Vocal Tract Length Perturbation (VTLP) improves speech recognition

Navdeep Jaitly NDJAITLY@CS.TORONTO.EDU
University of Toronto, 10 King's College Rd., Toronto, ON M5S 3G4 CANADA

Geoffrey E. Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 10 King's College Rd., Toronto, ON M5S 3G4 CANADA

Abstract

Augmenting datasets by transforming inputs in a way that does not change the label is a crucial ingredient of the state of the art methods for object recognition using neural networks. However this approach has (to our knowledge) not been exploited successfully in speech recognition (with or without neural networks). In this paper we lay the foundation for this approach, and show one way of augmenting speech datasets by transforming spectrograms, using a random linear warping along the frequency dimension. In practice this can be achieved by using warping techniques that are used for vocal tract length normalization (VTLN) - with the difference that a warp factor is generated randomly each time, during training, rather than fitting a single warp factor to each training and test speaker (or utterance). At test time, a prediction is made by averaging the predictions over multiple warp factors. When this technique is applied to TIMIT using Deep Neural Networks (DNN) of different depths, the Phone Error Rate (PER) improved by an average of 0.65% on the test set. For a Convolutional neural network (CNN) with convolutional layer in the bottom, a gain of 1.0% was observed. These improvements were achieved without increasing the number of training epochs, and suggest that data transformations should be an important component of training neural networks for speech, especially for data limited projects.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

1. Introduction

Data augmentation is a key ingredient of the state of the art systems for object recognition (LeCun et al., 1998; Krizhevsky et al., 2012; Simard et al., 2003; Ciresan et al., 2011). In these systems the data vectors are transformed in such a way as to preserve labels. Typically for vision systems, such transformations are easy to conceive of - translating, rescaling and distorting the image locally, etc, are transformations that do not affect the class label of the object in the image, but are yet able to create new related data examples. Data augmentation is very useful for small datasets where the number of examples is low, such as MNIST. However, it has also been helpful in large datasets such as ImageNET. With the widespread adoption of neural networks in speech recognition systems it is natural to ask if the same strategy could be applied here. In this paper we show that it is indeed possible to augment speech databases, and to use the augmented database to achieve improved accuraccy.

Voice Conversion is a natural candidate for generating transformations of utterances, to augment a database. Each utterance could be transformed to multiple speakers, generating new utterances. However, typical voice conversion algorithms require several utterances per speaker. In addition, the number of target speakers is limited to a few chosen speakers for whom good models can be developed. As such applying voice conversion to generate an augmented database is not a very attractive option.

An alternative method to generating variations in input data is to apply speaker normalization methods in the reverse manner - i.e. instead of using normalization methods to remove speaker to speaker variations, we can use them to *add* variations of input data to the training set. This can be done by normalizing input data to random targets, instead of to a cannonical mean. In this paper, we show that Vocal Tract Normalization (VTLN) techniques can be used successfully for this.

VTLN is used in speech recognition to remove speaker to speaker variability that result primarily from differences in vocal tract length(Lee & Rose, 1998). To achieve vocal tract normalization, the frequency axis of the spectrogram of each speaker is linearly warped using a warp factor α, that accounts for the relative length of their vocal tract compared to the cannonical mean. This factor is fit during training time for each speaker in the training database. For speakers in the testing database, different strategies may be applied to fit the warp factor during decoding (Lee & Rose, 1998).

In this paper we propose to augment the training database by randomly generating a random warp factor for each utterance during training. We call this VTLP (for Perturbation). We show that this strategy leads to significant, consistent gains. For TIMIT this strategy produced consistent gains of 0.1% to 1.1% (with an average of 0.65%) on the test set, when using DNNs of different depth. The strategy also produced a gain of 1.0% using a baseline CNN. Data augmentation through transformation, therefore, represents an important direction to be exploited, especially for projects, such as Babel `http://www.iarpa.gov/Programs/ia/Babel/babel.html`, where the amount of transcribed training data is limited.

2. Methods

Here we first briefly summarize how Mel filter bank features are computed traditionally. Then we summarize how we generate transformed utterances using VTLP and Mel filter banks. Lastly, we discuss how the neural network is trained, and used during decoding.

