PES UNIVERSITY

ANN Project

Picture to Speech

Presented by: Harshith N Srivatsa
(PES1201701183)
Karan Vikyath
(PES1201701747)

Under the guidance of : Prof. Karpagavalli P

ABSTRACT

Image recognition system with audio generation for the corresponding image is the problem statement of our project.

Image recognition, in the context of machine vision, is the ability of software to identify objects, places, people, writing and actions in images. Computers can use machine vision technologies in combination with a camera and artificial intelligence neural networks to achieve image recognition and generate corresponding image labels.

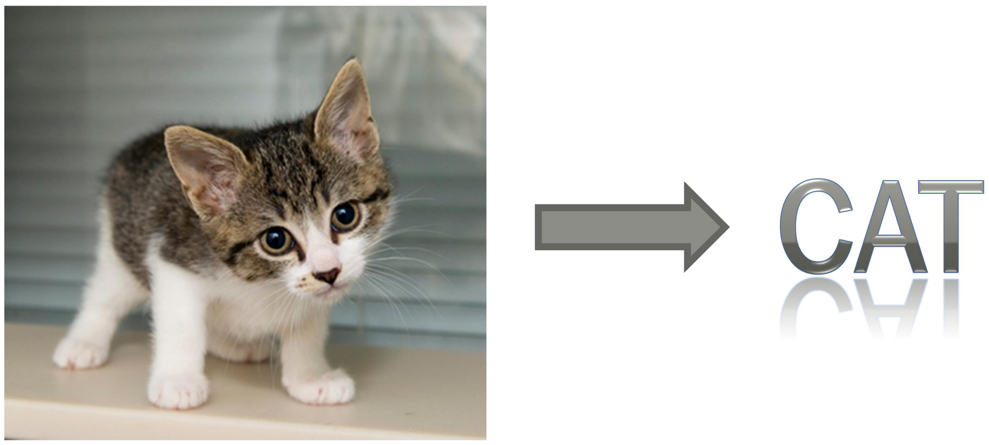
Speech synthesis is the artificial production of human speech. A computer system used for this purpose is called a speech computer or speech synthesizer, and can be implemented using artificial neural networks. The system converts normal the generated image labels into speech other systems render symbolic linguistic representations like phonetic transcriptions into speech.

INTRODUCTION

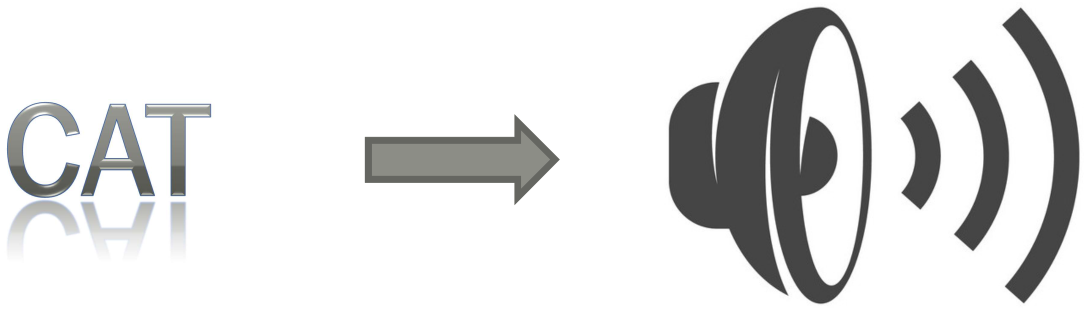
Semantic image segmentation or in other words Picture to Speech (PTS) is a process by which computers recognize and assign natural-language names to different objects in a photo or video for example Google Photos being able to not only see your dog in a picture but also identify it as a "dog" (versus, say, "cat" or "marmot") is the result of such a process. This project contains both the image identification problem as well as generating an audio which speaks out what the identified image is.

