

# Comparison of deep convolutional neural architectures for handwritten numerals



By:

**Ameera Arif**

**MS(CS) 318556**

Supervisor:

**Dr. Pakeeza Akram**

Masters of Computer Science

School of Electrical Engineering and Computer Sciences (SEECS)

National University of Sciences and Technology, NUST H-12, Islamabad

September 2021

# THESIS ACCEPTANCE CERTIFICATE

It is certified that final copy of MS/MPhil thesis written by Miss Ameera Arif, (Registration No MS (CS) 318556), of School of Electrical Engineering and Computer Science (SEECs) has been vetted by undersigned and is found complete in all respects as per NUST Statutes/Regulations. It is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature:



Name of Supervisor: Dr. Pakeeza Akram

Date: 26- Sep-2021

Signature (HOD): \_\_\_\_\_

Date: \_\_\_\_\_

Signature (Dean/Principal): \_\_\_\_\_

Date: \_\_\_\_\_

## APPROVAL PAGE

It is certified that the contents and form of the thesis entitled "Comparison of deep convolution neural architectures for handwritten numerals" submitted by AMEERA ARIF have been found satisfactory for the requirement of the degree.

Advisor: Pakeeza Akram

Signature: 

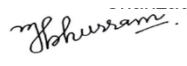
Date: 15-Sep-2021

Committee Member 1: Dr. Rabia Irfan

Signature: 

Date: 15-Sep-2021

Committee Member 2: Dr. Muhammad Khuram Shahzad

Signature: 

Date: 17- Sep-2021

Committee Member 3: Dr. Abdul Wahid

Signature: 

Date: 14-Sep-2021

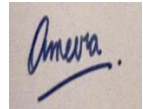
*THIS THESIS IS DEDICATED TO MY BELOVED  
PARENTS*

# Certificate of Originality

I hereby declare that this submission titled "Comparison of deep convolution neural architectures for handwritten numerals" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEecs or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEecs or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation, and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Author Name: Ameera Arif

Signature:

A small rectangular image showing a handwritten signature in blue ink on a light-colored background. The signature appears to be 'Ameera' followed by a flourish and a period.

# ACKNOWLEDGEMENT

This project would not have been possible without the support of many people. Special thanks to my advisor, Dr. Pakeeza Akram, who read my numerous revisions and helped make some sense of confusions. Also, thanks to my committee members, Dr. Khurram Shahzad, Dr. Abdul Wahid and Dr. Rabia Irfan, who offered their guidance and support.

Thanks to the National University of Sciences and Technology (NUST) for awarding me dissertation of Masters in Computer Science and providing me with the financial means to complete this project. And finally, thanks to my, parents, and friends who endured this long process with me, always offering their love and support.

# CONTENTS

<b>Introduction</b> .....	1
<b>A. Motivation:</b> .....	1
<b>B. Problem Statement:</b> .....	2
<b>C. Approach:</b> .....	3
<b>Literature Review</b> .....	4
<b>Methodology</b> .....	6
<b>A. Dataset:</b> .....	6
1. Collection of datasets: .....	6
2. Data Acquisition for Urdu numbers: .....	8
3. Other datasets (Persian dataset): .....	9
4. Other datasets (English Dataset): .....	9
<b>B. Pre-processing:</b> .....	10
<b>C. Image Augmentation:</b> .....	11
<b>D. Visualization of dataset:</b> .....	15
1. T-SNE: .....	15
2. T-SNE Visualization for Urdu dataset: .....	16
3. T-SNE Visualization for English dataset: .....	17
4. T-SNE Visualization for Persian dataset:.....	18
5. T-SNE Visualization for combination of data:.....	19
<b>E. Proposed Model</b> .....	31
1. CNN V1: .....	32
2. CNN V2: .....	34
3. CNN V3: .....	36
4. VggNet.....	38
5. GoogLeNet (Inception V3) .....	39
6. ResNet.....	40
7. Xception.....	41
<b>Results</b> .....	42
<b>A. Learning curves</b> .....	42
1. Custom built CNN models .....	42
2. Transfer Learning models .....	44
<b>B. Evaluation Metrics</b> .....	46

<b>C. Test Results</b> .....	47
<b>Discussion</b> .....	49
<b>A. Cost Calculations</b> .....	49
<b>B. Comparison with existing other techniques</b> .....	50
<b>Conclusion</b> .....	52
<b>A. Future Work</b> .....	52
<b>References</b> .....	54



# LIST OF ABBREVIATIONS

ABBREVIATION	MEANING
CNN	Convolution Neural Network
MNIST	Modified National Institute of Standards and Technology
RESNET	Residual Network
VGGNET	Visual Geometry Group Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
OCR	Optical Character Recognition
ICR	Intelligent Character Recognition
NADRA	National Database and Registration Authority
ADBase	Arabic Handwritten Digits Database
UNHD	Urdu Nastilque Handwritten Data
KNN	K- Nearest Neighbors
SVM	Support Vector Machine
ANN	Artificial Neural Network
RNN	Recurrent neural network
BLSTM	Bidirectional Long Short-Term Memory
MDLSTM	Multi-Dimensional Long Short-Term Memory
t-SNE	t-Distributed Stochastic Neighbor Embedding
PCA	Principal Component Analysis
ZCA	Zero phase Component Analysis
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
ADAM	Adaptive Momentum estimation
CNN V1	Convolution Neural Network Version 1
CNN V2	Convolution Neural Network Version 2
CNN V3	Convolution Neural Network Version 3
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

# LIST OF TABLES

Table 1: Urdu, PERSIAN, and English numbers.....	2
Table 2: Labels assigned to all numbers.....	7
Table 3: Augmentation techniques for our dataset .....	12
Table 4: COMPARISON OF SETTINGS MODIFIED FOR EACH ARCHITECTURE.....	38
Table 5: Comparison of accuracies of all models.....	47
Table 6: Accuracy, depth, weights and parameters of deep learning models .....	49
Table 7: Comparison with state-of-the-art models .....	50

# LIST OF FIGURES

Figure 1: Same label for Urdu, Persian, and English '1' .....	6
Figure 2: Same label for Persian and Urdu '2' .....	6
Figure 3: Different label for English '2' .....	7
Figure 4: Samples of collected data .....	8
Figure 5: A sample of the Persian digits .....	9
Figure 6: A sample of the English digits .....	9
Figure 7: A sample of crumpled paper is passed through Gaussian filter.....	10
Figure 8: T-SNE visualization of Urdu dataset.....	17
Figure 9: T-SNE visualization of English dataset.....	18
Figure 10: T-SNE visualization of Persian dataset .....	19
Figure 11: t-SNE visualization for Urdu '1', English '1' and Persian '1' .....	20
Figure 12: t-SNE visualization for Urdu '9', English '9' and Persian '9' .....	20
Figure 13: t-SNE visualization for Urdu '4', English '4' and Persian '4' .....	21
Figure 14: t-SNE visualization for Urdu '6', English '6' and Persian '6' .....	21
Figure 15: t-SNE visualization for Urdu '7', English '7' and Persian '7' .....	22
Figure 16: t-SNE visualization for Urdu '8', English '8' and Persian '8' .....	22
Figure 17: t-SNE visualization for English '0' and Persian '0' .....	23
Figure 18: t-SNE visualization for English '0' and Urdu '0' .....	23
Figure 19: t-SNE visualization for Urdu '0' and Persian '0' .....	24
Figure 20: t-SNE visualization for English '2' and Urdu '2' .....	24
Figure 21: t-SNE visualization for English '2' and Persian '2' .....	25
Figure 22: t-SNE visualization for Urdu '2' and Persian '2' .....	26
Figure 23: t-SNE visualization for English '3' and Persian '3' .....	26
Figure 24: t-SNE visualization for Urdu '3' and English '3' .....	27
Figure 25: t-SNE visualization for Urdu '3' and Persian '3' .....	28
Figure 26: t-SNE visualization for English '5' and Persian '5' .....	29
Figure 27: t-SNE visualization for Urdu '5' and English '5' .....	29
Figure 28: t-SNE visualization for Urdu '5' and Persian '5' .....	30
Figure 29: Methodology of Thesis .....	32
Figure 30: CNN V1 model .....	33
Figure 31: CNN V2 model .....	35
Figure 32: CNN v3 model .....	37
Figure 33: VGGNet model .....	39
Figure 34: GoogLeNet (Inception V3) model .....	40
Figure 35: ResNet50 model.....	40
Figure 36: Xception model summary .....	41
Figure 37: Loss and accuracy curve for CNN v1 .....	43
Figure 38: Loss and accuracy curve for CNN v2 .....	43
Figure 39: Loss and accuracy curve for CNN v3 .....	44
Figure 40: Loss and accuracy curve for VGGNet .....	45
Figure 41: Loss and accuracy curve for GoogLeNet (Inception V3).....	45
Figure 42: Loss and accuracy curve for ResNet50 .....	46
Figure 43: Loss and accuracy curve for Xception .....	46

Figure 44: Predicted labels for Xception .....48

# ABSTRACT

Urdu is a primitive and one of the most popular languages in South Asia. It has a rich history and worldwide appeal with it being spoken in more than 200 countries around the globe. But in terms of research and availability of data, it has been neglected for so long. People have built recognizers for the language but with their own specific set of data. So, with a dearth of data for research purposes we collect our own data for numerals and mix it with Persian numbers data. Urdu relies heavily on Arabic and Persian calligraphy for vocabulary, which in turn makes it a combination of loan words. Hence Urdu is thought to be an amalgamation of Arabic, Persian, Turkish and Sanskrit. Urdu, and Persian numbers are written on similar patterns with certain differences between some of their digits, so they are compatible for comparison purposes. Along with that English is the most widely spoken, written, understood language and also has huge datasets for research available. So, we employ Urdu and Persian handwritten numerals with English numerals to make a novel dataset that can be used for classification and recognition of all the three languages. We establish a unique way to exploit the similarities between these languages while keeping in view our main goal of reviving Urdu language's importance in research field.

We present a combination of Urdu, English, and Persian handwritten numbers to build deep learning-based models that can categorize the different numbers by understanding the subtle similarities between each other. Our novel dataset contains 9800 images of handwritten Urdu numerals written by over 200 individuals with their left and right hands in order to incorporate diversity in dataset. The popular Persian dataset by E. Kabir *et al.* and MNIST dataset for English numbers is incorporated to make a combined dataset of these three languages. We propose some versions of custom-built convolutional neural network to achieve remarkable accuracy in recognizing characters belonging to the proposed dataset. Along with our own proposed CNN, we use CNN architectures VGGNet, ResNet, GoogLeNet and Xception to achieve remarkable results. Our proposed models are powerful, yet simple, and obtain excellent performance when evaluated on our novel dataset.

# Chapter 1

## INTRODUCTION

### A. MOTIVATION:

The concept of optical character recognition was introduced in the early 1970s to recognize the printed text for blind people. It works by scanning printed characters using sensors to determine their edge information and then translating the input into characters [1]. Since then, it has been employed in various applications ranging from data entry to information extraction to image searching and further to build assistive technologies for handicapped people. This field gave birth to other domain specific applications like handwriting recognition, identification of handwritten digits, intelligent word recognition, real-time character recognition and furthermore. Among these, handwritten digits recognition is a classical but important problem in computer science. It has various applications in different areas and can be embedded into bigger systems. The very first OCR systems dealt with reading Latin numerals only [2]. Recent advancements in OCR lead to its adoption for recognizing cursive as well as non-cursive languages like Urdu, Portuguese, Russian, English, Mandarin, Arabic, Persian to digitalize existing printed text and handwritten documents. Urdu is a primitive and one of the most popular languages used in South Asia. With its rich history and worldwide appeal it is being spoken in more than 200 countries around the globe. There are around 80 to 90 million native speakers of Urdu [3]. There were 62 million in just India per the 2001 census making almost 6% of the population [4]. Out of these approximately 20 million reside in Pakistan or 7.57% per the 1998 census and several hundred thousand in Bangladesh [5]. It is the national language of Pakistan along with it being the official language of 6 districts in India namely Jammu and Kashmir, Uttar Pradesh, Bihar, Telangana, Jharkhand, and West Bengal. So, this ‘*living language*’ according to BBC is spoken by close to 100 million people around the world [6]. In Pakistan’s provinces, it is spoken with different dialects and accents as these provinces have other secondary languages too [7]. As a result, it is the base language of the country with it being taught as a compulsory subject in higher secondary and graduate schools.

Urdu is written in a transformed Arabic script form. In almost middle of the 8<sup>th</sup> century, Persians began to use the Arabic script with purpose of adding some letters for Persian sounds that were previously not used in Arabic language [8]. So, this brought refinement in Persian language. Besides that, Urdu script relies heavily on Arabic and Persian calligraphy for vocabulary, which in turn makes it a combination of loan words. Hence Urdu is thought to be an amalgamation of Arabic, Persian, Turkish and Sanskrit. It is bidirectional with its numerals written from left to right while the script is written in opposite direction. It has up to 40 letters in its script and 10 numerals. Same is the case for Persian which has almost 32 letters for script and 10 numbers. As is evident from our discussion that both Urdu and Persian have similar roots with certain prefatory differences in writing styles and some alphabets. Owing to their similarities recognizers for these languages face the same challenges and advances too.

On the other hand, we have English language which is not only spoken worldwide as a global language but is also the secondary official language of many countries. As of 2019, English is the official language of 82 countries [9] and over 2 billion people speak it. Because it is so widely spoken, it is referred to as *the lingua franca* of the modern era [10]. The print media or books

available worldwide is in English providing ease to people as it is most commonly understood. Computer science borrows extensively from English language as compared to other sciences where Greek and Latin are the major sources of vocabulary. Since the birth of computer sciences, computer users were limited to use English alphabets due to the lack of standards. As time progressed this ease of access developed into the new normal for the use of this language. So, it deemed necessary that we use English language along with Persian and Urdu for our study. The English language comprises of 26 letters having both upper- and lower-case forms and 10 numerals.

## B. PROBLEM STATEMENT:

Text recognizers for many languages like English, French, Persian are available in the market. For Urdu language much work has been done but is still limited due to lack of resources and required techniques. Thus, research work on Urdu and intelligent recognition of its text carries huge importance. As Urdu has similar appearance with Persian and Arabic script, so employing either one of these for research could substantially increase uses of such recognizers. We also add English language to this list because it is most widely spoken, written, understood and also has huge datasets for research available. Urdu and Persian are bidirectional languages with their script written from right direction to the left and numerals written from left direction to right. This becomes a problem in recognizing their similar patterns. Also, Urdu, and Persian numbers are written on similar patterns with certain differences between some of their digits. So, we will be focusing this thesis on Urdu and Persian numbers along with English numbers since they have vast differences but subtle similarities. Table 1 shows a sample of all these languages that show their similarities and differences.

TABLE 1: URDU, PERSIAN, AND ENGLISH NUMBERS

<b>Urdu</b>	.	۱	۲	۳	۴	۵	۶	۷	۸	۹
<b>Persian</b>	.	۱	۲	۳	۴	۵	۶	۷	۸	۹
<b>English</b>	0	1	2	3	4	5	6	7	8	9

There are huge datasets available for these languages and huge amount of work has already been done on them. But for Urdu language the research gap lies in the recognizer for handwritten Urdu numerals. There is no publicly available dataset for Urdu numbers so huge research can be done in this domain. So, the aim of this thesis is to bring Urdu language at academic level. By combining handwritten Urdu numerals dataset with handwritten Persian and English numbers, we can achieve a recognizer that can work in different domains. Its usability could be in Pakistan's NADRA system for processing national ID cards where both English and Urdu numbers could be identified. Pakistan's currency notes have both English and Urdu numbers written on them that could be categorized by using our recognizer. Although the number plates in Pakistan have been renewed recently by making them according to a pattern so that they can be identified by cameras. But still there are so many people who have not adopted this system so a recognizer that could read both Urdu and English numerals could help in this way. The development of handwritten numeral recognizer can remove the barriers faced in different writing styles, poor quality of image graphics or illegible handwriting.

The motivation behind developing Urdu numeral classifier is that majority of the work has been done on English numerals, so it is the need of the hour that techniques of deep learning are applied to our mother language ‘Urdu’. As no dataset of Urdu numerals is available publicly for research purposes so we present a novel dataset gathered specifically for this study. In this way the problems related to Urdu character recognition, OCR and Intelligent Character Recognition could be solved as efficiently as they are solved in other languages.

### **C. APPROACH:**

We present a combination of Urdu, English, and Persian handwritten numbers to build a deep learning-based model that can categorize the different numbers. Our novel dataset contains 9800 images of handwritten Urdu numerals written by over 200 individuals with their left and right hands in order to incorporate diversity in dataset. Another dataset for Persian numbers contains 13000 images of handwritten digits gathered from entrance exam to for an Iran university [11]. These images are different in sizes and angles. They are captured in varying lighting conditions containing shadows too. Thus, they contain rotation variations, noise, and distortion. The dataset for English numbers is the popular MNIST dataset by Y. LeCun of total 70,000 images written by high school students and employees [12]. The proposed data is a combination of these three datasets. It contains same label for similar digits while unique labels for different digits. We propose some versions of Convolutional neural network (CNN) to achieve remarkable accuracy in recognizing characters belonging to the proposed dataset. Along with our own proposed CNN, we use CNN architectures VGGNet, ResNet, GoogLeNet and Xception to achieve remarkable results. Our proposed models are significant, yet straightforward, and obtain performance equivalent or higher than state-of-the-art when evaluated on our novel dataset.



## Chapter 2

# LITERATURE REVIEW

One of the most challenging tasks in the field of pattern analysis and recognition is handwriting recognition. Along with its challenges, the scarcity of reasonable datasets and dearth of people ready to dive into analysis of Urdu language, the work carried out on our national language is very less. As we employ machine learning and deep learning techniques on Urdu, Persian and English numerals so our literature review consists of papers from all these domains. In 2015, N. Gautam, R. S. Sharma and G. Hazrati, [13] state work done on Eastern Arabic numerals through OCR. As Urdu and Persian numerals have their foundation in Arabic numerals, so using a combination of all these Eastern Arabic numerals can be fruitful for applications belonging to all these languages. They extract the features and classify with an accuracy of just 82.91%. S. Abdelazeem in [14] compare the problems encountered in Latin and Arabic handwritten numerals by using MNIST and Arabic Handwritten Digits Database (ADBase) respectively. They use 10 classifiers on both datasets differently which are mainly linear models, KNN, SVM, ANN and LeNet to get accuracies ranging from 92% to 96%. H. Kour and N. K. Gondhi propose a recognition system [15] based on approaches of segmentation for feature extraction, slant analysis for slant removal and dictionary search for classification. It results in recognition rate of 80.93% for isolated Urdu characters and numerals. In another study [16], J. Memon, M. Sami and R. A. Khan provide in-depth review of statistical, kernel, artificial neural network (ANN), template matching and structural methods for classification of OCR for both handwritten Urdu digits and characters. Their accuracies for all the techniques are of more than 90%.

