Convolutional Network For Pronunciation Variants In Speech Recognition

Eric Dagobert
CUNY Graduate Center

Abstract

In most of speech recognition systems, pronunciation is only theoretical and defined from databases containing official words phonetic such as the CMU dictionary. As a result, most system cease to be accurate when the speaker has a strong accent, foreign or local. Here, we try a different approach of phoneme recognition based on physiological structures, the goal being to quantify variations of pronunciation intrinsic to the speaker. To do so, we extract phonetic cues by processing spectrograms as images through a convolution neural network. In this study we used the speech data contained in the TIMIT([13]) corpus.

Convolution network for phoneme classification

Spectrogram to phoneme

A spectrogram is a visual representation of the frequencies and their energy in a sound signal. Some acoustic cues depending on frequency spectrums can therefore be detected in a spectrogram. This is the case of formants (figure 1) for example. A. Liberman [1] and later Gunnar Fant [12]

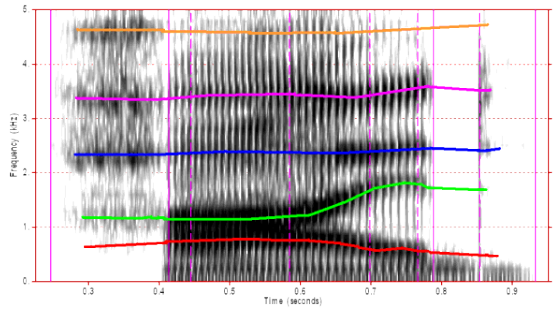


Fig. 1: Formants (source: Macquarie University)

define formants as a set of frequency components that can be associated to an action of the vocal tract. As phonemes are defined from these actions and also categorized in acoustic models such as IPA ([14]), we can therefore assume that a spectrogram contains visual information from which phonemes can be retrieved. So, by processing a spectrogram as an image, we expect a *visual recognition* of the elements of phonetics present in the picture.
In order to perform experiments on mapping spectrograms to phonemes, we have used the TIMIT corpus; the TIMIT database contains about 3000 English phonemes disseminated in 160 utterances spoken by 10 different speakers having different accents. More, TIMIT provides phoneme alignment which allows us to calculate spectrogram for each different phonetic frame. Nevertheless, we did not compute spectrograms from phoneme boundaries only (i.e. context independent), we rather extended phone segments of signal to their immediate neighborhood. Indeed, the first reason is that a phoneme's frequency spectrum is different towards the extremities, depending on the previous and the next phoneme (as shown in [1]). Another thing to consider is that TIMIT database contains some confusion patterns described in [10]: near the boundaries between phonemes there is often some frame overlap with the preceding and the proceeding ones. On the other hand, a context-dependent model may be biased as some phones can be confused with white noise, or show a really small number of visual features compared to their surroundings. Another drawback of a n-phone model is that requires a significant number of training data, which we do not have. For instance the number of possible triphones has a magnitude of $O(n^3)$
So we have defined an hybrid model that calculates a spectrogram from a segment of audio signal extended in both directions, backward and forward, balanced between the gathering of nearby information and limiting confusion. To do so, we defined the following empirical formula:

$$o = \min(s_p, s_c) \times k$$

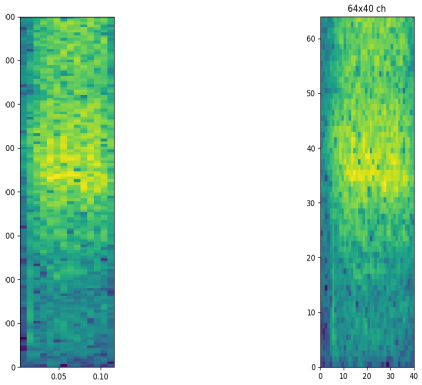
where o is the offset, forward or backward, s_p the size of the previous (or next) frame, s_c the current frame's size and k a coefficient between 0 and 1. Thus we make sure the a frame representation is always bigger than its neighbors while keeping surrounding information.

Spectrogram to fixed size image

A deep learning neural network accepts batches of data grouped into tensors as input, from which the forward propagation consists in a series of tensor dot products. In the case of image processing, usually the input tensor

has 4 dimensions, one being the batch size and three for image dimensions (height, width and pixel depth). Therefore, our first constraint was to produce fixed size spectrograms from variable length signals, each phoneme having its own duration depending on the nature of the phone and the speaker. Image height is simple to fix, as a spectrogram Y axis is scaled depending on the number of FFT used for the convolution. About the width, we processed the signal with both spectrogram parameters and projections: first we adjusted the number of overlaps so the width is close to the target one. Then,

- If the spectrogram was too narrow we applied a pad of -11 dB, which correspond to the lowest energy level.
- Otherwise the spectrogram was transformed to a square via PCA whitening and then, if necessary, a linear projection extracted the desired number of columns.