- <u>The project is divided into two parts:</u>

1.Picture to Text

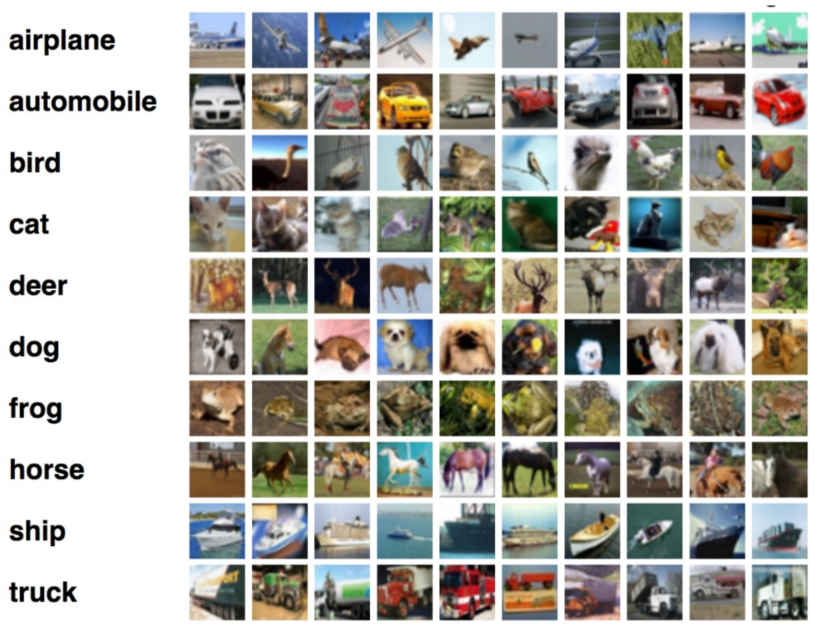


2.Text to Speech



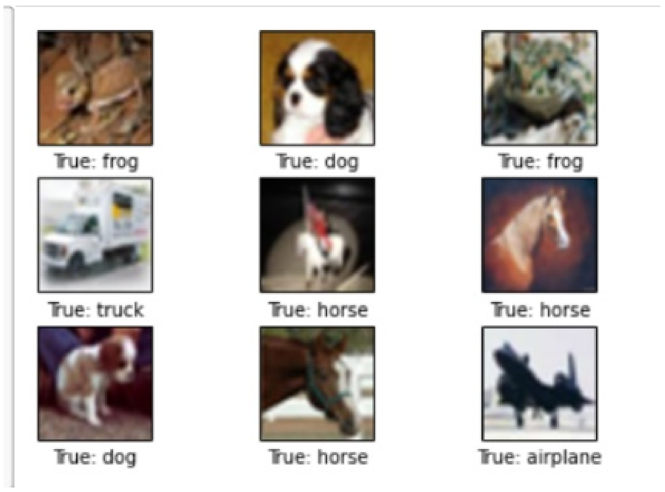
▪ Supervised Leaning

We are teaching the computer to recognise an image by categorizing it into the following classes. Before the system is able to recognize on its own it has to first learn what the picture is. The more pictures it sees of a particular object the better it gets at identifying the image. This is supervised learning. We give the system labelled data by which it recognises similar patterns in a picture of an object which is absent from the other objects and builds its own cognition based on it.



▪ Dataset

The dataset used here is the CIFAR-10 dataset which contains 60,000 images of size 32x32 pixels. There are 10 classes each containing 6,000 images. The images are small , clearly labelled and contain very little to no noise. Therefore, it is an ideal dataset for our problem statement



▪ Image Processing :

It has 3 main steps:

1.Pre-processing

2.Splitting the dataset

3.Building a convolution neural network

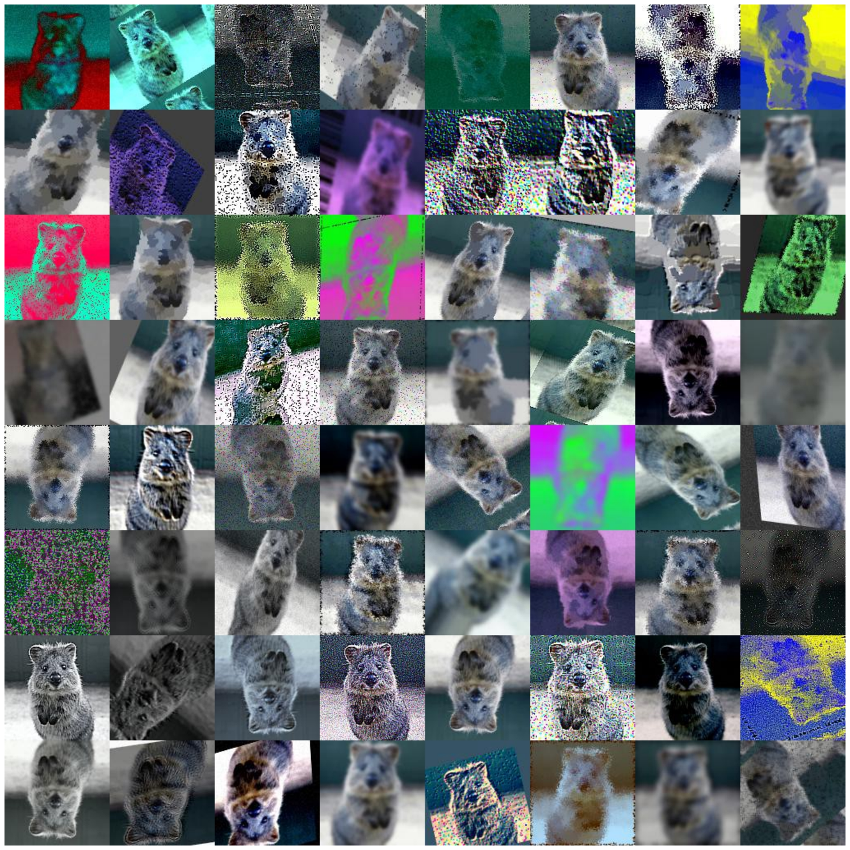
Step 1: Pre-Processing:

We need to add a little bit of variance to the dataset as it contains very little to no noise. We do so by artificially adding noise to the images by doing the combination of the following

1.Cropping different parts of the image.

2.Flipping the image horizontally.

3.Adjust the hue ,contrast and saturation.



Step 2:Splitting the dataset

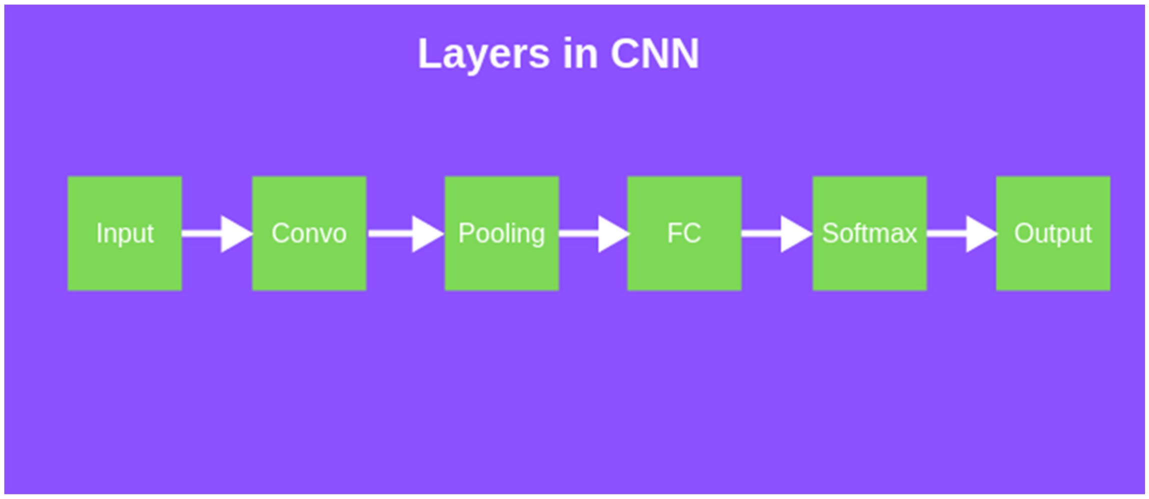
It takes a long time to calculate the gradient for the entirety of the large dataset.Therefore we are using only 500 images per class i.e, 5,000 images in total. 60% of the images are used as training set and 40% as testing set. We are training at 30 epochs per class i.e, 120 iterations per class.

