DECLARATION

We hereby declare that the project entitled Nepali Barna Recognition is an authentic record of our own work carried out in the Electronics and Computer Engineering Department, Himalaya College of Engineering under the guidance of Er. Shyam Krishna Khadka during 7th and 8th semester (2018).

Date:

Roll.No	Name	Signature
43788	Ronish Magar	_ _ _ _ _ _ _ _
43790	Roshan Bhattarai	_ _ _ _ _ _ _ _
43792	Rupesh Thakur	_ _ _ _ _ _ _ _
43797	Saroj Subedi	_ _ _ _ _ _ _ _

Counter Signed by

Project Supervisor:

Er. Shyam Krishna Khadka

Department Of Electronic and Computer Engineering

Himalaya College of Engineering

ACKNOWLEDGEMENT

We would like to thank Institute of Engineering for setting a Major Project as part of the fourth year curriculum. We would like to thank Himalaya College of Engineering for providing the physical infrastructure and platform to conduct our project.

We would like to thank Department of Electronics and Computer Engineering of Himalaya College of Engineering for encouraging and supporting us to do a project of this scale. We express our gratitude towards Er. Ashok Gharti Magar the Head of Department for supporting us and encouraging us do the project. We are thankful to our project coordinator Er. Narayan Adhikari Chhetri and supervisor Er. Shyam Krishna Khadka for guiding us to do our project.

We are equally grateful to all the teachers who directly or indirectly helped us build this idea for the project.

Group members.

Ronish Magar (43788)

Roshan Bhattarai (43790)

Rupesh Thakur (43792)

Saroj Subedi (43797)

ABSTRACT

Speech recognition has been playing a major role in the development of various virtual assistants popular in the market these days. It is the preliminary stage of development of an intelligent voice activated virtual assistants.

However, such speech recognition has not been available in Nepali language. Therefore, the creation of '"Nepali Barna Recognition', a Nepali speech recognition application, which will understand Nepali alphabets (Barna) which is an initial step in the development of Nepali speech recognition.

This project work has three primary contributions: first, a convolutional neural network architecture for Nepali alphabets. Second, creating standard datasets for nepali alphabets, Third, the use of audio data augmentation for overcoming the problem of data scarcity and explore the influence of data augmentation on the performance of the proposed CNN architecture.

Here, 4345 numbers of original data were augmented to create 12210 augmented data which sums up to 16555 total data and evaluated the effect of data augmentation to our speech model. This input data is split into train set (60%, 9933) and test set (40%, 6622) and fed into the CNN network. The prediction accuracy obtained by implementing this dataset to the best model gave the training accuracy of 98.42% and test accuracy of 98.31% .

Keywords: *Artificial Intelligence, Automatic Speech Recognition, Convolution Neural Network, MFCC, Natural Language Processing, Speech Processing, Virtual Assistant, Data Augmentation.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI : Artificial Intelligence

API : Application Programming Interface

ASR : Automatic Speech Recognition

CNN : Convolution Neural Network

Er. : Engineer

HMM : Hidden Markov Model

Inc. : Incorporation

LPC : Linear Predictive Coding

MFCC : Mel-Frequency Cepstral Coefficients

1. INTRODUCTION

1.1 Background

Speech recognition is the interdisciplinary sub-field of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text by computers. It is process of decoding acoustic speech signal captured by microphone or telephone, to a set of words. It is also known as "automatic speech recognition" (ASR), "computer speech recognition", or just "speech to text" (STT). It incorporates knowledge and research in the linguistics, computer science, and electrical engineering fields.

Speech recognition has been widely used in voice user interfaces such as voice dialing (e.g. "Call home"), call routing (e.g. "I would like to make a collect call"), home automation appliance control, search (e.g. find a podcast where particular words were spoken), simple data entry (e.g., entering a credit card number), preparation of structured documents (e.g. a radiology report), speech-to-text processing (e.g., word processors or emails), and aircraft (usually termed direct voice input).

Speech Recognition development has advanced over time in English language. However, only Google recently released speech to text conversion for Nepali language which still is on beta stage. This project aims to contribute to the development of more accurate Nepali speech recognition tool. This could further be used in the assistance to those Nepalese community who still don't know how to write and can be used for learning Nepali text.

Initially, after the completion of this project it is highly likely to be assistive to kids since they can learn Nepali Barnamala with fun.

1.2 Objectives

The objectives of this project are:

- To develop Convolutional Neural Network (CNN) based model for Nepali Speech Recognition.
- To learn more about various aspects of neural networks.

1.3 Features

- A prototype Nepali Speech Recognition application.
- Recognizes Nepali Barna (i.e. Nepali for alphabets) and convert it to text.

1.4 Scope and Application

There are limitless scope and application of Speech recognition such as in home automation and every voice activated devices. This project finds its scope mainly in Nepali community, since its aim is to provide a backbone in development of Nepali speech recognition application. This project can lead to another step to train it to understand everyday spoken Nepali word and we can further achieve it to understand full sentences. This can eventually lead to various possibilities like assistive typing for physically disabled ones, integration in voice activated machines and many more.

Initially, the completed project will be mainly applicable in following uses:

- Users will be able to type Nepali alphabets verbally without physical contact to the device.
- Helpful for physically disabled peoples.
- This application can be implemented in education applications for who cannot write and read Nepali can learn by speaking to the application.
- For making learning and studying easy for children.
- Base for developing virtual assistant applications in Nepali.

2. LITERATURE REVIEW

2.1 Early Development of Speech Recognizers

Early attempts to design systems for automatic speech recognition were mostly guided by the theory of acoustic-phonetics, which describes the phonetic elements of speech (the basic sounds of the language) and tries to explain how they are acoustically realized in a spoken utterance. These elements include the phonemes and the corresponding place and manner of articulation used to produce the sound in various phonetic contexts. For example, in order to produce a steady vowel sound, the vocal cords need to vibrate (to excite the vocal tract), and the air that propagates through the vocal tract results in sound with natural modes of resonance similar to what occurs in an acoustic tube. These natural modes of resonance, called the formants or formant frequencies, are manifested as major regions of energy concentration in the speech power spectrum.

In 1952, Davis, Biddulph, and Balashek of Bell Laboratories built a system for isolated digit recognition for a single speaker [1], using the formant frequencies measured (or estimated) during vowel regions of each digit.

1950's, Olson and Belar of RCA Laboratories built a system to recognize 10 syllables of a single talker and at MIT Lincoln Lab, Forgie and Forgie built a speaker-independent 10-vowel recognizer. In the 1960's, several Japanese laboratories demonstrated their capability of building special purpose hardware to perform a speech recognition task. Most notable were the vowel recognizer of Suzuki and Nakata at the Radio Research Lab in Tokyo [2], the phoneme recognizer of Sakai and Doshita at Kyoto University [3], and the digit recognizer of NEC Laboratories [4]. The work of Sakai and Doshita involved the first use of a speech segmenter for analysis and recognition of speech in different portions of the input utterance. In contrast, an isolated digit recognizer implicitly assumed that the unknown utterance contained a complete digit (and no other speech sounds or words) and thus did not need an explicit "segmenter." Kyoto University's work could be considered a precursor to a continuous speech recognition system.

The next milestone in the development of voice recognition technology was achieved in the 1970s at the Carnegie Mellon University in Pittsburgh,

Pennsylvania with substantial support of the United States Department of Defense and its DARPA agency. Their tool "Harpy" mastered with about 1000 words the vocabulary of a three-year-old. About ten years later the same group of scientists developed a system that could not only analyze individual words but entire word sequences enabled by the Hidden Markov Model. Thus, the earliest virtual assistants, which applied speech recognition software were automated attendant and medical digital dictation software. In the 1990s digital speech recognition technology became a feature of the personal computer with Microsoft, IBM, Philips and Lernout & Hauspie fighting for customers. Much later the market launch of the first smartphone IBM Simon in 1994 laid the foundation for smart virtual assistants as we know them today. [5]

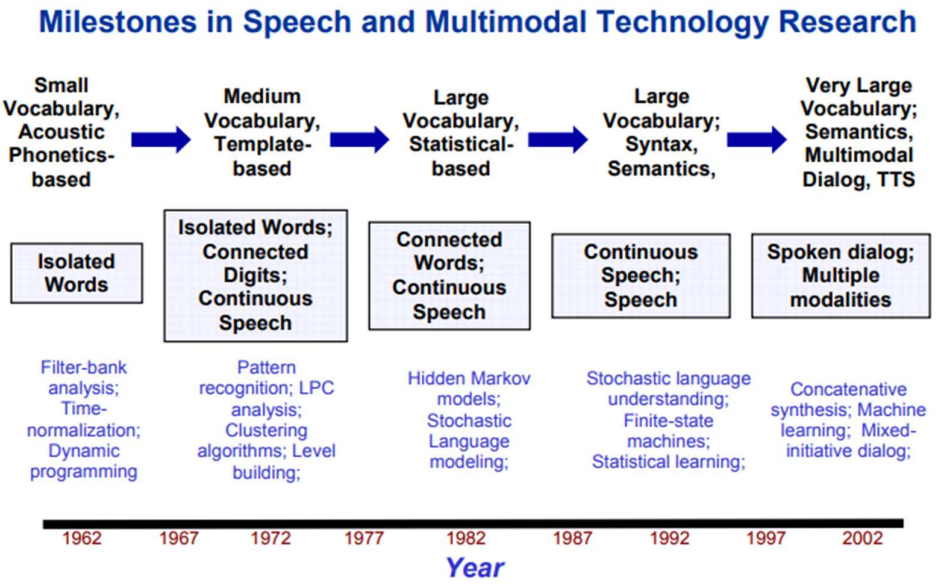

Fig. 2. 1 Milestones in Speech Recognition over 40 years [1]

2.2 Present Works on Speech Recognition

After more than six decades of research, the speech recognition have come up an advanced phase which has been our everyday need. Speech recognition has been implemented in every devices these days mainly on mobile devices. Since, the early development of Siri, every mobile device manufacturer has been working on their own speech recognition design approaches which then will be implemented as a virtual assistant on their manufactured devices.

These days speech recognition has been quite popular for its application in Virtual assistants. A virtual assistant, also called AI assistant or digital assistant, is an application program that understands natural language voice commands and completes tasks for the user.

2.2.1 Virtual Assistants

2.2.1.1 Siri

Siri is Apple Inc.'s cloud software that can answer users' various questions and give recommendations, due to its voice processing mechanisms. When in use, Siri studies the users' preferences (similar to contextual advertising) to provide each person with an entirely individual approach. This software solution is also useful for developers; in particular, the presence of API called SiriKit provides smooth integration with new applications developed for iOS and watchOS platforms.

2.2.1.2 Google Now

Google Now is an Android-based voice recognition application, which is launched by users uttering commands of the same name. This software features very advanced functions including web search, route optimization, memo scheduling etc. that can collectively help users solve a wide array of daily tasks. It finds quick answers and explore interest. It gets better the more it is used. Similar to Siri, the creators of Google Now offer Google Voice Interaction API. This interface can become a truly indispensable tool in the development of mobile applications for the Android platform.

2.2.1.3 Cortana

A virtual intelligent assistant with the function of voice recognition and AI elements, Cortana was developed for such platforms as Windows, iOS, Android, and Xbox One. It can predict users' wants and needs based on their search requests, e-mails, etc. One of Cortana's distinguishable features is her sense of humor. "She" can sing, make jokes and speak to users informally.

2.2.1.4 Amazon Echo

Amazon Echo combines, in itself, hardware and software that can search the web, help with scheduling of upcoming tasks and play various sound files all based on

voice recognition. A small speaker equipped with sound sensors, the device can be automatically activated by exclaiming “Alex.”

2.2.1.5 Bixby

Samsung’s Bixby application is another successful implementation of the AI concept. It also builds a unique user approach, based on interests and habits. Bixby features advanced voice recognition mechanisms, and uses the camera to identify images, based on markers and GPS. [6]

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

Since the project is aim to recognize some Nepali alphabets only, and the data sets seems relatively lower than to train the Neural Network for whole speeches and words storage is not likely to be the problem. Taking under consideration of the lots of computing power for efficient and fast data processing, we are developing the project on a system with a least following specifications:

- RAM : 8 GB minimum
- Processor : Intel i5 processor
- 64 bit system architecture

An additional high memory Dedicated Graphics card can enhance the computational performance.

3.2 Software Requirements

The following software applications are used in throughout the whole project life cycle:

- Linux/Windows OS
- Jupyter Notebook
- Python libraries : Librosa,
- Deep learning framework: Keras
- Web framework : Django

3.3 Functional requirements

The functional requirements of the system are as follows:

- The system should be able to detect speech from the given input.
- The system should be able to take training data, images to train the system.
- From the supplied training data, system should be able to train and save the model.
- The system should be able to classify the input image into satisfactory expression.
- The system should be able to give satisfactory result.

3.4 Non-Functional requirements

The nonfunctional requirements of our system are encompassed in the following points:

3.4.1. Performance

The system shall have a quick performance. Recording and prediction of the input shall not take more than 15 seconds.

3.4.2. Accuracy

The system shall have a fairly accurate result. Prediction may not be always accurate but it should give satisfactory result.

3.4.3. Scalability

The system shall be scalable so that it can incorporate new set of data into training the model and the system can use the new data along with previous data to save the model.

