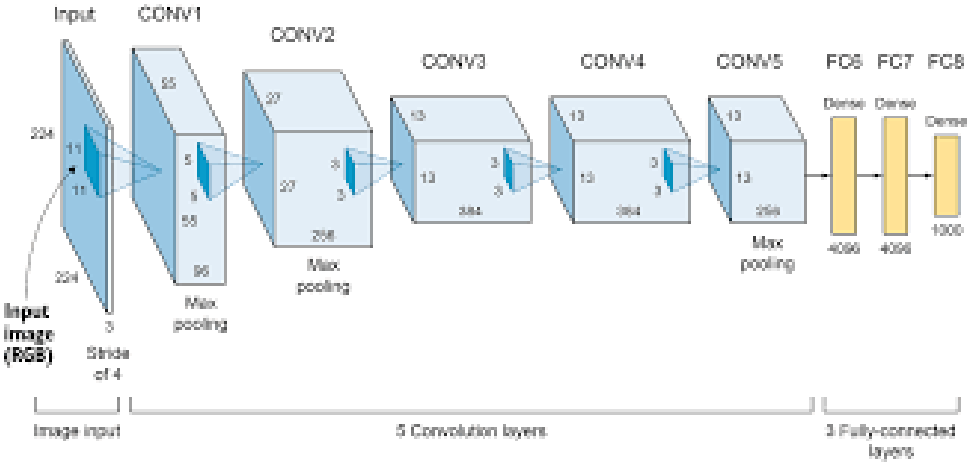


Fig 1.2 AlexNet CNN architecture

Activation Functions used:-

In convolution layer:- Relu

In Output layer:- Softmax

Optimizer used: - Stochastic Gradient Descent (SGD)

Loss Function used: - categorical_crossentropy

No of Epochs: - 20

Evaluation Metric used: - accuracy

To reduce overfitting, Batch Normalization and Dropout of 40% are used in CNN and ANN layers respectively.

Stage 2: Creating backend REST API of web application

The main objective of this stage is to build a REST API for the developed CNN model using flask. It consists of 3 pipelines.

Pipeline 1: - The trained AlexNet CNN model is stored in disk and is loaded whenever necessary. So, the main objective of this stage is to extract the Handwriting classification model from the disk and gets loaded with all the pretrained weights and can be used for further testing.

Pipeline 2: - Designing an Image Output Generator (IOG) method, which gives the classification results in the form of 20 votes, which were distributed among the possible writers, for an uploaded handwriting image of any writer. For ex: for any image if the model is 100% confident on a particular writer's handwriting, all the 20 votes are given to that writer, else it gets divided.

Pipeline 3: - Designing a Flask based API having 2 routes
i). '/' which is a GET route for displaying home page of the app
ii). '/predict' which is a POST route, that calls predict function and sends the result string to the frontend.

Stage 3: **Creating frontend user interface of web application**

The main objective of this stage is to design a simple user interface for testing the model by uploading handwriting images. It consists of three pipelines.

Pipeline 1: - Creating UI elements for uploading handwriting and making AJAX calls to the REST API to obtain the result for the uploaded handwriting.

Pipeline 2: - Formatting of the result string is done in this pipeline i.e. the writer name which got a greater number of votes out of 20 are considered as 1st priority and displayed appropriately.

Pipeline 3: - Designing other UI elements of the front end and rendering them appropriately for a better user experience.

1.4 CORRECTNESS:

To prove correctness of any deep learning algorithm, there are certain metrics like accuracy, precision, recall, loss, validation loss, validation accuracy etc to evaluate the model and which in turn prove its correctness. Various metrics should be considered in various situations. In image

classification using CNN mainly in every epoch we will be considering parameters like accuracy, validation accuracy, loss and validation loss.

Accuracy: - Accuracy is considered as a significant metric in any classification project. It compares the classified image with another data source that is considered to be accurate or ground truth data. Through field work and experimentation ground truth can be collected; however, this is highly expensive and more time taking. Briefly it can be considered as how much correct a classified result is, with respect to correct result. Higher the accuracy score, better the learning and performance of the model

Validation Accuracy: - It is nothing but accuracy score obtained on validation dataset. It shows how well the model is behaving on the unknown data after getting trained. Higher the validation accuracy, better the classification on unknown images.

Loss: - In order to optimize any algorithm, the function that is used to figure out a solution, is often considered as an objective function. The main aim is to maximize or minimize this objective function, which means that the solution which we are finding should have either highest score or lowest score respectively. Naturally, in neural networks, the error must be as minimum as possible. This objective function is considered as loss function or cost function and the value obtained by this loss function is simply called as loss.

Validation Loss: - Loss obtained on validation data is considered as validation loss. For a better generalized model, the loss and validation loss should be as minimum as possible. Higher validation loss and lower validation accuracy is a result of overfitting.
While training the CNN model on certain epochs, the following trends can be seen on these values.

1. Validation loss slowly starts to grow and validation accuracy slowly starts falling. This means model is stuffing values and not learning
2. Validation loss slowly starts to grow and validation accuracy also slowly raises. This must be a situation of overfitting or miscellaneous probability values in scenarios where SoftMax activation function is being used in the output layer
3. Validation loss starts decreasing and validation accuracy starts increasing. This is also acceptable which means that the model built is learning well and working fine.

While training our model we observed the 3rd trend, i.e. validation loss is decreasing and validation accuracy is increasing. This shows that the CNN model is generalized and not overfitting. In the 1st epoch, the accuracy is 0.3341, validation accuracy is 0.1870, loss is 2.0037 and validation loss is 3.4596, while in the final epoch i.e. 20th epoch, the accuracy is 0.9919, validation accuracy is 0.9900, loss is 0.0487 and validation loss is 0.0421. With these metrics we can say that the model is not overfitting and it is a highly generalized model. This proves the correctness of our CNN based Handwriting classification model.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

2.1 EXISTING METHODOLOGIES:

The traditional methodologies of offline text-independent handwriting identification have been classified into two types based on the extracted features. The first approach is based on manually designed features and the second approach is based on deep learning-based feature extraction approaches [1][8][6]. Some hand-designed features such as Scale-Invariant-Feature-Transforms [7], Local Binary Patterns [3], etc were extracted from the local patches of the handwriting images and models like bag of words [4] has been constructed to represent the obtained final feature vectors. To extract features from handwriting images for writer identification, filters like Gabor filters and local binary patterns were used. Other than this Most of the works are based on traditional methods like finding chi^2 distance between letters, using wavelet transformation which is computationally intensive, and other methods like grapheme generation which is time consuming. Many other researchers like Christlein et al. used CNN algorithms [2][6] but followed a technique of extracting features from local structures [5] which resulted in high feature loss and low accurate models. Tang and Wu used Bayesian networks and CNN algorithm on ICDAR 2013 dataset [9].

