Voice Gender and Age Recognition using Convolutional Neural Networks

Mark Truitt
Dr. Si Chen
West Chester University of Pennsylvania
CSC 490
April 2021

Project Summary:

By looking at spectrogram images (image conversions of audio files) of a person talking and identifying certain patterns in the image, it's possible to identify the rough age group and the perceived gender of the speaker. The intent of this project is to develop a neural network to be able to listen to, identify, and report back the perceived gender of the speaker in a voice clip. This has potential applications in security and in the alteration of voice for any reason (voice acting, speech pathology, transgender voice training, etc.) The project is programmed in Python using Tensorflow and Keras, plus Matplotlib, PIL, soundfile, Pydub, and MoviePy for data conversion and normalizing.

Existing Similar Works:

I found a handful of similar projects when I looked around to determine if something of this sort had been done ever before: One project called "Voice Gender", a solution by Nuance with no open source project or official name, general voice analysis apps such as Vocular, and a more advanced project called Speech2Face.

"Voice Gender" by Kory Becker (primaryobjects on Github) is a tool she developed in an interest to create a gender recognition program for use in transgender voice training. The project is written primarily in R and seems to use a smaller amount of machine learning, relying mainly on using statistics to identify the perceived gender of an input voice. It is trained on 3168 different voice samples from both male and female speakers, and doesn't include speaker age in its analysis. It has a high degree of accuracy on its training set, and a similarly high accuracy on its testing set.

Nuance is a company which offers business solutions for automated phone call menus intended to be used by people of any age. Nuance claims that it has developed a program which can detect "more than 1,000 "micro-characteristics" that the human ear can't process or comprehend" which can indicate the age and potentially gender of the speaker. From what I can tell, Nuance's primary focus is on making sure that they can identify the general age of a person on the phone to adjust the usability of the menu to accommodate people who may be less familiar with voice activated menus on the phone.

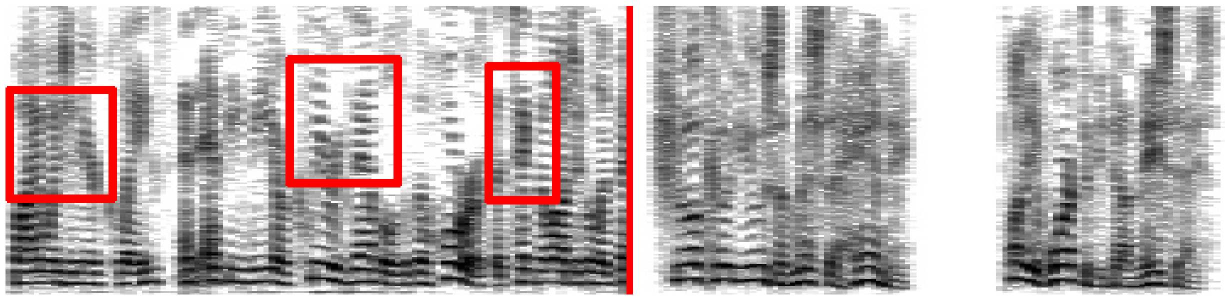
Vocular is an app available on Android and iOS platforms which allows a user to record a voice clip and analyze information such as the average, highs, lows, and variance of the pitch of their voice. It includes a large amount of voice analysis tools, and is capable of comparing a person's voice to the voices of famous celebrities. This could be loosely considered a way for the app to guess the perceived gender of a voice, but can encounter issues due to its use of pitch as a primary method for comparing voices, which isn't the only factor to the perceived gender of a voice.

Speech2Face is a larger-scale project written in Python using Tensorflow and Keras. Speech2Face is focused less on identifying the perceived age and gender of a user and outputting this information, but rather uses this information to generate a face based on the voice it hears. It uses spectrogram images to identify certain characteristics of a person's voice which are markers of their perceived age, gender, and ethnicity. The neural network has relative success with full faces; The generated faces won't match the actual face of the input exactly, but it will come oddly close, and will oftentimes get the shape of their nose very close to its actual shape.

