INTEGRATED SYSTEM FOR INTER-CONVERSION OF SPEECH AND INDIAN SIGN LANGUAGE

by

Zainab Feroz	**BE/10006/2014**
Nishant Mishra	**BE/10102/2014**
Prasun Anand	**BE/10158/2014**

Submitted in partial fulfillment of

Requirement for the award in the degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

Supervised by

Dr. MAHESH CHANDRA

Professor

Department of Electronics and Communication Engineering

Department of Electronics and Communication Engineering

Birla Institute of Technology, Mesra, Ranchi – 835215

May 2018

DECLARATION CERTIFICATE

This is to certify that the work presented in this project entitled "**Integrated system for Inter-conversion of speech and Indian Sign Language**", in partial fulfillment of the requirement for the award of Degree of **Bachelor of Engineering in Electronics & Communication Engineering**, submitted to the Department of Electronics & Communication Engineering of Birla Institute of Technology, Mesra, Ranchi, Jharkhand is a bona fide work carried out by **Zainab Feroz, Nishant Mishra** and **Prasun Anand** under my supervision and guidance.

To the best of my knowledge, the content of this project, either partially or fully, has not been submitted to any other institution for the award of any other degree.

Date:

Dr. Mahesh Chandra
Department of ECE
Birla Institute of Technology
Mesra, Ranchi.

Department of Electronics and Communication Engineering

Birla Institute of Technology, Mesra, Ranchi – 835215

CERTIFICATE OF APPROVAL

This is to certify that the project entitled "**Integrated system for Inter-conversion of speech and Indian Sign Language**" is hereby approved as a suitable design of an engineering subject, carried out and presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the project for which it is submitted.

Internal Examiner

Date:

External Examiner

Date:

Dr. S. Pal
Head of Department
Department of Electronics and Communication Engineering
Birla Institute of Technology
Mesra, Ranchi

ACKNOWLEDGEMENT

We owe our deepest gratitude to our advisor **Dr. Mahesh Chandra**, for his constant support and motivation, despite his extremely busy schedule. Our interactions with him always resulted in new ideas and proved beneficial towards our work. Without his constant presence and supervision our work would not have been successful.

We are very grateful to **Dr. S. Pal, Professor and Head, Department of Electronics and Communication Engineering, Birla Institute of Technology, Mesra** for extending all the facilities at all times for pursing this course.

And lastly our batch mates who have always been there with valuable suggestions and support in our endeavors.

Zainab Feroz

Nishant Mishra

Prasun Anand

ABSTRACT

Day-to-day Communication with verbally and hearing impaired people has been very limited due to lack of knowledge of sign languages among the general public. It is compounded due to the multiple systems of sign language going around, with no universally accepted standard. There is a need to bridge this communication gap in order to bring the specially abled into the mainstream. This project aims at the realisation of a system for interconversion of Indian sign language and speech. The system will be able to capture images of people gesturing various signs and return speech equivalents and vice versa. It aims to leverage computer vision and Deep learning algorithms to be able to learn features in order to distinguish between the different sign images and speech files.

Sign language is a system of communication using visual gestures and signs, as used by deaf and dumb people.

There are various types of sign language like ISL (Indian Sign Language), ASL (American Sign Language), BSL (British Sign Language), etc. We have created an integrated system that converts Indian Sign Language into Speech and vice versa. We have first captured and recognized speech using neural networks and converted into visual representations of its sign language.

Along the course of the project, we built and tested our systems for 2D sign images, RGB and depth images obtained using Microsoft Kinect sensor, on hindi digits speech data with varying levels of noise. We were able to train a gesture to speech system for 48 words (10 numbers, alphabets, common words) with a train/test accuracy of 97%/71%. We also trained a hindi digit speech recognition system that performed with an accuracy of 96%.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Sign language is a system of communication used by deaf and dumb people. There are various types of sign language like ISL (Indian Sign Language), ASL (American Sign Language), BSL (British Sign Language), etc. But none of the sign languages are universal or international. It is impossible to communicate with the specially abled people without sign language. Since most people don't know the sign language, there is a need for a medium that enables real time communication with impaired people.

Sign languages use manual communication to convey meaning. This can include simultaneously employing hand gestures, movement, orientation of the fingers, arms or body, and facial expressions to convey the ideas of a speaker. Sign languages have a lot of similarity to their spoken language, such as American Sign Language (ASL) with American English. Wherever communities of deaf people exist, sign languages have developed, and are at the cores of local deaf cultures. Although signing is used primarily by the deaf and hard of hearing, it is also used by hearing individuals, such as those unable to physically speak, or those who have trouble with spoken language due to a disability or condition.

Indian Sign Language (ISL) is the predominant sign language in South Asia, used by at least several hundred specially abled people. ISL alphabets derived from British Sign Language and French Sign Language alphabets. Unlike its American counterpart which uses one hand, ISL uses both hands to represent alphabets.

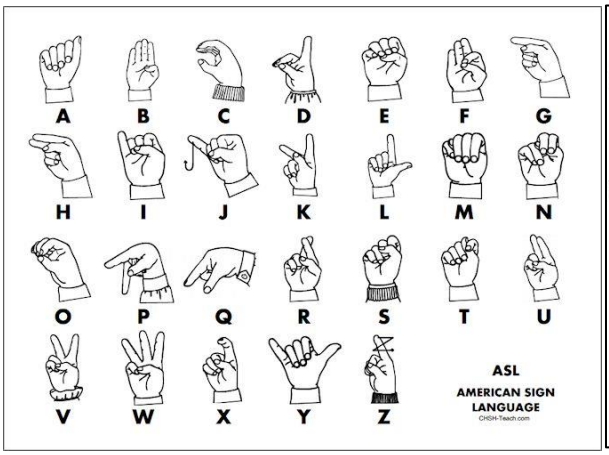


Fig 1: American Sign Language

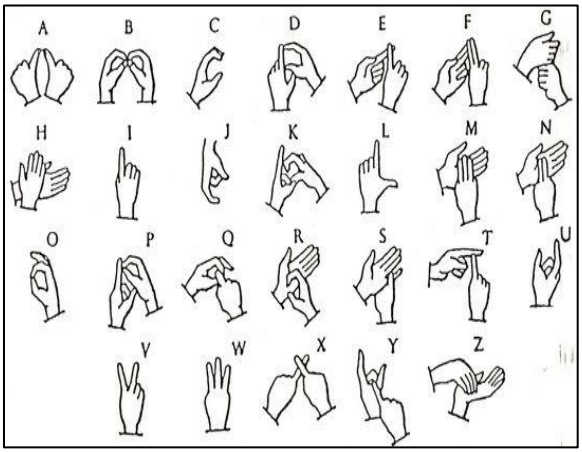


Fig 2: Indian Sign Language

PROCESS FLOW

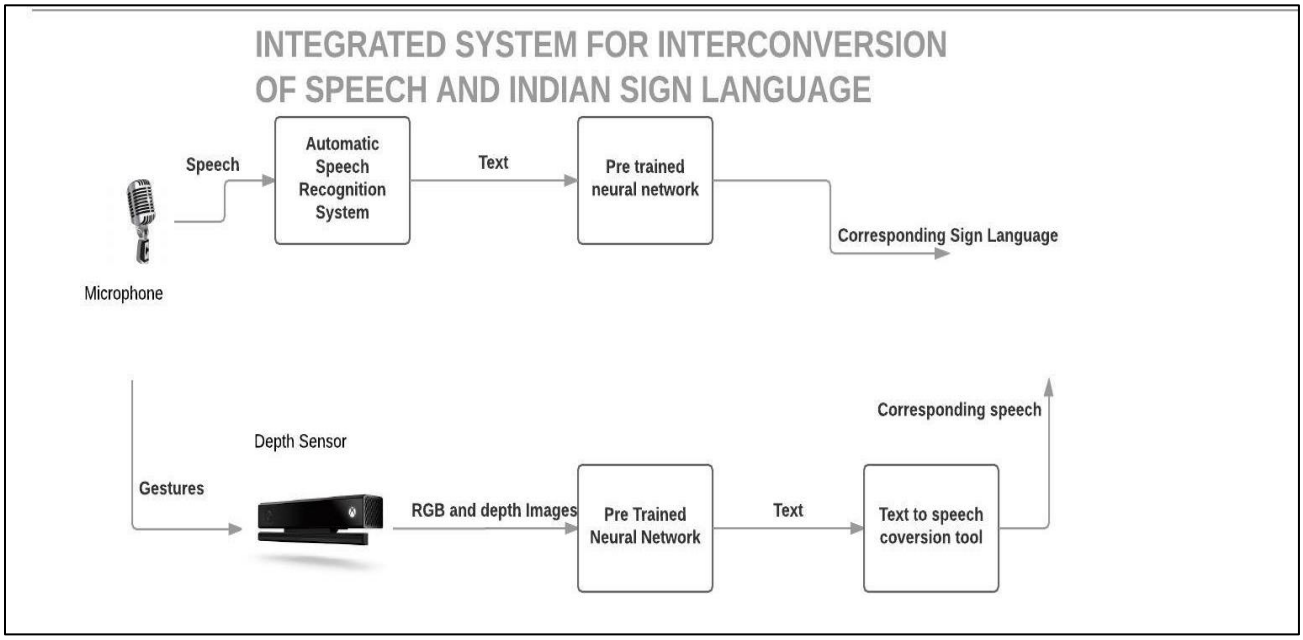
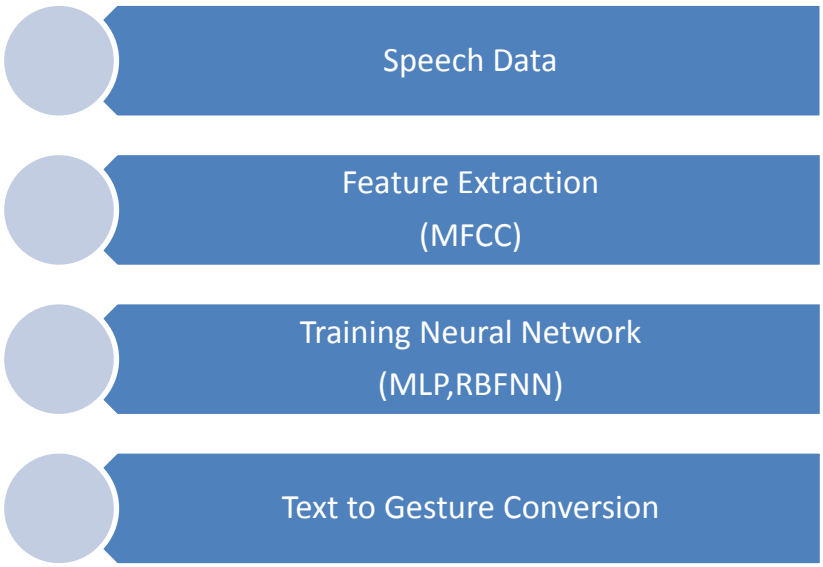


Fig 3: Process Flow

SPEECH TO GESTURE



SPEECH RECOGNITION

Automatic speech recognition consists of identifying what the speaker says based on the utterance. It can be classified into two types: Text Dependent Speech Recognition and Text Independent Speech Recognition. In text dependent systems, we use a predefined utterance to train as well as test the system. Text Independent systems have no constraint on their speech content. The utterances used for testing are independent of those used for training. The basic approach for designing a text dependent speech recognition system includes preparing a suitable database for training and testing the system, extracting features from the different speech samples, followed by the feature classification step.

