Machine Learning 03 - Entering the Uncanny Valley of Speech

SLQ Wiki Fabrication Lab 2024/05/03 15:22

~~REVEAL~~

Machine Learning 03 - Entering the Uncanny Valley of Speech

With State Library closed to the public due to COVID-19, for this Machine Learning (ML) workshop we won't be able to use the Digital Media Lab at The Edge.

This online workshop recaps our previous workshops, and explores the world of Text To Speech (TTS), voice synthesisers and Speech To Text (STT) voice recognition nbuilt with ML. The workshop is not an introduction to coding or math, but we will give a of general overview of how ML is defined and where it is commonly used today.

We've chosen an approach that demonstrates the power and limitations of ML and leaves you with an understanding of how use an online ML environment, along with ideas on how to use State Library resources to explore ML further.

The first half of the workshop will cover

- a basic explanation of ML
- recap of previous ML workshops
- ML for speech

The second half of the workshop explore how to impliment ML research using Google's Colab platform

Outcomes

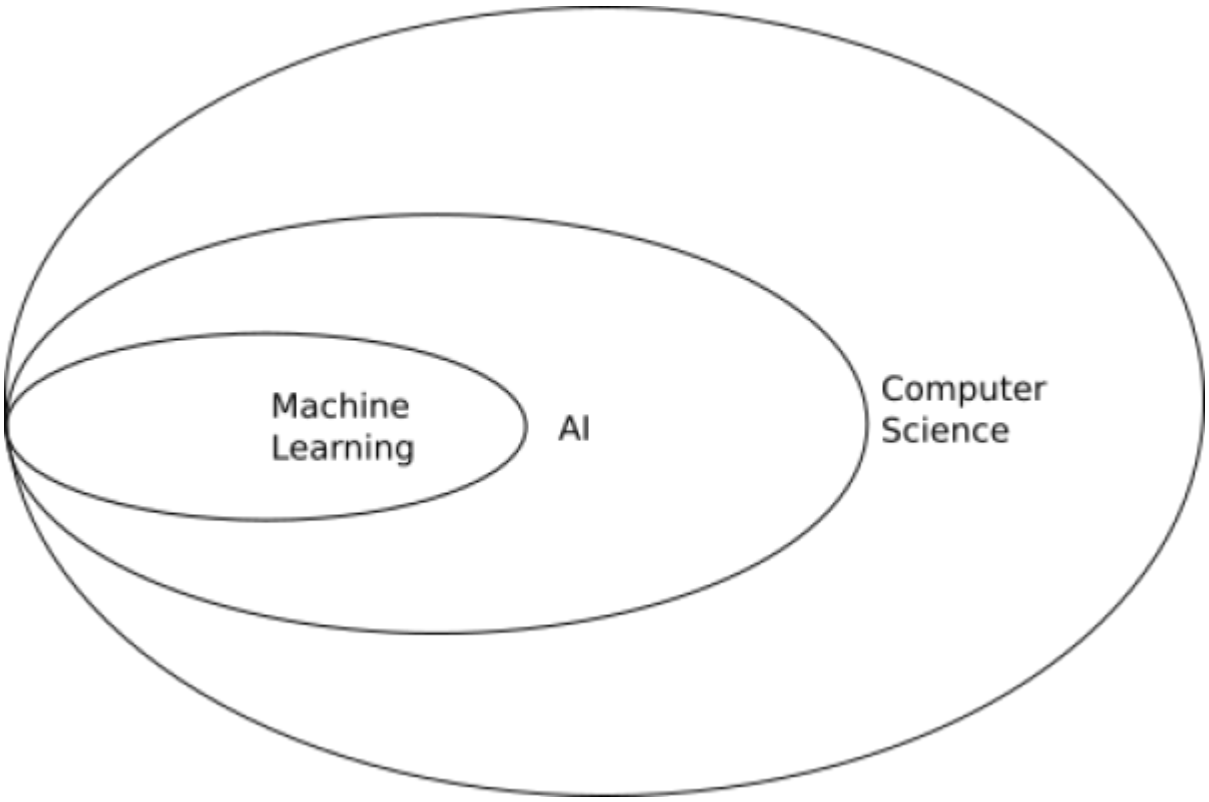
- A general basic ML background
    - ML for speech
- using Google Colab
    - Spleeter (audio source separation)
    - TTS (Mozilla TTS)
    - STT (Mozilla Deepspeech)

Requirements

All we need to get started for this workshop is a Google account to access Google Colab in the second half of the workshop. If you don't have one you can quickly sign up. If you don't want to create a Google account, you can always just follow along with the examples.

Background

Machine Learning(ML) is a subset of Artificial Intelligence (AI) which is is a fast moving field of computer science (CS). A good way to think about how these fields overlap is with a diagram.



Machine Learning - Why Now?

While many of the concepts are decades old, and the mathematical underpinnings have been around for centuries, the explosion in use and development of ML learning has been enabled by the creation and commercialisation of massively parallel processors. This specialised computer hardware most commonly found in Graphics Processing Units (GPUs) inside desktop and laptop computers and takes care of the display of 2D and 3D graphics. The same processing architecture that accelerates the rendering of 3D models onscreen is ideally suited to solve ML problems, resulting in specialised programming platforms, Application Programming Interfaces (APIs) and programming libraries for AI and ML.

Common Machine Learning Uses

One way to think of ML is as a *recommendation system*.

Based on input data (a lot of input data[1)])

- a machine learning system is trained

- a model is generated
- the model is used can make recommendations (is implimented) on *new* data.

:workshops:prototypes:machine_learning:ideepcolor:8725096366_d1fe677cc5_o.jpg

One extremely common application of this is image recognition.

