speech2phone: Phoneme Segmentation and Classification from Raw Audio

Seong-Eun Cho[1] , **Jared Nielsen**[1] , **Kyle Roth**[1]
[1]Brigham Young University
scj1420@gmail.com, jaredtnielsen@gmail.com, kylrth@gmail.com

Abstract

We extend the speech recognition problem by converting raw audio data into discrete phonemes, rather than directly into language. For classifying phonemes, we combine several classical machine learning techniques such as dimensionality reduction with the expressive power of deep neural nets. We compare recurrent and convolutional neural networks and various embeddings, achieving 74.9% on the TIMIT dataset. We also attempt phoneme segmentation, getting an average error of 9.398 ms with precision 70.65% and recall 88.62% using a reinforcement learning model.[1]

1 Introduction

Speech recognition is a fundamental discipline of computational linguistics. It has traditionally been a very difficult problem to solve even with statistical models, but has recently seen marked improvement through the use of neural networks.

Similar to speech recognition, phoneme recognition is the challenge of transcribing audio input into a phonetic alphabet, e.g. the International Phonetic Alphabet (IPA). While a language's regular alphabet and spelling are decided by history and don't necessarily match pronunciation, the goal of a phonetic alphabet is to represent exactly the individual sounds produced in the utterance of language. This has value beyond speech recognition as it can help identify the accent of the speaker and provides valuable insight to field linguists.

In the first section of this project, we use several machine learning models as baselines to perform phoneme classification, and demonstrate how various embedding algorithms enhance the results of those models. In this task we provide the model with individual segments containing single phonemes as training examples.

The more interesting part of this project is to build a model for phoneme recognition without explicitly providing the segment of audio containing each phoneme. We divide up this task into phoneme classification (discussed above) and phoneme segmentation. For phoneme segmentation, an entire utterance is the input and timestamps for boundaries between phonemes are expected as output.

[1]See https://github.com/jarednielsen/speech2phone for code and examples.

a	ae	ah	ao	aw	adx	axr	ay
b	bcl	ch	d	dcl	dh	dx	eh
el	em	en	eng	epi	er	ey	f
g	gcl	h#	hh	hv	ih	ix	iy
jh	k	kcl	l	m	n	ng	nx
ow	oy	p	pau	pcl	q	r	s
sh	t	tcl	th	uh	uw	ux	v
v	w	y	z	zh			

Table 1: All 61 phonemes in the TIMIT corpus, in their text representations [Garofolo *et al.*, 1993].

2 Related Work

Automatic phonetic transcription has been attempted by Van Bael et al. [2007], Schiel [2019], and many others, but was limited in scope and didn't use neural networks. Chang et al. [2000] used temporal flow networks to achieve 80% accuracy, but on a dataset of only 37 words spoken repeatedly over 2.5 hours. Their approach used the output of an array of networks to then predict phonemes; each network was trained to estimate a specifically chosen phonetic feature, such as place, front-backness, voicing, rounding, and manner.

3 Dataset

We used TIMIT [Garofolo *et al.*, 1993], a corpus of spoken sentences annotated with phoneme information. It contains 630 English speakers with 8 distinct dialects, each reading 10 sentences. While TIMIT is not publicly licensed, we obtained permission to use the dataset from a professor in the linguistics department of Brigham Young University. Table 1 contains the phonemes from the TIMIT dataset. Some phonemes bear great similarity, but each distinction is meaningful in the English language.

The large number of classes makes this task more comparable to CIFAR-100[2] [Krizhevsky *et al.*, 2009] than to MNIST[3] [LeCun and Cortes, 2010]. The baseline of most-common-value (the vowel /a/) achieves only 4% accuracy on TIMIT,

[2]a dataset with 100 distinct classifications
[3]a dataset with 10 classifications

ground truth

segments	1819		2640		3926		4200		4840		6809		7400		8187		9480		10360
phoneme		k		em		pcl		p		er		ax		bcl		el		tcl	

prediction

segments	1800		2300		2900		3800		4400		4900		7600		8300		9500		10300
top three		k		epi		en		pcl		b		er		bcl		uh		kcl	
predictions		g		kcl		em		bcl		p		axr		dcl		el		ix	
		t		pau		m		dcl		d		uh		b		ax		n	

Table 2: Snippet of audio from the test set segmented using the reinforcement learning agent. The segments created by the model are listed along with the resulting top three classifications given by the RNN model.

so we do not expect to see high accuracy scores. Rather, the question is how well we can improve on the baselines we establish.