The two most common techniques for feature extraction are LPC – Linear Predictive Coding and MFCC- Mel-Frequency Cepstral Coefficients. LPC parameters depend on speech production and are a linear combination of past values while MFCC depends on human hearing perception. Feature extraction is an important step which gives the specific information contained in the speech signal. These features depend on various parameters such as intensity, frequency, zero crossing rate, level crossing rate, etc. and also on the age of the speaker, gender, accent, speaking rate, dimensions of the vocal tract and environmental conditions.

Feature classification includes dividing the data in the category which it belongs to. The various models/algorithms that can be used for classification include Hidden Markov Model, Gaussian Mixture Model, Self Organising Maps, Neural Networks, etc.

SPEECH DATASET

The database we used for training and testing our automatic speaker recognition system included Hindi Digit (0-9) samples by 50 different speakers. Each digit was spoken 10 times, thus making the number of utterances by each speaker, 100 and the total number of utterances, 5000. The database consisted of clean data and noisy data with the noise levels varying from -5dB, 0dB, 5dB, 10dB, 20dB and 30dB.

MFCC FEATURE EXTRACTION

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.

STEPS

1. **Pre-emphasis -** The speech signal s(n) is sent to a high-pass filter:

   $s_2(n) = s(n) - a*s(n-1)$

   where $s_2(n)$ is the output signal and the value of a is usually between 0.9 and 1.0.

   The z-transform of the filter is : $H(z)=1-a*z^{-1}$

   The goal of pre-emphasis is to compensate the high-frequency part that was suppressed during the sound production mechanism of humans. Moreover, it can also amplify the importance of high-frequency formants.

2. **Framing -** The input speech signal is segmented into frames of 20~30 ms with optional overlap of 1/3~1/2 of the frame size. The speech data is considered stationary for the duration of the frame. If the frame size is too small, we won't be able to extract useful features, if the size is too large, the speech signal will be non- stationary.
3. **Windowing -** Each frame has to be multiplied with a hamming window in order to keep the continuity of the first and the last points in the frame. If the signal in a frame is denoted by s(n), n = 0,…N-1, then the signal after Hamming windowing is s(n)*w(n), where w(n) is the Hamming window defined by:

   $w(n, a) = (1 - a) - a \cos(2pn/(N-1)), \quad 0 \leqq n \leqq N-1$
4. **Fast Fourier Transform or FFT** - Spectral analysis shows that different timbres in speech signals corresponds to different energy distribution over frequencies. Therefore we usually perform FFT to obtain the magnitude frequency response of each frame.

   When we perform FFT on a frame, we assume that the signal within a frame is periodic, and continuous when wrapping around.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i 2\pi k n / N} \qquad k = 0, \ldots, N-1.$$

5. **Mel Filter Bank -** In this step, the above calculated spectrums are mapped on Mel scale to know the approximation about the existing energy at each spot with the help of Triangular overlapping window also known as triangular filter bank. These filter bank is a set of band pass filters having spacing along with bandwidth decided by steady Mel frequency time. During the mapping, when a given frequency value is up to 1000Hz the Mel-frequency scaling is linear frequency spacing, but after 1000Hz the spacing is logarithmic. The formula to convert frequency f hertz into Mel mf is given by Eq

$$m_f = 2595 log_{10} \left(\frac{f}{700} + 1\right)$$

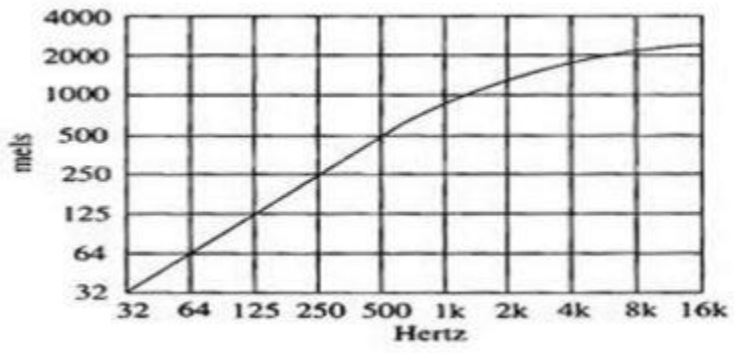


Fig 4: Mel Scale

6. **Discrete Cosine Transform or DCT -** This process of carrying out DCT is done in order to convert the log Mel spectrum back into the spatial domain. For this transformation either DFT or DCT both can be used for calculating Coefficients from the given log Mel spectrum as they divide a given sequence of finite length data into discrete vector. The output after applying DCT is known as MFCC (Mel Frequency Cepstrum Coefficient

$$C_n = \sum_{k-1}^{k} (\log D_k) \cos\left[m\left(k - \frac{1}{2}\right)\frac{\pi}{k}\right]$$

where m = 0, 1… k- 1

where Cn represents the MFCC and m is the number of the coefficients here m=13 so, total number of coefficients extracted from each frame is 13.

Thus, with the help of Filter bank with proper spacing done by Mel scaling it becomes easy to get the estimation about the energies at each spot and once this energies are estimated then the log of these energies also known as Mel spectrum can be used for calculating first 13 coefficients using DCT. Since, the increasing numbers of coefficients represent faster change in the estimated energies and thus have less information to be used for classifying the given images. Hence, first 13 coefficients are calculated using DCT and higher are discarded which give a sequence of acoustic feature vectors.

NEURAL NETWORKS

WHAT IS A NEURAL NETWORK?

An artificial neural network is a programmed computational model that aims to replicate the neural structure and functioning of the human brain.ANNs are processing devices (algorithms or actual hardware) that are loosely modeled after the neuronal structure of the mamalian cerebral cortex but on much smaller scales. A large ANN might have hundreds or thousands of processor units, whereas a mamalian brain has billions of neurons with a corresponding increase in magnitude of their overall interaction and emergent behavior.

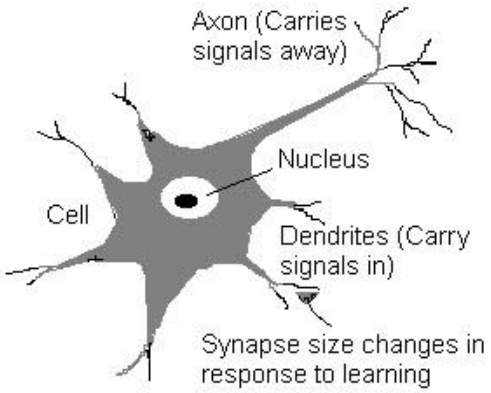


Fig 5: Neuron

ARCHITECTURE OF NEURAL NETWORKS

1. Feed-forward networks

Feed-forward ANNs (figure 1) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

2. Feedback networks

Feedback networks can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

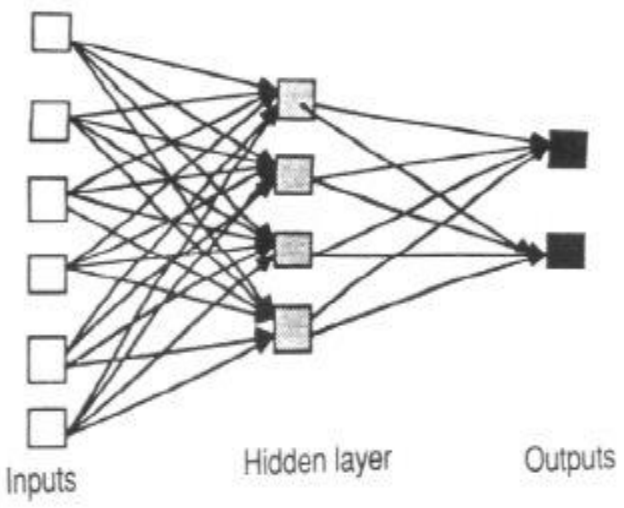


Fig 6 : An example of a simple feedforward network

3. Network layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of **"output**" units. (see Figure 4.1)

- The activity of the input units represents the raw information that is fed into the network.

- The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

- The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

We also distinguish single-layer and multi-layer architectures. The single-layer organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organisations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

4. Perceptrons

The perceptron turns out to be an MCP model (neuron with weighted inputs) with some additional, fixed, preprocessing. Units labelled A1, A2, Aj , Ap are called association units and their task is to extract specific, localised features from the input images. Perceptrons mimic the basic idea behind the mammalian visual system. They were mainly used in pattern recognition even though their capabilities extended a lot more.

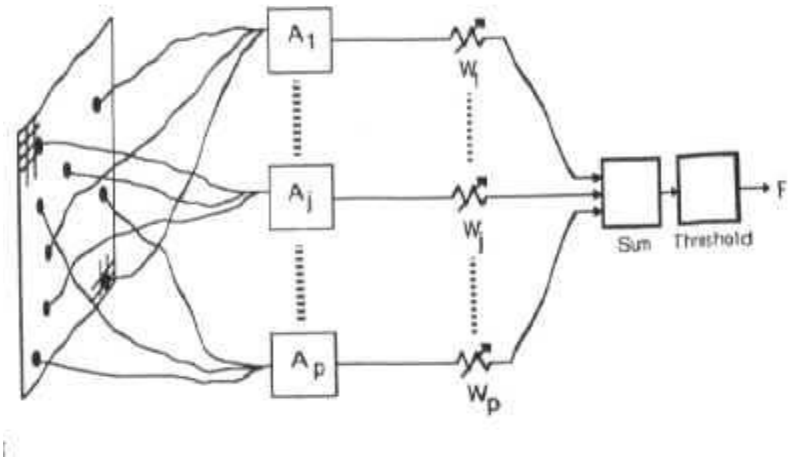


Fig 7: Perceptron

TYPES OF NEURAL NETWORKS

1. FEEDFORWARD NEURAL NETWORK

The simplest of all neural networks, the feedforward neural network, moves information in one direction only. Data moves from the input nodes to the output nodes, passing through hidden nodes (if any). The feedforward neural network has no cycles or loops in its network.

2. RADIAL BASIS FUNCTION NEURAL NETWORK

The RBF neural network is the first choice when interpolating in a multidimensional space. The RBF neural network is a highly intuitive neural network. Each neuron in the RBF neural network stores an example from the training set as a "prototype". Linearity involved in the functioning of this neural network offers RBF the advantage of not suffering from local minima.

3. KOHONEN SELF-ORGANIZING NEURAL NETWORK

Invented by Teuvo Kohonen, the self-organizing neural network is ideal for the visualization of low-dimensional views of high-dimensional data. The self-organizing neural network is different from other neural networks and applies competitive learning to a set of input data, as opposed to error-correction learning applied by other neural networks.

The Kohonen self-organizing neural network is known for performing functions on unlabeled data to describe hidden structures in it.

4. RECURRENT NEURAL NETWORK

The recurrent neural network, unlike the feedforward neural network, is a neural network that allows for a bi-directional flow of data. The network between the connected units forms a directed cycle. Such a network allows for dynamic temporal behavior to be exhibited. The recurrent neural network is capable of using its internal memory to process arbitrary sequence of inputs. This neural network is a popular choice for tasks such as handwriting and speech recognition.

5. MODULAR NEURAL NETWORKS

This interesting neural network comprises of a series of independent neural networks that are moderated by an intermediary. Each of these independent neural networks works with separate inputs, accomplishing subtasks that make up the task the network as whole hopes to perform. The intermediary accepts the inputs of each of these individual neural networks, processes them, and creates the final output for the modular neural network. The independent neural networks do not interact with each other.

6. PHYSICAL NEURAL NETWORK

This neural network aims to emphasize the reliance on physical hardware as opposed to software alone when simulating a neural network. An electrically adjustable resistance material is used for emulating the function of a neural synapse. While the physical hardware emulates the neurons, the software emulates the neural network.

