A Deep Neural Network for Acoustic-Articulatory Speech Inversion

Benigno Uria
Institute for Adaptive and Neural Computation
University of Edinburgh, UK
b.uria@ed.ac.uk

Steve Renals
Centre for Speech Technology Research
University of Edinburgh, UK
s.renals@ed.ac.uk

Korin Richmond
Centre for Speech Technology Research
University of Edinburgh, UK
korin@cstr.ed.ac.uk

Abstract

In this work, we implement a deep belief network for the acoustic-articulatory inversion mapping problem. We find that adding up to 3 hidden-layers improves inversion accuracy. We also show that this improvement is due to the higher expressive capability of a deep model and not a consequence of adding more adjustable parameters. Additionally, we show unsupervised pretraining of the system improves its performance in all cases, even for a 1 hidden-layer model. Our implementation obtained an average root mean square error of 0.95 mm on the MNGU0 test dataset, beating all previously published results.

1 Introduction

The acoustic-articulatory inversion mapping problem (or simply articulatory inversion) is that of trying to infer the position of the vocal tract articulators from an acoustic speech signal. This nonlinear regression problem is ill-posed: several articulator positions can generate the same sound.

Systems capable of approximating the position of the articulators from the acoustic signal are useful in several domains: Speech recognition, where articulatory information can improve the performance of the recognition systems [1]; speech synthesis, where it can be used to improve the quality or to modify the characteristics of the synthesised voice [2]; character animation, where it can be used to automate the facial animation of virtual characters in films and video-games [3].

Several mathematical techniques have been applied to the articulatory inversion problem [4, 5, 6]. The introduction of rich articulography datasets of precise quantitative articulatory position data along with recordings of the acoustic data produced, has made it possible to use standard machine learning methodologies like artificial neural networks [7] or hidden Markov models [8]. Deep architectures have recently been used to obtain state-of-the-art accuracies in phone classification [9, 10, 11, 12] (i.e. in classifying speech acoustic signals into phones). Motivated by their success in phone recognition, we hypothesised a deep belief network would be able to obtain high accuracy in articulatory inversion.

In this work we have implemented a deep belief network for articulatory inversion and evaluated its performance using the MNGU0 [13] test dataset. We obtained a root mean squared error (RMSE) of 0.95 mm, a significant improvement with respect to the best previously published results of 0.99 mm obtained by Richmond [14] using a trajectory mixture density network. This favourable result leads us to conclude that deep architectures are a suitable approach to articulatory inversion. Furthermore, given that our implementation uses a very simple dynamical model, we hypothesise an even higher degree of accuracy could be achieved using a more elaborate trajectory model.

2 Restricted Boltzmann machines and deep belief networks

Restricted Boltzmann machines (RBMs) [15] are undirected graphical models formed by two layers of probabilistic binary units: a visible layer $\mathbf{v}$, and a hidden layer $\mathbf{h}$. All units are fully connected to the units in the other layer and no connections between units of the same layer are present.

Restricted Boltzmann machines are energy-based models. They capture dependencies between variables by assigning an energy value to each configuration, with more probable configurations having a lower energy. The energy for a given configuration is defined to be:

$$E(\mathbf{v},\mathbf{h}) = \mathbf{v}^T W \mathbf{h} + \mathbf{b}^T \mathbf{v} + \mathbf{c}^T \mathbf{h}, \tag{1}$$

where W is the matrix of connection weights, $\mathbf{b}$ is the vector of visible layer biases and $\mathbf{c}$ is the vector of hidden layer biases.

The probability of a given configuration will be $P(\mathbf{v},\mathbf{h}) = \frac{1}{Z} e^{-E(v,h)}$, where $Z = \sum_{v,h} e^{-E(v,h)}$.

