Combining a Vector Space Representation of Linguistic Context with a Deep Neural Network for Text-To-Speech Synthesis

Heng Lu, Simon King, Oliver Watts

The Centre for Speech Technology Research, The University of Edinburgh, UK

hlu2@inf.ed.ac.uk, Simon.King@ed.ac.uk, owatts@staffmail.ed.ac.uk

Abstract

Conventional statistical parametric speech synthesis relies on decision trees to cluster together similar contexts, resulting in tied-parameter context-dependent hidden Markov models (HMMs). However, decision tree clustering has a major weakness: it use hard division and subdivides the model space based on one feature at a time, fragmenting the data and failing to exploit interactions between linguistic context features. These linguistic features themselves are also problematic, being noisy and of varied relevance to the acoustics.

We propose to combine our previous work on vector-space representations of linguistic context, which have the added advantage of working directly from textual input, and Deep Neural Networks (DNNs), which can directly accept such continuous representations as input. The outputs of the network are probability distributions over speech features. Maximum Likelihood Parameter Generation is then used to create parameter trajectories, which in turn drive a vocoder to generate the waveform.

Various configurations of the system are compared, using both conventional and vector space context representations and with the DNN making speech parameter predictions at two different temporal resolutions: frames, or states. Both objective and subjective results are presented.

Index Terms: TTS, speech synthesis, deep neural network, vector space model, unsupervised learning

1. Introduction

Traditionally, text-to-speech (TTS) conversion systems use a carefully-constructed *linguistic specification* as the interface between the text and the waveform. These representations might be created in a number of ways, from the hand-coded rules of a formant synthesizer [1, 2] to the complex multi-layered structures typically found in systems such as Festival [3]. The latter are built up in a number of steps, using numerous models (and a few rules). Each of these models is typically trained in a supervised fashion from labelled data (e.g, speech with hand-labelled phrase breaks). This makes such systems expensive to port to a new language and hard to adapt to a specific application domain.

From the linguistic specification, a waveform is generated. This might be done through the concatenation of short segments of speech, as in the unit selection method, or through a trainable statistical parametric model. The HMM-based statistical parametric speech synthesis method [4, **?**, 5, 6] is the most widely used statistical parametric approach, not least because it has advantages over unit selection such as smaller footprint, stable output, and more flexibility (e.g., speaker adaptation). However, the way in which the linguistic specification is mapped by decision trees onto a set of disjoint context-dependent models may not make the most effective use of training data.

1.1. Representing and modelling linguistic context

In speech synthesis, context-dependent models are necessary to capture contextual effects in speech, including co-articulation and supra-segmental variation. Wide context modelling is much more important than in automatic speech recognition. Current approaches effectively naively multiply out all the context features (e.g., next phone, preceding phone, position in syllable, ... etc) to create a vast state space with a cardinality equal to the product of the cardinalities of all the context features.

1.2. Data fragmentation and averaging

In order learn such a model from data in which only a tiny fraction of possible models actually have training examples, decision tree-based clustering is employed. Controlling the complexity of the resulting clustered model is non-trivial. The standard approach not only effectively fragments the data into disjoint subsets, for estimating the parameters of each context-dependent model[1], yet at the same time averages not just over multiple speech samples but over multiple clustered contexts. This averaging must inevitably lose some of the detail, which may be necessary for natural-sounding speech. This loss of fine detail is thought to be a main contributing factor to the 'muffled' sound of the generated speech.

1.3. The linguistic specification

Another problem is that, although the statistical parametric method learns from speech data using a well-defined objective function, it still relies on the linguistic specification, which is the basis for the context-dependent modelling. A properly-designed question set – for example, using categories of place and manner of articulation – is required when using these context features to cluster the acoustic models. Obtaining the knowledge and data required to construct the linguistic specification, and to properly use it for model clustering, is difficult for under-resourced languages.

1.4. An alternative approach to the linguistic specification: a Vector Space-based Front End

From the field of NLP, we have drawn inspiration from work in which letter and word representations are constructed in a way that is optimal for a certain specific task. [7] describes a self-organizing codebook that is jointly optimized with the text-

[1] Of course, Expectation-Maximisation training actually uses 'soft counting' and not strictly hard assignment of speech frames to single model parameters, but nevertheless each speech frame contributes to the estimate of only a very few model parameters.

to-speech model ensuring that the coding is optimal in terms of overall performance. Experiments showed that performance is improved compared to baseline system using orthogonal letter codes. [8, 9] take the idea of task-based estimation of word embeddings from neural net language modelling and apply it to an array of NLP tasks in a multitask learning framework. From these ideas, we have developed in previous work a Vector Space Model-based approach to constructing the linguistic specification.