3.4.4. Reliability

The system shall be reliable. It shall be able to recognize the given voice input with the same amount of accuracy and performance for the longer period of time.

4. METHODOLOGY

4.1 System Block Diagram:

Following figure shows the System diagram of implementation of the project. Each components is described along the report:

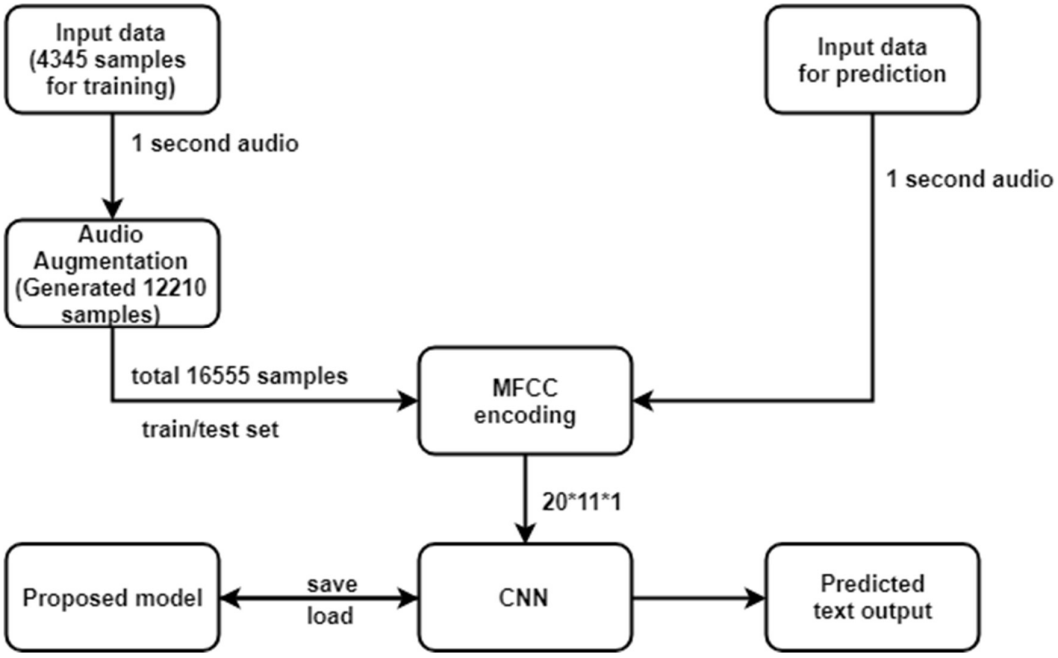

Fig. 4. 1 System Diagram

4.2 Input data:

Each data is recorded manually by the PC's microphone. User speaks an alphabet which will be recorded with sampling rate of 16000 Hz in mono mode. The audio clip will be one second long. Audio files are recorded for each nepali alphabets and organized in labels. A total of 4345 samples of data were recorded for training.

Fig. 4. 2 Labeled Datasets

4.4 Audio Augmentation

Data augmentation is a common strategy adopted to increase the quantity of training data. It is a key ingredient of the state of the art systems for image recognition and speech recognition. With the widespread adoption of neural networks in speech recognition systems which require a large speech database for training such a deep architecture, data augmentation is very useful for small data sets [7].

Data augmentation does not always give freedom or as much removal of overfitting as a new dataset will do since it only add some random but nearest values to standard data.

For data augmentation we used following techniques:

• **Pitch Shifting** : raising or lowering the pitch of the audio sample (while keeping the duration unchanged).

• **Dynamic Range Compression** : compressing audio signals dynamic range by reducing the volume of loud sounds or amplifying quiet sounds

• **Background Noise** : mixing the sample with noise created by differing the amplitude of the sample.

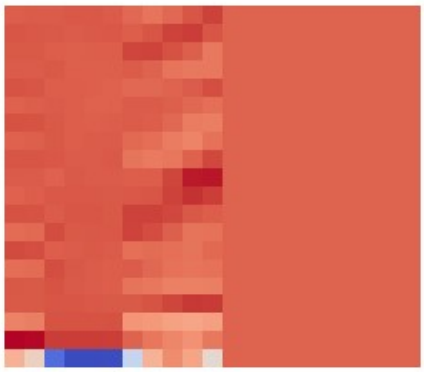

Fig. 4. 3 MFCC spectrum of Original 1 second “ka” audio clip

Fig. 4. 4 MFCC spectrum of pitch shifted “ka” audio clip

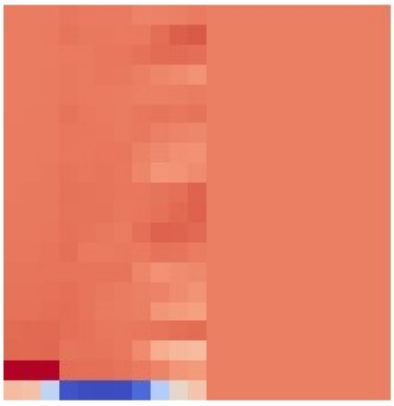

Fig. 4. 5MFCC spectrum of noise added “ka” audio clip

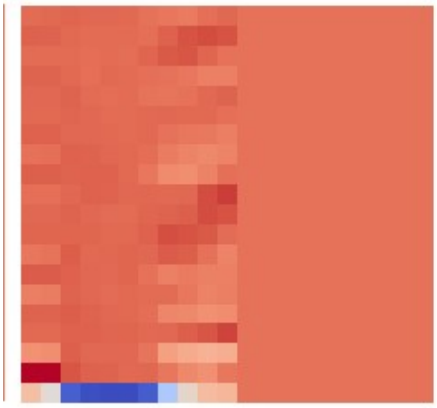

Fig. 4. 6 MFCC spectrum of dynamic range compressed “ka” audio clip

Training/ Test set:

Training set is the set of data created after all processings. It is the set of training data and augmented data. Training set consists a total of 16555 samples. The training set is splitted to train set and test set as well. 60% of the total training set is splitted to train set and the remaining is splitted to test set which is used to check the accuracy of the model.

4.4 MFCC Encoding

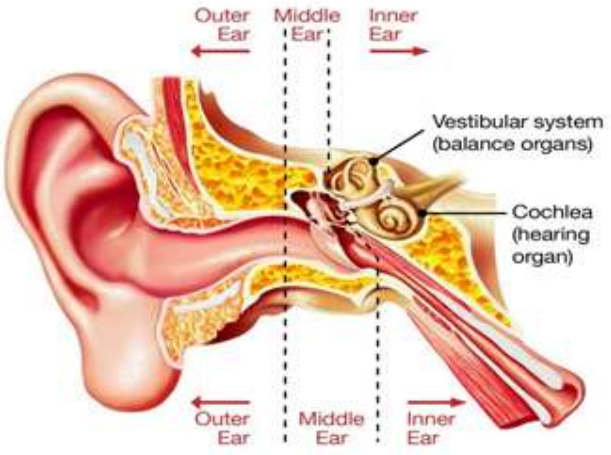

Fig. 4. 7 Human Ear

The main point to understand about speech is that the sounds generated by a human are filtered by the shape of the vocal tract including tongue, teeth etc. This shape determines what sound comes out. If we can determine the shape accurately, this

should give us an accurate representation of the phenomena being produced. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding all the other stuff which carries information like background noise, emotion etc.

Mel Frequency Cepstral Coefficients (MFCCs) are a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980's, and have been state-of-the-art ever since. Prior to the introduction of MFCCs, Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) and were the main feature type for automatic speech recognition (ASR), especially with HMM classifiers.

The most prevalent and dominant method used to extract spectral features is calculating Mel-Frequency Cepstral Coefficients (MFCC). MFCCs are one of the most popular feature extraction techniques used in speech recognition based on frequency domain using the Mel scale which is based on the human ear scale. MFCCs being considered as frequency domain features are much more accurate than time domain features. [8]

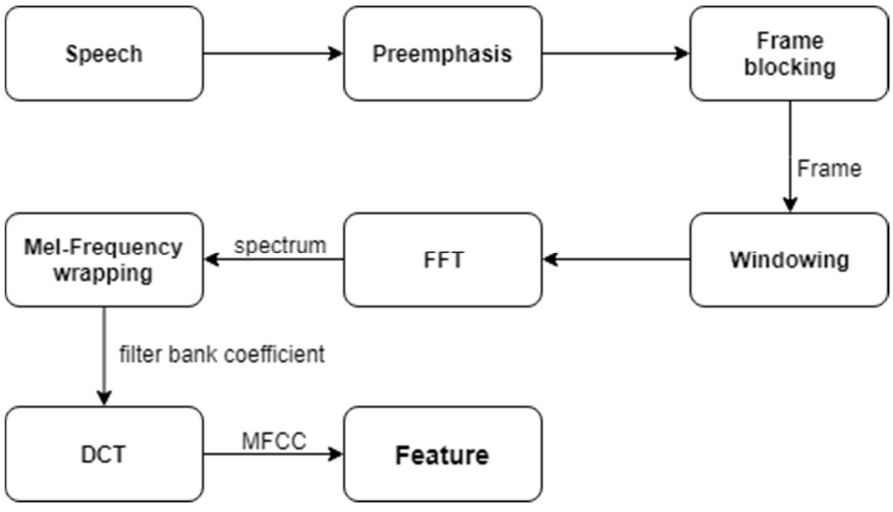

Fig. 4. 8 MFCC encoding process

4.4.1 Algorithm for MFCC Calculation

Step 1: Get the speech or audio file.

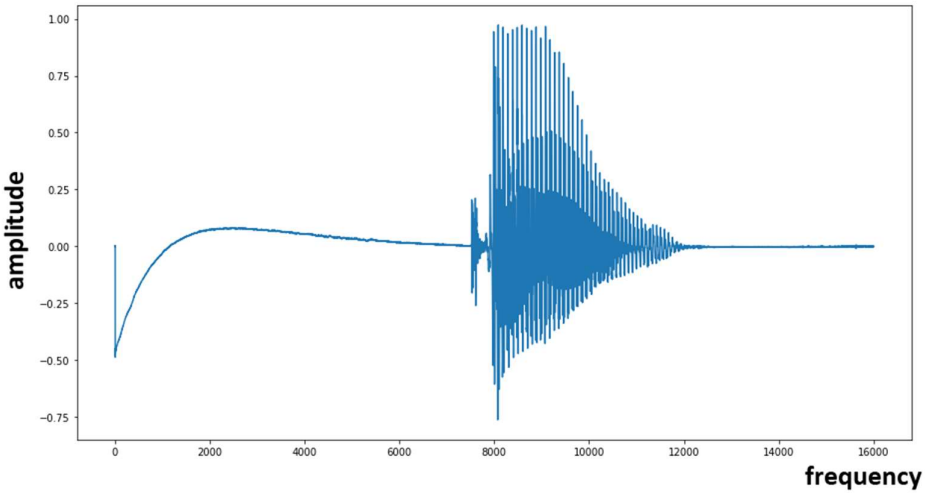

Fig. 4. 9 One second audio clip saying “ka” at 16000 Hz

Step 2: Pre-emphasis

In the process of speech signal pre-emphasis filter is required after the sampling process. The purpose of this filtering is to obtain a smoother spectral form of speech signal frequency. Where the spectral shape is relatively high value for low areas and tends to fall sharply to the area of frequency above 2000 Hz. The pre-emphasis filter is based on the input / output relationship in the time domain expressed in the following equation:

$$y(n) = x(n) - a * x(n-1) \text{ ----------(1)}$$

where:

y(n) is the output wave,

x(n) is the input wave,

a is pre-emphasis filter constant, it is usually $0.9 < a < 1.0$

Typical values for a (= 0.95 or 0.97).

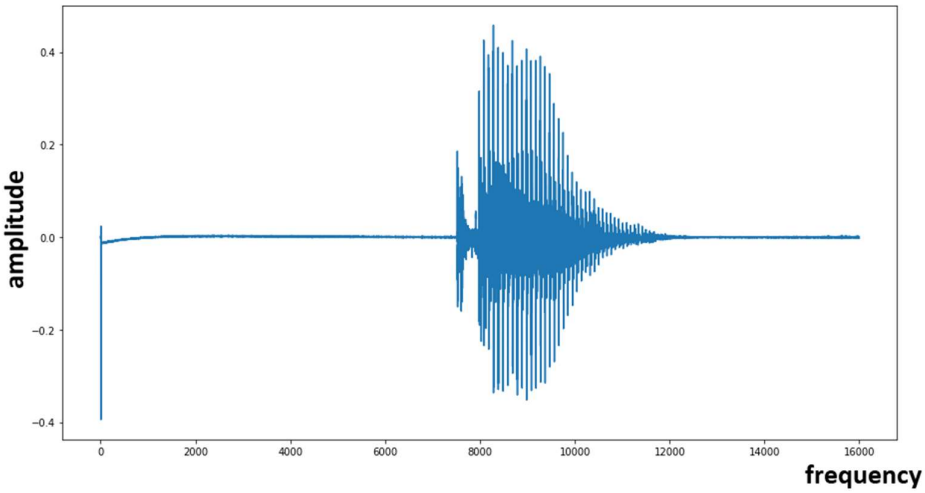

Fig. 4. 10 Pre-emphasized audio clip with a=0.97

Step 3: Framing

An audio signal is constantly changing, so to simplify things assumption is made that on short time scales the audio signal doesn't change much. This is why the signal is framed into 20-40ms frames. If the frame is much shorter there isn't enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.