Because no connections exist between units of the same layer, the probability of the units in each layer factorise given the state of the other layer, and have the expression:

$$\begin{aligned} P(\mathbf{v}_j = 1 \mid \mathbf{h}) &= \sigma(-\mathbf{b}_j - W_{j\cdot}\mathbf{h}) &(2)\\ P(\mathbf{h}_j = 1 \mid \mathbf{v}) &= \sigma(-\mathbf{c}_j - W_{\cdot j}^T\mathbf{v}), &(3)\end{aligned}$$

where σ is the logistic sigmoid function $\sigma(x) = \frac{1}{(1+e^{-x})}$.

Training an energy based model to capture a probability distribution is done by reducing the energy of configurations present in the training data and increasing the energy of other configurations dictated by the model probability. Maximum likelihood learning in an RBM can be done using the following expressions [15]:

$$\begin{aligned} \Delta W_{ij} &= \langle \mathbf{v}_i \mathbf{h}_j \rangle_0 - \langle \mathbf{v}_i \mathbf{h}_j \rangle_\infty &(4)\\ \Delta \mathbf{b}_i &= \langle \mathbf{v}_i \rangle_0 - \langle \mathbf{v}_i \rangle_\infty &(5)\\ \Delta \mathbf{c}_i &= \langle \mathbf{h}_i^1 \rangle_0 - \langle \mathbf{h}_i^1 \rangle_\infty, &(6)\end{aligned}$$

where$\langle\cdot\rangle_0$ denotes the average when the visible units are clamped to the input values and the hidden units are sampled from their conditional distribution (Equation 3), and $\langle\cdot\rangle_\infty$ denotes the average after assigning the input data to the visible units and running Gibbs sampling until the stationary distribution is reached.

Unfortunately, maximum likelihood learning is too slow to be applied in a practical setting. An approximation often used is contrastive divergence (CD) learning [15, 16] which uses the following update rules for the weights and biases:

$$\begin{aligned} \Delta W_{ij} &\propto \langle \mathbf{v}_i \mathbf{h}_j \rangle_0 - \langle \mathbf{v}_i \mathbf{h}_j \rangle_n &(7)\\ \Delta \mathbf{b}_i &\propto \langle \mathbf{v}_i \rangle_0 - \langle \mathbf{v}_i \rangle_n &(8)\\ \Delta \mathbf{c}_i &\propto \langle \mathbf{h}_i^1 \rangle_0 - \langle \mathbf{h}_i^1 \rangle_n, &(9)\end{aligned}$$

where $\langle\cdot\rangle_n$ denotes the average after assigning the input data to the visible units and performing just n updates (as in Gibbs sampling) to each layer. Whereas in Gibbs sampling the number of updates required to reach the stationary distribution can be very high, in contrastive divergence learning a very low number of updates is used, usually just 1. Contrastive divergence has been empirically shown to work reasonably well [17] in most cases. However, other alternatives exist like persistent contrastive divergence [18], or enhanced gradient descent [19].

2.1 Gaussian-Bernoulli RBM

For problems with real valued input features, having binary visible units is not an appropriate representation of the data. In these cases a Gaussian-Bernoulli RBM [20] can be used. In a Gaussian-Bernoulli visible units follow a Gaussian activation probability and the hidden units are normal binary units that follow a Bernoulli distribution.

The input data to Gaussian-Bernoulli RBMs is usually normalised over the training dataset to have standard deviation 1 for each unit. In which case the energy of a given configuration is:

$$E(\mathbf{v},\mathbf{h}) = \frac{1}{2}(\mathbf{v}-\mathbf{b})^T(\mathbf{v}-\mathbf{b}) - \mathbf{v}^T W \mathbf{h} - \mathbf{c}^T\mathbf{h}, \tag{10}$$

and as in the standard RBM the conditional distribution of $\mathbf{v}$ given $\mathbf{h}$ factorises with expression:

$$\begin{aligned} P(\mathrm{v_j} \mid \mathbf{h}) &\sim \mathcal{N}\left(-\mathbf{b}_j - W_{j\cdot}\mathbf{h}, 1\right) & (11)\\ P(\mathbf{h}_j = 1 \mid \mathbf{v}) &\sim \sigma(-\mathbf{c}_j - W_{\cdot j}^T\mathbf{v}). & (12)\end{aligned}$$

2.2 Deep belief networks

A deep belief network (DBN) [17] is a multi-layer generative probabilistic model, made up of stochastic binary units organised in layers. The bottom layer is the visible layer $\mathbf{v}$, while the rest of the layers are hidden layers $\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^l$.