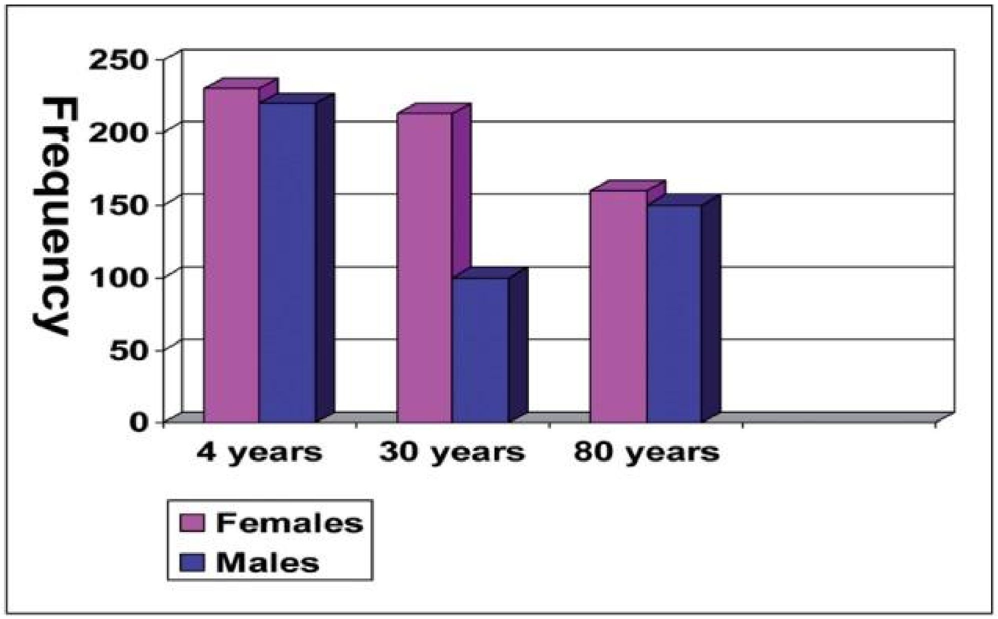
Project Basis:

Recognizing the perceived gender and age of a voice are very human things, much in the way recognizing a dog or cat are: It is a task which is easy for humans to do because our brains have developed to do so, and is hard to programmatically define, which is the exact place that AI becomes most useful in computer science. Our ears use the same sorts of information available in a spectrogram to determine this information, and research has found patterns in spectrograms which can be used to determine the perceived age and gender of a voice:

- **Perceived gender:** The perceived gender of a voice has a very clear set of markers on a spectrogram image of a voice recording. Spectrograms show the amplitude of the various frequencies in a sound. Voices tend to produce multiple frequencies, most of them resting on certain harmonic lines. For female voices, higher frequency harmonics tend to be stronger and more visible than in male voices. These indicators are boxed in red below for the female voice (right), while in male voices (right) the indicators are less frequent, less distinct, or not present in the spectrogram.



- **Perceived age:** The perceived age of a voice is less well documented in its presentation on a spectrogram. As a person ages, they undergo presbyphonia, the medical term for the weakening and thinning of the vocal cords. The weakening of the vocal cords creates the typical "shaky" voice older people tend to have, while the thinning of the vocal cords can create changes in the general pitch of a person's voice. As a person ages, the pitch of their voice tends to converge around the same point that the voice of someone of the opposite gender would. Both of these factors should be visible on spectrograms, though a human eye may not notice them as easily as a neural network may. (Figure on the next page) For children's voices, the average pitch of both male and female voices is higher than the average pitch for male or female adults, and both have similar resonances.