In a very interested work presented by Ahmed, S.B. et al. [17] obtained 6.04–7.93% error rate on 700 unique text lines (including Urdu numerals and Urdu handwritten samples) after applying 1-D bidirectional long short-term memory (BLSTM) networks. In [18] L. Javed, M. Shafi, M. I. Khattak, N. Ullah present utilization of Kohonen Self Organization Maps on 6000 hand-written Urdu numerals and obtained an efficiency of 91%. Kohonen self-organizing maps are very interesting unsupervised neural networks based on competition among neural networks neurons. During the initial phase, a winner is chosen whose weight vector coincides with the input pattern. This winner and its neighbors are permitted to update their weights in a topological fashion. A work similar to this paper is done by Saad Ahmad [19] where Urdu text is integrated with Modified National Institute of Standards and Technology dataset (MNIST) to learn similar nature of patterns. They used CNN and multi-dimensional long short-term memory (MDLSTM) on UNHD (Urdu Nastilque Handwritten Data) samples by pre-training network on MNIST. Their results show highest recall 91 of 0.84 and 0.93 for precision. But a catch in their work is that they train the models on MNIST data and use these pre-trained models for Urdu numbers classification. In [20] MA Chhajro *et al.* present a comparative study of Urdu handwritten digits where they implement Support Vector Machine (SVM), K-Nearest Neighbor (K-NN) algorithm, Multi-Layer

Perceptron (MLP), Concurrent Neural Network (CNN), Recurrent Neural Network (RNN) and Random Forest Algorithm (RF) for classification of digits. Although they perform meaningful work, but their findings show inconsistencies as their dataset is not available to replicate the results. Also, they use a dataset of merely 5000 images which cannot get good results for machine learning tasks.

Finally, a recent notable technique MetaQNN discussed in [21] was put forward in 2018. It relies on reinforcement learning for the design of CNN architectures and has its roots in neuro-evolution of committees of CNN. It has an error rate of 0.44% and 0.32% when using an ensemble of the most appropriate found neural networks. Another novel technique is Capsule Net that has been presented by Geoffrey Hinton [22] recently and is implemented for handwritten Urdu numerals recognition in [23]. When a group of additional neurons are put in a typical convolutional neural network, they form a capsule. Such capsules have activity vector to denote initialization parameters for an object. For classification and recognition of handwritten Urdu digits, they achieve 98.5% accuracy which is better than CNN's 96% accuracy. In [24] techniques similar to our paper are employed on Urdu digits and characters on almost 7000 samples. Specifically, they implement the following top-notch models: LeNet, AlexNet, VGGNet, DenseNet, Xception. Their best accuracy for digits is achieved by Xception which is 98.94%.

Lastly, since data collection covers a huge portion of this paper so major work is done on preprocessing of images. Ahmed, R. et al. in [25] provide the insight on how to go forward with the data collection and preprocessing. They implement algorithms of binarization, dots removing and thinning which are used for our feature extraction phase.

## Chapter 3

# METHODOLOGY

### A. DATASET:

#### 1. COLLECTION OF DATASETS:

For machine learning and deep learning tasks, data is the essential key to receive accurate results. A well collected and refined dataset leads to exemplary evaluation of the mathematical configurations that are assessed on it. Moreover, it is pertinent to mention here that a dataset is needed to set a benchmark. For our problem at hand which is to classify similar digits as one label and distinct images according to their preferred label, we combine three datasets. One is our proposed handwritten Urdu numerals dataset while the others are Persian and English handwritten numbers dataset. For instance, the numeral for Urdu '1', English '1' and Persian '1' is written in same way, so we assigned one label to it. While Urdu '2' and Persian '2' have same shape so they are labelled in one folder and English '2' is kept distinct as is evident in Figure 2.



FIGURE 1: SAME LABEL FOR URDU, PERSIAN, AND ENGLISH '1'



FIGURE 2: SAME LABEL FOR PERSIAN AND URDU '2'

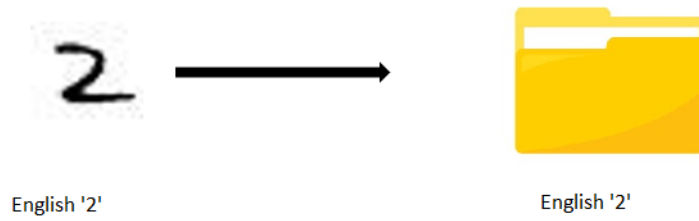


FIGURE 3: DIFFERENT LABEL FOR ENGLISH '2'

Similarly, we combined labels for Urdu '0' and Persian '0' as they both are written as a dot while English '0' is written in a different manner. The Urdu and Persian numbers '3' and '5' were combined in the same way while English '3' and '5' were given different labels. For '4', '6' and '8' of all the three languages, we gave different labels as they have distinct writing style. Lastly, '9' was assigned one label for the three languages as it is written in similar pattern. Following Table 2 gives a detailed insight on how we categorized these numerals:

TABLE 2: LABELS ASSIGNED TO ALL NUMBERS

Urdu	Persian	English	Label
•	•	0 0	Urdu and Persian same, English different label
۱ ۱	۱ ۱	۱ ۱ ۱	All same label
۲ ۲ ۲	۲ ۲ ۲	2 2	Urdu and Persian same, English different label
۳ ۳	۳ ۳	3 3	Urdu and Persian same, English different label
۴ ۴	۴ ۴	4 4	All different labels
۵ ۵	۵ ۵	5 5	Urdu and Persian same, English different label
۶ ۶	۶ ۶	6 6	All different labels
۷ ۷	۷ ۷	7 7	All different labels
۸ ۸	۸ ۸	8 8	All different labels
۹ ۹	۹ ۹	9 9	All same label

## 2. DATA ACQUISITION FOR URDU NUMBERS:

Image acquisition is the first step in any computer vision task. Acquiring data in digital form for the purpose of manipulating is known as image acquisition. It can be gathered by two methods mainly - offline method and online method. In online method the data can be entered using a tablet and a pen into the computer. Scanning the text into images via camera or a scanner can be termed as offline method. The images can be stored in any format like jpeg, bmp, png etc. In order to continue with our research work, we sought on making our own unique dataset for Urdu numbers by offline method. Our dataset contains a total of 9800 images of 10 Urdu numerals written from left and right hands. We scrutinized our data by collecting it from following categorized groups of people:

- Different age groups (12-60 years)
- Social circles (family, friends, neighbors, servants)
- Various handwriting styles

This was done subconsciously so as to bring diversity in our collection. Each person wrote 0 to 9 numerals 4 times twice with their left hand and twice from right hand. These were people belonging to various age groups and different fields of life. These people had different writing styles and by writing with their left hands it brought so much variation in our dataset that it encompassed all the behaviors of writing patterns. Also, as the dataset was not by one single person, so it was very unlikely for our model to overfit. In order to keep the uniform structure for all datasets we collected data on white blank sheets with black marker. Then we scanned the text pages on a flatbed scanner. Our inspiration for dataset collection was MNIST dataset which is for English numerals and is used all around the globe for academic purposes. In order to maintain proximity with the real world, we added ink blots, crumbled some pages, tore some other pages. Some samples of the data collected are as follows in Figure 4:

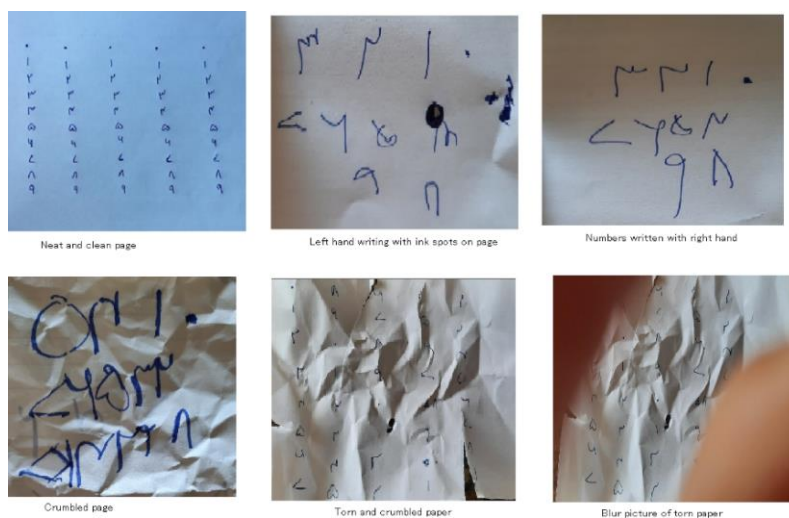


FIGURE 4: SAMPLES OF COLLECTED DATA

### 3. OTHER DATASETS (PERSIAN DATASET):

To include diversity and maintain usability of our project we included Persian numerals dataset to our previous set. Urdu and Persian numbers have certain similar numbers, so it is easier to classify such numbers as one. While some numerals of both the languages are different which again brings in heterogeneity in our data. Persian script is written from right to left and its digits are written from left to right just like Urdu. We used the popular dataset by E. Kabir and H. Khosravi which was published in their paper titled ‘Introducing a very large dataset of handwritten Farsi digits and a study on their varieties’ [11]. The dataset contains binary images of 102,352 images written by students applying to university on registration forms. These digits are in blue or black ink and the background is almost red which is binarized to get binary images. A sample of the images extracted is in Figure 5.

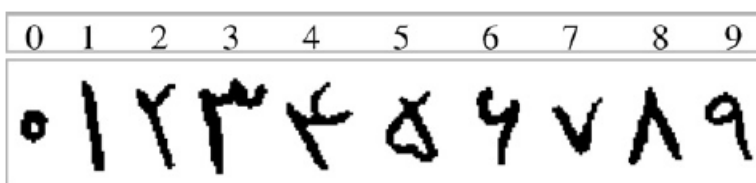


FIGURE 5: A SAMPLE OF THE PERSIAN DIGITS

### 4. OTHER DATASETS (ENGLISH DATASET):

Another dataset that we used for our study is the popular English numerals’ dataset by LeCun et al [12]. MNIST database of handwritten English digits has a training set of 60,000 examples and test set of 10,000 images. These digits were collected by 250 people of 50% high school students and 50% Census Bureau employees. The images were entered in a  $28 \times 28$  size with 256 gray levels. They were normalized and centered in a fixed size image. Sample of the collected dataset is as follows:



FIGURE 6: A SAMPLE OF THE ENGLISH DIGITS

## B. PRE-PROCESSING:

In this phase, the images acquired from all the three datasets went through some pre-processing steps so that they could be manipulated without any confusion by deep learning algorithms. Our Urdu dataset had images of smudged, torn, blotted papers and some images were also blur or disoriented. To contain all types of noise in our dataset, we crumbled some pages, added additional dots on the page and dropped ink spots so the classifier does not have a simple version of the dataset but instead has complex samples. This all was done specifically to include noise distortions because previous Urdu datasets were so clean that the models were unable to learn the real-world variations. During pre-processing, a number of techniques like image thresholding, smoothing, binarization and cropping were applied. In [26], the writers deemed it necessary to remove noise from images by converting the grayscale images into binary which we test on our data too. This resulted in less error prone results for classification and recognition. Hussain et al. [27] found in their paper that the data contained irregularities called hooks because of erratic handwriting or the movement of pen up and down inaccuracies by users who were writing on tablets. So, they applied smoothing to remove these points. Khan et al. [28] and Jan et al. [29] used filtering techniques to extract the region of interest and remove noise.

The simplest form of image segmentation is thresholding which is to convert an RGB or gray image to a bi-level image [30]. The Urdu and Persian images were RGB while English numbers were gray images. So we applied appropriate thresholding on all these three sets to get bi-level images having 2 levels for their pixels which is either 0 or 1.

Then we moved on to suppressing the noise by applying Gaussian filter. An appropriate Gaussian filter not only subdues the effect of noise but also maintains the sharp edges [31]. Filters with different sigma values were applied and the ideal sigma value was found to be 3 which was checked in accordance with thresholding.

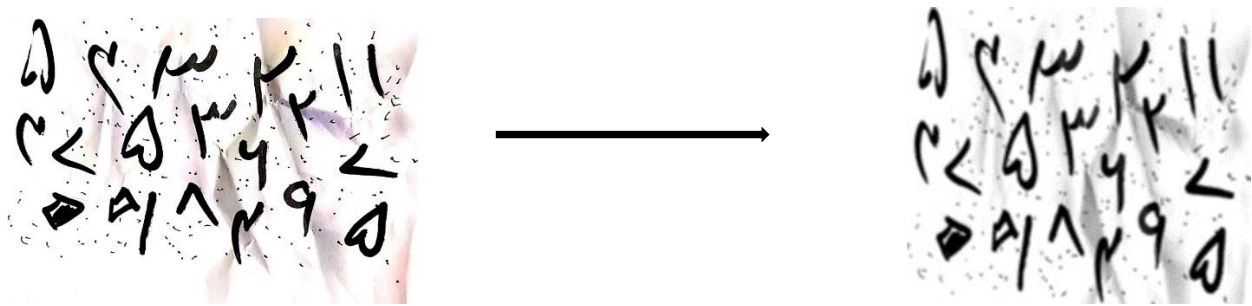


FIGURE 7: A SAMPLE OF CRUMBLER PAPER IS PASSED THROUGH GAUSSIAN FILTER

Finally, the Urdu images were cropped in a way so as to remove maximum background and obtain images like MNIST. Then all the three sets were resized to  $28 \times 28$  pixels and saved in their respective folders for ease of labelling.

## C. IMAGE AUGMENTATION:

One of the problems encountered during deep learning-based tasks is limited data. A huge amount of data plays a vital role in the sense that the model can learn the variations as well as similarities in it. This not only stops the model from overfitting but also gets a well-generalized model. These skillful models employ image data augmentation to artificially expand the size of the dataset by some modifications on the actual dataset. Augmentation is one of the popular techniques that includes a range of operations to be performed on the dataset like changes in image contrast, zoom, flip, rotation, brightness, and many more. The purpose is to expand the dataset with different varied examples of the original dataset but preserving the underlying characteristics and also retaining same labels post-augmentation. For example, in a classification problem where the model has to classify cats and dogs given some images of both the classes, a horizontal flip of picture of dog makes sense. This could mean that the photo was taken from the right. Another example could be of a zoomed in picture which would be able to show variations. But a vertical flip of the dog does not validate the experiment as it is very unlikely that the model is given an upside-down picture of dog.







In our dataset, we experimented with different types of augmentation techniques like changing:






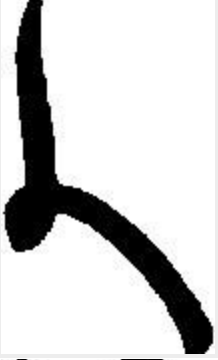


- brightness range
- rotation range
- ZCA whitening
- Zoom range
- Horizontal and vertical width shift range
- Horizontal and vertical flip
- Cropping
- Color space variations

A safe augmentation technique is the one where label is preserved post transformation. For instance, flips and rotations are safe for ImageNet dataset but not safe for numerals dataset, where an English '6' could be transformed into '9' upon such a transformation. So, considering this observation we carefully applied all these techniques. Intuitively if we apply horizontal flip on Urdu '2' or Persian '2' it becomes the language's '6'. After applying both the horizontal and vertical flips, we confirmed our intuition as it considerably decreased the model's accuracy. Same happened for the horizontal and vertical shifts. As we can instinctively say that shifting an image horizontally when the numeral is written on the edges, it would remove the information within the image. Like if Urdu '3' or Persian '3' is written on the right edge of image and we shift it horizontally, some of its pixels will drop and result in Persian or Urdu '2'. When we applied this augmentation technique on our data, the accuracy of our models decreased considerably. So, we decided that it is not safe to use such augmentation techniques for our data. The following table gives an insight of the techniques that we researched on and out of them which one were found to be precise for our dataset.



TABLE 3: AUGMENTATION TECHNIQUES FOR OUR DATASET

Augmentation Technique	Property of technique	Original Image	Effect of technique	Applied or not	Why?
Brightness	Increases or decreases brightness of image			No (range of [0.5, 1.0] to darken the image)	Pixels are already in black or white form so no need for more brightness
Rotation	Rotates image at a specified angle			Yes (applied for 25 degrees)	A slight rotation retains the shape of original image
Horizontal and vertical width shift	Shifts image horizontally or vertically			Yes (applied for 0.1 value)	A small shift range does not change the image Vertical shift did not work because it clipped off important part of number

<b>Zoom</b>	Zooms in or out of image			Yes (for range =0.1)	It zoomed a tiny amount bringing uniform change into image
<b>Horizontal flip</b>	Flips image in a horizontal fashion			No	It changed the label for image. The original image of Urdu '2' when flipped becomes Urdu '6'
<b>Vertical flip</b>	Flips image vertically			No	It changes the label for image as is evident in example.
<b>Cropping</b>	Crops a random part of image			No	It changes label of image when applied for a number like '3'

Color variation	Adds or removes color varieties to the image		No	It brings no effect as images are black and white
-----------------	--	--	----	---

Then we browsed into other techniques and implemented them individually on our data. Firstly, we added random brightness with a minimum range onto the image. This one help model to generalize well by getting images taken from different lighting levels. It specifies a min-max range for selecting a brightness amount. It has an ideal amount of 1.0 which has no effect on the image. Values less than 1.0 dim the lighting of image while those greater than 1.0 have a brighter effect on the image. So, we have to define a range from 1.0 to a value less than or more than 1.0 to be applied on the image. In our case, we applied range of [0.5, 1.0] to darken the image as our images were already relatively bright. Individually this worked well with our model, but when it was applied in combination with other augmentation techniques it gave very little accuracy, so we dropped it.

The zoom augmentation zooms into the image or zooms out of the image randomly. We have to specify a zoom range [1- value, 1+ value]. This works by adding and subtracting the value specified from 1 to make its range. For our dataset we applied, zoom of 0.1 which made a range of [0.9, 1.1] to be applied on the image. As we applied a tiny amount to be zoomed, this not only brought change in our original image but also zoomed uniformly so none of our information was lost.

Along with that we applied random rotation which ranges from 0 to 360 degrees in clockwise direction. It rotates the image in such a way that the pixels are rotated out of the image frame and leaves certain pixels without data. We checked for many values and finally picked out 25 degrees which yields the best results on numbers dataset. So, this again brings disparity in the dataset which helps the model learn distinct variations and helps it not to overfit. Lastly, we did random shifting of images but in horizontal fashion so as to retain maximum information. Again, we used a minimum value, so it does bring some change but also retains the original label. A value of 0.1 was tested which gave accurate results. All these three augmentation techniques were applied in combination to yield different variations of the dataset and get least overfitting.