Step 3: Building a convolution neural network

A convolution neural network is just like any other network except that it has a convolution layer. We are using it for two reasons ,to increase the accuracy of the image classification and to reduce the dimension of the image.Suppose we have a 227x227x3 size image then we need 154,587 neurons therefore using convolution to reduce it to 1x1x1000 where we need 1000 neurons which is much more manageable.

There are six different layers in CNN

- Input layer
- Convo layer (Convo + ReLU)
- Pooling layer
- Fully connected(FC) layer
- Softmax/logistic layer
- Output layer



1) Input Layer

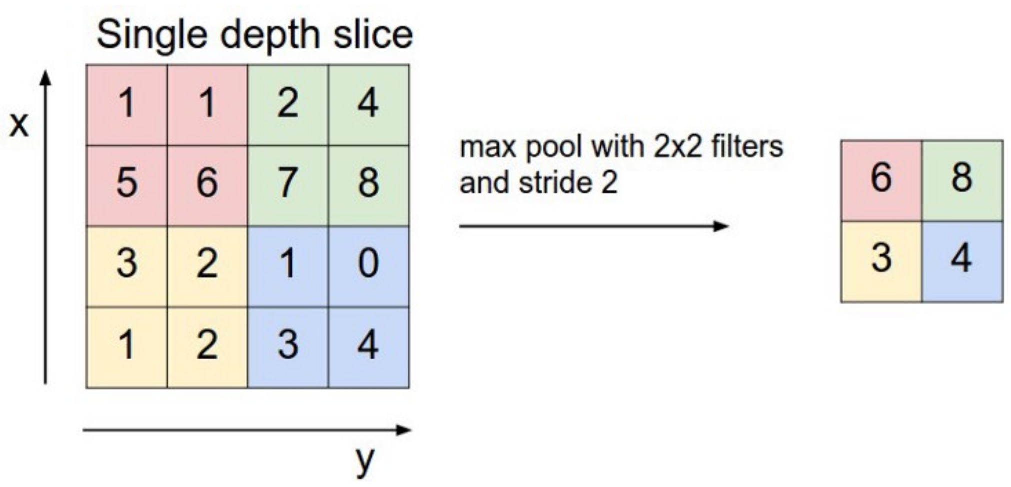
Input layer in CNN should contain image data. Image data is represented by three dimensional matrix as we saw earlier. You need to reshape it into a single column. Suppose you have image of dimension 28 x 28 =784, you need to convert it into 784 x 1 before feeding into input.

2) Convo Layer

Convo layer is sometimes called feature extractor layer because features of the image are get extracted within this layer. First of all, a part of image is connected to Convo layer to perform convolution operation as we saw earlier and calculating the dot product between receptive field(it is a local region of the input image that has the same size as that of filter) and the filter. Result of the operation is single integer of the output volume. Then we slide the filter over the next receptive field of the same input image by a Stride and do the same operation again. We will repeat the same process again and again until we go through the whole image. The output will be the input for the next layer. Convo layer also contains ReLU activation to make all negative value to zero.

3) Pooling layer

Pooling layer is used to reduce the spatial volume of input image after convolution. It is used between two convolution layer. If we apply FC after Convo layer without applying pooling or max pooling, then it will be computationally expensive and we don’t want it.Max-pooling is a technique used to reduce the dimensions of an image by taking the maximum pixel value of a grid. This also helps reduce overfitting and makes the model more generic. The example below show how 2 x 2 max pooling works



4) Fully Connected Layer(FC)

Fully connected layer involves weights, biases, and neurons. It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.We use bpa here to train our weights and get the desired output during testing of the network

5) **Softmax / Logistic Layer**

Softmax or Logistic layer is the last layer of CNN. It resides at the end of FC layer. Logistic is used for binary classification and softmax is for multi-classification.