(a) 'ch' spectrogram **(b)** 'ch' 64x40 image

Fig. 2: Spectrogram fit to size

Figure 2. shows an example of spectrogram transformation to image, (a) is a spectrogram corresponding to a phoneme ('ch') with 256 Fourier Transforms, and (b) is the spectrogram fitting a 64x40 pixels image.

Multiclass phoneme model

TIMIT utterances transcriptions contains the 61 English phones, but, as many people do ([2, 3]), we used a lexicon of 39 phonemes. Also, we wanted a phoneme model with the following properties: ability to compute a distance, and quantify the notion of pronunciation variant, as well as a representation of acoustic features. To do so, we chose a simplified version of the IPA phonetic charts. For instance the vowel charts depicted figure 3 is represented as a three-dimensional feature space in our model.

In reality we have two feature spaces, one for vowels and another one for consonants. Diacritics and other categories have been omitted here for the sake of simplicity. This

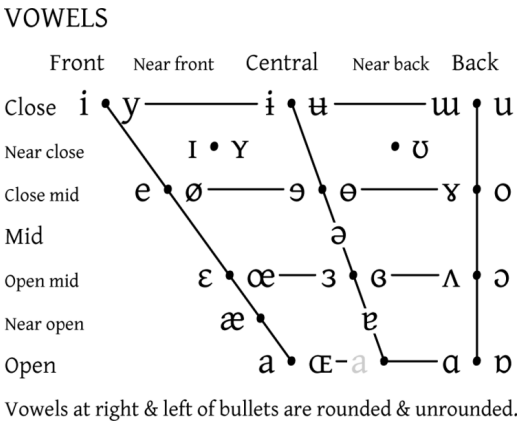


Fig. 3: IPA vowel charts

feature space is organized to have similar phonemes close to each other, but the notion of distance is inspired from IPA charts. The first dimension of this space defines vowel, consonant and silence. Therefore, these three categories see their acoustic features defining distinct sub-spaces, which is what we want as categories of vowel and consonants are not comparable.

Consonants and vowels share the same first feature containing the following values: 'None' is present everywhere and

Feature 1
none
vowel
consonant

Tab. 1: common features

represents a cross-feature value associated with 'silence', 'pause', or background noise. We needed to define such value to represents the zero, for calibration purposes during the neural network's training phase, as well as to set up the white noise level and thus contributes to noise elimination. Another advantage is that the convolutional network, recognizing silences, can be used for automatic phoneme alignment.

Vowel's specific features are defined as displayed in Table 2; all features but feature 5 are mapped to IPA definitions.

We had to define Feature 5 only to take care of particular

Feat. 2	Feat. 3	Feat. 4	Feat. 5
none	none	none	none
front	close	unrounded	default
near-front	near-close	rounded	rhotacized
central	close-mid		near-back2
near-back	mid		near-close2
back	open-mid		
	near-open		
	open		

Tab. 2: Vowels features

cases : 'rhotacized' represents the phoneme 'er' (also called r-colored); near-back2 and near-close2 are a quick way to

define the English double vowels : 'aw','ay', 'ow' and 'oy' (we are aware this is not the best way to include these particular cases).
Consonants features also describe a three-dimensional space, the fourth feature being used for particular cases (which here is 'eng')

Feat. 2	Feat. 3	Feat. 4	Feat. 5
none	none	none	none
bilabial	nasal	voiced	default
labio-dental	plosive	voiceless	diac.syllabic
linguo-labial	sibilant-affricate		
dental	non-sibilant-affricate		
alveolar	sibilant-fricative		
palato-alveolar	non-sibilant-fricative		
retroflex	approximant		
alveolo-palatial	flap		
palatal	trill		
velar	lateral-affricate		
uvular	laterale-fricative		
epiglottal	lateral-approximant		
glottal	lateral-flap		
labio-velar			

Tab. 3: Consonants features

We agree that this feature model is not accurate enough to represent the entire IPA set but it is sufficient for the subset of 39 English phonemes. Most importantly, all of these features are present on a spectrogram.