## D. VISUALIZATION OF DATASET:

### 1. T-SNE:

A well-known method for visualizing datasets in high dimensions is t-Distributed Stochastic Neighbor Embedding (t-SNE). It works by appointing each datapoint from multiple classes a location in two or three-dimensional map. In simpler terms, it is suitable for high dimensional data that lies on several different but related low dimension manifolds like images belonging to multiple classes. It was presented by Hinton and Maaten in 2008 to reveal structure of dataset at different scales [32].

t-SNE's algorithm works in a nonlinear fashion by performing various transformations on different regions and adapting to the underlying data. It works on two types of parameters which need to be optimized – cost function parameters and optimization parameters. Perplexity which balances the attention between local and global aspects of data is the cost function parameter. Its typical value is between 5 and 50. Number of iterations, learning rate and momentum are the optimization parameters. It computes the similarity between datapoints in the low dimension space using the pairwise similarities function in equation 1:

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma^2)}{\sum_{k \neq i} \exp(-\|x_k - x_i\|^2 / 2\sigma^2)}, \quad (1)$$

But this similarity function causes issues when either of the points is an outlier. For such points the pairwise distance  $\|x_i - x_j\|^2$  becomes very large for  $x_i$  and the consequence is that  $p_{ij}$  becomes very small for the respective  $j$ . This makes the effect of outlier very little on the cost function which does not determine the position of that point as compared to the positions of other points. So, Hinton introduced the usage of joint probabilities  $p_{ij}$  in high dimensional space to cater this problem. It is set to be:

$$P_{ij} = \frac{P_{j|i} + P_{i|j}}{2n} \quad (2)$$

This makes sure that all the datapoints  $x_i$  make significant contribution to the cost function. Hence the conditional probability is given by equation 3:

$$P_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}, \quad (3)$$

For each iteration, it computes low dimensional affinities  $q_{ij}$  and gradient  $\delta C / \delta Y$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}. \quad (4)$$

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j) (1 + \|y_i - y_j\|^2)^{-1} \quad (5)$$

These parameters minimize the following cost function:

$$\text{set } \mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) (\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}) \quad (6)$$

So, t-SNE works by emphasizing on modeling the dissimilar points by means of their large pairwise distances while modeling the similar points by their small pairwise distances. These characteristics of t-SNE make it easier to optimize its cost function by using joint probabilities.

## 2. T-SNE VISUALIZATION FOR URDU DATASET:

To get a better understanding of our new Urdu dataset, we deemed it necessary to visualize it so we could get an understanding of our data. Also, before moving on to the implementation, we wanted to see whether our collected data had some underlying structure or not.

We start by computing Principal Components Analysis (PCA) of our dataset. This helped us reduce the dimensionality of the dataset which was set to 3 and also accelerates the process of calculating the pairwise similarities. For each class a color is set, which is used to determine the location of the data points belonging to each cluster. This color coding also helped us to evaluate the similarities within each cluster. It is plotted in the form of a scatter plot in two dimensions. For hyperparameter tuning, we used perplexity and number of iterations. Perplexity can have a value between 5 and 50. As for number of iterations, there is no such ideal value because different dataset can yield different results and also take different number of iterations for convergence. For our Urdu dataset, the ideal perplexity value turned out to be 30 with 1500 iterations.

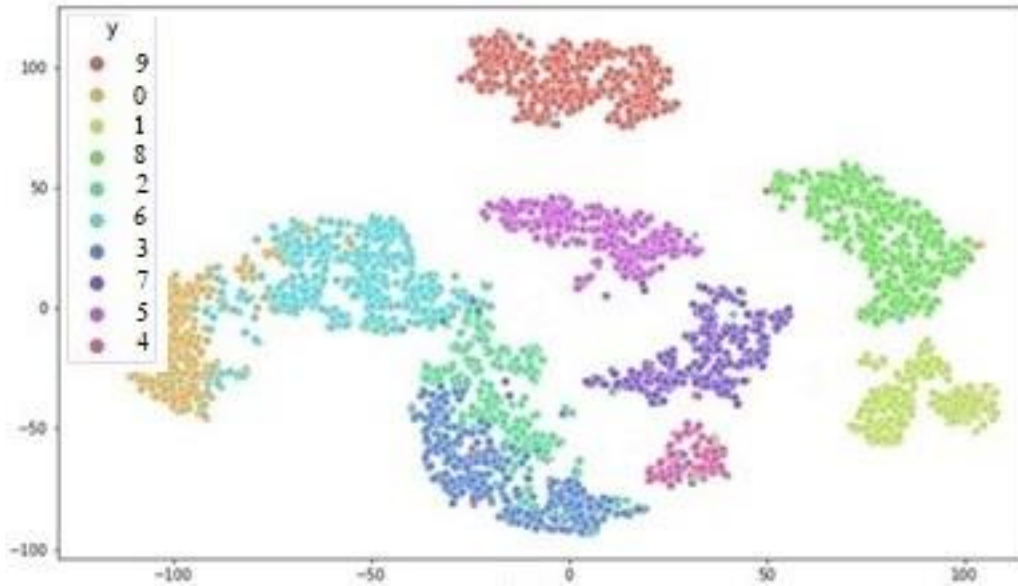


FIGURE 8: T-SNE VISUALIZATION OF URDU DATASET

As shown in the figure 7, datapoints are clustered into their respective clusters while some of them show overlap. Urdu '2' and Urdu '3' have almost similar shape so this can be seen on the bottom side that blue and green dots are overlapping. Similarly Urdu '0' and Urdu '5' have similar shapes and they have coincident points on the extreme left side in blue and brown shades.

### 3. T-SNE VISUALIZATION FOR ENGLISH DATASET:

Similarly in order to manipulate our datasets, we visualized them before combining them into one set. Figure 8 shows the t-SNE visualization for English numerals which are extracted from the popular MNIST dataset. The perplexity value of 25 with almost 1500 iterations yield the best results. The separate clusters show that each shape of English numeral is dissimilar which validates our thesis. A point to be noted here is that the distance between these clusters relate no information about the properties or similarities of points in the 2D space.

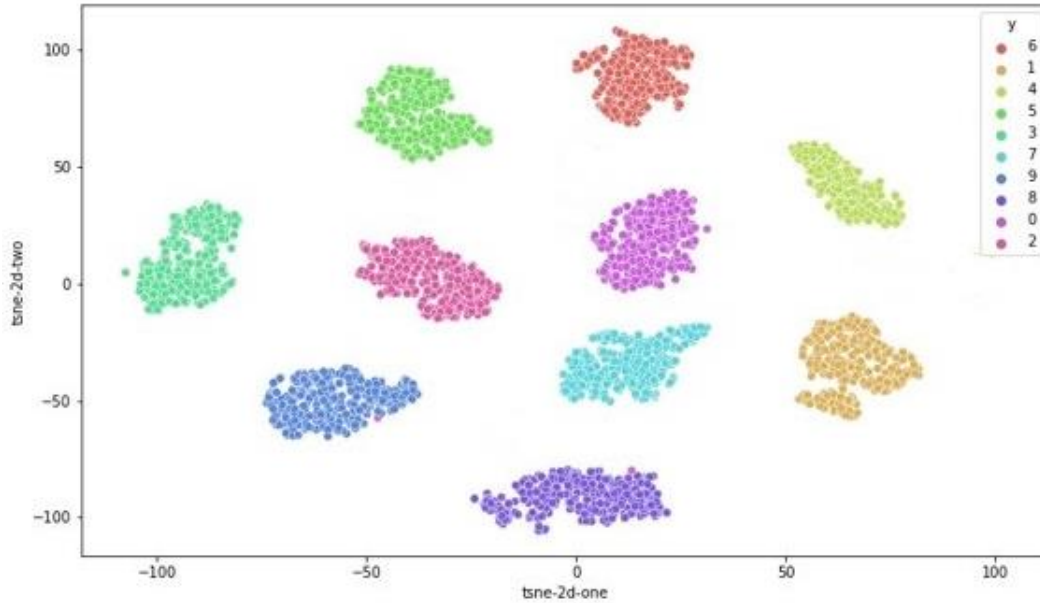


FIGURE 9: T-SNE VISUALIZATION OF ENGLISH DATASET

#### 4. T-SNE VISUALIZATION FOR PERSIAN DATASET:

For Persian handwritten numerals dataset, we extracted the images from the popular dataset by E. Kabir and H. Khosravi. The Persian script has similar numerals as Urdu with only '4', '6', '7', '8' differing in writing style. The ideal perplexity and iterations value was picked out to be 25 and 1500 respectively. The numbers '2' and '3' are written in almost same way so they are clustered in an overlapping fashion on the most left side in blue and green points. The rest of the digits are clustered in their respective circles.

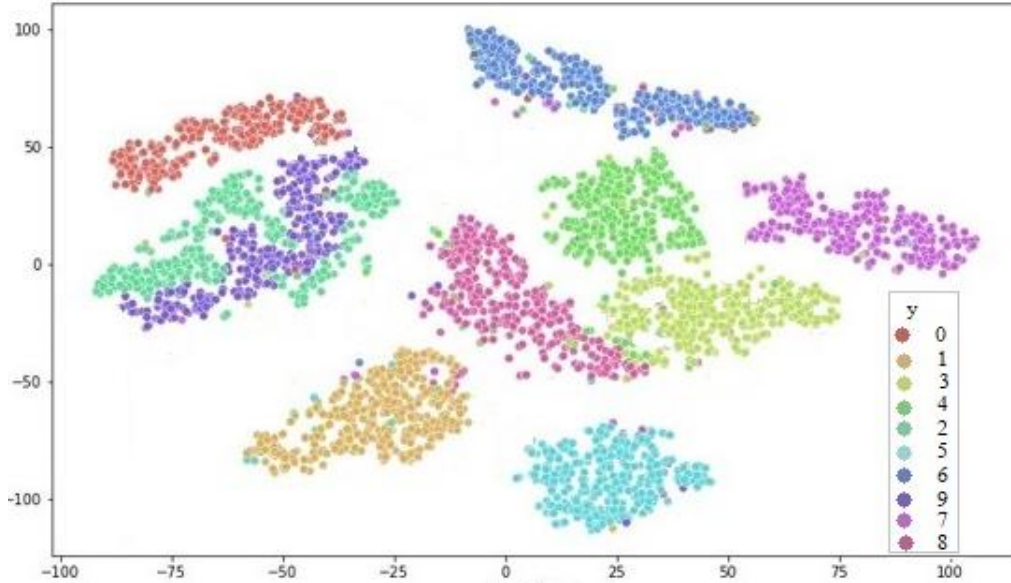


FIGURE 10: T-SNE VISUALIZATION OF PERSIAN DATASET

The overlapping points represent certain variations between the digits plotted in the graph. Yet again the distances or closeness between certain clusters gives no information about whether they contain coincident points or not.

#### 5. T-SNE VISUALIZATION FOR COMBINATION OF DATA:

Finally, after ample validation of our datasets we combined them into their respective folders based on their similarities or dissimilarities with each other as shown in table 2. In order to authenticate our experiments, we visualized these similar numerals. For instance, the numerals '1' and '9' are same for all the three datasets so in their t-SNE visualization they are clustered together as shown in figure 10 and 11. Similarly, '4', '6', '7', '8' are written in different patterns in all the three languages so the t-SNE plots for these numbers showed no overlapping points. This is evident in the figures 12, 13, 14, and 15. The rest of the digits have similarity in Urdu and Persian like Urdu '0' and Persian '0' are written in the same way while English '0' is written in another pattern. The Urdu '2', '3' and '5' have same shape as that of Persian letters so they are close to each other while distant from their respective English labels.



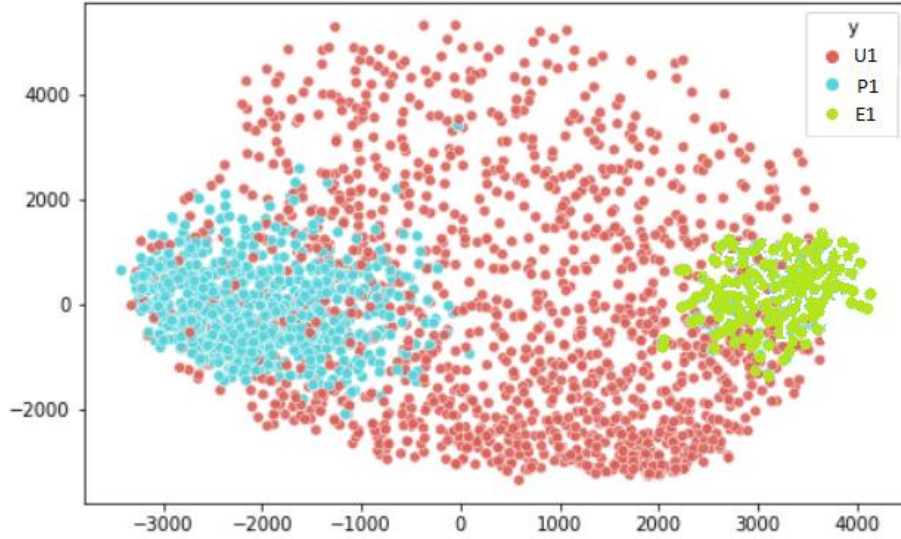


FIGURE 11: T-SNE VISUALIZATION FOR URDU ‘1’, ENGLISH ‘1’ AND PERSIAN ‘1’

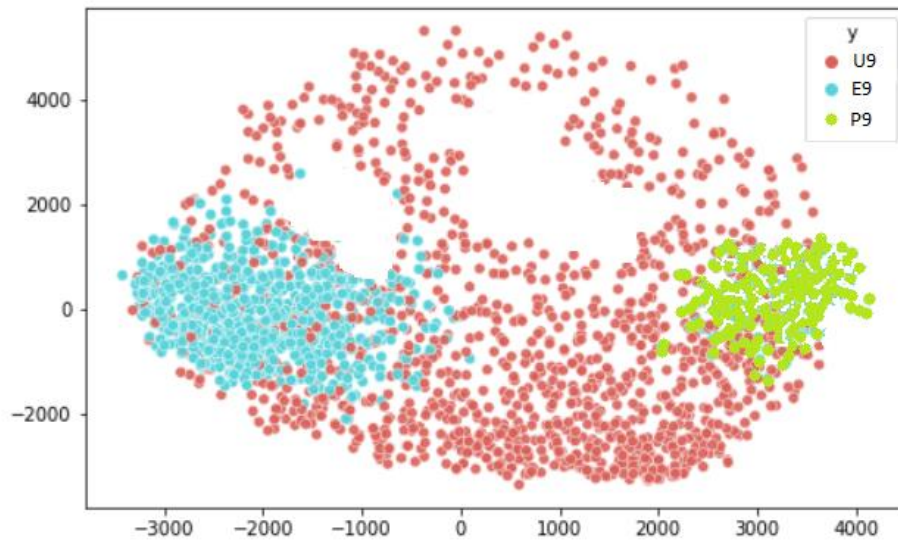


FIGURE 12: T-SNE VISUALIZATION FOR URDU ‘9’, ENGLISH ‘9’ AND PERSIAN ‘9’

As depicted in Figure 11 and Figure 12, all the datapoints overlap because of similarity in shape. For Figure 11, the green points represent English’s ‘1’ data, blue points show Persian’s ‘1’ datapoints while the red dots show Urdu’s ‘1’ points. The Urdu points are scattered all over the plane because the images taken for this experiment were 500 in number. Persian and English datapoints were almost 250 in number so their quantity is clearly shown by t-SNE. Besides that, we experimented with various perplexity values for this experiment. A very low perplexity value like 2 made the local variations supersede so it became hard to detect similarities in the plot. However, increasing the perplexity value to almost 500 showed no structure of the dataset. By

these experiments of perplexity value, we concluded that perplexity value should range between the datapoints in the plot but should not be necessarily very small.

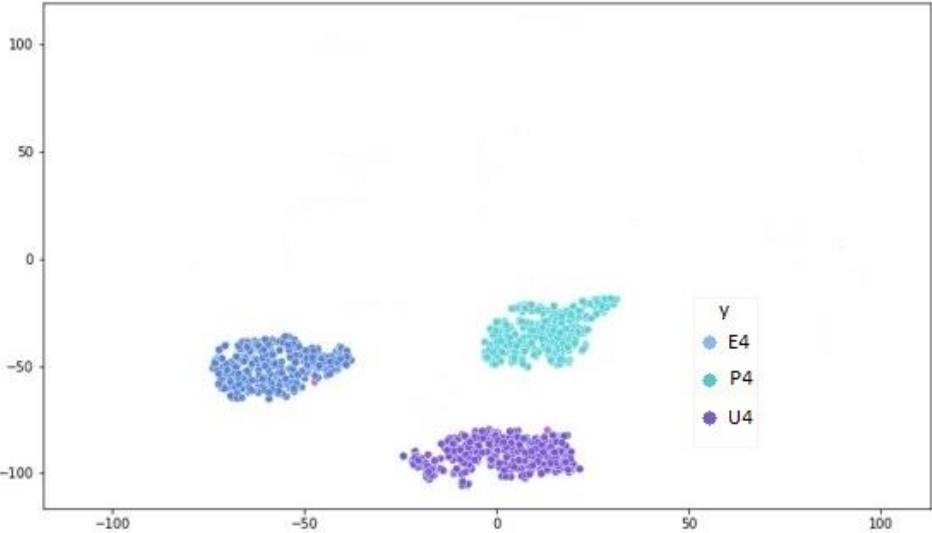


FIGURE 13: T-SNE VISUALIZATION FOR URDU ‘4’, ENGLISH ‘4’ AND PERSIAN ‘4’

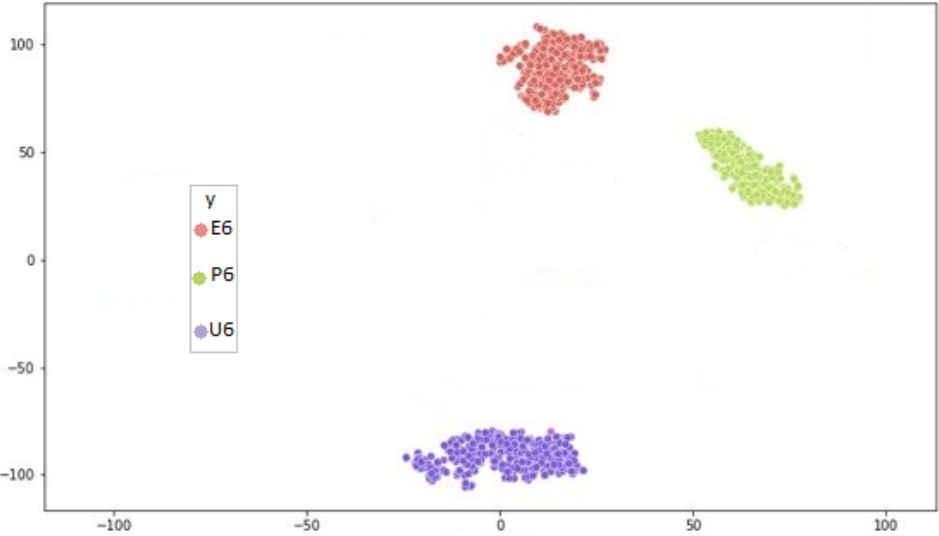


FIGURE 14: T-SNE VISUALIZATION FOR URDU ‘6’, ENGLISH ‘6’ AND PERSIAN ‘6’

