Using an autoencoder with deformable templates to discover features for automated speech recognition

Navdeep Jaitly[1], *Geoffrey E. Hinton*[1,2]

[1]Department of Computer Science, University of Toronto, Toronto, M5S 3G4, Canada
[2]Google Inc, Canada

ndjaitly@cs.toronto.edu, hinton@cs.toronto.edu

Abstract

In this paper we show how we can discover non-linear features of frames of spectrograms using a novel autoencoder. The autoencoder uses a neural network encoder that predicts how a set of prototypes called *templates* need to be transformed to reconstruct the data, and a decoder that is a function that performs this operation of transforming prototypes and reconstructing the input. We demonstrate this method on spectrograms from the TIMIT database. The features are used in a Deep Neural Network - Hidden Markov Model (DNN-HMM) hybrid system for automatic speech recognition. On the TIMIT monophone recognition task we were able to achieve gains of 0.5% over Mel log spectra, by augmenting traditional the spectra with the predicted transformation parameters. Further, using the recently discovered 'dropout' training, we were able to achieve a phone error rate (PER) of 17.9% on the dev set and 19.5% on the test set, which, to our knowledge is the best reported number on this task using a hybrid system.

Index Terms: DNN-HMM, feature discovery, autoencoders

1. Introduction

Spectrograms are a very powerful representation for encapsulating information in raw speech signals. However acoustic information is often spread over multiple frequency bins, which makes automated analysis using spectrograms complicated. For example a formant appears as increased energy over several frequency bins and not as a single, resolved peak. Discovering a formant peak is further complicated by the presence of additional peaks, such as the harmonic stack of the fundamental frequency (F0). Further, small changes in acoustics, such as a change in the frequency of a formant peak or F0 can cause a non-linear change in the spectrogram representation as the peaks hop from one frequency bin to the neighbouring bins.

Mel scaled log filter-banks mitigate this problem of 'dimension hopping' by adding rescaled intensity information from multiple neighbouring bins together. Here, the information about the precise location of the formant peak is obviously lost, but robustness for speech recognition is gained because speaker to speaker variability is reduced by the wide triangular windows. Vocal Tract Length Normalization (VTLN), similarly, mitigates this problem by rescaling the frequency axis of spectrograms across speakers - the transformation makes formant peaks more consistent from speaker to speaker [1].

In this paper we use the above motivation to develop a method that discovers non-linear features of spectrograms. We motivate our method with the following simple modification to how a Mel filterbank spectrum is computed. Traditionally, Mel filter-banks have fixed center frequencies and window widths, and windowing functions (example triangular). Imagine a method which customizes how a Mel filterbank spectrum is computed by predicting the center frequency, the window width and windowing type. Such a method has the potential of being more sensitive than Mel filter-banks by integrating information across neighbouring bins in a more precise manner, depending on the data.

Here we use a less constrained approach than that suggested above. We use prototypical patterns or *templates* that are frames of spectrograms. The templates are stretched or compressed along the frequency axis, and are blended together at different intensity levels to compute a reconstruction. The intensity level and the degree of stretching / compression required for each template to make a good overall reconstruction is predicted by using an encoding neural network. The parameters of the neural network and the templates are learnt by minimizing the reconstruction error over training cases.

In machine learning literature, an autonecoder typically uses a neural network for encoding and another (possibly with weights tied to the encoder) for decoding. For this work, we wanted to use a decoder that was constrained to the requirements of the domain - namely the stretching and compressing of the frequency domain, such as is used in VTLN. Thus, we use a novel autoencoder where the encoder is still a neural network, but the decoder is hard-wired to be an algorithm that takes prototypes, transforms them and computes a reconstruction. In addition, the decoder is also able to compute the gradients of the reconstruction error with respect to the prototypical patterns, and predicted transformations. These gradients are backpropagated through the neural network encoder during learning, and neural network parameters, and template parameters are learnt by stochastic gradient descent with momentum.

We applied this method to learn a model for frames of spectrograms from the TIMIT database. We used the features for speech recognition by augmenting traditional Mel log spectra with the predicted intensity parameters. When these features were used in a Deep Neural Network - Hidden Markov Model (DNN-HMM) hybrid setup for the TIMIT monophone recognition task, gains of 0.5% were observed (see table 1). Using the recently discovered 'dropout' training, we were able to achieve a PER of 17.9% on the dev set and 19.5% on the test set.