Project Process - Original versus result:

For this project, I initially expected to need to do a lot more sample collection and that I would need to do multiple rounds of training for the different categories. My original plan was as follows:

1. Collect voice samples using an online form, plus extract audio from roleplaying games *(for this, Bethesda's Fallout series proved to be the easiest and likely the best choice, both for ease of extraction and quality of audio)*
2. Create the learning system including the neural network(s)
3. Train gender male vs. female network and age child vs. adult vs. old network separately
4. Train a third network to combine the outputs of those networks
5. Train a fourth network to identify the same voice consistently
6. Release

As I worked, most of these steps started to seem less and less necessary. Originally I expected to need more than one neural network because I had assumed it wouldn't be able to learn to differentiate between multiple different categories, but the more I read and learned about neural networks, the more I realized this wasn't the case. I also very quickly discovered that in extracting audio from just four roleplaying games and processing it, I had well over 20,000 voice clips at least 5 seconds in length with a sufficiently high information density–There was no need to create a website and collect voice samples (which couldn't be reliably checked for quality control). I also determined that some of the "stretch goals" for the project, such as voice recognition, may not be necessary. My end process is as follows:

1. Extract audio from selected role playing games (*Fallout 3*, *Fallout: New Vegas*, *Fallout 4*, and *Fallout 76*)

2. Convert all audio files to mono-channel .WAV format, clip to 5 seconds/exclude shorter clips, and create spectrogram images of each clip
3. Feed these images into the neural network and tweak parameters until the network reaches an acceptable score on the training and testing data set
4. Adjust the program to take one input and output the neural network's perception as a point on a grid
5. Design a wrapper program which takes the output point-on-grid and stores the last result, then compares any new result and outputs whether or not it thinks it's the same user based on point-to-point distance
6. Release

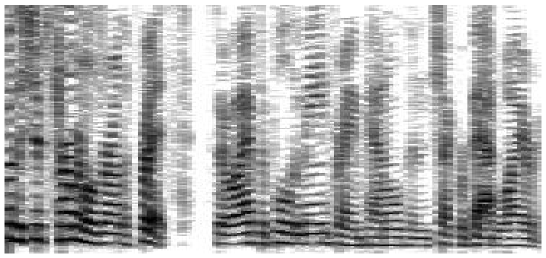
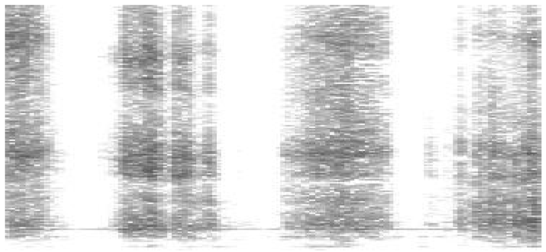
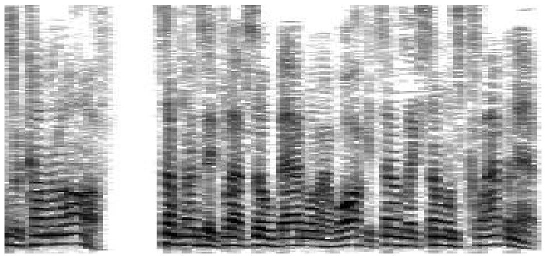
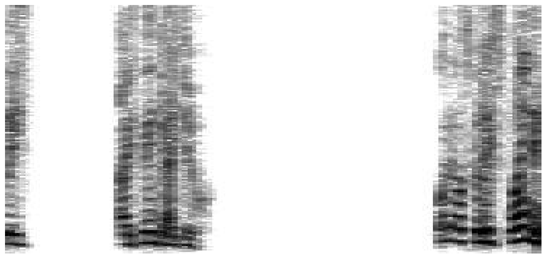
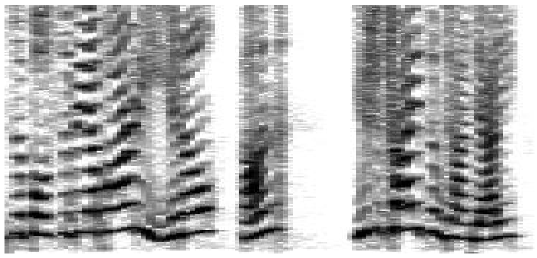
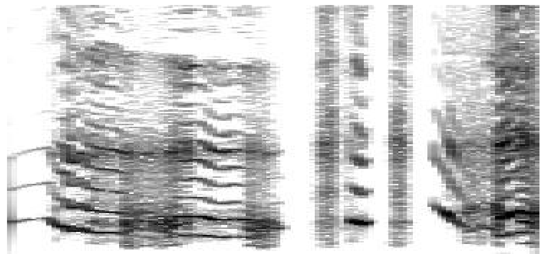
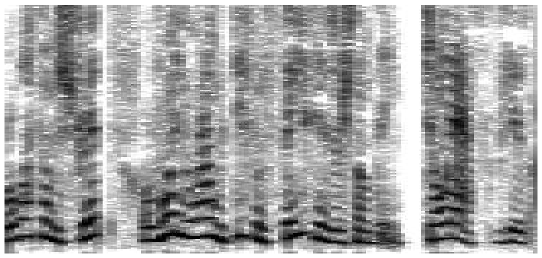
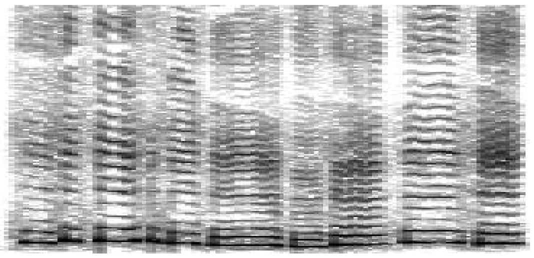
Data collection, conversion, and normalization:

As mentioned in the project process section, my original plan to collect voice samples from users very quickly proved to be unnecessary. When I originally extracted the voice line files from *Fallout 3*, *Fallout: New Vegas*, *Fallout 4*, and *Fallout 76*, I ended up with a grand total of 312,949 voice files of varying length. I needed to use a separate program to unpack some of the voice files, and then created a small C# program to recursively search through the folders and remove any non-audio files. I selected a small subset of the voice line sets (specifically choosing sets with the best data labelling– Ignoring anything where age/gender was unspecified, or the voice would likely be distorted such as in the case of robotic voices), which gave me a grand total of 132,447 voice lines.

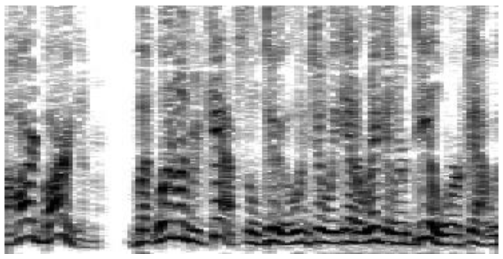
With these 132,447 voice clips (average length around 3 seconds, longest clip about 17 seconds long) separated from the rest, I set my program to work converting all of them into mono-channel .WAV format, and clip it to the specified clip length (5 seconds) to make sure the output images are all the same time. This uses Pydub, soundfile, and MoviePy. Filtering to only clips 5 seconds long or longer, the total size of the data set is reduced to 22,728 clips. Once converted and clipped, the audio clips are turned into spectrogram images 370px wide by 170px tall, which is then assessed for "information density". If the information density is less than the specified threshold (0.10), the program rejects the clip and skips its conversion. The end result of this process is 22,555 different sorted spectrogram images with an average information density around 0.2 and high of about 0.4, 16,917 of which are used for training and 5,638 of which are used for testing. The training likely has an easier time with adult male (11,433 samples) and adult female (7,331 samples) voices, then old female (1,332 samples) and old male (1,901 samples) voices, followed by child male (380 samples) and child female (178 samples) voices.

Data samples can be found below, showing a category representative for each of the six categories and a problem representative for each of the six main problem types spotted that could skew the training. The problem representatives are the "best of the worst"; Despite seeming like they may pose problems to the neural network, they pass the information density threshold nonetheless. Each problem has a brief explanation of why it may be a problem, and includes which data set the problem sample comes from.

