Towards Deep Generative Speech Modeling

Robert Pinsler
Department of Computer Science, TU Darmstadt
robert.pinsler@stud.tu-darmstadt.de

Abstract

Generating sequences of human speech is particularly challenging because of both the high dimensionality and variability of the speech signal. With recent advances in deep learning, recurrent neural networks (RNNs) have become a powerful tool to model such sequences. In this paper, RNNs are applied to generate human speech. We make use of mixture density networks, a combination of a neural network and a mixture model, to account for the complexity underlying speech signals. First, we generate spectrograms of speech and later extend our experiments to produce speech waveforms. Whereas the additional mixture model helped to capture variances, experimental results were still far from resembling realistic speech.

1 Introduction

Using generative models to artificially create new and realistic sequences of data has long been an appealing yet very challenging research direction. Until recently, this field was dominated by dynamic bayesian networks such as hidden Markov models (Chung et al., 2015). However, due to recent advances in deep learning, recurrent neural networks (RNNs) are increasingly becoming popular again. For example, they were successfully applied to generate text (Sutskever et al., 2011) and music (Boulanger-Lewandowski et al., 2012).

RNNs are well-suited for such tasks because they recurrently process an input sequence while maintaining some hidden state. This allows them to capture correlations across multiple timesteps. RNNs can be trained for sequence generation by iteratively predicting the next element of a sequence. Given this output is probabilistic, it is possible to sample from the predicted output distribution of the trained network. Each generated output is then incorporated as an input for the respective next step, thereby defining a conditional distribution over sequences that depends on previous inputs. This justifies the generative nature of the approach, even though the network itself is inherently deterministic (Graves, 2013).

Human speech is a particularly interesting type of sequence as the underlying high-dimensional speech waveform is the result of complex physiological interactions during human speech production. By extracting few characteristic parameters, it is possible to learn a model that takes a sequence of speech parameters and recursively generates a feature vector at the next time step.

In this paper, the goal is to generate speech by applying RNNs. First, we generate spectrograms that should *visually* resemble human speech. This is conceptually simpler, but does not allow to listen to the output. We then extend this approach to generate speech waveforms. However, for the sake of visualization we will also utilize spectrograms to present those results.

Predicting real-valued sequences such as speech can be formulated as a standard regression problem, where the goal is to minimize the mean squared error (MSE) between the ground truth and the predicted values. One drawback of this formulation is that it can only model a single Gaussian distribution. This assumption does likely not hold for applications such as modeling human speech. Therefore, we make use of a mixture density network (Bishop, 1994) that combines a mixture model with a neural network. Our RNN is composed of multiple long short-term memory (LSTM) units, which are special gated activation functions to improve the memory of the network.

Speech samples for training are taken from the Simple4All Tundra Corpus (Stan et al., 2013), comprising of short excerpts of audio books. We restrict ourselves to English speech.

Our approach is inspired by the work of Graves (2013), who applied similar techniques to generate handwriting[1]. Chung et al. (2015) recently introduced the variational RNN (VRNN) as an alternative deep generative model, which was able to achieve state-of-the-art results for speech generation. Generating and synthesizing natural-sounding speech is also a well-known problem in the context of text-to-speech (TTS) systems (Black et al., 2007). Zen et al. (2013) successfully replaced HMMs by deep networks to produce speech from linguistic inputs. This approach was later extended by including a mixture density network (Zen and Senior, 2014).

The remainder of the paper is organized as follows. In section 2, we introduce the theoretical background for sequence generation with RNNs. The experimental setup is outlined in section 3, followed by the results presented in section 4. Section 5 summarizes our findings and suggests future work.

2 Sequence Generation with Recurrent Neural Networks

RNNs can be used to generate sequences by recursively processing an input sequence $x = (x_1, x_2, ..., x_T)$ and iteratively producing an output vector sequence, $y = (y_1, y_2, ..., y_T)$, while maintaining some hidden state sequence $h = (h_1, h_2, ..., h_T)$. Given an input symbol x_t for each timestep t, the hidden state h_t is updated as follows:

$$h_t = f(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$

where f is the hidden layer activation function, b_h denotes the bias term of the hidden layer, and W_{ih} as well as W_{hh} represent the input-hidden layer and the recurrent hidden layer weight matrices, respectively. From the hidden state h_t, the output is obtained by:

$$\hat{y}_t = b_y + W_{hy}h_t$$
$$y_t = g(\hat{y}_t)$$

where g is the output layer activation function and the indexes follow the convention introduced before. This effectively defines a mapping from input histories $x_{1:t}$ to output vectors y_t. Those can in turn be used to define the predictive distribution $p(x_{t+1}|y_t)$ over possible next states x_{t+1} by conditioning on y_t. The joint sequence probability distribution is therefore factorized into a product of conditional probabilities:

$$p(x) = \prod_{t=1}^{T} p(x_{t+1}|y_t)$$

with the negative log likelihood as the corresponding sequence loss function $\mathcal{L}(x)$:

$$\mathcal{L}(x) = -\sum_{t=1}^{T} \log p(x_{t+1}|y_t) \tag{1}$$

The network can be trained using gradient-based optimization, after the gradient was obtained by backpropagation through time (Williams and Zipser, 1995).