NEURAL NETWORKS USED FOR CLASSIFICATION

We have used the following neural networks for Speaker Recognition and provided a comparative study of their performance :

A. Single Layer Feedforward Network (SLFN)

In this architecture there are three layers of neurons. Each neuron applies a nonlinear activation function to its input to generate its output except the neuron in the first layer which just acts as interface between the network and its environment. The output of the last layer is compared to that of required result and the error between them is obtained. This error is then reduced by using backpropagation iteratively on the training data. The trained model is used to classify unseen data to its proper class.

B. Probabilistic Neural Network (PNN)

A PNN learns by approximating the probability distribution function of training dataset. The closeness of input data is compared with all the training neurons and the data is classified into the category with maximum closeness.

C. Deep Neural Network (DNN)

The Deep Neural Network architecture used in this study is a 4-layer perceptron . By definition any neural network architecture that has more than one hidden layer is a DNN. It has a similar working principle to that of a 3-layer perceptron and uses the same method for reducing error. The advantage of an additional layer is that it is better at non-linear separation and has better noise tolerance. DNNs, though, are very computationally intensive and data hungry.

D. Radial Basis Function Neural Network (RBFNN)

Radial Basis Function Neural Network is based on the principle of Cover's theorem which states that a complex pattern classification problem when casted into a higher dimensional space nonlinearly, is more likely to be separable than in a low-dimensional space. In the most basic form it is a three-layer network with the following function:

First layer acts as an interface between the network and the environment i.e. it accepts the input data. Second layer is the only hidden layer and is used to map the input space to higher dimensional

space through a nonlinear transformation. Gaussian functions, multi-quadrics, inverse-multi quadrics etc. can be used for the same. Third Layer gives the output of the network which is then compared to required output to obtain error. This error is then reduced to requisite level iteratively, using Least Mean Square algorithm to train the network.

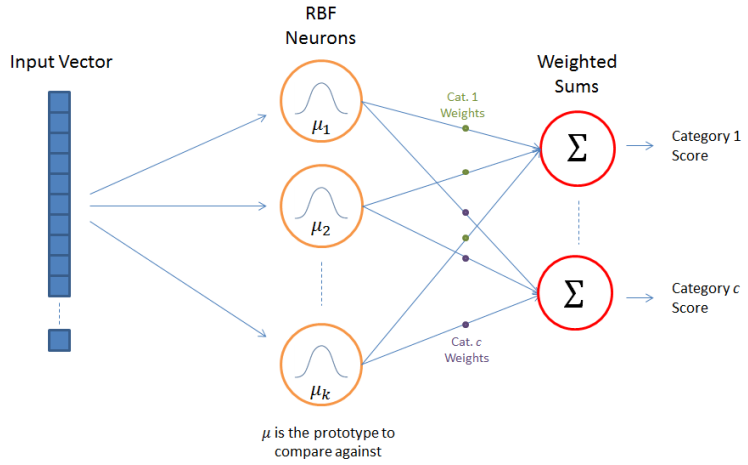


Fig 8: RBFNN Architecture

TABLE I. ACCURACY OF SPEAKER RECOGNITION USING SLFN

Number of Speakers	**Clean Data**	**Noisy Data**					
		-5dB	0dB	5dB	10dB	20dB	30dB
50	92.18	76.81	84.86	88.92	91.096	92.352	92.24
40	94.53	76.95	86.54	90.57	92.66	94.15	94.81
20	96.56	82.85	89.35	92.62	94.31	95.24	95.73
10	98.8	90.68	95.64	94.76	97.64	98.44	99.08

TABLE II. ACCURACY OF SPEAKER RECOGNITION USING PNN

Number of Speakers	**Clean Data**	**Noisy Data**					
		-5dB	0dB	5dB	10dB	20dB	30dB
50	96.096	31.47	51.72	71.44	85.99	95.23	95.76
40	97.2	34.07	54.02	73.82	86.85	95.73	97.13
20	97.82	44.32	62.54	78.68	90.84	96.61	97.72
10	99.2	59.48	73.28	75.32	96.36	99.08	99.24

TABLE III. ACCURACY OF SPEAKER RECOGNITION USING DNN

Number of Speakers	**Clean Data**	**Noisy Data**					
		-5dB	0dB	5dB	10dB	20dB	30dB
50	86.07	61.26	72.87	80.87	85.6	87.00	87.90
40	89.37	60.34	66.87	82.13	83.87	87.79	89.03
20	96.13	78.90	87.85	91.33	94.28	95.34	96.01
10	98.6	90.12	95.60	95.56	98.16	98.6	98.72

TABLE IV. ACCURACY OF SPEAKER RECOGNITION USING RBFNN

Number of Speakers	**Clean Data**	**Noisy Data**					
		-5dB	0dB	5dB	10dB	20dB	30dB
50	96.22	85.76	89.62	92.71	94.44	95.84	96.29
40	97.19	87.44	90.72	93.18	95.15	96.96	97.23
20	97.47	89.36	92.11	94.39	95.96	97.52	97.99
10	98.8	91.46	94.4	94.72	97.64	98.64	98.74

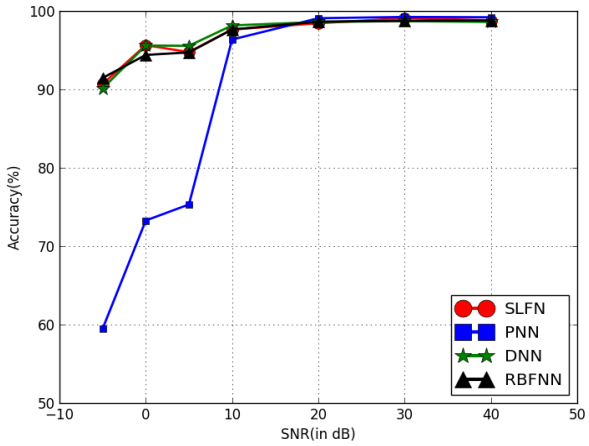


Fig 9: Accuracy Vs SNR (in dB) for 10 speakers

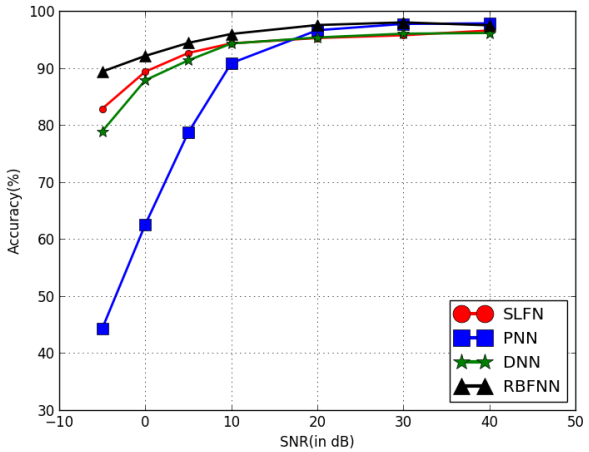


Fig 10: Accuracy Vs SNR (in dB) for 20 speakers

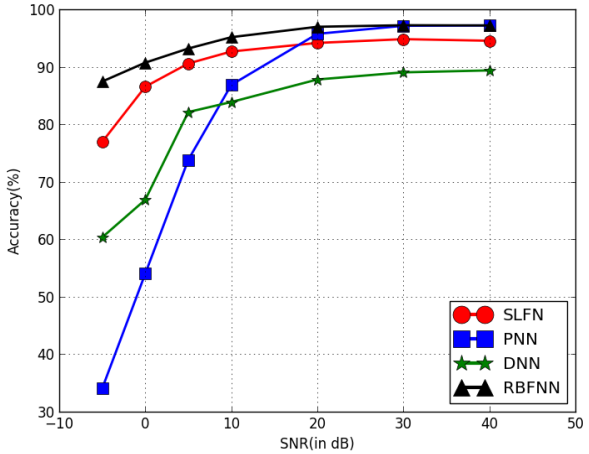


Fig 11: Accuracy Vs SNR (in dB) for 40 speakers

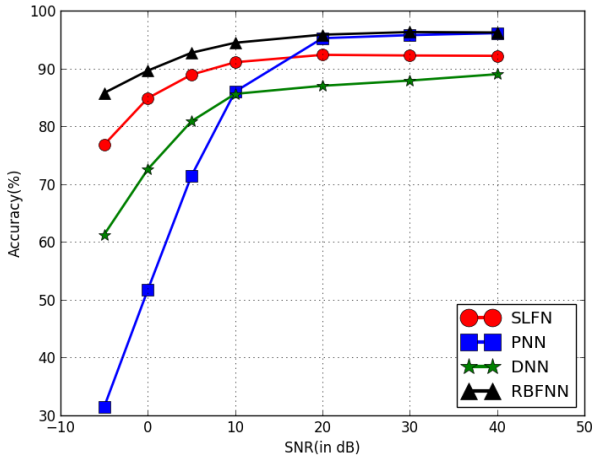
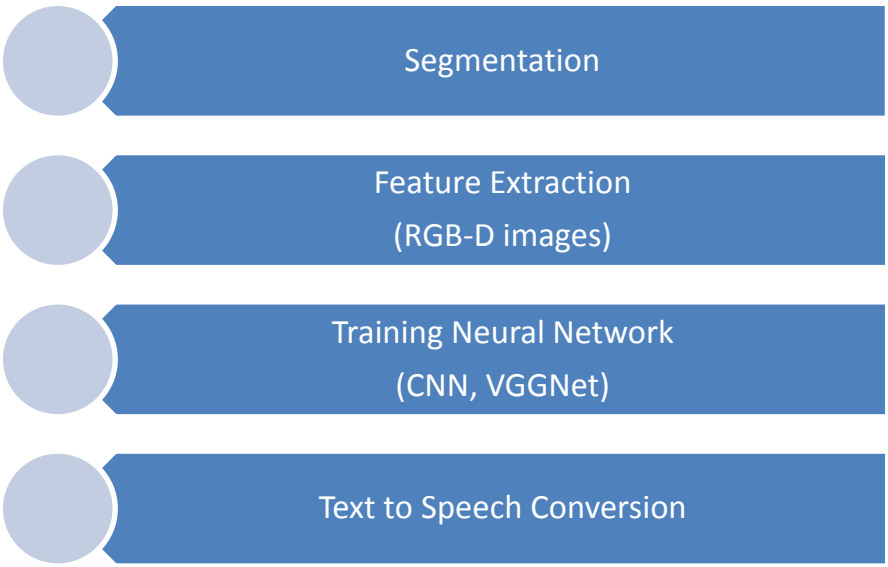


Fig 12: Accuracy Vs SNR (in dB) for 50 speakers

RESULTS AND DISCUSSIONS

We found out the MFCC features of each utterance using the MIR Toolbox in MATLAB. The features were vectors of length 13 each. We used these vectors to train all the four neural networks for speaker recognition for 10, 20, 40 and 50 speakers respectively, on both clean and noisy data. The accuracy obtained for all four neural networks has been tabulated below. Assuming SNR of clean data to be 40dB (Pspeech = 10000 Pnoise), a comparison of performances of all four neural networks for different number of speakers is also shown as plots of Accuracy vs SNR (in dB). As can be inferred, the overall performance of RBFNN was the most consistent for all noise levels, while DNN learnt more complex features and, provided more data, should trump RBFNN in terms of accuracy. PNN on the other hand, was the fastest to train and compute, although it requires more memory and shows the largest fluctuations in accuracy with varying levels of noise.

