
Audio Feature Extraction Using Transfer Learning

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Technology
in
Degree in Computer Science Engineering

By

Sandeepan Adhikari
(D/17/CS/003)

Mayank Kr. Mishra
(D/17/CS/110)

Hibu Talyang
(D/17/CS/113)

Under the Guidance of

Prof Kapang Legoh
(Associate Professor CSE)
(Project Supervisor)



**Department of Electronics & Communication Engineering North Eastern
Regional Institute of Science and Technology**

(Deemed to be University Under MHRD, Govt. of INDIA) Nirjuli,
Arunachal Pradesh – 791109

January 2022



पूर्वोत्तर क्षेत्रीय विज्ञान एवम् प्रौद्योगिकी संस्थान
North Eastern Regional Institute of Science & Technology

(Deemed to be University u/s 3 of the UGC Act, 1956)
Under the Ministry of Education, Govt. of India

निर्जुली - ७९१ १०९ (ईटानगर)
अरुणाचल प्रदेश, भारत

Nirjuli - 791 109 (Itanagar)
Arunachal Pradesh, India

CERTIFICATE

This is to certify the project report entitled, “Audio Feature Extraction Using Transfer Learning “, submitted in partial fulfilment of the requirements for the award of Degree Certificate of Bachelor of Technology in Computer Science Engineering to the North Eastern Regional Institute of Science and Technology, Nirjuli, Arunachal Pradesh, is a record of Bonafede academic work carried out by

Mr. Sandeepan Adhikari

D/17/CS/003

Mr. Mayank Kr. Mishra

D/17/CS/110

Mr. Hibu Talyang

D/17/CS/113

Under our guidance and supervision, the results embedded in the project work have not been submitted to any other University or Institute for the award of any Diploma or Degree.

Prof. Kapang Legoh
(Associate Professor CSE)
(Project Supervisor)

Department of Computer Science Engineering NERIST

Mr. Ashwini Kumar Patra
Project Coordinator
Assistant Professor
Department of CSE
NERIST

Dr. Marjit Singh
Head of the Department
Professor
Department of CSE
NERIST

Phone : (0360) 2257401 - 11(O)
Fax : (0360) 2257872 / 2258533
Gram : NERIST, Nirjuli

Email : @nerist.ernet.in
: @nerist.ac.in
Website : www.nerist.ac.in

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to Prof. KapangLegoh Associate Professor, Department of Computer Science and Engineering North Eastern Regional Institute of Science and Technology, whose role as our project guide was invaluable for the project. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us. We are also indebted to our Course Coordinator Prof Ashwini Kumar Patra for his unconditional help and inspiration,

DECLARATION

We hereby declare that the project work entitled, “*Audio feature extraction using transfer learning*” submitted to the Computer Science Engineering Department of NERIST, is a record of an original work done by us under the guidance of Prof. Kapang Legoh, Associate Professor, Department of Computer Science Engineering, NERIST, and this project is submitted in the partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology.

Sandeepan Adhikari
(D/17/CS/003)

Mayank Kr. Mishra
(D/17/CS/110)

Hibu Talyang
(D/17/CS/113)

TABLE OF CONTENTS

TOPIC	PAGE NO
Acknowledgement.....	iii
Declaration.....	iv
Abstract.....	vi
List of Figures	ix
Chapter1:Introduction.....	1
Chapter 2:Literature Survey.....	3
Chapter 3. Methodology	
3.1 What is Transfer Learning?	
3.2 Why use TransferLearning?	
3.3 Use of Transfer Learning in Speech-to-textprogram	
3.4 Implementation of transfer Learning	
3.5 FeatureExtraction	
3.6 FineTuning	
Chapter 4. Conclusion.....	

ABSTRACT

In this era of technological advancement, cloud input base services and speech recognition technologies are making our lives more and more easier. Nowadays almost everything can be or one the way to be controlled by giving voice commands. We don't have to type or give instructions by hands, instead we give voice commands to the application and the desired functionality is achieved. Modern applications nowadays use cloud inputbase services to store user input in cloud storage available on the internet. Thus eliminating the problem of installing hardware to increase storage capacity of the system. Using these technologies and creating something new and unique is really exciting. We will be building a website where users can upload their media files to a cloud storage by using our applications as an interface between the user and the cloud storage. Thus solving the problem of storing large media files in our local systems. We will also use voice recognition technology to try and make a media player which can be controlled by giving voice commands, much more flexible to use than any other traditional media player.

LIST OF FIGURES

FIGURE NO.	FIGURE	PAGE NO.
1.	Idea of Input Transfer	iii
2.	Feature Extraction	vii

CHAPTER 1: INTRODUCTION

Discourse can be considered as series of acoustic signs which convey some clarity. The hints of communicated in language have been learned at two distinct levels:

- a. Phonetic parts of verbally expressed words, e.g., vowel and consonant sounds
- b. Coustic wave patterns.

A language can be separated into a tiny number of fundamental sounds, called phonemes. An acoustic wave is a grouping of changing vibration designs.

Programmed discourse acknowledgment (ASR) framework can be characterized as the ability of machine to imitate human discourse acknowledgment (HSR) framework knowledge to distinguish discourse and presents its result in type of message or words or sentences. Separated word ASR for fixed vocabularies has been effectively carried out utilizing HMM, ANN and SVM yet experiences absence of versatility to different dialects and expansion in intricacy as number of vocabularies increments. phonemes, the littlest unit of human discourse sounds are evidently more helpful to address the fundamental structure block for cross-language planning.

The Speech to Text version relies upon Deep Learning. Removing the features of various thousand sound reports and setting them up is computationally exorbitant. So We use a method called move Learning. At Transfer Learning versions are arranged then saved so later on the Saved version could be used for any endeavor. Like Deep Learning version Transfer Learning furthermore have a colossal number of neurons and many layers.

Nevertheless, In move Learning the layers are frozen as the version has been currently arranged. We can in like manner add new layers with the version really planning that close by ahead of time available realities, we can deal with new input of our own to make the version more perfect. For giving this Speech to Text Program we used Speech Recognition Library. The library is openly delivered and at this point contains a version ready. The pre-arranged version concentrates mfc or Mel-frequencycepstrum.

In sound dealing with, the Mel-repeat cepstrum is a depiction of the transient power scope of a sound, considering an immediate cosine change of a log power range on a nonlinear mel size of repeat. In short this exceptional part helps us with isolating between sound waves. We use this strong version from Speech Recognition. We simply import the library. Then we similarly Import Pydub. Pydub is a Python library to work with so to speak .wav records. By using this library we can play, split, consolidate, change our . wav sound records. As of now an enormous piece of the sound record comes in Mp3 plan.

Consequently, we Import ffmpeg Which changes over our Mp3 record to a wav report. As of now when we insert the mp3 record our program transforms it over totally to wav archive then, at that point, does the talk to message task on top of the Pre arranged Speech Recognition library and returns us the message that sound report contained.