[15] describes in detail our approach to the unsupervised construction of representations for context modelling in TTS – here we just give a brief summary of how this approach works. The vector space model (VSM) is well established in Information Retrieval (IR) and Natural Language Processing (NLP) as a way of representing objects such as documents and words as vectors of descriptors. To build vector space models, co-occurrence statistics are gathered in matrix form to produce high-dimensional representations of the distributional behaviour of word and letter types in the corpus. Lower dimensional representations are obtained by approximately factorizing the matrix of raw co-occurrence counts by the application of slim singular value decomposition (SVD). The main advantage of the VSM model over the traditional decision tree is that, instead of querying conventional features of linguistic objects, such as the phonetic class to which a phoneme belongs or the part of speech to which a word belongs, the objective distance between the VSM output values directly represents the similarity of two units. The VSMs are learned in an unsupervised fashion from text: no labeled speech is required.

In that previous work, we employed this novel front end in conjunction with conventional decision tree-based model clustering of HMMs-of-context-dependent-units [15]. However, the recursive hard partitioning of the vector space by the decision tree is not entirely satisfying, since it may lose some of the representational power of the VSM and in particular is unable to take best advantage of any factorial structure in the space. Therefore, we now consider a different way to map from the continuously-valued linguistic specification constructed by the VSM, to the parameters of Gaussians from which we can ultimately generate synthetic speech.

1.5. Deep Neural Networks

Compared with the decision tree clustering method, neural networks may be able to better model the interactions between linguistic and acoustic features, including correlations and factorial behaviour. Neural networks were used in [13] to map between a sequence of phonemes and an acoustic description from which a speech waveform can be generated. More recently, in [14] a DNN is employed to establish a mapping between phoneme-level labels represented as a large set of binary input features (derived from the question set that would otherwise have been used in decision tree clustering), numerical input features (for position and phone/syllable number) and target acoustic features. The DNN predicts output on a frame-by-frame basis.

There are several design choices to be made in creating a DNN-based synthesiser, which are not explored in [14]. So, in this paper we offer some comparisons of different input representations and different timescales for predicting the acoustic features.

We compare the DNN approach to TTS with three kinds of input representation: letter-based binary input, phoneme-based binary input, and a letter-based VSM. The binary input is similar to that in [14], with one binary feature for every possible two-way partition of the linguistic feature space. The original motivation behind the VSM front-end, as alluded to earlier, is an attempt to construct a linguistic specification from text using only unsupervised learning, motivated by a need to build TTS systems for under-resourced languages. The resulting representation comprises a continuous vector space which reflects the distributional properties of the training material.

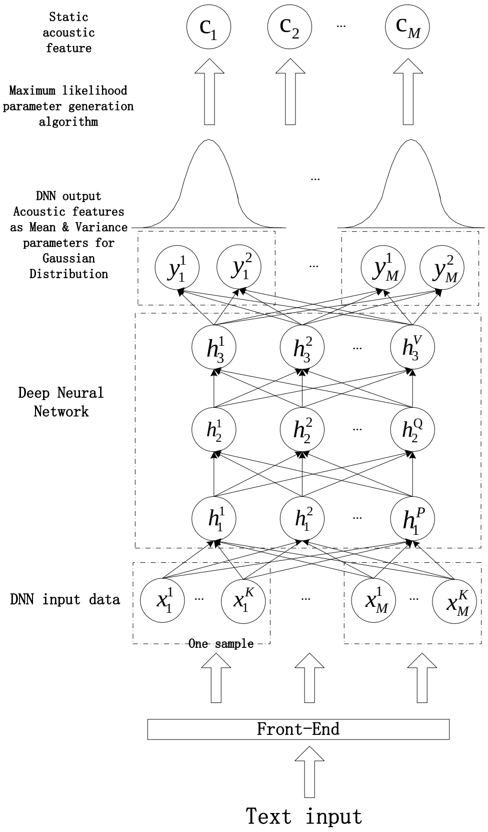

Figure 1: Framework for Deep Neural Networks based TTS

2. System architecture

2.1. Framework

Figure 1 illustrates the framework of the DNN-TTS method. In the training stage, text is first transformed into DNN input labels x_i^j. Where $i = 1, 2, ..., M$ denotes the ith DNN input label vector, and $j = 1, 2, ..., K$ denotes the jth element in the ith DNN input label vector. In this work, both frame-based and state-based systems were trained. In the case of the frame-based DNN, M is set to the total number of frames in the utterance, and the DNN is trained to map from binary labels to acoustic features y_k^l, where k indexes the DNN output vector and l indexes the elements of the output vector. In addition to static spectral envelope features, the acoustic feature vectors also include dynamic features derived from them.

