Bottleneck Features for Automatic Speaker-Independent Vowel Representation

Arvid Fahlström Myrman
School of Computer Science and Communication
KTH Royal Institute of Technology
100 44 Stockholm, Sweden
arvidfm@kth.se

Abstract

A supervised dimensionality reduction technique using artificial neural networks with a narrow bottleneck layer is presented and applied to the task of automatically finding a two-dimensional representation of American English vowels. The network is shown to be able to find a good vowel representation directly from speech frames, with the resulting vowel space being relatively speaker invariant compared to the traditional formant-based vowel space.

1 Introduction

Vowel visualisation has been employed in areas such as language learning (Dowd et al. 1998; Paganus et al. 2006; Wik and Escribano 2009) and speech-language pathology (e.g. VowelViz,[1] specifically developed for use in speech therapy), where a visual representation provides the user with real-time feedback on their pronunciation of vowels. One common approach is to represent the vowel as a point on a two-dimensional plane, with the goal being for the user to "home in" on the target vowel to the best of their ability.

The most common method to project a vowel onto low-dimensional space is to represent it in terms of its formants, the peaks in the power spectrum that are said to be vital for vowel perception (Peterson and Barney 1952). This results in a relatively intuitive vowel representation, as the formants are strongly correlated with mouth (e.g. tongue and lip) configuration (Ladefoged 1982, ch. 9). For instance, the first formant, F1, of the high vowel /iː/ in /biːt/ *beat* might typically be around 300 Hz, while the F1 of the low vowel /æ/ in /bæt/ *bat* would be closer to 800 Hz in modern RP (De Jong et al. 2007).

However, some issues arise with this approach: Most importantly, even for vowels with the same perceived quality, there is significant variation between speakers when it comes to the mean frequencies of the formants (Peterson and Barney 1952), and the perceived quality of a vowel is not determined directly by the absolute values of its formants (Traunmüller 1981; Fahey et al. 1996). Additionally, naive methods of formant extraction, such as peak picking in the frequency response of the linear predictive filter, is known to suffer from peak merging, where two formants with similar frequencies cannot be distinguished as separate formants (Markel 1972).

While there exist a variety of methods for speaker normalisation and more robust formant extraction, I propose a radically different approach: Using a non-linear supervised dimensionality reduction technique, a low-dimensional vowel representation can be found *automatically*, with no prior knowledge of the structure of the audio signal other than what vowel class it belongs to. Specifically, I propose to train a bottlenecked artificial neural network

[1] http://completespeech.com/vowel-viz/vowelviz-overview/what-is-vowelviz/

to map speech frames to the corresponding vowel class. A bottleneck layer is responsible for finding a low-dimensional representation that the following layers can still use to discriminate between the vowels, and by preserving information about the classes corresponding to the input, the network is encouraged to find a speaker-independent representation.

The problem of automatically modelling the vowel space using neural networks has been studied previously in Nearey and Kiefte (2003), where a neural network was used to map a set of K stimuli in the form of a 1-of-K-encoded input to the corresponding vowel class, through a hidden layer with two units. As the input is not in the form of speech frames, but rather an index corresponding to some abstract stimuli, the resulting scheme is akin to that of an autoencoder performing dimensionality reduction on discrete vowel classes.

The authors conclude that as their two-dimensional model was unable to fit the data to a satisfactory degree, this provides evidence that the underlying vowel space must be at least three-dimensional. However, the fact that they only used a single hidden layer means that the mapping from the two-dimensional representation to vowel classes must be performed using a linear logistic classifier, in which case the network will perform poorly unless all vowels are linearly separable. It is possible that the model would have performed better with a deeper network.

In a related vein, Nagayama et al. (1994) describes performing dimensionality reduction using Sammon mapping, after which a neural network is trained to learn the non-linear mapping, so that the dimensionality reduction can be extended to unseen data. The network is trained to produce the dimensionality-reduced vowels as output rather than using bottleneck features. No particular precautions are taken to ensure that the low-dimensional representation is speaker invariant.