4 Embeddings

Choosing an efficient representation of speech data is important, because segments of raw audio can lie in $\mathbb{R}^{1000}$ or more. For this reason it's important to be able to project the data into a lower-dimensional space. This map is referred to as an embedding. Embedding algorithms we used include resampling, the Mel spectrogram, random projections, and t-SNE. Each embedding was used in the preprocessing step for each model, and the results are reported in the Experiments section.

4.1 Resampling

Phonemes vary in length. Vowels tend to be longer than consonants, and the shortest phonemes are often stops like /d/. (Figure 3 demonstrates the mean and variance of each phoneme in TIMIT.) Many of the models we use require input of fixed length, meaning that the raw audio must be resampled to the correct size in order to be passed as input to these models. One method of resampling involves interpolation of the raw audio samples, and then evaluation of the interpolating function to get the resampled points [Rajamani *et al.*, 2000].

Short phonemes are often difficult to recognize because there are so few samples that the variance overpowers the signal. To provide better signal we pad every phoneme with up to 500 samples on either side. This increased the accuracy of our models by about 10%. Unfortunately, padding also means that fewer samples actually come from the window where the phoneme is expressly being uttered; this could increase the model's confusion between phonemes that appear in similar contexts (e.g. /d/ and /t/ in "bat" and "bad").

4.2 Mel spectrogram

The Fourier transform is a common signal processing technique that finds the representation of a segment of audio in terms of linear combinations of frequencies [Wikipedia contributors, 2019]. An example of the difference between raw audio and the Fourier transform is shown in Figure 4. The melody filter (or mel filter) was developed to draw out the frequencies in a way that represents the range perceived by the

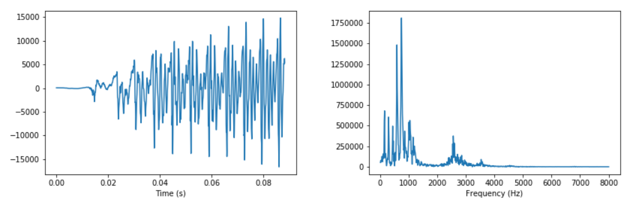

Figure 1: On the left, the raw audio waveform of one utterance of the phoneme /b/ is plotted over time. On the right, the same audio has been mapped to the frequency domain using the discrete Fourier transform. Generally, similar-sounding phonemes have similar representations in the frequency domain.

human auditory system [Stevens, 1937]. The intuition of its use as an embedding is that this exact type of preprocessing occurs in the beginning stages of human auditory processing as well.

5 Experiments: Classification

We start by producing simple baselines for classifying phonemes using common machine learning models. These achieve impressive performance given the 61 possible classes. We also train neural networks of two kinds, a residual network with 1-D convolutions and a recurrent neural network with LSTMs. We then compare results of these models on different embeddings.

5.1 Random Forests

Random forests consistently lie among the top-performing classical machine learning methods [Caruana *et al.*, 2008]. Random forests are an ensemble of decision trees, with each decision tree trained on a subset of the training features and a subset of the training data [Koehrsen, 2019]. We performed hyperparameter optimization over the number of decision trees included.

5.2 KNN

K-nearest neighbors (KNN) is an approach that decides the classification of a new data point by taking the mode of the classes of its k nearest neighbors in the data space. The accuracy of this model depends heavily on the choice of embedding. KNN has extremely low performance on raw audio, because audio segments that may be very similar but are shifted by half a wavelength will have a very large Euclidean distance. But over the right embedding space, KNN could provide a metric for an audio segment's "phoneme-ness", or likelihood of being a phoneme. This will be useful for the segmentation task in future work.

5.3 RNN

Recurrent neural networks (RNN) allow neural networks to take input with variable size. We use a simple 1-D Convolution banks to downsize a sample of raw signal and run the variable-sized output through three layers of Long Short-Term Memory (LSTM) cells [Hochreiter and Schmidhuber, 1997] which summarize the output to a single vector. We then use two feed-forward layers to obtain a probability distribution of outputs. We achieved 73.0% accuracy with this approach.