GESTURE TO SPEECH



GESTURE RECOGNITION ON 2D IMAGES

As the starting and before the availability of Kinect sensor, we worked on gesture recognition using modified VGG (Visual Graphics Group) network and achieved an accuracy of 99 %.

We used a dataset of 7000 2D pictures (100 x 100 RGB images) all taken in different gradient plain background and for further testing 350 unseen images were used.

Images were sign language equivalent of 0-9 and A-Z.

The concept of transfer learning was used where we used weights pre-trained on Imagenet data set. This was done in order to save the unnecessarily large training time needed in training from scratch and instead using the weights of a network pre-trained on a huge number of objects.

The weights of the first 10 layers were kept non-trainable as these layers anyway compute generic fundamental features like edges, shapes etc., and the rest were fine-tuned with respect to our own data. Thus allowing us the luxury of using a dense architecture such as VGG net without having to train it from scratch.

Fully connected layers of original VGG net were modified to custom layers (3) with 4096,512 and 36 neurons in each respectively.

CONVOLUTIONAL NEURAL NETWORK

In machine learning, a 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. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

They have applications in image and video recognition, recommender systems and natural language processing.

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. We discuss them further below:

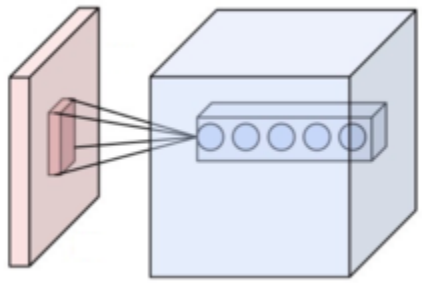


Fig 13: Neurons of a convolutional layer (blue), connected to their receptive field (red)

I.Convolutional layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

I. Pooling layer

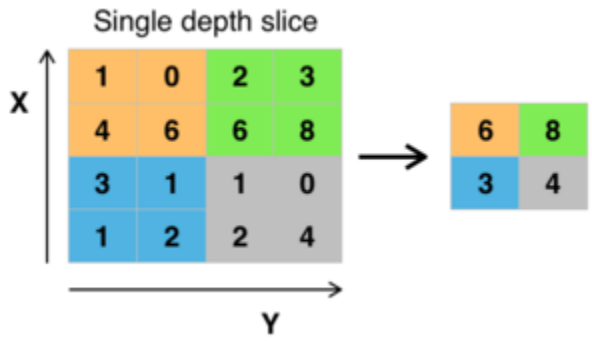


Fig 14: Pooling Layer

Max pooling with a 2x2 filter and stride = 2

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling among which *max pooling* is the most common. It partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting.

II. ReLU layer

ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function . It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Other functions are also used to increase nonlinearity, for example the saturating hyperbolic tangent , and the sigmoid function . ReLU is often preferred to other functions, because it trains the neural network several times faster without a significant penalty to generalisation accuracy.

III. Fully connected layer

Finally, after several convolutional and max pooling layers, the high-level reasoning in the neural network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

IV. Loss layer

The loss layer specifies how training penalizes the deviation between the predicted and true labels and is normally the final layer. Various loss functions appropriate for different tasks may be used there. Softmax loss is used for predicting a single class of K mutually exclusive classes. Sigmoid cross-entropy loss is used for predicting K independent probability values in Euclidean loss is used for regressing to real-valued labels .

ARCHITECTURE OF VGG NETWORK USED

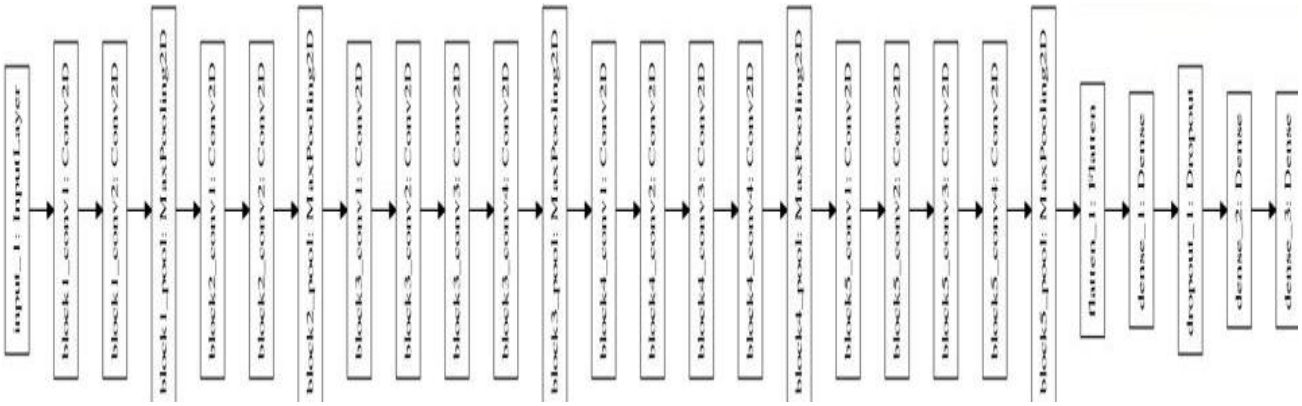


Fig 15: VGGNet

TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.

Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.

Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning.

Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained.

Transfer learning only works in deep learning if the model features learned from the first task are general.

How to Use Transfer Learning?

You can use transfer learning on your own predictive modeling problems.

Two common approaches are as follows:

1. Develop Model Approach
2. Pre-trained Model Approach

Develop Model Approach

1. **Select Source Task**. You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
2. **Develop Source Model**. Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.
3. **Reuse Model**. The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
4. **Tune Model**. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

Pre-trained Model Approach

1. **Select Source Model**. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
2. **Reuse Model**. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
3. **Tune Model**. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

This second type of transfer learning is common in the field of deep learning.

Transfer Learning with Image Data

It is common to perform transfer learning with predictive modeling problems that use image data as input.

This may be a prediction task that takes photographs or video data as input.

For these types of problems, it is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.

The research organizations that develop models for this competition and do well often release their final model under a permissive license for reuse. These models can take days or weeks to train on modern hardware.

These models can be downloaded and incorporated directly into new models that expect image data as input.

Three examples of models of this type include:

- Oxford VGG Model
- Google Inception Model
- Microsoft ResNet Model

VGGNET (Visual Graphics Group Network)

It is a 16 or 19 layer deep convolutional neural network that can be used for complex classification task.

v. VGG16 and VGG19

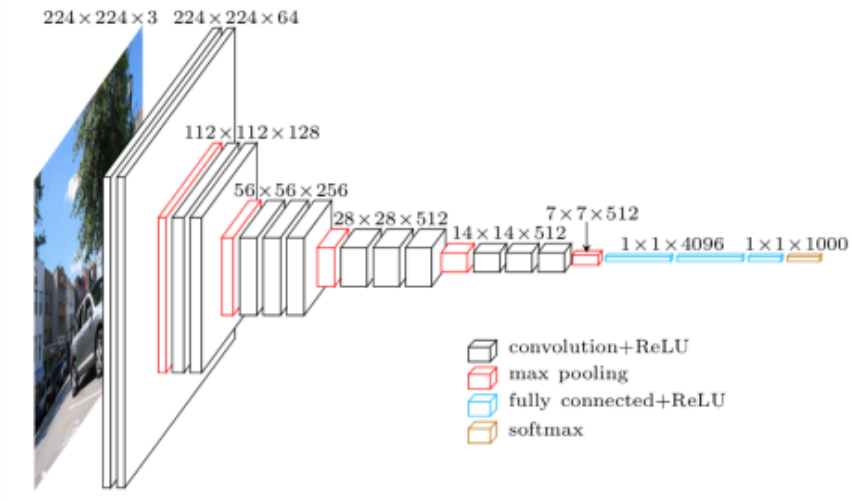


Figure 16: A visualization of the VGG architecture.

The VGG network architecture was introduced by Simonyan and Zisserman in their 2014 paper. This network is characterized by its simplicity, using only *3×3* convolutional layers stacked on top of each other in increasing depth. Reducing volume size is handled by max pooling. Two fully-connected layers, each with 4,096 nodes are then followed by a softmax classifier.
The "16" and "19" stand for the number of weight layers in the network :

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Fig 17: Layer Architecture of VGGNet

In 2014, 16 and 19 layer networks were considered *very* deep (although we now have the ResNet architecture which can be successfully trained at depths of 50-200 for ImageNet and over 1,000 for CIFAR-10).
Simonyan and Zisserman found training VGG16 and VGG19 challenging (specifically regarding convergence on the deeper networks), so in order to make training easier, they first trained smaller versions of VGG with less weight layers (columns A and C) first.

The smaller networks converged and were then used as initializations for the larger, deeper networks — this process is called pre-training.

While making logical sense, pre-training is a very time consuming, tedious task, requiring an entire network to be trained before it can serve as an initialization for a deeper network.

We no longer use pre-training (in most cases) and instead prefer Xaiver/Glorot initialization or MSRA initialization (sometimes called He et al. initialization from the paper. Unfortunately, there are two major drawbacks with VGGNet:

1. It is painfully slow to train.
2. The network architecture weights themselves are quite large (in terms of disk/bandwidth).

Due to its depth and number of fully-connected nodes, VGG is over 533MB for VGG16 and 574MB for VGG19. This makes deploying VGG a tiresome task.

We still use VGG in many deep learning image classification problems; however, smaller network architectures are often more desirable (such as SqueezeNet, GoogLeNet, etc.).

MODEL ACCURACY & LOSS PLOT OF THE TRAINED VGG NETWORK

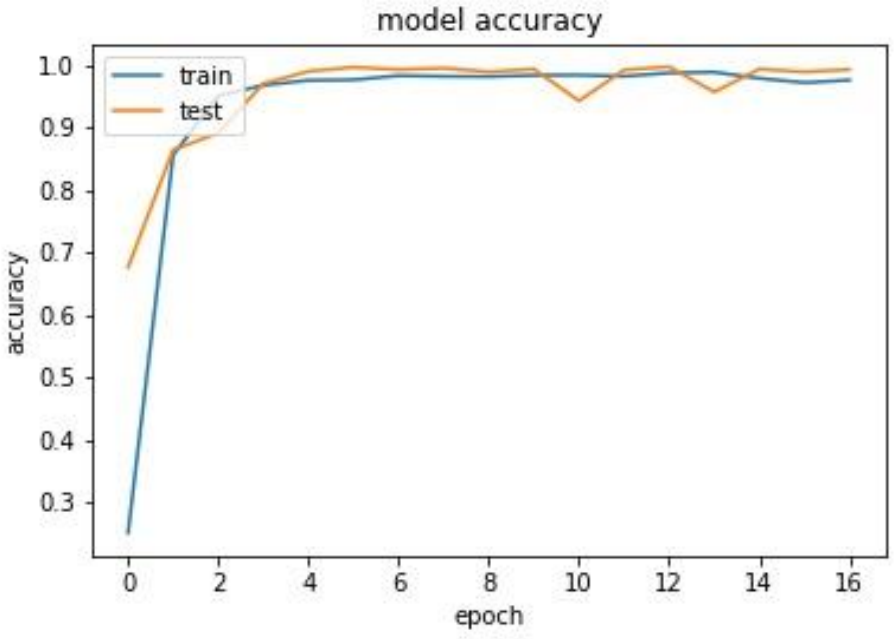
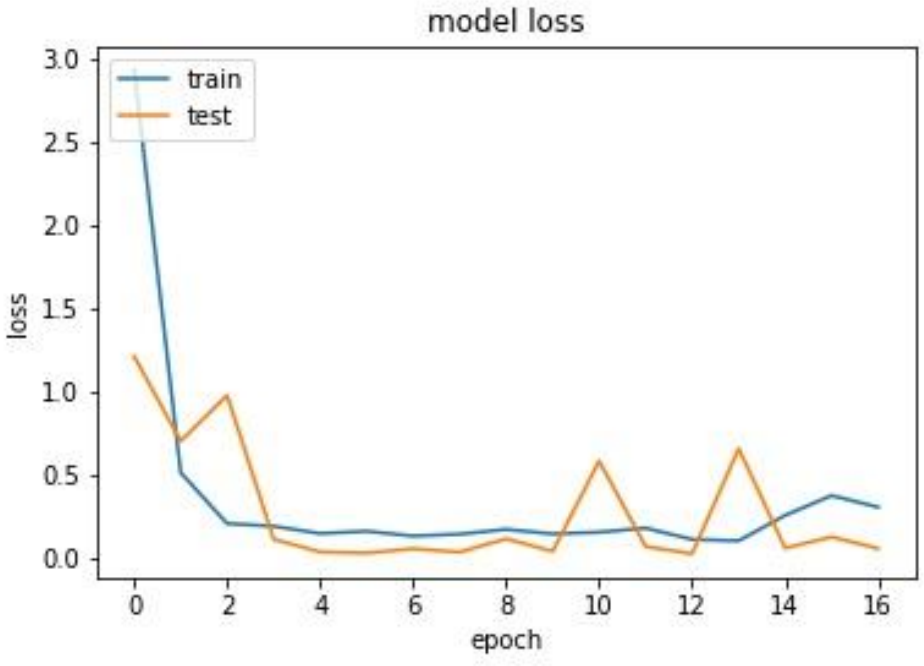