In the case of the state-based DNN, the DNN is trained to map from context-dependent labels to the HMM model parameters for each context-dependent HMM state. The full-context space in speech synthesis is the product of cardinalities of each context feature, and is thus of enormous size.

To summarise, three various representations for the input x_i^j are compared:

1. Phone-based binary features, in which the linguistic context features are converted to a binary representation according to the answers to the conventional set of questions used to build parameter-tying decision trees

2. Letter-based binary features, constructed in an analogous way

3. VSM-based continuous features, as described in section 1.4.

At the synthesis stage, distributions over acoustic features y_k^l are predicted by the DNN from the input features x_i^j for each frame, or for each HMM state. Finally, static acoustic features $\boldsymbol{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_M\}$ are generated from those distributions using MLPG, and a vocoder is employed to synthesise a waveform from the generated acoustic features. In our experiments, only the means of Gaussian distributions are predicted, and we use pre-computed fixed variances.

2.2. Deep Neural Network Training

In contrast to the data fragmentation inherent in decision tree clustering, a DNN is trained as a single model using all data, via back-propagation, meaning that every training sample effectively contributes to the training of every model parameter (the weights in the DNN). In addition to varying the input representation and temporal granularity, our experiments also varied the number of hidden layers from 1 to 3 layers; we use the weights trained in shallower DNN to initialize the weights for Deeper DNN in the training process. The up to 3 hidden layers have 1000, 500, and 250 units, respectively. And linear layer is used as the output layer.

2.3. Maximum likelihood parameter generation

After generating the acoustic features y_k^l (including both static and dynamic features) from the DNN, we use the maximum likelihood parameter generation (MLPG) method [4] to generate smooth parameter trajectories to drive a vocoder.

3. Experiments

Six systems were built for comparison, as defined in Table 1. Systems C and E are HMM-based benchmark systems that use decision trees to map from the linguistic specification to model parameters; the remainder are DNN systems. Because there are typically more letters than phonemes in a given word or phrase, for the letter-based systems 3 state HMMs are used, whereas for phones 5 state HMMs are used.

3.1. Database

A database of a British English male speaker sampled at 16kHz was used for the experiments. There were 1000 utterances in the database, from which 860 were used for training, and 140 were held out for subjective and objective evaluation. 21-order Line Spectral Pair (LSP) plus delta and delta-delta were extracted to represent the spectrum. Log F0 plus delta and delta-delta were used to model pitch. 10-order features were extracted to model aperiodicity (AP). 3 binary values are used to represent voicing and its delta and delta delta.

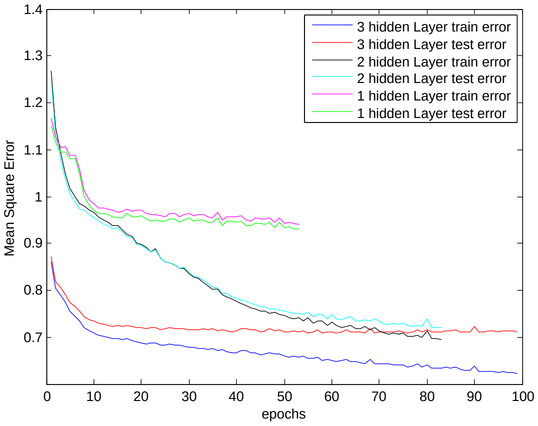

Figure 2: Mean square error for system A (letter-based binary DNN-TTS)