Reference	Features	Model	data-set	Accuracy (%)
Fiel and Sablatnig [20]	Raw pixels by CNN	χ^2 distance	ICDAR 2011 ICDAR2013 CVL	98.6 97.6 40.5
Christlein et al. [19]	Raw pixels by CNN	Exemplar SVM	ICDAR 2013 CVL	0.989 0.994 mAP
Christlein et al. [2]	SIFT, Raw pixels by ResNet	Exemplar SVM	ICDAR 2017-Historical-WI	88.9
Christlein et al. [18]	Raw pixels by LeNet and ResNet	Exemplar SVM	ICDAR13 CVL KHATT	99.6 99.5 99.6
Yang et al. [24]	Raw pixels by CNN	CNN	CASIA-OLHWDB1.0	99.52
Xing and Qiao in [25]	Raw pixels by CNN	CNN	IAM HWDB	98.80 93.45
Tang and Wu [26]	Raw pixels by CNN	Baysian Network and CNN	ICDAR2013 CVL	99.0 99.7

Fig 2.1 Existing Works on Writer Identification

These methodologies didn't consider, custom (real time) writers' handwritings rather used benchmark datasets handwritings as shown above. They also didn't used any modern State of art CNN architecture models like ResNet, VGG, AlexNet, GoogleNet etc which are highly accurate and powerful.

2.2 PROPOSED METHOD: MERITS &DEMERITS

In this paper, we follow a technique of extracting features from Global structures than local structures, due to which the feature loss is minimized and performance gets improved. Apart from it we used a CNN architecture inspired from the famous State of Art CNN architecture, AlexNet.

The entire project is carried out on custom writers than using the existing old benchmark datasets like QUWI, ICDAR-13 etc., on which more research has already been done. The proposed model performs better than existing models in the areas of Image Augmentation, efficient ways of storing the images, better CNN architecture with a very high accuracy etc.

Analysis on Image Augmentation techniques:

The proposed scheme used basic as well as latest image augmentation techniques like rotation and random distortions respectively. With this, the final datasets have high variant images which is very useful for avoiding overfitting and feature loss. With the technique of random distortion in the line alignments, we can easily relate it with the daily life scenario where writers always won't write uniformly. We followed random cropping technique over sliding window technique in order to ensure the capturing of as many features as possible. With sliding window technique there is loss in features like word gap, curvature and line height etc. It also avoids overfitting of the model.

Architecture Analysis:

The architecture is built by taking reference of AlexNet CNN architecture. Since the handwriting images that we are using in this paper are not a part of the ImageNet dataset, the actual weights used in AlexNet architecture didn't gave a valid accuracy. Hence Transfer learning techniques cannot be applied directly. Instead, tweaking the configurations like number of filters, strides etc in each layer by keeping the same architecture worked well for us. Usage of Stochastic Gradient Descent optimizer over other optimizers like Adam which was used in other existing works gave a relatively less loss. The number of convolution layers and pooling layers can still be optimized. The total trainable parameters can still be reduced.

CHAPTER 3

SOURCE CODE

3.1 HandwritingClassifierModel.py

Code for creating a handwriting classifier model on 10 writers handwriting

Pipeline 1:

```
from google.colab import drive
drive.mount('/content/gdrive')
!pip install mapper
import cv2
import numpy as np
import mapper
import matplotlib.pyplot as plt
from PIL import Image
def show_image(image_object,name):
  plt.figure(figsize=(14, 7))
  plt.title(name)
  plt.imshow(image_object)
edged_image=""
def preprocess1(image_path):
  image=cv2.imread(image_path)   #read in the image
  #image=cv2.resize(image,(1300,800))
  #resizing because opencv does not work well with bigger images
  show_image(image,"Input_Image")
  gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)  #RGB To Gray Scale
  show_image(gray,"Grayscale_Image")
  blurred=cv2.GaussianBlur(gray,(5,5),0)
  #(5,5) is the kernel size and 0 is sigma that determines the amount of blur
  show_image(blurred,"Blurred_Image")
  edged=cv2.Canny(blurred,30,50)  #30 MinThreshold and 50 is the MaxThreshold
  show_image(edged,"Edged_Image")
  new_image_name=input("Enter name for the processed image to get saved..")
  cv2.imwrite(new_image_name,edged)
  global edged_image
  edged_image=new_image_name
preprocess1("/content/ashishhw.jpeg")
```