5.4 1-D CNNs

Convolutions have shown great success in speech and image classification, so we adopt the 1-D convolutional neural net (CNN) described by Rajpurkar et al. [2017]. Their architecture is a deep CNN with 33 layers, residual connections, and an increasing number of channels combined with decreasing dimensionality. We found residual connections necessary to train anything deeper than 3 layers. Opting for a simpler architecture, we used only 9 layers and achieved 76.1% top-1 accuracy and 92.3% top-3 accuracy.

6 Experiments: Segmentation

6.1 Reinforcement Learning

In a reinforcement learning environment, the problem is represented by

- an agent taking an action,
- a space of all possible actions that the agent can make,
- a space of possible states which the agent observes when making an action, and
- a reward the agent receives from making an action.

The goal is to train the agent to maximize the reward it can get by choosing actions that cause favorable states.

Consider a sliding window stretching out a certain distance from the beginning of an audio segment. We can represent the segmentation problem as a reinforcement learning environment by considering this window to be an agent that can either accept the current segment as a phoneme or expand the window to consider more samples of audio. We then define the state as the audio samples currently inside the window. Thus at each stage the window decides either to "expand" to consider more audio or "match" the current audio. The process then repeats starting at the end of the matched segment from the completed iteration.

For the reward function, we trained a machine learning model that is able to identify whether a given sample of audio is a phoneme segment or not. The architecture of this model is identical to the classification model, but instead of outputting a probability distribution, it outputs a single value that represents how likely it is that a given sample is a phoneme.

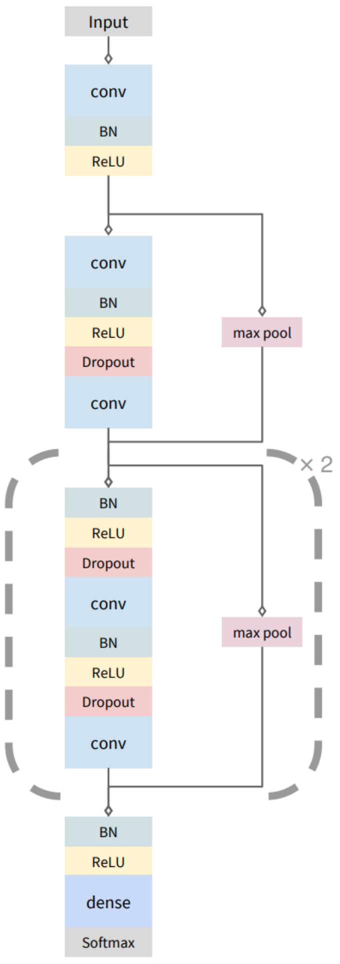

Figure 2: Architecture of the 1-D convolutional ResNet.

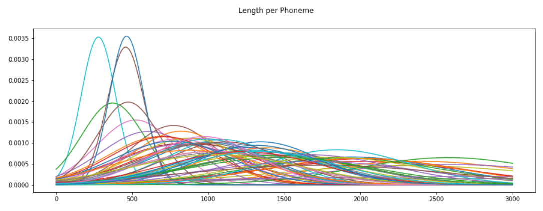

Figure 3: Density functions of the normal distributions matching the experimental mean and variance of each phoneme's length in TIMIT. The shortest phonemes were /b/, /d/, and /ɹ/ (e.g. "d" in "muddy"), while the longest were /oi/ ("oy" in "boy"), /au/ ("ou" in "about"), and /æ/ ("a" in "bat"). Note the power-law distribution of mean lengths.

We accomplished this task using the same TIMIT training data, and we generated a batch of random audio segments (not phonemes) from the same training audio.

Let U be the space of all possible audio segments of our training data and $P \subset U$ be the space of phoneme audio segments. $x \sim P$ is an audio segment from the training data and $z \sim U$ is a randomly generated audio segment. Let $R : U \to \mathbb{R}$ be a score function, mapping from the audio segment to a real number (high score means a likely phoneme, low score means not a phoneme). We attempt to minimize the loss:

$$\begin{aligned} L =& \mathbb{E}\left[\text{clip}\left\{R(z), -10, 10\right\}\right] \\ &- \mathbb{E}\left[\text{clip}\left\{R(x), -10, 10\right\}\right] \\ &- \mathbb{E}\left[\left|R(z) - \text{clip}\left\{R(z), -10, 10\right\}\right|\right] \\ &- \mathbb{E}\left[\left|R(x) - \text{clip}\left\{R(x), -10, 10\right\}\right|\right] \end{aligned}$$

where $\text{clip}\{a, b, c\} = \min\{\max\{a, b\}, c\}$.