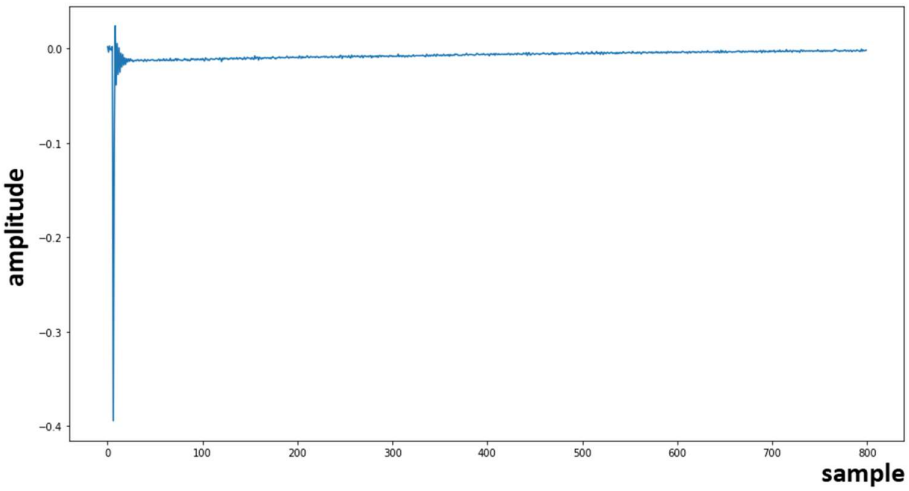

Fig. 4. 11 First 800 samples framed at 50ms

Step 4: Windowing

Windowing is a process for analyzing long sound signals by taking a sufficiently representative section. Windowing is a Finite Impulse Response (FIR) digital filter approach. This process removes the aliasing signal due to the discontinuity of the signal pieces. Discontinuities occur due to the frame blocking process. If the window is defined as w(n),$0 \leq n \leq N-1$, where N is the number of samples in each frame, the result of windowing is a signal:

$$y(n) = x(n) * w(n), \quad 0 \leq n \leq N-1 \text{ ----------(2)}$$

y(n) is the result signal of the convolution between the input signal and the window function and x(n) represents the signal to be convolved by the window function. Where w(n) usually uses window Hamming which has the form:

$$w(n) = 0.54 - 0.46 \cos(2*pi*n/N-1), \quad 0 \leq n \leq N-1 \text{ ----------(3)}$$

where, N is the Frame length or samples in frame.

window function is applied to each frames.

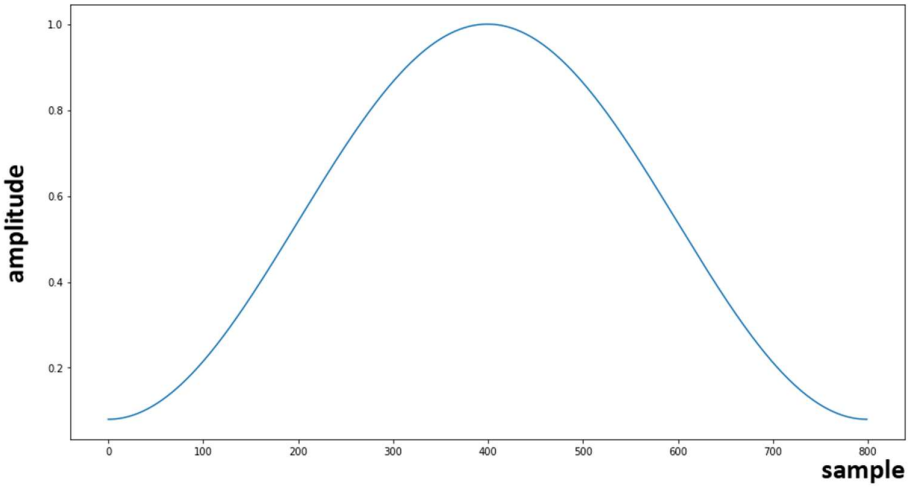

Fig. 4. 12 Simple hamming window for 800 samples.

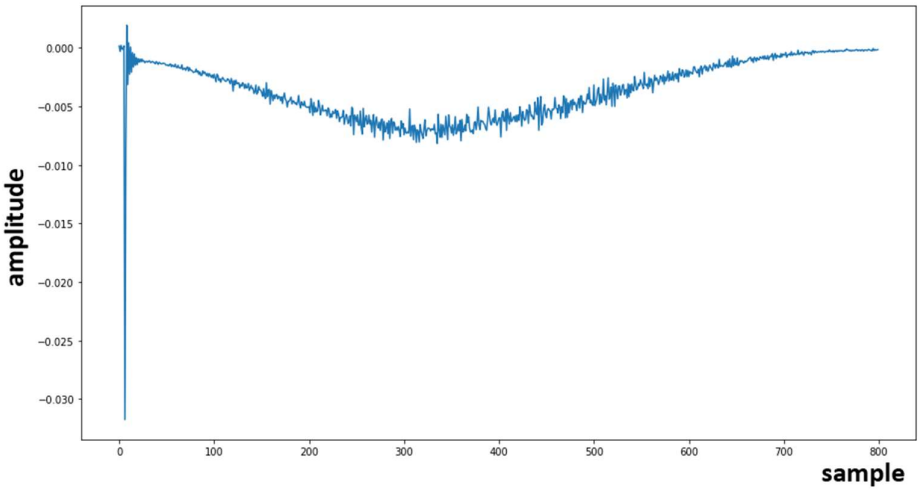

Fig. 4. 13 Hamming window applied to the first frame

Step 5: Fast Fourier Transform (FFT)

To convert each frame of N samples from time domain into frequency domain. The frame that has undergone the windowing process is converted into a frequency spectrum. FFT is a fast algorithm of Discrete Fourier Transform (DFT) which is useful for converting every frame to N samples from time domain into frequency domain. It is applied to each frames in the wave signal. It is also known as STFT (short time fourier transform). FFT reduces the repeatable multiplication contained in the DFT.

$$X_n = \sum_{k=0}^{N-1} x_k \, e^{-2*pi*j*k*n/N} \qquad \text{----------(4)}$$

Where, n = 0,1,2,3 ...N-1. Generally, N (= 256 or 512).

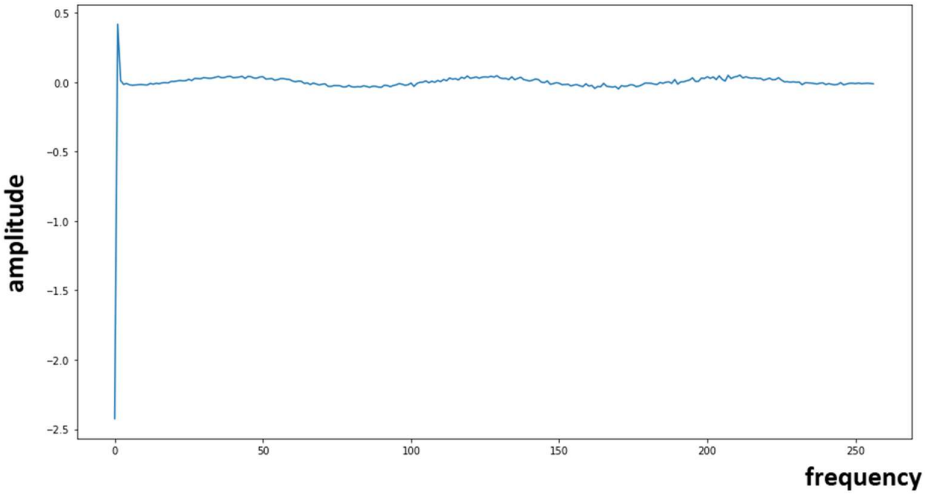

Fig. 4. 14 FFT of the first frame including the negative values. (N=512)

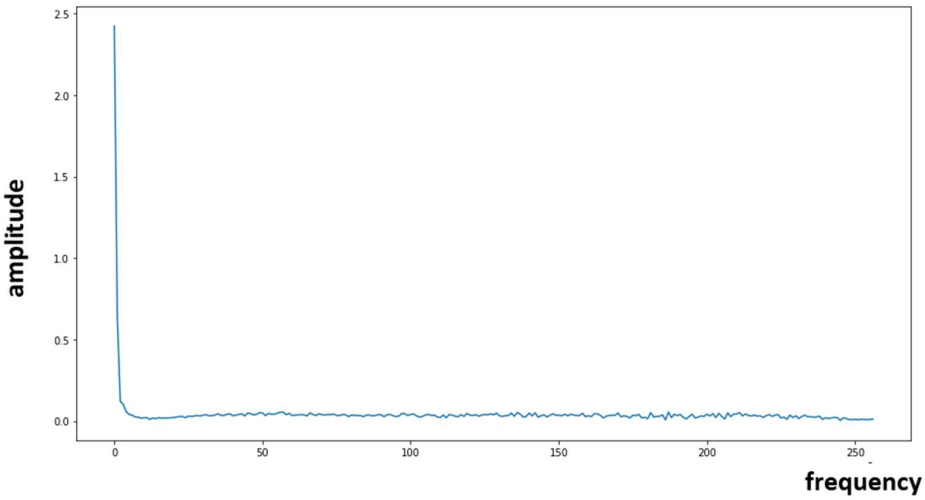

Fig. 4. 15 FFT of the first with absolute value. (N=512)

And, then the power spectrum of this absolute FFT was computed.

$$P = |FFT(X_i)|^2 / N \text{ ----------(5)}$$

Where, X_i is the i^{th} frame of the signal X.

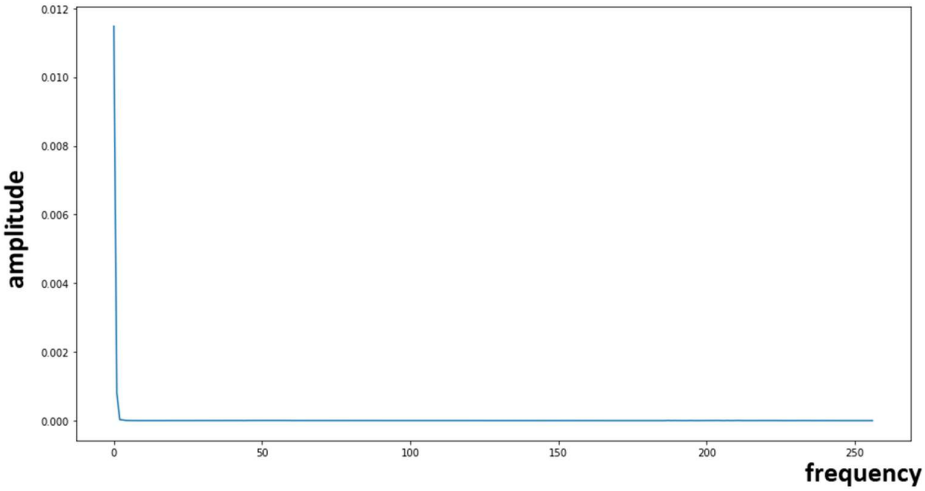

Fig. 4. 16 Power spectrum of the FFT with N=512.

The result of this stage is usually called Spectrum or periodogram.

Step 6: Mel Filter Bank Processing.

The perception of the human ear against the sound frequency does not follow the linear scale. The actual frequency scale uses units of Hz. The scale that works on the human ear is called the frequency Mel scale. The scale of Mel-Frequency is a low frequency that is linear under 1000 Hz and a logarithmic high frequency above 1000 Hz.

Conversion between Hertz (f) and Mel (m) is done using:

$m = 2595 \log_{10}(1 + f/700)$ ----------(6)

$f = 700 (10^{m/2595} - 1)$ ----------(7)

The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency. Humans are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes the features match more closely what humans hear.

The final step to computing filter banks is applying triangular filters, typically 40 filters on a Mel-scale to the power spectrum to extract frequency bands. The Mel-scale aims to mimic the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies.

Each filter in the filter bank is triangular having a response of 1 at the center frequency and decrease linearly towards 0 till it reaches the center frequencies of the two adjacent filters where the response is 0.

This is modeled by following equation:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k < f(m) \\ 1 & k = f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) < k \le f(m+1) \\ 0 & k > f(m-1) \end{cases} \quad \text{----------(8)}$$

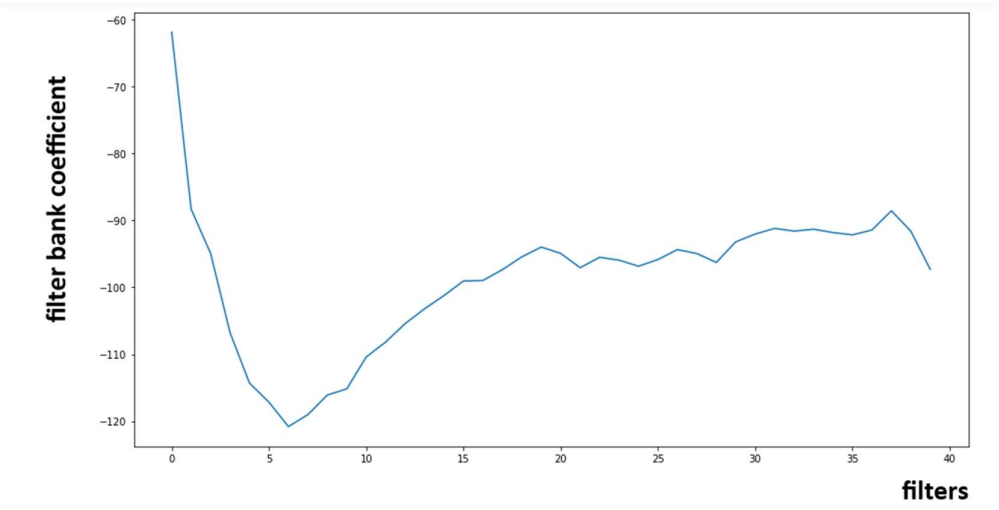

Fig. 4. 17 Filter Bank coefficients from first frame with 40 filters.