In a DBN the top two layers ($\mathbf{h}^{l-1}$ and $\mathbf{h}^l$) are connected in an undirected manner and form an RBM. The rest of layers are connected in a top-down directed fashion.

The joint probability of a DBN factorises as follows:

$$P(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,\ldots,\mathbf{h}^l) = P(\mathbf{v} \mid \mathbf{h}^1)P(\mathbf{h}^1 \mid \mathbf{h}^2)\ldots P(\mathbf{h}^{l-2} \mid \mathbf{h}^{l-1})P(\mathbf{h}^{l-1},\mathbf{h}^l). \tag{13}$$

The probability of activation of a unit in any layer below the top two is conditionally independent of the rest of units given the state of the layer directly on top of it, and follows the expression:

$$P(\mathbf{h}_j^i \mid \mathbf{h}^{i+1}) = \sigma(-\mathbf{b}_j^i - W_{j\cdot}^i\mathbf{h}^{i+1}). \tag{14}$$

Therefore, generating samples from a DBN can be achieved by doing Gibbs sampling of the top two layers. Then, doing ancestral sampling of the lower layers using equation (14) until we reach the visible layer.

2.2.1 Training a deep belief network

The key to successfully training a DBN is to do it in a greedy, layer-wise manner [17]; adding layers on top one at a time.

Initially, a DBN will consist of only 2 layers, the visible and the first hidden layer. These initial two layers are connected in an undirected way and trained as an RBM. If a new layer is to be put on top, the connections between the previous top two layers are transformed into directed top-down connections, and a new layer is added on top, connected to the previous top layer in an undirected fashion. These new top two layers are, again, trained as an RBM. To obtain the visible data of this new RBM, we must get samples from the previous top layer by running the data examples through the previous layers of the DBN using equation[1]:

$$P(\mathbf{h}_j^l \mid \mathbf{h}^{l-1}) = \sigma(-\mathbf{c}_j^l - W_{\cdot j}^{l-1^T}\mathbf{h}^{l-1}). \tag{15}$$

2.2.2 Adapting a deep belief network for regression

In order to use a DBN for regression two main possibilities exist [21]: (1) Using a DBN to model the joint probability distribution of the input and outputs. Then clamping the input units and sampling

[1]For notational convenience we will denote $\mathbf{v}$ as $\mathbf{h}^0$ here.
Usually, the mean-field value is used and no sampling of the binary values is done [21].

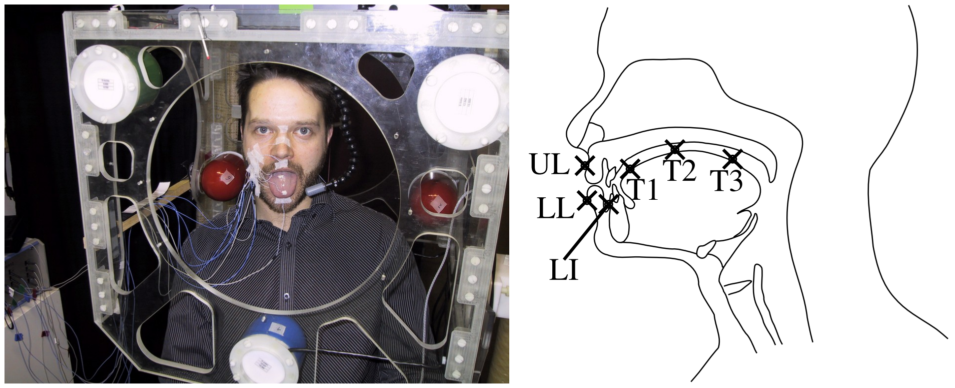

Figure 1: Left: Photograph of an EMA setup taken during an MNGU0 dataset recording session. Right: Positioning of electromagnetic coils in the MNGU0 dataset. The articulators tracked are: upper lip (UL), lower lip (LL), lower incisor (LI), tongue tip (T1), tongue blade (T2), and tongue dorsum (T3).