Intuitively, minimizing the loss function will minimize the score we get from the randomly generated segments and maximize the score of the actual phoneme segments. We restrict the scores to be between -10 and 10 to avoid exploding scores, but we penalize the score for leaving this boundary. These penalties help the optimization avoid local minima. The explanation for each term in the loss is as follows.

- Minimize the output for random audio segments.
- Maximize the output for segments containing phonemes.
- Penalize scores outside $(-10, 10)$ for random audio segments.
- Penalize scores outside $(-10, 10)$ for segments containing phonemes.

Once we have a trained recognizer, we can use the scores it outputs as the reward for our reinforcement learning environment. We use Proximal Policy Optimization (PPO) algorithm to accomplish the reinforcement learning task [Schulman *et al.*, 2017]. Since the states are segments of audio that are of variable length, we use the output of the LSTM so that the states are always a fixed size. The policy network and the value network of the PPO implementation are both composed of 5 fully connected layers. The policy network outputs a probability distribution of each action to take, and the agent samples an action from that distribution. This way, the agent can explore less probable actions while exploiting more probable actions. It then uses the output of the value network of the same input state in order to learn a better action distribution.

6.2 Scoring

Since there isn't a standardized way of scoring a time series segmentation, we devised a distance metric for positively matched segments and also report the precision and recall. We start by pairing each predicted segment to the closest target segment. Some predicted segments will be paired with the same target segment, and so we remove the ones that are more distant. The unpaired predicted segments are counted as false positives while the unpaired target segments are counted as false negatives. Using these scoring methods, the reinforcement learning segmentation model positively matched segments within 9.398 ms on average, with precision of 70.65% and recall of 88.62%.

Batch Size	Top-1 Accuracy	Top-3 Accuracy
16	57.2	84.1
32	62.5	87.0
64	66.6	89.6
128	67.8	92.2
256	69.2	92.1
512	69.6	**92.6**
1024	71.8	92.0
2048	**74.9**	92.1
4096	67.7	90.0

Table 3: Effect of batch size on test accuracy. Experiments were performed with the 1-D convolutional ResNet for 10 epochs.

7 Optimization Techniques

Hyperparameter Search

Hyperparameter selection is of paramount importance. Some techniques, in order of increasing quality, are manual search, grid search, random search, and Bayesian optimization. Manual search, also called graduate student descent [sciencedryad, 2014], involves a human trying one configuration, then trying another, and so on. Grid search tries all possible combinations of various hyperparameter configurations which are manually chosen. Random search tries random combinations within the grid, and has been shown to perform better than grid search [Bergstra and Bengio, 2012]. Bayesian optimization uses prior evaluations to construct a probability distribution (such as a Gaussian mixture model) over the hyperparameter space, similar to the multi-armed bandit problem. Bayesian optimization has been shown to produce superior results, so we use it via the *hyperopt* package [Bergstra *et al.*, 2013].

Batch Size

We found that larger batch sizes improved test accuracy, with a size of 512 performing best. At the one extreme, using the entire dataset (batch size 132,000) to compute a single gradient is a slow, yet highly accurate gradient estimate, with little variance. At the other extreme, using a single draw of data (batch size 1) gives a fast, yet possibly inaccurate gradient estimate, with high variance. We desire the golden mean of these two extremes, with enough accuracy to converge to a global minimum but enough variance to escape poor local minima.

Learning Rate

The learning rate is crucial to training, but selecting an optimal fixed value is difficult. Simulated annealing, where the learning rate is gradually decreased over time, consistently outperforms fixed learning rates. Cyclical learning rate schedules do even better, but can be tough to tune and require stochastic gradient descent instead of the Adam optimizer. Since algorithms, architectures, and embeddings are intrinsically more interesting than hand-tuned learning rate

LR Schedule	Top-1 Accuracy	Top-3 Accuracy
None	69.3	90.2
StepLR	71.2	91.1
ReduceLROnPlateau	**72.4**	**92.9**
ExponentialDecay	70.0	91.1

Table 4: Effect of learning rate schedules on accuracy of model on the test set. Top-1 refers to the accuracy of the classification given highest probability by the model, and top-3 refers to the accuracy of the three classifications with highest probability. Experiments were performed with the 1-D convolutional ResNet for 30 epochs, batch size 512, Adam with initial learning rate 1e-2. StepLR multiplies the learning rate by 0.1 every 10 epochs. ReduceLROnPlateau multiplies the learning rate by 0.1 whenever validation accuracy does not improve for three epochs in a row. ExponentialDecay multiplies the learning rate by 0.9 every epoch.