Fig 18 : Model Accuracy of trained VGG Netwrok

Fig 19: Loss plot of the trained VGG Network



WORKING ON RGB-D IMAGES USING KINECT

RGB vs RGB-D images :

A RGB-D image is simply a combination of a RGB image and its corresponding depth image. A depth image is an image channel in which each pixel relates to a distance between the image plane and the corresponding object in the RGB image.

You can use Kinect to capture such RGB-D images. If the use of Kinect-like hardware is not available in your task, you may need to estimate the depth from images of the same scene taken from multiple cameras, coming down to a computer vision problem. Another solution is to collect training data containing RGB-D images and to use machine learning techniques.

Why Kinect?

We decided to use Kinect Sensor in order to accommodate more complex backgrounds that we could come across in everyday situation instead of the simple backgrounds as in 2-D dataset.

Also, since Indian sign language is two handed, we need to count in the occlusion of one hand by the other as well as different relative positions of both hands, and Kinect with its depth rendering capacity should help to improve performance in such cases.

Microsoft Kinect Sensor

Recent advances in 3D depth cameras such as Microsoft Kinect sensors have created many opportunities for multimedia computing. Kinect was built to revolutionize the way people play games and how they experience entertainment. With Kinect, people are able to interact with the games with their body in a natural way. The key enabling technology is human body language understanding; the computer must first understand what a user is doing before it can respond. This has always been an active research field in computer vision, but it has proven formidably difficult with video cameras.

The Kinect sensor lets the computer directly sense the third dimension (depth) of the players and the environment, making the task much easier. It also understands when users talk, knows who they are when they walk up to it, and can interpret their movements and translate them into a format that developers can use to build new experiences. Kinect's impact has extended far beyond the gaming industry. With its wide availability and low cost, many researchers and practitioners in computer science, electronic engineering, and robotics are leveraging the sensing technology to develop creative new ways to interact with machines and to perform other tasks, from helping children with autism to assisting doctors in operating rooms. Microsoft calls this the Kinect Effect.

Figure 1. Microsoft Kinect sensor. (a) The Kinect sensor for Xbox 360. (b) The infrared (IR) projector, IR camera, and RGB camera inside a Kinect sensor.

Fig 20: Kinect Sensor

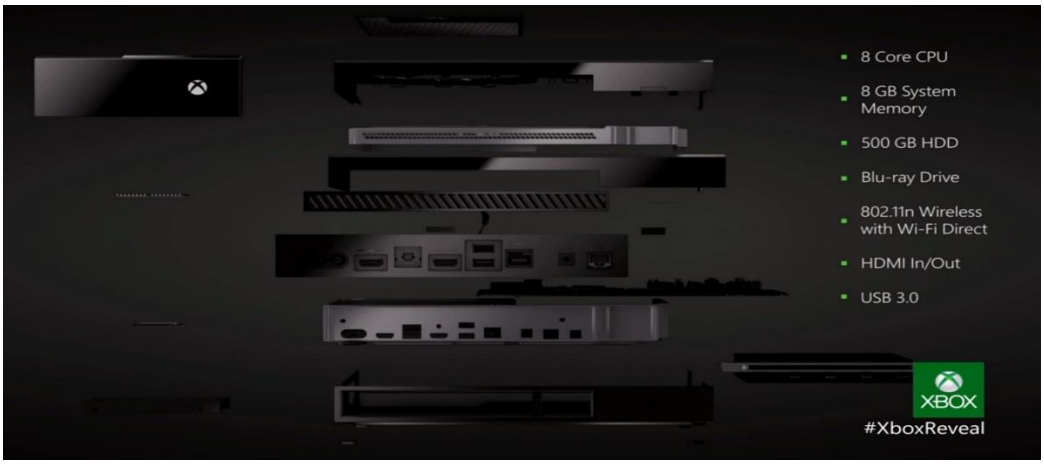


Fig 21: Kinect sensor with Specifications

The innovative technology behind Kinect is a combination of hardware and software contained within the Kinect sensor accessory that can be added to any existing Xbox 360. The Kinect sensor is a flat black box that sits on a small platform, placed on a table or shelf near the television you're using with your Xbox 360. Newer Xbox 360s have a Kinect port from which the device can draw power, but the Kinect sensor comes with a power supply at no additional charge for users of older Xbox 360 models. For a video game to use the features of the hardware, it must also use the proprietary layer of Kinect software that enables body and voice recognition from the Kinect sensor.There's a trio of hardware innovations working together within the Kinect sensor:

- **Color VGA video camera** - This video camera aids in facial recognition and other detection features by detecting three color components: red, green and blue. Microsoft calls this an "RGB camera" referring to the color components it detects.
- **Depth sensor** - An infrared projector and a monochrome CMOS (complimentary metal-oxide semiconductor) sensor work together to "see" the room in 3-D regardless of the lighting conditions.
- **Multi-array microphone** - This is an array of four microphones that can isolate the voices of the players from the noise in the room. This allows the player to be a few feet away from the microphone and still use voice controls.

HEADER /	
Brand	Microsoft
Product Line	Microsoft Kinect
Model	for Xbox One
Packaged Quantity	1
Compatibility	Game console
INPUT DEVICE /	
Product Type	motion sensor
Connectivity Technology	wired
Interface	USB
MISCELLANEOUS /	
Color Category	black
Compatible Game Consoles	Microsoft Xbox One
GENERAL /	
Manufacturer	Microsoft

Fig 22 Kinect Specifications

SKIN DETECTION

Skin detection is the process of finding skin-coloured pixels and regions in an image or a video. This process is typically used as a pre-processing step to find regions that potentially have human faces and limbs in images. Skin image recognition is used in a wide range of image processing applications like face recognition, skin disease detection, gesture tracking and human-computer interaction. The primary key for skin recognition from an image is the skin colour. But colour cannot be the only deciding factor due to the variation in skin tone according to different races. Other factors such as the light conditions also affect the results. Therefore, the skin tone is often combined with other cues like texture and edge features. This is achieved by breaking down the image into individual pixels and classifying them into skin coloured and non-skin coloured. One

simple method is to check if each skin pixel falls into a defined colour range or values in some coordinates of a colour space. There are many skin colour spaces like RGB, HSV, YCbCr, YIQ, YUV, etc. that are used for skin colour segmentation. The following factors should be considered for determining the threshold range:

- Effect of illumination depending on the surroundings.
- Individual characteristics such as age, sex and body parts.
- Varying skin tone with respect to different races.
- Other factors such as background colours, shadows and motion blur.

The skin detection is influenced by the parameters like Brightness, Contrast, Transparency, Illumination, and Saturation. The detection is normally optimized by taking into consideration combinations of the mentioned parameters in their ideal ranges.

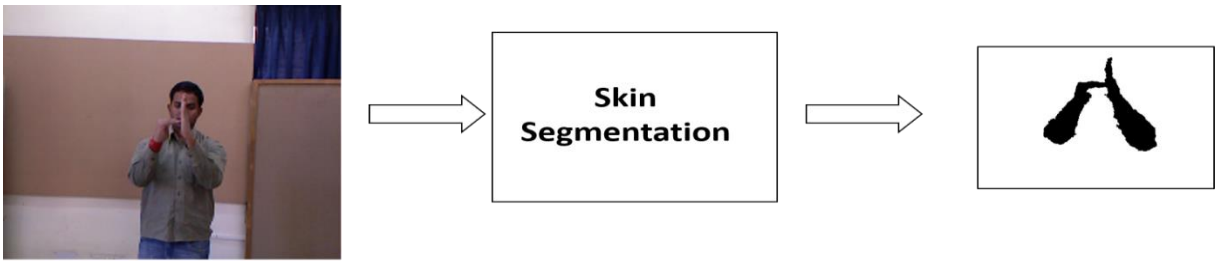


Fig 23: Skin Segmentation

HSV COLOUR MODEL

Hue Saturation Value (HSV) Colour Model The HSV colour space is more intuitive to how people experience colour than the RGB colour space. As hue (H) varies from 0 to 1.0, the corresponding colours vary from red, through yellow, green, cyan, blue, and magenta, back to red. As saturation(S) varies from 0 to 1.0, the corresponding colours (hues) vary from unsaturated (shades of grey) to fully saturated (no white component). As value (V), or brightness, varies from 0 to 1.0, the corresponding colours become increasingly brighter. The hue component in HSV is in the range 0° to 360° angle all lying around a hexagon. With RGB the colour will have values like (0.5, 0.5, 0.25), whereas for HSV it will be (30°, √3/4, 0.5). HSV is best used when a user is selecting a colour interactively It is usually much easier for a user to get to a desired colour as compared to using RGB.

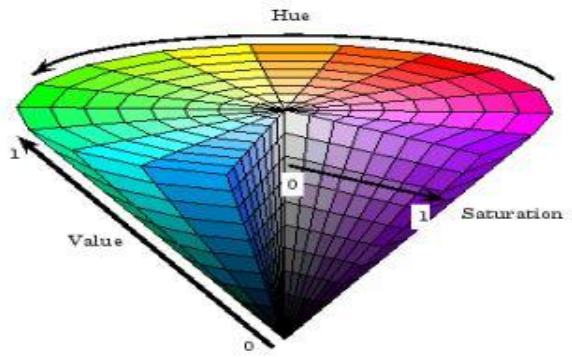


Fig 24:HSV Model

We trained a Gaussian Probability Model using the HSV values of skin pixels AS VARIABLES. When any pixel (viz the HSV values of the pixel) is fed into this model, it outputs the probability of that pixel being skin or non skin.

Formula for Multivariate Gaussian Mixture Model :

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_i)^{\mathrm{T}} {\Sigma_i}^{-1} (\vec{x} - \vec{\mu}_i)\right)$$

$$\sum_{i=1}^{K} \phi_i = 1$$

After having trained the GMM on RGB images, in order to further improve the skin segmentation procedure by reducing the number of false positives, such as those in background etc., we used the depth information of the depth images. Here we found out the regions closest to the camera using the intensity of depth image pixels by thresholding. Now, only those pixels that were among the intersection of both the GMM predicted pixels and the depth thresholded pixels were classified as being skin pixel.