The aim of this step is to extract the frequency bands.

Step 7: Discrete Cosine Transform (DCT)

It turns out that filter bank coefficients computed in the previous step are highly correlated, which could be problematic in some machine learning algorithms. Therefore, Discrete Cosine Transform (DCT) is applied to decorrelate the filter bank coefficients and yield a compressed representation of the filter banks.

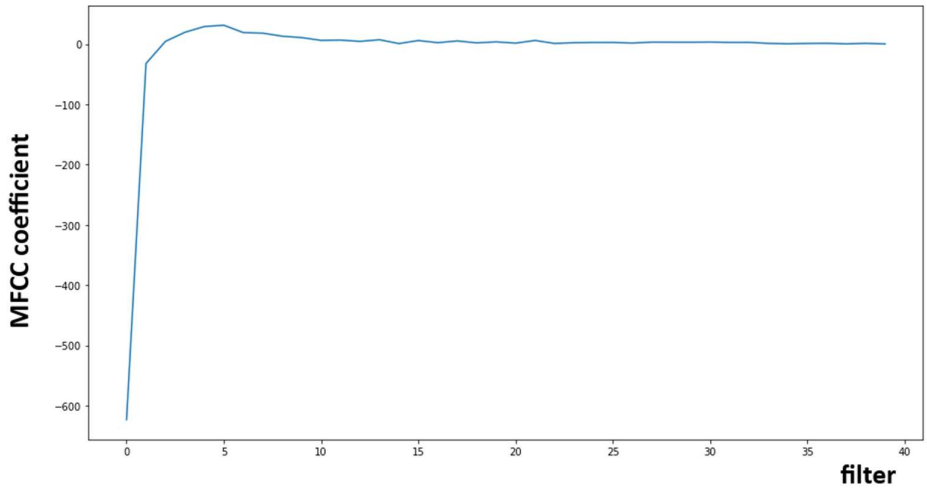

Fig. 4. 18 MFCC coefficients applying DCT to the filter bank coefficients.

Typically, for Automatic Speech Recognition (ASR), the resulting cepstral coefficients 2-13 are retained and the rest are discarded.

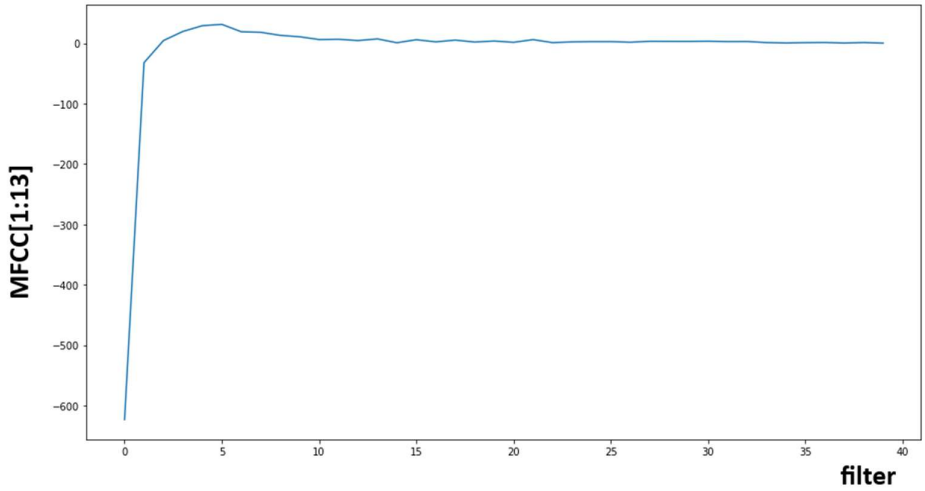

Fig. 4. 19 Selected coefficients for ASR from first frame.

The reasons for discarding the other coefficients is that they represent fast changes in the filter bank coefficients and these fine details don't contribute to Automatic Speech Recognition (ASR).

MFCCs :

```
[-6.23218146e+02 -3.22926874e+01  4.36254078e+00  1.94511272e+01
  2.89873750e+01  3.11704800e+01  1.89413759e+01  1.80248223e+01
  1.29816968e+01  1.06144694e+01  6.04145170e+00  6.50056320e+00
  4.36262293e+00  7.07713459e+00  7.41694578e-01  5.73204686e+00
  2.18112352e+00  5.09241685e+00  1.87941161e+00  3.56707288e+00
  1.42705597e+00  5.89400980e+00  9.36457885e-01  2.11104764e+00
  2.54217415e+00  2.55082534e+00  1.58108537e+00  3.11885895e+00
  3.01815484e+00  2.97528527e+00  3.24405802e+00  2.65866053e+00
  2.73651325e+00  1.05289757e+00  3.40152992e-01  9.17464402e-01
  1.18806576e+00  2.30268013e-01  1.07214148e+00  9.84504945e-02]
```

MFCC[1:13]

```
[-32.29119197   4.38572384  19.46564035  29.00265648  31.14021462
  18.91988628  18.01198616  12.98201656  10.61655626   6.03209177
   6.47562165   4.39357422]
```

Fig. 4. 20 MFCC extracted from Fig 4.16 and Fig.4.17

The result of the conversion is called Mel Frequency Cepstrum Coefficient. The set of coefficient is called acoustic vectors.

```
[[ 73.67742862  75.70143781  64.83667237  79.45842655  95.49710765
   52.23258217  13.93273989 -11.44592013 -40.37280182 -33.13126914
   41.23767422   0.           0.           0.           0.
    0.           0.           0.           0.           0.
    0.        ]]
...
[[  3.0324303    3.62034132   4.15016324   3.44037593   6.59755781
    8.90579758  11.34040784   8.15445391  -4.94723667 -19.00414971
  -15.72847944   0.           0.           0.           0.
    0.           0.           0.           0.           0.
    0.        ]]
```

Fig. 4. 21 MFCC Embedding and spectrum of audio “ka”

4.5 Convolutional Neural Network

Convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity.

CNN consists of one or more pairs of convolution and max pooling layers. A convolution layer applies a set of filters that process small local parts of the input where these filters are replicated along the whole input space. A max-pooling layer generates a lower resolution version of the convolution layer activations by taking the maximum filter activation from different positions within a specified window. Higher layers use more broad filters that work on lower resolution inputs to process more complex parts of the input. Top fully connected layers finally combine inputs from all positions to do the classification of the overall inputs. This hierarchical organization generates good results in image processing tasks. [5]

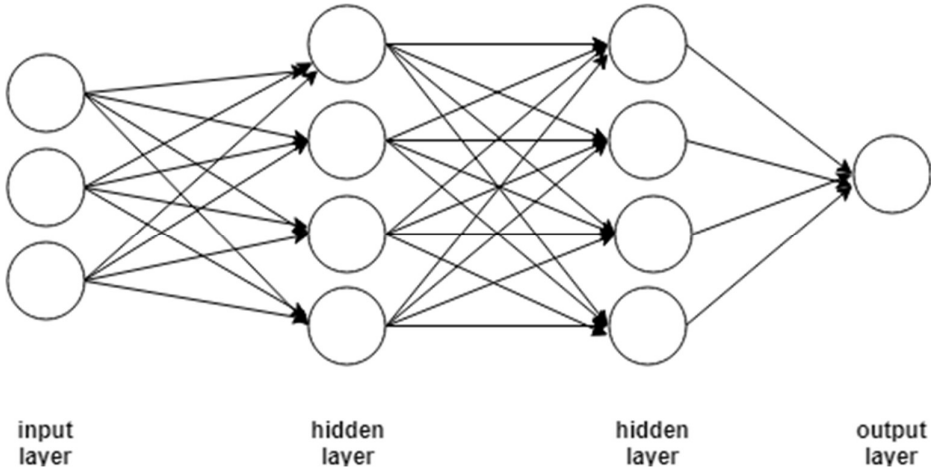

Fig. 4. 22 A 3 layered Fully Connected Convolutional Neural Network

4.5.1 Various terms in CNN:

Convolution layer:

The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

For a given input of size (WxHxD), the output from the convolution layer can be calculated using the formula:

$$W=(\frac{W-F+2P}{S})+1 \ \& \ H=(\frac{H-F+2P}{S})+1$$

where, *W= width of input*

H= height of input

F= Filter size

P= Padding size

S= Stride

D= Depth (or channel size)

Pooling layer: A pooling layer performs a down sampling operation along the spatial dimensions (width, height), resulting in lesser volume of input. This scans across the image using a window and compresses the image extracting features. [9] **Max pooling** and **Average pooling** are the most common pooling functions. Max pooling takes the largest value from the window of the image currently covered by the kernel, while average pooling takes the average of all values in the window.
For a given input of size (W1xH1xD1), the output from the max-pooling layer can be calculated using the formula:

- Requiring two hyper parameters:
  - their filter size F,
  - the stride S,
- Produces a volume of size W2×H2×D2, where:
  - W2=(W1−F)/S+1
  - H2=(H1−F)/S+1
  - D2=D1

Fully connected layer:

This layer is attached to the end of the network. This layer basically takes an input volume (whatever the output is of the conv or ReLU or pool layer preceding it) and outputs an N dimensional vector where N is the number of classes that the program has to choose from.

Basically, a FC layer looks at what high level features most strongly correlate to a particular class and has particular weights so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

Dropout Layer:

Using “dropout", certain units (neurons) are randomly deactivated in a layer with a certain probability p from a Bernoulli distribution. When half of the activations of a layer to zero, the neural network won’t be able to rely on particular activations in a given feed-forward pass during training. As a consequence, the neural network will learn different, redundant representations; the network can’t rely on the particular neurons and the combination (or interaction) of these to be present. Another nice side effect is that training will be faster. [10]

Activation Function: Activation function of a node defines the output of that node, given an input or set of inputs. It is used to determine the output of neural network like yes or no. It maps the resulting values in between 0 to 1 or -1 to 1 etc. (depending upon the function).

There are various types of Activation functions available such as Sigmoid, Tanh, ReLU, Softmax, etc. ReLU activation function is used in hidden layers and Softmax at last layer.

ReLU: f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

$$f(z) = \begin{cases} z \; ; z \geq 0 \\ 0 \; ; z < 0 \end{cases}$$

Softmax: The softmax function is a more generalized logistic activation function which is used for multiclass classification. The softmax function, or normalized exponential function, is a generalization of the logistic function that "squashes" a K-dimensional vector of arbitrary real values to a K-dimensional vector where each entry is in the range (0, 1], and all the entries add up to 1.

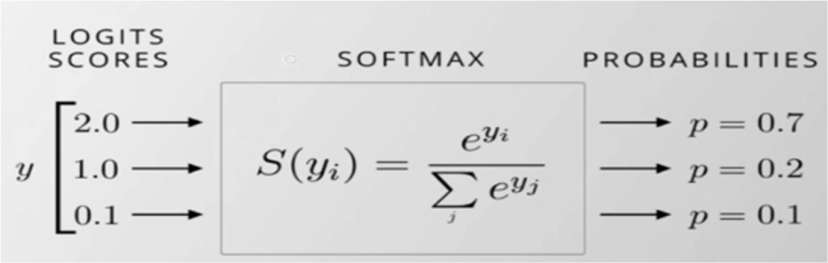

Fig. 4. 23 Softmax Function [11]

Stride: Stride is the number of pixels by which sliding is done in the filter matrix over the input matrix. When the stride is 1 then the filters are moved one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as sliding them around. Having a larger stride will produce smaller feature maps.

Training:

When the CNN starts, the weights or filter values are randomized. The filters don't know to look for edges and curves. The idea of being given an image and a label is the training process that CNNs go through.

The way the computer is able to adjust its filter values (or weights) is through a training process called backpropagation.

Back Propagation:

Backpropagation can be separated into 4 distinct sections, the forward pass, the loss function, the backward pass, and the weight update.

During the **forward pass**, the image is passed through the whole network. Since all of the weights or filter values were randomly initialized, the output will be an array, that doesn't give preference to any number in particular. The network, with its current weights, isn't able to look for those low level features or thus isn't able to make any reasonable conclusion.

This goes to the loss function part of backpropagation. A loss function can be defined in many different ways but a common one is MSE (mean squared error).

$\mathbf{E_{Total} = \sum ½*(target - output)^2}$

The loss will be extremely high for the first couple of training images. The amount of loss should be minimized. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases.

Backward pass helps to determine which weights contributed most to the loss and finding ways to adjust them so that the loss decreases.

Weight update is the process to take all the weights of the filters and update them so that they change in the opposite direction of the gradient.