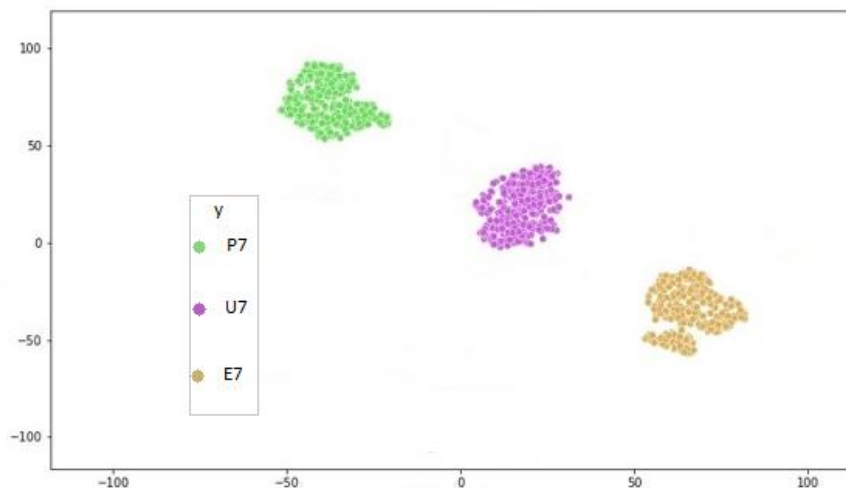


FIGURE 15: T-SNE VISUALIZATION FOR URDU ‘7’, ENGLISH ‘7’ AND PERSIAN ‘7’

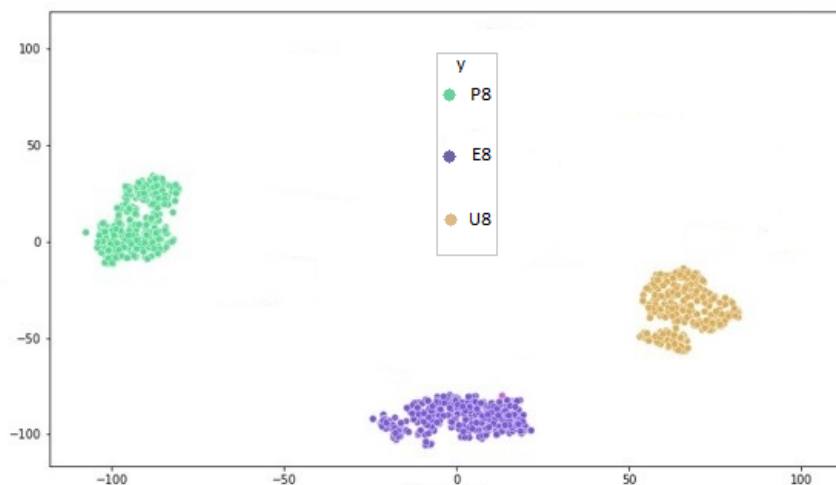


FIGURE 16: T-SNE VISUALIZATION FOR URDU ‘8’, ENGLISH ‘8’ AND PERSIAN ‘8’

The figures above show the datapoints namely ‘4’, ‘6’, ‘7’ and ‘8’ in t-SNE plane visualized according to their respective clusters. The size of datapoints in each folder was kept 50 so as to get clear segmented clusters. Intuitively it can be deduced that the digit ‘4’ for each of the three languages should belong to separate clusters because its writing style is unrelated for each language. Our intuition makes sense when we tune the perplexity value to 20 for each experiment. Figure 13 shows various shades of blue to denote the number ‘4’ belonging to separate classes. Figure 14 uses red, green, and purple color palette to depict the different clusters of number ‘6’. For numeral ‘7’ the datapoints are aligned in separate clusters according to their respective labels in Figure 15. Lastly, Figure 16 exhibits the clusters of number ‘8’ in non-identical clusters. It should be noted here that we kept the perplexity value neither too small nor very large (approaching the number of datapoints) in order to get perfect non-overlapping clusters.

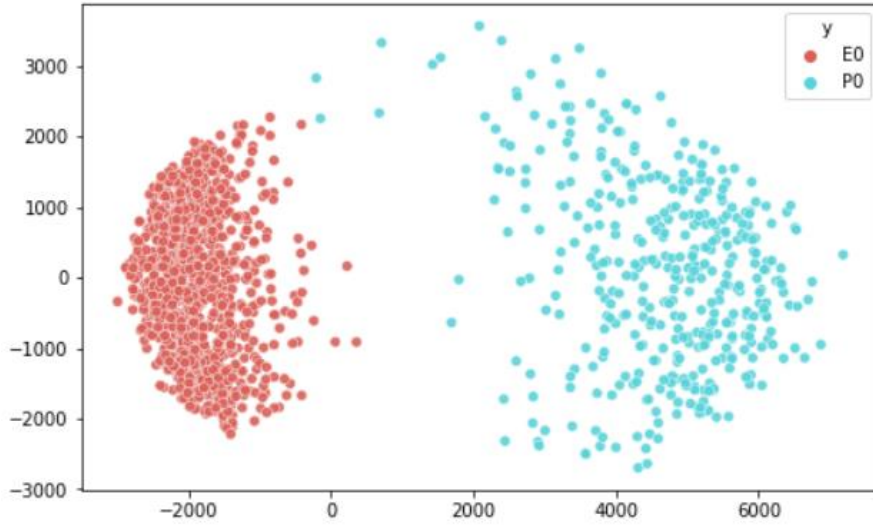


FIGURE 17: T-SNE VISUALIZATION FOR ENGLISH ‘0’ AND PERSIAN ‘0’

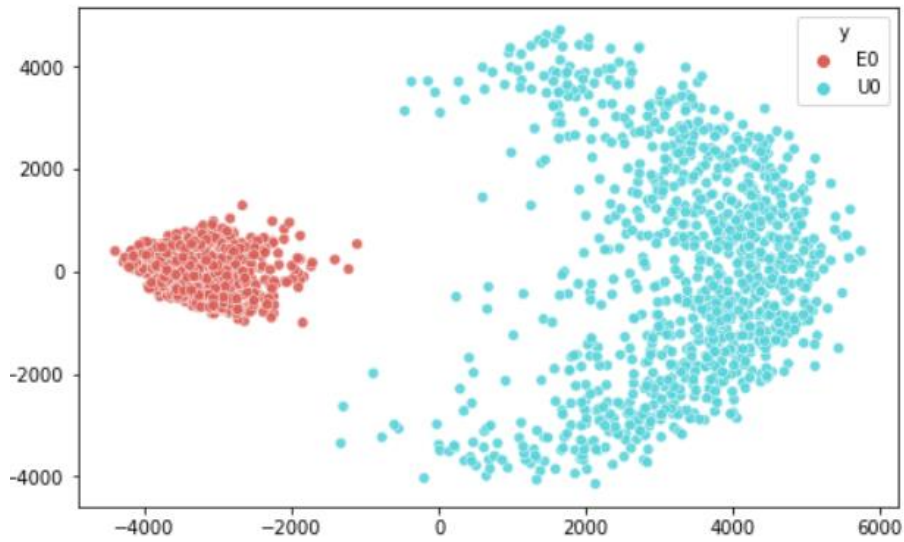


FIGURE 18: T-SNE VISUALIZATION FOR ENGLISH ‘0’ AND URDU ‘0’

We move onto those digits next which have similarities between two classes, so we get 3 types of plots for each of them. Figure 17 shows the distinct points for English ‘0’ and Persian ‘0’. While Figure 18 displays the separation of clusters for Urdu digit ‘0 and English digit ‘0’. This being said we have Figure 19 to display the similarity of Urdu ‘0’ and Persian ‘0’ owing to their similarity in writing style. Keeping in mind these differentiations, we assign the Persian ‘0’ and Urdu ‘0’ a same label. Similarly, English ‘0’ is assigned a separate label because of its dissimilarity between its counter Urdu and Persian numbers.

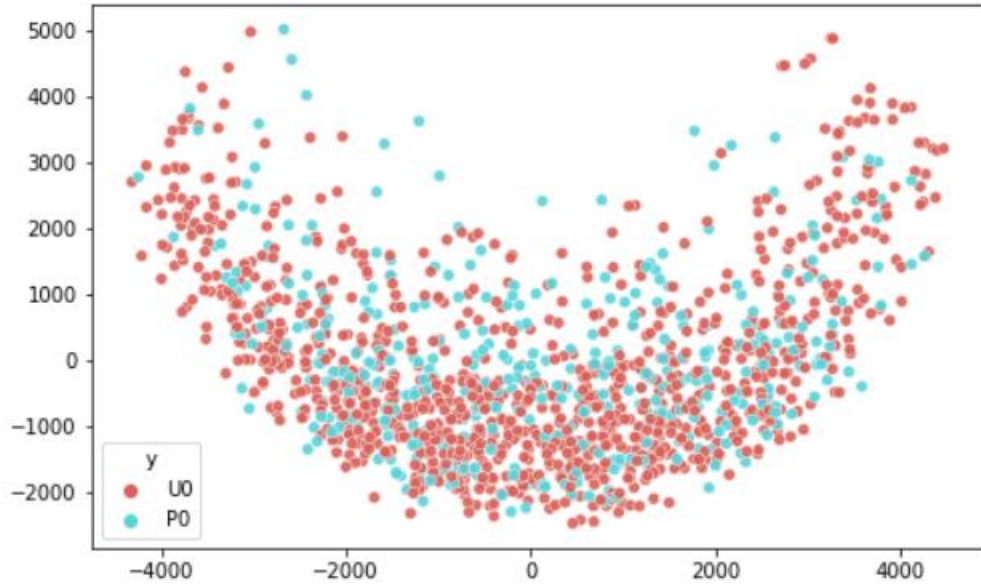


FIGURE 19: T-SNE VISUALIZATION FOR URDU '0' AND PERSIAN '0'

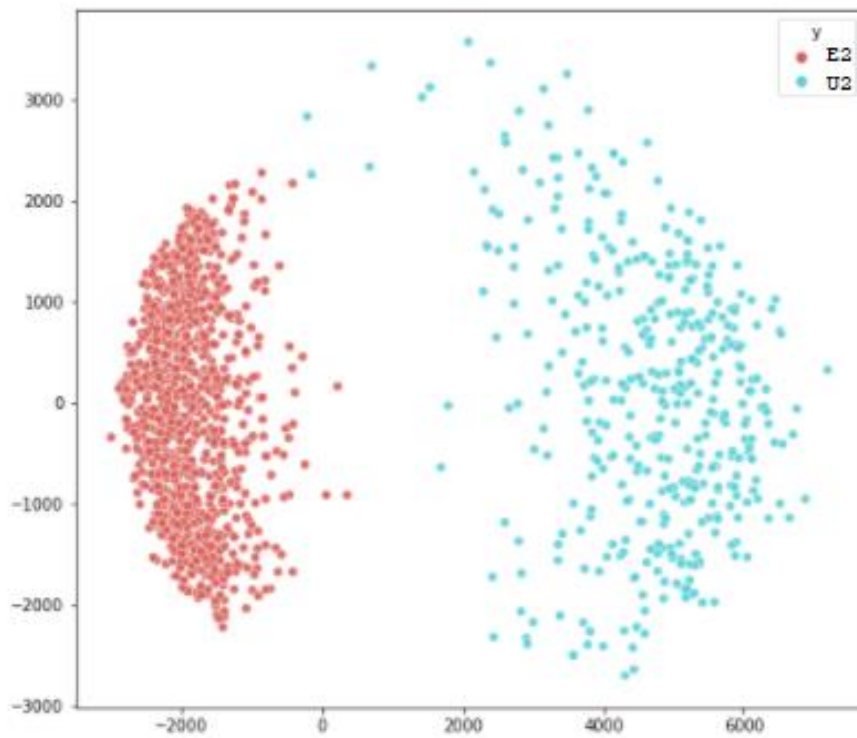


FIGURE 20: T-SNE VISUALIZATION FOR ENGLISH '2' AND URDU '2'

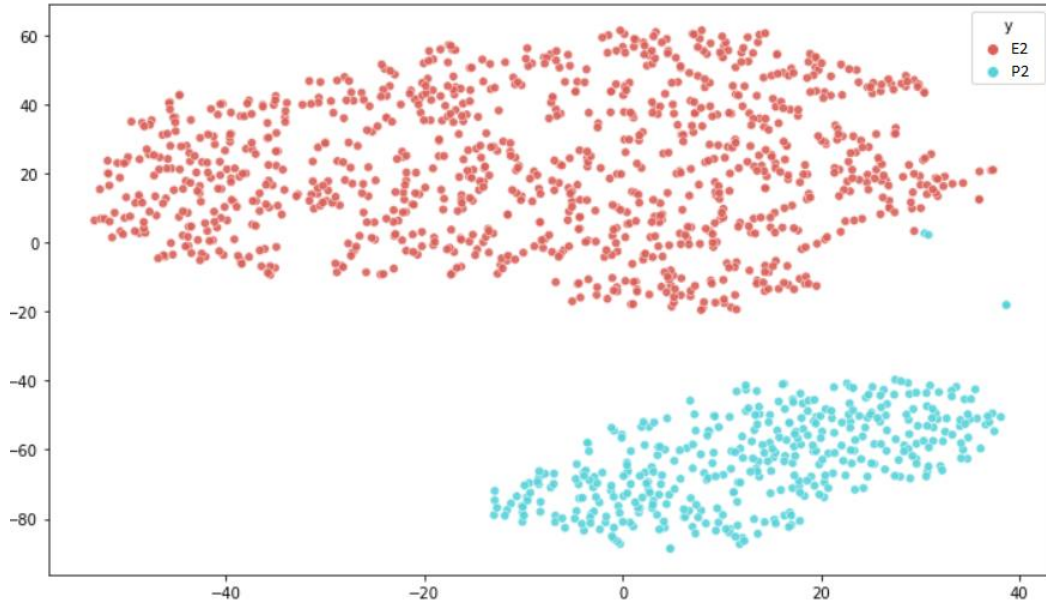


FIGURE 21: T-SNE VISUALIZATION FOR ENGLISH ‘2’ AND PERSIAN ‘2’

Here again we have similar patterns of t-SNE for the digit ‘2’ in Figure 20 and Figure 21. The numerals for Urdu and Persian labelled as ‘2’ have identical shape. It only differs when a person writes with a slant or when the image is noisy. This disruption is depicted in the form of outliers in plot; wherever the points are out of range or coinciding with the other cluster it is because of such problems. This not only gives us variations of a cluster but also validates a part of our proposed thesis that we have used a variable dataset.

A perfect overlap of points is visible in Figure 22 which plots the Persian digit ‘2’ and Urdu digit ‘2’. Instinctively this authenticates our claim that such digits should belong to a same class. Not only does this experiment reduce the number of classes to be learned but also lessens the parameters, weights and time associated in training of its networks.



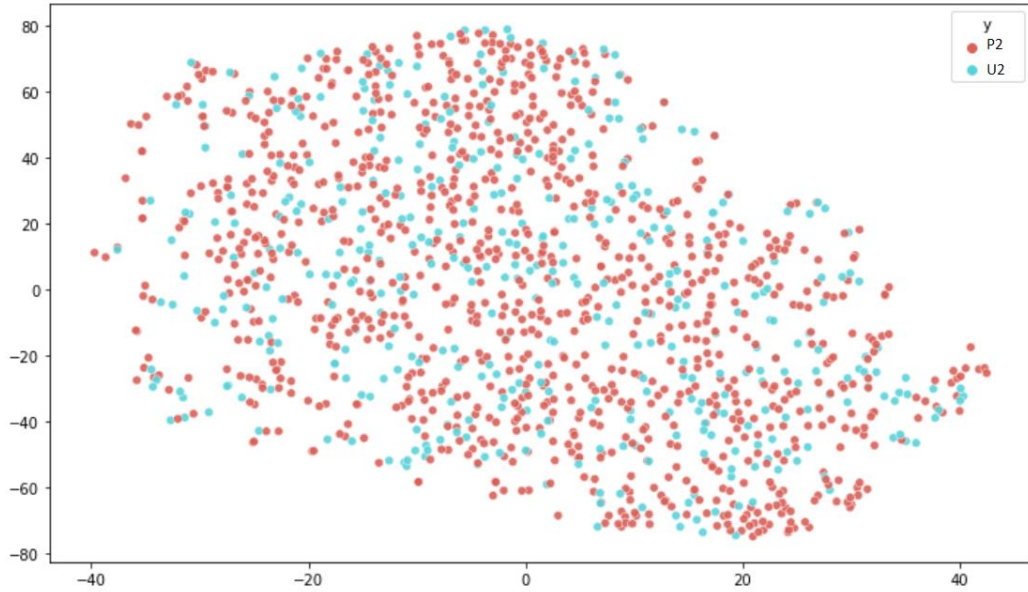


FIGURE 22: T-SNE VISUALIZATION FOR URDU '2' AND PERSIAN '2'

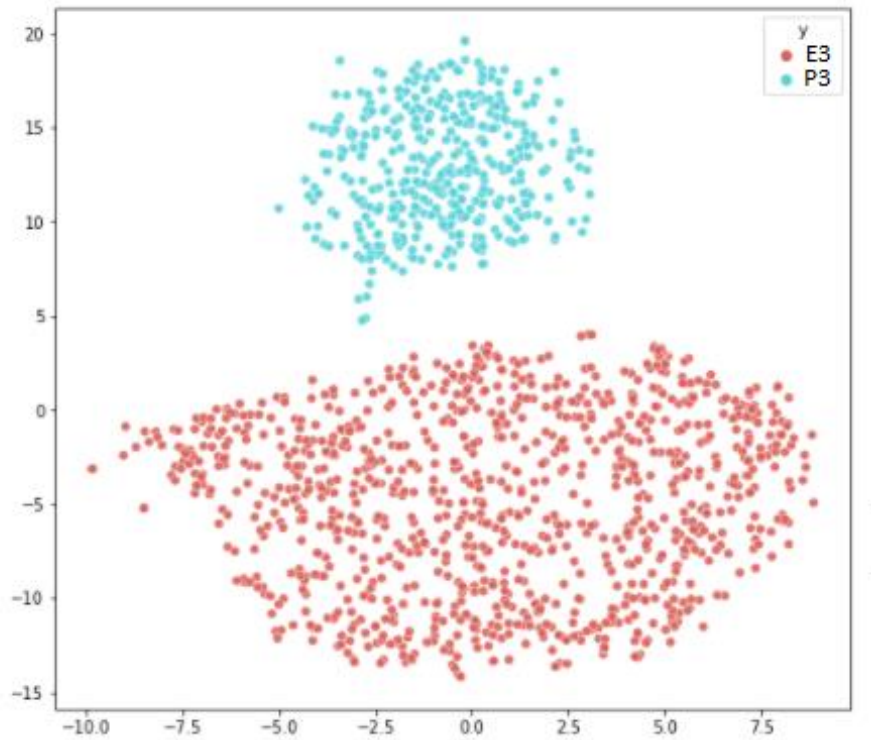


FIGURE 23: T-SNE VISUALIZATION FOR ENGLISH '3' AND PERSIAN '3'

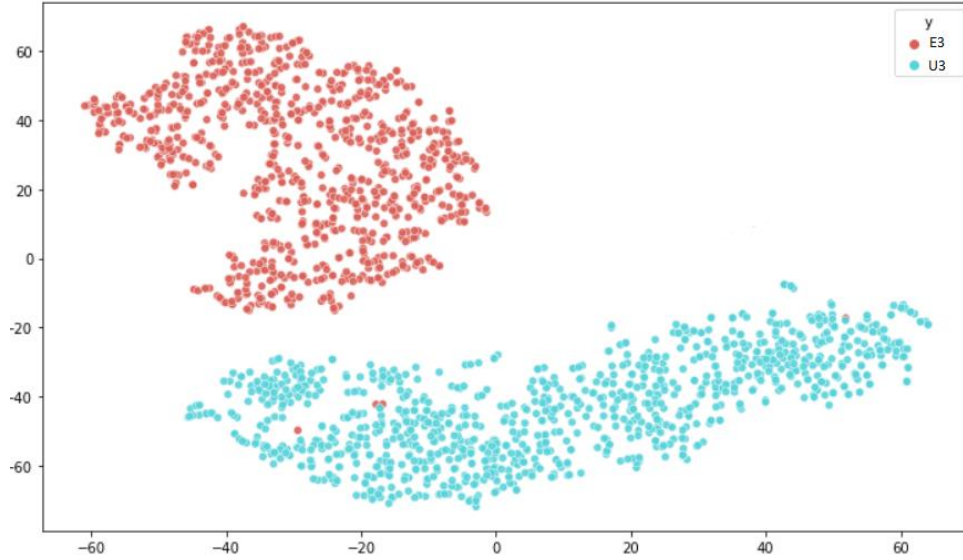


FIGURE 24: T-SNE VISUALIZATION FOR URDU ‘3’ AND ENGLISH ‘3’

In Figure 23 and Figure 24, t-SNE illustrates the differences between digit ‘3’ for English and Persian classes, and English, Urdu class respectively. For this plot we experimented with number of iterations as the hyperparameter. We started with 25 epochs which gave us pinched clusters in the plot, so we increased the epochs to a random large value 250. But these epochs did not give a stable result too. So, we came back to the original number of epochs and gradually started increasing its number. At epoch 70 we reached a relatively stable configuration hence we deduced the conclusion that a particular range of epochs does not work every time. Hence it is much needed to fluctuate the number or iterations.