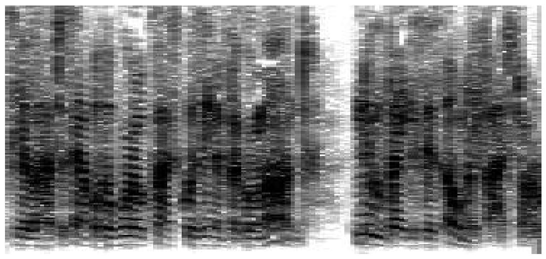
Data samples:

Category representative	Problem representative
Adult female 	Adult female  Noisy, no definite harmonics lines. Likely breathing with no speech, which means no gender/age information
Adult male 	Adult male  Large amount of empty space, potentially too little information to analyze
Child female 	Child female  Odd harmonics line patterns, potentially musical in nature, can cause problems
Child male 	Child male  Too many constant definite harmonics lines, may be difficult to classify

Old female

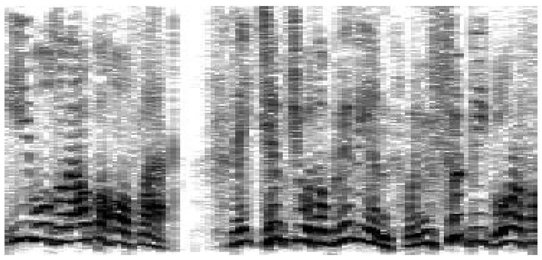


Old female

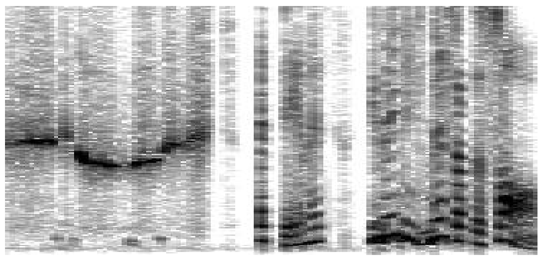


Potentially too information-dense, unable to clearly discern harmonics lines

Old male



Old female



Odd shape in the first half may pose a problem for the neural network

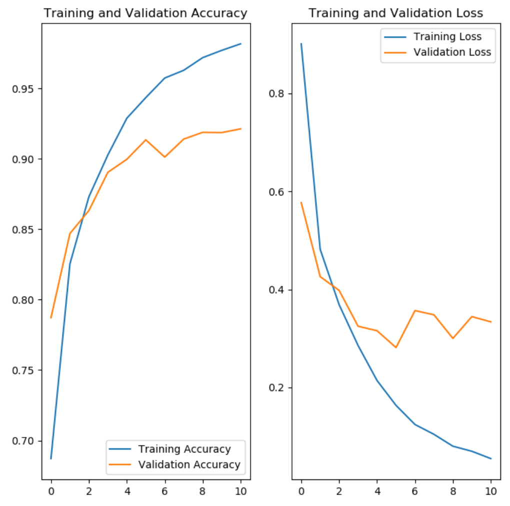
Neural network design:

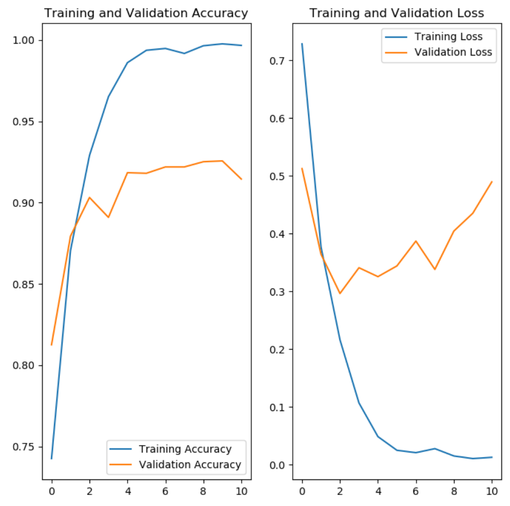
The finalized design for the neural network is a fairly typical image classification network. Originally, I expected I would need a much more complex neural network with a larger number of hidden layers, and hidden layers with more neurons in it. In addition to the three hidden layers of 16, 32, and 64 neurons, the network also has a dropout layer meant to help decrease the AI's chances of overfitting the data. Originally, I had set the neural network to train over the course of 20 epochs with a batch size of 65 images, expecting this to be a good starting point, but upped it to 50 epochs after erroneously thinking that the training had completed too quickly and that the results were invalid, despite showing promising levels of success on the training and test set. This has since then been bumped down to 11 epochs, which seems to be a good value to consistently get a good train with the current settings. The batch size remains the same, and had only been adjusted once or twice to determine what effect that would have on the training process.