Model	mel	random proj.
log. reg.	0.344	
FCNN	0.348	
random forest	**0.415**	
XGBoost	**0.357**	0.109
KNN	**0.377**	
QDA	0.232	
Length k-means	0.053	

Table 5: Accuracy scores for classification on TIMIT using various models and embeddings. The best three appear in bold. Length k-means was a classification based on purely the length of the segment.

schedules, we took the middle ground of automatically decreasing the learning rate upon plateau—whenever validation accuracy hadn't improved for three epochs, the learning rate was divided by 10. We believe this approach is a vast improvement over selecting a fixed rate, and it requires no additional manual rate tuning.

8 Results

Table 5 lists the accuracy scores for classification on TIMIT using several of the above-mentioned models and embeddings. It is interesting to note the high performance of KNN on the mel embedding, which implies that mel is a good separating representation. We may be able to use this embedding as a loss function to train segmentation models in the future, as described in the Future Work section.

Figure 4 contains the confusion matrix for the 1-D CNN model, which was the best-performing model we trained. The best models we trained were an RNN and a 1-D CNN. They achieved 73.0% and 76.1% accuracy respectively.

Table 2 shows a small example of how the segmentation agent and the classification model transcribe a section of audio into phonemes. The top three predictions of the phonemes for each segments can then be used to graph a statistical model that produces sentences in the desired language.

9 Future work

We plan to use our classification models to create a new loss function for training segmentation models, instead of using the "phoneme-or-not" model. In particular, we will perform more experimentation to determine a better reward function for our reinforcement learning approach. Other models for segmentation could be devised by embedding the training data in a representative space and then using the distance from k nearest neighbors as a metric for "phoneme-ness".

We also plan to experiment with the hyperparameters of the embeddings we used, as well as new embedding algorithms like UMAP [McInnes *et al.*, 2018]. Our results demonstrate the value of good preprocessing when working with highly correlated data like audio.

TIMIT is a relatively small dataset, and we feel that we could use semi-supervised learning techniques to help models generalize to a wider variety of data. Semi-supervised learning will likely improve the performance on utterances from speakers unseen by the model.

With similar datasets from other languages, we could also build a phoneme recognition model that could transcribe any spoken language into phonetic alphabets, which makes a universal acoustic model possible. We can then build a statistical language model specific to each language, similar to how speech recognition is traditionally done, which would graph the phonemes into standard orthography.

10 Ethical Considerations

To our knowledge, all data has been collected in an ethical way, from speakers who provided informed consent for their spoken audio to be used for research purposes. Since it's possible to automatically recognize the individual phonemes from audio, it's also possible to recognize the identity of the speaker. Depending on the use of the model in a real-world system, speaker identification could be unethically used to identify and make decisions affecting the speaker, without his or her informed consent [Pastukhov and Kindt, 2016]. We doubt that our model could be used for effective speaker identification because the purpose of the embedding is to represent the same phoneme the same way (regardless of the speaker). Nonetheless, we plan on doing experiments in the future to see whether our model is sensitive to the identity of the speaker.