2. Autoencoder Details

Figure 1 shows an overview of the autoencoder. A neural network (labeled E) takes as input a frame of a log-amplitude spectrogram, $\mathbf{v}$ and predicts T sets of values $\mathbf{z}_t = (f_t, a_t)$, $1 \leq t \leq T$. Each set, (f_t, a_t), specifies the parameters of the transformation (described below) that is applied to the corre-

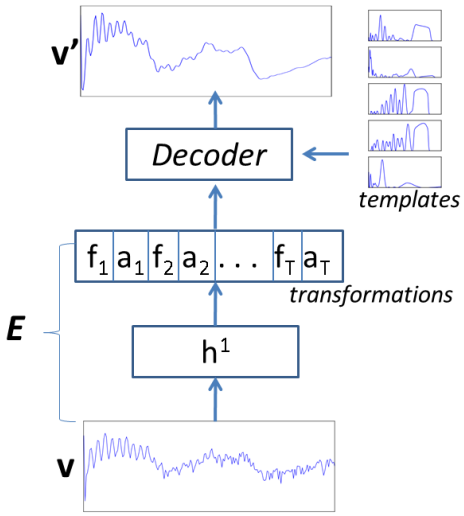

Figure 1: *Overview of the autoencoder. The encoding neural network predicts transformation variables for the template patterns. The decoder applies the transformations to the template patterns and computes the reconstruction, $\mathbf{v}'$, of the input, $\mathbf{v}$, from the transformed templates.*

sponding template frame $\mathbf{s}_t$. The decoder applies these transformations to the templates, and adds the resulting vectors to create the reconstruction of the input frame (equation 1). The model is learnt by minimizing the total reconstruction error over the database (equation 2).

2.1. Model Details

Specifically the autoencoder takes an input data frame $\mathbf{v}$, and computes a reconstruction $\mathbf{v}'$ as follows. The input data frame $\mathbf{v}$ is forward-propagated through the neural network using its parameters $\mathbf{\Theta}$ (weights and biases of the layers), to compute the encoding $\mathbf{h} = [f_1, a_1, f_2, a_2, \cdots f_T, a_T]' = \mathcal{N}(\mathbf{v}, \mathbf{\Theta})$, where $\mathcal{N}$ represents the neural network function. The decoder transforms each template $\mathbf{s}_t$ as follows. It first resamples the template along the frequency axis at a new rate f_t, using linear interpolation. For example, if $\mathbf{s}_t = [s_1, s_2, s_3 \cdots]'$ and $f_t = 1.2$ then $\mathbf{s}'_t = [s1, (0.8s_2 + 0.2s_3), (0.6s_3 + 0.4s_4) \cdots]'$. Let $\mathbf{r}(.,.)$ represent this multivariate transformation function. Note that $\mathbf{r}(.,.)$ retains the property that consecutive peaks of harmonics stacks are equally spaced. The decoder multiplies $\mathbf{s}'_t$ by a_t and adds all the transformed templates together to compute a reconstruction for the input. Mathematically,

$$\mathbf{v}' = \sum_t a_t \mathbf{r}(\mathbf{s}_t, f_t) \tag{1}$$

2.2. Parameter Learning

Learning is done by finding parameters $\mathbf{\Theta}$ for the neural network, and $\hat{\mathbf{s}_1}, \cdots, \hat{\mathbf{s}_T}$ for the T templates that minimize the total reconstruction error, E, over the training data, i.e.:

$$\left(\hat{\mathbf{s}_1}, \cdots, \hat{\mathbf{s}_T}, \hat{\mathbf{\Theta}}\right) = \underset{\mathbf{s}_1, \cdots, \mathbf{s}_T, \mathbf{\Theta}}{\operatorname{argmin}} \sum_{\mathbf{v} \in \mathcal{D}} \|\mathbf{v}' - \mathbf{v}\|^2 \tag{2}$$

The above function is differentiable with respect to all the parameters $\mathbf{s}_1, \cdots \mathbf{s}_T$ as the bilinear interpolation operation is a simple linear operation. Similarly, the gradients of the error with respect to f_t and a_t are also easily computed [1]. Gradients of the reconstruction error with respect to the neural network parameters are computed by back-propagating the gradients with respect to f_t, a_t, through the neural network.