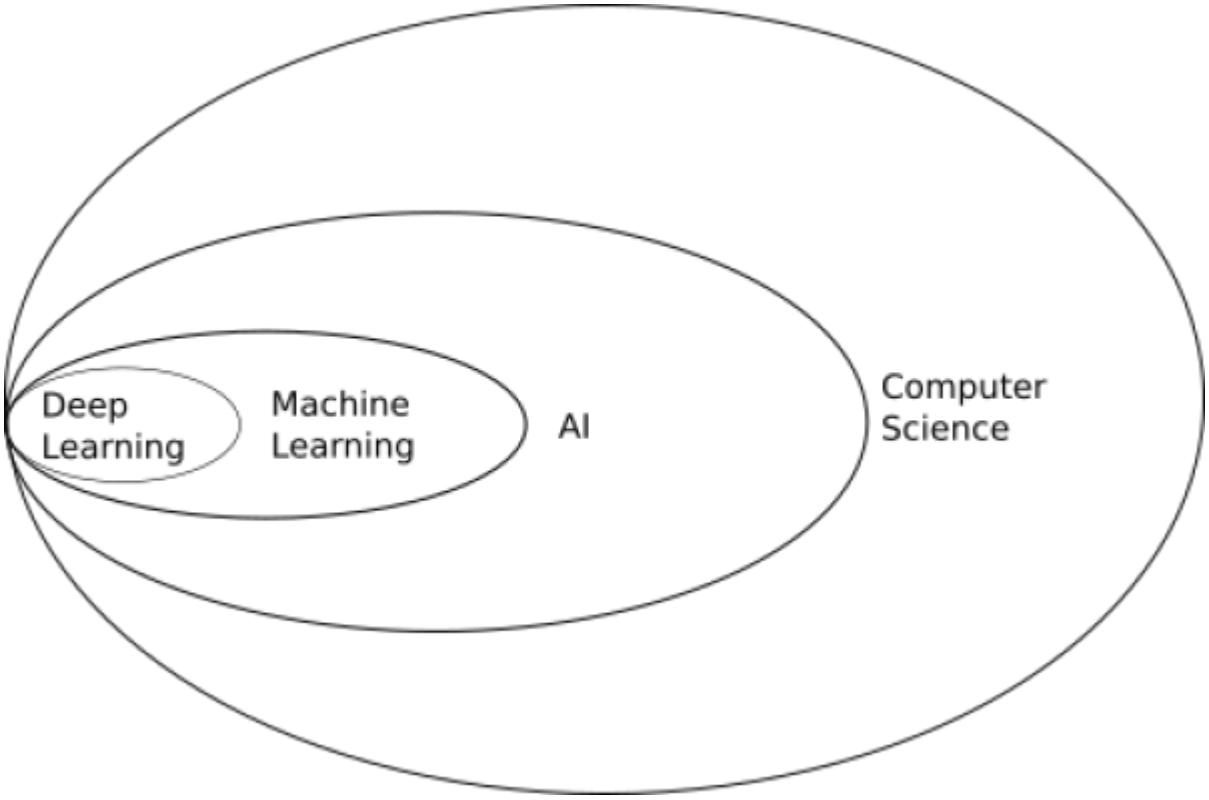
:workshops:prototypes:machine_learning:ideepcolor:2020-02-21_14_19_57-8725096366_d1fe677cc5_o_fb.jpg

When facebook asks you to tag a photo with names, you are providing them with a nicely annotated data set for **supervised learning**. They can then use this data set to train a model than then recognises (makes a recommendation) about other photos with you or your friend in it.

Snapchat filters use image recognition to make a map of your features, then applies masks and transformations in real-time.

Deep Learning

Today we are going to go a little “deeper” inside ML, exploring deep learning. Deep learning used multiple layers of algorithms, in an artificial neural network, inspired by the way the human neural networks inside all of us.



Interactive Deep Colorization

Lets take a look at the subject of our first ML workshop Real-Time User-Guided Image Colorization with Learned Deep Priors (ideepcolor), by Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu and Alexei A. Efros.

Here is a talk about the details of their paper.

Video

I encourage you to watch the above talk in full....

but the TLDR version is that they have:

- trained a neural network on millions of images
- combined this with simulated human interaction
- produced a model that recommends an initial colourisation
- that takes user input to refine the colourisation.

The user input is provided through a Graphical User Interface (GUI), and the end result can be exported, along with information about how the model made its recommendations.

You can check out a video of the demo in action here.

Video

ML - From Paper to Product

We'll be exploring a few ML ideas, but to start with lets follow "Real-Time User-Guided Image Colorization with Learned Deep Priors", by Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S. Lin, Tianhe Yu, Alexei A. Efros from paper to product. Its not a recent paper in ML terms where it seems every month brings another breakthrough, but we can follow this paper right through to its release in Adobe Photoshop Elements 2020.

Research Papers on arXiv.org

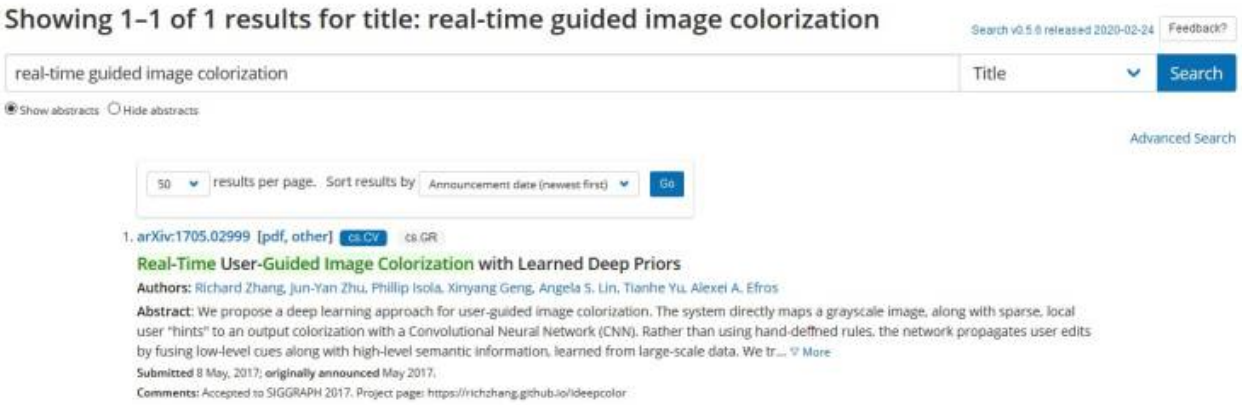
arXiv.org is probably the worlds biggest and fastest growing collection of preprint electronic scientific papers from mathematics, physics, astronomy, electrical engineering, computer science, quantitative biology, statistics, mathematical finance and economics. All of the ML ideas we will be looking are either first published on arXiv.org, or reference papers on the site.

Finding a Paper

Papers on arXiv.org are moderated but not peer-reviewed, which means the speed and volume of publishing on this open-access repository is overwhelming. But to get started, lets say we are interested in re-colourisation of black and white images of our grandparents at their wedding, but we

don't want to do all the work ourselves. We'd also like to interact in real-time and guide the process, so we can get the colour of grandad's suit and grandma's bouquet just right.

So lets search for “real-time guided image colourisation” but we'll use the American English spelling “ colorization”. Searching for “real-time guided image colorization” brings up our paper straight away, with handy PDF link[2].



Examining the Abstract

All research papers begin with an abstract, and a well written abstract will tell us all we need to know about whether the paper is relevant for you, particularly if we are looking for working demonstration. This time we are in luck - there is a link to a an ideepcolor demo site at the end of the abstract.

Check out the Demo

The demo for ideepcolor looks great and we've got a link at the top of the page, where ideepcolor is implemented on github.

Code Implementation on github.com

Github.com is a website used by software developers to create, collaborate and share source code, and is most likely the largest repository of source code in the world. Github is named after git, a free and open-source(FOSS) distributed version-control system for tracking changes in source code during software development. Git means that developers from all over the world can work on the same code and if the project is open source, build on, expand and re-purpose shared codes[3]. But lets back up a bit and cover off on what source code is.

Using the Source

source code is the instructions for a computer program contained in a simple text document.