$W = W_i - \eta * (dL/dW)$; W = weight, W_i = initial weight , η = learning rate

Testing:

Finally, to see whether or not the CNN works, different set of images and labels (other than the trained images) were passed through the CNN and the output results are seen and compared to calculate accuracy and loss.

Epochs, Batch size, Iterations: One Epoch is when an entire dataset is passed forward and backward through the neural network only once.

Since, one epoch is too big to feed to the computer at once, it is divided in several smaller batches.

Batch size is the total number of training examples present in a single batch.

Iterations is the number of batches needed to complete one epoch.

Learning rate: Learning rate is a hyper-parameter that controls how much adjusting of weights is needed with respect the loss gradient. It takes a long time to converge.

The following formula shows the relationship:

New weight = existing weight – learning rate * gradient

If the learning rate is set too low, training will progress very slowly as there are tiny updates to the weights in the network. However, if the learning rate is set too high, it can cause undesirable divergent behavior in the loss function. [12]

Optimizers: Optimizers helps to minimize (or maximize) an Error function E(x) which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values(Y) from the set of predictors(X) used in the model [13].

Various available optimizers are Adam, Adagrad, AdaDelta, Gradient Descent, Stochastic Gradient Descent (SGD), Momentum, Nesterov Accelerated Gradient (NAG), Mini Batch Gradient Descent, etc.

Adam: Adam stands for Adaptive Moment Estimation. Adaptive Moment Estimation (Adam) is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients Adam keeps an exponentially decaying average of past gradients M(t).

AdaDelta: Adadelta is a more robust extension of Adagrad which tends to remove the decaying learning Rate problem of it. Instead of accumulating all previous squared gradients, Adadelta limits the window of accumulated past gradients to some fixed size w. This way, Adadelta continues learning even when many updates have been done. For AdaDelta no default learning rate is required.

4.5.2 The proposed CNN Architecture for Nepali Barna Recognition:

The Architecture of this model is shown below:

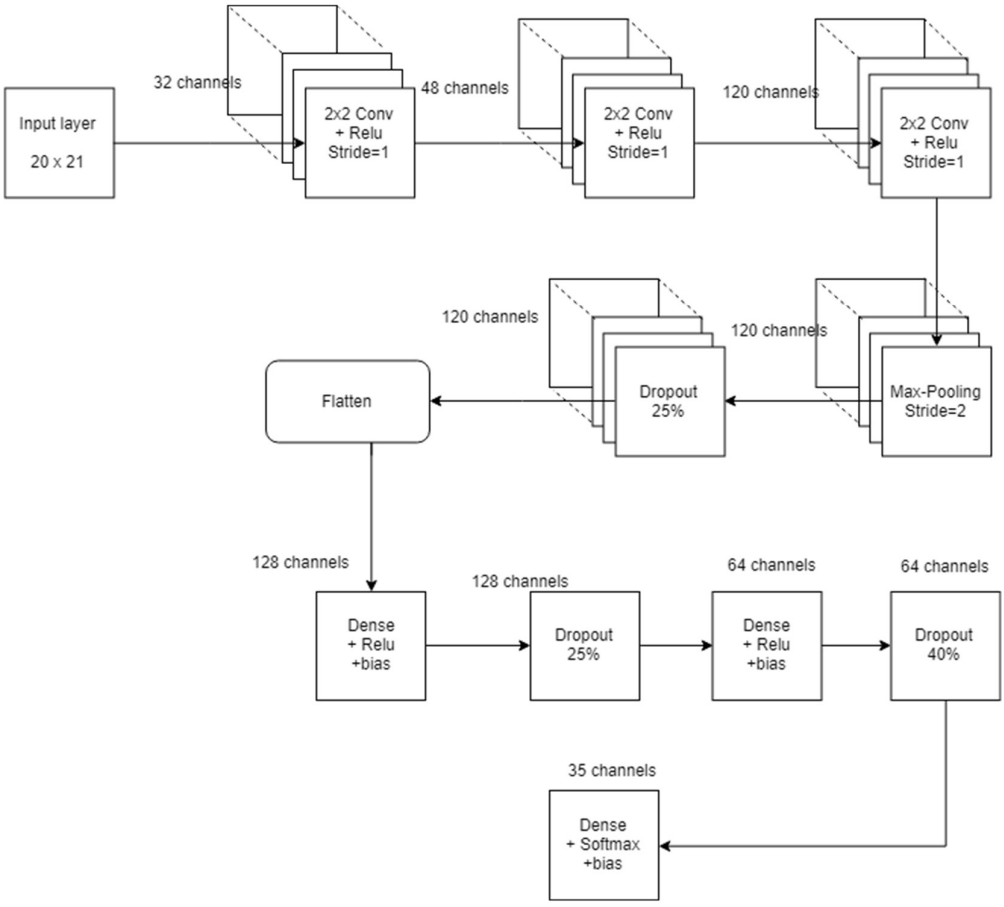

Fig. 4. 24 CNN Architecture

The Convolutional network proposed in this project is comprised of 10 layers, i.e. 3 Convolution layers, 1 pooling layers (Max Pooling), 3 Dropout layers and 3 fully-connected (Dense) layer. The input fed to the first layer is a MFCC vector of size 20 x 21 and the grayscale MFCC has channel 1 hence the input size is (20, 21, 1). The above architecture was selected by reviewing the various models which can be seen below with their training and test accuracy as well as respective losses:

Table 4. 1 Models with higher accuracy

Model name	Networks layers (sequential)	Training Data		Test Data	
		Accuracy	Loss	Accuracy	Loss
model1_aug	convolutional*3,Pooling, (dropout,dense)*3	0.9842	0.04826	0.9831	0.07495
model2_0_2_aug	convolutional*3,Pooling, (dropout,dense)*3	0.9646	0.116	0.9814	0.0759
model3_2	conv,(conv,pooling)*3, (dropout,dense)*3	0.9824	0.05989	0.8815	0.6425
model4_aug	(conv,pooling)*2 ,(dropout,dense)*3	0.9759	0.09357	0.9823	0.08093

Fig. 4. 25 Test accuracy of best models

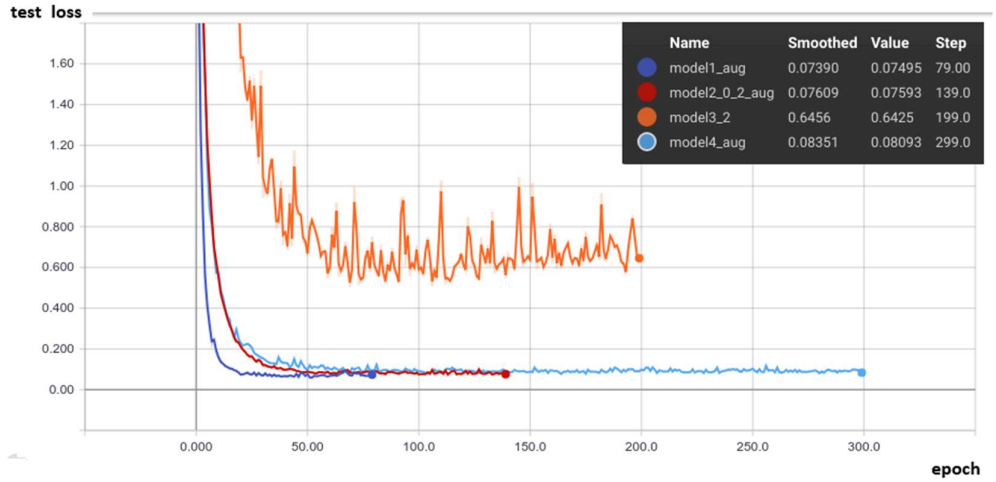

Fig. 4. 26 Test loss of best models

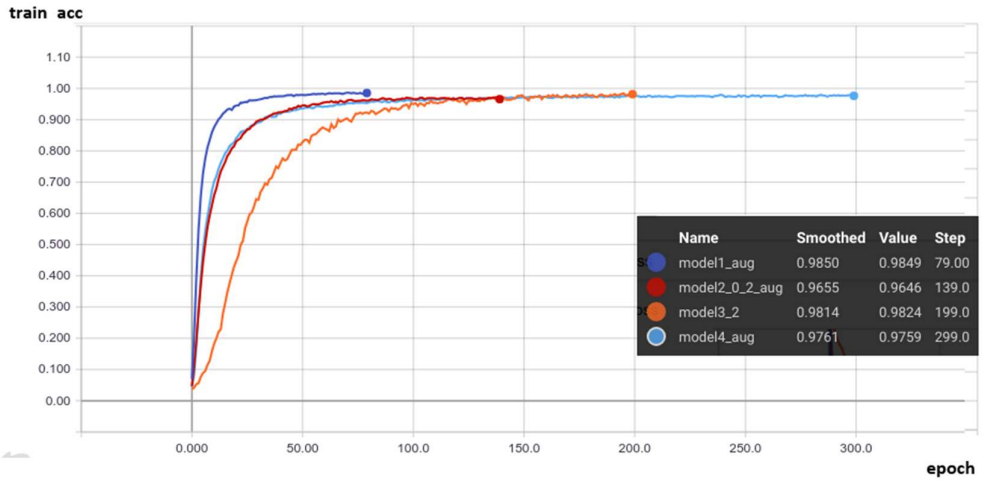

Fig. 4. 27 Train accuracy of best models

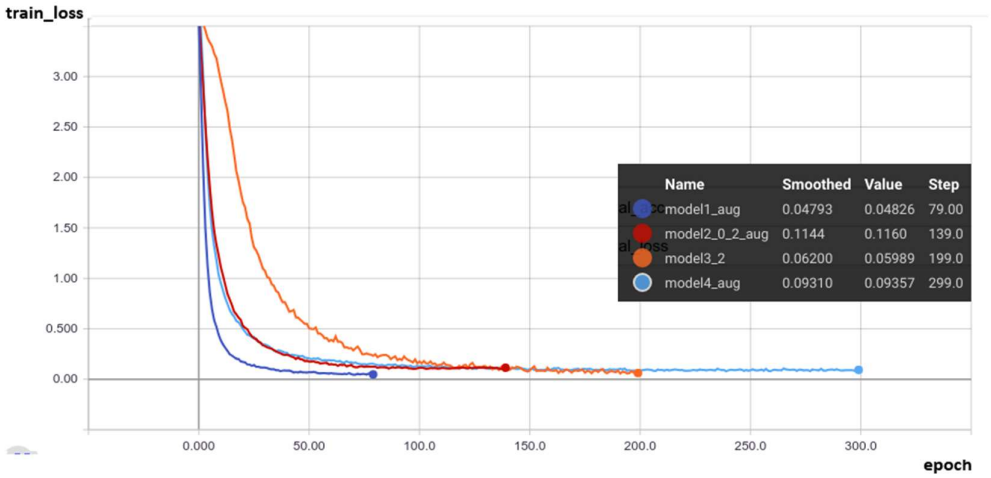

Fig. 4. 28 Train loss of best models

The above architectures had the highest accuracy among all the architectures tried. And on the basis of the test accuracy the first model is chosed from above table (model1_aug) since it had the highest test accuracy of 98.31%.

4.6 Model

This is the best model generated by the Proposed CNN.

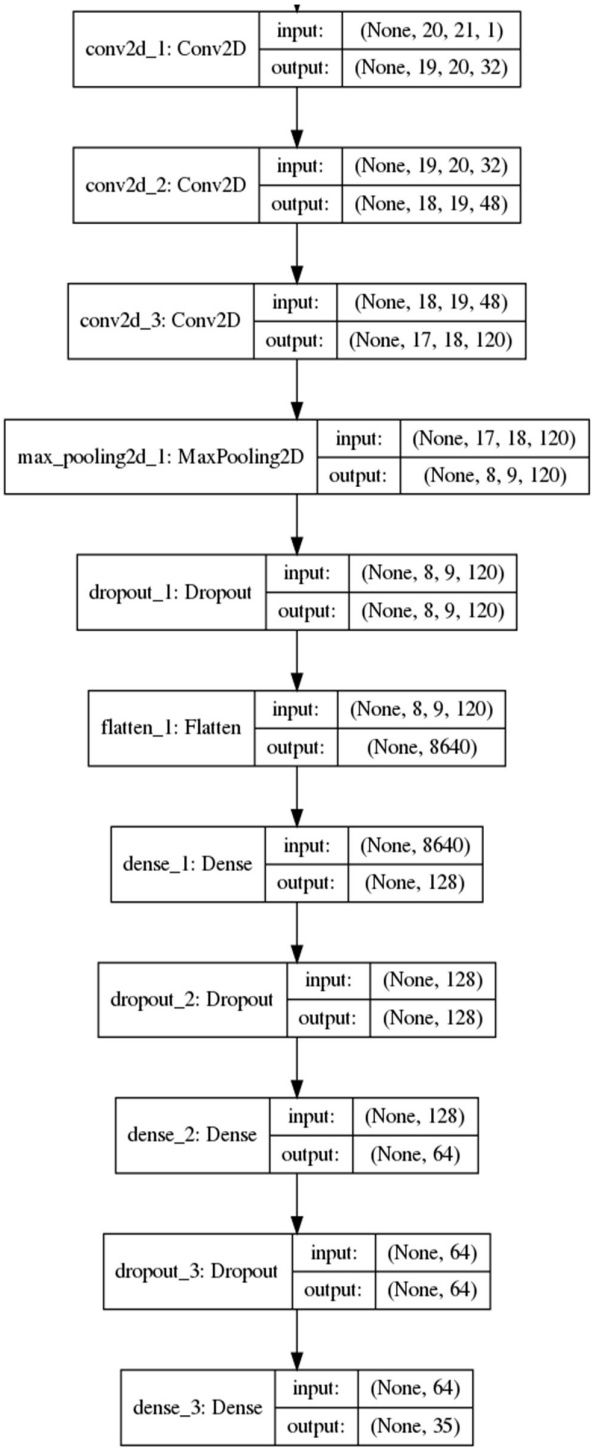

Fig. 4. 29 CNN Model

The CNN model can be parameterized as follows:

- **Layer 1:** Convolution layer with 32 filters, kernel size of 2x2, stride of 1 and ReLU activation function is used where an input shape (20,21,1) is fed. The output will be (19x20x32)
- **Layer 2:** Convolution layer with 48 filters with a kernel size of (2,2), and stride 1. This is followed ReLU activation function. The output of this layer has the shape (18,19,48).
- **Layer 3:** Convolution layer with 120 filters with a kernel size of (2,2), and stride 1. This is followed ReLU activation function. The output of this layer has the shape (17,18,120).
- **Layer 4:** Max-pooling layer of pool size (2,2) and stride size(2,2) which gives the output of shape (8,9,120).
- **Layer 5:** The input from previous layer is then fed to Dropout layer where 25% of the nodes are deactivated. The output from this layer is of shape (8,9,120).
- **Layer 6:** The flatten layer flattens the input into one dimensional array of 8640 inputs.
- **Layer 7:** A fully connected layer (Dense layer) with ReLU activation function and bias gives the output of size 128.
- **Layer 8:** The 128 input is then fed to a Dropout layer where 25% of the nodes are deactivated.
- **Layer 9:** The output from previous layer is fed to another Dense layer with ReLU and bias which gives the output of size 64.
- **Layer 10:** The output from previous layer is fed to the dropout where 40% of the nodes are deactivated.
- **Layer 11:** The input of size 64 is then fed to a final Dense layer with Softmax activation function gives the output size as per the number of labels (35 in this case).

The model is then compiled with Adam optimizer and categorical cross-entropy.

Use of ReLU and Softmax as Activation function and Adam as optimizer.

ReLU is used as an activation functions in hidden layers because the major benefits of ReLUs are sparsity and a reduced likelihood of vanishing gradient. By the definition of a ReLU : $h=\max(0,a)$ where, $a=Wx+b$. One major benefit is the reduced likelihood of the gradient to vanish. This arises when $a>0$. In this regime the gradient has a constant value. In contrast, the gradient of sigmoids becomes increasingly small as the absolute value of x increases. The constant gradient of ReLUs results in faster learning. The other benefit of ReLUs is sparsity. Sparsity arises when $a\leq 0$. The more such units that exist in a layer, the sparser the resulting representation. Sigmoids on the other hand are always likely to generate some non-zero value resulting in dense representations. Sparse representations seem to be more beneficial than dense representations.

Softmax is used at the final layer of the model since it squashes the sparsity of output from previous layer and gives the output between 0 and 1, which is the probability of an output.

Since the input data is sparse thus, optimizing methods such as SGD,NAG and momentum are inferior and perform poorly. For sparse data adaptive learning-rate methods should be used. An additional benefit is that learning rate need not to be set and still likely achieve the best results with the default value. Adam, being the adaptive learning-rate methods is used to optimize the model.

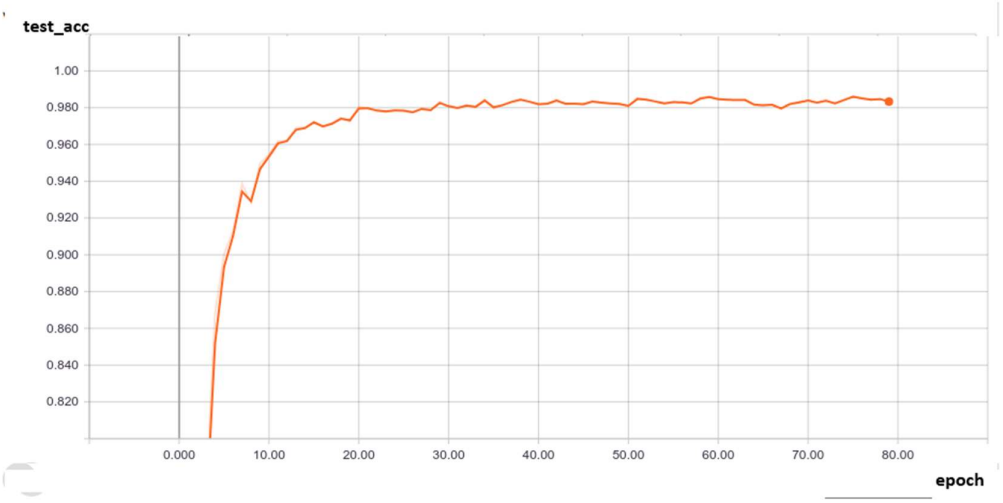

Fig. 4. 30 Test accuracy of model1_aug

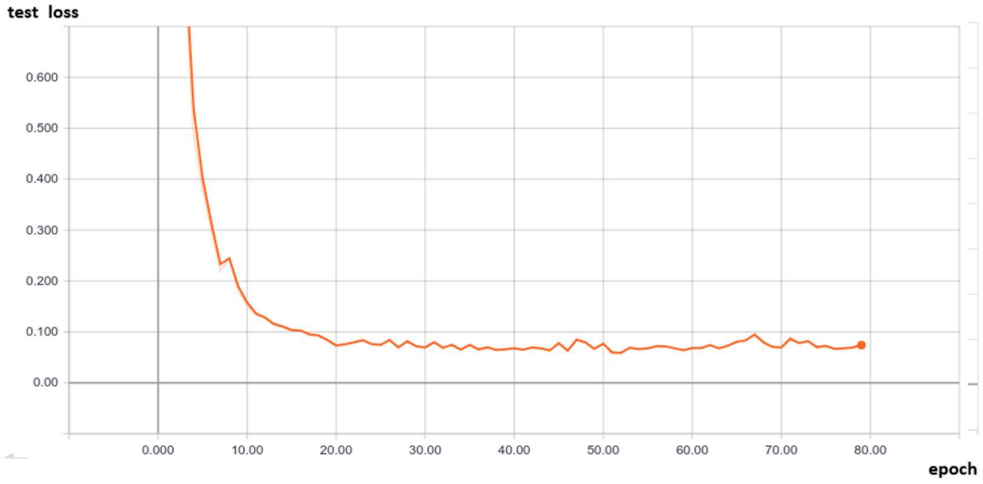

Fig. 4. 31 Test loss of model1_aug

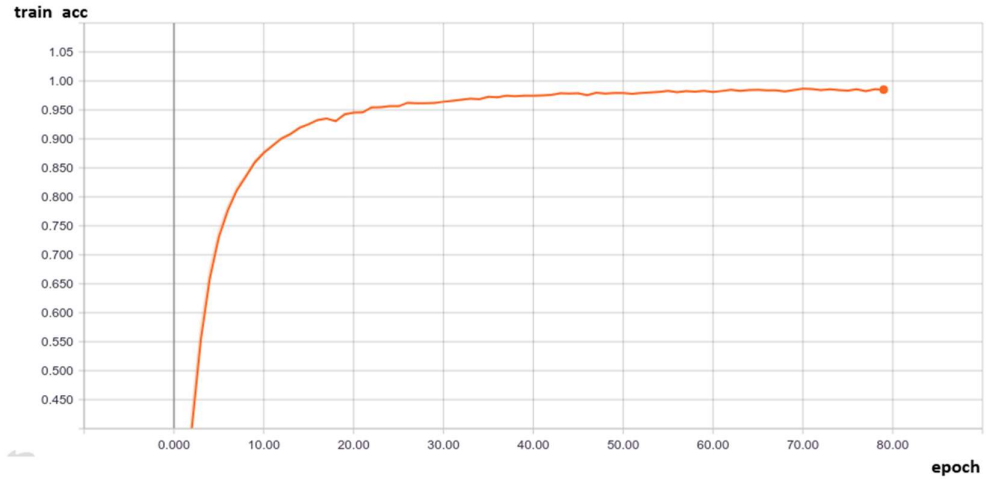

Fig. 4. 32 Train accuracy of model1_aug

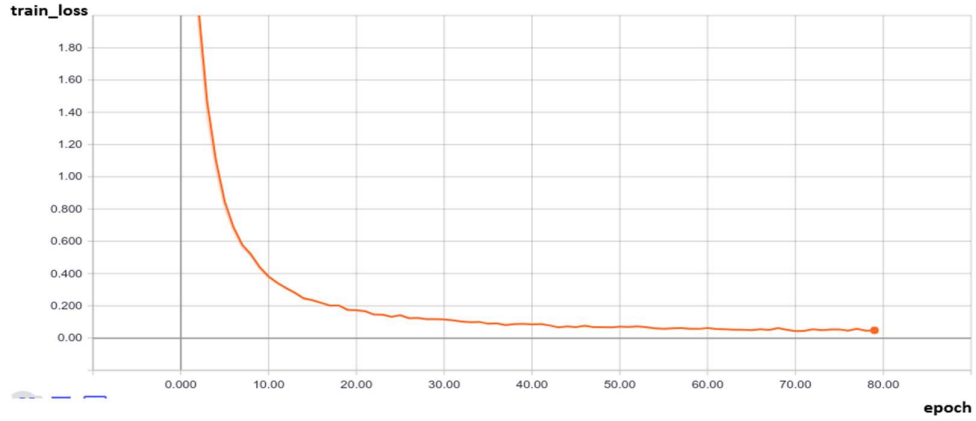

Fig. 4. 33 Train loss of model1_aug

4.7 Text Output

Finally, the alphabet is identified after comparing with the acoustic model and the alphabet is displayed as the result in web interface.

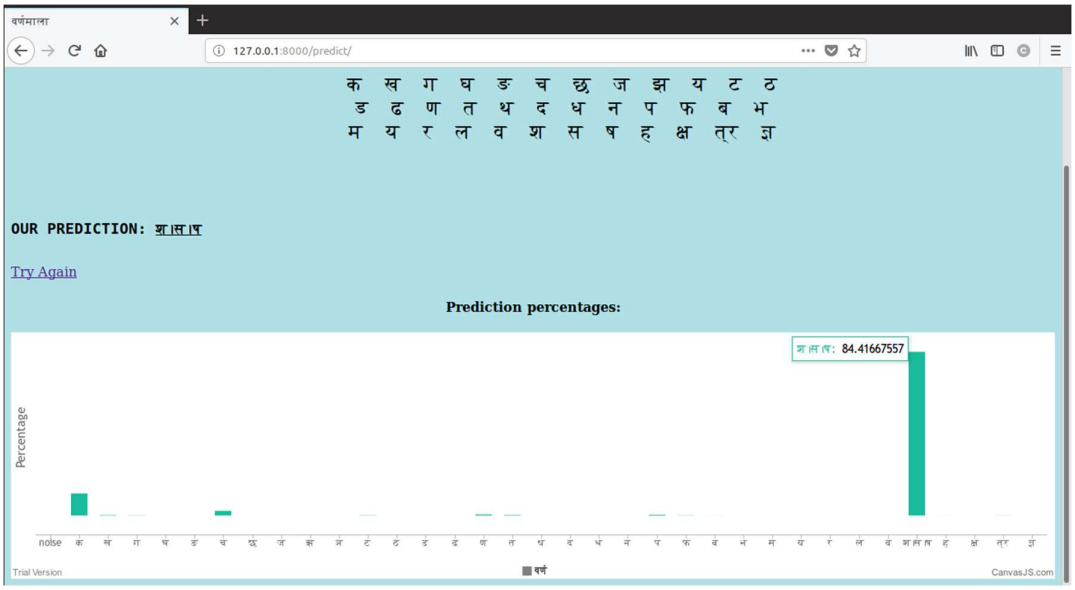

Fig. 4. 34 Predicted output in web interface.

4.8 Libraries and Interpreters Used

4.8.1 Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. [14]

Features of Keras:

- **User friendliness:** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity:** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.
- **Easy extensibility:** New modules are simple to add (as new classes and functions), and existing modules provide sample examples. To be able to easily

create new modules allows for total expressiveness, making Keras suitable for advanced research.

- **Work with Python:** No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

4.8.2 Tensorflow

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. A computation expressed using TensorFlow can be executed with little or no change on a wide variety of heterogeneous systems, ranging from mobile devices such as phones and tablets up to large-scale distributed systems of hundreds of machines and thousands of computational devices such as GPU cards. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields, including speech recognition, computer vision, robotics, information retrieval, natural language processing, geographic information extraction, and computational drug discovery [15].