Convolutional network and image recognition

Convolutional networks ([4]) are very successful for processing 1-D time series and 2-D images, more generally all kind of data that can be organized into a grid, which is our case with the spectrogram. Like a dense ReLU layer [6], a convolutional layer produces piecewise linear functions, but the major difference is the ordering. Indeed, a convolutional layer keeps data order along defined axes (e.g. time in case of a signal).
In the case of an image, convolution can eliminate the noise and modify the gradient in order to reveal the most important features, like an edge detection process could do. In our case we wanted to extract visual features such as formants, but in reality, the back-propagation mechanism adjusts the convolution kernel to keep the most significant patterns. It is interesting to note that a spectrogram is the result of a convolution, and we could have used the raw signal as input, transforming it twice through two convolution layers, a one-dimensional and next a 2-D.
In this experiment we had a **convolution window of size 8 × 8**, an output space of **176 nodes** and a **ReLU activation function**. These parameters have been obtained through trial and error as a compromise between memory and accuracy.
Typically a convolutional network contains also a pooling layer which purpose is to down-sample the result of convolution in order to make it invariant to small translations. In our case we use **Max Pooling** ([7]) to compensate for translations due to pitch or speech speed variations for example.
Next, the product of convolution and pooling is processed by **six dense ReLU layers**: we applied a rule of thumb to determine the number of dense layers, which is to add layers until overfitting and then add **Dropout** layers. The last activation function is the **Softmax** (multi-dimension Logistic) function, particularly adapted to our multi-label classification model. As Softmax works well with maximum-likelihood based loss functions, we chose the **cross-entropy cost function**.
Neural network back-propagation uses Stochastic Gradient Descent algorithm and its variants (Nesterov momentum, etc.), and we had excellent results with a method called **Adadelta** ([8]) which provides, among other properties, the advantage of adapting the learning rate.

Results

We built a neural network (using Python) with the help of Keras [9], a very useful front-end to both Theano[1] and TensorFlow[2], and trained it on 80% of the 160 utterances contained in TIMIT, reserving 20% to the tests; the rotation estimation has been performed at least 10 times. In order to map the neural network output to the subset of 39 phonemes, we have used a decision tree classifier trained on the training sets outputs and targets. Indeed, a predicted vector of features does not always match with an existing phoneme so we needed a mechanism to perform the mapping . If we add the fact that each feature has different probabilities of success, we arrive to something quite complex, then we chose the quick solution of decision tree. Nevertheless, instead of a classifier, we should have used a supplementary dense layer mapping the features to a phoneme, but this would have been more complicated to analyze.

Accuracy

After training, the neural network gives an accuracy of **80%** on test sets, which represents the mean cross-entropy, in other words an average accuracy per feature (in average, 4 out of 5 are correct). Table 4 shows the percentages of correct features predicted from the test set; of course, these percentages also represent probabilities of correct prediction. As expected, the average accuracy is comparable to

Feat.1	Feat.2	Feat.3	Feat.4	Feat.5	Average
91.1%	74.8%	76.0%	84.4%	87.6%	87 %

Tab. 4: Accuracy per feature

the cross-entropy. But an average success rate of 4/5th of the features does not imply a high phoneme prediction rate for which a all five features should match. Thus, when we apply predictions to the test sets and count the number of exact matches we have a success rate of **65%**. We then can compare to [3] performed on the same database (Table 5).

[1] `http://www.deeplearning.net/software/theano`
[2] `https://www.tensorflow.org`

RAW	MFCC	CDBN	CDBN+MFCC	**CNN**
39.4%	70.8%	64.4%	80.3%	**65%**

Tab. 5: Phoneme classifications accuracy

[3] uses a supervised classifier (SVM, GDA or KNN) directly with spectrograms (RAW), with MFCC features (MFCC), with a convolutional neural network (CDBN) or with a combination of MFCC and CNN (CDBN+MFCC).
We can note that our method, if not showing some improvement in accuracy, retrieves anyway some information from unsupervised graphic features. Most importantly, our approach seems to succeed to identify phonemes vectors, which is very important to establish phoneme variation.

Phoneme confusion

Among the 35% mismatches, some predictions are relatively close (by our definition f distance) to targets. To estimate the proximity of two phonemes, we compute the l2-norm of their vector difference. Features are numerically encoded using a label encoder, with the following particularity: none, vowel and consonant vectors are in parallel 4-D spaces. Here is some examples of phonemes with their IPA and vector representation:

ix:	central, close, unrounded, vowel	(1,4,7,14,16)
m:	bilabial, consonant, nasal, voiced	(20,21,35,48,50)
eh:	close-mid, front, unrounded, vowel	(1,2,9,14,16)
t:	alveolar, consonant, plosive, voiceless	(20,25,36,49,50)
b:	bilabial, consonant, plosive, voiced	(20,21,36,48,50)
oh:	back, close-mid, rounded, vowel	(1,6,9,15,16)