Chapter 2: LITERATURE SURVEY

During 1960s, IBM was absolutely getting everything moving with a gigantic talk affirmation project that provoked a very compelling huge language isolated word record system and a couple of little language control structures. In the middle nineties IBM's Voice Type, Dragon Systems' Dragon Dictate, and Kurzweil Applied Intelligence's Voice Plus were the notable PC talk affirmation things accessible. These packs consistently required extra high level sign dealing with PC gear.

They were around 90% careful for general correspondence and required a short pause between words. They were called discrete talk affirmation structures. Today the term isolated word is more typical. Various procedures/computations have been made by talk experts as part extractions and classifiers in talk affirmation. In most of the scholarly works drove, HMM is continually used as the benchmark for ASR since this verifiable system has every one of the reserves of being the best classifier due to its suitability to follow the remarkable in talk anyway it requires steady effort in exhibiting all phonemes in a particular language.

Going before the part extraction, the front-end preprocessing especially phoneme division is a critical part which fills in as the essential affirmation unit. The multifaceted design can go from perceiving voiced and unvoiced angles to as jumbled as recognizing unequivocal phonemes. For phonemes, something past end-point revelation cause issue in talk taking care of. End-point just separate quietness and talk signal and the accuracy reduces as unvoiced sounds occurring close to the beginning or end of an articulation.

Tanaka presented that phoneme affirmation system include three crucial parts as following:

- a. phonemic incorporate extraction, depiction and decision of standard phonemic categories(SPC)
- b. differentiate
- c. label

If acknowledged phonemes as portrayed in phonemic as classes to be recalled that, it is genuinely difficult. This is a result of its assortments in coarticulation effects and speaker's dependence. Another way is by including VCV or CVC as affirmation unit yet will cause incalculable SPC and a couple of monotonous cases.

Chapter 3: METHODOLOGY

3.1 What is Transfer Learning?

Everything revolves around utilizing the learnings from pre-prepared version so we don't need to prepare the new version from ground up. It makes the errand simpler for us to branch into explicit streams.

The pre-prepared versions are now prepared on enormous inputsets by researchers or nonconformist in the counterfeit learning scene. The loads got from the versions can be reused in other comparative or rather unambiguous errands. Frequently the current loads are as of now somewhat wonderful in its own particular manner and we better not play with the current loads.

These versions can be utilized to straightforwardly foresee the new undertakings and coordinate into cycle of preparing the new version, our commitment being just the last layers where we diverted it towards a particular errand, which remembers our base versions for another version which prompts lower preparing time and furthermore lesser speculation mistake.

It is especially helpful when we have little preparation inputset. For this situation, we may for instance, use loads from base versions to introduce loads of new version. Move learning will be applied to discourse acknowledgment framework that would do unmistakable assignments.

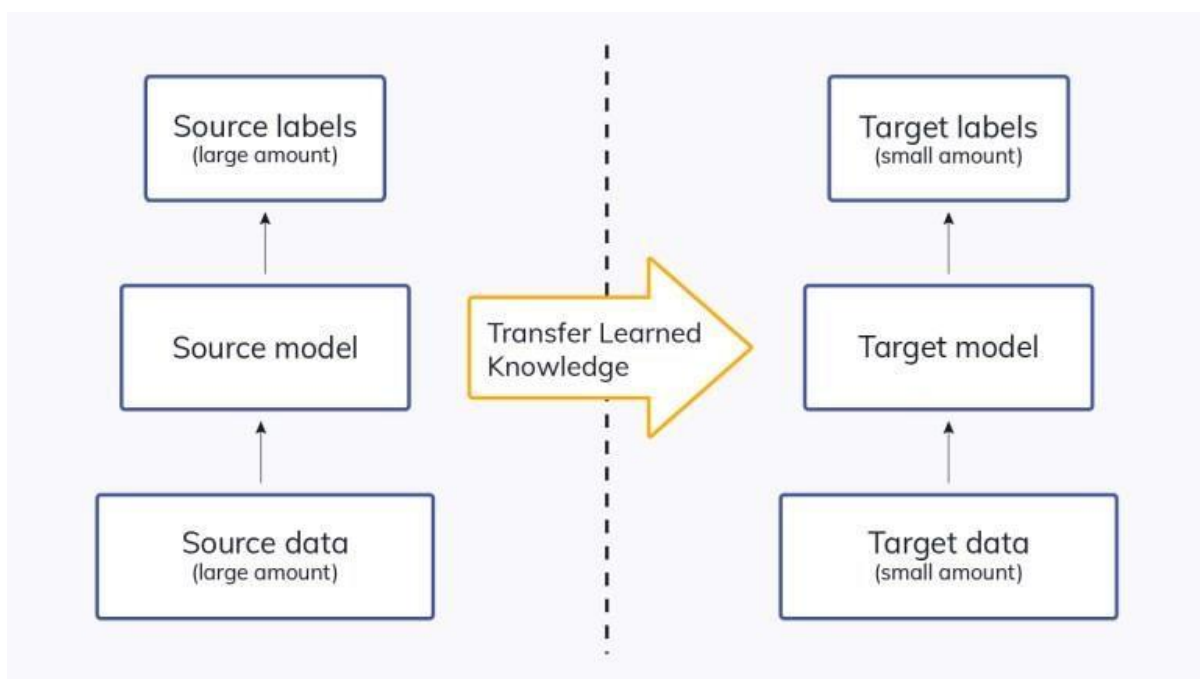


Fig. 1. The Idea of Input Transfer

The motivations behind applying move learning is that they are uncommonly generally valuable sort with the end result of being used in other authentic applications.

For example:

Versions ready on the large opensourced base versions can be used in authentic picture game plan issues. This is because the inputset contains in excess of 100 classes. We can use these versions and conform to permit them to orchestrate bugs.

Describing message requires input on word depiction in some vector space. We can get ready vector depictions ourself. The test we could look here is that it could have not have a satisfactory number of input to let embedding have the arrangement. Besides, getting ready will consume an enormous lump of the day. For this present circumstance, We can use a pre-arranged word embedding like free programming projects to rush our developmental procedure.

3.2 Transfer Learning benefits?

- Planning versions with high accuracy requires a lot of input. We can recollect it from version that the inputset of the open library contains pictures more than 1 million. In certified world, it is most likely not going to have such a significant set of inputset.
- Resulting to expecting having that kind of inputset, we could regardless not be having the resources expected to set up the version with a significant game plan of inputset. Subsequently move learning seems, by all accounts, to be genuine if we don't have to figure resources expected to plan versions on massive inputsets in like manner.
- Whether or not we truly need to enlist resources accessible to us, we really need to believe that days and months will set up a version as this one. Hence using a pre-arranged version will save us our valuable time.

3.3 Use of Transfer Learning in Speech-to-text program

To deliver such a Speech to Text Program we uses Speech Recognition Library. This library is freely delivered and at this point contains a version ready. This pre-arranged version extracted mfc or Mel-repeat cepstrum.

Sound taking care of have the Mel-repeat cepstrum which is the depiction of brief range power scope of sound, considering straight cosine change of log power range on non-direct Mel-size of this level of repeat. In short design one of this noteworthy component helps us with requesting among sound and waves. We need to use this strong version from Speech Recognition.