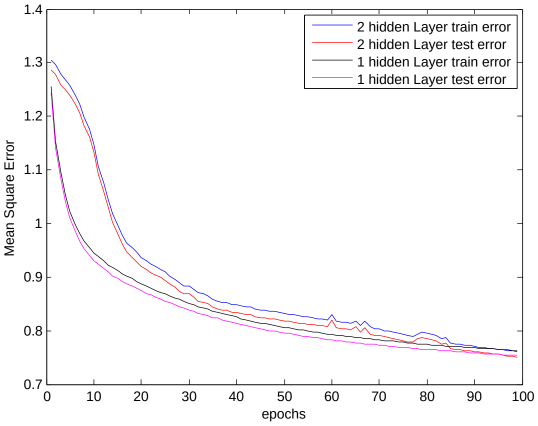

Figure 3: Mean square error for system B (letter-based VSM DNN-TTS)

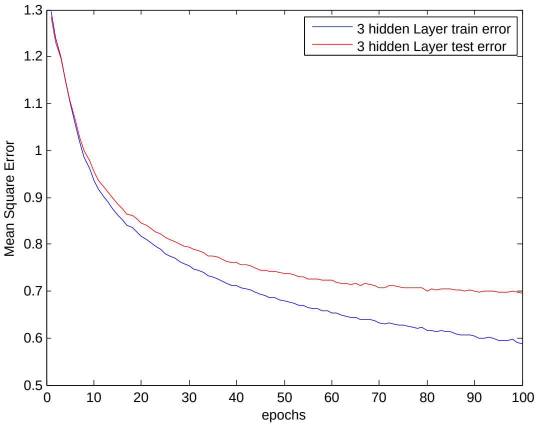

Figure 4: Mean square error for system F (phone-based context-dependent state mapping DNN-TTS)

System	Type	Linguistic unit	Input	Timescale
A	DNN	letter	binary	state
B	DNN	letter	continuous	state
C	Decision tree	letter	labels	state
D	DNN	phone	binary	frame
E	Decision tree	phone	labels	state
F	DNN	phone	binary	state

Table 1: Summary of systems built

3.2. Binary Front end

For the phone-based binary representation, each context-dependent phonetic label is rewritten as a 1256-dimensional binary vector. Features incorporated into this vector include the identities and phonetic categories of the {previous, current, following} phones, as well as word and phrase level positional features. Utterance-level features were excluded in these experiments in order to reduce the input vector dimensionality. In the case of the frame-based DNN, frame position within the current phone (both forwards and backwards) and total length in frames of the current phone are appended as numerical features to the DNN input vector. In the case of the state-based DNN, 5 binary values are appended to the input vector to encode state index. For the binary letter-based representation, letter-related context features are encoded in a 1134-dimensional binary vector for input to the DNN. For the VSM method, each letter in the input text is represented as a 27-dimensional vector of continuous values.

3.3. Objective evaluation

Figures 2 to 4 show the the training and testing error for systems A, B and F respectively. In 2, in can be seen that as the number of DNN hidden layers in increased, error decreases. The plots for the DNNs with 3 hidden layers exhibit some evidence of over-training. In 3, the 2 hidden layer error is not obviously smaller than the 1 hidden layer neural network. This is probably because the input dimension for VSM is only 27, so the DNN does not need so many parameters for the mapping. The training and test error for the 3 hidden layer result is shown in figure 4 for system F, where training and test error is smaller than the equivalent letter-based system A in figure 2. Figure 5 shows MLPG-generated LSP trajectories from system F, compared to natural LSPs. The predicted LSPs lack detail, compared with the natural ones.

Root Mean Square Error (RMSE) between generated LSPs and natural LSPs for a total of 140 utterances were calculated for each system. The results are shown in Table 3. From this result we can see that the two HMM + decision tree baseline systems C and E (letter and phone based respectively) still perform better than the DNN-TTS systems. System F is the best amongst the DNN-based systems, and informal listening confirmed that the voice generated by this system sounds reasonable.

3.4. Subjective evaluation

Six pairwise AB naturalness tests were conducted. 40 utterances were synthesized for each AB test, and 19 native speakers of English were asked to choose the more natural one from each pair. Results are shown in Table 2. All preferences in this Table are significant. The subjective listening test results are in general agreement with the objective measure.

System	A	B	C	D	E	F
A		>				<
B	<		<			
C		>			<	
D						<
E			>			>
f	>			>	<	

Table 2: Subjective listening test results. Blank cells indicate comparisons that were not tested. > indicates that the system named in the row was judged to be better than the system named in the column, and < indicates the reverse.