from the output units. (2) Using a DBN to model the probability distribution of the inputs and using the top hidden layer units as inputs features to a standard regression method.

The second method is most commonly used. The simplest option for a regression problem is to add a linear regression layer on top, and use the DBN weights and biases as the initialization point for the parameters of a multilayer neural network [21].

However, performance using the initial weights from a DBN, is usually not very good. To improve its performance the model is fine-tuned by performing gradient descent; generally, by using the backpropagation algorithm.

3 Electromagnetic midsagittal articulography dataset

Electromagnetic midsagittal articulography (EMA), a technique that uses electromagnetic transducer coils glued to the vocal-tract articulators to record precise measurements of their position [22], is the most widely used articulography technique for the creation of parallel acoustic and articulator-position recordings.

The MNGU0 EMA dataset used in this paper, consists of 1263 utterances recorded from a single speaker in a single session. Parallel recordings of acoustic data and the position of 6 coils is available. Transducer coils were placed in the midsagittal plane at the upper lip, lower lip, lower incisors, tongue tip, tongue blade and tongue dorsum (see Figure 1). Each EMA data frame is made up of 12 coordinates, two (x and y position) for each articulator tracked, with a sampling frequency of 200 Hz. The acoustic data consists of frames of 40 frequency warped line spectral frequencies (LSFs) [23] and a gain value, the frame shift step is 5 ms in order to obtain acoustic features at the same frequency as the EMA data.

In our experiments we will use a context window of 10 acoustic frames selecting only every other frame. Therefore, each input window will span a period of 100 ms. As output, we will use the EMA frame that corresponds to the time at the middle of the current acoustic window, i.e. between acoustic frames 5 and 6.

The dataset is partitioned in three sets: a validation and a testing set comprising 63 utterances each, and a training set consisting of the other 1137 utterances.

4 Benchmarks

To measure the accuracy of our system we will use the root mean-squared error (RMSE), which is the most widely used measure of an articulatory inversion system performance, and is defined as $\sqrt{\frac{1}{N}\sum_i (e_i - t_i)^2}$, where e_i is the estimated tract variable and t_i the actual tract variable at time i.

In order to judge the relevance of our results, we need to set some reference points with which we can compare them using the test set of the MNGU0 dataset.

A linear model that uses as input a window of acoustic data and as output a frame of articulatory data, obtains an average RMSE of 1.52 mm. A regular one-hidden-layer artificial neural network with 300 units, obtained an average RMSE of about 1.13 mm.

To set some reference points for what would be state-of-the-art accuracies, Richmond's trajectory mixture density networks obtain, when using one Gaussian-mixture per articulatory channel, an average RMSE of 1.03 mm. When using 1, 2 or 4 Gaussian-mixtures (the number is optimised for each channel) the average RMSE is reduced to 0.99 mm [14].

5 A deep neural network for articulatory inversion

5.1 Pretraining

To create our articulatory inversion system, first we trained a deep belief network to model speech acoustic data[2]. Given that acoustic data is real valued, we used a Gaussian-Bernoulli restricted Boltzmann machine for the first layer of our DBN. To train it we used the contrastive divergence with 1 sampling step (CD1).

Figure 2 shows the receptive fields learnt by forty hidden units of the Gaussian-Bernoulli RBM. We can observe some of them show blobs of high or low activation (light and dark regions respectively), these blobs capture continuity patterns of the acoustic data in the time and frequency domains. Some other units, display vertical patterns of high activation flanked by low activation (light vertical lines flanked by dark vertical lines) and usually occupy the whole frequency (vertical) spectrum. These features capture activation contrasts between contiguous acoustic frames.