Tensorflow also provide a visualization tools called TensorBoard. The computations you'll use TensorFlow for - like training a massive deep neural network - can be complex and confusing. TensorBoard makes it easier to understand, debug, and optimize TensorFlow programs. TensorBoard can be used to visualize your TensorFlow graph, plot quantitative metrics about the execution of your graph, and show additional data like images that pass through it. When TensorBoard is fully configured, it looks like this:

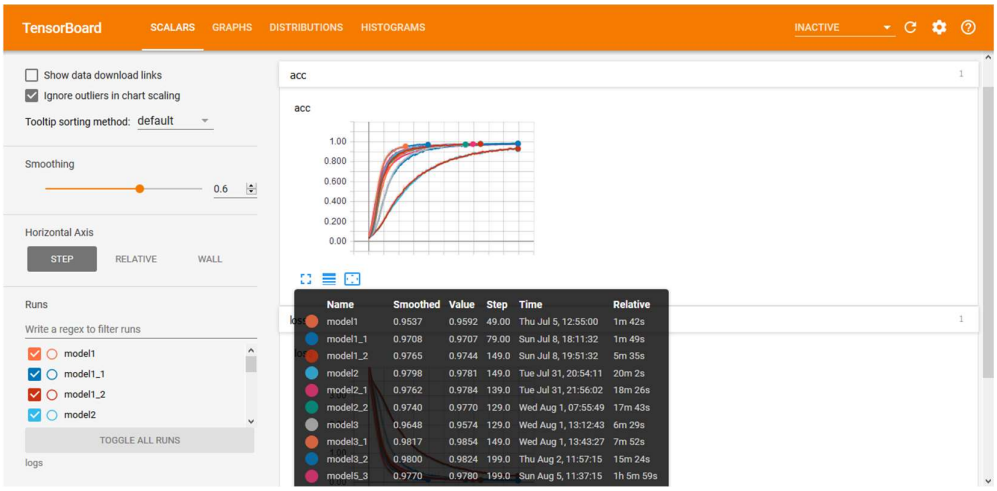

Fig. 4. 35 Tensorboard for the visualization of models

5. RESULTS

After several runs altering different factor that might affect the model, following results were performed at each runs:

Table 5. 1 Various models with their accuracy

Model name	Networks layers (sequential)	Train Data		Test Data	
		Accuracy	Loss	Accuracy	Loss
model 1	convolutional*3,Pooling,(dropout,dense)*3	0.9592	0.1278	0.9134	0.4108
model 1_1	**convolutional*3,Pooling,(dropout,dense)*3**	**0.9707**	**0.08641**	**0.9145**	**0.3845**
model1_aug	**convolutional*3,Pooling,(dropout,dense)*3**	**0.9842**	**0.04826**	**0.9831**	**0.07495**
model1_2	convolutional*3,Pooling,(dropout,dense)*3	0.9744	0.07137	0.8335	0.7936
model2	convolutional*3,Pooling,(dropout,dense)*3	0.9781	0.07323	0.8717	0.6981
model2_0	convolutional*3,Pooling,(dropout,dense)*3	0.9774	0.06161	0.8734	0.6189
model2_0_1	convolutional*3,Pooling,(dropout,dense)*3	0.9839	0.05632	0.8867	0.6632
model2_0_2	**convolutional*3,Pooling,(dropout,dense)*3**	**0.9839**	**0.05347**	**0.8872**	**0.6338**
model2_0_2_ aug	**convolutional*3,Pooling,(dropout,dense)*3**	**0.9646**	**0.116**	**0.9814**	**0.0759**
model2_1	convolutional*3,Pooling,(dropout,dense)*3	0.9776	0.07243	0.8615	0.6501
model2_2	convolutional*3,Pooling,(dropout,dense)*3	0.977	0.07416	0.8792	0.5647
model3	conv,(conv,pooling)*3,(dropout,dense)*3	0.9574	0.1427	0.8418	0.6615
model3A	conv,(conv,pooling)*3,(dropout,dense)*3	0.9747	0.0829	0.8487	0.8288
model3_0	conv,(conv,pooling)*3,(dropout,dense)*3	0.9689	0.1236	0.8521	0.6989
model3_1	conv,(conv,pooling)*3,(dropout,dense)*3	0.9854	0.05629	0.8671	0.6961
model3_2	**conv,(conv,pooling)*3,(dropout,dense)*3**	**0.9824**	**0.05989**	**0.8815**	**0.6425**
model3_aug	**conv,(conv,pooling)*3,(dropout,dense)*3**	**0.978**	**0.0783**	**0.9314**	**0.3236**
model3_3	conv,(conv,pooling)*3,(dropout,dense)*3	0.9835	0.05407	0.8786	0.6209
model4	(conv,pooling)*2,(dropout,dense)*3	0.9451	0.1794	0.8642	0.5154
model4_1	(conv,pooling)*2,(dropout,dense)*3	0.9547	0.156	0.882	0.478
model4_2	(conv,pooling)*2,(dropout,dense)*3	0.959	0.1351	0.8843	0.5096

model4_3	**(conv,pooling)*2,(dropout,dense)*3**	**0.9659**	**0.1259**	**0.8855**	**0.4896**
model4_aug	**(conv,pooling)*2,(dropout,dense)*3**	**0.9759**	**0.09357**	**0.9823**	**0.08093**
model4_4	(conv,pooling)*2,(dropout,dense)*3	0.9716	0.1027	0.8619	0.7086
model5_1	conv,(conv,pooling)*3,(dropout,dense)*3	0.9831	0.0727	0.9653	0.1651
model5_2	**conv,(conv,pooling)*3,(dropout,dense)*3**	**0.9818**	**0.06101**	**0.9678**	**0.1425**

Table 5. 2 Models with varied hyperparameters.

Model name	**Input Dimension**	**Epochs**	**Batch size**	**Samples**		**Kernel size**	**Optimizer**	**Activation**	**Network classifier**	**Bias**
				Train	**Test**					
model 1	20*11*1	50	100	480	320	2*2	Adadelta	Relu	softmax	off
model 1_1	**20*11*1**	**80**	**100**	**480**	**320**	**2*2**	**Adadelta**	**Relu**	**softmax**	**off**
model1_ aug	**20*21*1**	**80**	**100**	**9933**	**6622**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model1_2	20*11*1	150	100	900	600	2*2	Adadelta	Relu	softmax	off
model2	20*21*1	150	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model2_0	20*21*1	140	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model2_0 _1	20*21*1	140	50	2607	1738	2*2	Adadelta	Relu	softmax	off
model2_ 0_2	**20*21*1**	**140**	**50**	**2607**	**1738**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model2_ 0_2_aug	**20*21*1**	**140**	**50**	**9933**	**6622**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model2_1	20*21*1	140	50	2172	2173	2*2	Adadelta	Relu	softmax	off
model2_2	20*21*1	130	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model3	20*21*1	130	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model3A	20*21*1	130	50	2607	1738	2*2	Adadelta	Relu	softmax	off
model3_0	20*21*1	130	50	2607	1738	2*2	Adadelta	Relu	softmax	on
model3_1	20*21*1	150	50	2607	1738	2*2	Adadelta	Relu	softmax	off

model3_2	**20*21*1**	**200**	**100**	**2607**	**1738**	**2*2**	Adadelta	**Relu**	**softmax**	**on**
model3_aug	**20*21*1**	**200**	**100**	**11434**	**7623**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model3_3	20*21*1	200	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model4	20*21*1	200	100	2607	1738	2*2	Adadelta	Relu	softmax	off
model4_1	20*21*1	200	100	2607	1738	2*2	Adadelta	Relu	softmax	on
model4_2	20*21*1	250	100	2607	1738	2*2	Adadelta	Relu	softmax	on
model4_3	**20*21*1**	**300**	**100**	**2607**	**1738**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model4_aug	**20*21*1**	**300**	**100**	**9933**	**6622**	**2*2**	**Adadelta**	**Relu**	**softmax**	**on**
model4_4	20*21*1	300	50	2607	1738	2*2	Adadelta	Relu	softmax	on

The effect of data augmentation can be seen below:

For model3 with conv,(conv,pooling)*3,(dropout,dense)*3 , epoch= 200 and batch size=100 4345, samples were augmented to make total 19057 data

Training Data		**Test Data**		**Data Type**
Accuracy	**Loss**	**Accuracy**	**Loss**	
0.9824	0.05989	0.8815	0.6425	Unaugmented
0.978	0.0783	0.9314	0.3236	Augmented

Table 5. 3 Accuracy Comparison of model3

For model2 with conv,(conv,pooling)*3,(dropout,dense)*3 , epoch= 140 and batch size=80, 4345 samples were augmented to make total 16555 data

Table 5. 4 Accuracy Comparison of model2

Training Data		**Test Data**		**Data Type**
Accuracy	**Loss**	**Accuracy**	**Loss**	
0.9839	0.05347	0.8872	0.6338	Unaugmented
0.9646	0.116	0.9814	0.0759	Augmented

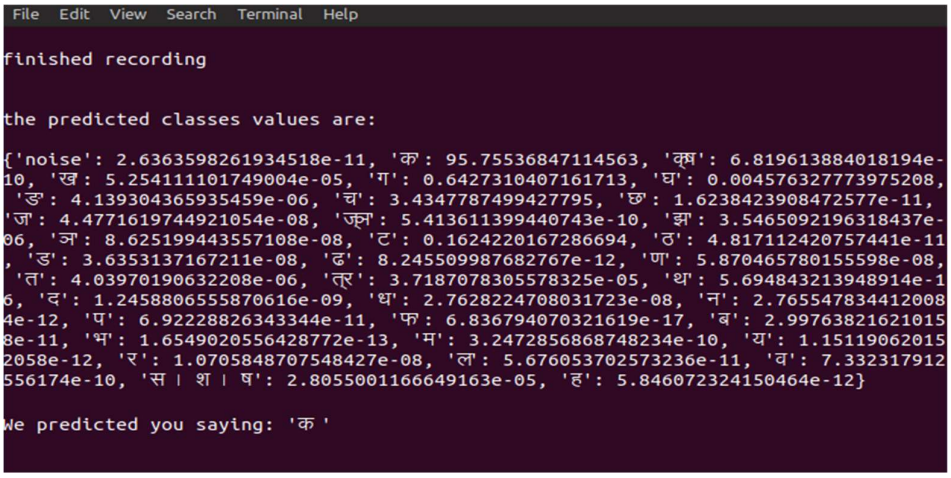

Fig. 5. 1 Screenshot of correct prediction of “ka”

By selecting the best model, the model is implemented in a django framework and the results of speech recognition can be seen below:

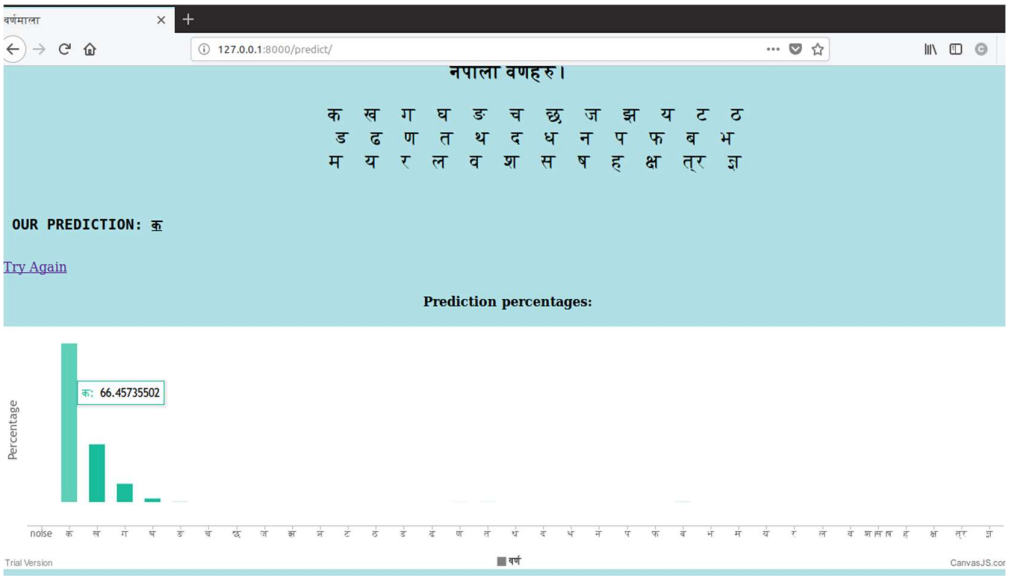

Fig. 5. 2 Screenshot of correct prediction of “ka” in web.

6. DISCUSSION

During the project, we spent most of our time trying out different methods of speech recognition and found out use of CNN would be an efficient implementation. We started making datasets with the sampling rate of 16000 Hz because it provides more accurate high frequency information. For most ASR applications, sampling rates higher than about 22kHz is a waste. While researching about various speech recognition attempts over the web, we found that the most of the datasets were recorded in mono channel and found out mono is sufficient because of less size and efficient computation.

Prior to the introduction of Mel-frequency cepstral coefficients (MFCCs), Linear Prediction Coefficients (LPCs) and Linear Prediction Cepstral Coefficients (LPCCs) were the main feature type for automatic speech recognition (ASR), especially with HMM classifiers. To extract a feature vector containing all information about the linguistic message, MFCC mimics some parts of the human speech production and speech perception. MFCC mimics the logarithmic perception of loudness and pitch of human auditory system and tries to eliminate speaker dependent characteristics by excluding the fundamental frequency and their harmonics. Hence MFCC is used.

Then we fed the data to the CNN model. The CNN architecture we proposed uses ReLU and Softmax Activation function and Adam/ Adadelta Optimizer. Since there is no perfect model, or a fixed algorithm or a method to build a model. We built 26 models and selected the model with the best accuracy.