Such a definition is far from perfect, and there are much better ways to define a phonetic distance, such as the one described in [11]. For example, the biggest drawback here is that the scale is the same in every direction, which is rather unrealistic: for instance, in the case of consonants, a difference of 1 on the feature 4 (voice/voiceless) is clearly not as important as a difference of 1 in feature 2 (manner of articulation). But some examples will show that such a distance is somewhat realistic:

dist(t,b):	4.12
dist(t,m):	4.24
dist(ix,eh):	2.82
dist(ix,oh):	3.0

(We can verify on the vowel chart that 'eh' is closer to 'ix' than 'oh', and 't' sounds closer to 'b' than 'm').
The causes for confusion have been extensively analyzed in [10], which describes two confusion patterns intrinsic to TIMIT database: the first one is pronunciation/white noise. The second pattern evoked is the frame confusion mentioned earlier in this paper.
We have compared these expected confusions with the closest phonemes predicted from our model and summarized the results in Table 6: For each target phone we show the closest amongst the list of mismatches as well as their distance to target. Phonemes in bold are those contained in the confusion table described in [10]. Interestingly,

Phone	Confusion	Distance
ix	**ih**	1.41
n	m	4
s	**z , sh**	1.0
r	**er,**	22.6
l	n	11.0
ih	**ix**	1.41
k	t	5.0
t	**d**	1.0
ae	**eh**	4.0
q	hh	4.0
m	**n**	4.0
z	**s**	1.0
w	**l**	10.3
d	**t**	1.0
aa	**ah**	2.0
eh	**ix**	2.82
dh	**d**	4.12
ay	**aa**	5.0
ow	**ah**	3.0
ey	ix,**eh**	3.6
ux	**ix**	1.00
ah	**aa**	2.00
f	**v**	1.00
er	**ah**	2.23
hh	k	5.0
sh	**s**	1.0
b	**p**	1.0
p	**b**	1.0
v	**f**	1.0
y	hh	4.24
g	**k**	1.0
jh	**ch**	1.0
oy	aa	5.1
aw	**ae**	2.0
th	**dh**	1.0
uh	**ix**	1.7
eng	**n**	5.1

Tab. 6: Phoneme confusion: closest predicted phonemes. Bold font represents an expected confusion as described in [10]

most of the phonemes in Table 6 are part of the set of expected confusion, and that would mean two things: for one, the convolutional network errors are comparable to other systems, and secondly our acoustic model is coherent as the variants elements defined in terms of distance can be explained with pronunciation or frame confusion. That would mean an altered signal (due to background noise for example), or a pronunciation variant could lead to the prediction of a phone acoustically very close to the expectation.

Conclusion

We wanted to define a system that could handle phoneme variations by using the properties of spectrograms to identify and quantify the acoustic features defining phones. In that sense, results are encouraging, even if our model lacks

perfection. Despite the sketchy vector space definition and a lack of accuracy in the convolution part, the system performed at least as well as the average ones. Certainly, improvements are to be made particularly at the following two levels: first, the convolution part; using the audio signal as input to perform two convolution, a 1-D equivalent to spectrogram computing, and a 2-D similar to the one we described here, would help with frame confusion and keep more information thus improve accuracy.Next, concerning the output,a choice exists between building a hidden feature model where the output vector is replaced with network nodes, or keeping the IPA model on condition that a refinement of the notion of distance is done by rescaling the dimensions. We prefer the latter as it allows more control and attenuate the 'black box' effect caused by neural networks.

References

[1] Liberman, A. M. 1957: Acoustical Society of America Journal, 29, 117.

[2] Li Deng, Dong Yu, and Alex Acero: A generative modeling framework for structured hidden speech dynamics. In Proceedings of NIPS Workshop on Advances in Structured Learning for Text and Speech Processing, Whistler, BC, Canada, December 2005.

[3] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. 2009: Unsupervised feature learning for audio classification using convolutional deep belief networks. In Proceedings of the 22nd International Conference on Neural Information Processing Systems (NIPS'09), Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (Eds.). Curran Associates Inc., USA, 1096-1104.

[4] Le Cun,Y.: Generalization and network Design strategies. Technical report CRG-TR-89-4, University of Toronto.

[5] Goodfellow, I. Bengio, Y. Courville A.: Deep Learning, chap.9 . The MIT Press.

[6] [Arora et al.(2016)] Arora, R., Basu, A., Mianjy, P., & Mukherjee, A.: Understanding Deep Neural Networks with Rectified Linear Units 2016.

[7] Zhou, Y. and Chellappa, R.: Computation of optical flow using a neural network, Neural Networks, 1988, IEEE International Conference pp 71-78.