6) Output Layer

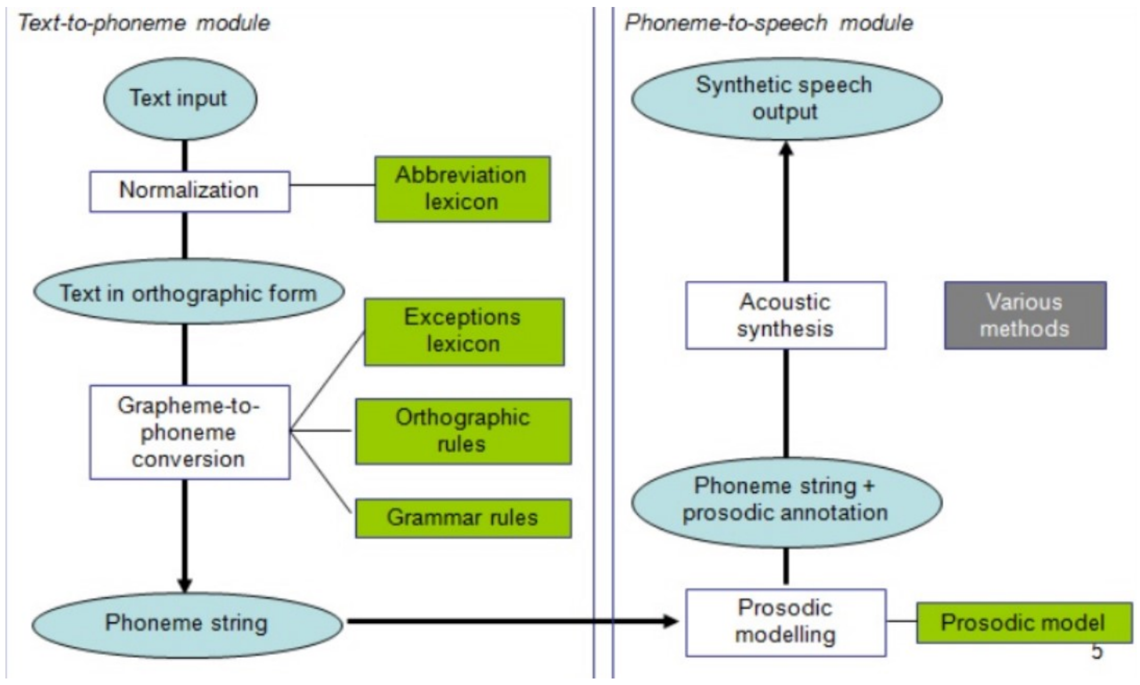
Output layer contains the label of the classes and gives the predicted label.

- ## Result of part 1

After the building of the neural network is done. 60% images i.e, 3,000 images are used to train the network. The system was then made to interpret 2,000 unkown images and got an accuracy of 78.4% (1568/2000)

▪ Part 2 : Text to speech

A text-to-speech system is composed of two parts: a front-end and a back-end. The front-end has two major tasks. First, it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization, pre-processing, or tokenization. The front-end then assigns phonetic transcriptions to each word, and divides and marks the text into prosodic units, like phrases, clauses, and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end—often referred to as the synthesizer—then converts the symbolic linguistic representation into sound which is then imposed on the output speech.



STEP : 1 Text identification

- The given string is broken to characters
- Each character is identified using UNICODE

STEP : 2 Conversion of text to grapheme

- Each alphabet has its own way of pronunciation based on its position
- These are classified different classes

  Ex: In the words “HONEST” and “HORSE” the way of pronouncing the
. letter H is different and is stored as H1a and H2c

  Ex: The letters “s” and “h” have separate Phonemes but when they appear
. together the acoustics are different.

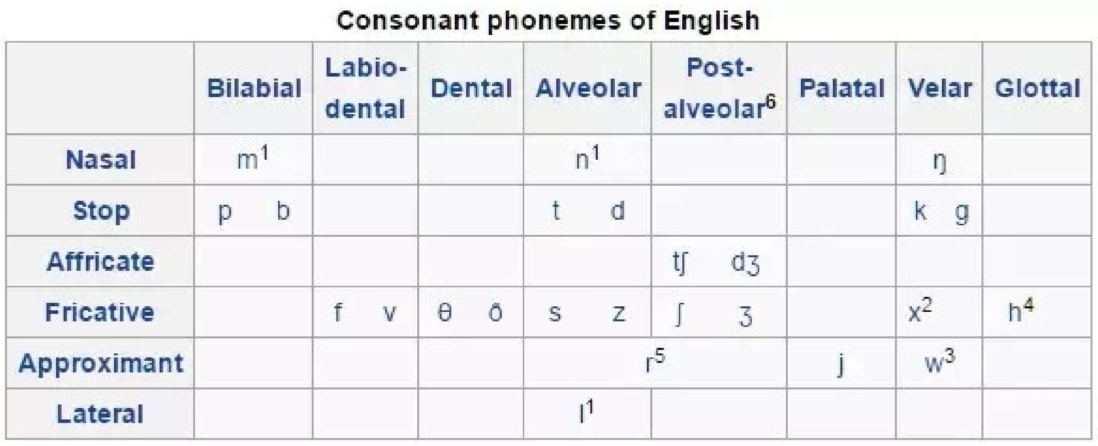
- All these are plotted on a graph called Grapheme

STEP 3: Grapheme to Phoneme

- Each grapheme has a corresponding phoneme for it.
- Based on the position of the character the phones are appended to form a phoneme string

STEP 4: Gramatical and Acoustic Synthesis

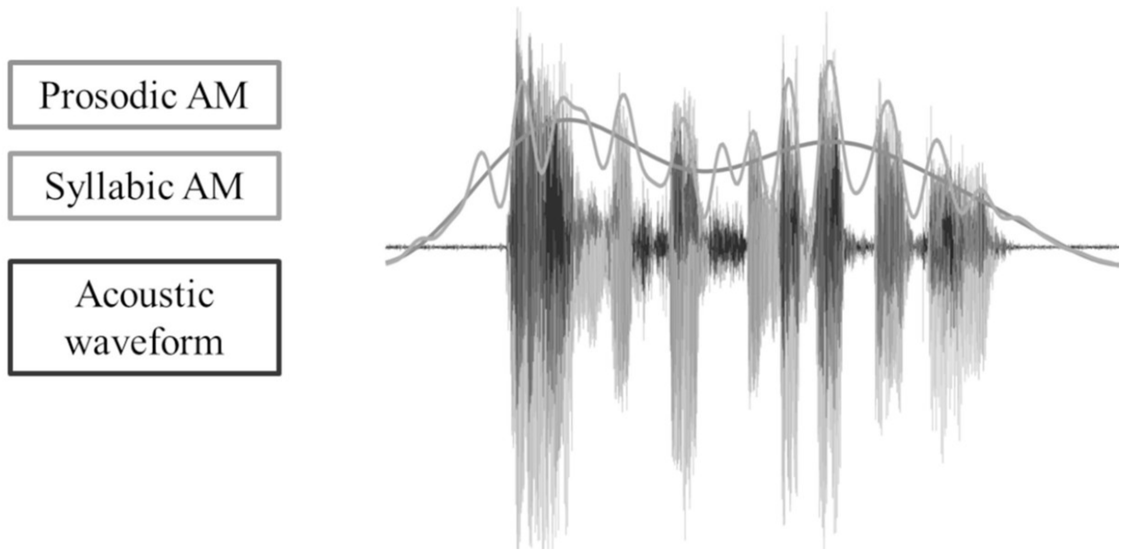
- The last step is for grammatical adjustments and acoustic synthesis
- The final word audio is generated .