Although, we weren't concerned with the accuracy of the skin segmentation model, as this was not our specific problem statement, but it performed admirably well for our use case as you can see from the several results below. The skin segmentation was performed in order to reinforce the skin regions in the RGB and depth image before being passed on to the Deep neural networks for classification in order to improve the accuracy. Reinforcement or emphasizing was done as necessary on both the RGB and depth images. For example, the skin pixel regions were further darkened in the depth images for ease of classification.

DATA

We collected RGB-D data for different Indian Signs. The Kinect based RGB-D data was made for around 48 different entities. These include both RGB and Depth images of digits, alphabets and a few common words. The dataset comprises of around 36 images per word in our vocabulary, contributed by 18 different people. The vocabulary is as given below:

• 0	• L
• 1	• M
• 2	• N
• 3	• O
• 4	• P
• 5	• Q
• 6	• R
• 7	• S
• 8	• T
• 9	• U
• 10	• V
• A	• W
• B	• X
• C	• Y
• D	• Z
• E	• HELP
• F	• HOW
• G	• IT
• H	• ME
• I	• NEGATIVE
• K	• OKAY

• SOME	• PRAY
• GOOD	• HERE
• STUDY	• YOU
• WORK	• UP

Table : List of Vocabulary

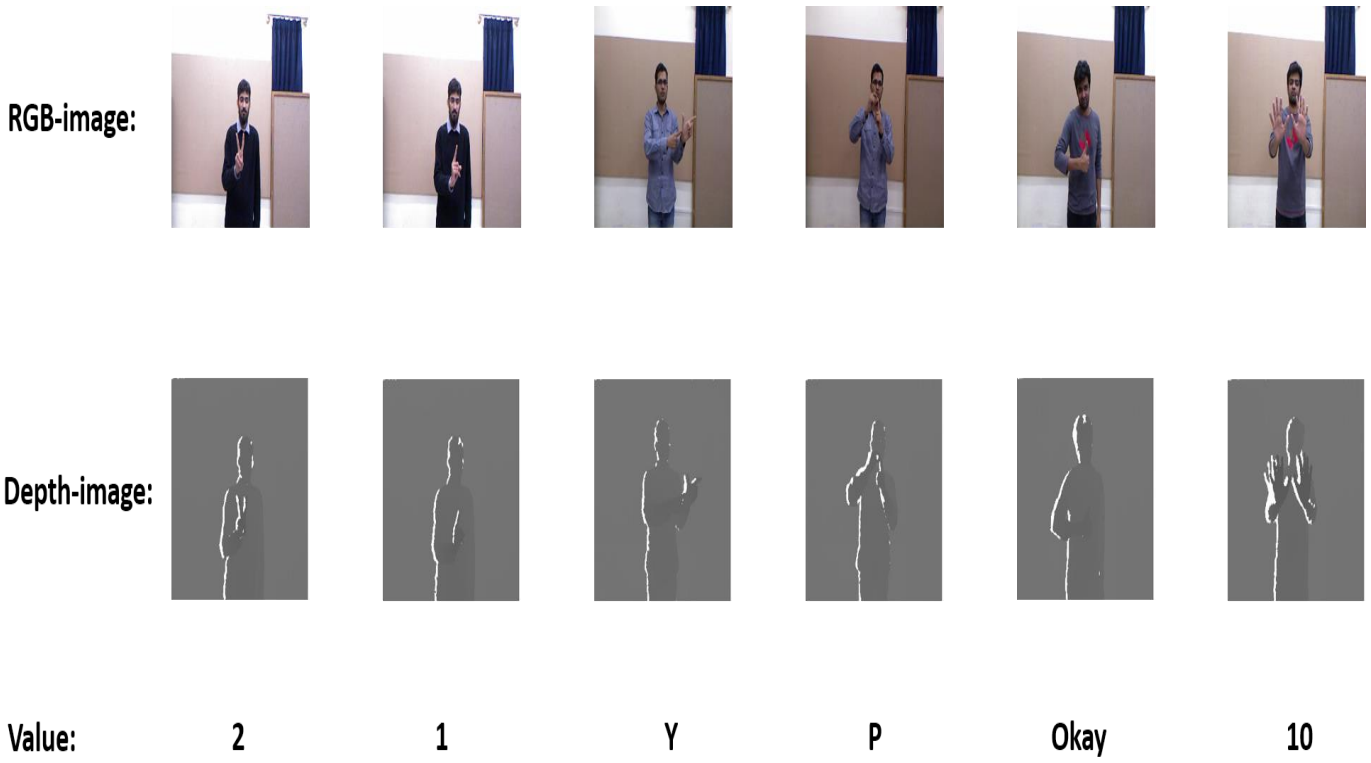


Fig 25 : Few instances from our dataset

As mentioned above, only 36 images were recorded for each gesture which would not have been enough for training our deep learning models and learning all the variances. To address this problem, we came up with a workaround in the form of data augmentation.

DATA AUGMENTATION

Data augmentation refers to the process of multiplying data points in order to increase the training data size by employing various techniques to the already available data such as blurring, cropping, scaling, translating, rotating, etc. It is an important step to boost the performance of deep neural networks especially in cases where the data available is limited.

Data augmentation adds value to base data by adding information derived from internal and external sources within an enterprise. Data is one of the core assets for an enterprise, making data management essential. Data augmentation can be applied to any form of data, in our case, images. We augmented our data to six times its original size by applying various spatial transformations like translation, cropping, etc. thereby ending up with around 200 images per word in our vocabulary. We performed augmentation using the imgaug library in python. We ensured that the same transformations were applied to both the RGB and Depth components of the images to maintain concurrency between them.

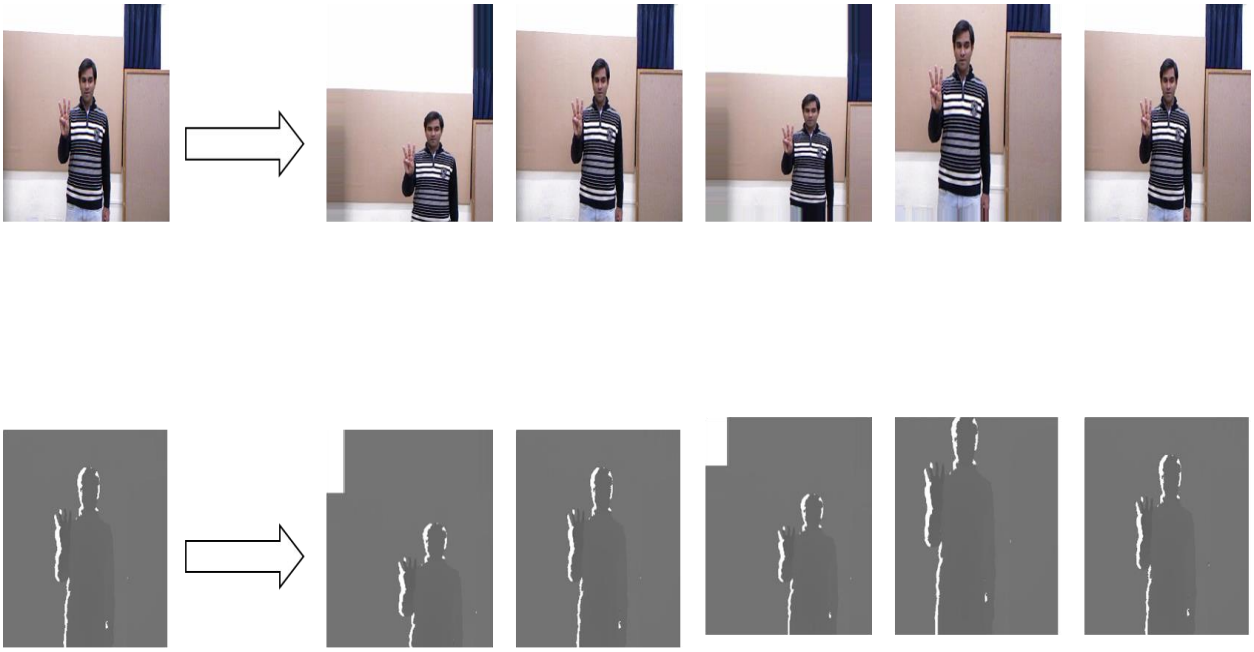


Fig 26: Data Augmentation

APPROACH FOR GESTURE RECOGNITION WITH RGB-D DATA

ResNet

Unlike traditional sequential network architectures such as AlexNet, OverFeat, and VGG, ResNet is instead a form of "exotic architecture" that relies on micro-architecture modules (also called "network-in-network architectures").

First introduced by He et al. in their 2015 paper, Deep Residual Learning for Image Recognition, the ResNet architecture has become a seminal work, demonstrating that extremely deep networks can be trained using standard SGD (and a reasonable initialization function) through the use of residual modules.

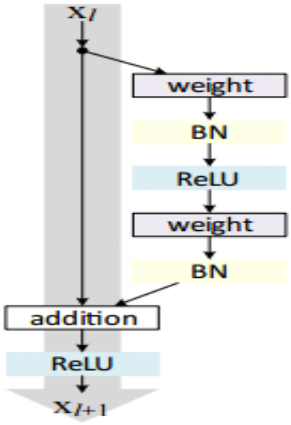


Fig 27: Basic Residual Network.

Intuition behind Residual Networks

Make it deep, but remain shallow

Given a shallower network - how can we take it, add extra layers and make it deeper - without losing accuracy or increasing error? It's tricky to do but one insight is that if the extra layers added to the deeper network are identity mappings, they become equivalent to the shallower network. And hence, they should produce no higher training error than its shallower counterpart. This is called a solution by construction by the authors.

Understanding residual

A residual is the error in a result.

Let's say, you are asked to predict the age of a person, just by looking at her. If her actual age is 20, and you predict 18, you are off by 2. 2 is our residual here. If you had predicted 21, you would have been off by -1, our residual in this case. In essence, residual is what you should have added to your prediction to match the actual.

What is important to understand here is that, if the residual is 0, we are not supposed to do anything to the prediction. We are to remain silent since the prediction already matched the actual.

THE RESIDUAL NETWORK

So we want a deeper network where:

- We want to go deeper without degradation in accuracy and error rate. We can do this via injecting identity mappings.
- We want to be able to learn the residuals so that our predictions are close to the actuals.

That's what the Residual Network does. This is realized by feedforward neural network with shortcut connections. As the paper says:

Shortcut connections are those skipping one or more layers. In our case, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Identity shortcut connections add neither extra parameter nor computational complexity. The entire network can still be trained end-to-end by SGD with backpropagation, and can be easily implemented using common libraries without modifying the solvers.

The network can be mathematically depicted as:

$$H(x) = F(x) + x, \text{ where } F(x) = W2*relu(W1*x+b1)+b2$$

During training period, the residual network learns the weights of its layers such that if the identity mapping were optimal, all the weights get set to 0. In effect F(x) become 0, as in x gets directly mapped to H(x) and no corrections need to be made. Hence these become your identity mappings which help grow the network deep. And if there is a deviation from optimal identity mapping, weights and biases of F(x) are learned to adjust for it. Think of F(x) as learning how to adjust our predictions to match the actuals. These networks are stacked together to arrive at a deep network architecture.

OUR APPROACH

As part of our initial approach, we trained a Residual Network (RESNET) on RGB and Depth images stacked vertically over each other. The residual network has 50 layers. The stacked images were resized into 224*224 images before being passed on to the network for classification.