Pipeline 2:

```
!pip install Augmentor
#ZIPPING FOLDERS AND DOWNLOAD THE ZIP FILE AUTOMATICALLY
import shutil
from google.colab import files
def make_zipfolder(zipfoldername,folder_path):
  shutil.make_archive(zipfoldername, 'zip', folder_path)
  files.download(zipfoldername+".zip")
# DELETING A FOLDER WITH CONTENT DIRECTLY
def delete_folder(path):
  shutil.rmtree(path)
#ZIPPER CELL
#make_zipfolder("nags_augmented_imgs",output_directory)
make_zipfolder("ashishcroppedimgs","/content/ashishcroppedimgs")  #give anyname that u like
#delete_folder('/content/origimgs')
#FOR DISPLAYING SINGLE IMAGE
def view_image(path):
  print("path=",path)
  img = cv2.imread(path)
  img_cvt=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  plt.imshow(img_cvt)
  plt.show()
view_image("/content/ashishedged.jpeg")
#FOR SHOWING ALL THE IMAGES IN A FOLDER
import os
directory = "/content/origimgs"
output_directory="/content/origimgs/output"
def display_images(directory,cnt=50):
  no=0
  for filename in os.listdir(directory):
    print(filename)
    if(filename==".ipynb_checkpoints"):
      continue
    view_image(directory+"/"+filename)
    no=no+1
    if(no==cnt):
      break
#RENAMING FILES IN A FOLDER
def rename_files(dir_path,operation,execs):
  count=1
  for filename in os.listdir(dir_path):
    if(filename[:8]=='origimgs'):
```

```
      dest=operation+"-"+str(execs)+"-"+str(count)+".jpeg"
      os.rename(dir_path+"/"+filename,dir_path+"/"+dest)
      count+=1
import Augmentor
def init_augmentor(flag):
  if(flag):
    os.makedirs('/content/origimgs')
    shutil.move(edged_image, '/content/origimgs/')
  p = Augmentor.Pipeline("/content/origimgs/")
  return p
def rotate(no_of_images=10,lr=10,rr=10,execs=1):
  p=init_augmentor(0)
  p.rotate(probability=0.9, max_left_rotation=lr, max_right_rotation=rr)
  p.sample(no_of_images)
  rename_files(output_directory,"rotate",execs)
def random_distortion(no_of_images=10,gridwdth=4,gridht=4,mag=15,execs=1):
  p=init_augmentor(0)
  p.random_distortion(probability=1, grid_width=gridwdth, grid_height=gridht, magnitude=mag)
  p.sample(no_of_images)
  rename_files(output_directory,"distortion",execs)
def zoom(no_of_images=10,execs=1):
  p=init_augmentor(0)
  p.zoom_random(1, percentage_area=0.5)
  p.sample(no_of_images)
  rename_files(output_directory,"zoomed",execs)
init_augmentor(1)
rotate(50,14,12,1)
rotate(50,8,10,2)
random_distortion(50,10,10,12,1)
random_distortion(50,4,4,18,2)
#RANDOM SILCES OF 224*224
def get_random_crop(image, crop_height, crop_width):
    max_x = image.shape[1] - crop_width
    max_y = image.shape[0] - crop_height
    x = np.random.randint(0, max_x)
    y = np.random.randint(0, max_y)
    crop = image[y: y + crop_height, x: x + crop_width]
    return crop
path3="/content/origimgs/output/" #path of folders having 200 images
def sliding_window(no_of_crops,foldername,htsize=224,wdsize=224):
  cnt=1
  for imagename in os.listdir(path3):
    if(imagename=='.ipynb_checkpoints'):
```

```
      continue
    example_img=path3+imagename
    iimg = cv2.imread(example_img)
    for i in range(no_of_crops):
      iimg2=get_random_crop(iimg,htsize,wdsize)
      cv2.imwrite('/content/'+foldername+'/cropimg'+str(cnt)+'.jpeg', iimg2)
      #cv2.imwrite('/content/croppedimgs68/cropimg'+str(cnt)+'.jpeg', iimg2)
      cnt=cnt+1
cropped_folder=input("Enter a name for the empty folder to save 224*224 slices")
os.makedirs("/content/"+cropped_folder)
sliding_window(5,cropped_folder)
#check folder then go above upto zipper cell..zip this folder and download
make_zipfolder(cropped_folder,"/content/"+cropped_folder)
#UNZIPPING A ZIPPED FILE INTO A FOLDER
!unzip augimgs1.zip -d augmented_imgs1
```

Pipeline 3:

```
#os.makedirs("/content/gdrive/My Drive/datasetedgedhws/datasetedged/train_set/aswadHW")
#os.makedirs("/content/gdrive/My Drive/datasetedgedhws/datasetedged/test_set/aswadHW")
def upload_to_drive(foldername):
  testdir="/content/gdrive/My Drive/datasetedgedhws/datasetedged/test_set/"+foldername
  traindir="/content/gdrive/My Drive/datasetedgedhws/datasetedged/train_set/"+foldername
  os.makedirs(traindir)
  os.makedirs(testdir)
  for cnt in range(1,1001):
    if(cnt<=800):
      #print("/content/"+cropped_folder+"/cropimg"+str(cnt)+".jpeg")
      shutil.move("/content/"+cropped_folder+"/cropimg"+str(cnt)+".jpeg", traindir)
    else:
      shutil.move("/content/"+cropped_folder+"/cropimg"+str(cnt)+".jpeg", testdir)
foldernameindrive=input("Enter the name of the folder to be created in drive")
upload_to_drive(foldernameindrive)
delete_folder('/content/'+cropped_folder)
delete_folder('/content/origimgs')
```

Pipeline 4:

```
import tensorflow as tf
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten, Conv2D,
MaxPooling2D,BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
test_datagen = ImageDataGenerator()
```

```
test_set =
test_datagen.flow_from_directory('/content/local_datasetedged/datasetedged/test_set',class_mode='categori
cal',target_size = (224, 224), batch_size=96)
train_datagen=ImageDataGenerator()
train_set=train_datagen.flow_from_directory('/content/local_datasetedged/datasetedged/train_set',class_mo
de='categorical',target_size=(224,224), batch_size=96)
model4 = Sequential()
# 1st Convolutional Layer
model4.add(Conv2D(filters=32, input_shape=(224,224,3), kernel_size=(7,7),strides=(2,2),
padding='valid'))
model4.add(Activation('relu'))
# Pooling
model4.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
# Batch Normalisation before passing it to the next layer
model4.add(BatchNormalization())
# 2nd Convolutional Layer
model4.add(Conv2D(filters=64, kernel_size=(5,5), strides=(2,2), padding='valid'))
model4.add(Activation('relu'))
# Pooling
model4.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
# Batch Normalisation
model4.add(BatchNormalization())
# 3rd Convolutional Layer
model4.add(Conv2D(filters=96, kernel_size=(3,3), strides=(1,1), padding='valid'))
model4.add(Activation('relu'))
# Batch Normalisation
model4.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='valid'))
model4.add(BatchNormalization())
# 4th Convolutional Layer
model4.add(Conv2D(filters=128, kernel_size=(1,1), strides=(1,1), padding='valid'))
model4.add(Activation('relu'))
# Batch Normalisation
model4.add(MaxPooling2D(pool_size=(3,3), strides=(2,2), padding='valid'))
model4.add(BatchNormalization())
# Passing it to a dense layer
model4.add(Flatten())
# 1st Dense Layer
model4.add(Dense(256, input_shape=(224*224*3,)))
model4.add(Activation('relu'))
# Add Dropout to prevent overfitting
model4.add(Dropout(0.4))
# Batch Normalisation
model4.add(BatchNormalization())
```

```
# 2nd Dense Layer
model4.add(Dense(256))
model4.add(Activation('relu'))
# Add Dropout
model4.add(Dropout(0.4))
# Batch Normalisation
model4.add(BatchNormalization())
# Output Layer
model4.add(Dense(10))
model4.add(Activation('softmax'))
model4.summary()
model4.compile(optimizer = 'sgd' , loss = 'categorical_crossentropy' , metrics=['accuracy'])
model4.fit(x = train_set, validation_data = test_set, epochs = 20)
result=model4.evaluate(test_set)
print("loss=",result[0])
#print("training accuracy= 0.9919")
print("testing accuracy=",result[1])
import numpy as np
from keras.preprocessing import image
classes=['ammaHW','ashishHW','aswadHW','DKmamHW','nagsHW','nandanaHW','pavanHW','pruthviHW'
,'sravyaHW','yeshwanthHW']
def test_on_a_image(path="/content/datasetedgedv1/datasetedged/test_set/64/cropimg1000.jpeg"):
  test_image = image.load_img(path, target_size = (224, 224))
  test_image = image.img_to_array(test_image)
  test_image = np.expand_dims(test_image, axis = 0)
  result = model4.predict(test_image)
  for i in result:
    count=0
    for j in i:
      z=float(format(j,'.5f'))
      print(classes[count]+"="+str(z*100)+'%',end="\n")
      count+=1
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/dkmamHW/cropimg888.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/dkmamHW/cropimg999.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/aswadHW/cropimg802.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/ammaHW/cropimg952.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/nagsHW/cropimg802.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/ashishHW/cropimg888.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/nandanaHW/cropimg1000.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/pavanHW/cropimg900.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/pruthviHW/cropimg888.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/sravyaHW/cropimg815.jpeg")
test_on_a_image("/content/local_datasetedged/datasetedged/test_set/yeshwanthHW/cropimg976.jpeg")
```

```
model4.save('alexnetcustom10writers')
"""#Utility Functions for colab"""
make_zipfolder("alexnetcustom10writers","/content/alexnetcustom10writers")
#make_zipfolder("local_datasetedged","/content/gdrive/My Drive/datasetedgedhws") #commenting download code to save time
!unzip /content/local_datasetedged.zip -d local_datasetedged
#Linux commands for copying a folder in drive to colab
!cp "/content/gdrive/My Drive/dataset" -r "/content"
#Linux commands for creating a sym link for folder in drive
!ln -s "/content/gdrive/My Drive/datasetedgedhws" "/content/edgeddatasetcopy"
```