We tried different sizes for the hidden layer of the Gaussian-Bernoulli RBM. We found that the final average reconstruction error decreases as the number of hidden units is increased, but saturates at 300 units and does not improve by adding more hidden units after that.

More layers were trained on top by following the conventional procedure for DBNs: freezing the weights of the layer just trained and using its hidden layer activation probabilities as input for the new layer. In these higher layers both visible and hidden units take binary values and are therefore Bernoulli-Bernoulli RBMs.

5.2 Adaptation for articulatory regression

In order to adapt the deep belief network trained to do regression of the articulator positions, we transformed it into an artificial neural network, by using the DBN generative weights as the initial weights of the ANN, and adding a linear regression layer on top with one output unit per articulatory channel to infer.

To train this ANN, we first trained the top regression layer by using the pseudo-inverse method. Then we performed backpropagation as in a regular artificial neural network to fine-tune the features captured by the hidden layers and the top-layer linear regression weights.

5.3 Results

Using the procedure explained in the two previous sections, we trained a variety of DBNs with different numbers of layers and units per layer using the training set from the MNGU0 dataset.

[2]To be more precise, speech acoustic data produced by the speaker in the MNGU0 dataset.

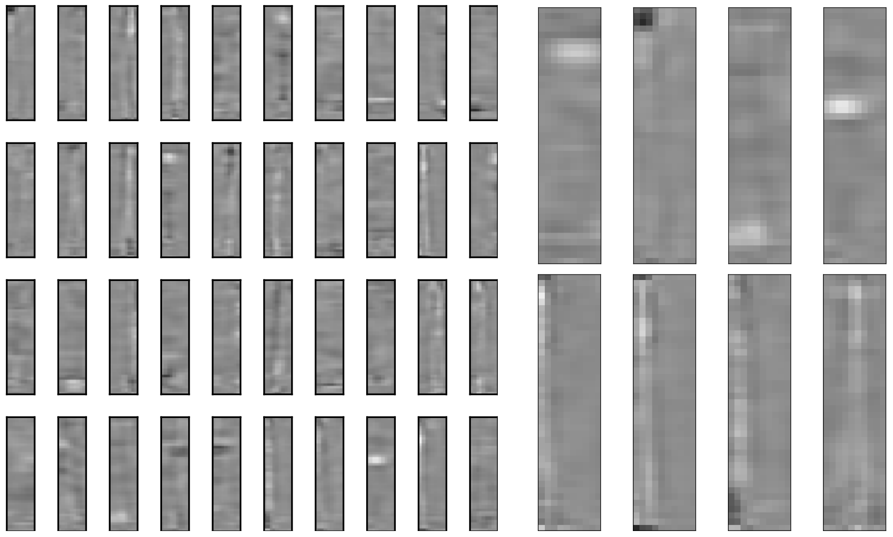

Figure 2: Left: receptive fields of 40 units (in this experiment the hidden layer had a total of 300 hidden units) of a Gaussian-Bernoulli RBM trained as the first layer of our DBN. Each receptive field is displayed by plotting the connection weight to each of the 410 visible units. Each column in a field connects to each of the ten acoustic frames used an input. Each point in a column shows an LSF coefficient (lower frequencies at the top) and the gain appears at the bottom. Top-right: receptive fields of four hidden units that capture blob-like patterns. Bottom-right: receptive fields of four hidden units that capture column-like patterns.

Their performance on the test set of the MNGU0 dataset using the root mean square error (RMSE) criterion is shown in Figure 3 on the next page.

We can observe that the best results are obtained using 3 hidden layers and 300 units per hidden layer, with an average RMS error of 1.026 mm.