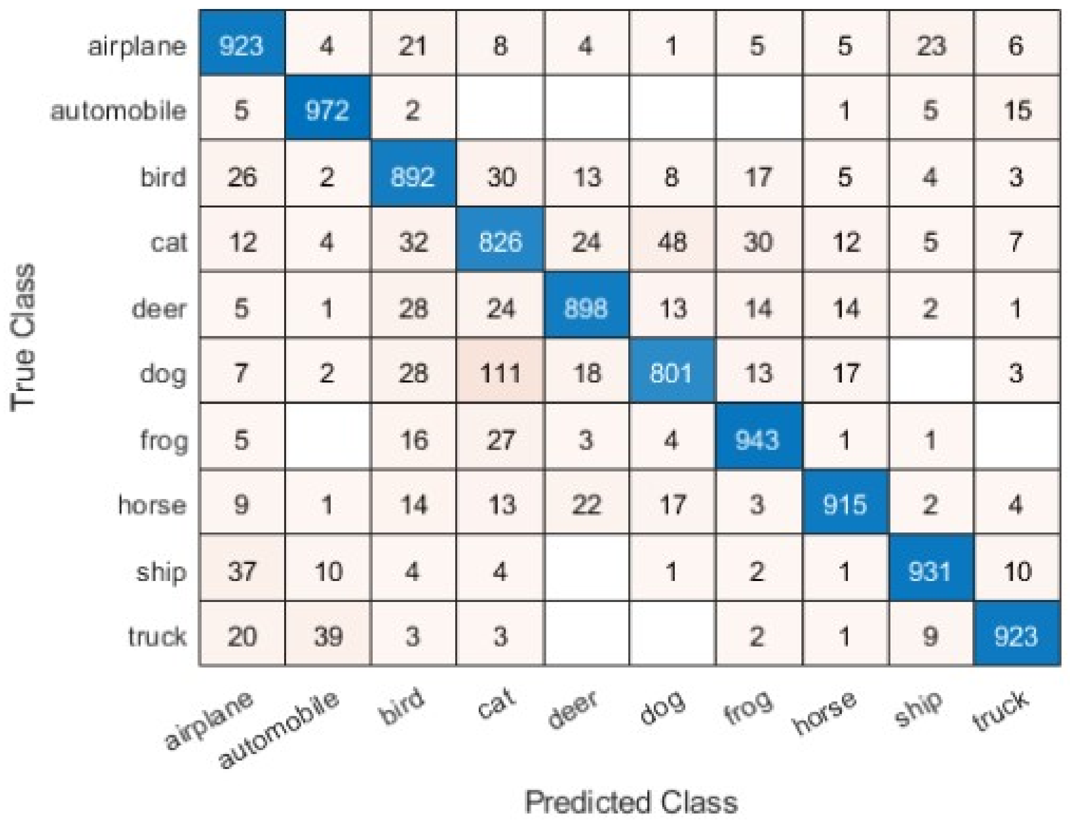
Consonant phonemes of English

	Bilabial	Labio-dental	Dental	Alveolar	Post-alveolar[6]	Palatal	Velar	Glottal
Nasal	m[1]			n[1]			ŋ	
Stop	p b			t d			k g	
Affricate					tʃ dʒ			
Fricative		f v	θ ð	s z	ʃ ʒ		x[2]	h[4]
Approximant				r[5]		j	w[3]	
Lateral				l[1]				