3.2 Web application Backend.py

Code for creating a REST API for handwriting classification model using flask

Pipeline 1:

```
!pip install flask-ngrok
!pip install gevent
from flask_ngrok import run_with_ngrok
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
from flask import Flask, redirect, url_for
from __future__ import division, print_function
import sys
import os
import glob
import re
import json
#UPLOAD THIS MODEL ZIP FILE BEFORE RUNNING THIS AND THEN RUN
!unzip alexnetcustom10writers.zip -d alexnetcustom10writes
import tensorflow as tf
from tensorflow import keras
model=keras.models.load_model("/content/alexnetcustom10writes")
```

Pipeline 2:

```
import cv2
import numpy as np
from numpy import random
import matplotlib.pyplot as plt
from keras.preprocessing import image
```

```
from PIL import Image
edged_image=""
iimg2=""
lst=[]
x=0
edged=""
classes=['ammaHW','ashishHW','aswadHW','DKmamHW','nagsHW','nandanaHW','pavanHW','pr
uthviHW','sravyaHW','yeshwanthHW']
def
test_on_a_image(path="/content/datasetedgedv1/datasetedged/test_set/64/cropimg1000.jpeg"):
  test_image = image.load_img(path, target_size = (224, 224))
  test_image = image.img_to_array(test_image)
  test_image = np.expand_dims(test_image, axis = 0)
  result = model.predict(test_image)
  max_person_score=-10
  max_name="default"
  for i in result:
    count=0
    print("------------------------------------------------")
    for j in i:
      if(j>max_person_score):
        max_person_score=j
        max_name=classes[count]
      z=float(format(j,'.5f'))
      print(classes[count]+"="+str(z*100)+'%',end="\n")
      count+=1
    print("------------------------------------------------")
  return max_name
def get_random_crop(image, crop_height, crop_width):
  max_x = image.shape[1] - crop_width
  max_y = image.shape[0] - crop_height
  x = np.random.randint(0, max_x)
  y = np.random.randint(0, max_y)
  crop = image[y: y + crop_height, x: x + crop_width]
  return crop
def sliding_window(no_of_crops,htsize=224,wdsize=224):
  iimg = edged
  writers1=[]
  for cnt in range(1,no_of_crops+1):
      global iimg2
      iimg2=get_random_crop(iimg,htsize,wdsize)
```

```
      cv2.imwrite('/content/croppedimg'+str(x)+'.jpeg', iimg2)
      cnt=cnt+1
      show_image(iimg2,"final"+str(cnt))
      ans=test_on_a_image('/content/croppedimg'+str(x)+'.jpeg')
      #print("returned_ans=",ans)
      writers1.append(ans)
      #print(writers1)
  dct={}
  for i in writers1:
    if i in dct:
      dct[i]=dct[i]+1
    else:
      dct[i]=1
  print(dct)
  max=-2222
  max_voted=""
  for key,value in dct.items():
    if(value>max):
      max=value
      max_voted=key
    value=(value/20)*100
  print(max_voted,max)
  return str(dct)
def show_image(image_object,name):
  plt.figure(figsize=(14, 7))
  plt.title(name)
  if(name!='Grayscale_Image'):
    plt.imshow(image_object)
  else:
    plt.imshow(image_object, cmap='Greys_r')
def preprocess1(image_path):
  image=cv2.imread(image_path)   #read in the image
  gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)  #RGB To Gray Scale
  blurred=cv2.GaussianBlur(gray,(5,5),0)  #(5,5) is the kernel size and 0 is sigma that determines
the amount of blur
  global edged
  edged=cv2.Canny(blurred,30,50)  #30 MinThreshold and 50 is the MaxThreshold
  global x
  x=random.randint(100)
  new_image_name='edgedimg'+str(x)+'.jpeg'
  while x in lst:
```

```
    x=random.randint(100)
    new_image_name='edgedimg'+str(x)+'.jpeg'
  lst.append(x)
  global edged_image
  edged_image=new_image_name
  return sliding_window(20)
```

Pipeline 3:

```
#Before running the below code, create a empty folder ips and create a folder templates and move front end files into it i.e index.html and base.html
app=Flask(__name__,template_folder='/content/templates')
UPLOAD_FOLDER='/content/ips'
app.config['UPLOAD_FOLDER']=UPLOAD_FOLDER
run_with_ngrok(app)

@app.route("/")
def index():
  return  render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        print("Image recieved on server")
        f = request.files['file']   # Get the file from post request
        file_path = os.path.join(app.config['UPLOAD_FOLDER'] ,f.filename)
        f.save(file_path)
        preds = preprocess1(file_path)
        print("preds=",preds)
        return preds
    return None

app.run()
```

3.3 Web Application Frontend.py

Index.html

```
{% extends "base.html" %} {% block content %}
<h2>Image Classifier</h2>
<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <label for="imageUpload" class="upload-label">
            Choose...
```

```
        </label>
        <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
    </form>
    <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div>
        </div>
        <div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
        </div>
    </div>
    <div class="loader" style="display:none;"></div>
    <h3 id="result">
        <span> </span>
    </h3>
</div>
{% endblock %}
```