There has also been research on the general case of generating features using bottleneck layers in neural networks. Grézl et al. (2007) describes a method of generating features for use with a GMM-HMM speech recognition system by injecting a narrow layer into a phone classifier. Yu and Seltzer (2011) further builds on this approach by initialising the classifier using restricted Boltzmann machine pre-training. Sivaram and Hermansky (2012) takes a related approach by generating sparse features of the same dimension as the input, enforcing sparsity in the bottleneck layer using a differentiable regularisation term.

2 Method

2.1 Feedforward neural networks

A feedforward neural network, also known as a multi-layer perceptron, is a simple network composed of layers, which are in turn composed of so-called units. Every unit calculates a weighted sum of the outputs of the previous layer's units. The sum is then passed to an activation function, whose output is taken as the output of the unit.

The calculations can be expressed in terms of matrices and vectors, by considering the output of layer l as a row vector $\mathbf{y}^{(l)}$ of dimension $n^{(l)}$, with the corresponding weight matrix $\mathbf{W}^{(l)}$ of dimension $n^{(l-1)} \times n^{(l)}$, and bias (threshold, intercept) vector $\mathbf{b}^{(l)}$ of dimension $n^{(l)}$. The output of layer $l+1$ is then given by

$$\mathbf{h}^{(l+1)} = \mathbf{y}^{(l)}\mathbf{W}^{(l+1)} + \mathbf{b}^{(l+1)}$$
$$\mathbf{y}^{(l+1)} = \mathbf{f}^{(l+1)}(\mathbf{h}^{(l+1)})$$

where $\mathbf{f}^{(l)}(\mathbf{h})$ is the activation function of layer l.

The first layer is referred to as the input layer, with $\mathbf{y}^{(0)} = \mathbf{x}$ where $\mathbf{x}$ is the input. Layer N, where N is the number of non-input layers, is the output layer. In the case of classification, the output layer generally corresponds to the posterior distribution $y_i^{(N)} = P(y = i \mid \mathbf{x})$. The $N-1$ layers between the input and output layers are referred to as *hidden* layers.

The network is trained by feeding it input data with known corresponding output data, and minimising an objective function comparing the expected output to the actual output of the network. The objective function is minimised using gradient descent by calculating

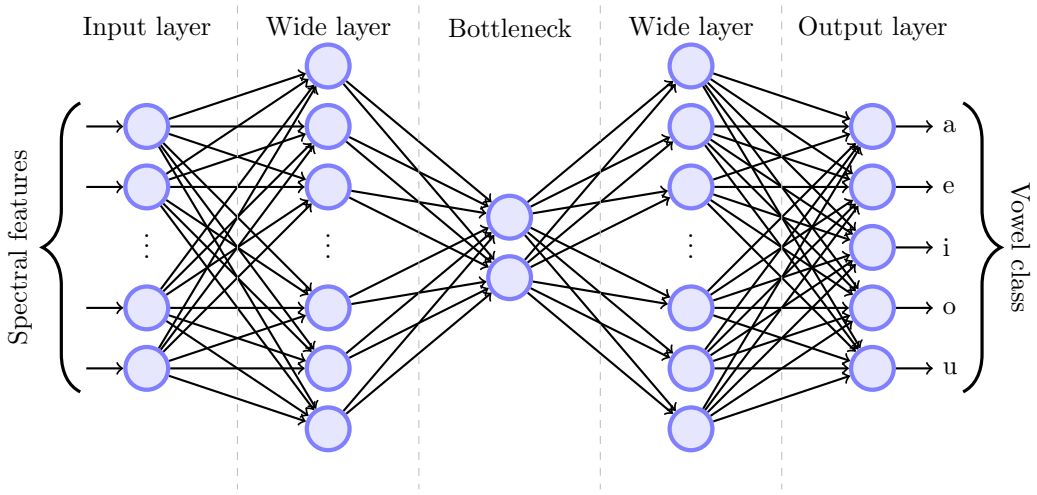

Figure 1: An example of a bottlenecked feedforward neural network mapping speech frames to vowel classes. The network contains three hidden layers, of which the middle layer is the bottleneck layer.

the gradient with respect to the weights and biases. The algorithm commonly used for calculating the gradient is known as the back-propagation algorithm (Rumelhart et al. 1986).