Comparison between the Prosodic , Syllabic and Acoustic waveform



The confusion Matrix

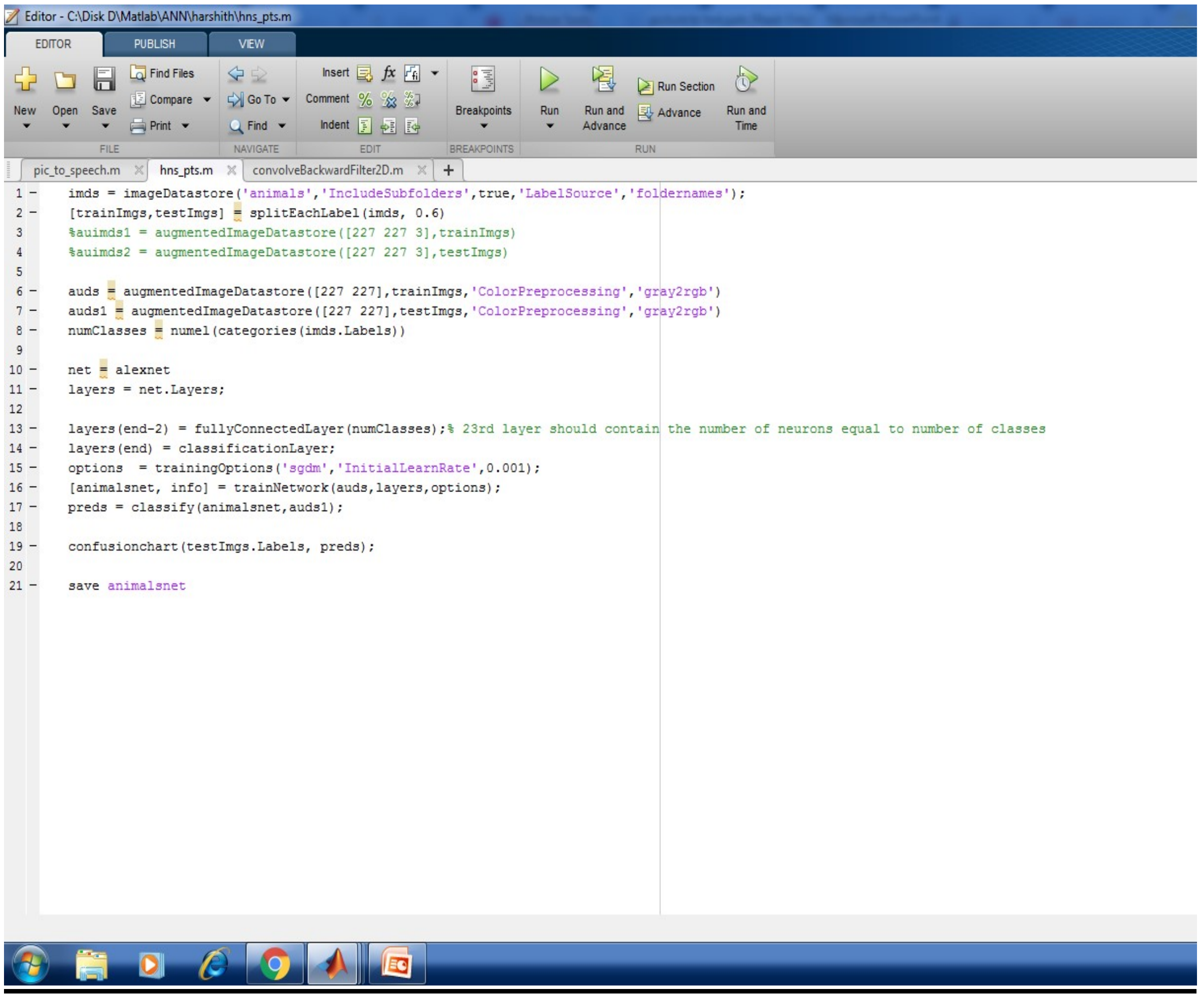


	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	923	4	21	8	4	1	5	5	23	6
automobile	5	972	2					1	5	15
bird	26	2	892	30	13	8	17	5	4	3
cat	12	4	32	826	24	48	30	12	5	7
deer	5	1	28	24	898	13	14	14	2	1
dog	7	2	28	111	18	801	13	17		3
frog	5		16	27	3	4	943	1	1	
horse	9	1	14	13	22	17	3	915	2	4
ship	37	10	4	4		1	2	1	931	10
truck	20	39	3	3			2	1	9	923

Future Scope

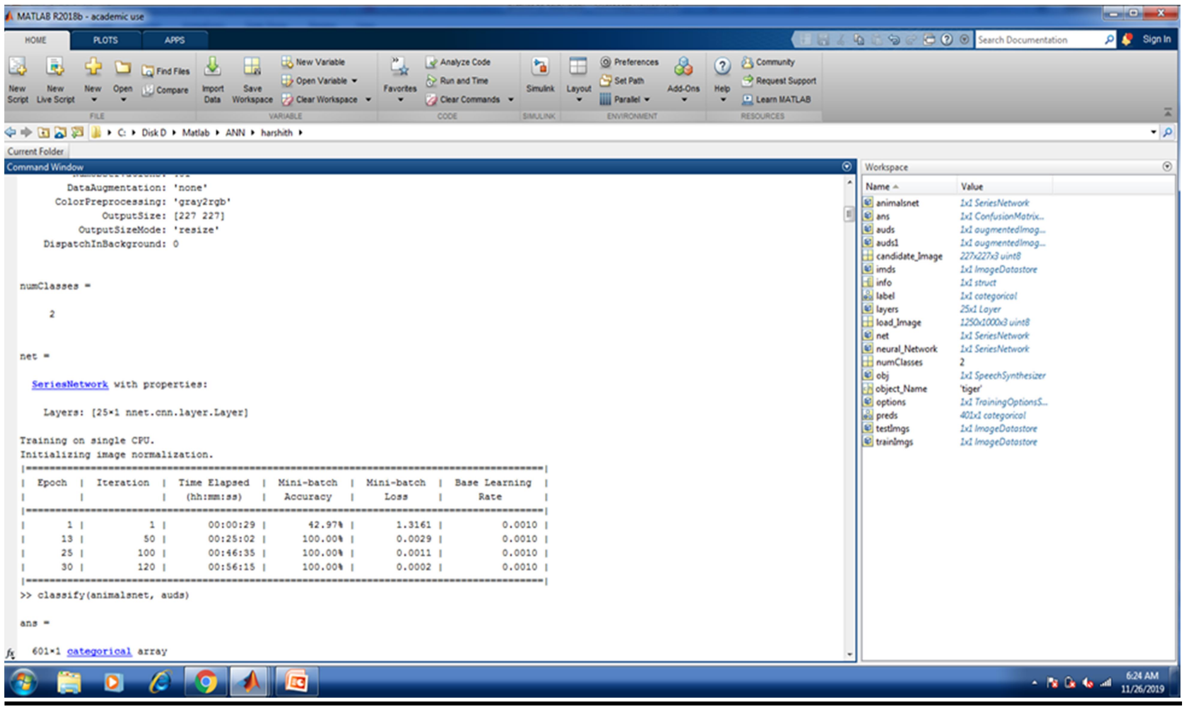
- Can be integrated with Google lens and other image recognition devices which identifies objects and speaks it out.
- Can be used to identify sign boards and symbols
- Can be used to decipher other languages
- Can be used in species identification and biological and medical research
- Can be used to aid the blind and differently in reading out texts and identification of images