Base.html

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>AI Demo</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <style>
    .img-preview
{
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
```

```
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}
.upload-label:hover{
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}
@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
    </style>
```

```
</head>

<body>
   <nav class="navbar navbar-dark bg-dark">
      <div class="container">
         <a class="navbar-brand" href="#">AI Demo</a>
         <button class="btn btn-outline-secondary my-2 my-sm-0" type="submit">Help</button>
      </div>
   </nav>
   <div class="container">
      <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div>
   </div>
</body>
<footer>

   <script src="{{ url_for('static', filename='/content/static/js/main.js') }}"
type="text/javascript"></script>
   <script type="text/javascript">
      $(document).ready(function () {
   // Init
   $('.image-section').hide();
   $('.loader').hide();
   $('#result').hide();

   // Upload Preview
   function readURL(input) {
      if (input.files && input.files[0]) {
         var reader = new FileReader();
         reader.onload = function (e) {
            $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
            $('#imagePreview').hide();
            $('#imagePreview').fadeIn(650);
         }
         reader.readAsDataURL(input.files[0]);
      }
   }
   $("#imageUpload").change(function () {
      $('.image-section').show();
      $('#btn-predict').show();
      $('#result').text('');
      $('#result').hide();
```

```
        readURL(this);
    });

    // Predict
    $('#btn-predict').click(function () {
        var form_data = new FormData($('#upload-file')[0]);
        // Show loading animation
        $(this).hide();
        $('.loader').show();
        // Make prediction by calling api /predict
        $.ajax({
            type: 'POST',
            url: '/predict',
            data: form_data,
            contentType: false,
            cache: false,
            processData: false,
            async: true,
            success: function (data) {
                // Get and display the result
                $('.loader').hide();
                $('#result').fadeIn(600);
                $('#result').text(' Result:  ' + data);
                console.log('Success!');
            },
        });
    });

});
    </script>
</footer>
</html>
```

CHAPTER 4

SNAPSHOTS

4.1 classification model:

Pipeline 1

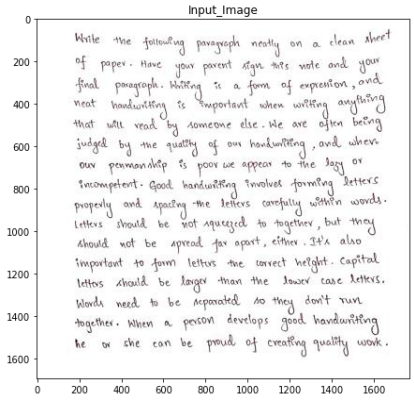


Fig 4.1 Input Handwriting Image

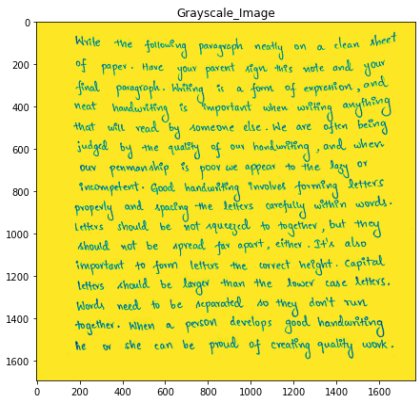


Fig 4.2 Grayscale image

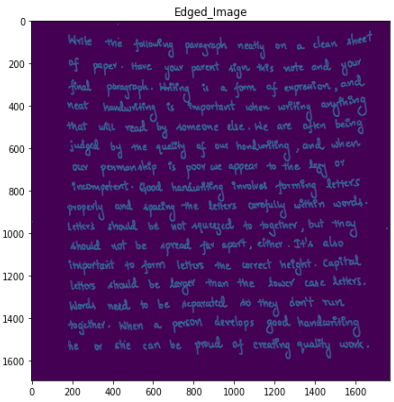


Fig 4.3 Final Edged Image 1

The above images Fig 4.1, Fig 4.2 and Fig 4.3 are outputs after applying filters like grayscale/ gaussian blur and canny edge detectors on the input handwriting image of a writer.

Pipeline 2

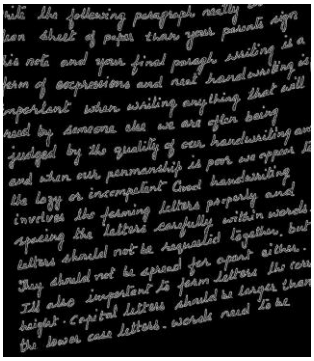


Fig 4.4 Anticlockwise Rotation

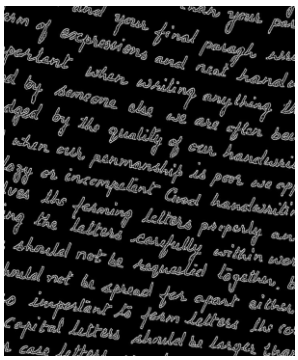


Fig 4.5 Clockwise Rotation

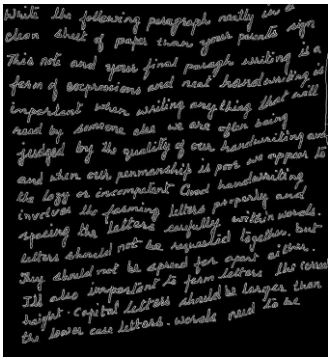


Fig 4.6 Random Distortion 1

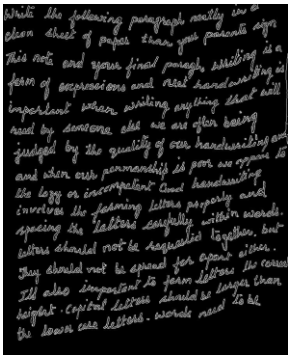


Fig 4.7 Random Distortion 2

The above images Fig 4.4 is the image after applying a rotation of 8 degree anticlockwise, Fig 4.5 is image after doing 12 degrees rotation in clockwise. Fig 4.6 and Fig 4.7 are the outputs after applying random distortion to the line alignment by grid width if 10 units, grid height of 10 units and grid width of 4 units and grid height of 4 units respectively.

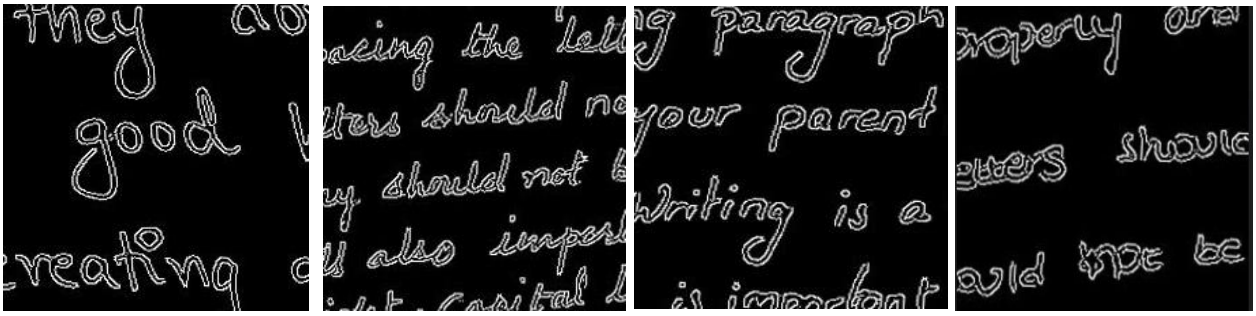


*Fig 4.8 Random crops of size 224*224*

The above shown image Fig 4.8 is a random cropped snapshot from handwriting image of a writer. The size of these snapshots is 224*224. These are the images that were sent to the CNN to get trained. For every testing image, the same sized snapshot is generated and tested on the trained model for evaluating accuracy.

Pipeline 3:

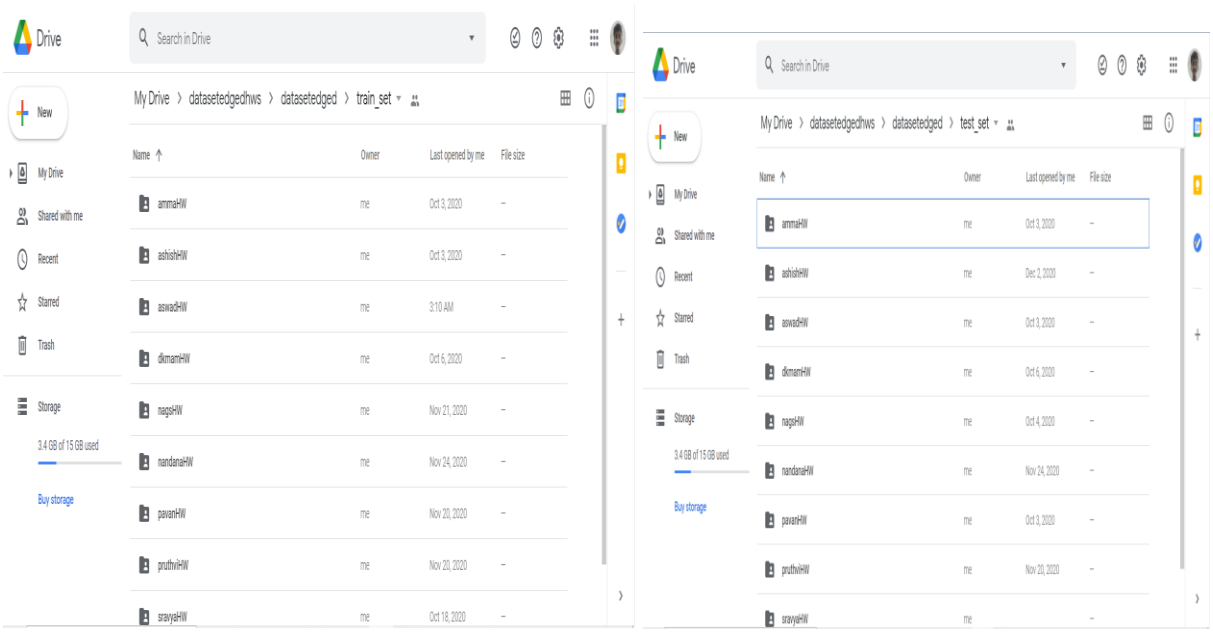


Fig 4.9 Training and Testing datasets stored in Drive

The above image Fig 4.9 are snapshots of training image dataset and testing image dataset which are stored in google drive. Loading these datasets into working environment (i.e. google colab) is very easy from google drive. There are 2 folders in the folder datasetedgedhw, which are trainset and testset. There are 10 sub folders in each of those 2 folders. Each folder is for one writer. For every writer in trainset there are 800 images. So total 8000 images and 200 images for every writer in testset. So, a total of 2000 images in testset.

Pipeline 4:

```
Epoch 1/20
84/84 [==============================] - 18s 220ms/step - loss: 2.0037 - accuracy: 0.3341 - val_loss: 3.4596 - val_accuracy: 0.1870
Epoch 2/20
84/84 [==============================] - 18s 217ms/step - loss: 0.9841 - accuracy: 0.6538 - val_loss: 1.5705 - val_accuracy: 0.4140
Epoch 3/20
84/84 [==============================] - 18s 218ms/step - loss: 0.6252 - accuracy: 0.7859 - val_loss: 0.8910 - val_accuracy: 0.6520
Epoch 4/20
84/84 [==============================] - 18s 215ms/step - loss: 0.4513 - accuracy: 0.8543 - val_loss: 0.3959 - val_accuracy: 0.8465
Epoch 5/20
84/84 [==============================] - 18s 214ms/step - loss: 0.3499 - accuracy: 0.8907 - val_loss: 0.2045 - val_accuracy: 0.9505
Epoch 6/20
84/84 [==============================] - 18s 214ms/step - loss: 0.2743 - accuracy: 0.9218 - val_loss: 0.1579 - val_accuracy: 0.9605
Epoch 7/20
84/84 [==============================] - 18s 212ms/step - loss: 0.2253 - accuracy: 0.9401 - val_loss: 0.1192 - val_accuracy: 0.9725
Epoch 8/20
84/84 [==============================] - 18s 210ms/step - loss: 0.1897 - accuracy: 0.9504 - val_loss: 0.1198 - val_accuracy: 0.9675
Epoch 9/20
84/84 [==============================] - 18s 209ms/step - loss: 0.1555 - accuracy: 0.9607 - val_loss: 0.0968 - val_accuracy: 0.9725
Epoch 10/20
84/84 [==============================] - 18s 212ms/step - loss: 0.1288 - accuracy: 0.9707 - val_loss: 0.0929 - val_accuracy: 0.9775
Epoch 11/20
84/84 [==============================] - 18s 210ms/step - loss: 0.1256 - accuracy: 0.9668 - val_loss: 0.0777 - val_accuracy: 0.9805
Epoch 12/20
84/84 [==============================] - 18s 215ms/step - loss: 0.1064 - accuracy: 0.9756 - val_loss: 0.0999 - val_accuracy: 0.9680
Epoch 13/20
84/84 [==============================] - 18s 212ms/step - loss: 0.0941 - accuracy: 0.9793 - val_loss: 0.0554 - val_accuracy: 0.9865
Epoch 14/20
84/84 [==============================] - 18s 212ms/step - loss: 0.0910 - accuracy: 0.9770 - val_loss: 0.0615 - val_accuracy: 0.9830
Epoch 15/20
84/84 [==============================] - 18s 212ms/step - loss: 0.0737 - accuracy: 0.9843 - val_loss: 0.0441 - val_accuracy: 0.9905
Epoch 16/20
84/84 [==============================] - 18s 211ms/step - loss: 0.0730 - accuracy: 0.9827 - val_loss: 0.0430 - val_accuracy: 0.9900
Epoch 17/20
84/84 [==============================] - 18s 213ms/step - loss: 0.0656 - accuracy: 0.9851 - val_loss: 0.0402 - val_accuracy: 0.9890
Epoch 18/20
84/84 [==============================] - 18s 211ms/step - loss: 0.0576 - accuracy: 0.9889 - val_loss: 0.0615 - val_accuracy: 0.9830
Epoch 19/20
84/84 [==============================] - 18s 218ms/step - loss: 0.0561 - accuracy: 0.9911 - val_loss: 0.0372 - val_accuracy: 0.9905
Epoch 20/20
84/84 [==============================] - 18s 214ms/step - loss: 0.0487 - accuracy: 0.9919 - val_loss: 0.0421 - val_accuracy: 0.9900
```

Fig 4.10 Training process of classification model

The above image Fig 4.10 shows the evaluation parameters in every epoch. From the image we can see that the loss constantly getting decreased and accuracy constantly increasing. Similar trend can be seen in validation loss and validation accuracy.

4.2 Web Application Backend:

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://9162373488bb.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
127.0.0.1 - - [01/Dec/2020 18:28:30] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Dec/2020 18:28:34] "GET /static//content/static/js/main.js HTTP/1.1" 308 -
127.0.0.1 - - [01/Dec/2020 18:28:37] "GET /static/content/static/js/main.js HTTP/1.1" 404 -
127.0.0.1 - - [01/Dec/2020 18:28:41] "GET /static/content/static/js/main.js HTTP/1.1" 404 -
127.0.0.1 - - [01/Dec/2020 18:28:41] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [01/Dec/2020 18:29:21] "GET / HTTP/1.1" 200 -
```