2.2 Vowel space modelling

A feedforward neural network is constructed to map speech frames to vowel classes. The outputs are represented using a 1-of-K encoding, with one output per vowel class. To model the vowel space a narrow bottleneck layer with few units is inserted into the network. The bottleneck layer forces the network to find a stable representation of the vowels in low-dimensional space, that can still be used by later layers to discriminate between the vowels. See fig. 1 for a graphical illustration of such a bottlenecked network.

The vital difference between this approach and unsupervised dimensionality reduction techniques such as autoencoders or principal component analysis (PCA), is that this is a *supervised* dimensionality reduction technique. By explicitly telling the network what vowel class each speech frame belongs to, the network is encouraged to find a common representation for all vowels of the same class, even if the features used as input look quite different.

This work focuses primarily on two-dimensional vowel representations. This is mainly to ease the interpretability of the representation, though it also grounded in the hypothesis that two dimensions are enough to represent the vowel space of American English dialects: While height (F1) and backness (F2) are contrastive in American English, no two vowels are discriminated between using only lip rounding (F3) (Ladefoged 1999).

The activation function to use for the bottleneck layer requires some deliberation. While a sigmoid function such as tanh might at a glance seem appropriate as it bounds the representation within the range of the function, its tendency to saturate for large input values risks yielding a representation where vowels tend to be "smeared" against the bounding box.

A better choice is a non-sparse activation function exhibiting locally linear behaviour over most of its domain. An obvious such example is a fully linear function such as the identity function $f^{(k)}(\mathbf{h})_i = h_i$, where k is the bottleneck layer. Another function satisfying the requirements is the maxout activation function (Goodfellow et al. 2013). As is standard for classification, the activation function of the output layer is the softmax function:

$$f^{(N)}(\mathbf{h})_i = \frac{e^{h_i}}{\sum_j e^{h_j}}.$$

Table 1: Vowels in TIMIT. Examples are taken from the TIMIT documentation.

TIMIT vowel	IPA symbol	Example word	Comment
		Monophthongs	
iy	[i]	beat	
ih	[ɪ]	bit	
eh	[ɛ]	bet	
ae	[æ]	bat	
aa	[ɑ]	bot	Father-bother merger
ah	[ʌ]	but	
ao	[ɔ]	bought	
uh	[ʊ]	book	
uw	[u]	boot	
ux	[ʉ]	toot	Fronted /u/
er†	[ɝ]	bird	
		Diphthongs	
ey	[eɪ]	bait	
aw	[aʊ]	bout	
ay	[aɪ]	bite	
oy	[ɔɪ]	boy	
ow	[oʊ]	boat	
		Weak/marginal vowels	
ax	[ə]	a̲bout	
ix	[ɨ]	debi̲t	
axr	[ɚ]	butte̲r	R-coloured /ə/
ax-h	[ə̥]	su̲spect	Devoiced /ə/

† Not used in this work.

The question arises as to how diphthongs should be treated. For instance, the onset of the diphthong [eɪ] is phonetically similar to [ɛ], while the offset is more similar to [i], a property that we would like to preserve in the generated vowel space. To avoid the issue of having to decide what class to map each part of the diphthongs to, only monophthongs will be used to train the network. Additionally, r-coloured and weak or marginal vowels are not used.

3 Experiments

The experiments were performed using the phonetically annotated TIMIT corpus, consisting of 10 sentences each spoken by speakers of different American English dialects. The TIMIT dataset is divided into a two parts: the training set, consisting of 462 speakers, and the test set, consisting of 168 speakers. The training set was further split into a validation set containing the utterances of 47 of the speakers, and a reduced training set consisting of the remaining 415 speakers. The samples corresponding to monophthongs (see table 1 for an overview of the vowels used for phonetic transcription in TIMIT) were extracted, and the first and last 20% of the samples belonging to each vowel utterance discarded.

The remaining samples were divided into speech frames using overlapping windows. As vowel utterances are typically very short, using only the samples corresponding to vowels results in a significant reduction in the amount of available data, especially if context windows are to be used. In an attempt to combat this, the window size was set to a relatively short 5 ms, with an overlap of 2 ms. The Hamming window was applied to each speech frame, after which the power spectral density was computed using the fast Fourier transform. Finally, filter bank features were extracted from the power spectrum using 40 triangular overlapping filters spaced linearly along the mel scale within the frequency range of 20 Hz to the Nyquist frequency (8000 Hz). All data was normalised to have zero mean and unit variance using the mean and variance estimated from the training set.