Using all these derivatives, we can perform learning in the model using any of the standard derivative-based optimization schemes. For this paper the parameters of the models were learned by performing gradient descent on the error function. A small learning rate of $1e-5$ per average data case was used. The parameters of each template were constrained to have an Euclidean norm of 1, after each gradient update, by renormalizing its length. This allows intensity values for different templates to be somewhat comparable.

In practice, the error function in equation 2 is modified by adding an L1 penalty to the template intensities a_t, i.e.

$$E = \sum_{\mathbf{v} \in \mathcal{D}} \|\mathbf{v}' - \mathbf{v}\|^2 + \lambda \|\mathbf{a}(\mathbf{v})\| \tag{3}$$

where $\mathbf{a}(\mathbf{v})$ is the vector of intensity instantiation parameters predicted for $\mathbf{v}$, i.e. $\mathbf{a}(\mathbf{v}) = [a_1, a_2, a_3 \cdots]'$, where a_t is the intensity of template t, predicted by the encoder, when data vector $\mathbf{v}$ is presented as input.

This penalty term attempts to make the solution more sparse, and produces more identifiable features.

3. Experimental Methods and Results

We created a Kaldi [2] recipe to train a monophone model with a biphone language model on TIMIT. Spectrograms, log Mel spectra and forced alignment labels for individual frames were exported from this recipe. The spectrograms and log Mel spectra were computed over 25ms intervals with a stride of 10 ms. 40 dimensional log Mel spectra were computed and deltas and accelerations were appended. Spectrograms were 201 dimensional, since the FFT were computed over 400 samples of raw signal. The spectrograms were used for unsupervised learning with the autoencoder. The discovered features were appended to log Mel spectra and used in a Hybrid Deep Neural Network Hidden Markov Model (DNN-HMM) for automated speech recognition.

3.1. Feature Learning

An autoencoder model with templates was trained to reconstruct the spectrograms as described below (also see figure 1).

We used a two layer neural network with 2000 nodes in the hidden layer, and $2T$ nodes in the second layer. The $2T$ nodes in the second layer correspond to T pairs of (f_t, a_t) values, $1 \leq t \leq T$. The f_t values were outputs of sigmoid units whose output range of 0-1 was rescaled to $-\log F, \log F$, (we used $F = \exp(.5)$). The a_t values were outputs of rectified linear units (ReLU) [3, 4]. Each template $\mathbf{s}_t$ frame was resampled at a new rate of $\exp(f_t)$ by taking interpolated values at $(0, \exp(f_t), 2\exp(f_t), ..)$, using bilinear interpolation. Negative values of f_t lead to stretching out of the templates (corresponding to increasing the frequencies of patterns) and positive values lead to compressing of the templates (corresponding to reducing the frequencies of patterns). The stretched or compressed template vector is then multiplied by a_t.

[1] $\mathbf{r}(\mathbf{s}_t, f_t)$ is technically not differentiable w.r.t. f_t at $f_t = 1$ since the left and right derivatives may not be equal. However this is trivial and can easily be avoid by using higher order interpolations e.g., splines. For convenience we just used a value of 0 for this case.

Figure 2 shows a plot of the templates learnt from an experiment using 20 templates. Figure 3 shows reconstructions of a spectrogram using these templates (each frame was separately reconstructed). It can be seen from figure 3 that a very good reconstruction of the spectrogram is possible from just 20 templates. Several templates have identifiable characteristics, e.g. templates 2 and 15 (0 based index) seem to correspond to fricatives, as can been seen from their high intensity during fricative portions of the spectrogram. Template 13 (column 4 from left, row 3 from top) in figure 2 clearly corresponds to a harmonic stack.

3.2. Automatic Speech Recognition

For the speech recognition experiments, we used our features in a hybrid DNN-HMM system [5, 6]. Neural networks were trained to predict the phoneme state labels, and the predictions were converted to scores that reflected the generative probability of the frame from the HMM model. These scores were fed into Kaldi, which performed the decoding. A fixed acoustic model scale of 1.0 was used in the DNN-HMM system. Before the neural networks were trained, we pretrained the weights using a Deep Belief Network (DBN) [7].

Template intensities (but not frequencies) were appended to the 120 log Mel spectra vectors. The log Mel spectrum dimensions were normalized to mean 0 and unit standard deviation. However, the template intensities were outputs of ReLU's, and we didn't think mean centering and normalizing would be an appropriate operation for these inputs. This is because a small template intensity, could possibly end up having a large negative value, for ReLUs. Instead the intensity values were normalized by their standard deviation, but not mean centered.