Fig 4.11 Server Processing

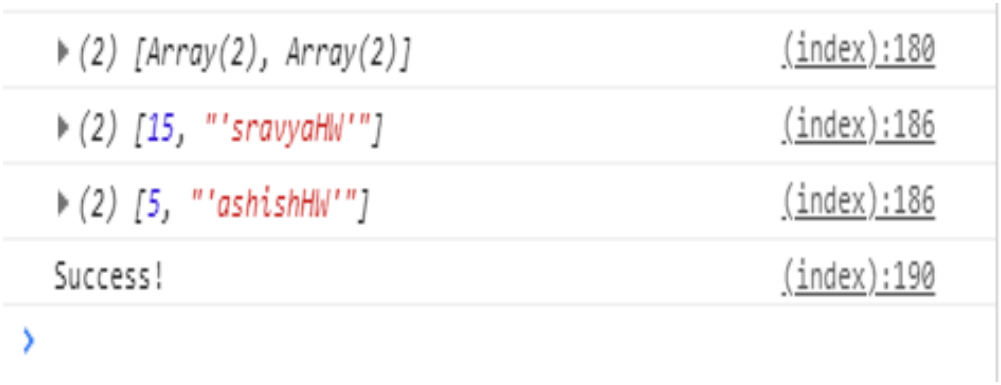


Fig 4.12 Output sent from API to frontend

The image 4.11 shows various URLs that were generated as soon as running the server. The web app can be seen live on clicking the link http://9162373488bb.ngrok.io. Various logs can also be viewed below.

The image 4.12 shows the result that was sent to frontend from API. From the image it is clearly visible that the API sent 75% chance that the input handwriting is of Sravya and 25% chance that the input image handwriting is of Ashish.

4.3 Web Application Frontend

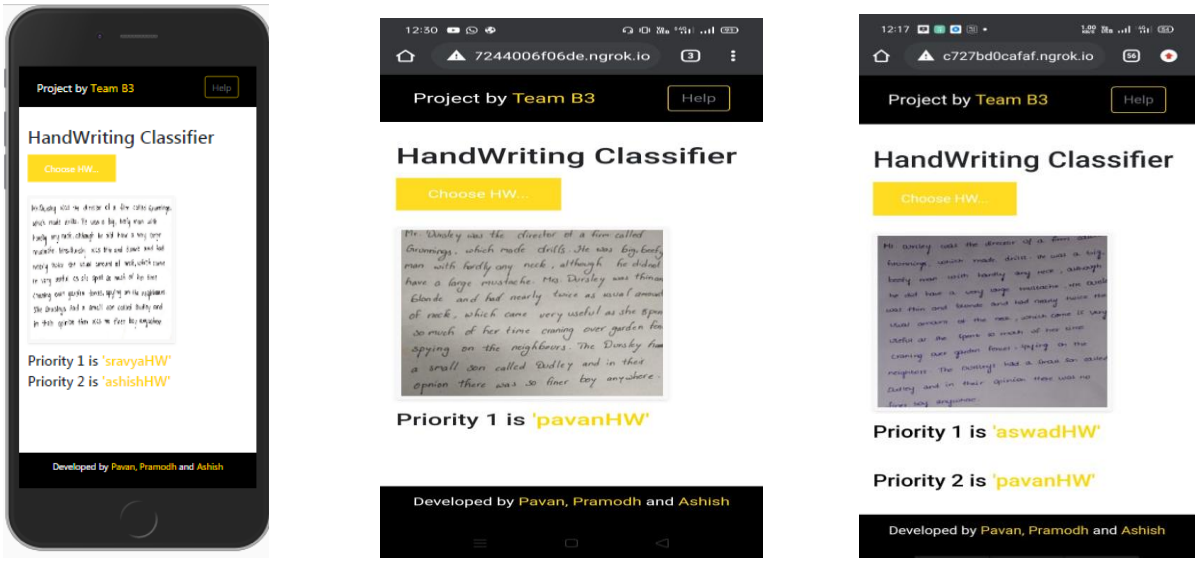


Fig 4.13 UI images of the webapp frontend

The above image fig 4.13 show the UI of webapp in various devices. We can also see the result string formatting and result displayed in terms of priority.