We basically import this Library. Then, at that point, we truly need to Import Pydub. pydub here is a Python library all together permitted it to work with figuratively speaking .wav file. By using this kind of library we edit, merge, play our .wav sound reports. As of now most of these sound records goes with

course of action of MP3. So We expected to Import ffmpeg which changed over our Mp3 report to much required wav record for convenience. As of now when we expected to install the mp3 record our program exchanged it over totally to wav report then the talk to message task on top of this given Pre arranged Speech Recognition library and returns message which contained the sound archive.

3.2 Implementation of Transfer Learning

- Gotten the pre trained version
- The incredibly first advance toward get this pre-arranged version is that we like to use by any stretch of the imagination for our anxiety. The various wellsprings of pre-arranged versions are covered in the different portion.

- Make a based version
- Normally, our underlying move toward fire up set up base version using one of our designs like ResNet or Xception. We can moreover then again download our pre-arranged weight. Downloading loads, we truly need to use this plan to permit it to set up our version from ground. Survey the base version which for the most part contains more number of units in clear outcome layer as relationship with our requirement. After making this base version, we, subsequently, need to dispose of the last outcome layer. Later on, we will add a last outcome layer that is feasible with our anxiety.

- Freeze layers
- Freezing the layers from the pre-arranged version is fundamental. This is because we don't keep up with that the heaps in those layers ought to be once again introduced. If they are, we will lose all the finding that has proactively happened. This will be equivalent to setting up the version without any planning.

•

Add new workable layers

The resulting stage is to add new workable layers that will change old components into assumptions on the new

inputset. This is huge in light of the fact that the pre-arranged version is stacked without the last outcome layer.

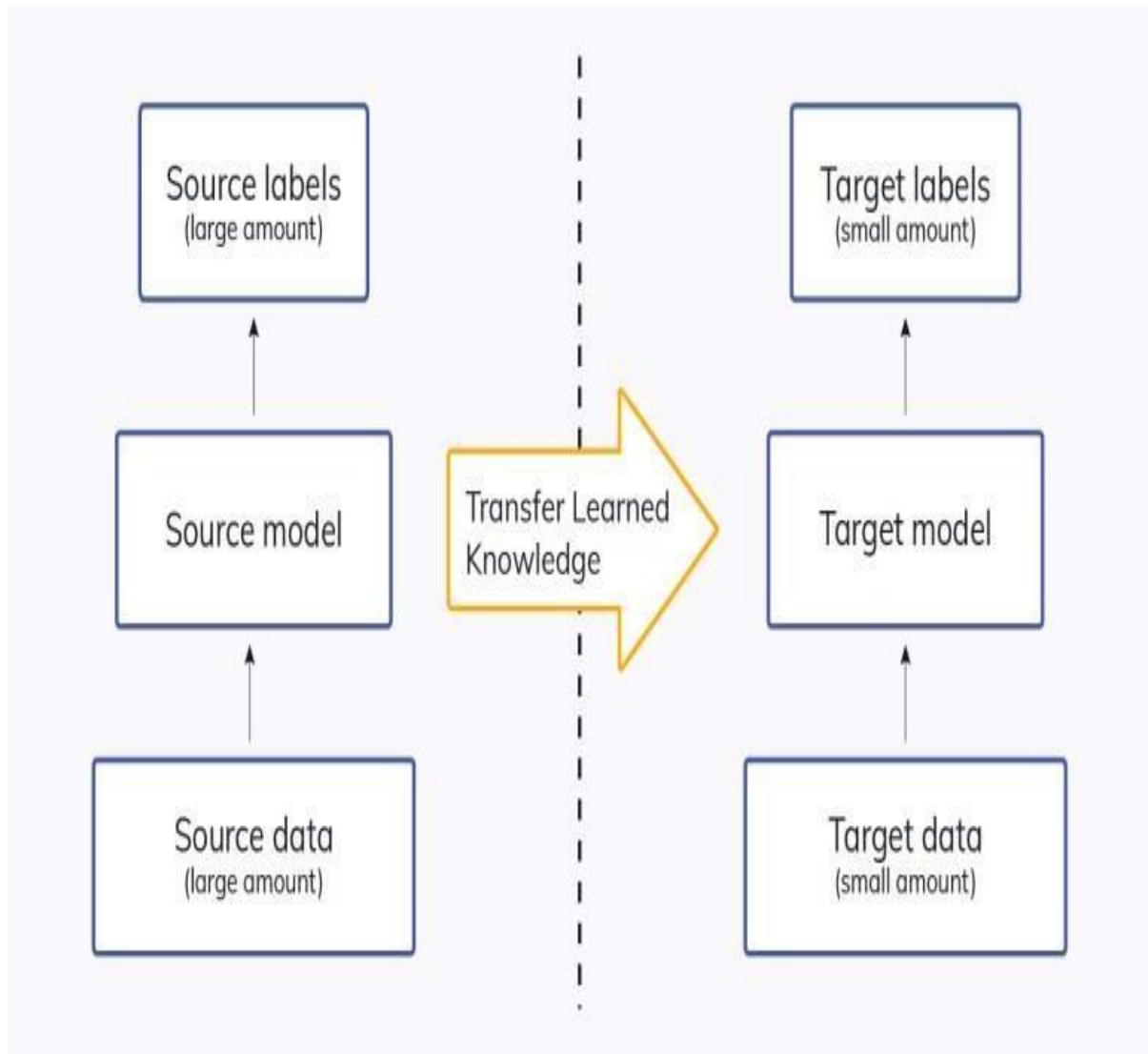


Fig. 1. The Idea of Input Transfer

➤ Train the new layers on the new, smaller inputset

Recall that the pre-arranged version's last outcome will most likely be special according to our ideal outcome for our version. For example, pre-arranged versions ready on the opensourcedinputset will yield many classes. Regardless, our version might just have a couple hundred classes. For this present circumstance, we want to set up the version with one more outcome layer set up.

In this manner, we will add a couple of new thick layers anyway we see fit, specifically, a last thick layer with units contrasting with the amount of results expected by our version.

➤ Improving the version via fine-tuning

Whenever we have done the previous step, we will have a version that can make conjectures on our inputset. On the other hand, we can chip away at its show through tweaking. Aligning is done by defrosting the base version or part of it and setting up the entire version again generally inputset at an astoundingly low learning rate. The low learning rate will extend the display of the version on the new inputset while hindering overfitting.

The learning rate should be low considering the way that the version is extremely gigantic while the inputset is close to nothing. This is a recipe for overfitting, thusly the low learning rate. Recompile the version at whatever point we have carried out these enhancements with the objective that they can create results. This is because the approach to acting of a version is frozen whenever we call the request capacity. That suggests that we want to call the request capacity again whenever we want to change the version's approach to acting. The resulting stage will be to set up the version in the future while noticing it through callbacks to promise it doesn't overfit.