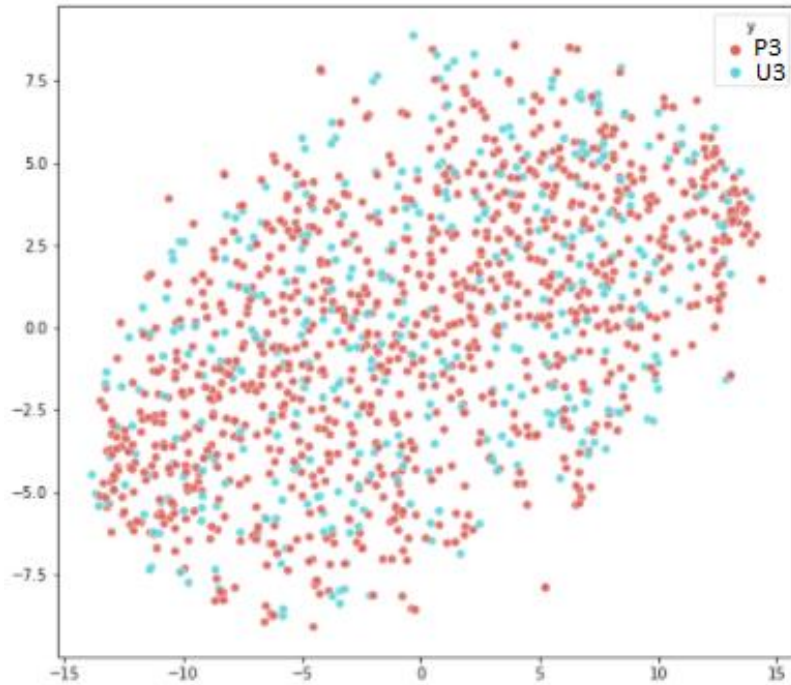


FIGURE 25: T-SNE VISUALIZATION FOR URDU '3' AND PERSIAN '3'

Figure 25 shows a remarkable curve for Urdu and Persian digit '3'. This almost perfect curve is just because of the similarity in shape of both the classes. The blue dots denote '3' for Urdu language while the red one shows it for Persian. This again attests the truth of our intuition that numbers belonging to Urdu and Persian for digit '3' should be labelled in the same class. Hence based on these results we identify same label for Urdu and Persian '3' but different label for English '3'.

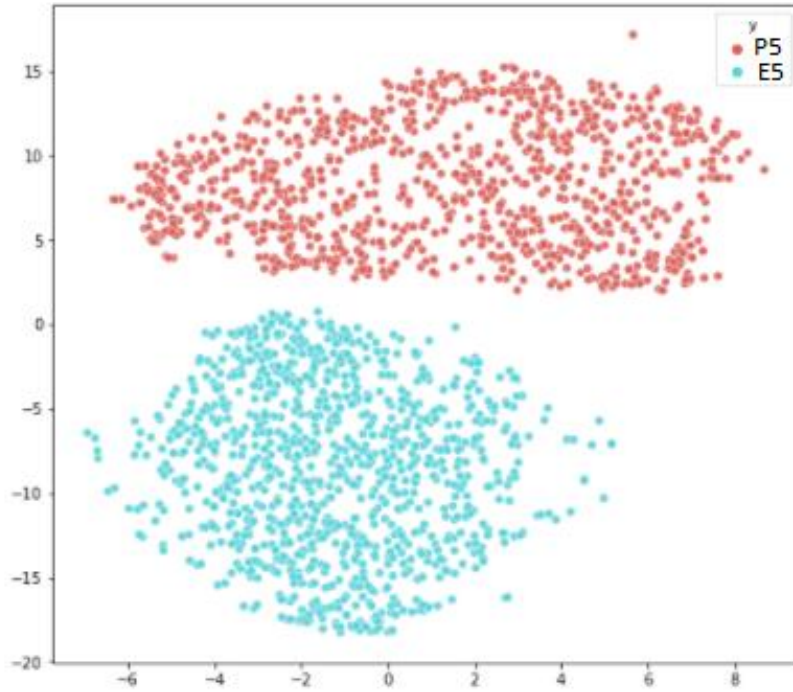


FIGURE 26: T-SNE VISUALIZATION FOR ENGLISH ‘5’ AND PERSIAN ‘5’

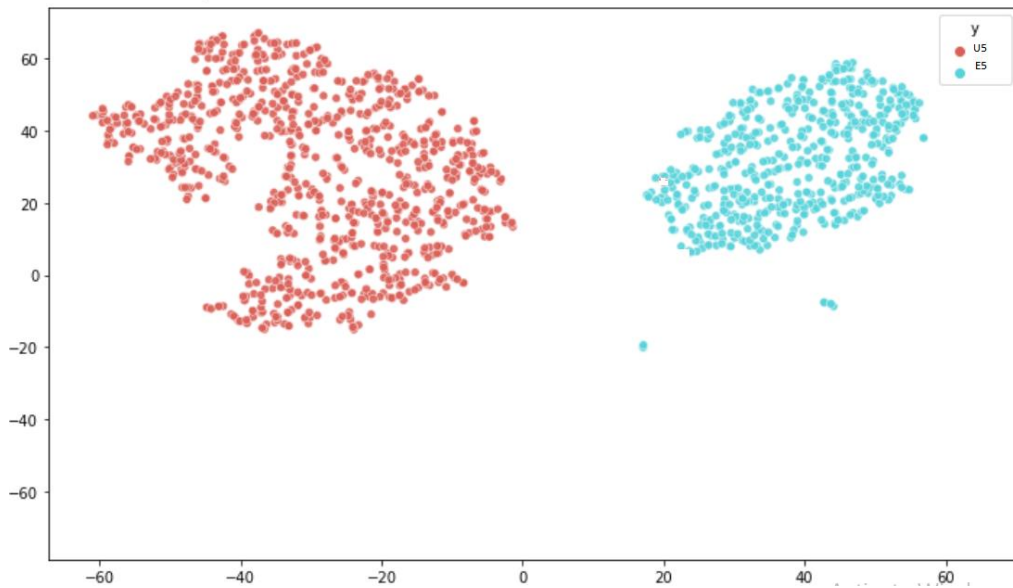


FIGURE 27: T-SNE VISUALIZATION FOR URDU ‘5’ AND ENGLISH ‘5’

Digits labelled as Urdu ‘5’ and Persian ‘5’ have similar shape as that of a vertically flipped heart. An interesting phenomenon that we encountered while playing with different values of hyperparameters was that each run gave us a different result. After studying its underlying paper,

we came to the conclusion that t-SNE has an objective function that is non-convex. Its objective function uses random weights upon initialization; hence it was okay for different hyperparameter values to display variable results.

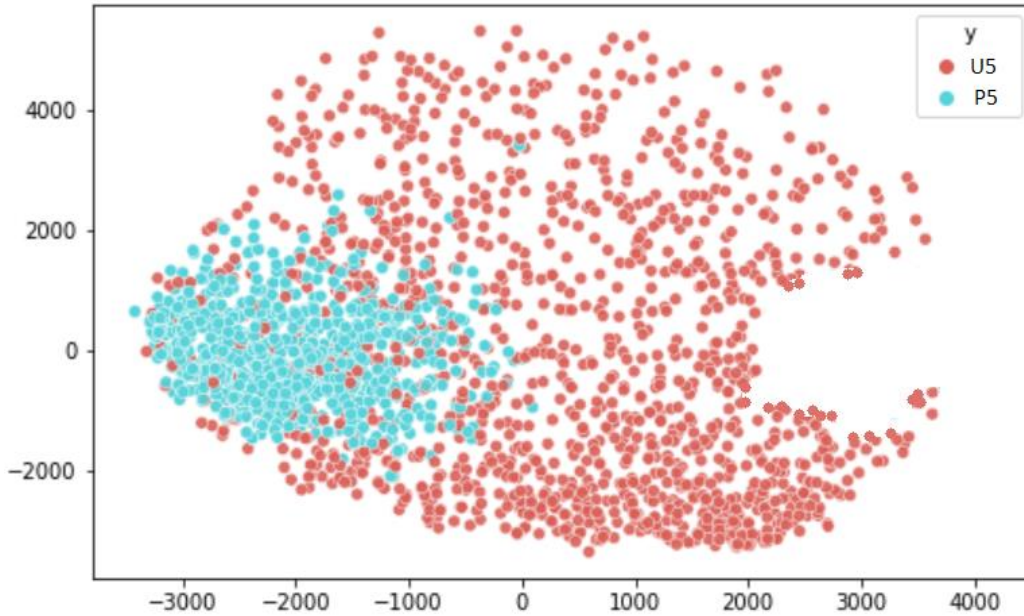


FIGURE 28: T-SNE VISUALIZATION FOR URDU ‘5’ AND PERSIAN ‘5’

Figure 26, Figure 27, and Figure 28 display the results for digit ‘5’ for all the classes. The distinctions between points are clearly visible for Urdu and English, Persian and English. An important point to note here is that since all the images are different so the points belonging to same classes are also non-overlapping. So, when points overlap in Figure 28 that does not necessarily mean that only some of them match particular images. It is actually because we have incorporated so many variations during pre-processing step that almost each image is different from the other.

Our illustrations of t-SNE results not only authenticates our intuition but also builds up the foundation for authenticity of our models. The high accuracies that we achieve further down the chapters despite so much pre-processing is because our models are able to capture the underlying distribution of the dataset.

## E. PROPOSED MODEL

Convolutional neural network is the progressive neural network that has a huge accuracy in learning features of visual data. In a CNN, the input is transformed to get accurate predictions by traversing through some layers. It consists of 4 core sub-structures which are used repeatedly with different activation functions to deduce best results.

- The input layer contains raw pixel values and in this case each image of size  $28 \times 28 \times 3$  pixels is fed to the CNN. Here 28 represents the width and height of image while 3 is the color channels- red, green, and blue.
- Convolution layer connects local receptive field of the input with neurons in next layer. This is done by a simple dot product of kernel and input image. Kernel size of  $3 \times 3$  is maintained throughout the model whereas padding is set to 1.
- The pooling layer downsamples input along spatial dimensions. One of the most famous pooling layers is 'Max Pooling' which is used here to extract highest pixel value in current space. These extracted features are then fed to the classifiers which are discussed further.

Our proposed model classifies 22 classes of handwritten Urdu, Persian and English numerals using convolutional neural networks with feature mapped output layer. We use custom CNNs along with popular CNN architectures for our dataset to get a comparison of the various classification methods that can be applied on our dataset. The details of the different models that we applied on our dataset are explained in the following sections. Figure 29 explains the methodology that has been followed in this thesis.

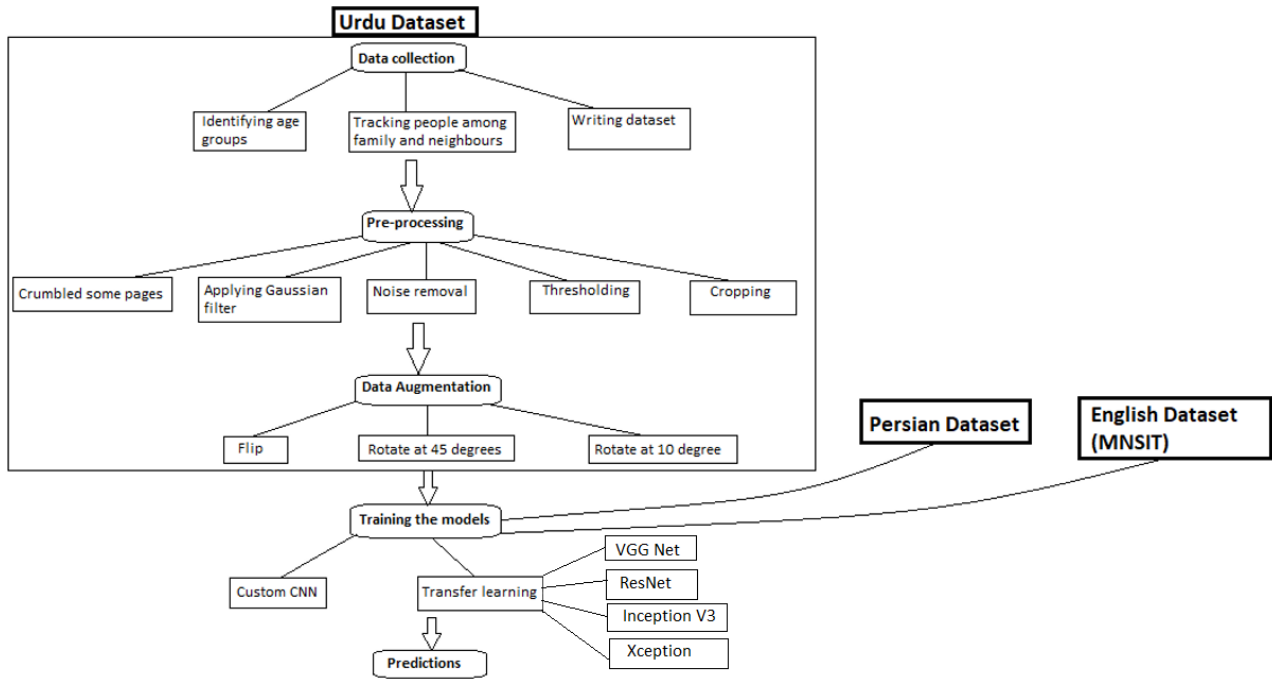


FIGURE 29: METHODOLOGY OF THESIS

## 1. CNN V1:

We started by making a simple model for training so that we could evaluate the quality of our dataset and then move on to better models. For our first model, we combined the primitive layers - convolution layers, activation function and pooling layer. Convolution layer works by connecting the pixels in input with neurons in the next layer after taking dot product of kernel and input. Kernel of size  $5 \times 5$  is maintained throughout the model whereas padding is set to 1. Padding ensures that image is not shrunk by adding zero value pixels along the border of input [33]. It is followed by activation function where each output from neurons of previous layer are fed to the RELU activation function. We chose RELU activation function as it works better than other functions like sigmoid in terms of vanishing gradient problem [34]. It was picked out of other non-linearities after comparing their results in our CNN model. Lastly pooling layer was applied to get reduced number of parameters in the end and to reduce overfitting [35]. Three sets of these layers were applied to get summarized results which were fed into the fully connected layers using the Softmax classifier. We used the Softmax classifier as it works best for classification problems. During compilation of model, we used stochastic gradient descent (SGD) function for optimization. It achieves a reasonable gradient at a low convergence rate at a minimum cost, so it is mostly preferred in huge datasets. A summary of this model is as follows:

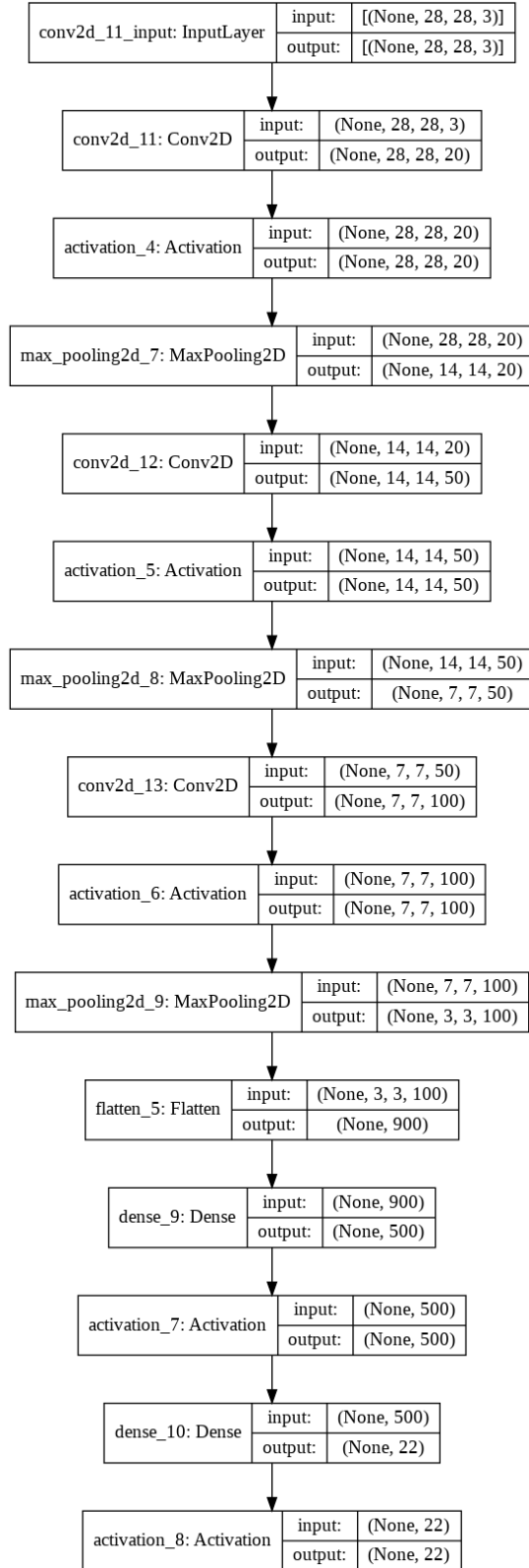


FIGURE 30: CNN V1 MODEL

## 2. CNN V2:

We improved the model by looking at ideas that could help us tune our model in a better way. We started by tuning the learning rate for SGD. First, we experimented with very large values and moved onto smaller values. This gave us an almost ideal value of 0.001 as compared to the previous value of 0.1. Then we dug into literature and found Rumelhart, William and Hinton's paper of backpropagation learning which introduced the concept of momentum in combination with SGD [36]. With momentum the gradient of loss is accelerated as compared to the classical SGD which made the gradient travel in the same direction and thus preventing any kind of oscillations. Along with momentum we found Nesterov accelerated momentum which in combination with momentum for SGD gives better and faster results. Instead of evaluating gradient at the current position, Nesterov momentum keeps a check on the lookahead gradient step. Thus, it helps the model adapt to the update of error function and speed up SGD in turn [37].