CHAPTER 5

CONCLUSION AND FUTURE PLANS

5.1 CONCLUSION:

Automatic Handwriting Identification is one of the fascinating research problems in the fields of document analysis and criminological analysis etc. The effective execution of handwriting identification systems can be applicable in banks, check processing, historical and forensic analysis, signature identification, graphology, legal documents and ancient manuscripts etc. In this paper we have implemented a classification model based on AlexNet state of art CNN architecture model on ten different writers and got a decent accuracy and validation accuracy without any overfitting issues. These results and analyses show the correctness of the implemented model and shows the productiveness of the proposed implementation for writer identification.

5.2 FUTURE PLANS:

Apart from AlexNet, there are many state of art CNN architecture models like VGG, GoogleNet, ResNet etc. Implementing the same model using any other such architecture can be worked out. Emphasis can be further increased in optimization of architecture by using techniques like Hyper Parameter Tuning, Normalization, Effective dropout ratios etc. Constrains on uploading user input images can be improved, so that unnecessary variations that occur in line spacing while capturing the photo can be removed. Dataset size and number of writers can be further increased. The performance of the model can also be tweaked using other optimizers and loss functions with different number of epochs and can be compared with the implemented model's configurations.

CHAPTER 6

REFERENCES

[1] A. Rehman, S. Naz, and M. I. Razzak, ‘‘Writer identification using machine learning approaches: A comprehensive review,’’ Multimedia Tools Appl., pp. 1–43, Sep. 2018. doi: 10.1007/s11042-018-6577-1.

[2] V. Christlein, M. Gropp, S. Fiel, and A. Maier. (2017). ‘‘Unsupervised feature learning for writer identification and writer retrieval.’’ [Online]. Available: https://arxiv.org/abs/1705.09369.

[3] P. Pandey and K. R. Seeja, ‘‘Forensic writer identification with projection profile representation of graphemes,’’ in Proc. 1st Int. Conf. Smart Syst., Innov. Comput. Singapore: Springer, 2018, pp. 1

[4] A. Durou, I. Aref, S. Al-Maadeed, A. Bouridane, and E. Benkhelifa, ‘‘Writer identification approach based on bag of words with OBI features,’’ Inf. Process. Manage., vol. 56, no. 2, pp. 354–366, 2019.

[5] I. Siddiqi and N. Vincent, ‘‘Combining contour-based orientation and curvature features for writer recognition,’’ in Proc. Int. Conf. Comput. Anal. Images Patterns. Berlin, Germany: Springer, 2009, pp. 245–252.

[6] V. Christlein, D. Bernecker, A. Maier, and E. Angelopoulou, ‘‘Offline writer identification using convolutional neural network activation features,’’ in Proc. German Conf. Pattern Recognit. Cham, Switzerland: Springer, 2015, pp. 540–552.

[7] S. Fiel and R. Sablatnig, ‘‘Writer identification and retrieval using a convolutional neural network,’’ in Proc. Int. Conf. Comput. Anal. Images Patterns. Cham, Switzerland: Springer, 2015, pp.

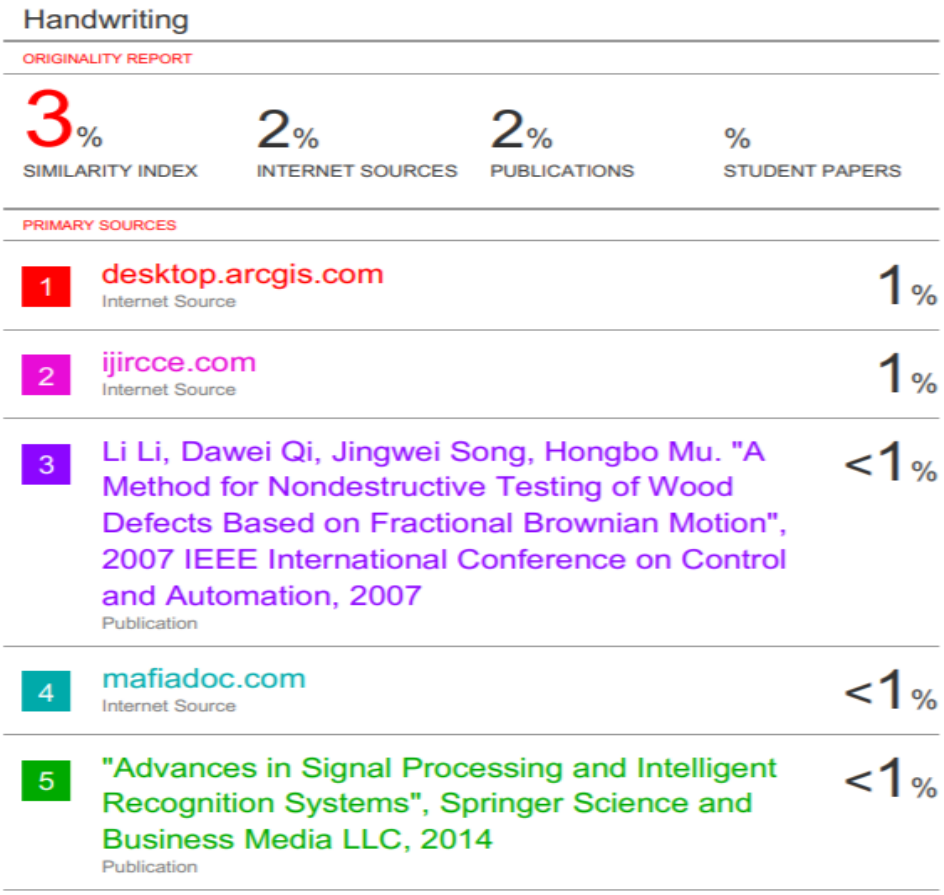
[8] S. Naz et al., ‘‘Urdu Nastaliq recognition using convolutional–recursive deep learning,’’ Neurocomputing, vol. 243, pp. 80–87, Jun. 2017.

[9] Y. Tang and X. Wu, ‘‘Text-independent writer identification via CNN features and joint Bayesian,’’ in Proc. 15th Int. Conf. Frontiers Handwriting Recognit. (ICFHR), Oct. 2016, pp. 566–571.

CHAPTER 7

APPENDIX

7.1 Similarity Check Report



Handwriting

ORIGINALITY REPORT

3%	2%	2%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	desktop.arcgis.com Internet Source	1%
2	ijircce.com Internet Source	1%
3	Li Li, Dawei Qi, Jingwei Song, Hongbo Mu. "A Method for Nondestructive Testing of Wood Defects Based on Fractional Brownian Motion", 2007 IEEE International Conference on Control and Automation, 2007 Publication	<1%
4	mafiadoc.com Internet Source	<1%
5	"Advances in Signal Processing and Intelligent Recognition Systems", Springer Science and Business Media LLC, 2014 Publication	<1%