3.1 Feature Extraction from the final layers

For this present circumstance, the consequence of the layer before the last layer is dealt with as commitment to another version. The goal is to use the pre-arranged version, or a piece of it, to pre-process pictures and get principal features.

Then, we pass these features to another classifier — try not to retrain the base version. The pre-arranged convolutional mind network at this point has features that are make a big difference to the primary work.

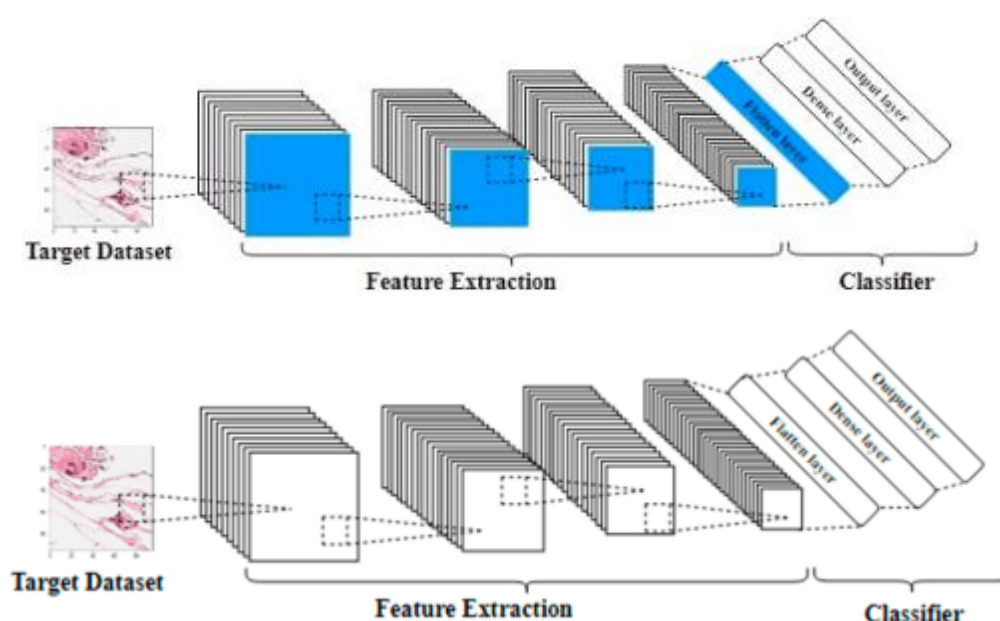


Fig 2. Feature Extraction

Notwithstanding, the pre-prepared version's last part doesn't move over in light of the fact that it's well defined for its inputset. In this way, we need to construct the last piece of our version to accommodate our inputset. In any case, the pre-arranged version's last part doesn't move over considering the way that it's obvious for its inputset. Along these lines, we want to build the last piece of our version to oblige our inputset.

In the ordinary language dealing with space, pre-arranged word embedding can be used for feature extraction. The word embeddings help to place words in their right circumstance in a vector space. They give significant information to a version since they can contextualize words in a sentence. The key objective of word embeddings is semantic cognizance and the association between words. In this manner, these word embeddings are task pragmatist for standard language issues.

3.2 Fine Tuning

Right when our new classifier is ready, we can use changing in accordance with work on its precision. To do this, we defrost the classifier, or part of it, and retrain it on new input with a low learning rate.

Tweaking is essential to make feature depictions from the base version (got from the pre-arranged version) more pertinent to we express task.

3.3 Features used to train the version

Acoustic features: Acoustic features is a collective form of age, gender, location and many more reasons which causes human voices to differ from person to person. All of this refers to the acoustic features of human speech

Linguistic Features: Linguistic features defines the probability of a word's likeliness to stay at a sentence. For example, The words 'Red' and 'Read' is pronounced in similar way. But we use the color 'red' word as for example : The ball is red
And we might use the word 'read' like this : I have read a book.
In both cases the words are pronounced similarly. However it's the linguistic features which helps us to differentiate between this 2 words.

Now we have to teach this 2 features to our AI version as well.

How will we teach acoustic features to our version: To teach acoustic features of speech to our AI version we use an acoustic version. This acoustic version is nothing but an recurrent neural network version which have more than 70 million parameter and have been trained with more than 10 thousand hours of human voice.

How will we teach Linguistic features to our version: To teach linguistic features of speech to our AI version we use rescoring algorithm. It works something like this, The rescoring algorithm uses CTC beam search which looks at the probability if the sentence is going to contain the word “Read” or “Red”

Input preprocessing: In Input preprocessing we transform audio waves in to mel spectrograms as features to feed it in to the neural network.

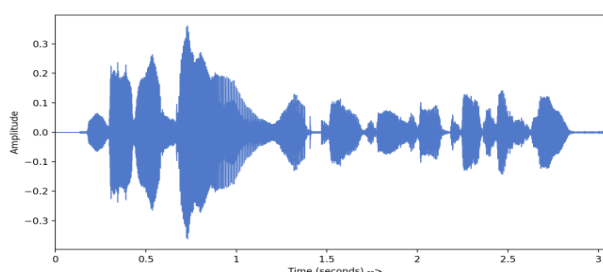


Fig. 3. Vanilla Spectrography

Mel Spectrograms: Mel spectrogram is a spectrogram that is converted to a Mel scale.

Mel means ‘Melody’. A spectrogram is a visualization of the frequency spectrum of a signal, where the frequency spectrum of a signal is the frequency range that is contained by the signal. The Mel scale mimics how the human ear works, with research showing humans don’t perceive frequencies on a linear scale. Humans are better at detecting differences at lower frequencies than at higher frequencies. In short words Mel spectrograms allows machines to understand pitch/melody just like humans. Mel spectrogram is basically a pictorial representation of audio.

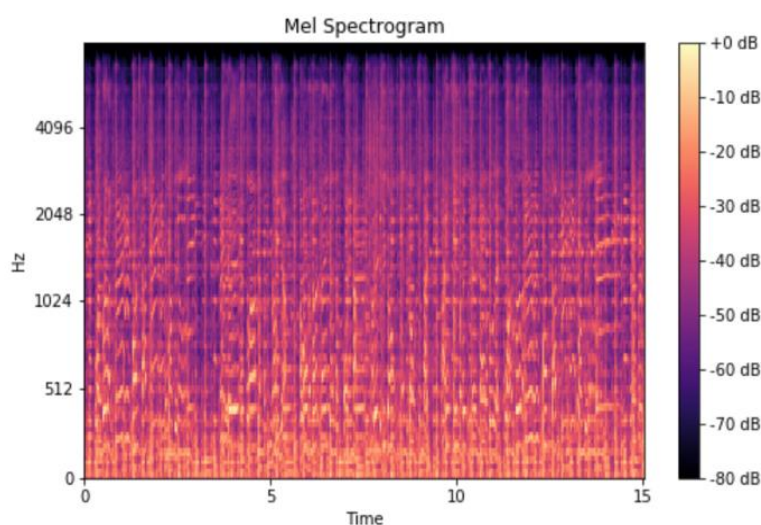
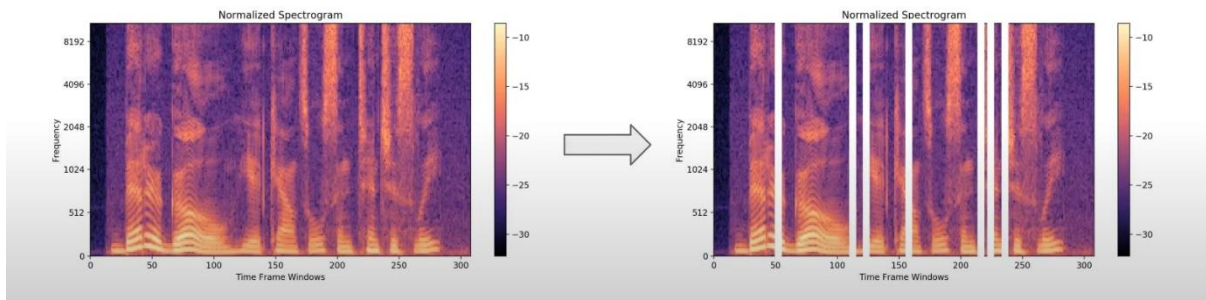