DBN pretraining was performed by training Restricted Boltzmann Machines (RBMs) for 50 epochs for each layer. The bottom layer was trained as a Gaussian-Binary RBM while the others were trained as Binary-Binary RBMs. Unlike [8] we used only 50 epochs of pretraining - further pretraining did not seem to help the PER after decoding for our system.

A neural network was trained to predict the phoneme state label of a frame using the frame with +/- 7 frames of context. Learning was done using stochastic gradient descent on mini batches of size 100 , with momentum of 0.0 and learning rate of 0.1 over the average gradient per case, for the first epoch. After that a momentum of 0.9 was used. At the end of each epoch the validation set of 400 utterances was decoded with the hybrid system in Kaldi. If the PER increased over the previous epoch, the learning rate was decreased by a factor of half, and the parameters were reset to the values at the start of the epoch. This process was repeated until the learning rate was decreased eight times. The test set was then decoded with the final parameters.

Table 1 shows PER as a function of depth for the dev and core test set in TIMIT, using log Mel spectra and log Mel spectra with template intensities, as the feature inputs to the hybrid system. The dimensionality of log Mel spectra only inputs was 1800 (=15 * 120), while the dimensionality of log Mel spectra plus template intensities was 2100 (=1800 + 15 * 20). Each hidden layer used 2000 nodes. Thus the network with appended features had a slightly larger number of weights (15*20*2000 extra), which is not much considering the total number of weights in these neural networks. It can be seen that the best results on the dev set with log Mel spectra inputs was with 5 layers, resulting in a dev PER of 19.4% and a test PER of 21.1%. For log Mel spectra with template intensities the best dev set result was 19.1% using 7 layers. The corresponding core test set PER was 20.6%. Thus the small number of extra concatenated features seems to help recognition accuracy.

We note here that frequency instantiation parameters did not seem to help speech recognition accuracy. It is possible that we didn't find the right hyper parameters for neural network training. However its equally possible that, the frequency parameters learnt to create a normalized representation over speakers.

Table 2 shows the impact of the regularization parameter, and the number of templates used on the phone recognition accuracy. For these experiments we used 5 layers since that seemed to be a good compromise between speed and accuracy. Using regularization λ larger than 0.1 impacts PER negatively, probably because it is too restrictive. Using more templates does not improve PER either. Manually inspecting the templates learnt we found that when we used more templates, the neural network ignored them by predicting template intensities as small values, effectively creating dead templates. This is probably because a small number of templates is powerful enough to reconstruct the spectrogram (as can be seen from figure 2). It is possible that more templates would be useful for larger databases that have more variable acoustic recordings.

Table 1: *Recognition results using different numbers of hidden layers. It can be seen that concatenating capsule intensities to log Mel spectra improves recognition accuracy.*

Features/depth	3	4	5	6	7
FBanks,Δ,$\Delta\Delta$	19.4/21.7	19.5/21.8	19.4/21.1	19.6/21.3	19.6/21.6
+ a_i	19.4/21.3	19.4/21.0	19.3/20.8	19.2/20.3	19.1/20.6

We wanted to assess whether the gains produced from the capsules were possibly just a result of better optimization from the added features. So we used the recently introduced method of dropout to train another neural network [9]. Dropout is a method that relies on model averaging over a large number of models, and has recently been shown to be a very effective method for avoiding optimization problems and overfitting during learning of parameters in neural networks. Our assumption was that if the modified inputs vectors were indeed more informative for speech recognition, training with dropout would also lead to be better results.

We trained a DBN with 4 layers and 4000 units per layer to conform to the architecture used in [9]. The weights learnt were used to initialize the neural network that was then fine tuned by backpropagation and dropout. A PER of 17.9% and 19.5% was observed for the dev and test sets respectively, compared to the PER of 18.2% and 19.7% that was reported in [9], leading us to believe that the appended features are indeed more informative.