The biases were initialised to 0, and the weights were drawn from a Gaussian distribution with a standard deviation of 0.01. The networks were regularised during training using dropout (Srivastava et al. 2014) with a dropout rate of 0.5 for the non-bottleneck hidden layers, and Gaussian noise with a standard deviation of 0.2 was added to the input each iteration. The networks were trained using cross-entropy as the objective function, with the filter bank features with a context of 5 frames on both sides as input, and a batch size of 64. The learning rate was set to 0.01 initially and scaled by a factor of 0.1 every time no improvement of the validation error, measured as the number of correctly classified frames in the validation set, had been observed for 20 epochs, until the learning rate dropped below 10^{-5} at which point the training stopped.

An initial survey was conducted using non-bottlenecked networks. Out of the networks tested, best performance was achieved using a network with two hidden layers of 1000 maxout units each. This architecture was used as a base for the bottlenecked networks. Performance of the networks was evaluated using the phone error rate (PER) on the test set, i.e. the percentage of speech frames that were correctly classified. All bottlenecked networks tested used maxout units for the hidden layers, with the exception of the bottleneck layer for which a variety of activation functions were evaluated.

4 Results

4.1 Classification performance

Table 2 shows the final PER on the test set for different configurations of hidden layers and bottleneck activation functions. Each network in the table is the best one of its configuration, as based on the final validation error. While maxout performed the best in this case, the different activation functions perform comparably. In all cases the networks performed better with at least two wide hidden layers placed before the bottleneck.

Table 2: Phone error rate on the test set of the best network with the given configuration.

Hidden layers	Bottleneck activation	PER (%)
1000-1000	N/A	27.66
1000-2-1000	Linear	29.50
1000-1000-2	Linear	28.30
1000-1000-2-1000	Linear	28.61
1000-2-1000	Maxout	28.86
1000-1000-2	Maxout	28.45
1000-1000-2-1000	Maxout	28.11
1000-2-1000	tanh	29.66
1000-1000-2	tanh	29.56
1000-1000-2-1000	tanh	28.36

4.2 Resulting vowel representation

After training the networks the vowel representation was extracted at the bottleneck layer after exciting the network with the input data. Figure 2 shows what effect the choice of activation function has on the resulting representation. All data was collected from the networks with the hidden layer configuration 1000-1000-2-1000 using the test set as input.

Perhaps unsurprisingly, the linear and maxout bottlenecks result in very similar projections. While there is some overlap, the vowels form clear and relatively stable clusters. On the other hand, as expected, the tanh bottleneck shows a tendency for the vowels to be placed near the bounding box due to the saturation of the activation function for large input values.

In order to compare the bottleneck representation to other forms of 2D vowel visualisation, the test set was additionally processed using formant extraction and t-SNE. For the formant extraction the vowel samples were split up using 20 ms long windows with 10 ms overlap.

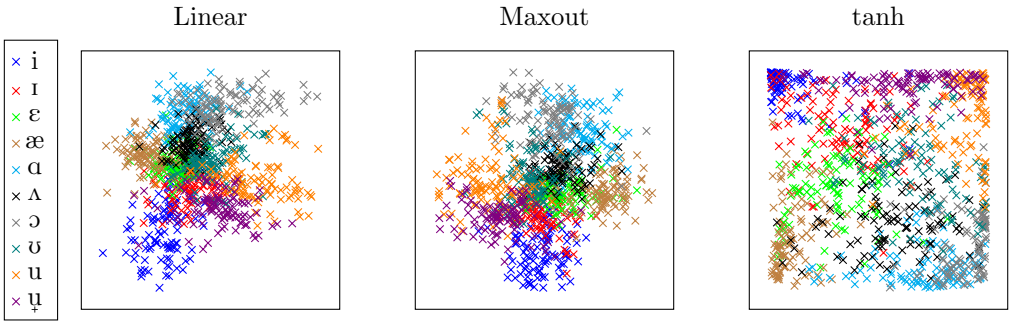

Figure 2: Representation of the vowels in the test set extracted at the bottleneck layer for different activation functions.