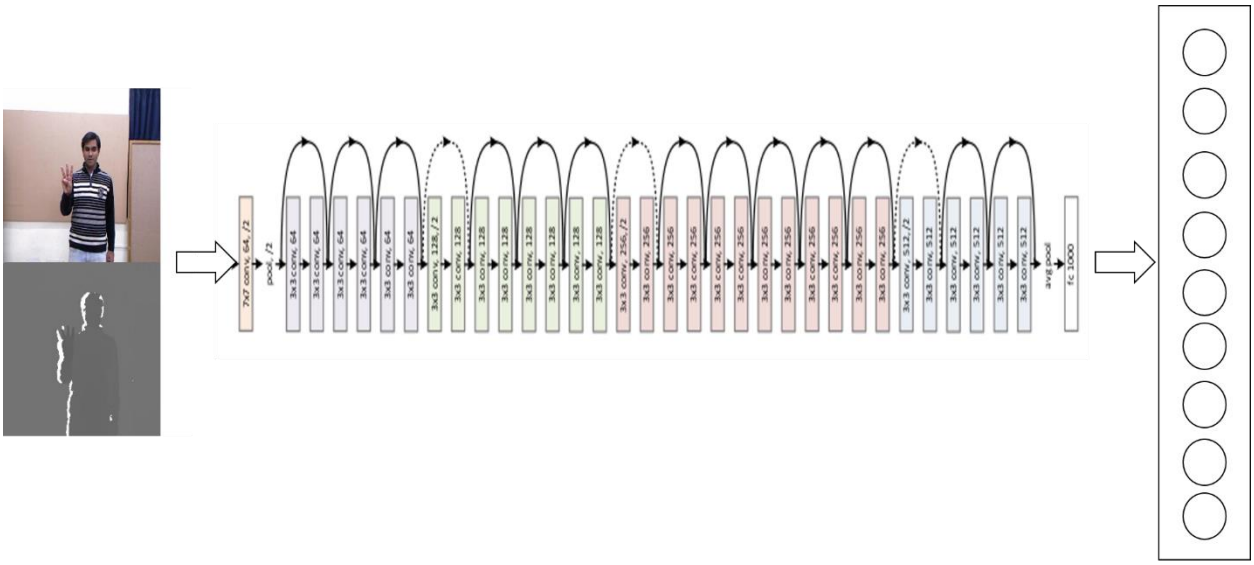


Fig 28: Using Resnet and RGBD image for sign language detection

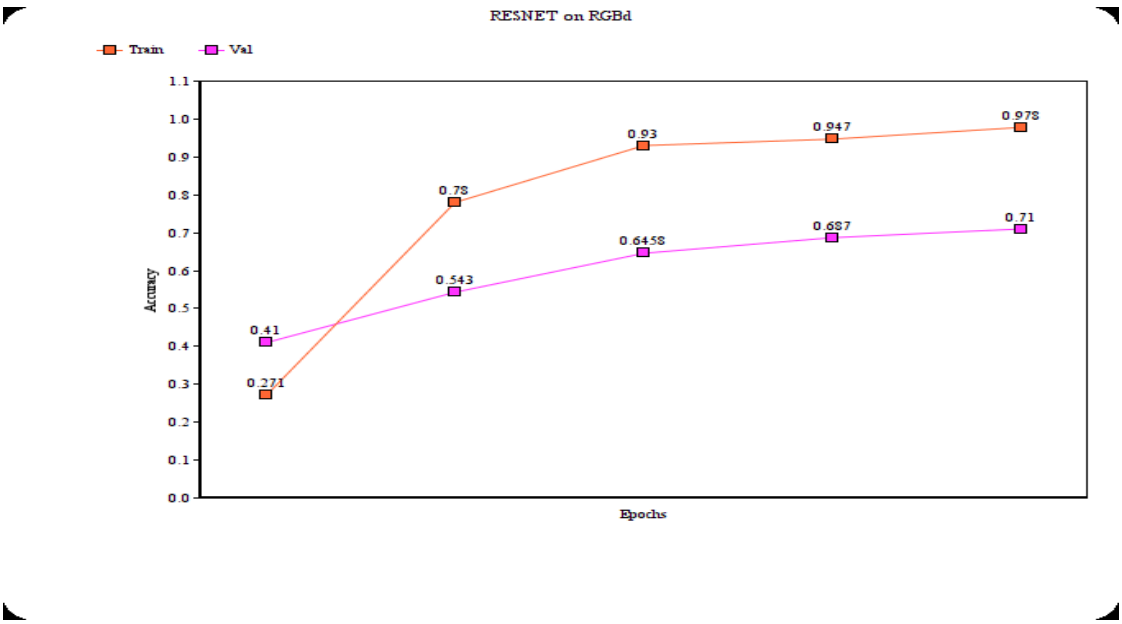


Fig 29: Performance of ResNet 50 on RGBD data

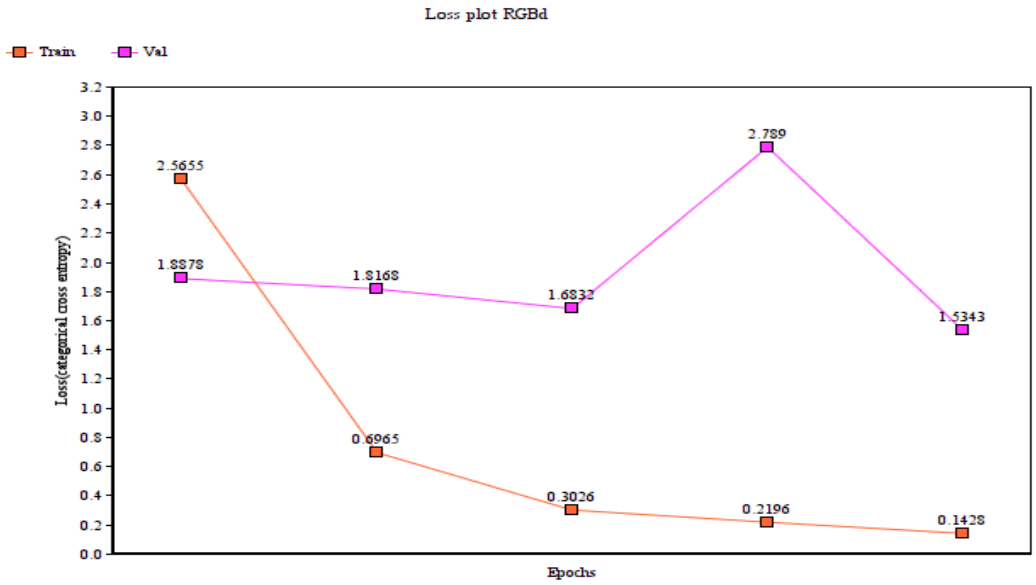


Fig 30: Loss Plot of ResNet 50 on RGBD data

We found at that ResNet 50 architecture gave 97.8% training accuracy and 71% validation accuracy .

BILINEAR CNN

Our Bilinear CNN architecture:

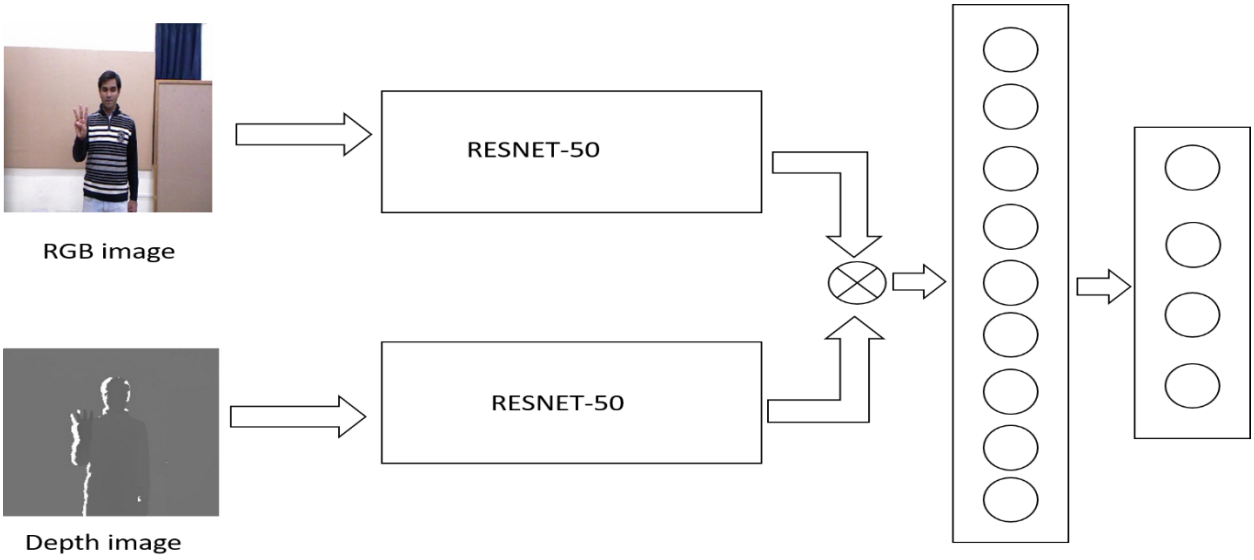


Fig 31:Bilinear CNN for RGBD image classification

Stacking the RGB and depth image and passing it through a ResNet model had its own shortcoming. Due to stacking of the images and reducing it 24*224*3 images a lot of information is lost. Also because of a single network handles both image RGB and depth, it has to find a compromising solution that converges for both the image hence not providing the best possible accuracy.

To deal with above stated issues we made use of Bilinear CNN. In this model we fed RGB image to one branch and Depth with another branch of the Bilinear model. These bottleneck features so obtained were then fed to densely connected neural network for classification.

We observed that in Bilinear model Training accuracy was 98% and validation accuracy was 79% on RGB image and for the Depth image training accuracy was 89% and validation accuracy was 63%. The mean accuracy by this model on test set was 84%.

One of the issue that was really concerning with respect to this model was getting inferences from this model was very expensive computationally. The size of the model itself was at around 800 MB. Bilinear model would have been a better choice if we could manage to get good accuracy with a smaller size model. This would have been possible only if we have more data as we could have decreased number of layers for similar performance hence generating a smaller size Bilinear model.