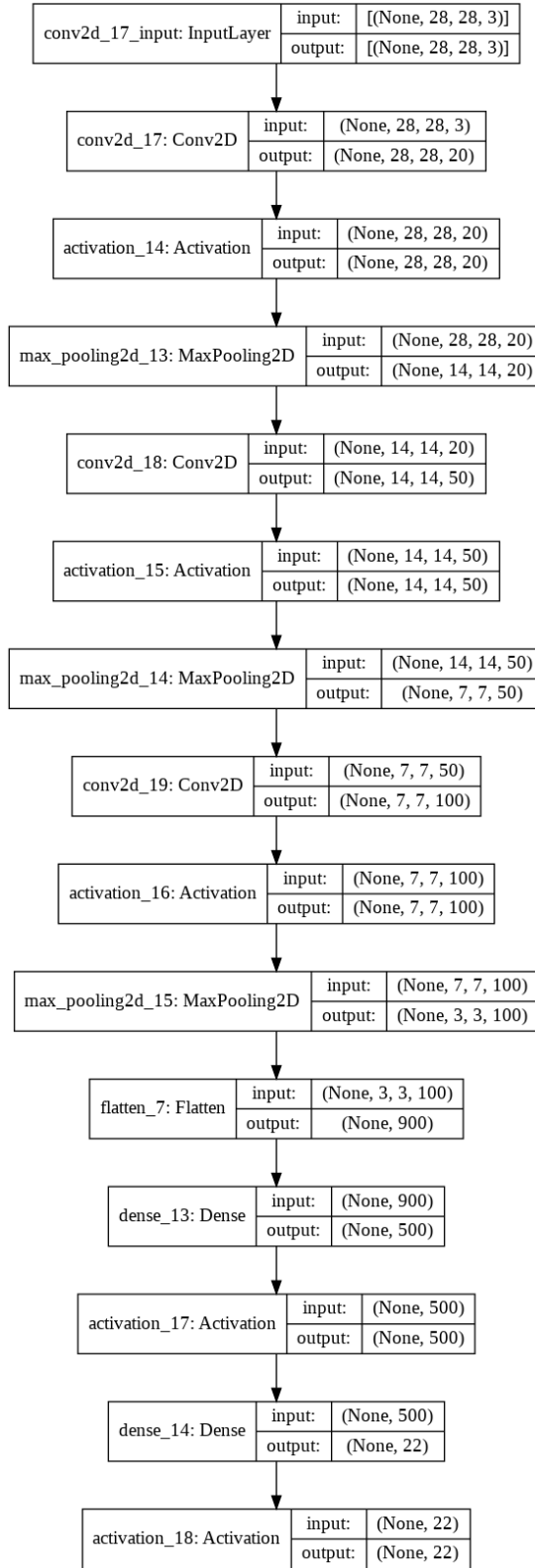


FIGURE 31: CNN V2 MODEL



### 3. CNN V3:

In order to achieve the best results for our dataset, we moved onto advanced optimization algorithms and retuned our hyperparameters. Adam [38] is one of the latest optimization algorithms and takes very little time to converge and promises best results. It works by computing adaptive learning rates for each parameter by keeping an exponentially decaying mean of past calculated gradients [39]. It is also used as an alternative for SGD+ Nesterov as that requires intense tuning of hyperparameters. We used the default value of learning rate = 0.001 for Adam. Another technique to improve a model's performance is batch normalization. It simply standardizes the layers for each batch which stabilizes the learning process in a few epochs. So, our model was modified by keeping in view these additions and it a glimpse of it is as follows:

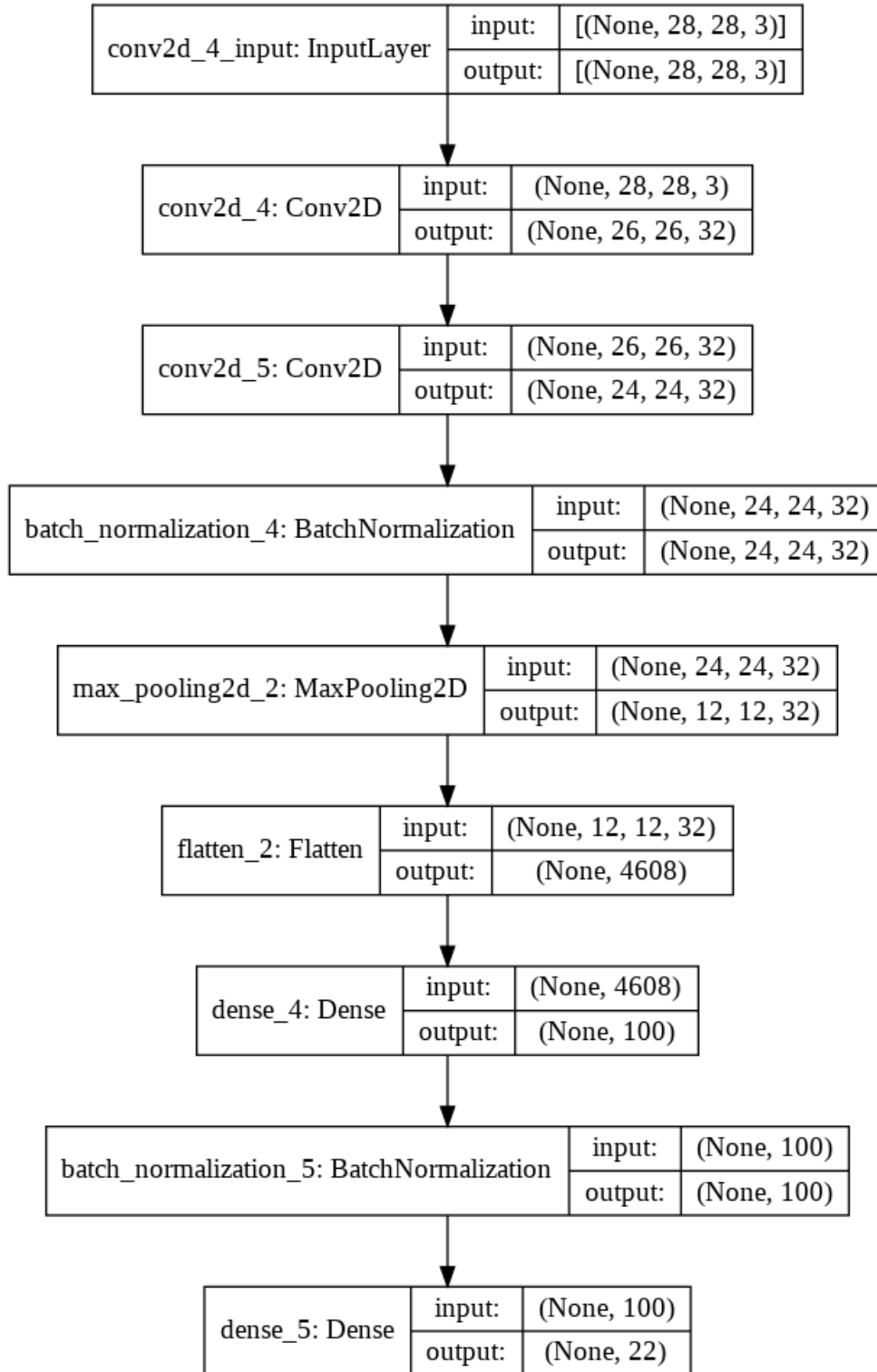


FIGURE 32: CNN V3 MODEL

After considerable research, we came across another method to solve overfitting and get quick convergence. Previously we were using 150 to 200 epochs to get convergence which in turn lead

to overfitting. So, by we employed early stopping which needs an arbitrary number of epochs to be set but it stops training when the model’s performance reaches its best on validation data i.e. it stops improving. Its parameters monitor, mode, and patience need tuning to get accurate results. For the training to end, a parameter has to reach some convergence. The parameter whose convergence we need to acquire is specified in ‘monitor’. ‘Mode’ defines the objective of selected metric to be achieved whether its minimum or maximum value is required. To cater the delay to the trigger for epochs on which we would like to see no improvement we use the ‘patience’ argument. Its accurate value varies between the ideal model for different datasets. We monitored the minimum validation loss for a patience value of 5 to get precisely accurate results.

#### 4. VGGNET

Deep learning has become a key instrument in artificial intelligence applications [40]. Research fields like natural language processing, computer vision and speech recognition have produced remarkable results in deep learning. This growing interest has given birth to innovations in this field. Transfer learning is one such aspect where pre-trained deep learning architectures that have won the ImageNet competition are implemented on relatively contrasting datasets. We present the implementation of 4 models on our dataset. For all these 4 implementations, we keep some redundant settings same. The last layers are freezed and replaced with global average pooling fed into the RELU activation map. Also, the image size is kept as 75 x75 x3 for all the architectures. We only feed the augmented images into the models while using early stopping.

TABLE 4: COMPARISON OF SETTINGS MODIFIED FOR EACH ARCHITECTURE

Model	Depth	Top-5 accuracy	Settings added
VGGNet	16	0.901	<ul style="list-style-type: none"> <li>• Global average pooling,</li> <li>• ReLU activation map,</li> <li>• Dense neurons,</li> <li>• Fully connected layer of 22 neurons</li> </ul>
ResNet50	50	0.933	<ul style="list-style-type: none"> <li>• Global average pooling,</li> <li>• ReLU activation map,</li> <li>• Fully connected layer of 22 neurons with Softmax activation</li> </ul>
GoogLeNet (Inception V3)	48	0.941	<ul style="list-style-type: none"> <li>• 2D Global average pooling,</li> <li>• ReLU activation map,</li> <li>• Fully connected layer of 22 neurons with Softmax activation</li> </ul>
Xception	36	0.945	<ul style="list-style-type: none"> <li>• 2D Global average pooling,</li> <li>• ReLU activation map,</li> <li>• Fully connected layer of 22 neurons</li> </ul>

VGGNet was the runner-up for ILSVRC 2014 and was presented by Karen Simonyan and Andrew Zisserman [41]. It was one of the first deep networks to be presented as it had 16-18 layers. This increased depth was proven to be a critical component to achieve good performance. It works on the phenomenon of smaller filters and deeper networks. The benefit of these smaller filters is that they have same receptive field as a larger filter but with lesser parameters. They replaced a 7x7 filter with a stack of three 3x3 filters which yields quick computations with less parameters. Hence less storage and time was required for it despite it being a deep network. The final version features a homogeneous architecture that only performed 3 x3 convolutions and 2 x2 pooling from the very beginning till the end. With the correct selection of hyperparameters we achieved best accuracy in almost 45 epochs with validation accuracy greater than 95%.

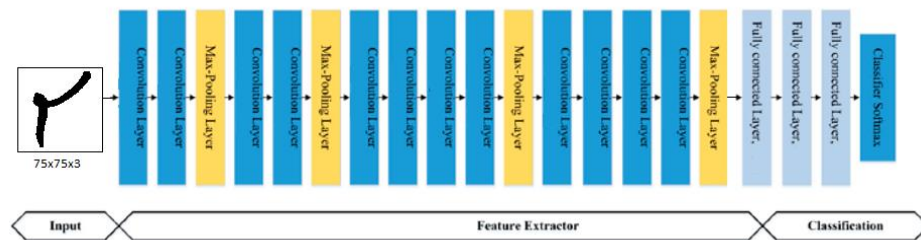


FIGURE 33: VGGNET MODEL

## 5. GOOGLENET (INCEPTION V3)

GoogLeNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014 [42]. It is based on the idea of inception layer which covers a large area but maintains fine resolution on dataset for small information. The convolution and projection layers use ReLU as activation function while working on receptive field of 224x224 with 3 color channels. It is 27 layers deep if we count the layers with parameters too. It includes convolution layers, max pooling layers, fully connected layers, and finally linear layers. Because GoogLeNet achieved top 5 error-rate of 6.67% so we used it to train numeral dataset.

We used its version 3 for our task as this version had the latest improvements in the model. Along with the basic features of GoogLeNet it adds on label smoothing, factorized 7 x 7 convolutions and auxiliary classifier to deliver label information down the model [43]. A major task was to tune parameters mainly number of epochs, batch size, and learning rate. But as we used Adam optimizer and early stopping, they solved the problem of learning rate and number of epochs. For batch size we trained on many values and found 128 to be the appropriate size. It achieved accuracy greater than 90 in almost 12 epochs when trained with the proper parameters.

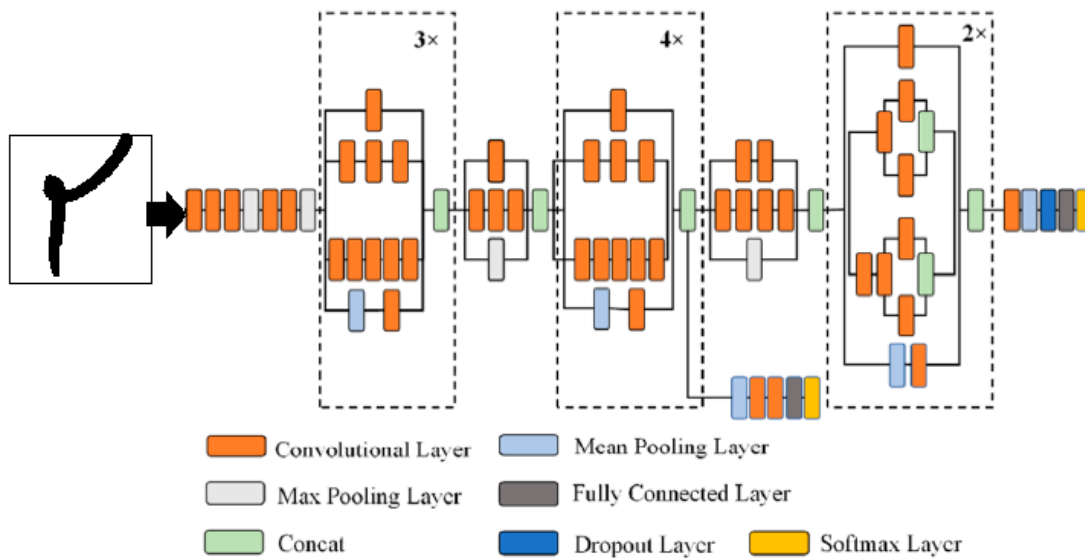


FIGURE 34: GOOGLNET (INCEPTION V3) MODEL

## 6. RESNET

Like GoogLeNet, ResNet was the winner of ILSRVC 2015 with an error rate of 3.6%. It is based on very deep neural networks using residual connections. A residual connection (also called skip connection) copies the learned layers from shallow model and sets additional layers to identity mapping [44]. Instead of making a network deeper and deeper which causes problems like vanishing gradient and severe overfitting on both training and validation data, it uses layers to fit a residual map. The idea was to stack these residual blocks to make a deep network instead of simply stacking layers.

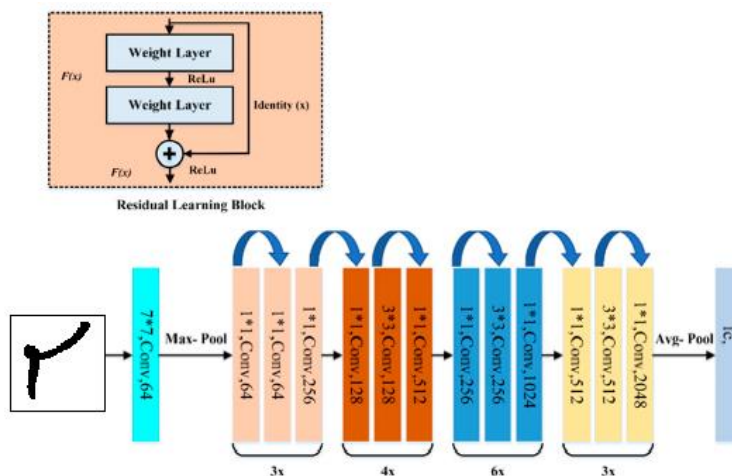


FIGURE 35: RESNET50 MODEL

For our dataset, we kept images of size  $75 \times 75 \times 3$  and used the ResNet50 version which has 50 layers in its network. We used the Adam optimizer to yield best results with a batch size of 128 and almost 47 epochs.

## 7. XCEPTION

Xception is based on the idea that inception modules in CNN are an intermediate step in normal convolutions and depth wise separable convolution [45]. Figure 36 gives a complete description of the specifications of the network where data enters through the entry flow followed by middle flow and then out of the exit flow. As we are working on classification task, so the convolutional mesh of networks will end on a logistic regression layer. Linear residual connections are used in the convolutional layers which are gathered into 14 modules. According to its settings this model beats the Inception V3 for ImageNet dataset. It uses the model parameters in an efficient way as compared to Inception V3. Here a  $1 \times 1$  convolution is done before an  $n \times n$  convolution so it changes the order of operations. It also dismisses non-linearities throughout the model. For our model we used,  $75 \times 75 \times 3$  image size to work for 27 epochs and get best results. Simply speaking it is a linear pile of depthwise separable convolutions having residual connections. Hence, the model is very easy to modify and define with approx. 40 lines of code.