Each window was processed using a pre-emphasis filter followed by the Hamming window function. The first two formants were then extracted using naive peak picking as described by Markel (1972), using 20 filter coefficients. t-SNE (Van der Maaten and Hinton 2008) was run on a randomly selected subset of 20,000 speech frames using the implementation provided by Scikit-learn (Pedregosa et al. 2011). The dimensionality reduction was performed on unnormalised filter bank features.

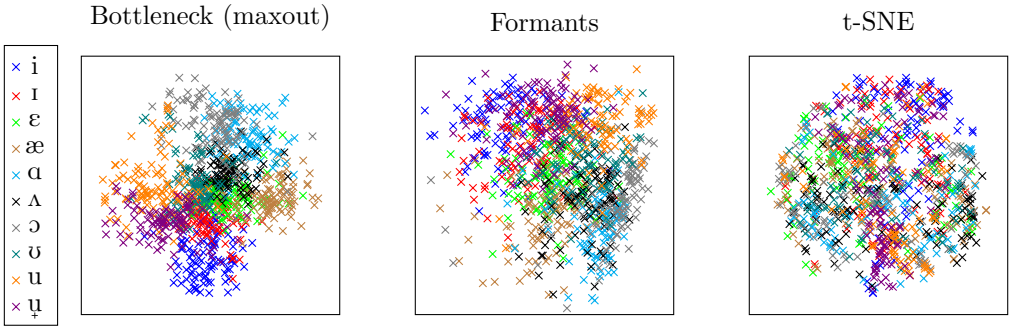

Figure 3: Comparison of different vowel visualisation techinques: Supervised dimensionality reduction using a maxout bottleneck in a neural network, automatic extraction of absolute F1 and F2 values, and unsupervised dimensionality reduction using t-SNE.

As can be seen in fig. 3, the vowel representation in formant space is similar to that of the bottleneck representation, though the vowel clusters suffer from more overlap in formant space. t-SNE seemingly completely fails to find a useful representation of the data, although some tendencies for speech frames belonging to the same vowel class to end up close to each other can be observed.

Figure 4 compares three vowels spoken by male and female speakers of the same dialect. With the exception of the vowel [u], it is evident that the neural network manages to significantly reduce the effect of inter-speaker variation. On the other hand, in formant space there is a clear tendency for female speakers to have a raised F1 and F2, resulting in some separation of vowel clusters corresponding to the same vowel. Finally, while t-SNE does indeed manage to cluster similar vowels together as long as the speakers are similar, it spectacularly fails at any kind of speaker normalisation—though this is not unexpected from an unsupervised dimensionality reduction technique.

Figure 5 models the generated vowel space using 50% confidence intervals of multivariate Gaussian distributions estimated from the bottleneck representation of the test set when fed to the maxout-bottlenecked 1000-1000-2-1000 network. The vowel space bears a striking resemblance to the traditional formant-based vowel space, which is particularly remarkable as the network was not given any prior information regarding vowel similarity.

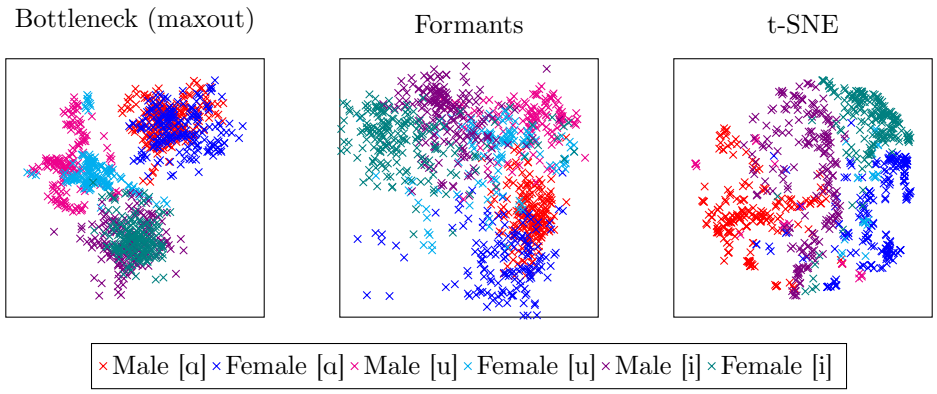

Figure 4: Comparison of male and female speakers from the South Midland dialect region.