2.1. Mel filter banks

Mel filter banks are filter banks where the spacing between consecutive triangular filters in the frequency domain is motivated by psycho-acoustic considerations (Davis & Mermelstein, 1980). In this scheme, a set of N filter banks are centered at, equi-distant points on the Mel scale. The Mel-scale for a frequency, f is defined as: $m(f) = 1127.01 * \log\left(1 + \frac{f}{700}\right)$. Thus for N filter banks, between a frequency range of F_{min} and F_{max}[1], the center frequency of the i^{th} filter bank is:

[1]We used $F_{min} = 0$Hz and $F_{max} = 8000$ Hz although, $F_{min} = 300$ may be more typical.

$$f(i) = m^{-1}\left(m\left(F_{min}\right) + \frac{m\left(F_{max}\right) - m\left(F_{min}\right)}{N-1}(i-1)\right) \tag{1}$$

where $m^{-1}(s) = 700\left(\exp\left(\frac{s}{1127.01} - 1\right)\right)$ is the inverse of the Mel scale function. Each filter bank starts from the center of the previous filter bank and ends at the center of the next filter bank. We used a triangular window function for each filter bank, with the max value of 1 at the center and 0 at the boundaries.

2.2. Augmenting database with random VTLP

For VTLP, we generate a random warp factor α for each utterance, and warp the freqency axis, such that a frequency f is mapped to a new frequency f' using an approach similar to that outlined in (Lee & Rose, 1998):

$$f' = \begin{cases} f\alpha & f \leq F_{hi}\frac{\min(\alpha,1)}{\alpha} \\ S/2 - \frac{S/2 - F_{hi}\min(\alpha,1)}{S/2 - F_{hi}\frac{\min(\alpha,1)}{\alpha}}(S/2 - f) & \text{otherwise} \end{cases} \tag{2}$$

where S is the sampling frequency, and F_{hi} is a boundary frequency chosen such that it covers the significant formants. We used $F_{hi} = 4800$.

Traditionally, in VTLN approaches the warp factor is assumed to lie between 0.8 and 1.2. Since our goal was not to normalize, but to corrupt, we chose a smaller range of 0.9 and 1.1. A larger range may create unrealistic distortions by distorting utterances with actual high warps or low warps beyond the boundaries of 0.8 and 1.2.

The warping procedure is such that it can be applied directly to the filter banks themselves rather than distorting the spectrogram first. To do this, we merely remap the center frequencies $f(i)$ of the $1 \leq i \leq N$ filter-banks using equation 2 to new frequencies $f(i)'$, and generate the triangular filter banks at these warped frequencies.

2.3. Neural Network Training

During neural network training, at the start of each epoch, we warped each utterance with a random warp factor α and generated log Mel-filter bank values over 40 coefficients. The α values were generated from a normal distribution centered at 1, with a standard deviation of 0.1. Values outside the 0.9-1.1 range were clipped to the boundaries. Since the warping is a very fast procedure, a random warp factor could be generated for each input vector during testing, and would probably improve accuracy. However, for this paper, for convenience, we use a single random α over all the frames of an utterance.

Deltas and accelaration parameters were also computed and appended to the data. A context of +/-7 frames was used for each data vector, to predict the phoneme state label from forced alignment.

2.4. Decoding

Once a neural network has been trained with randomly transformed inputs, the decoding can be performed in several ways. The traditional method would be to fit a warp factor to each speaker that gives rise to the best log likelihood of the utterances for that speaker from the HMM model. This warp factor would then be used for decoding the utterances from that speaker. The same strategy may be applied on an utterance level. However, we found that if the database was not trained with VTLP this only produced a minor improvement.

Instead we approached the problem from a model averaging perspective. At test time, V variant vectors were generated for each utterance by using equally spaced VTLP warp factors between 0.9 and 1.1. The predicted posterior probabilities of the HMM states were obtained by performing a forward pass through the neural network, for each of these variants. These multiple predictions were combined by the following methods:

- Probabilities were averaged (*Avg*).
- Probabilities were geometrically averaged (*Prod*).
- The maximum probability for each state was calculated (*Max*).

The merged probabilities were then used in the Hybrid system to perform decoding.

3. Experiments and Results

We created a Kaldi (Povey et al., 2011) recipe to train a monophone model with a biphone language model on TIMIT. Spectrograms and forced alignment labels for individual frames were exported from this recipe. The spectrograms were computed over 25 ms intervals with a stride of 10 ms. Spectrograms were 201 dimensional, since the FFT were computed over 400 samples of raw signal. 40 dimensional Mel scaled filter banks were generated. Deltas and accelerations were appended (actually we striped the coefficients so that filter bank values and their deltas and accelerations were in adjacent indices - this makes it easier to code up the CNNs, which have a assumption of pixel locality built into them). Mean and Standard deviations for each bin were computed by generating 5 random variants for each utterance in the database. The data vectors were then normalized so that each bin had a zero mean and a unit standard deviation.