Code for training images



```
imds = imageDatastore('animals','IncludeSubfolders',true,'LabelSource','foldernames');
[trainImgs,testImgs] = splitEachLabel(imds, 0.6)
%auimds1 = augmentedImageDatastore([227 227 3],trainImgs)
%auimds2 = augmentedImageDatastore([227 227 3],testImgs)

auds = augmentedImageDatastore([227 227],trainImgs,'ColorPreprocessing','gray2rgb')
auds1 = augmentedImageDatastore([227 227],testImgs,'ColorPreprocessing','gray2rgb')
numClasses = numel(categories(imds.Labels))

net = alexnet
layers = net.Layers;

layers(end-2) = fullyConnectedLayer(numClasses);% 23rd layer should contain the number of neurons equal to number of classes
layers(end) = classificationLayer;
options  = trainingOptions('sgdm','InitialLearnRate',0.001);
[animalsnet, info] = trainNetwork(auds,layers,options);
preds = classify(animalsnet,auds1);

confusionchart(testImgs.Labels, preds);

save animalsnet
```

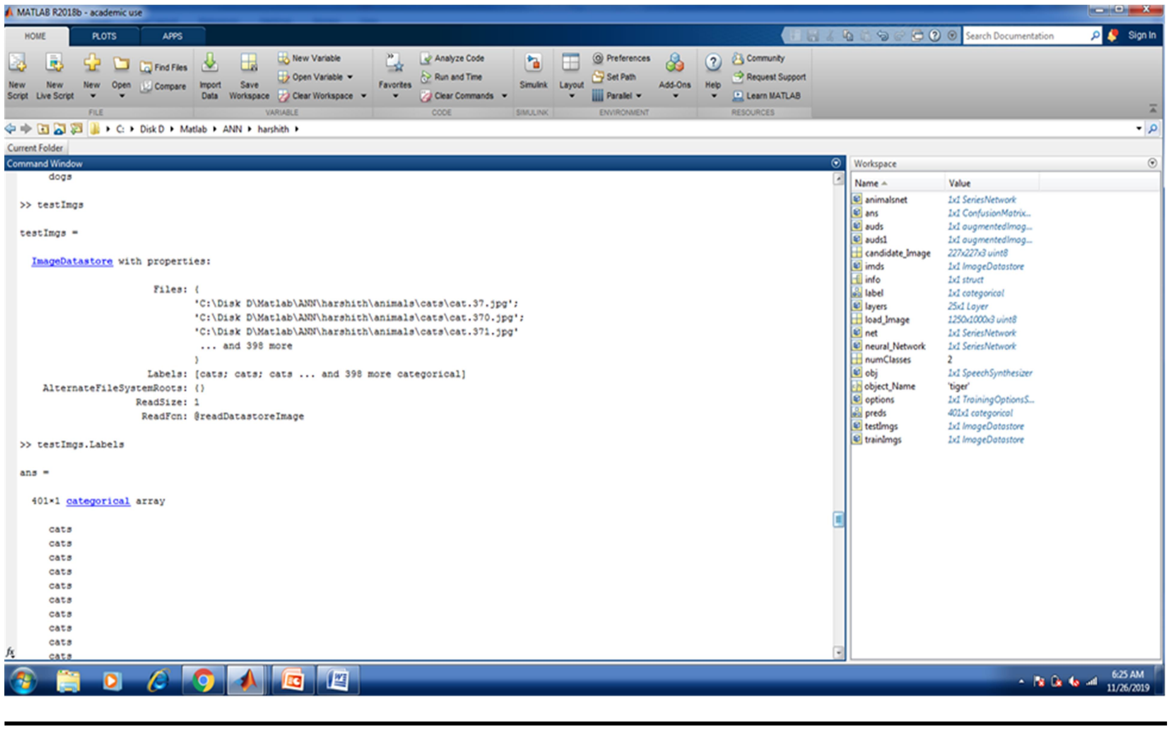
Training of images


MATLAB R2018b - academic use
HOME
PLOTS
APPS
C: ▸ Disk D ▸ Matlab ▸ ANN ▸ harshith ▸
Current Folder
Command Window
Workspace
6:24 AM
11/26/2019

```
        DataAugmentation: 'none'
      ColorPreprocessing: 'gray2rgb'
              OutputSize: [227 227]
          OutputSizeMode: 'resize'
    DispatchInBackground: 0


numClasses =

     2


net =

  SeriesNetwork with properties:

    Layers: [25×1 nnet.cnn.layer.Layer]

Training on single CPU.
Initializing image normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:29 |       42.97% |       1.3161 |          0.0010 |
|      13 |          50 |       00:25:02 |      100.00% |       0.0029 |          0.0010 |
|      25 |         100 |       00:46:35 |      100.00% |       0.0011 |          0.0010 |
|      30 |         120 |       00:56:15 |      100.00% |       0.0002 |          0.0010 |
|========================================================================================|
>> classify(animalsnet, auds)

ans =

  601×1 categorical array
```

Name	Value
animalsnet	1x1 SeriesNetwork
ans	1x1 ConfusionMatrix...
auds	1x1 augmentedImag...
auds1	1x1 augmentedImag...
candidate_Image	227x227x3 uint8
imds	1x1 ImageDatastore
info	1x1 struct
label	1x1 categorical
layers	25x1 Layer
load_Image	1250x1000x3 uint8
net	1x1 SeriesNetwork
neural_Network	1x1 SeriesNetwork
numClasses	2
obj	1x1 SpeechSynthesizer
object_Name	'tiger'
options	1x1 TrainingOptionsS...
preds	401x1 categorical
testImgs	1x1 ImageDatastore
trainImgs	1x1 ImageDatastore