Table 2: *Summary of Speech Recognition Results on TIMIT (5 hidden layers were used in each of the following experiments).*

Features	# of templates	λ	PER (dev/test)
FBanks,Δ,$\Delta\Delta$			19.4 / 21.1
+ a_i	20	0.1	19.3 / 20.8
		0.2	19.5 / 21.0
		0.4	19.8 / 21.2
+ a_i	20	0.1	19.3 / 20.8
	40		19.5 / 20.8
	60		19.3 / 21.0

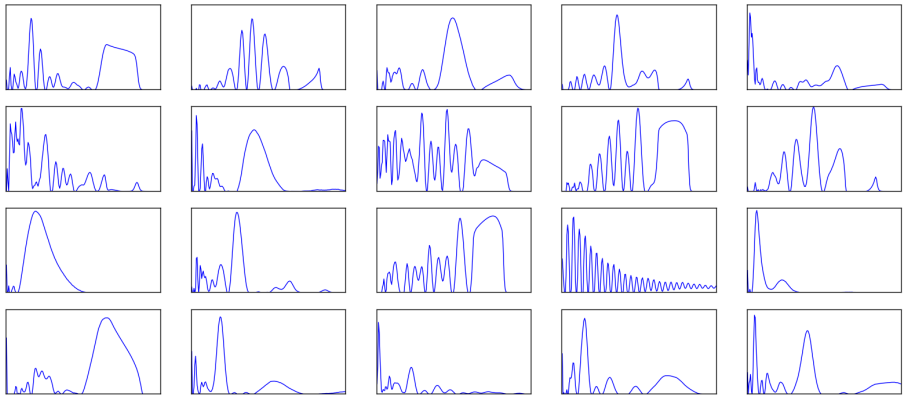

Figure 2: *Template features discovered by our method. x-axis is frequency bin (0-200) and y-axis is intensity.*

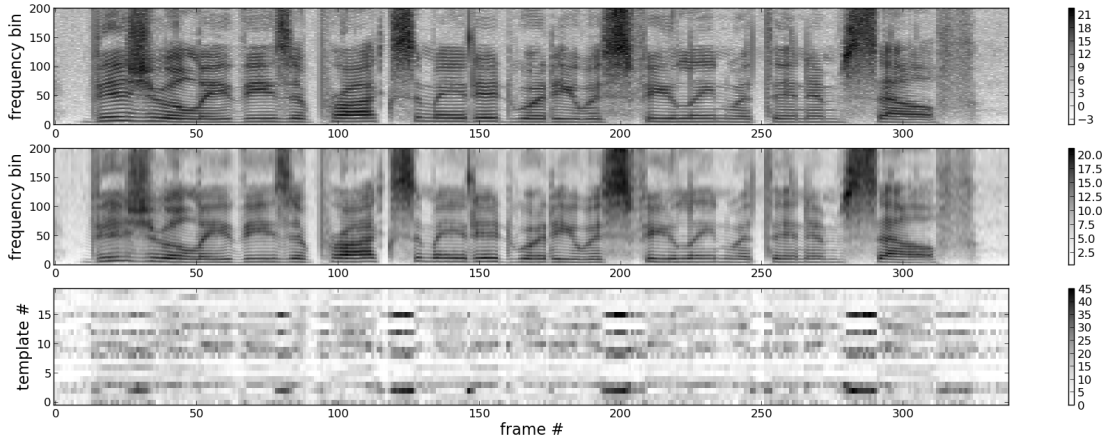

Figure 3: *Reconstructions of spectrogram from our method. Top figure shows original spectrogram, middle figure shows reconstructed spectrogram and bottom figure shows intensities for the 20 templates.*

4. Conclusions

We have shown how a new type of autoencoder can be used to learn meaningful features called templates, on single frames of spectrograms. These templates are where shown to improve speech recognition results when they were concatenated to Mel-log spectra, leading to what we beleive are the best reported results with a Hybrid-DNN-HMM system. In the future we want to apply these autoencoders to patches of spectrograms rather than individual frames. Using patches over multiple frames will allow us to capture temporal aspects that the current system ignores, and possibly lead to further improvements in accuraccy.

5. References

[1] L. Lee and R. Rose, "A frequency warping approach to speaker normalization," *Speech and Audio Processing, IEEE Transactions on*, vol. 6, no. 1, pp. 49–60, 1998.

[2] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, Dec. 2011.

[3] V. Nair and G. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th International Conference on Machine Learning*, 2010.

[4] N. Jaitly and G. Hinton, "Learning a better representation of speech soundwaves using restricted boltzmann machines," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011, pp. 5884 –5887.

[5] H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A. rahman Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *Signal Processing Magazine*, 2012.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *Trans. Audio, Speech and Lang. Proc.*, vol. 20, no. 1, pp. 14–22, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1109/TASL.2011.2109382

[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.