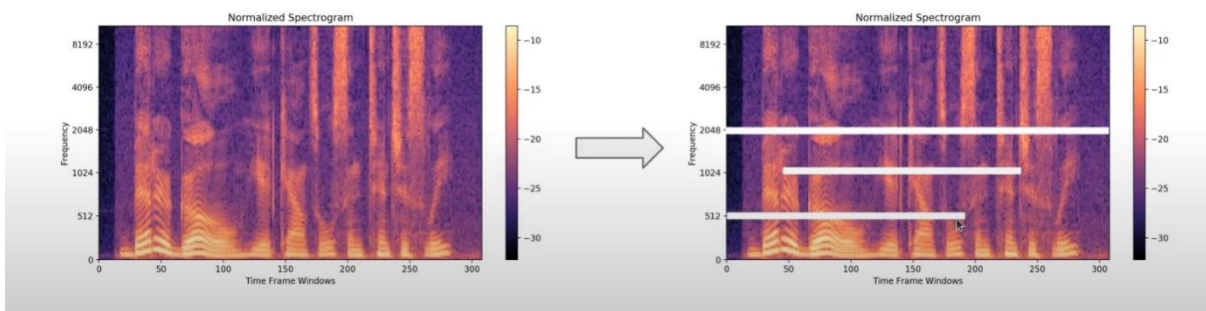
Fig.4. Mel Spectrogram

Making Labels for each characters: We transform our labels in to integer indexes. For example we label a as 0, b as 1, c as 2 and so on.

Spectrogram Augmentation: Spectrogram augmentation is a technique which randomly cuts out or put “0” values in time and frequency domain. This is done to improve the robustness of the version so that it can still convert speech to text if it misses some alphabets.

*Fig.5*

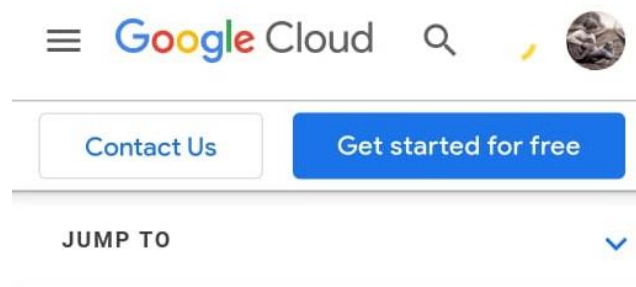
This is How we cut random values from Time domain by doing Time masking. The vertical white lines in X axis represents the null value of time.

*Fig. 6*

And This is How we cut random values from frequency domain by doing Frequency masking. The vertical white lines in Y axis represents the null value of frequency.

3.8 Version used

Here we used the ready-made trained version or API given by Google which is open-source



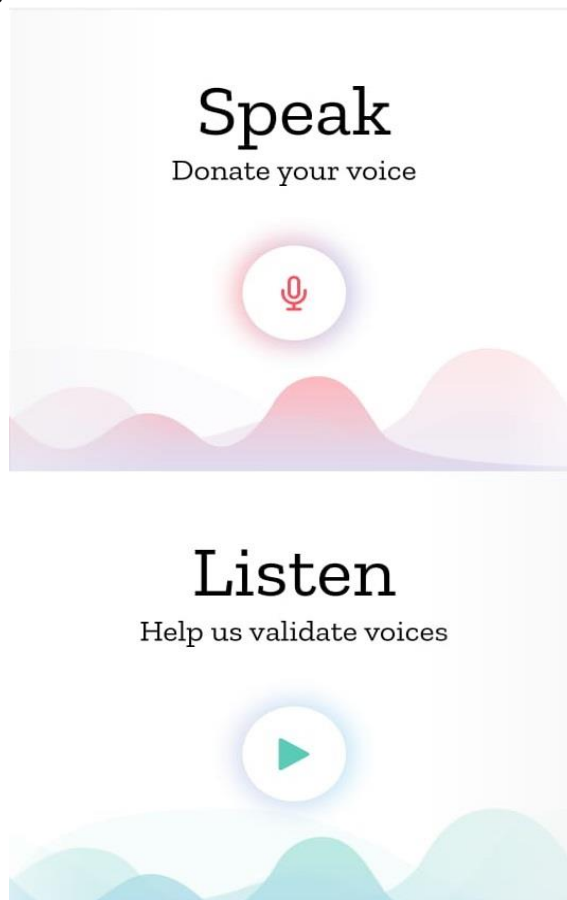
Speech-to-Text

Accurately convert speech into text with an API powered by the best of Google's AI research and technology.

3.9 Input set used

From Common Voice Mozilla we took inputsets of 10000 hours of voice recording to Train our version and work like a human ear.

Through this we are try **Common Voice** moz://a 0 0 ⋮ n voices.