The AI is currently using the best train out of multiple different training attempts for predictions; The validation loss has a tendency to trend upwards near the end which seems to be better or worse at random, though the validation accuracy remains fairly similar regardless of the change to validation loss.

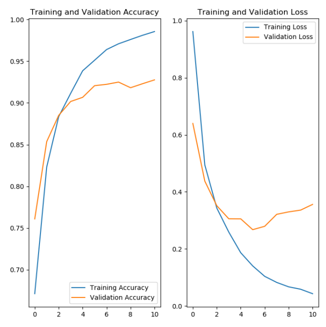
Training results:

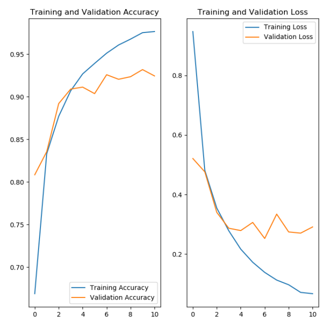
The end results of training after tweaking the values for the neural network a handful of times are shockingly accurate given how quickly the network trains. Despite cycling through 16,917 images per epoch, each step takes between 7 and 9 seconds and consistently increases the testing and validation accuracy in a fairly normal manner.

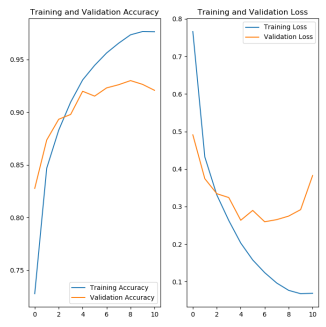




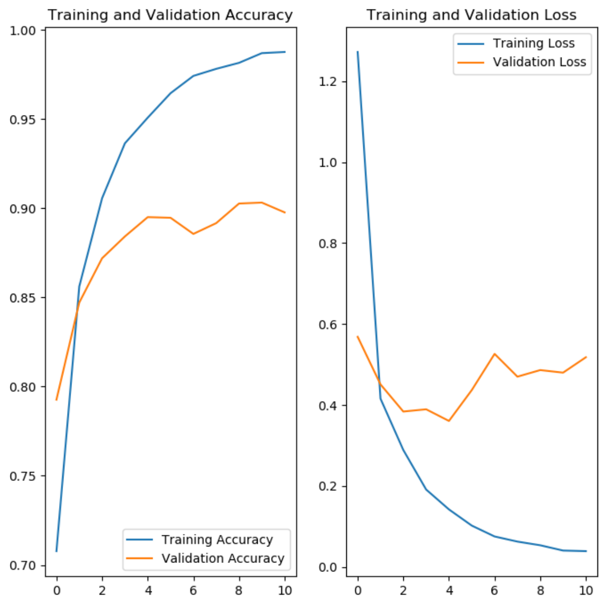
The efforts to avoid overfitting seem to be successful, but are most successful with a dropout value of 0.4 (shown above, right, versus the same network without the dropout layer, shown above, left). Higher and lower values increase the validation loss while training accuracy/loss plus validation accuracy seem to remain consistent with these changes. The three graphs below show the progress of training for a network with the dropout layer value set to 0.5 (left), 0.6 (middle), and 0.7 (right). The training for the network with a dropout value of 0.6 came the closest in terms of performance with the dropout value of 0.4, but this may be due to random chance.