For a computer to run a program, the source code either has to be

***compiled** into binary machine code by a compiler:

- his file is executable - in this case execute just means can be read, understood and acted on by the computer, or
- **interpreted** by another program, which directly executes the code

Here is a example of source code. in this case its a simple program in the C programming language that shows on the screen “Hello, World”

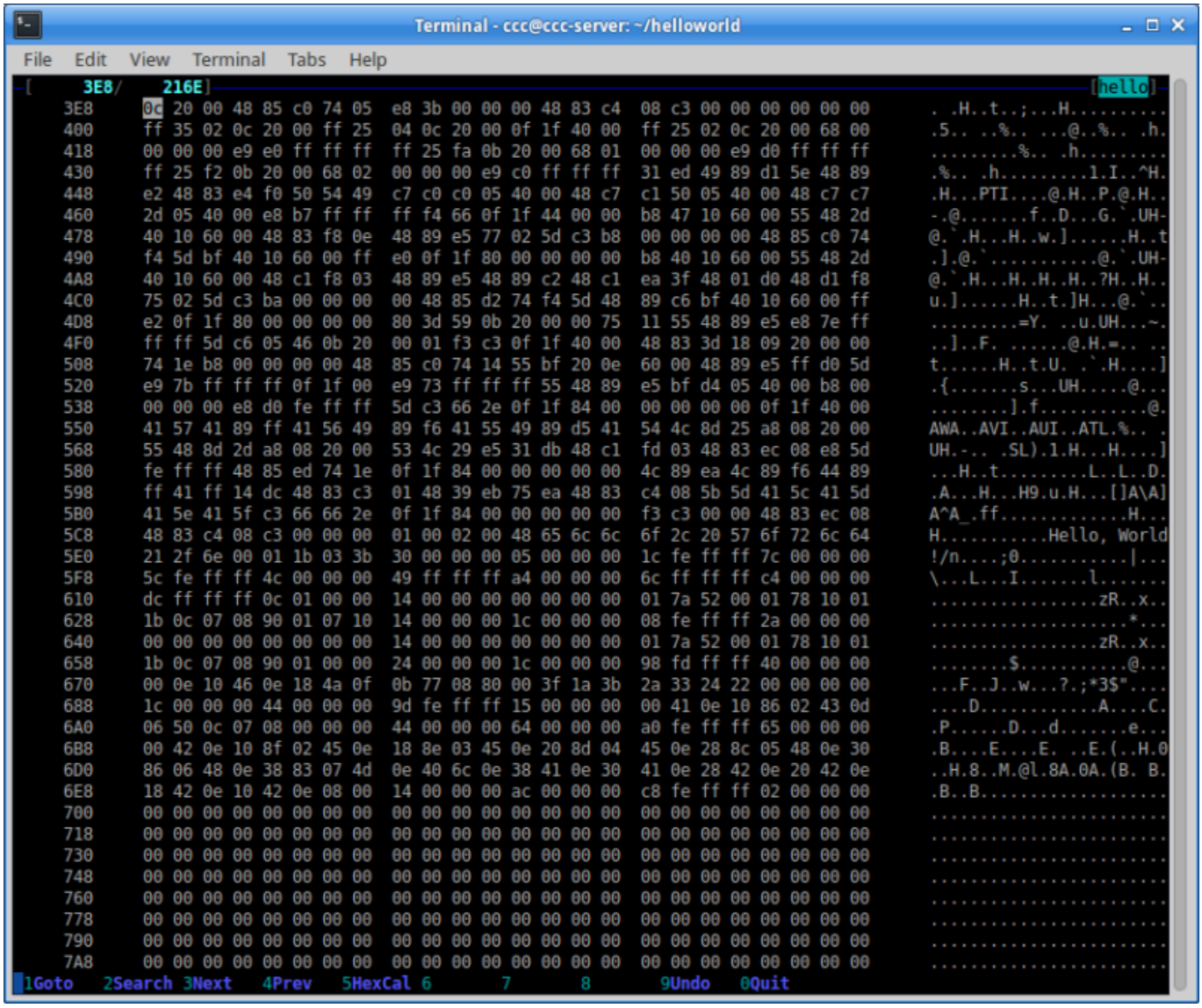
```
 #include <stdio.h>

 int main(void)
{
 printf("Hello, world!\n");
 return 0;
}
```

Despite the strange symbols, if you know how the C language is written, this program is **human readable**.

Once this code is run through a compiler, we get a binary executable file - which is **machine readable**.

But with the right tools (like a HEX editor) we can still open the file and edit it.

Here is the binary for our “Hello World!” program.



Open Source

Dokuwiki (the software we are using for this wiki) is open source, and developed publicly, and freely available on the internet. Anyone is able to grab the source code and run it, modify it or redistribute it.

Below is and example of the **open source** code for this wiki, which is written in a language called php.

```
  // define all DokuWiki globals here (needed within test requests but also helps to keep track)
  global $ACT, $INPUT, $QUERY, $ID, $REV, $DATE_AT, $IDX,
  $DATE, $RANGE, $HIGH, $TEXT, $PRE, $SUF, $SUM, $INFO, $JSINFO;
  if(isset($_SERVER['HTTP_X_DOKUWIKI_DO'])) {
  $ACT = trim(strtolower($_SERVER['HTTP_X_DOKUWIKI_DO']));
  } elseif(!empty($_REQUEST['idx'])) {
```

```
$ACT = 'index';
} elseif(isset($_REQUEST['do'])) {
$ACT = $_REQUEST['do'];
} else {
$ACT = 'show';
}
```

How did we get hold of the source code for this wiki? In this case all we did was look in the dokuwiki source found on github pick bit of code at random and throw it in our wiki.

So, finding the source for open software is easy. but to do the same thing with closed source program is usually difficult or impossible. Either you purchase or are given access to the code. Any other method may break all manner of licenses and laws.