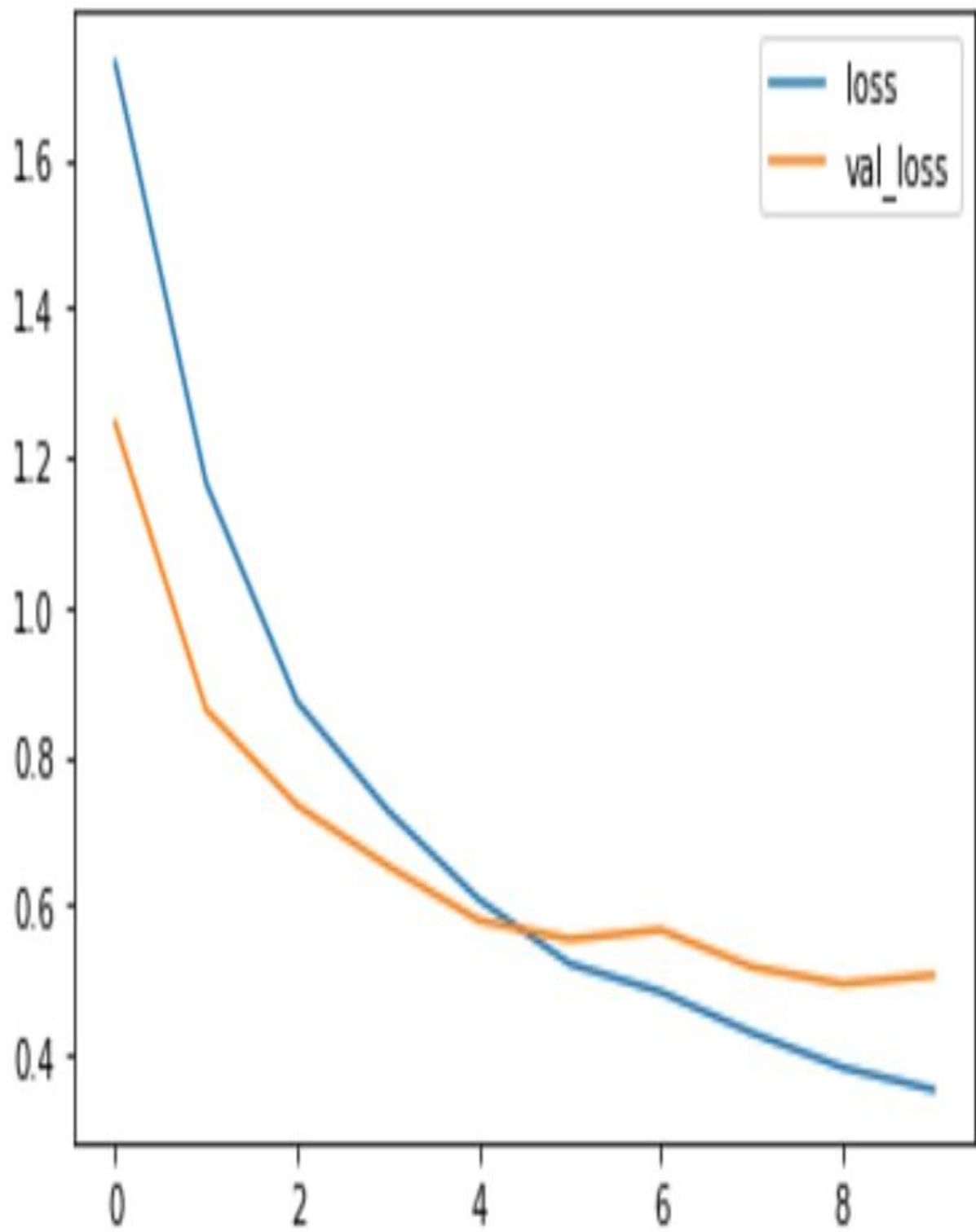


Input shape: (124, 129, 1)

Model: "sequential"

Layer (type)	Output Shape	Param #
resizing (Resizing)	(None, 32, 32, 1)	0
normalization (Normalization)	(None, 32, 32, 1)	3
conv2d (Conv2D)	(None, 30, 30, 32)	320
conv2d_1 (Conv2D)	(None, 28, 28, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
flatten (Flatten)	(None, 12544)	0
dense (Dense)	(None, 128)	1605760
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 8)	1032

Fig. Layers used



```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "name": "audio_features.ipynb",
      "provenance": [],
      "collapsed_sections": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "code",
      "source": [
        "!pip install SpeechRecognition\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "NUVoPZQ5N6J2",
        "outputId": "21aec037-45ed-48f0-d1bd-919014a92943"
      },
      "execution_count": null,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": []

```

```

Collecting SpeechRecognition\n",
  "  Downloading SpeechRecognition-3.8.1-py2.py3-none-any.whl (32.8 MB)\n",
  "\u001b[K |████████████████████████████████████████| 32.8 MB 1.4 MB/s \n",
  "\u001b[?25hInstalling collected packages: SpeechRecognition\n",
  "Successfully installed SpeechRecognition-3.8.1\n"
]
}
],
{
  "cell_type": "code",
  "source": [
    "from os import path\n",
    "from pydub import AudioSegment\n",
    "\n",
    "# files\n",
    "src = \"test.mp3\"\n",
    "dst = \"test.wav\"\n",
    "\n",
    "# convert mp3 to wav\n",
    "sound = AudioSegment.from_mp3(src)\n",
    "sound.export(dst, format=\"wav\")"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "bHJdbGJESWgl",
    "outputId": "9dd320c9-d6ea-47eb-aba9-902c276d08de"
  },
  "execution_count": null,
  "outputs": [

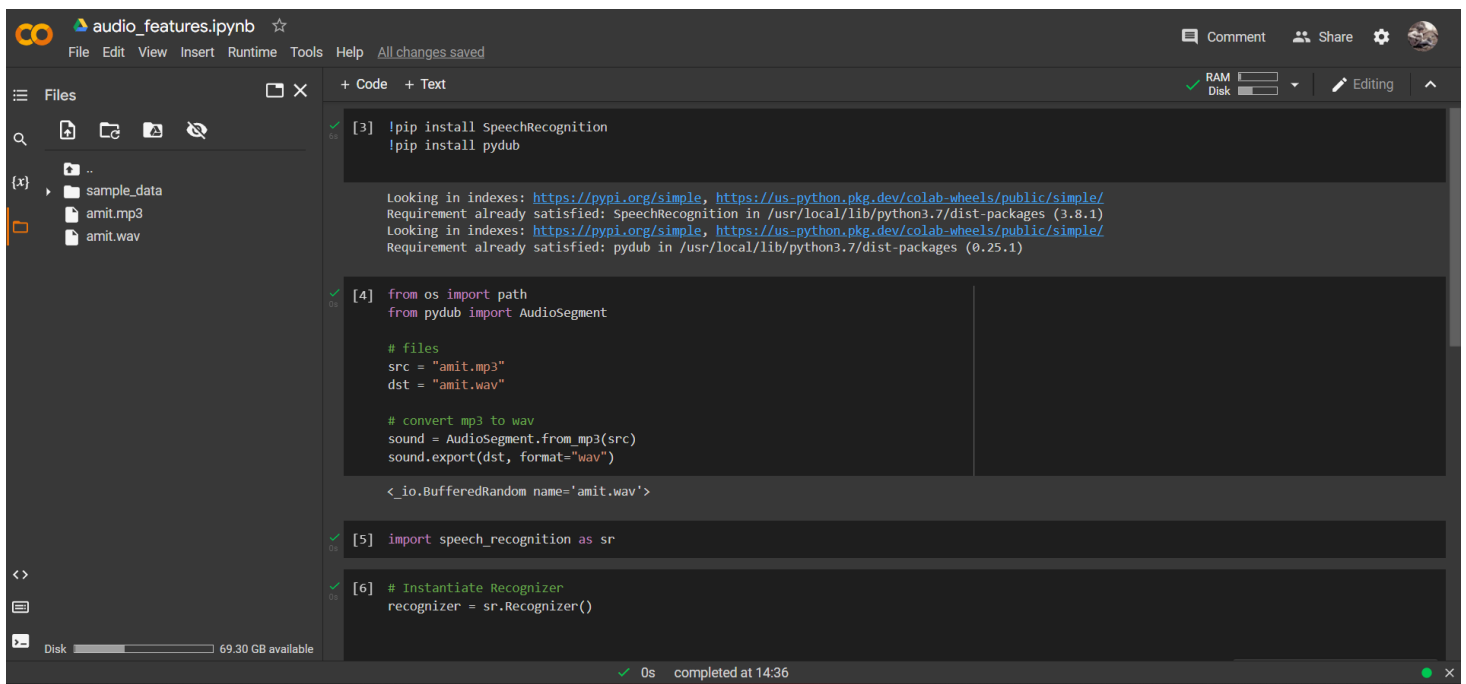
```

```

    "print(text)"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "id": "pIDuKc250sMM",
    "outputId": "57fdc8ee-8c81-4bdf-8804-97ebef8966e1"
  },
  "execution_count": null,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "I am busy right now\n"
      ]
    }
  ],
},
{
  "cell_type": "code",
  "source": [
    ""
  ],
  "metadata": {
    "id": "pbnY_r430sRh"
  },
  "execution_count": null,
  "outputs": []
}
]
}

    "with audio as source:\n",
    "    audio = recognizer.record(source)"
  ],
  "metadata": {
    "id": "gnPB3IpD0Ii9"
  },
  "execution_count": null,
  "outputs": []
},
{
  "cell_type": "code",
  "source": [
    "# Transcribe AudioData to text\n",
    "text = recognizer.recognize_google(audio,\n",
    "                                   language=\"en-US\")\n",