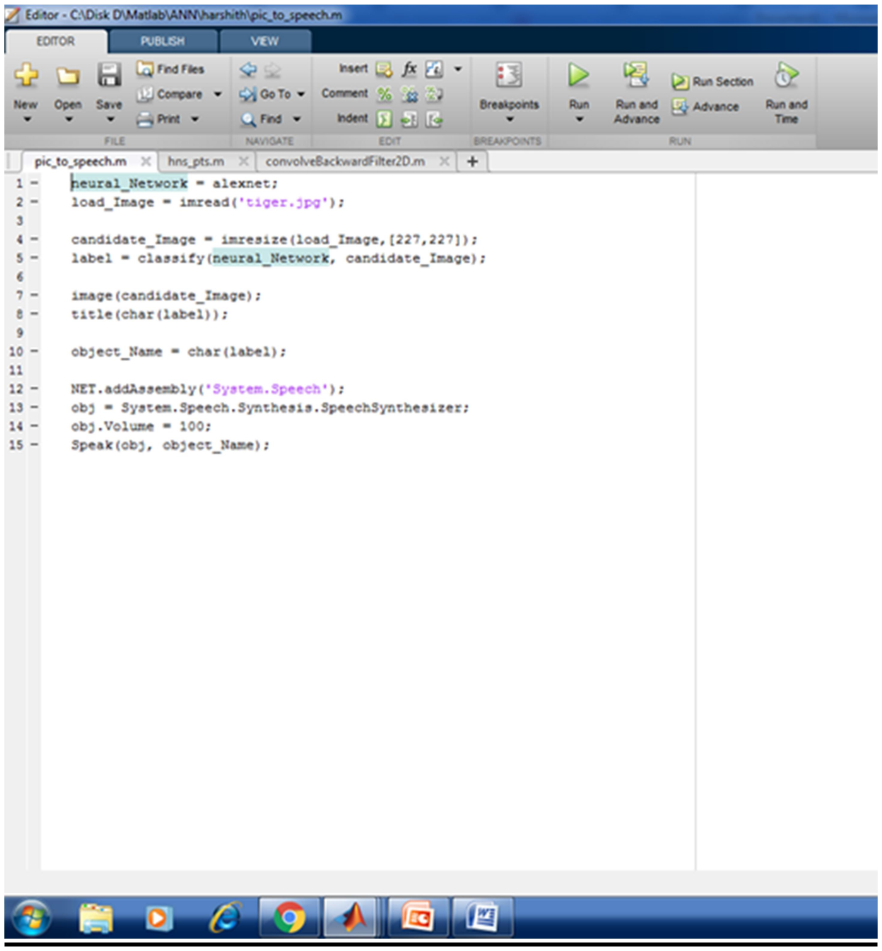
Screenshot of training image


MATLAB R2018b - academic use
HOME
PLOTS
APPS
C: ▸ Disk D ▸ Matlab ▸ ANN ▸ harshith ▸
Current Folder
Command Window
Workspace
6:25 AM
11/26/2019

```
    dogs

>> testImgs

testImgs =

  ImageDatastore with properties:

                       Files: {
                              'C:\Disk D\Matlab\ANN\harshith\animals\cats\cat.37.jpg';
                              'C:\Disk D\Matlab\ANN\harshith\animals\cats\cat.370.jpg';
                              'C:\Disk D\Matlab\ANN\harshith\animals\cats\cat.371.jpg'
                               ... and 398 more
                              }
                      Labels: [cats; cats; cats ... and 398 more categorical]
    AlternateFileSystemRoots: {}
                    ReadSize: 1
                     ReadFcn: @readDatastoreImage

>> testImgs.Labels

ans =

  401×1 categorical array

    cats
    cats
    cats
    cats
    cats
    cats
    cats
    cats
    cats
    cats
```

Name	Value
animalsnet	1x1 SeriesNetwork
ans	1x1 ConfusionMatrix...
auds	1x1 augmentedImag...
auds1	1x1 augmentedImag...
candidate_Image	227x227x3 uint8
imds	1x1 ImageDatastore
info	1x1 struct
label	1x1 categorical
layers	25x1 Layer
load_Image	1250x1000x3 uint8
net	1x1 SeriesNetwork
neural_Network	1x1 SeriesNetwork
numClasses	2
obj	1x1 SpeechSynthesizer
object_Name	'tiger'
options	1x1 TrainingOptionsS...
preds	401x1 categorical
testImgs	1x1 ImageDatastore
trainImgs	1x1 ImageDatastore

Text to speech : Testing


Editor - C:\Disk D\Matlab\ANN\harshith\pic_to_speech.m
EDITOR
PUBLISH
VIEW
pic_to_speech.m
hns_pts.m
convolveBackwardFilter2D.m

```
neural_Network = alexnet;
load_Image = imread('tiger.jpg');

candidate_Image = imresize(load_Image,[227,227]);
label = classify(neural_Network, candidate_Image);

image(candidate_Image);
title(char(label));

object_Name = char(label);

NET.addAssembly('System.Speech');
obj = System.Speech.Synthesis.SpeechSynthesizer;
obj.Volume = 100;
Speak(obj, object_Name);
```

Conclusion

The use of CNN for both image recognition and speech generation produces good accuracy and can be used as the base to build various other models and applications. The network uses BPA to adjust its weight accordingly based on the desired outputs that we give to it. So by using this technique of transfer learning manpower required for the model becomes less and the results obtained have high accuracy. The final output of our project yielded an audio which gave the label name of the corresponding image after recognition with an accuracy of 78.4%.

Acknowledgment

I would like to take this opportunity to thank our mentor Prof. Karpagavalli for her continuous guidance and for giving us this opportunity to do this project as part of our Artificial Neural Network elective, which helped us understand the subject better and also apply it onto practical aspects of life.

Refrences

- https://ieeexplore.ieee.org/document/56867
- https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529
- https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4
- https://arxiv.org/pdf/cs/9811032.pdf
- https://Neural%20Networks%20for%20T2S%20phoneme%20recognition%20(1).pdf