Data augmentation was necessary as the dataset we had was very small because we had to record it ourselves. So because of the small dataset, the accuracy we got was less than 70%. After augmentation, we quadrupled our dataset which lead to a massive increase in accuracy i.e. more than 90%. The data augmentation we applied at first included pitch shifting, speed change, dynamic range compression and noise addition where the changes in variables were randomly generated which led us to increase in accuracy in test process but difficulty to detect new voices. Hence , we put a predefined variables changes and implemented the data augmentation only by doing pitch shifting, change in dynamic range and noise addition which not only

increases the test accuracy but also gives accurate result when testing with new voices.

7. CONCLUSION

Since this project primarily focuses on implementation of speech recognition specifically for Nepali alphabets, we successfully implemented a Convolutional Neural Network and trained it to detect the Nepali alphabets. With limited amount of dataset and not so quality recording materials we were able to recognize much of the alphabets in an isolated environment whereas the efficiency decreased with the increases in environmental noises. During the project we also found the importance of data augmentation which we discussed on previous chapter.

8. LIMITATIONS AND FUTURE ENHANCEMENTS

Although we managed to implement the basic backbone to Nepali Speech Recognition, the Speech recognition model we developed hasn't been trained on enough datasets since in real life the way a person says "ka" differs from one person to another. The amount of dataset used for training also increases the confidence of the model to predict the correct alphabet regardless of the higher noise environment. The project could be further improved to detect continuous voice and with the application of next alphabet prediction algorithm we could implement the model to detect Nepali words rather than just alphabets.

Also some Nepali alphabets pronunciation are fairly similar, i.e. Ta and ta, Da and da, the three sa's. It's difficult to differentiate between these alphabets.

In future, the model and the recording process can be modified to extent the use case of our Speech Recognition Application to detect continuous speech. Furthermore, the dataset collection can be widen beyond four of us by the implementation of dedicated voice collection web interface where a user can contribute their voice.

REFERENCES

[1] R. B. a. S. B. K. H. Davis, "Automatic Recognition of Spoken Digits," *J. Acoust. Soc. Am,* vol. 24, no. 6, pp. 627-642, 2004.

[2] J. S. a. K. Nakata, "Recognition of Japanese Vowels—Preliminary to the Recognition of Speech," *J. Radio Res. Lab,* vol. 37, no. 8, pp. 193-212, 1961.

[3] J. S. a. S. Doshita, The Phonetic Typewriter, Information Processing 1962, Munich: Proc. IFIP Congress, 1962.

[4] Y. K. a. S. C. K. Nagata, "Spoken Digit Recognizer for Japanese Language," *NEC Res. Develop,* no. 6, 1963.

[5] B. J. a. L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology," p. 6, 2004.

[6] M. Rouse, "TechTarget," [Online]. Available: http://searchcrm.techtarget.com/definition/virtual-assistant. [Accessed 25 December 2017].

[7] I. &. B. Y. &. M. W. &. L. J.-P. Rebai, "Improving speech recognition using data augmentation and acoustic model fusion," *j.procs.2017.08.003.,* vol. 112, pp. 316-322, 2017.

[8] "practicalcryptography," [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Accessed 06 03 2018].

[9] cs231n. [Online]. Available: http://cs231n.github.io/convolutional-networks.

[10] [Online]. Available: https://github.com/rasbt/python-machine-learning-book/blob/master/faq/dropout.md.

[11] [Online]. Available: https://medium.com/@uniqtech/understand-the-softmax-function-in-minutes-f3a59641e86d.

[12] J. Jordan, "Jeremy Jordan," 1 March 2018. [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/.

[13] A. S. Walia, "Towards Data Science," 10 june 2017. [Online]. Available: https://towardsdatascience.com/types-of-optimization-algorithms-used-in-neural-networks-and-ways-to-optimize-gradient-95ae5d39529f.

[14] F. a. o. Chollet, "Keras Documentation," 2015. [Online]. Available: https://keras.io/.

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo,Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis,Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,, "Tensorflow," 2015. [Online]. Available: https://www.tensorflow.org.

APPENDIX

Screenshots

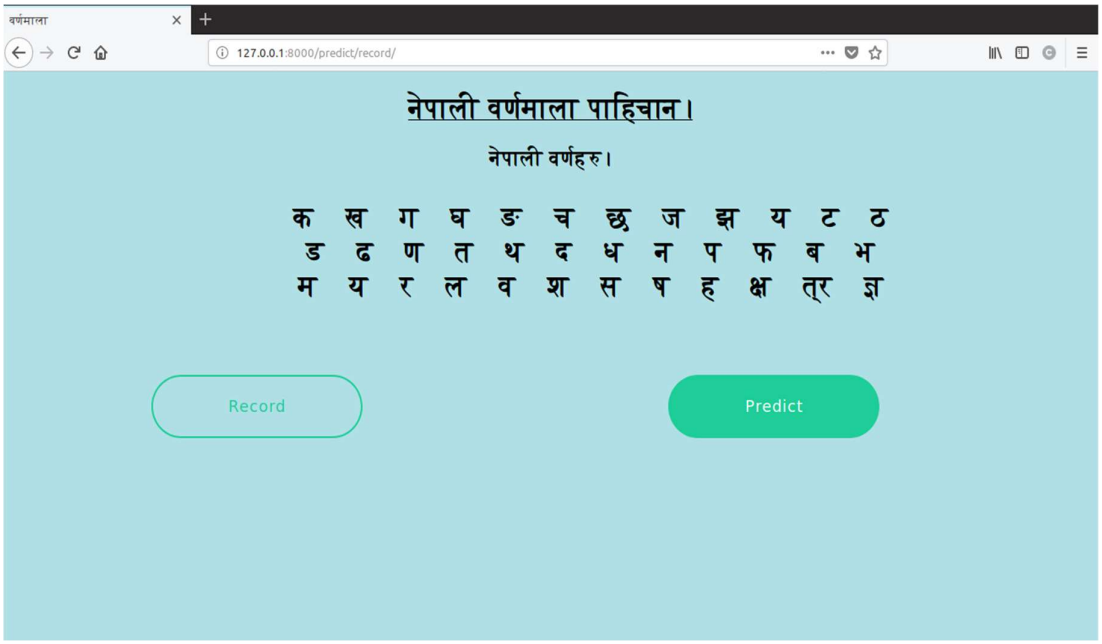

Fig: Homepage

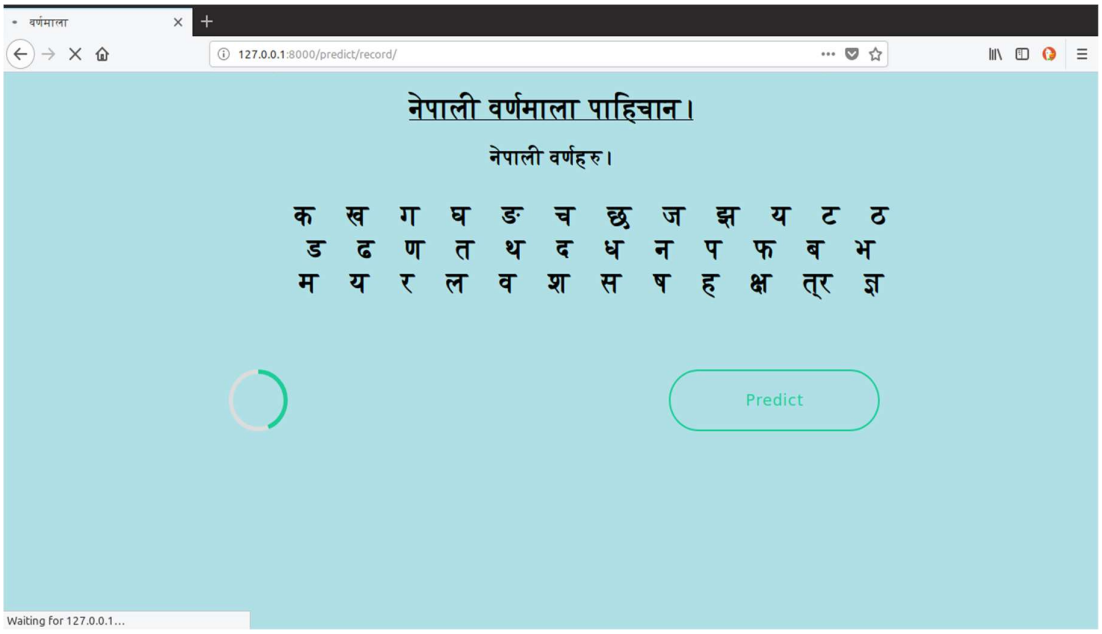

Fig: Record

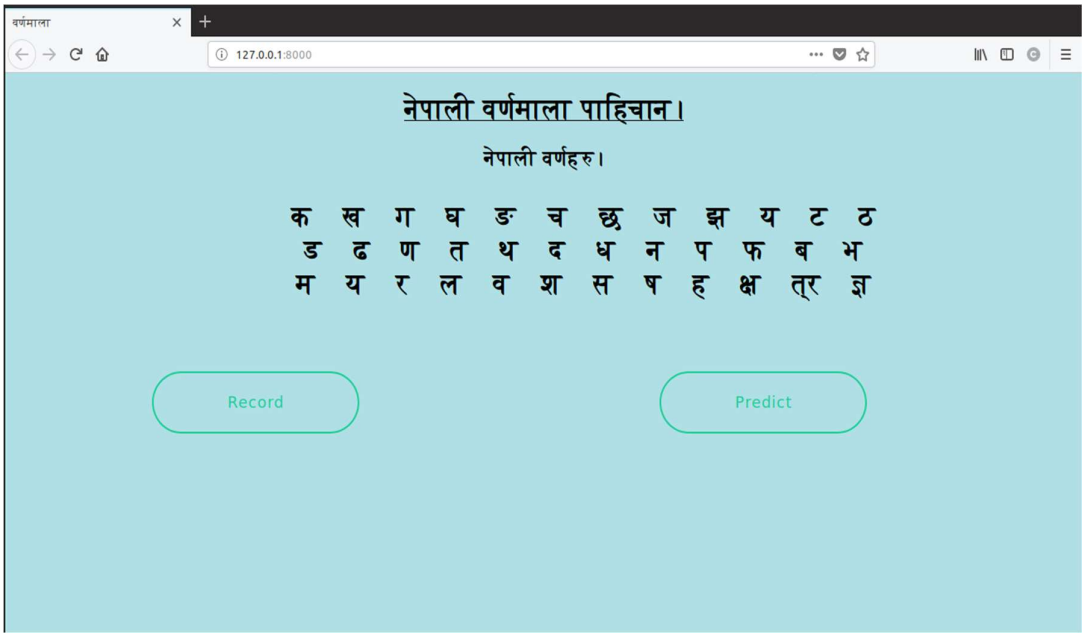

Fig: Predict

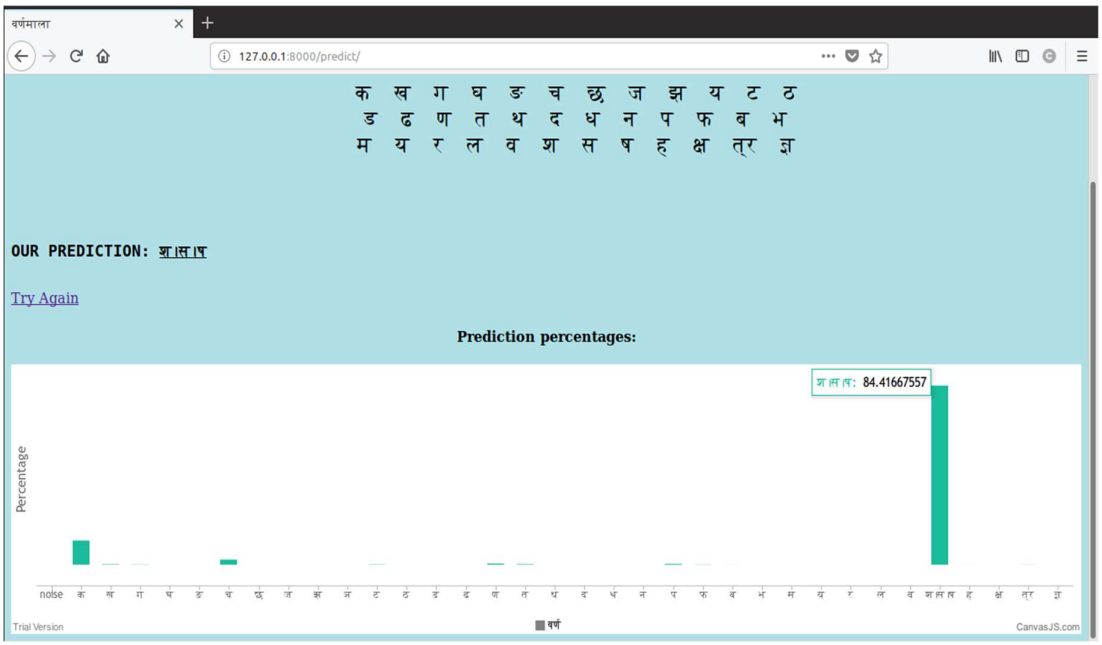

Fig: Output of correct prediction.