We trained DNNs of different depths and a CNN with a convolutional layer in the bottom, similar in architecture to that described in (Abdel-Hamid et al., 2012).

Learning of neural network parameters was performed using back propagation with stochastic gradient descent and momentum. A learning rate of 0.1 was used in the first epoch (0.03 for the CNN). At the end of each epoch the learning rate was annealed by a factor of 0.5 (0.9 for CNN) if the prediction accuracy over the dev set frames did not increase, and parameters were reset to their values at the start of the epoch [2]. For DNN training, a momentum of 0.0 was used for the first epoch, and 0.9 for the rest. For CNN training a momentum of 0.8 was used in the first epoch and 0.9 after.

3.1. Fully Connected Network

We trained fully connected DNNs with 2 to 7 layers of hidden units between the inputs and outputs. Each layer had 2000 hidden units. These layers were pretrained by using a Deep Belief Network (DBN) that had a Gaussian-Binary layer at the bottom and Binary-Binary layers above (Hinton et al., 2006). The input to the DBN was generated using the same scheme that is used for the neural networks - at the start of each epoch, a random warp factor is applied to each utterance in the database. These are normalized and used as inputs to the DBNs.

3.2. Convolutional Network

The CNN we used had a similar architecture to (Abdel-Hamid et al., 2012). 91 local filters that looked at 8 consecutive frequency bins (and associated deltas and accelerations) over all the 15 temporal frames were applied at 6 consecutive frequency bins. These values were max-pooled together. 20 such different sets of filters were applied starting at each even numbered bin. The max-pooled activities were passed through a Rectified Linear Unit (ReLU) (Nair & Hinton, 2010). Thus there were 1820 (=91*20) units in this layer. There were two additional layers of fully connected hidden units of 1000 ReLUs, each before the output layer. During back propagation through the convolutional layers, the gradient is only computed for those

[2]Note that this is different from (Abdel-Hamid et al., 2012; Mohamed et al., 2011) where annealing is done based on the PER from decoding the dev set

Table 1. Phone Error Rate (PER) on the test set for the different experiments. Each configuration was run with and without random transformations of utterances. During decoding, the VTLN warp factor, α, was set to 1 when no combination was performed. For decoding while averaging over multiple α's, 5 variants of each data point were generated at equally spaced values between (a) 0.9 - 1.1, and (b) 0.95-1.05.

	Unaugmented Training							Random VTLN Training						
Arhictecture / α	Avg			Prod		Max		Avg			Prod		Max	
	1	a	b	a	b	a	b	1	a	b	a	b	a	b
2 layer	22.2	22.3	22.5	22.5	22.5	22.3	22.3	22.1	21.9	21.8	22.0	21.9	21.9	22.0
3 layer	21.9	21.9	22.0	22.3	22.4	22.0	21.9	21.5	21.4	21.1	21.4	21.1	21.3	21.3
4 layer	21.6	22.0	21.7	21.9	21.9	22.2	21.9	20.9	20.4	20.5	20.6	20.6	20.4	20.6
5 layer	21.4	22.1	21.8	22.1	22.0	22.0	22.0	21.3	21.4	21.3	21.2	21.3	21.5	21.3
6 layer	21.0	21.6	21.3	21.6	21.6	21.7	21.3	20.9	20.3	20.5	20.2	20.7	20.6	20.2
7 layer	21.6	21.6	21.6	21.6	21.7	21.6	21.6	20.9	20.9	20.6	20.9	20.9	20.8	20.8
Convolutional	21.7	21.7	21.7	21.6	21.9	21.6	21.6	21.1	20.7	20.7	20.5	20.6	20.8	20.8

units that were chosen as the max units in the pooling.

4. Results and Discussion

For each network architecture, we ran two different training runs - one with the original database, and the other in which random warps were generated at the start of each epoch (see table 1). For each of these runs we decoded the test data [3] with and without combining predictions from multiple warp factors, α. The first run, was done with $\alpha = 1$ (columns 2,8). For each of the three methods of decoding, *Avg,Prod,Max* mentioned in section 2.4, we used two different ranges of the warp factor: (a) 0.9 - 1.1 and (b) 0.95 - 1.05. We averaged over 5 equally spaced values of α on this range.