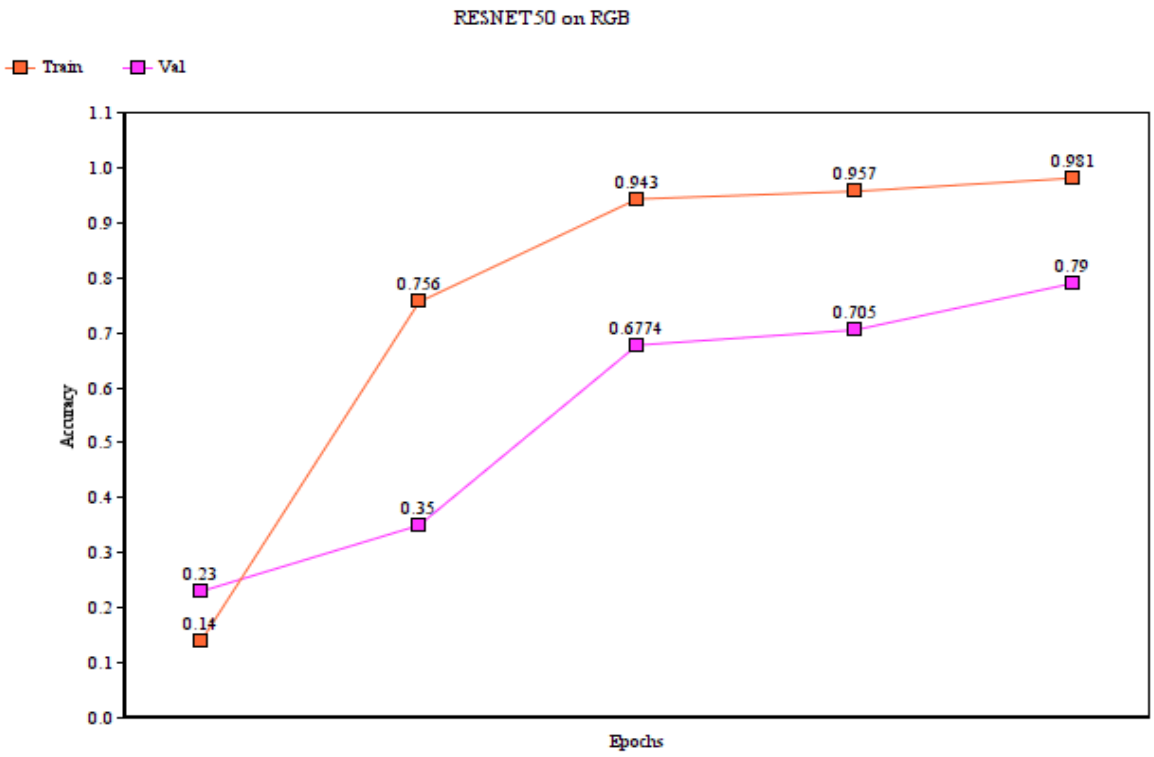


Fig 32: Performance of Bilinear CNN with ResNet 50 at one branch on RGB image.

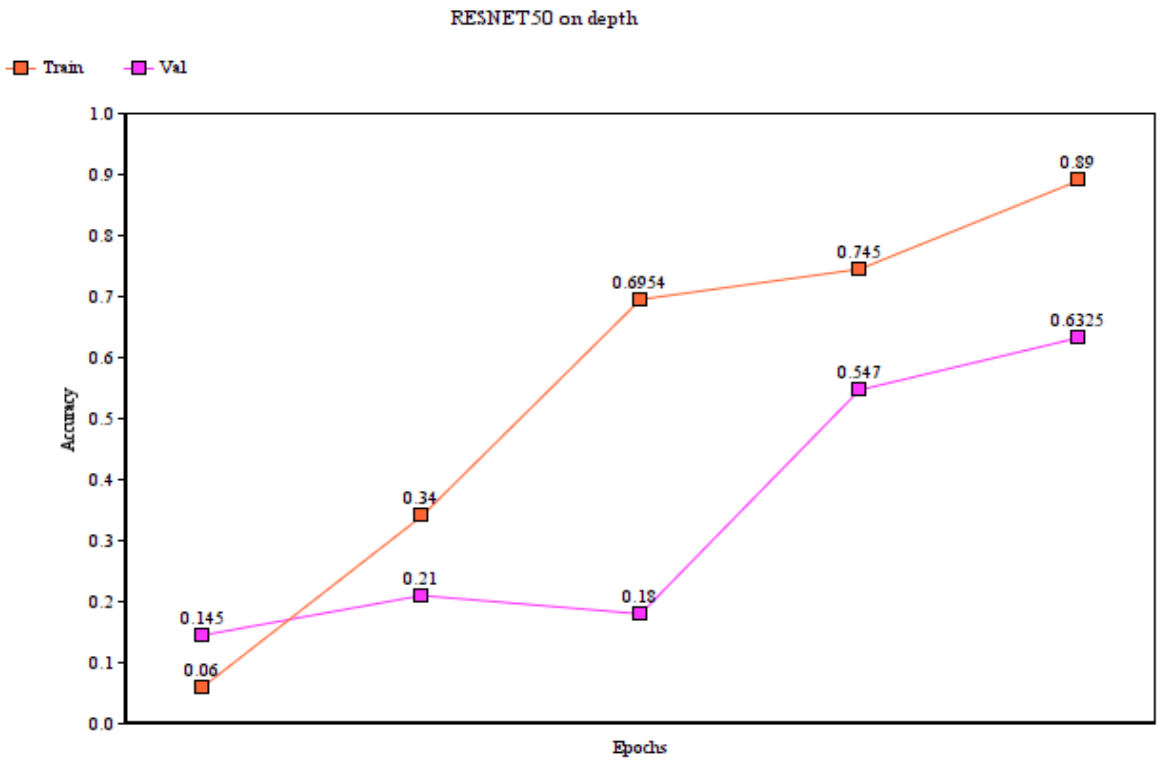


Fig 33: Performance of Bilinear CNN with ResNet 50 at one branch on Depth image.

GOOGLE TEXT TO SPEECH (OR GOOGLE TTS):

The sign language interpreted by our models generate output in terms of text. These texts are then converted to corresponding speech. To convert these texts to speech we use an API provided by Google called Google text to speech more popularly known as Google TTS.

CONCLUSION

- Therefore, we were able to explore the possibilities in terms of potential of deep learning and computer vision in addressing the problem of sign language conversion to speech and vice versa which could have a major impact in enabling normal seamless communication between the differently abled people and the rest of the world without the complexity of having to learn the different sign languages.
- Here are the major highlights of our project-
- We trained and compared several different neural networks for Hindi digit speech recognition, with Radial Basis function neural network outperforming the rest with an accuracy of >96%. This was done on a dataset of around 5000 utterances by 50 different speakers each uttering every digit 10 time. We used this model to predict spoken audio and mapping them to corresponding signs.
- We then proceeded to build our gesture to speech conversion subsystem. We trained our model on 2D images of various Indian sign language gestures taken in varying backgrounds and lighting conditions. The model used here was convolutional neural network, the architecture being VGG19. This model performed admirably well to classify the images into 34 classes (10 digits,24 alphabets) with an accuracy of 97%. We went ahead to build our Text to Speech(TTS) system on top of it for the complete gesture to speech conversion pipeline.
- But this was not all, we wanted to go further ahead test the performance on more generic, real world data, also in order to see the effect of depth information given by kinect sensor on the accuracy and performance. For this we wanted to use Microsoft's proprietary Kinect sensor to capture both RGB and depth information and use both of them in training our deep learning models.
- Hence, we created and recorded sign data images (RGB-D for 48 different entities) using the kinect sensor. There were about 36 images for each category contributed by 18 individuals. Considering the complexity of the problem this was a very small dataset. Therefore we augmented our data to include variations, using operations like scaling, translation, cropping etc. randomly. This inflated our data to 6 times its original number.
- After collection, there were two ways for us to approach the gesture to sign conversion on RGB-D data. One was to stack the RGB and its corresponding depth image vertically and passing it on to our CNN, which in this case was ResNet50,a deeper,50 layer architecture that can learn complex features, provided limited data.
- This model had a training accuracy of 97% and validation accuracy of 71% on previously unseen data, which was a great performance considering the limited data and its realworld complexity. This goes to show the potential of our model, provided more data. Another positive inference was the size of the model, which was only about 100 MB which is small compared to industrial standard models. This size will be further lowered upon increasing

data that would lead to lesser parameters required hence making this model ideal for deployment even on mobile devices.

- The second approach was borne out of our observation that due to stacking, we were losing a lot of information, than if we had separate models for both the RGB and Depth images. Also there was a compromise in terms of optimal weights for both of them that was being made. Hence we decided to test our hypothesis by building a homogenous bilinear CNN model where both RGB and Depth images were being fed to respective ResNets. The feature vectors then output by an intermediate layer were taken from both arrangements, concatenated and then sent to a linear classifier.
- This resulted in significant improvement as we had expected as it achieved 84% accuracy. But there was a tradeoff to be made as the computation complexity of the models and hence the execution time were both compromised in a way that overshadowed the accuracy gain. This made it practically infeasible for it to be real-time. But at least we can agree on the impressive potential of this bilinear implementation which could perform well when we have the computational resources to handle the complexity.
- Hence to conclude not only did we make a full Indian sign to speech interconversion system albeit for limited vocabulary, we also explored the different possibilities and potential improvements in the system.

FUTURE WORK

- Generating sequence of signs directly from speech input without the need for intermediate conversion of speech to text. This can be done by taking various features extracted from speech data as input to train the neural network.
- Extrapolating the system to support multiple languages and different sign language systems .
- Adding features like animated signing to make it more interactive.
- We could add functionalities so that people can use it to learn sign languages (virtual sign language tutor).

Work could be done to turn this into a compact little system that is easy to use and affordable and so on

REFERENCES

[1] K. Sarji, David. (2008). **HandTalk: Assistive Technology for the Deaf**. Computer. 41. 84 - 86. 10.1109/MC.2008.226.

[2] Kramer J, Parker M, Herrera D, Burrus N and Echtler F 2012 **Hacking the kinect**. Apress

[3] S. Liwicki and M. Everingham. **Automatic recognition of finger spelled words in british sign language**. In IEEE Workshop on CVPR for Human Communicative Behavior Analysis, 2009

[4] **ProDeaf Translator** (2016, August 9). Retrieved from https://play.google.com/store/apps/details?id=com.Proativa.ProDeafMove

[5] Asteriadis, Stylianos & Chatzitofis, Anargyros & Alexiadis, Dimitrios & Zarpalas, Dimitrios & Daras, Petros. (2013). **Estimating human motion from multiple Kinect Sensors**. ACM International Conference Proceeding Series. . 10.1145/2466715.2466727.

[6] Beigi, Homayoon. Fundamentals of speaker recognition. Springer Science & Business Media, 2011.

[7] O'Shaughnessy, Douglas. "Linear predictive coding." IEEE potentials 7.1 (1988): 29-32.

[8] Tiwari, Vibha. "MFCC and its applications in speaker recognition." International journal on emerging technologies 1.1 (2010): 19-22., 92, pp.68-73

[9] Martinez, Jorge, et al. "Speaker recognition using Mel frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques." Electrical Communications and Computers (CONIELECOMP), 2012 22nd International Conference on. IEEE, 2012.

[10] Lecun, Y. (1988). A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA (pp. 21-28). Morgan Kaufmann.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, Parallel Distributed Processing, volume 1. MIT Press, Cambridge, MA, 1986.

[12] Specht, Donald F. "Probabilistic neural networks." Neural networks 3.1 (1990): 109-118.

[13] Richardson, Fred, Douglas Reynolds, and Najim Dehak. "Deep neural network approaches to speaker and language recognition." IEEE Signal Processing Letters 22.10 (2015): 1671-1675.

[14] Gardner, Matt W., and S. R. Dorling. "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences." Atmospheric environment 32.14 (1998): 2627-2636.

[15] Strumiłło, Paweł, and Władysław Kamiński. "Radial basis function neural networks: theory and applications." Neural Networks and Soft Computing. Physica, Heidelberg, 2003. 107-119.

[16] Park, Jooyoung, and Irwin W. Sandberg. "Universal approximation using radial-basis-function networks." Neural computation 3.2 (1991): 246-257.

[17] Cover, Thomas M. "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition." IEEE transactions on electronic computers 3 (1965): 326-334.

[18] Liu, Weifeng, Puskal P. Pokharel, and Jose C. Principe. "The kernel least-mean-square algorithm." IEEE Transactions on Signal Processing 56.2 (2008): 543-554.

[19] Lartillot, Olivier, and Petri Toiviainen. "A Matlab toolbox for musical feature extraction from audio." International Conference on Digital Audio Effects. 2007.

[20] MATLAB 2015a, The MathWorks, Natick, 2015.