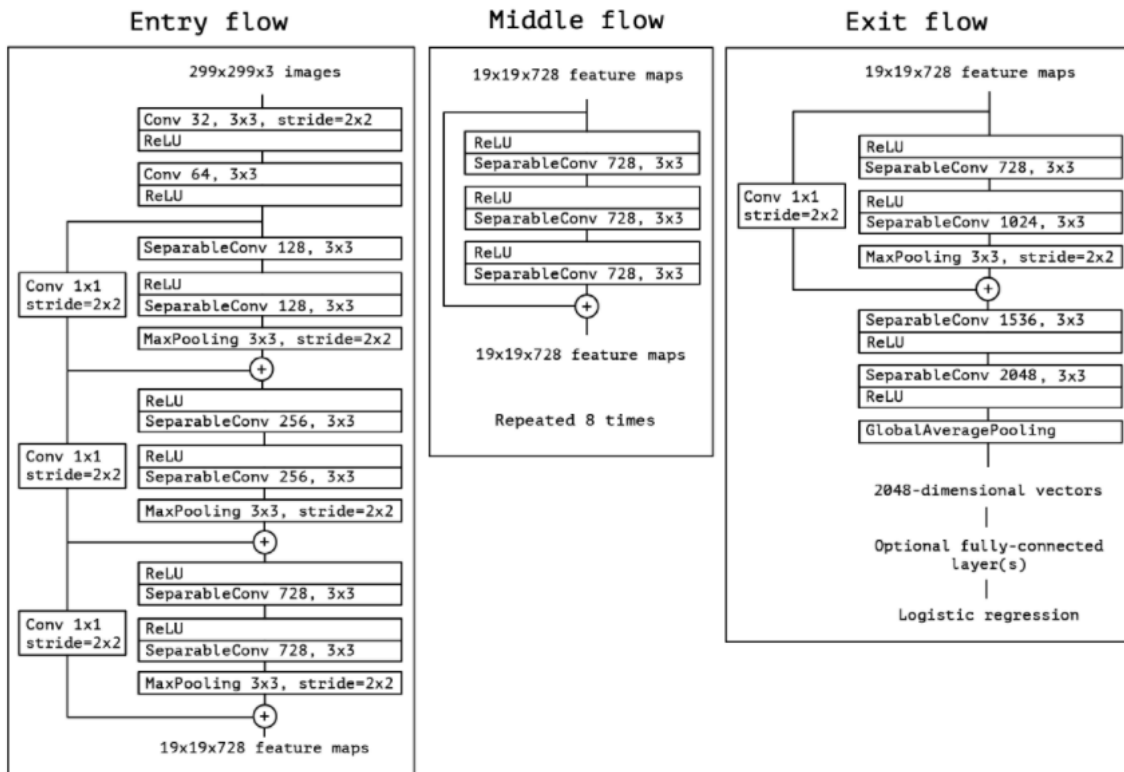


FIGURE 36: XCEPTION MODEL SUMMARY

## Chapter 4

# RESULTS

Our integrated dataset of handwritten numerals is manually developed and has 22 classes for Urdu, Persian and English numerals, the details of which are explained in the previous chapters. These classes have 1000 images each so making a total of 22000 images by 28x28x3 shape. We segregate our models into two categories – one is the custom-built CNN which has 3 versions and other is the transfer learning model which has 4 architectures. We divided our dataset in 80- 20 ratio for train and test data. Out of train data we picked validation data of 10%. The experiments that we performed with all these models to get their finest results are discussed in the following subsections.

We conducted the experiments using Python Keras language on an Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz @2.30 GHz desktop system with 16 GB RAM and an Nvidia K80 / T4 GPU. However, the CPU is not used for training or testing. Our computer works on Windows 20H2 version. For development purposes we used Python 3.6. 9, Keras 2.4.3, and TensorFlow 2.6.0.

### A. LEARNING CURVES

#### 1. CUSTOM BUILT CNN MODELS

For our three versions of custom-built CNN, we tried out different parameter and modified their settings to yield accurate results. Their learning curves gave us huge insight as to what kind of learning experience each of these had. Figure 37, Figure 38 and Figure 39 show the learning curves for each of the three CNN versions that we tried out. All of the plots show no overfitting or underfitting which is a good measure of the accuracy of our models. It is to be noted here that these plots were the one that were achieved after considerable hyperparameter tuning and though they may appear to be almost perfect but show either less accuracy or huge epochs for convergence. The plots for training loss and validation loss decrease till they achieve a point of stability. It is useful to track training and validation loss to evaluate the batches during forward pass. A very high or very low learning rate impacts these plots by showing an unnatural curve which would turn upwards or downwards in an abrupt manner. A low learning rate makes improvements in an almost linear fashion while a high value results in an exponential growth of curve. So, an ideal value is achieved after considerable selection of learning rates. Another major point to be considered in this implementation is that certain values for learning rate work well with SGD optimizer while others work best for Adam. Since, we used SGD in CNN v1 and v2, we kept a value of 0.1 for both of these. But when we added more parameters like momentum and Nesterov accelerated momentum for v3 we changed this value to 0.001 after trying out various values as it submitted the prime results. Also, for both CNN v1 and v2, we used 80 and 50 epochs respectively where

they finally converged. These epochs were yet again tuned by hit and trial. The Figure 38 shows convergence of values after almost 30 epochs and iterations more than that were not necessary.

The CNN v3 employs Adam as an optimizer, early stopping as a halting technique and batch normalization for standardization process of layers, so it yields the best accuracy among the three of them. Its optimization learning curve in Figure 39 shows a wiggle in the loss which is because of Adam optimizer's performance. Moreover, its performance learning curve shows a remarkable achievement of accuracy in almost 10 epochs which is remarkable considering we had huge dataset. This was possible because of early stopping as it got excellent result in minimum time.

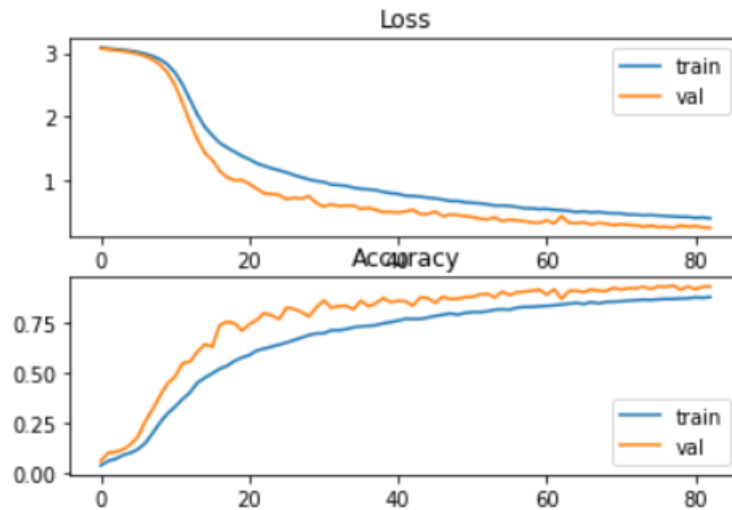


FIGURE 37: LOSS AND ACCURACY CURVE FOR CNN V1

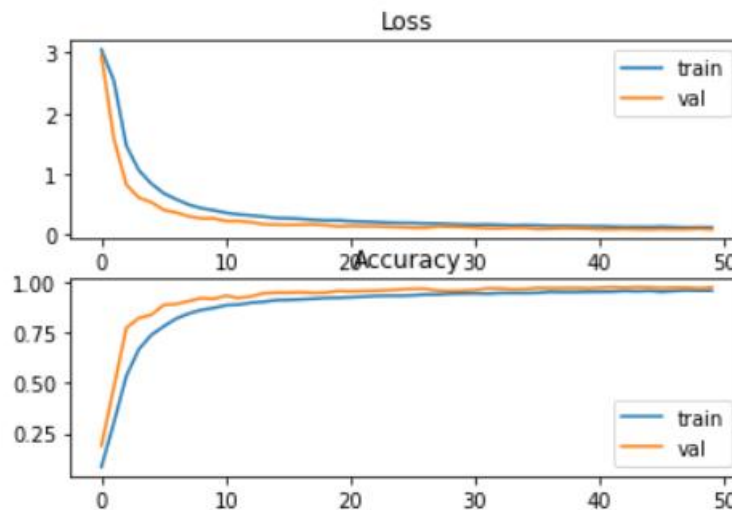


FIGURE 38: LOSS AND ACCURACY CURVE FOR CNN V2



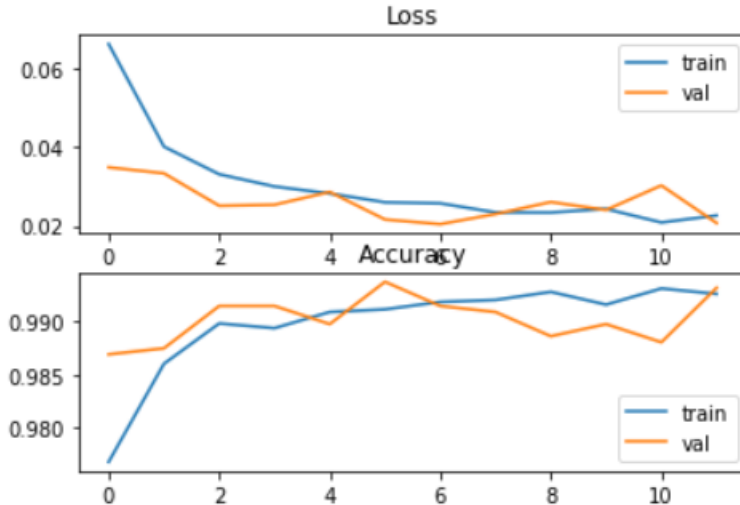


FIGURE 39: LOSS AND ACCURACY CURVE FOR CNN V3

## 2. TRANSFER LEARNING MODELS

In order to evaluate the performance of our models, we used models from Keras.applications library. For all these 4 implementations, we keep some redundant settings same. The last layers are frozen and replaced with global average pooling fed into the RELU activation map. Also, the image size is kept as 75 x75 x3 for all the architectures. We only feed the augmented images into the models while using early stopping. Figure 40, Figure 41, Figure 42 and Figure 43 demonstrate the optimization and performance learning curves for our integrated dataset. We used the popular architectures VGGNET, ResNet, GoogLeNet because they show excellent results for problems ranging from image classification to semantic and instance-based object segmentation. Moreover, the advanced techniques like RNN, fast RCNN, Feature pyramid networks are built on the lines of these networks. Thus, it is deemed necessary that we use these architectures too to validate our study on our novel dataset.

All the models depicted reasonable convergence in a small number of epochs after considerable hyper parameter tuning except for GoogLeNet V3 which converged in a time span of 7 minutes only. The learning curve for VGGNet shows its convergence at 45 iterations with an accurate fit of training and validation accuracies as shown in Figure 40. Due to its depth and complexity of operations it takes longer time in comparison to other architectures. So, although it achieves remarkable results it is tiresome to deploy it owing to its large weights, training time and resources required. The GoogLeNet V3 boosts the accuracy of inception module for classification tasks. It is inception module acts as a multi-level feature extractor thus reducing the need for redundant convolution layers. This argument is validated in Figure 41 where a start of huge accuracy differences leads to convergence in just a few epochs without using immense resources. Out of all the four models, it converged quickly in over 5 minutes.

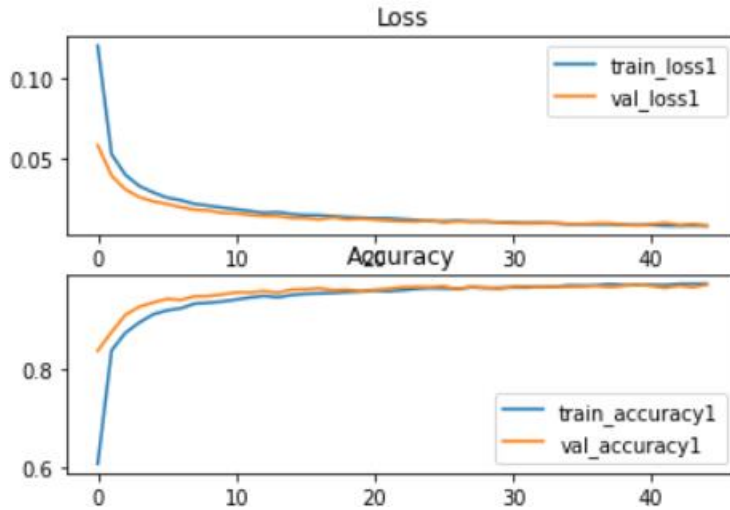


FIGURE 40: LOSS AND ACCURACY CURVE FOR VGGNET

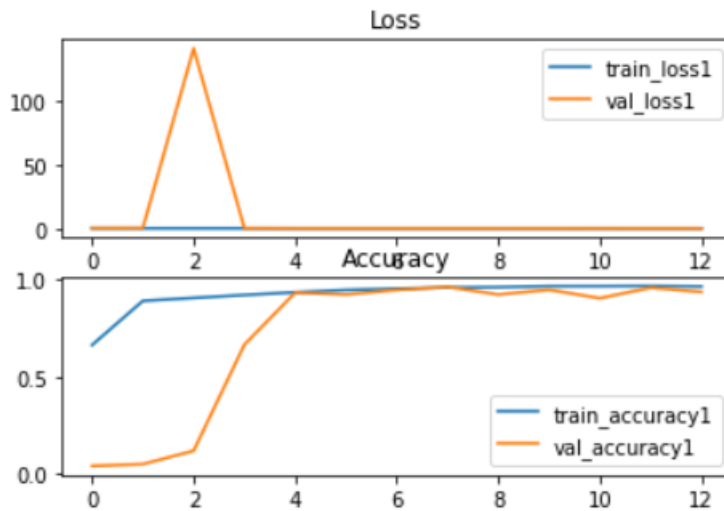


FIGURE 41: LOSS AND ACCURACY CURVE FOR GOOGLNET (INCEPTION V3)

Like GoogLeNet, ResNet has been the winner of ILSRVC in the early 2000s. It basically relies on micro-architecture modules to build extremely deep networks. It drastically reduces the feature space and attains a smaller model by using global average pooling rather than simple fully connected layers. Figure 42 shows its representation of accuracy and loss curve which yet again shows an accurate fit curve. The last architecture under discussion is the advanced Xception which is an extension of the inception net architecture. It uses L2 regularization as a built-in technique which surges the learning curves. But as all is well that ends well, it converges beautifully in minimum number of iterations.

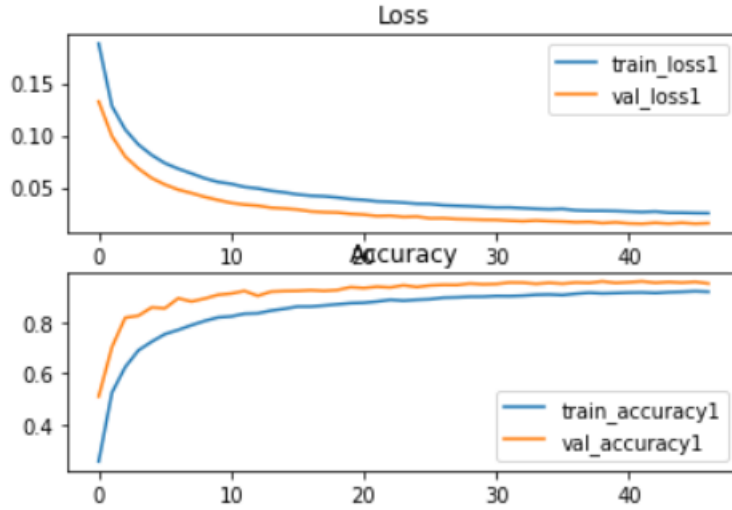


FIGURE 42: LOSS AND ACCURACY CURVE FOR RESNET50

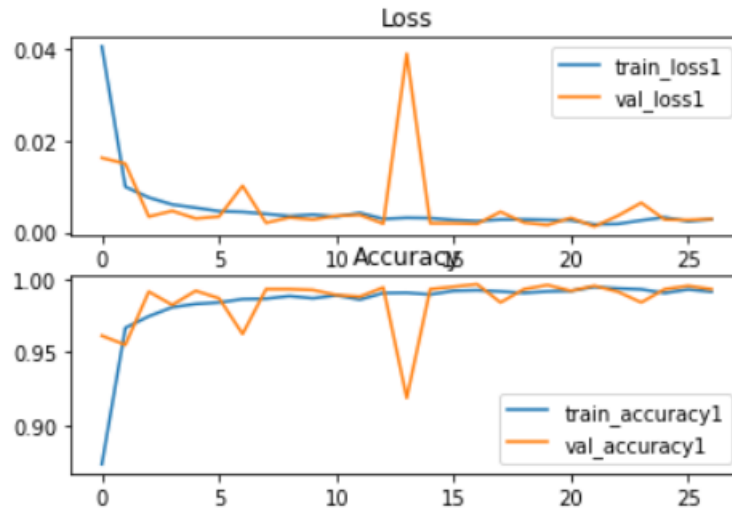


FIGURE 43: LOSS AND ACCURACY CURVE FOR XCEPTION

## B. EVALUATION METRICS

To achieve the optimal state of models, we picked out accuracy to get a fair comparison of the experiments that we carried out. Accuracy is the ratio of correctly detected images to the total number of input images which is written in equation 7 as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (7)$$

Here TP = True positive which denotes the correctly predicted labels,

TN = True negative stands for the correctly predicted labels that do not belong to a particular class,

FP = False positive denotes those incorrectly identified labels that did belong to the class,

FN = False negative represents the incorrectly predicted identities that did not belong to the detected class.

So, classification accuracy concerns with how often the classifier detected the correct labels. It is quite noteworthy that our custom-built neural network that we call ‘CNN v3’ achieved remarkable accuracy which is almost equivalent to that achieved by deep learning models. It not only learned the underlying distribution of the data but also performed exceptionally well on test data. There was slight variation of the training and validation accuracy for the proposed model, which shows that the model was not subjected to overfitting. This also changed the validation and testing score of the model by decreased it slightly upon increase of training data size. The highest accuracy is achieved by Xception which is the latest architecture in the deep learning era. It has less parameters, requires less hyperparameter tuning, consumes least memory among all and is relatively deep. Although VGGNet, GoogLeNet and ResNet are runner ups in their respective order but still their accuracies are above 95% which is groundbreaking for a unique dataset. Another aspect is that these high accuracies are achieved on augmented dataset so that also brought enough variations into the data. The custom-built CNN v1 and v2 achieved reasonable accuracies but they are not worth using since they just show the excess experiments that we performed. Table 5 shows the accuracies achieved on train, validation, and test data for each model.