ideepcolor on Github

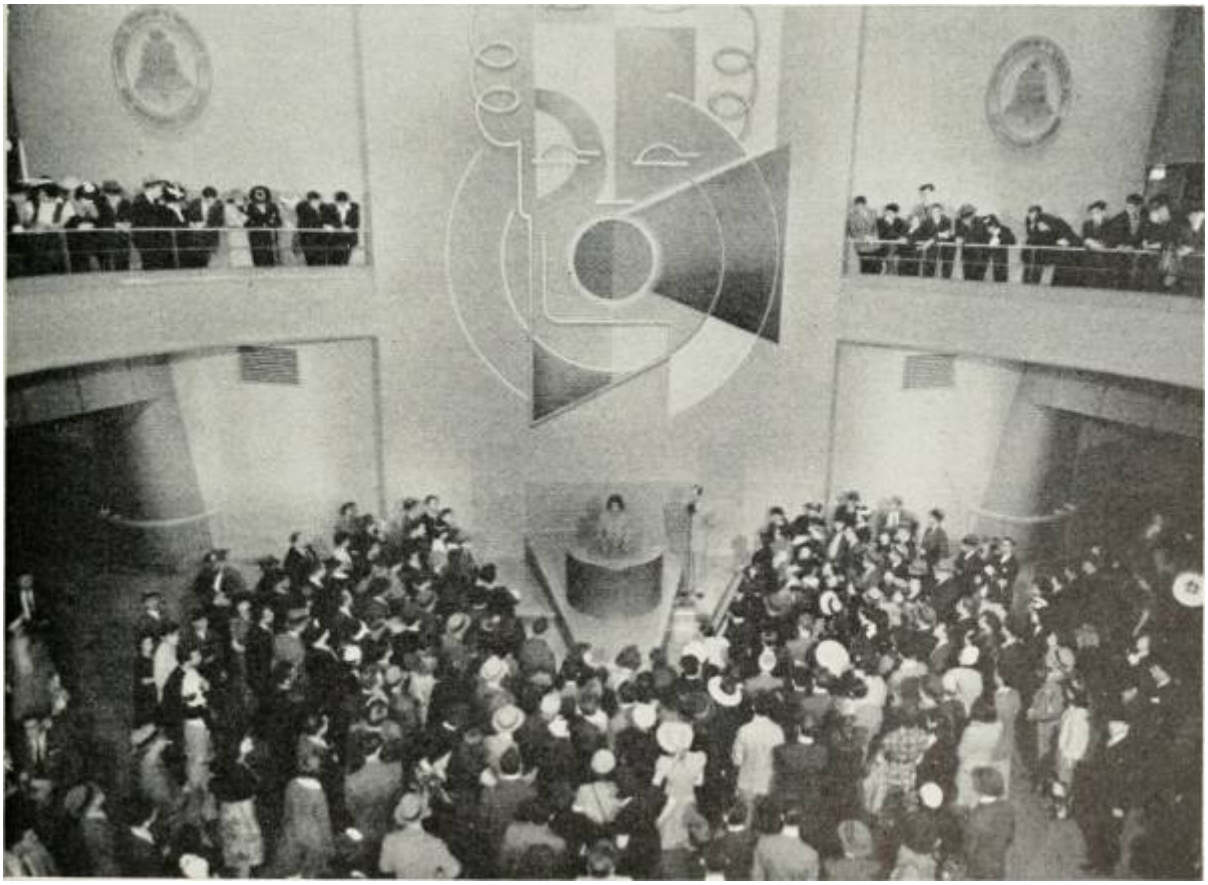
For a project like ideepcolor, Github is where researchers and developers describe how they achieved their results with real, working code. There is generally an introduction, which should contain any major updates to the project, then we go through the prerequisites, setting up or getting started, installation, training and (hopefully) application. There are number of ways to demonstrate application of a model, ideepcolor has built a custom Graphical User Interface (GUI), which we demonstrated this project in our first ML workshop. More commonly demos are done with a jupyter notebook or Google Colab notebook. Now, lets look at the updates at the top of the ideepcolor repository - which tell us:

> 10/3/2019 Update: Our technology is also now available in Adobe Photoshop Elements 2020. See this blog and video for more details.

So it looks like this project has moved to the next stage - integrating ideepcolor into a commercial product.

Speech Synthesis

Like many of the 20th century's technological inovations, the frst modern speech synthesiser can be traced back to the invention of the vocoder at Bell Labs. Derived from this, the Voder was demonstrated at the 1939 World Fair.



4)

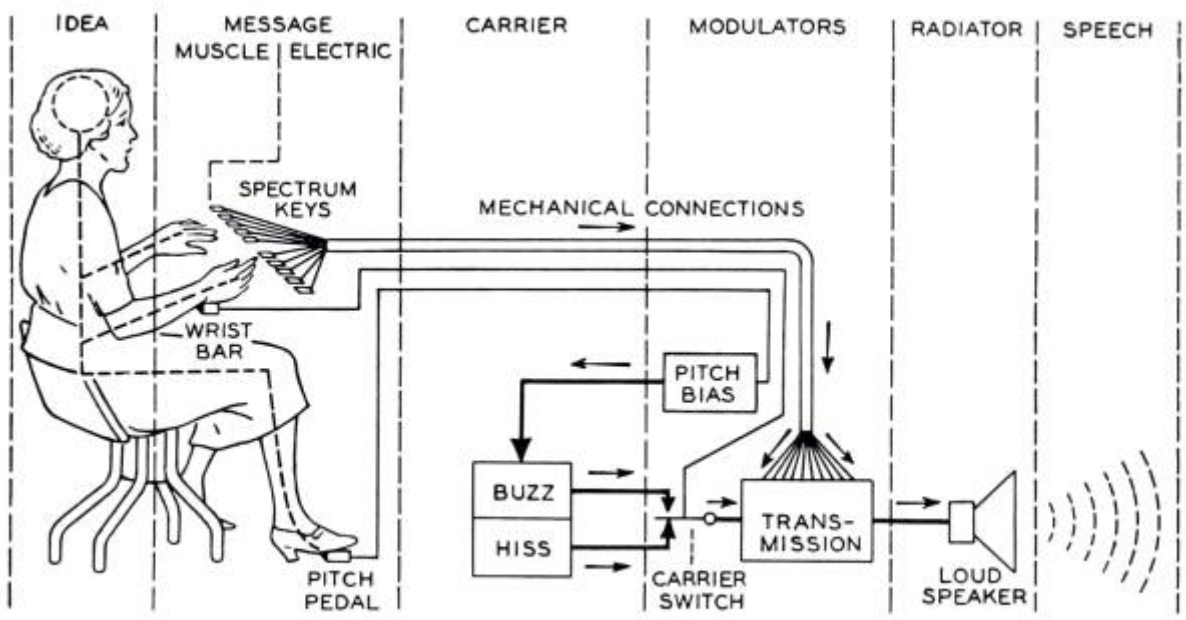


Fig. 8—Schematic circuit of the voder.

Historical Audio Examples

Here is a playlist of various historical TTS methods.

https://soundcloud.com/user-552764043

Modern State of the Art TTS

Now - it time to have some fun with TTS - check out the man holding the frog below...

https://vo.codes/#speak

And have a listen to some interesting examples from pop/meme culture.

https://fifteen.ai/examples

https://www.youtube.com/watch?v=drirw-XvzzQ

Wavenet

Modern deep learning based synthesis started with the release of Wavenet in 2016 by Google's Deepmind.

WaveNet changes this paradigm by directly modelling the raw waveform of the audio signal, one sample at a time. As well as yielding more natural-sounding speech, using raw waveforms means that WaveNet can model any kind of audio, including music.[5)]

Tacotron and Tacotron2

Wavenet was followed by Tacoctron (also from Google) in 2017.

https://google.github.io/tacotron/publications/tacotron/index.html

Then Tacotron2

https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html

Google Colab

Google's Colaboratory[6)], or “Colab” for short, allows you to write and execute Python in your browser, with

- Zero configuration required

- Free access to GPUs
- Easy sharing

Python

Python is an open source programming language that was made to be easy-to-read and powerful[7]). Python is:

- a high-level language, (Meaning programmer can focus on what to do instead of how to do it.)
- an interpreted language (Interpreted languages do not need to be compiled to run.)
- is often described as a “batteries included” language due to its comprehensive standard library.