Our results suggest that a deep architecture can obtain better results than a shallow 1-hidden layer system. However, it could be argued that it is the use of a greater number of parameters and not the hidden layers that matters. To test that possibility, we also plot the RMSE of each ANN versus its number of tunable parameters. We can observe that a 3 hidden-layers architecture is superior to shallower architectures even when using the same number or even fewer tunable parameters. Therefore, we conclude a deep architecture works better than a shallow one to perform articulatory inversion.

To check whether we could have achieved similar results without the DBN pretraining phase, we trained a set of randomly initialised artificial neural networks. The results are shown in Figure 4. We can observe that in all cases an ANN pretrained as a DBN obtains better results than a randomly initialised ANN. However, in contrast to what is usually posited in the deep architecture literature, we found no trouble in training 2 and 3 hidden-layer ANNs with randomly initialised weights.

5.4 Low-pass filtering

This simple deep architecture performs regression from one window of acoustic data to one frame of articulatory output with no regard for the continuity or dynamical properties of the articulator trajectories. Therefore, it is quite remarkable that it is able to beat a one-Gaussian-mixture trajectory mixture density network, which takes into account the dynamic constraints of the articulatory data by using delta and delta-delta output features to infer the most probable trajectory of the articulators by using maximum-likelihood parameter generation [24].

In order to account for the dynamic nature of articulatory data in a simple manner, we can take the approach followed by Richmond in his early work [7], where in order to eliminate the higher

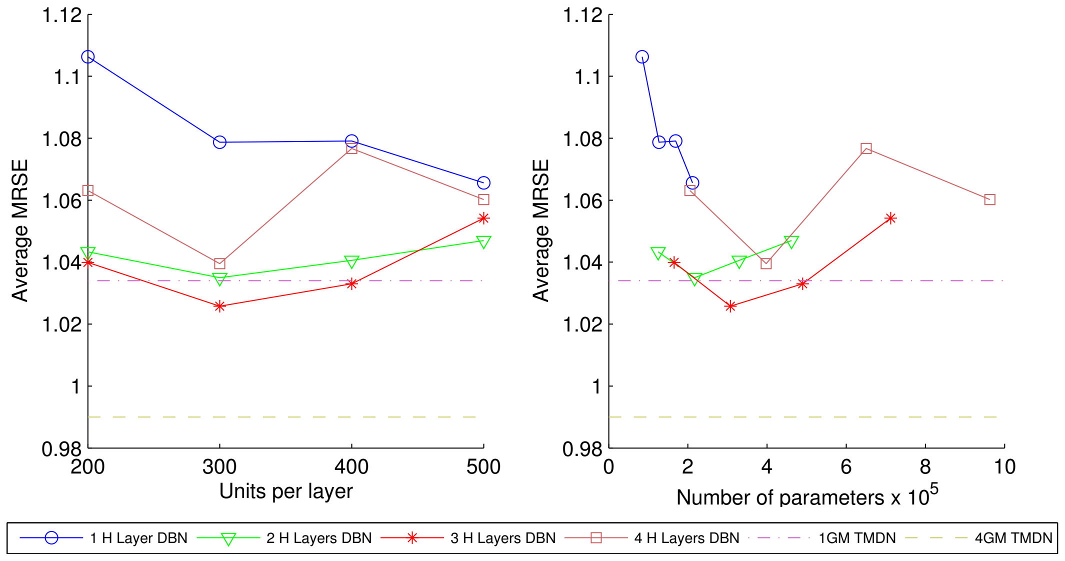

Figure 3: Left: RMSE performance of ANNs pretrained as DBNs on the MNGU0 test dataset as a function of the number of units per layer. Four series are shown, using 1,2,3 and 4 hidden layers respectively. For reference, the accuracies of a 1GM TMDN and 4GM TMDN are also shown. Right: RMSE as a function of the number of tunable parameters. A 3 hidden layer DBN obtained better results than 1 or 2 hidden layer DBNs even when using the same number or fewer tunable parameters.