TABLE 5: COMPARISON OF ACCURACIES OF ALL MODELS

Models	Train accuracy	Validation accuracy	Test accuracy
CNN v1	93.46%	87.65%	92.98%
CNN v2	97.84%	96.19%	97.09%
CNN v3	<b>99.31%</b>	<b>99.37%</b>	<b>98.91%</b>
VGGNet16	97.24%	97.04%	97.18%
GoogLeNet (Inception V3)	96.92%	96.42%	93.72%
ResNet50	96.02%	92.14%	94.79%
Xception	<b>99.46%</b>	<b>99.65%</b>	<b>99.01%</b>

## C. TEST RESULTS

To validate our results, we give some random sample images to all our models and judge its predicted label. By Table 5, we can deduce that Xception and custom-built CNN performed best out of all the models. Although the accuracy of CNN V3 is not on that much of a difference with

Xception but still its runner up. The theory behind this difference of accuracies could be evaluated based on Xception's paper [45]. This model performs depthwise convolution and pointwise convolutions instead of conventional convolution. Depthwise convolutions are the channel-wise convolution in spatial  $n \times n$  dimension so instead of lengthy and intense operations like in conventional convolution operation, it just performs convolution in  $n \times n$  dimension. Pointwise convolution is another name for the  $1 \times 1$  convolution so it works only to change the dimensions of channels. Hence, upon comparison with original convolution operations, these modifications are performed across specified channels which reduce the parameters, weights to be learned and also the time complexity of model. Another important change in this network is that pointwise convolution is done before depthwise convolution which is inspired from GoogLeNet Inception V3. Hence, it outperforms all the other models mentioned in this thesis. In the Figure 44 below we show random samples of images and their predicted labels by the best model out of all i.e., Xception.

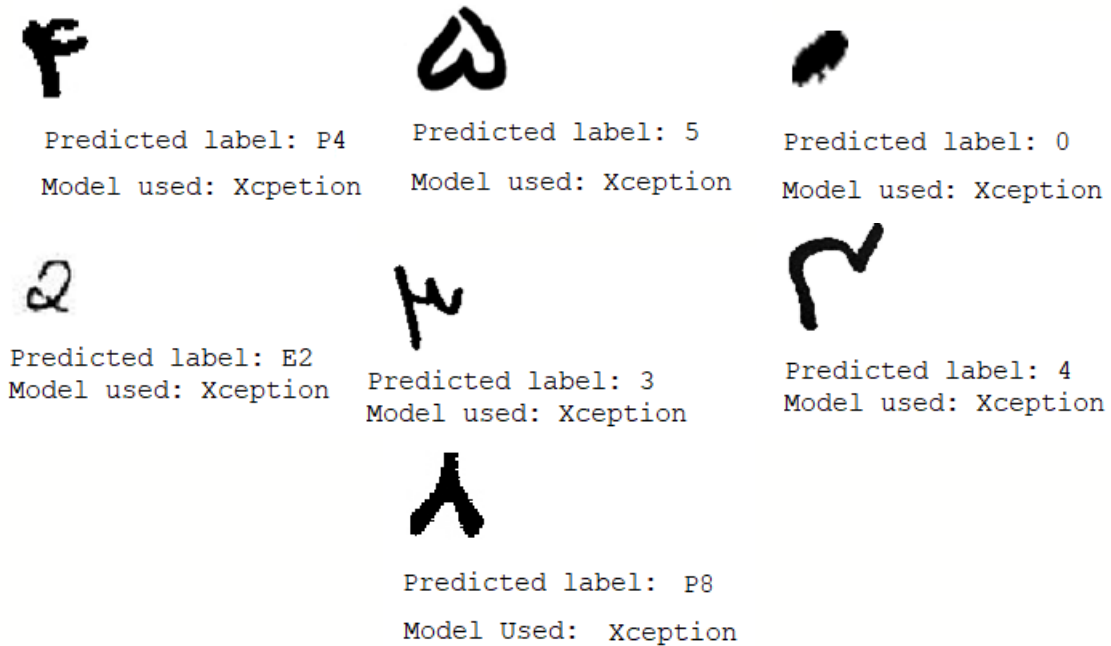


FIGURE 44: PREDICTED LABELS FOR XCEPTION

## Chapter 5

# DISCUSSION

Moving forward we discuss the performance, parameters, cost, and computational size of each model. The VGGNet16 achieved better performance in terms of classification accuracy and optimization losses. Its convergence rate seemed to be significant at initial epochs in comparison to other models. With our representable portion of dataset, Inception V3 did not experience divergence which is a clear sign of exact fit instead of overfit. Also, it does not demand huge epochs to get better accuracy thus, early stopping proved to be excellent for it. Out of all the models ResNet50 performed worse. It achieved very low scores at initial epochs but improved with larger iterations. This happened because of residual connections which increases the complexity of the model. Lastly, Xception outperformed all the models by achieving excellent accuracies for all subsets of data.

### A. COST CALCULATIONS

To get further insight of our models, we analyze the cost and space specifications of all these models. Although they are deep learning models which are supposed to be heavy weight and expensive to perform, still usage of proper hyperparameters can yield an intermediate sized model which can achieve better accuracies. It is evident from Table 6 that our custom built CNNs use minimum number of parameters, weights and in turn have less depth but still outperform the deep learning models.

TABLE 6: ACCURACY, DEPTH, WEIGHTS AND PARAMETERS OF DEEP LEARNING MODELS

Models	Accuracy	Depth	Weights size	Parameters
CNN V1	93.46%	15	96MB	24,093,244
CNN V2	97.84%	15	97 MB	25,609,913
CNN V3	99.31%	9	84 MB	14,722,960
VGGNet	97.24%	16	528 MB	138,357,544
ResNet	96.92%	50	99 MB	25,636,712
GoogLeNet (Inception V3)	96.02%	48	92 MB	23,851,784
Xception	99.46%	36	88 MB	22,910,480

VGG model is relatively four time deep than the custom-built CNN and have approximately 50 times more parameters and in turn more weights. On the other hand, ResNet model is similar to VGG model in terms of architecture but is deeper with extra parameters. Its number of parameters

are approximately 5 to 6 times more than VGGNet. Lastly, the number of parameters and depth of Inception V3 is similar to that of ResNet. Among these deep network architectures, only Xception is the one with fewer layers, parameters, weights and also achieves remarkable accuracy on our novel data. So, it was found that Xception and CNN v3 are the least expensive models out of all.

Table 6 shows comparison between parameters, weights, depth, and time complexity of each model. We conducted the experiments using Python Keras language on an Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz @2.30 GHz desktop system with 16 GB RAM and an Nvidia K80 / T4 GPU. However, the CPU is not used for training or testing. Our computer works on Windows 20H2 version. For development purposes we used Python 3.6.9, Keras 2.4.3, and TensorFlow 2.6.0. Our custom built CNNs use minimum time since they are built with less layers and have less parameters. VGGNet takes the most time since it has huge number of parameters and calculates more non-linearities for each layer. GoogLeNet and ResNet take almost similar time as their parameters are also similar in number but still, they get lost in finding features within their modules. Xception beats the timing among transfer learning models since it has lesser parameters, layers, and weights.

## B. COMPARISON WITH EXISTING OTHER TECHNIQUES

The proposed methods are compared with other CNN based architectures for handwritten numerals belonging to Urdu, Persian, English, or a set of these classes. The Table 7 shows that our proposed models have outperformed the previous architectures in terms of accuracy and amount of dataset.

TABLE 7: COMPARISON WITH STATE-OF-THE-ART MODELS

Reference	Dataset	No. of images	Classifier	Accuracy
[46]	Urdu numbers from printed documents	2000 samples	KNN and SVM	98.128%
[47]	Handwritten Urdu characters	900 samples	AlexNet, GoogLeNet, ResNet18	AlexNet (93.14%), GoogLeNet (91.04%), ResNet18 (89.97%)
[48]	Handwritten Urdu numbers	8000 numeral images	CNN	98.3%
[49]	Urdu handwritten numbers	17740 numerals	OCR-AlexNet, OCR-GoogLeNet	OCR-AlexNet (96.3%), OCR-GoogLeNet (94.7%)
[50]	Handwritten Urdu digits	7000 numbers	Xception	98.94%

[51]	Urdu digits and characters		Random forest	98.44%
Proposed Custom CNN v3	<b>Handwritten Urdu, Persian and English numbers</b>	<b>22,000 images</b>	<b>CNN</b>	<b>99.31%</b>
Proposed Xception model	<b>Handwritten Urdu, Persian and English numbers</b>	<b>22,000 images</b>	<b>Xception</b>	<b>99.46%</b>

Also, in comparison to the previous papers, in terms of test accuracy, our approach achieved 99.31% as compared to their previous maximum accuracy of 98.3% which is optimal considering that our numeral dataset is novel. This dataset shows promising training and validation accuracies on GoogLeNet, ResNet, VGGNet, Xception and proposed approach.



## Chapter 6

# CONCLUSION

In this dissertation, we present important research in the field of handwritten text in Urdu, Persian and English languages that yields benchmark performance on all the proposed architectures. As Urdu is a complex language which is bidirectional too with its numerals written from left to right while script written in opposite direction. This induces complexities in the recognition process. Its numerals are written on similar patterns as of Persian. So, we employ Urdu and Persian handwritten numerals with English numerals to make a novel dataset that can be used for classification and recognition of all the three languages. We present a unique way to exploit the similarities between these languages while keeping in view our main goal of reviving Urdu language's importance in research field.

Our proposed approach is remarkably noteworthy for Urdu text research and its related practical applications. We performed repetitive experiments in detail to incorporate the strokes in handwriting styles which are present in Persian and English languages so that our model can work on similar pattern for Urdu language too. By the inclusion of deep learning architectures, the capability of learning process of model is enhanced, and hence state-of-the-art results are deduced. We present a combination of Urdu, English, and Persian handwritten numbers to build a deep learning-based model that can categorize the different numbers. Versions of custom-built convolutional neural network (CNN) are implemented to achieve remarkable accuracy in recognizing numerals. Along with our own proposed CNN, we use CNN architectures VGGNet, ResNet, GoogLeNet (Inception V3) and Xception to achieve remarkable results.

### **A. FUTURE WORK**

In the future, we plan on increasing Urdu numeral dataset and then make it publicly available so as to motivate researchers to work in this field. Increasing this dataset will also increase the accuracies on all models. Besides this, our dataset and CNN can help develop a system to identify and count currency notes as Pakistani currency notes have both English and Urdu digits written on them. Using our model, OCR system will be able to read both kinds of numbers as it has been trained on a data that is an amalgamation of both these languages. Its usability could be in Pakistan's NADRA system for processing national ID cards where both English and Urdu numbers could be identified. Although the number plates in Pakistan have been renewed recently by making them according to a pattern so that they can be identified by cameras. But still there are so many people who have not adopted this system so a recognizer that could read both Urdu and English numerals could help in this way. The development of handwritten numeral recognizer can remove the barriers faced in different writing styles, poor quality of image graphics or illegible

handwriting. Since performance of deep learning algorithms in real world applications is of utmost importance, so we plan on testing it on other applications such as recognizing Surah numbers of The Holy Quran and numbers on Pakistani postage stamps. The sole motivation of this paper is to bring our mother language Urdu on competitive level with all the latest research.

## REFERENCES

- [1] R. Singh, "A Literature Review On Handwritten Character Recognition Based On Artificial Neural Network," *International Journal Of Computer Sciences And Engineering*, vol. 6, no. 11, pp. 753-758, 2018.
- [2] H. N. a. H. Y. Shunji Mori, "Optical character recognition," John Wiley & Sons, Inc., 1999.
- [3] "Urdu 11th most spoken language in world: Study," NATION, CURRENT AFFAIRS, [Online]. Available: <https://www.deccanchronicle.com/nation/current-affairs/200119/urdu-11th-most-spoken-language-in-world-study.html>. [Accessed 25 May 2021].
- [4] "Abstract of speakers' strength of languages and mother tongues," Government of India, 2001.
- [5] M. Lewis, *Ethnologue: Languages of the World*, Summer Institute of Linguistics, Inc., 2009.
- [6] BBC, "BBC - Languages - Urdu - A Guide To Urdu - 10 Facts About The Urdu Language," 2020. [Online]. Available: <https://www.bbc.co.uk/languages/other/urdu/guide/facts.shtml>. [Accessed 2021].
- [7] M. Mirzayeva, "History Of Urdu Language And Its Status In India And Pakistan," *ACADEMICIA: AN INTERNATIONAL MULTIDISCIPLINARY RESEARCH JOURNAL*, vol. 11, no. 2, pp. 584-591, 2021.
- [8] "What are the top 200 most spoken languages?," [Online]. Available: <https://www.ethnologue.com/guides/ethnologue200>. [Accessed 09 April 2021].
- [9] Central Intelligence Agency, "Field Listing - Languages, The World Factbook," Central Intelligence Agency, 2000.
- [10] D. Graddol, "The Future of English?," The British Council, 1997.
- [11] H. K. & E. Kabir, "Introducing a very large dataset of handwritten Farsi digits and a study on their varieties," *Pattern Recognition Letters*, vol. 2, no. 28, 2007.
- [12] L. e. B. Y. B. P. H. Yann LeCun, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.

- [13] R. S. S. G. H. Neha Gautam, "Eastern Arabic Numerals: A Stand out from Other Jargons," in *International Conference on Computational Intelligence and Communication Networks (CICN)*, 2015.
- [14] S. Abdelazeem, "Comparing Arabic and Latin Handwritten Digits Recognition Problems," *International Journal of Computer and Information Engineering* , pp. 1583 - 1587, 2009.
- [15] N. K. G. Herleen Kour, "Machine Learning approaches for Nastaliq style Urdu handwritten recognition: A survey," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020.
- [16] J. Memon, M. Sami, R. A. Khan and M. Uddin, "Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR)," *IEEE Access*, vol. 8, pp. 142642-142668, 2020.
- [17] S. M. Shoab Ahmed Khan, "Urdu online handwriting recognition.," in *Proceedings of the IEEE Symposium on Emerging Technologies*, Islamabad, Pakistan, 2005.
- [18] M. S. M. I. K. N. U. L. JAVED++, "Hand-written Urdu Numerals Recognition Using Kohonen Self Organizing Maps," *SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)*, vol. 47, no. 3, pp. 403-406, 2015.
- [19] I. A. H. S. N. M. I. R. R. Y. SAAD BIN AHMED, "Evaluation of Handwritten Urdu Text by Integration of MNIST Dataset Learning Experience," *IEEE Access*, vol. 7, pp. 153566-153578, 2019.
- [20] H. K. F. K. K. K. A. A. W. S. S. M Ameen Chhajro, "Handwritten Urdu character recognition via images using different machine learning and deep learning techniques," *INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY*, p. 9, 2020.
- [21] Y. S. P. I. Alejandro Baldominos, "Evolutionary Convolutional Neural Networks: An Application to Handwriting Recognition," *Neurocomputing* , vol. 283, p. 38–52, 2018.
- [22] G. E. Hinton, A. Krizhevsky and S. D. Wang., "Transforming Auto-Encoders," *ICANN 2011*, pp. 44-51, 2011.
- [23] H. A. ., M. M. S. ., S. K. C. T. Talha Iqbal, "Capsule-Net for Urdu Digits Recognition," in *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Metz, France, 2019.
- [24] W. Jiang, "Evaluation of deep learning models for Urdu handwritten characters," in *Journal of Physics: Conference Series*, China, 2020.

- [25] M. M. Rawia Ahmed, "Preprocessing Phase for Offline Arabic Handwritten Character Recognition," *International Journal Of Computer Applications Technology And Research* , vol. 5, no. 12, pp. 760-763, 2016.
- [26] S. A. H. S. N. H. u. R. Tabassam Nawaz, "Optical character recognition system for Urdu (Naskh font) using pattern matching technique," *Int. J. Image Process*, vol. 3, no. 3, pp. 92-103, 2009.
- [27] A. S. F. A. S. A. Husain, "Online Urdu character recognition system," in *Proc. IAPR Conf. Mach. Vis. Appl. (MVA)*, 2007.
- [28] R. U. N. A. K. N. Khalil Khan, "Urdu Character Recognition using Principal Component Analysis," *International Journal of Computer Applications* , vol. 60, no. 11, pp. 1-4, 2012.
- [29] M. S. M. A. K. A. A. a. M. M. Zahoor Jan, "Online Urdu handwriting recognition system using geometric invariant features," *The Nucleus*, vol. 53, no. 2, pp. 89-98, 2016.
- [30] S. S. a. A. Wahab, "Optical character recognition system for Urdu," in *2010 International Conference on Information and Emerging Technologies*, Karachi, 2010.
- [31] N. H. Khan and A. Adnan, "Urdu Optical Character Recognition Systems: Present Contributions and Future Directions," *IEEE Access*, vol. 6, pp. 46019-46046, 2018.
- [32] G. H. Laurens van der Maaten, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579-2605, 2008.
- [33] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: zero-padding vs. interpolation," *Journal of big data*, vol. 6, no. 8, 2019.
- [34] H. Ide and T. Kurita, "Improvement of learning for CNN with ReLU activation by sparse regularization," *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2684-2691, 2017.
- [35] H. K. Hossein Gholamalinezhad, "Pooling Methods in Deep Neural Networks, a Review," *Computer Vision and Pattern Recognition*, 2020.
- [36] D. E. Rumelhart, G. Hinton and R. J. Williams, "Learning representations by back-propagating errors," vol. 323, p. 533–536.
- [37] M. B. Chaoyue Liu, "Accelerating SGD with momentum for over-parameterized learning," *arXiv preprint*, 2018.

- [38] J. B. Diederik P. Kingma, "Adam: a Method for Stochastic Optimization," in *International Conference on Learning Representations*, 2015.
- [39] T. Dozat, "INCORPORATING NESTEROV MOMENTUM INTO ADAM," ICLR, 2016.
- [40] Yann LeCun, Y. B. & and G. Hinton, "Deep Learning," *Nature*, pp. 436-44, 2015.
- [41] A. Z. Karen Simonyan, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint* , 2014.
- [42] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. A. R. Christian Szegedy, "Going Deeper with Convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.
- [43] W. L. Y. J. P. S. S. R. D. A. D. E. V. V. A. R. Christian Szegedy, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [44] K. He, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [45] F. Chollet, "'Xception: Deep Learning with Depthwise Separable Convolutions.'", *arXiv preprint*, 2017.
- [46] H. Sharma, D. V. Sharma and G. S. Lehal, "Extraction and Recognition of Numerals from Machine-Printed Urdu Documents," in *Computer Vision and Image Processing*, Jaipur, India, 2019.
- [47] S. P. M. Y. C. Mohammed Aarif KILVISHARAM OZIUDDEEN, "A Novel Deep Convolutional Neural Network Architecture Based on Transfer Learning for Handwritten Urdu Character Recognition," 2020.
- [48] M. Husnain, M. M. S. Missen, S. Mumtaz, M. Z. Jhanidr, M. Coustaty and M. M. Luqman, "Recognition of Urdu Handwritten Characters Using Convolutional Neural Network," *MDPI* , 2019.
- [49] S. P. Mohammed Aarif K.O, "OCR-Nets: Variants of Pre-trained CNN for Urdu Handwritten Character Recognition via Transfer Learning," in *Third International Conference on Computing and Network Communications (CoCoNet'19)*, 2020.
- [50] W. Jiang, "Evaluation of deep learning models for Urdu handwritten characters recognition," in *Journal of Physics: Conference Series*, 2020.

[51] M. A. Chhajro, H. Khan, F. Khan, K. Kumar, A. A. Wagan and S. Solangi<sup>2</sup>, "Handwritten Urdu character recognition via images using different machine learning and deep learning techniques," in *INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY*, 2020.