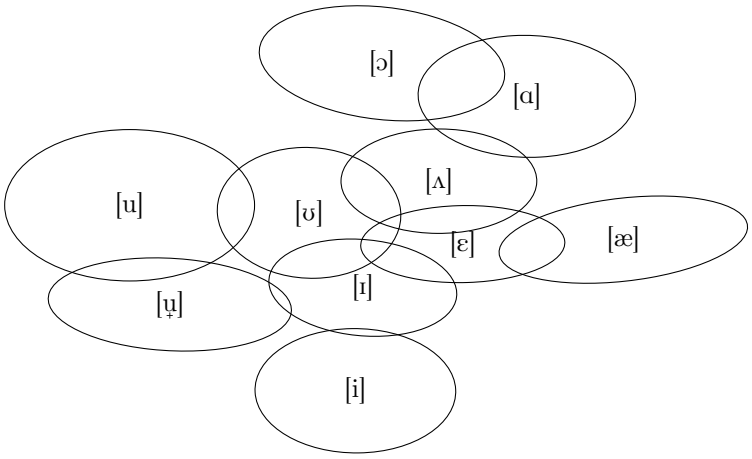

Figure 5: 50% confidence intervals modelling the automatically generated vowel space.

5 Discussion and conclusions

The network managed to find a good two-dimensional vowel representation, that in addition strongly corresponds to the traditional formant-based vowel space. The goal of doing automatic speaker normalisation was also fulfilled to a large extent, although more so for some vowels than others. The failure of the network to normalise specific vowels warrants investigation into whether this is truly a fault of the network, or simply a sociolectal difference in vowel quality.

The actual classification performance of the network was underwhelming, both for the bottlenecked and non-bottlenecked networks, with the phone error rate being well above the around 17–20% PER reported for related architectures for the full TIMIT phone recognition task (Mohamed et al. 2012; Graves et al. 2013; Tóth 2014). Possible explanations for this bad performance include sub-optimal architecture or hyperparameter settings, data sparsity as a result of only using vowel samples, or that vowel recognition might simply be a harder problem than consonant recognition, so that focusing on only vowels inflates the error rate. This latter hypothesis is supported by e.g. the confusion matrix reported by Deng et al. (2013), which shows a systematically higher error rate for vowels than for other phones.

Nevertheless, several alternative architectures are available for consideration. Two types of architectures that have been shown to be promising for speech recognition are convolutional neural networks (Sainath et al. 2013) and recurrent neural networks (Graves et al. 2013). However, for real-time vowel visualisation the information of interest is the pronunciation at the current time instance, with the context being of less use, possibly making RNNs ill-suited for the task. On the other hand, CNNs are thought to reduce the effect of speaker

variation (Abdel-Hamid et al. 2014), making them a prime candidate for use in the task at hand.

Other aspects to tune include the weight initialisation strategy, learning rate scheme, and regularisation techniques. Additionally, training using a more appropriate dataset than TIMIT may increase performance, robustness to noise, and stability of the vowel representation, and in particular, using datasets that include vowels from several different languages may increase the vowel space coverage. It is also possible that other input features may lead to better performance, such as linear predictive coding coefficients, mel-frequency cepstral coefficients, or features that include derivative values.

Finally, more investigation into the properties of the generated vowel space is needed. For instance, if limited to a single speaker, is there a linear correspondence between the bottleneck representation and the formant values of vowels? Does the speaker normalisation performed by the network correspond to any known formant normalisation scheme? A more robust formant extraction technique is also required to facilitate more reliable comparisons.

References

Abdel-Hamid, O., A.-r. Mohamed, H. Jiang et al. (2014). 'Convolutional neural networks for speech recognition'. In: *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 22.10, pp. 1533–1545.

De Jong, G., K. McDougall, T. Hudson et al. (2007). 'The speaker discriminating power of sounds undergoing historical change: A formant-based study'. In: *Proceedings of the 16th International Congress of Phonetic Sciences*, pp. 1813–1816.