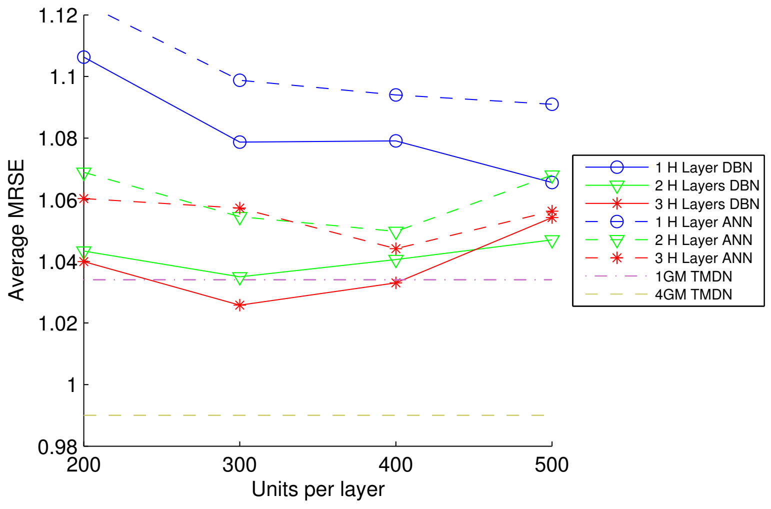

Figure 4: Comparison of RMSE performance on the MNGU0 test dataset between ANNs pretrained as DBNs, and ANNs initialised randomly. Six series are shown, using 1,2 or 3 hidden layers respectively for each kind of ANN. For reference, the accuracies of a 1GM TMDN and 4GM TMDN are also shown. In all cases ANNs pretrained as DBNs beat randomly initialised ANNs.

frequency components of his output, he applied a low-pass filter obtaining as a results trajectory estimations with a lower root mean square error.

Using the same approach as Richmond, we apply a zero-shift second order Butterworth low-pass filter to our output. The movement of different articulators can have different dynamical characteristics. Therefore, we calculated the RMSE on the MNGU0 training dataset using integer cut-off frequencies in the range 1-20 Hz for each channel and chose the optimal for each of them.

The new results of our system after low-pass filtering surpass the best results of Richmond's trajectory mixture density networks, even using a much simpler dynamical model. The best results were obtained, again, using a 3 hidden-layer architecture with 300 units per layer with a record average RMS error of 0.954 mm on the test set of the MNGU0 dataset.

6 Conclusion and future work

In this work we have shown that a deep neural network is capable of obtaining accurate estimates of articulatory trajectories from acoustic data. The best results were obtained using 3 hidden layers of 300 units each, with an RMSE of 0.95 mm on the test set of the MNGU0 dataset. A significant error reduction compared to the previous best results published on the same dataset of 0.99 mm using trajectory mixture density networks.

In our work, we show that adding hidden layers (up to three) to the architecture, improves the accuracy. We also show, this is a result of a higher expressive capability of a deep architecture [25], and not only a result of adding more parameters to the model.

We also show the advantage of doing unsupervised pretraining, where a deep belief network is trained before being transformed into an artificial neural network. In all cases analysed, pretraining had a positive effect on the results.

Even though low-pass filtering improved our results considerably, it would be desirable to implement a more advanced trajectory method, as it is done in the most recent publications by Richmond [26] or Renals [8]. Using the maximum likelihood parameter generation (MLPG) algorithm requires a probability distribution of the position, velocity and acceleration of the articulators. However, our present system is only capable of inferring point estimates of those features. We intend to adapt our present system to act as a "deep" mixture density network. To do this we will have to modify our system to output the mean, covariance matrix and mixing factors of a Gaussian mixture model. These changes only affect the backpropagation phase of the training. Having a Gaussian mixture as the output of our system will also allow us to explicitly tackle the non-uniqueness of the articulatory inversion problem, leaving for the MLPG algorithm to decide the most likely trajectory through that multimodal probability estimation.