The results for the baseline, representing the unaugmented training for a DNN, or a CNN, with a decoding that uses no VTLP are in the second column of table 1. This can be compared to column 8, which are results for the same decoding, but where the database was generated by random perturbations of the utterances. An absoluted improvement of PER of 0.1,0.4,0.7,0.1,0.1 and 0.3% was seen for DNNs with 2-7 layers, and 0.6% for the convolutional network. Further gains are achieved by using the decoding methods that combine predictions from transforming the same data using multiple warp factors. For example, comparing column 8 to column 10 shows that by averaging over range b (=0.95-1.05), a further improvement of 0.3,0.4,0.4,0.0,0.4,0.3% absolute is achieved for DNNs with 2-7 layers and 0.4% for CNNs. On average, comparing case (b) with random database, with the baseline (column 2), we see an improvement of 0.65% on DNNs. Improvements larger than 0.5% are seen for each method of decoding, compared to the baseline run.

[3]We do not report dev set results because we think they play a significant role in the training, by guiding the annealing schedule. Significant but somewhat smaller gains, were also seen for most of the dev set runs.

Interestingly, averaging over multiple VTLP warp factors at test time leads to worse results for DNNs trained without random VTLPs (columns 3-7 compared to column 2). For CNNs trained without random VTLPs, however, this degradation is not observed. This observation actually aligns with our expectations - convolutional neural networks are designed to be, at least partially, invariant to small local changes. It is clear that here the convolutional networks are indeed reslient in the face of transformed inputs. At the same time, it is clear that even CNNs can be improved by augmenting the database with these transformations during training.

The improved performance gained by combining multiple predictions for each data point, comes at the cost of multiple forward passes through the neural network. However, only a single round of decoding is required, and no speaker adaptation is performed.

We found that using only a single, optimal, α for each utterance did not improve performance significantly over column 8 of table 1. It is our belief that combining predictions over multiple warp factors, improves results because it allows us to average over different variations in the data, and not because the method finds one single good α for an utterance. Thus we think that the method should be helpful even in LVCSR tasks, because it can average over multiple predictions for the same data.

It must be noted here that the strategy of generating perturbations to augment the training data is very well used in vision. However, it seems to have been ignored in speech. It is possible that this may be due to the difficulty in finding a reasonable transformation function. In fact we have tried with several transformations that seem to have been ineffective in providing improvements - warping log spectrograms and using them as input to neural networks, warping Mel log filter bank spectra along the frequency domain and the time domain, applying small changes to LPC coefficients and reconstructing wav files, etc. It is our opinion that perturbation is most effective when applied to the linear spectrograms before the application of filter banks. In addition, perturbation of the input by itself is not as effective without averaging the predictions over multiple perturbations of the data. The two strategies need to be combined for best results.

It should also be noted that our strategy of generating perturbations only increases the number of examples for phoneme contexts that are in the training transcripts. In low resource settings, it would be useful if it were also possible to manufacture examples of different phoneme contexts. However, this method cannot do this, since it is designed to only create examples of the same phoneme class as the seed utterances.

5. Conclusions and Future Work

In this paper we have shown that generating random, linearly warped variants of spectral data can be used to enhance recognition accuracy significantly. We have shown how improved decoding results can be achieved by averaging over multiple predictions over the same data. However, we have only scratched the surface of the variations that can be applied. Naturally, temporal distortions can be applied to spectrograms. Other distortions, such as non linear distortions can also be applied along the frequency dimension (probably most effectively, if applied before multiplication with filter banks). It is our contention that the strategy should also be useful for large datasets, and so in the future we would like to apply it to larger databases such as Wall Street Journal and Switchboard.

References

Abdel-Hamid, O., Mohamed, A., Jiang, H., and Penn, G. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pp. 4277–4280, 2012.

Ciresan, Dan C., Meier, Ueli, Masci, Jonathan, Gambardella, Luca Maria, and Schmidhuber, Jürgen. High-performance neural networks for visual object classification. *CoRR*, abs/1102.0183, 2011.

Davis, S. and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, August 1980.

Hinton, G. E., Osindero, S., and Teh, Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *NIPS*, pp. 1106–1114, 2012.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

Lee, Li and Rose, R. A frequency warping approach to speaker normalization. *Speech and Audio Processing, IEEE Transactions on*, 6(1):49–60, 1998.

Mohamed, A., Dahl, G.E., and Hinton, G.E. Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech, and Language Processing*, 99:1–10, January 2011.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, 2010.

Povey, Daniel, Ghoshal, Arnab, Boulianne, Gilles, Burget, Lukas, Glembek, Ondrej, Goel, Nagendra, Hannemann, Mirko, Motlicek, Petr, Qian, Yanmin, Schwarz, Petr, Silovsky, Jan, Stemmer, Georg, and Vesely, Karel. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, December 2011.

Simard, P.Y., Steinkraus, D., and Platt, J.C. Best practices for convolutional neural networks applied to visual document analysis. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pp. 958–963, 2003.