Deng, L., O. Abdel-Hamid and D. Yu (2013). 'A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion'. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 6669–6673.

Dowd, A., J. Smith and J. Wolfe (1998). 'Learning to pronounce vowel sounds in a foreign language using acoustic measurements of the vocal tract as feedback in real time'. In: *Language and Speech* 41.1, pp. 1–20.

Fahey, R. P., R. L. Diehl and H. Traunmüller (1996). 'Perception of back vowels: Effects of varying F1- F0 Bark distance'. In: *The Journal of the Acoustical Society of America* 99.4, pp. 2350–2357.

Goodfellow, I. J., D. Warde-Farley, M. Mirza et al. (2013). 'Maxout networks'. In: *arXiv preprint arXiv:1302.4389*.

Graves, A., A.-r. Mohamed and G. Hinton (2013). 'Speech recognition with deep recurrent neural networks'. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 6645–6649.

Grézl, F., M. Karafiát, S. Kontár et al. (2007). 'Probabilistic and bottle-neck features for LVCSR of meetings'. In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE, pp. IV–757.

Ladefoged, P. (1982). *A course in phonetics*. 2nd ed. Harcourt Brace Jovanovich.

Ladefoged, P. (1999). 'American English'. In: *Handbook of the International Phonetic Association* 4144.

Markel, J. D. (1972). 'Digital inverse filtering-a new tool for formant trajectory estimation'. In: *Audio and Electroacoustics, IEEE Transactions on* 20.2, pp. 129–137.

Mohamed, A.-r., G. E. Dahl and G. Hinton (2012). 'Acoustic modeling using deep belief networks'. In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1, pp. 14–22.

Nagayama, I., N. Akamatsu and T. Yoshino (1994). 'Phonetic visualization for Speech Training System by using Neural Network'. In: *Third International Conference on Spoken Language Processing*.

Nearey, T. M. and M. Kiefte (2003). 'A neural network approach to the dimensionality of the perceptual vowel space'. In: *Canadian Acoustics* 31.3, pp. 16–17.

Paganus, A., V.-P. Mikkonen, T. Mäntylä et al. (2006). 'The vowel game: continuous real-time visualization for pronunciation learning with vowel charts'. In: *Advances in Natural Language Processing*. Springer, pp. 696–703.

Pedregosa, F., G. Varoquaux, A. Gramfort et al. (2011). 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12, pp. 2825–2830. URL: `http://scikit-learn.org/`.

Peterson, G. E. and H. L. Barney (1952). 'Control methods used in a study of the vowels'. In: *The Journal of the acoustical society of America* 24.2, pp. 175–184.

Rumelhart, D. E., G. E. Hinton and R. J. Williams (1986). 'Learning representations by back-propagating errors'. In: *Nature* 323, pp. 533–536.

Sainath, T. N., A.-r. Mohamed, B. Kingsbury et al. (2013). 'Deep convolutional neural networks for LVCSR'. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pp. 8614–8618.

Sivaram, G. S. and H. Hermansky (2012). 'Sparse multilayer perceptron for phoneme recognition'. In: *Audio, Speech, and Language Processing, IEEE Transactions on* 20.1, pp. 23–29.

Srivastava, N., G. Hinton, A. Krizhevsky et al. (2014). 'Dropout: A simple way to prevent neural networks from overfitting'. In: *The Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Tóth, L. (2014). 'Convolutional deep maxout networks for phone recognition.' In: *INTERSPEECH*, pp. 1078–1082.

Traunmüller, H. (1981). 'Perceptual dimension of openness in vowels'. In: *The Journal of the Acoustical Society of America* 69.5, pp. 1465–1475.

Van der Maaten, L. and G. Hinton (2008). 'Visualizing data using t-SNE'. In: *Journal of Machine Learning Research* 9.2579-2605, p. 85.

Wik, P. and D. L. Escribano (2009). 'Say 'Aaaaa' interactive vowel practice for second language learning'. In: *Proc. of SLaTE Workshop on Speech and Language Technology in Education*.

Yu, D. and M. L. Seltzer (2011). 'Improved Bottleneck Features Using Pretrained Deep Neural Networks.' In: *INTERSPEECH*. Vol. 237, p. 240.