Acknowledgments

We would like to thank Iain Murray for his helpful advice during the writing of this paper.

References

[1] J. Sun and L. Deng. An overlapping-feature-based phonological model incorporating linguistic constraints: Applications to speech recognition. *The Journal of the Acoustical Society of America*, 111:1086, 2002.

[2] Z. Ling, K. Richmond, J. Yamagishi, and R. Wang. Integrating articulatory features into HMM-based parametric speech synthesis. *IEEE Transactions on Audio, Speech and Language Processing*, 17(6):1171–1185, 2009.

[3] Gregor Hofer and Korin Richmond. Comparison of HMM and TMDN methods for lip synchronisation. In *Proc. Interspeech*, pages 454–457, Makuhari, Japan, September 2010.

[4] BS Atal, JJ Chang, MV Mathews, and JW Tukey. Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique. *The Journal of the Acoustical Society of America*, 63:1535, 1978.

[5] M.G. Rahim, C.C. Goodyear, W.B. Kleijn, J. Schroeter, and M.M. Sondhi. On the use of neural networks in articulatory speech synthesis. *The Journal of the Acoustical Society of America*, 93:1109, 1993.

[6] S. Dusan and L. Deng. Estimation of articulatory parameters from speech acoustics by kalman filtering. *Proc. of CITO Researcher Retreat-Hamilton Canada*, 1998.

[7] K. Richmond. *Estimating Articulatory Parameters from the Acoustic Speech Signal*. PhD thesis, The Centre for Speech Technology Research, Edinburgh University, 2002.

[8] L. Zhang and S. Renals. Acoustic-articulatory modeling with the trajectory HMM. *IEEE Signal Processing Letters*, 15:245–248, 2008.

[9] A. Mohamed, G. Dahl, and G. Hinton. Deep belief networks for phone recognition. In *Proc. of NIPS 2009 Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

[10] G.E. Dahl, A.M. Marc Aurelio Ranzato, and G. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. *Advances in Neural Information Processing Systems*, 24, 2010.

[11] A. Mohamed, G. Dahl, and G. Hinton. Acoustic modeling using deep belief networks. *Audio, Speech, and Language Processing, IEEE Transactions on*, (99), 2011.

[12] Navdeep Jaitly and Geoffrey E. Hinton. Learning a better representation of speech soundwaves using restricted Boltzmann machines. In *ICASSP*, pages 5884–5887, 2011.

[13] K. Richmond, P. Hoole, and King S. Announcing the electromagnetic articulography (day 1) subset of the mngu0 articulatory corpus. In *Proc. Interspeech*, 2011.

[14] K. Richmond. Preliminary inversion mapping results with a new EMA corpus. In *Proc. Interspeech*, pages 2835–2838, Brighton, UK, September 2009.

[15] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[16] M.A. Carreira-Perpiñán and G.E. Hinton. On contrastive divergence learning. 2005:17, 2005.

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

[18] T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[19] K. Cho, T. Raiko, and A. Ilin. Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In *Proc. ICML*, Bellevue, USA, 2011.

[20] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19, page 153, Vancouver, Canada, December 2007. The MIT Press.

[21] G. Hinton. A practical guide to training restricted Boltzmann machines. Technical report, Department of Computer Science, University of Toronto, 2010.

[22] J.S. Perkell, M.H. Cohen, M.A. Svirsky, M.L. Matthies, I. Garabieta, and M.T.T. Jackson. Electromagnetic midsagittal articulometer systems for transducing speech articulatory movements. *The Journal of the Acoustical Society of America*, 92:3078, 1992.

[23] H.W. Strube. Linear prediction on a warped frequency scale. *The Journal of the Acoustical Society of America*, 68:1071, 1980.

[24] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. ICASSP*, pages 1315–1318, Istanbul, Turkey, June 2000.

[25] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[26] K. Richmond. A trajectory mixture density network for the acoustic-articulatory inversion mapping. In *Proc. Interspeech*, Pittsburgh, USA, September 2006.