```



```
[3] !pip install SpeechRecognition
!pip install pydub

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: SpeechRecognition in /usr/local/lib/python3.7/dist-packages (3.8.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pydub in /usr/local/lib/python3.7/dist-packages (0.25.1)

[4] from os import path
from pydub import AudioSegment

# files
src = "amit.mp3"
dst = "amit.wav"

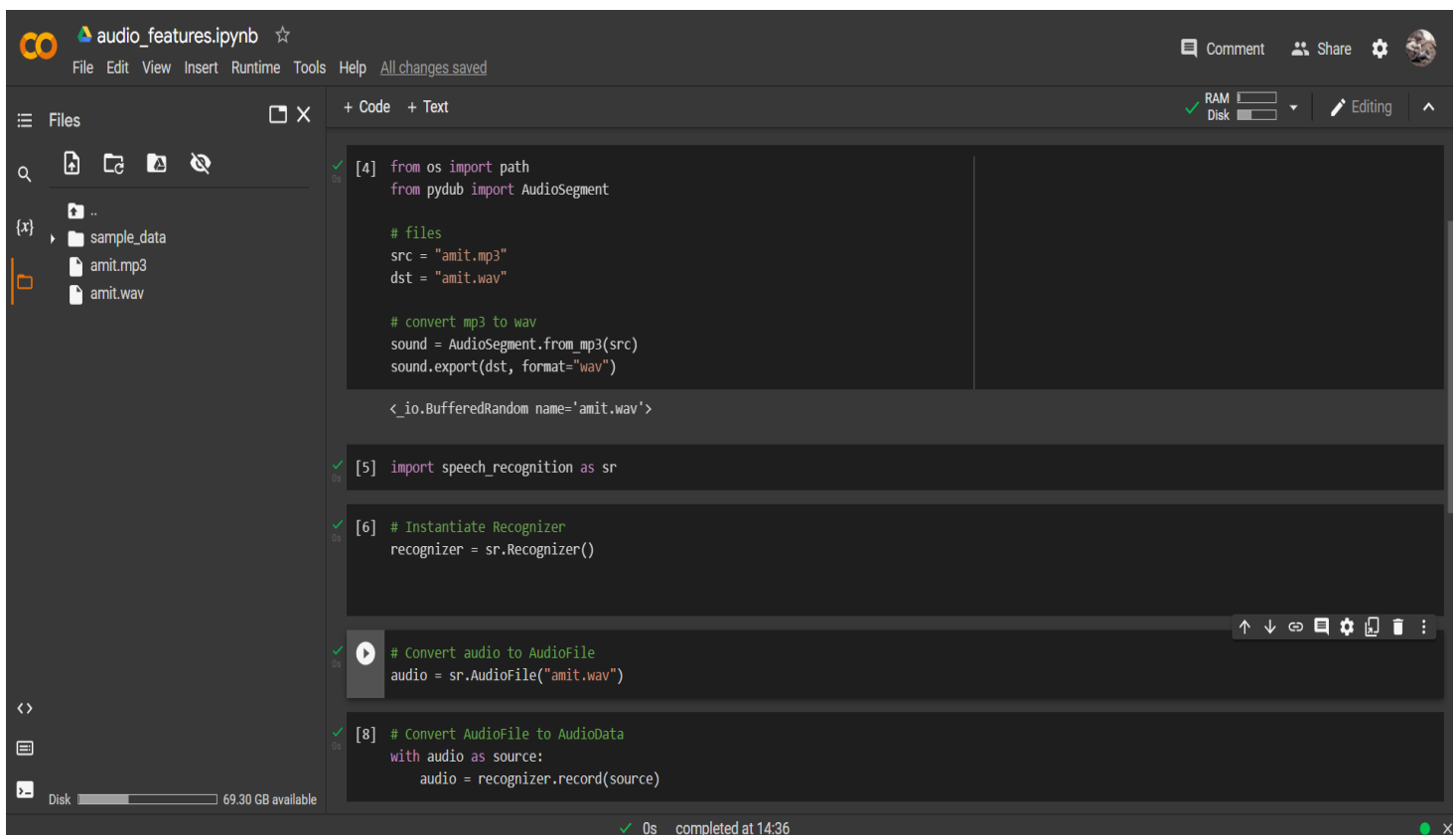
# convert mp3 to wav
sound = AudioSegment.from_mp3(src)
sound.export(dst, format="wav")

<_io.BufferedRandom name='amit.wav'>

[5] import speech_recognition as sr

[6] # Instantiate Recognizer
recognizer = sr.Recognizer()
```