[8] Zeller, M.D. ADADELTA: An Adaptive Learning Rate Method, arXiv:1212.5701

[9] Chollet, François and others: Keras, GitHub `https://github.com/fchollet/keras`

[10] Andrew Lovitt and Joel Pinto and Hynek Hermansky: On confusions in a phoneme recognizer, IDIAP Research, 2007.

[11] Lynette Melnar and Chen Liu. 2006. A combined phonetic-phonological approach to estimating cross-language phoneme similarity in an ASR environment. In Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology and Morphology (SIGPHON '06). Association for Computational Linguistics, Stroudsburg, PA, USA, 1-10.

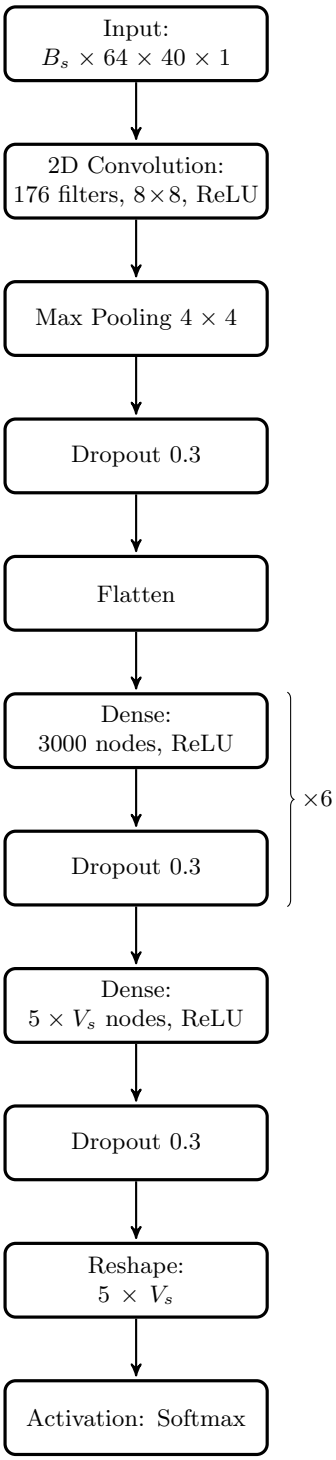
[12] Fant, G. 1960: Acoustic theory of speech production. Mouton, the Hague. 2nd ed., 1970

[13] Garofolo, John, et al.: TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[14] International Phonetic Association: `https://www.internationalphoneticassociation.org/`

Appendix

Neural Network structure



- B_s, the batch size is 1000, for a training set of 4692 picture/phones and a test set of size 1173.
- V_s is the result of labels hot encoding (binary transformation): 53 bits. The neural network output is therefore a vector of 5× 53-bits vectors corresponding to a 5-features vector.
- Final activation **Softmax** is applied to the entire features vector.
- The loss function associated with Softmax is **Categorical Cross-Entropy**, applied on each group of 5 53-bits vectors. Accordingly, the accuracy computed is **Categorical accuracy**, i.e the percentage of correct matches, or correct bits on 53-bits vectors.

Accuracy calculated from test predictions

Accuracy is calculated as the number of correct matches (predicted = expected) divided by the total number of expected for the category. The test set contains 1173 phones, approximately as many consonant as vowels.

Vowel accuracy: **91.4**%:

Feat. 2		Feat. 3		Feat. 4		Feat. 5	
front	77	close	80	unrounded	92	not-rhot	90
near-front	44	near-close	41	rounded	39	rhotacized	52
central	80	close-mid	53			near-back2	35
near-back	0	open-mid	49			near-close2	72
back	67	open	79				
Average	72		65		85		82

Tab. 7: Vowels:Accuracy per category, in percentage

Consonant accuracy: **89.8%**

Feat. 2		Feat. 3		Feat. 4		Feat. 5	
bilabial	48	nasal	85	voiced	77	diac.syll.	54
labio-dent.	65	plosive	84	voiceless	86	non-diac	90
dental	35	sibilant-aff.	63				
alveolar	84	sibilant-fric.	98				
palato-alv.	82	non-sib.-fric.	58				
palatal	40	approximant	62				
velar	70	trill	73				
glottal	57	lateral-app.	60				
labio-velar	76						
Average	72		78		80		89

Tab. 8: Consonants:Accuracy per category, in percentage

N.B. Despite low scores, average accuracy seems quite high. This is because this average is a weighted one, therefore low accuracy categories are the under-represented ones (which can possibly be explained by the fact that rare items are often under-fitted).