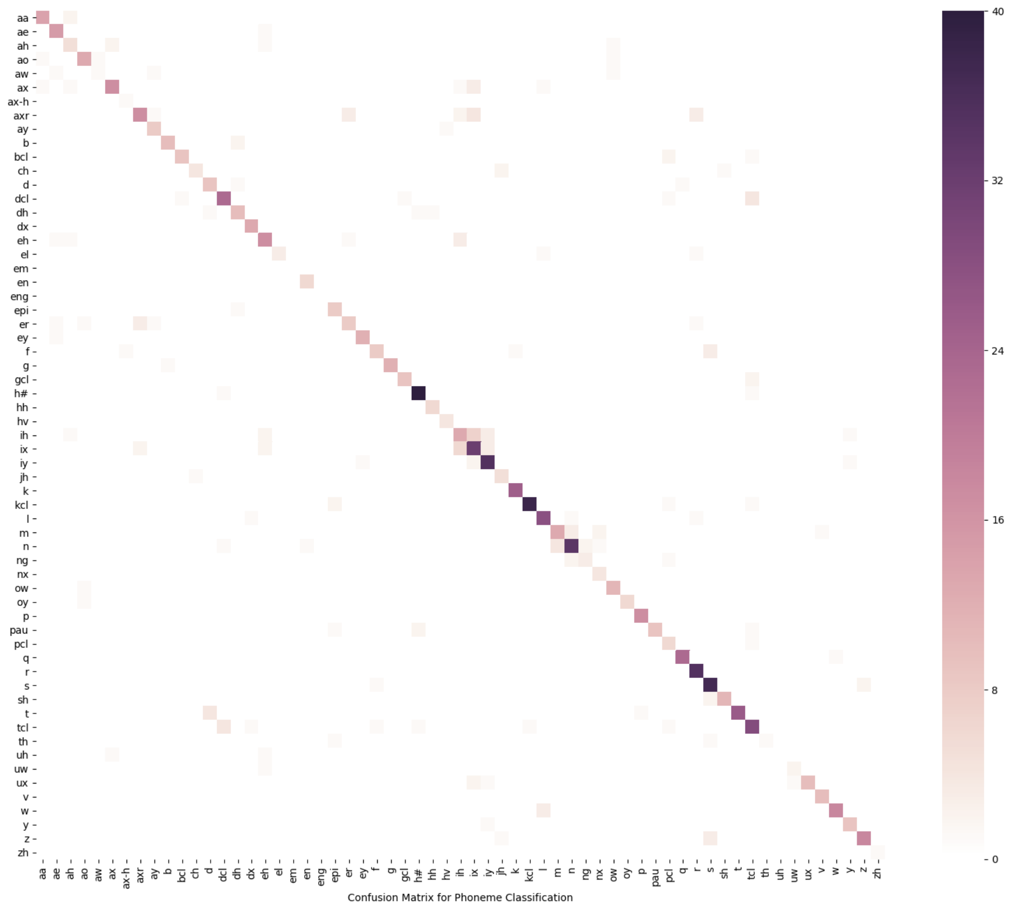

Figure 4: Confusion matrix for phoneme classification on the test set with the 1-D convolutional model.

References

[Bergstra and Bengio, 2012] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[Bergstra *et al.*, 2013] James Bergstra, Dan Yamins, and David D. Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms, 2013.

[Caruana *et al.*, 2008] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 96–103, New York, NY, USA, 2008. ACM.

[Chang *et al.*, 2000] Shuangyu Chang, Lokendra Shastri, and Steven Greenberg. Automatic phonetic transcription of spontaneous speech (American English). In *In International Conference on Acoustics Speech and Signal Processing*, pages 330–333, 2000.

[Garofolo *et al.*, 1993] J S Garofolo, L F Lamel, W M Fisher, J G Fiscus, D S Pallett, and N L Dahlgren. DARPA TIMIT acoustic phonetic continuous speech corpus CDROM, 1993.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[Koehrsen, 2019] William Koehrsen. Build a random forest algorithm. *enlight*, August 2019.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (Canadian institute for advanced research). 2009.

[LeCun and Cortes, 2010] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[McInnes *et al.*, 2018] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[Pastukhov and Kindt, 2016] Oleksandr Pastukhov and Els Kindt. Voice recognition: Risks to our privacy. *Forbes*, Oct 2016.

[Rajamani *et al.*, 2000] Kannan Rajamani, Yhean-Sen Lai, and C W Farrow. An efficient algorithm for sample rate conversion from CD to DAT. 7(10), Oct 2000.

[Rajpurkar *et al.*, 2017] Pranav Rajpurkar, Awni Y. Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *CoRR*, abs/1707.01836, 2017.

[Schiel, 2019] Florian Schiel. Automatic phonetic transcription of non-prompted speech. Apr 2019.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[sciencedryad, 2014] sciencedryad. Grad student descent, 2014.

[Stevens, 1937] S. S. Stevens. A Scale for the Measurement of the Psychological Magnitude Pitch. *Acoustical Society of America Journal*, 8:185, 1937.

[Van Bael *et al.*, 2007] Christophe Van Bael, Lou Boves, Henk van den Heuvel, and Helmer Strik. Automatic phonetic transcription of large speech corpora. *Comput. Speech Lang.*, 21(4):652–668, October 2007.

[Wikipedia contributors, 2019] Wikipedia contributors. Fourier transform — Wikipedia, the free encyclopedia, 2019. [Online; accessed 2019-04-09].