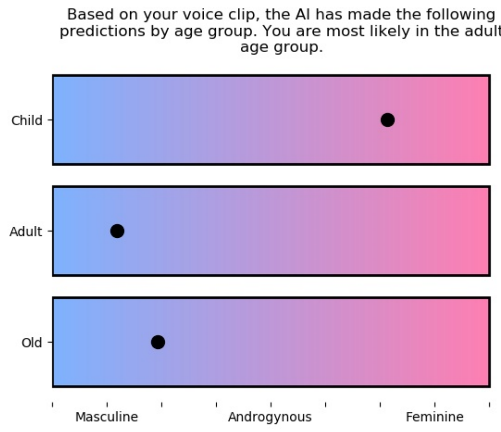


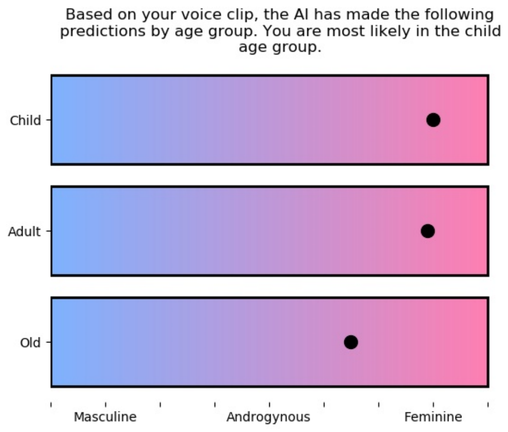
Regarding the size of the neural network, using the same general structure and layer settings as any given image classification network seem to be the best for this purpose. Removing one of the hidden layers to decrease the size of the neural network results in a much worse training result, with lower accuracy and higher loss nearing the end of the training. The training progress graph for this training session shows a major difference between training accuracy and validation accuracy, as well as the stagnation of the validation accuracy. The increase of the validation loss also indicates that the network is likely overfitting near the end, even with subpar accuracy results.

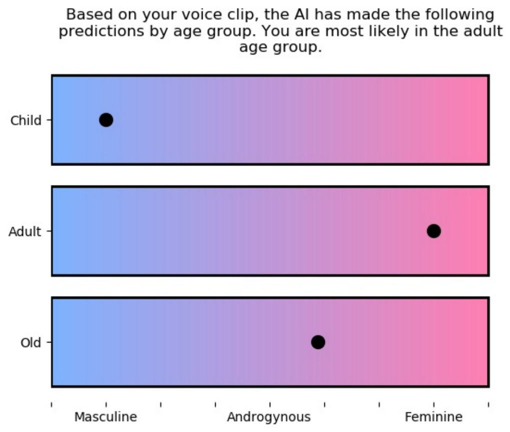


After training, I re-tested all of the data using a more granular method; The AI counts a result as incorrect if it fails on either the age or gender of the voice clip, which removes any possibility for "partial failure" which this neural network ideally *should* have. In doing this secondary hand-constructed testing, the results show the neural network is able to get voice clips correct or at least partially correct in a higher percentage of cases than the actual training and testing accuracies show. The neural network more often got the age of a voice wrong than the gender of a voice, which influenced how I generate the output image for a single prediction.

Hand-testing individual voice clips also shows that the model is capable of classifying voices correctly on a consistent basis; Despite my best attempts to throw a voice clip at the AI which it may struggle with (accents, background noise, quiet voice lines, etc.), it was able to correctly classify the voice's gender, and oftentimes age as well.







These outputs all show examples of how the AI responds to certain voices. The first output (top left) shows the output for an adult male voice; The AI thinks the voice most likely belongs in the adult age group, and within that group, it sounds male. If the person speaking knows their voice doesn't belong in that age group, they can look at the predictions for child and old voices to determine what gender the AI perceives their voice as. The same goes for the output for a child female voice (top right), and adult female voice (bottom middle).

The dot for any given age group tends to lie towards the ends of the gradient, and all dots are restricted to the masculine and feminine lines on the "X axis" of the graph. The reason for this is likely due to the fact that Tensorflow and Keras are geared to lessen the returned value for non-correct classes, and as such, it tends to very strongly lean towards one end of the spectrum or the other for the "most likely" age group.