A program called an interpreter runs Python code on almost any kind of computer. In our case python will be interpreted by google colab, which is based on Jupyter notebooks.

Jupyter Notebooks

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text[8]. Usually Jupyter notebooks require set-up for a specific purpose, but Colab takes care of all this for us.

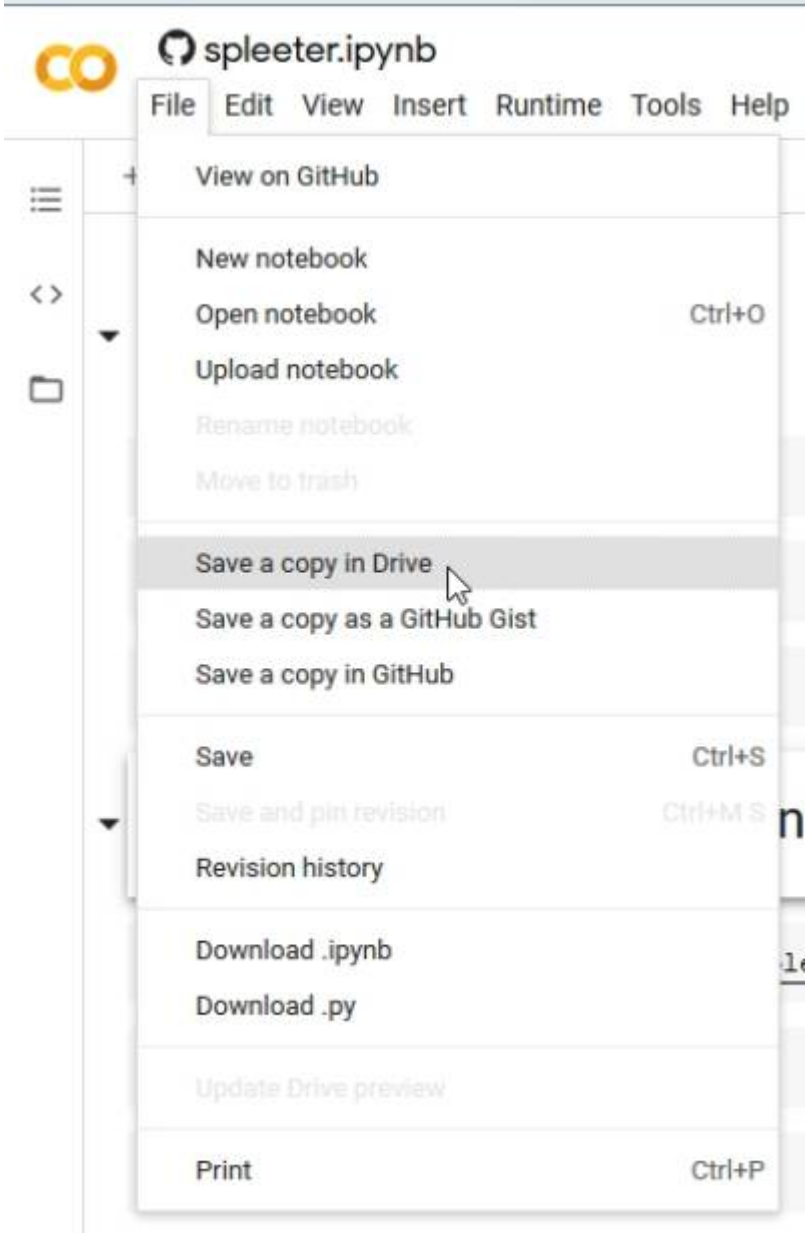
Getting Started with Colab

The only requirment for using Colab is (unsurprisingly) a Google account. Once you have a google account, lets jump into our first ML example - Spleeter - that we mentioned earlier. Go to the Colab here:

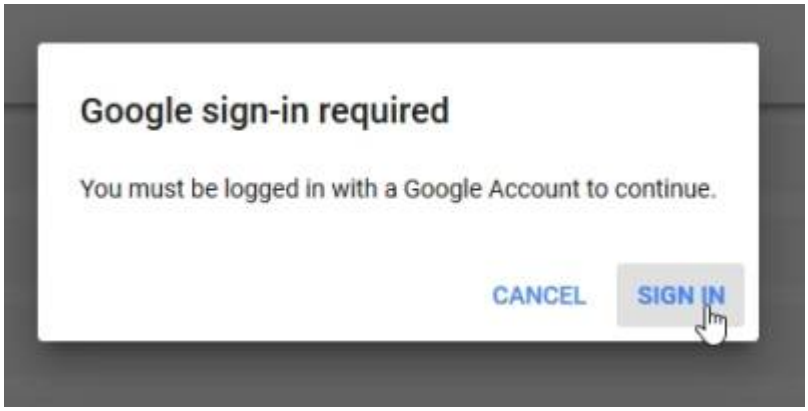
https://colab.research.google.com/github/deezer/spleeter/blob/master/spleeter.ipynb

Making a Colab Copy

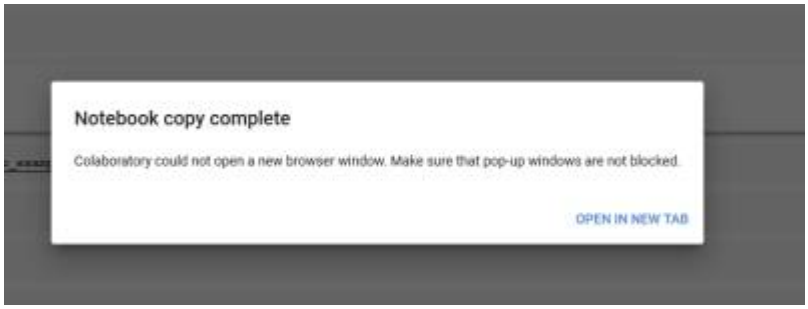
The first step is make a copy of the notebook to our Google drive - this means we can save any changes we like.



This will trigger a google sign-in

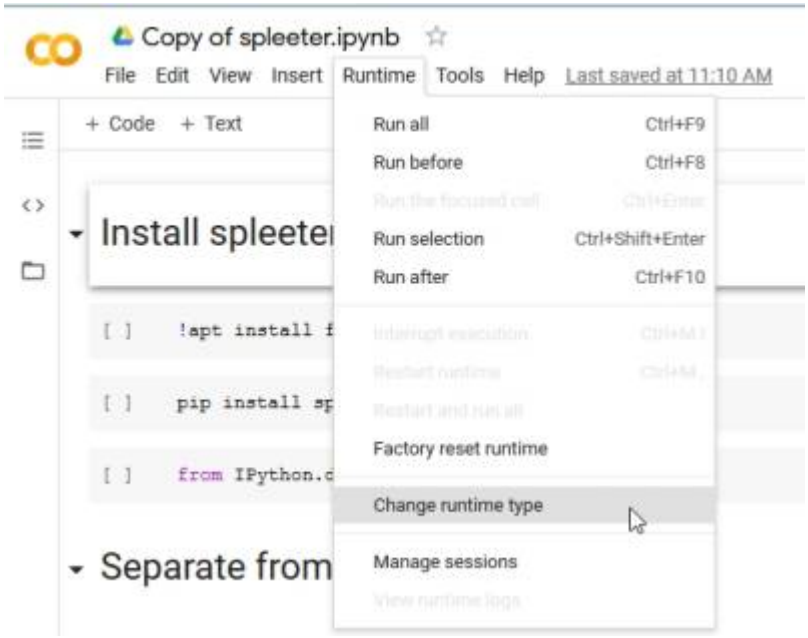


and the your copy will open in a new tab.