System	LSP Error
A	0.222398
B	0.228053
C	0.157672
D	0.244260
E	0.135460
F	0.176447

Table 3: Root Mean Square Error for LSP

4. Conclusions

We have presented an attempt to use DNNs to replace decision tree parameter clustering, motivated by a desire to take better advantage of the continuously-valued features produced by our novel VSM-based front-end. From the results obtained, the HMM benchmark systems still outperform the DNNs. It is not surprising that phone-based systems are better than letter-based ones – the point of the letter-based system is not to be better, but rather to be easier to construct for many different languages. The VSM front end does not do as well as hoped, but the comparison with the other systems is not quite fair because the VSM system uses no suprasegmental contextual information. Future work includes refining the input representation further, combining the binary and VSM features into a single system, training the DNN on substantially more data.

5. Acknowledgements

This research was supported by an EPSRC programme grant grant EP/I031022/1 (Natural Speech Technology) and has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 287678 (Simple4All).

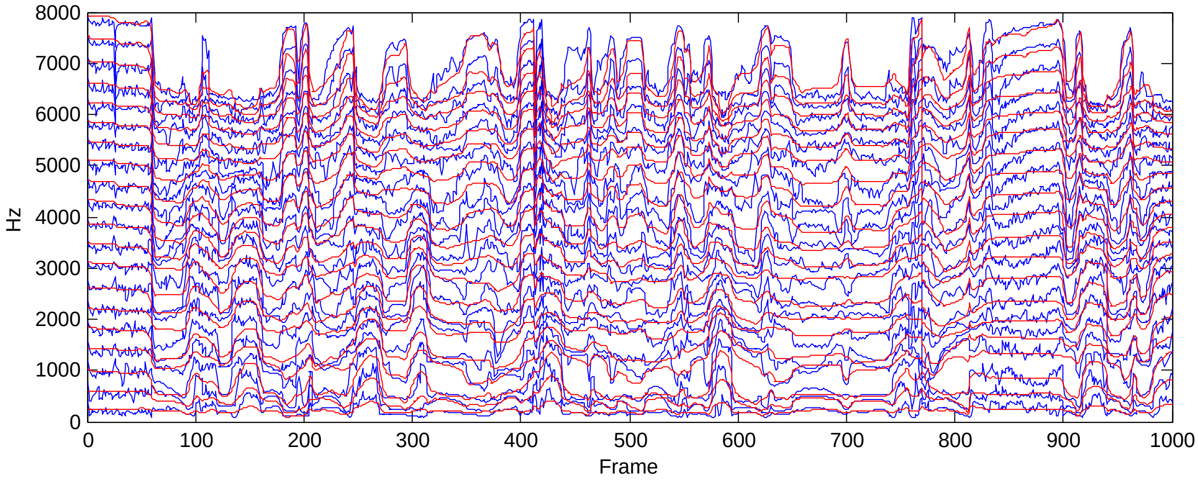

Figure 5: MLPG generated LSP by system F (phones, state units, DNN) in red compared with natural LSPs in blue.

6. References

[1] D. Klatt, "Review of text-to-speech conversion for english," *J. Acoust. Soc. Amer.*, vol. 82, pp. 737–793, 1987.

[2] J. Allen, S. Hunnicutt, and D. Klatt, "From text to speech: The mitalk system," *Cambridge Univ. Press*, 1987.

[3] R. A. J. Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the Festival speech synthesis system," *Speech Communication*, vol. 49, no. 4, pp. 317–330, 2007.

[4] K. Tokuda, T. Yoshimura, T., Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," *Proc. of ICASSP*, pp. 1315–1318, 2000.

[5] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, "Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis," *Proc. of Eurospeech*, pp. 2347–2350, 1999.

[6] "HTS," http://hts.sp.nitech.ac.jp/.

[7] Kåre Jean Jensen and Søren Riis, "Self-organizing letter code-book for text-to-phoneme neural network model," in *INTERSPEECH*, 2000, pp. 318–321.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa, "Natural language processing (almost) from scratch," *CoRR*, vol. abs/1103.0398, 2011.

[9] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *International Conference on Machine Learning, ICML*, 2008.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, pp. 1527–1554, 2006.

[11] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[12] G. Dahl, E. George, Y. Dong, D. Li, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, pp. 30–42, 2012.

[13] O. Karaali, G. Corrigan, and I. Gerson, "Speech synthesis with neural networks," *Proceedings of the 1996 World Congress on Neural Networks*, 1996.

[14] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," *Proc. of ICASSP*, 2013.

[15] O. S. Watts, "Unsupervised learning for text-to-speech synthesis," *Ph.D. dissertation, University of Edinburgh*, 2012.