Here, we uploaded a recorded audio file



```
[4] from os import path
from pydub import AudioSegment

# files
src = "amit.mp3"
dst = "amit.wav"

# convert mp3 to wav
sound = AudioSegment.from_mp3(src)
sound.export(dst, format="wav")

<_io.BufferedRandom name='amit.wav'>

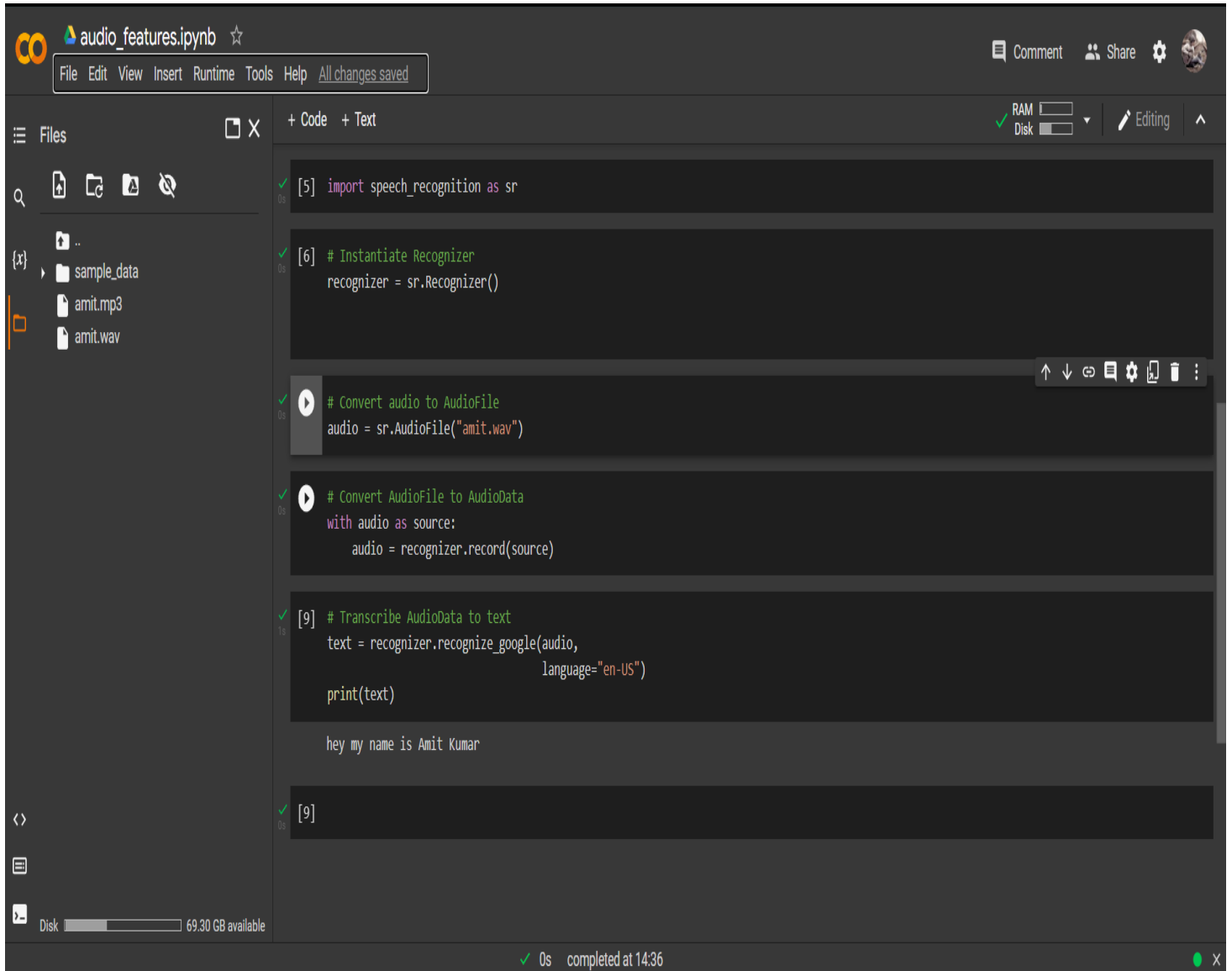
[5] import speech_recognition as sr

[6] # Instantiate Recognizer
recognizer = sr.Recognizer()

[7] # Convert audio to AudioFile
audio = sr.AudioFile("amit.wav")

[8] # Convert AudioFile to AudioData
with audio as source:
    audio = recognizer.record(source)
```

Here we compile the input



The screenshot shows a Jupyter Notebook titled "audio_features.ipynb". The left sidebar displays a file explorer with a folder named "sample_data" containing two files: "amit.mp3" and "amit.wav". The main area contains five code cells, each with a green checkmark and a "0s" execution time. The code in the cells is as follows:

```
[5] import speech_recognition as sr
```

```
[6] # Instantiate Recognizer
recognizer = sr.Recognizer()
```

```
# Convert audio to AudioFile
audio = sr.AudioFile("amit.wav")
```

```
# Convert AudioFile to AudioData
with audio as source:
    audio = recognizer.record(source)
```

```
[9] # Transcribe AudioData to text
text = recognizer.recognize_google(audio,
                                   language="en-US")
print(text)
```

The output of the fifth cell is displayed below the code:

```
hey my name is Amit Kumar
```

The bottom status bar indicates the notebook is "completed at 14:36" with a green dot and a close button (X).

Here we get the output “Hey my name is Amit Kumar” .

FUTURE DEVELOPMENTS

- Adding new languages and scriptures
- Increasing the accuracy
- Utilize the version in different applications

CHAPTER 6: CONCLUSION

The main objective of this project is to design a program applicable in audio extraction with the help of understanding in deep learning. We have explored the basic ideas of Transfer learning and their applications and implementations in speech-to-text program. In the next phase of the project is to design the working version and extract results from the examples all through by applying the theoretical literature surveys and methodology we have understood so far .

REFERENCES :

1. <https://realpython.com/python-speech-recognition/>
2. https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_speech_recognition.htm
3. <https://towardsdatascience.com/speech-recognition-in-real-time-using-python-fbbd62e6ff9d>
4. <https://dphi.tech/blog/audio-input-analysis-using-deep-learning-with-python-part-1/>
5. <https://www.ibm.com/cloud/learn/convolutional-neural-networks>
6. <https://www.google.com/>
7. <https://www.geeksforgeeks.org/>
8. <https://www.youtube.com/>
9. https://www.google.com/search?q=google+speech+recognition+api&client=ms-android-oppo-rev1&sxsrf=ALiCzsYq67_v1hoGfS4MHakIVCEmVQJFcA%3A1655296420987&ei=pNGpYoznO52z4-EPpu2wiA4&oq=google+speech+recognition+api&gs_lcp=ChNtb2JpbGUtZ3dzLXdpei1zZXJwEAMyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDIFCAAQgAQyBQgAEIAEMgUIABCABDoHCCMQsAMQJzoHCAAQRxCwAzoHCAAQsAMQZoSCC4QxwEQ0QM QyAMQsAMQXgBOhUILhDHARDRAxDUAhDIAxCwAxBDGAE6BwgjEOoCECc6BAgjECc6EAguELEDEIMBEMcBENEDEEM6CwgAEIAEELEDEIMBOgQIABBD0hYILhCABBCHAhCxAXCDARDHARDRAxAUOhAIAABCABBCHAhCxAXCDARAUAUOgoIABCxAXCDARBDOg0IABCAB BChAhCxAXAUOqglABCABBChAzoLCAAQsQMqgEQkQI6CAgAELEDEIMBOgcIABCABB AKOgoIABCABBCHAhAUSgUIOBIBMUoECEEYAFcxCvI6OGCcSWgEcAF4BIAB2AKIAeUz kgEGMi0yMi4zmAEAoAEBsAEPyAEQwAEB2gEECAEYCA&sclient=mobile-gws-wiz-serp
10. <https://commonvoice.mozilla.org/en>