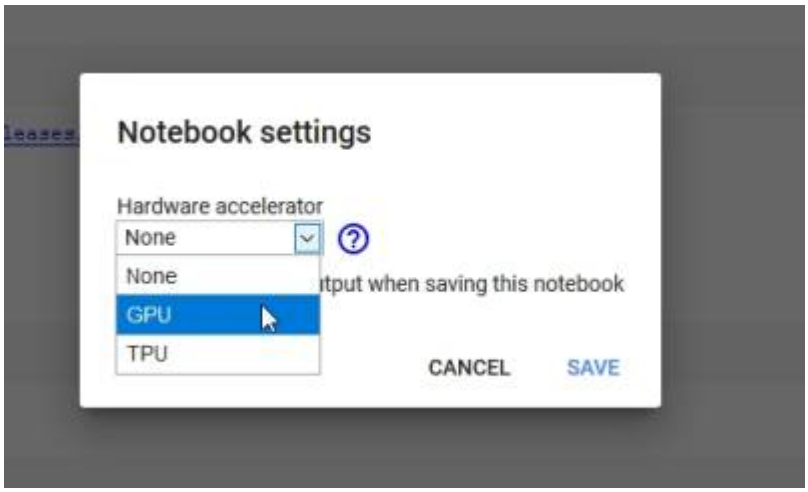


Select a Runtime

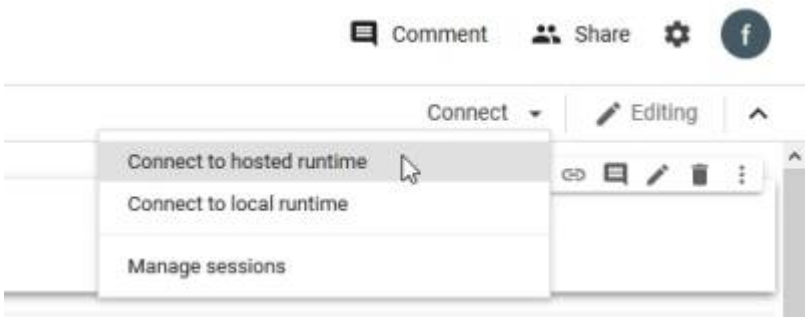
Next we change our runtime (the kind or processor we use)



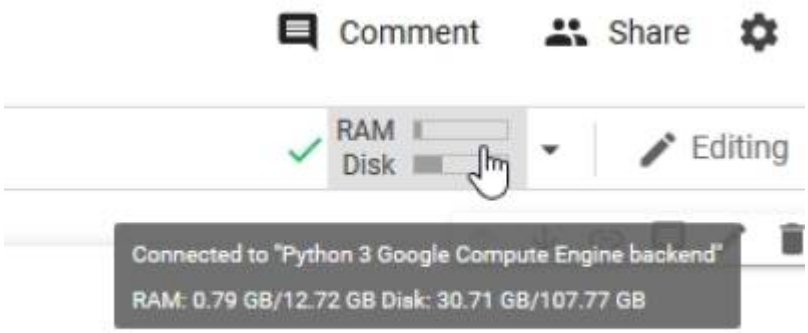
to a GPU to take advantage of Googles free GPU offer.



Now lets connect to our hosted runtime



and check the specs...



Step Through the Notebook

Now its time to actually use the notebook! Before we start, lets go over how the notebooks work:

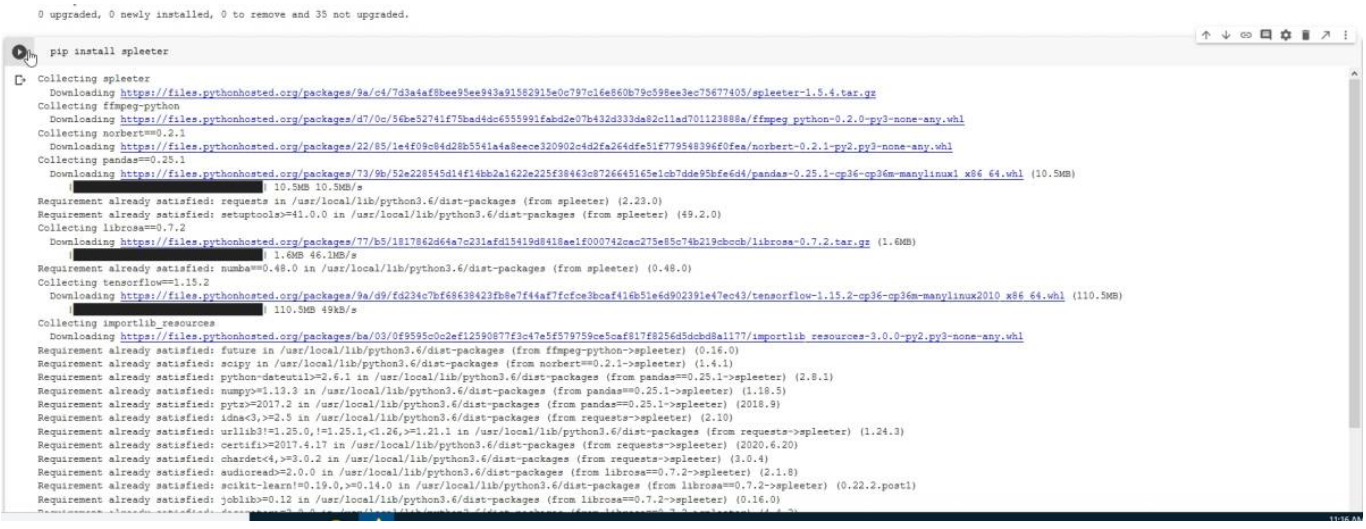
- The notebook is divided into sections, with each section made up of cells.
- These cells have code pre-entered into them,
- A play button on the runs (executes) the code in the cell.
- The output of the cell is printed (or displayed) directly below each cell.
- The output could be text, pictures, audio or video.

Cells usually contain python code, but can also be coded in bash - the UNIX command line shell. Cells containing bash commands start with an exclamation mark `!`

Our first section is called “Install Spleeter” and contains the bash command `apt install ffmeg` . This installs ffmeg in our runtime, which is used to process audio. Press the go button..

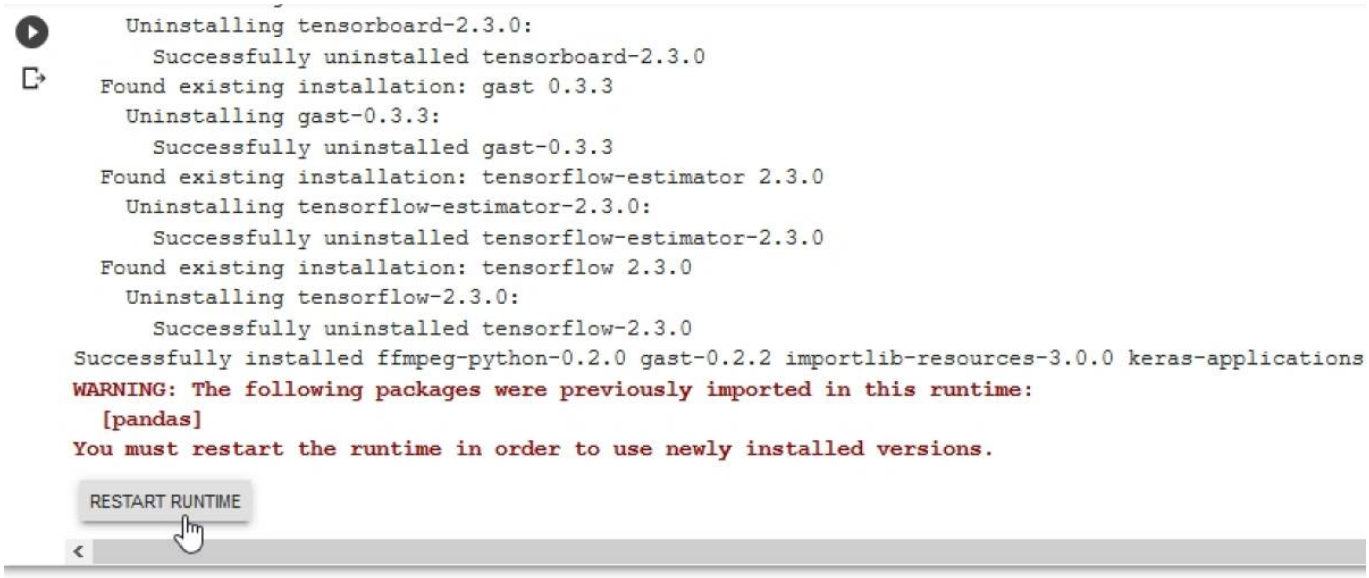
ffmpeg will be downloaded and installed to our runtime.

Next we will run a python command `pip`to use the python package manager to install the spleeter python package.

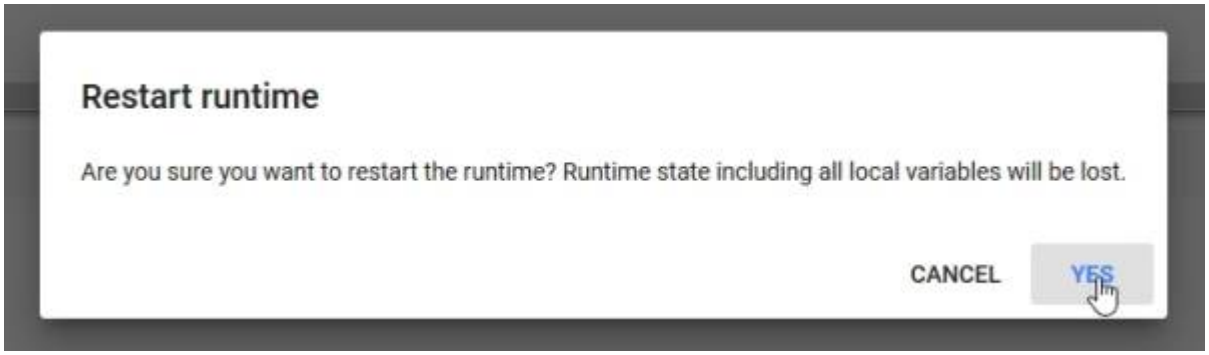


This will take a while - and at the end we will get a message saying we need to restart our runtime

due to some compatibilty issues [9)]



Go ahead and restart



Next is another bash command

```
wget
```

we use to (web)get our example audio file.

And the next cell uses the python `Audio` command to give us a nice little audio player so we can hear our example.

Now its finally time to use the spleeter tool with the `separate` command [10] as `!spleeter separate` , and lets pass the `-h` flag [11] to show us the built in help for the command.

```
!spleeter separate -h
```

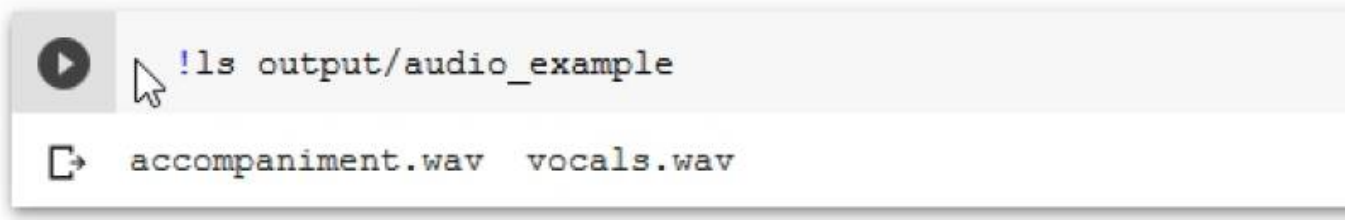
```
usage: spleeter separate [-h] [-a AUDIO_ADAPTER] [-p CONFIGURATION]
                         [--verbose] -i INPUTS [INPUTS ...] [-o OUTPUT_PATH]
                         [-f FILENAME_FORMAT] [-d DURATION] [-s OFFSET]
                         [-c {wav,mp3,ogg,m4a,wma,flac}] [-b BITRATE] [-m]
                         [-B {tensorflow,librosa,auto}]

optional arguments:
  -h, --help            show this help message and exit
  -a AUDIO_ADAPTER, --adapter AUDIO_ADAPTER
                        Name of the audio adapter to use for audio I/O
  -p CONFIGURATION, --params_filename CONFIGURATION
                        JSON filename that contains params
  --verbose             Shows verbose logs
  -i INPUTS [INPUTS ...], --inputs INPUTS [INPUTS ...]
                        List of input audio filenames
  -o OUTPUT_PATH, --output_path OUTPUT_PATH
                        Path of the output directory to write audio files in
  -f FILENAME_FORMAT, --filename_format FILENAME_FORMAT
                        Template string that will be formatted to
                        generatedoutput filename. Such template should be
                        Python formattablestring, and could use {filename},
                        {instrument}, and {codec}variables.
  -d DURATION, --duration DURATION
                        Set a maximum duration for processing audio (only
                        separate offset + duration first seconds of the input
                        file)
  -s OFFSET, --offset OFFSET
                        Set the starting offset to separate audio from.
  -c {wav,mp3,ogg,m4a,wma,flac}, --codec {wav,mp3,ogg,m4a,wma,flac}
                        Audio codec to be used for the separated output
  -b BITRATE, --birate BITRATE
                        Audio bitrate to be used for the separated output
  -m, --mwf             Whether to use multichannel Wiener filtering for
                        separation
  -B {tensorflow,librosa,auto}, --stft-backend {tensorflow,librosa,auto}
                        Who should be in charge of computing the stfts.
                        Librosa is faster than tensorflow on CPU and uses less
                        memory. "auto" will use tensorflow when GPU
                        acceleration is available and librosa when not.
```

Now that we know what we are doing - we run the tool for real, and will use the `-i` flag to define the input as our downloaded example, and the `-o` flag to define our output destination as the directory (folder) `output`. By default spleeter will download and use the2stems model.

```
!spleeter separate -i audio_example.mp3 -o output/
```

```
INFO:spleeter:Downloading model archive https://github.com/deezer/spleeter/releases/download/v1.4.0/2stems.tar.gz
INFO:spleeter:Validating archive checksum
INFO:spleeter:Extracting downloaded 2stems archive
INFO:spleeter:2stems model file(s) extracted
INFO:spleeter:File output/audio_example/accompaniment.wav written succesfully
INFO:spleeter:File output/audio_example/vocals.wav written succesfully
```

Another bash command `ls` (list) shows us the contents of our output directory



And finally onother couple of `audio` commands to hear our result!

Things to try

Check out the usage instructions for the separate tool on the Github site and try your own 4stem and 5tem separations. Use your own audio files to test the separation.

Speech to Text with Mozilla Deepspeech

Our next challenge will be to adapt the latest version of Mozilla's Deepspeech for use in Google Colab.

We will be using the documentation here:

https://deepspeech.readthedocs.io/en/v0.8.0/USING.html#getting-the-pre-trained-model

To adapt this colab notebook to run the latest version of Mozilla Deepspeech:

https://colab.research.google.com/github/tugstugi/dl-colab-notebooks/blob/master/notebooks/MozillaDeepSpeech.ipynb#scrollTo=4OAYywPHApuz

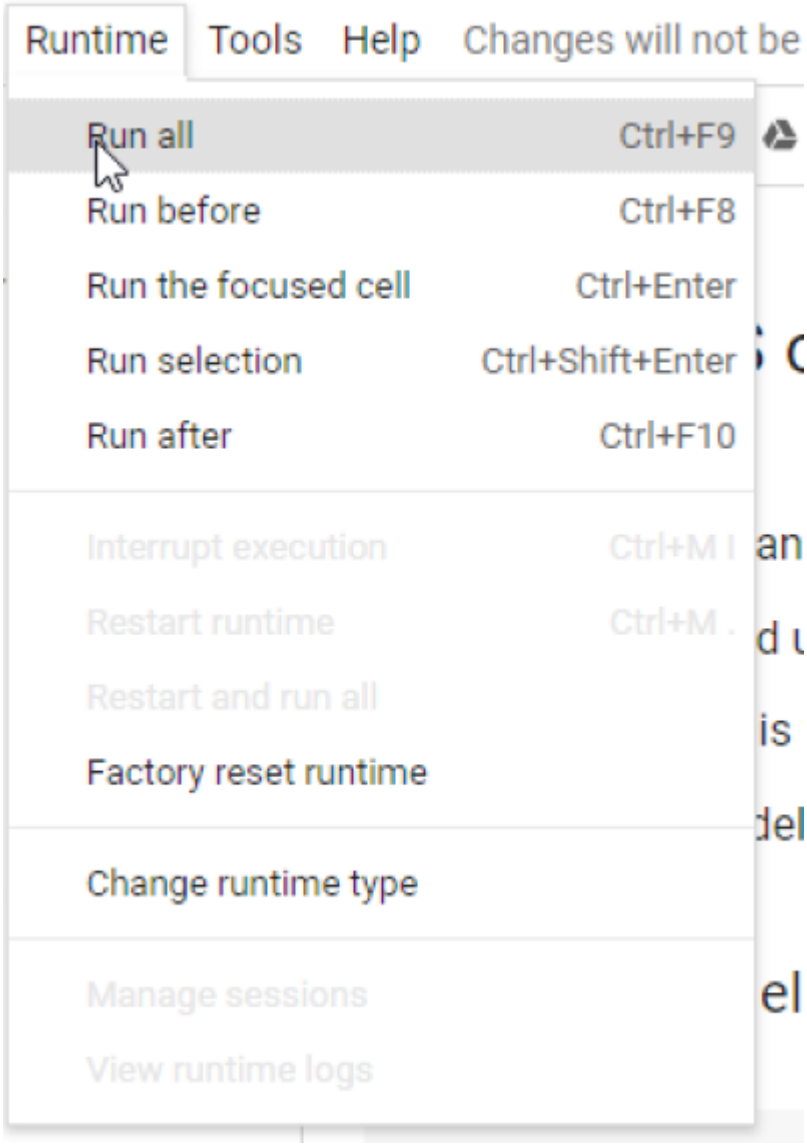
Text to Speech with Mozilla TTS

Our final example is TTS with Mozilla TTS:

https://colab.research.google.com/drive/1u_16ZzHjKYFn1HNVuA4Qf_i2MMFB9olY?usp=sharing#scrollTo=6LWsNd3_M3MP

You can dive straight into this and use it to generate speech. This example usesTacotron2 and MultiBand-Melgan models and LJSpeech dataset.

Run All Cells



Generate Speech

Going Further

ML is such a big and fast moving area of research there are countless other ways to explore and learn, here are a few two-minute videos to pique your interest:

- Video restoration
- OpenAI Plays Hide and Seek

Make sure you check out the resources in Lynda, which you will have free access to as a State Library of Queensland member

Links

https://machinelearningforkids.co.uk/#!/links#top

https://experiments.withgoogle.com/collection/ai

https://openai.com/blog/

1)
the ideepcolor training set is 1.3 million images
2)
this is obviously a contrived example of course, but the principle applies regardless
3)
if made available under an appropriate license
4)
By Internet Archive Book Images - https://www.flickr.com/photos/internetarchivebookimages/14776509983/Source book page: https://archive.org/stream/belltelephonemag19amerrich/belltelephonemag19amerrich#page/n78/mode/1upReference[Fig.4] The Voder Fascinates the Crowds from: Williams, Thomas W. (January 1940) I. At the New York World's Fair. "Our Exhibits at Two Fairs". Bell Telephone Quarterly XIX (1): 65."The Voder Fascinates the Crowds - The manipulative skill of the operator s fingers makes the Voders voice almost loo good to be true ", No restrictions, https://commons.wikimedia.org/w/index.php?curid=43343073
5)
https://deepmind.com/blog/article/wavenet-generative-model-raw-audio
6)
https://colab.research.google.com/notebooks/intro.ipynb
7)
https://simple.wikipedia.org/wiki/Python_(programming_language
8)
https://jupyter.org/
9)
this is not unusual when using a hosted runtime
10)
confusingly we need to call it from bash (with the exclamation
11)
a fancy way of saying option