Despite their ability to store information about sequences, standard RNNs are unable to keep track of long-term dependencies in the data. One possible remedy is to use gating mechanisms, e.g. as in long short-term memory (Hochreiter and Schmidhuber, 1997; Gers et al., 1999) or gated recurrent units (Cho et al., 2014). In the following, we will focus on the well-studied long short-term memory (LSTM) formulation.

2.1 Long Short-Term Memory

LSTM replaces the element-wise sigmoid activation function typically found in RNN hidden layers by a memory cell to store and retrieve information. As depicted in Figure 1, the memory cell itself is composed of different gating activation functions designed to improve its memory:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$
$$h_t = o_t \tanh(c_t)$$

where the logistic sigmoid function is denoted as σ and i, f, c and o represent the *input gate*, *forget gate*, *cell* and *output gate*, respectively. The weight matrices are indexed according to where they belong to, e.g. W_{ho} is the hidden-output gate weight matrix.

As in standard RNNs, backpropagation through time (BPTT) can be used to optimize the network.

[1] Throughout this paper, we largely adopt notation of Graves (2013).

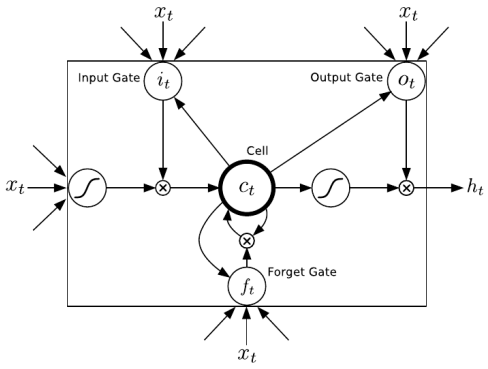


Figure 1: LSTM cell (Graves, 2013)

2.2 Mixture Density Network

Neural networks that were trained to minimize the MSE learn to approximate the conditional mean of the target data. In case of speech modeling, where the output distribution is multimodal, it is very unlikely that averaging over possible solutions leads to some meaningful output. Furthermore, it is only possible to model a single Gaussian distribution. Mixture density networks (MDNs) solve those issues by combining a neural network with a mixture model (Bishop, 1994). This can be achieved by parametrizing the mixture distribution with the network's output, where some of the outputs are transformed into mixture weights while parameters of the individual mixture components are modeled by the remaining output values. If some of the parameters are bounded, the output has to be adapted accordingly. MDNs can be extended to work with RNNs by conditioning the output distribution on the complete history of inputs (Schuster, 1995). In this paper, an isotropic Gaussian mixture model (GMM) is used to define the distribution over possible next input vectors x_{t+1} given output vector y_t:

$$p(x_{t+1}|y_t) = \sum_{i=1}^{m} \alpha_t^i \, \mathcal{N}(x_{t+1}|\mu_t^i, \sigma_t^i) \qquad (2)$$

where y_t consists of a set of means μ^j, standard deviations σ^j and mixture weights α^j corresponding to the M mixture components, i.e. $y_t = \{\mu_t^j, \sigma_t^j, \alpha_t^j\}_{j=1}^M$. The output vector y_t is constructed by applying suitable activation functions to the RNN output $\hat{y}_t$:

$$\hat{y}_t = \{\hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\alpha}_t^j\}_{j=1}^M = b_y + W_{hy} h_t$$

The weights α_t^j have to form a probability distribution, which can be achieved by utilizing the softmax function:

$$\alpha_t^j = \frac{\exp(\hat{\alpha}_t^j)}{\sum_{j'=1}^{M} \exp(\hat{\alpha}_t^{j'})}$$

such that $\sum_{j=1}^{m} \alpha_t^j(x) = 1, \ \alpha_t^j \in (0, 1)$. Applying an exponential function ensures standard deviations σ_t^j are non-negative:

$$\sigma_t^j = \exp(\hat{\sigma}_t^j)$$

The means μ_t^j are unconstrained and therefore do not need to be transformed:

$$\mu_t^j = \hat{\mu}_t^j$$

The network is trained by minimizing the loss function $\mathcal{L}(x)$, which can be obtained by combining Eq. (1) and Eq. (2):

$$\mathcal{L}(x) = \sum_{t=1}^{T} -\log \left(\sum_{j=1}^{M} \alpha_t^j \, \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j) \right) \qquad (3)$$

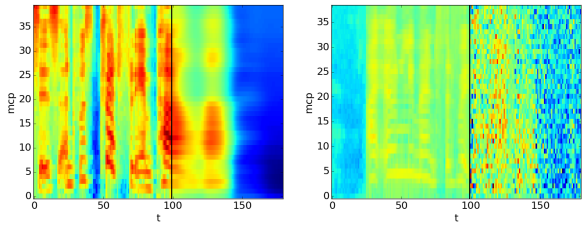
3 Experimental Setup

4.5 hours of English speech from the Simple4All Tundra Corpus (Stan et al., 2013) were used as training data. Each sample is a short excerpt (e.g. a sentence) from an audio book recited by a single female speaker. It is based on the novel *Living Alone* written by Stella Benson in 1919, which commonly uses poetic language.

The audio files were downsampled from 44.1KHz to 16KHz. From that, 40 mel-cepstral coefficients (*mcp*) were extracted at a framerate of 80fps and a window size of 0.025s. In a first experiment, those features were utilized to generate novel *mcp* vectors, from which a spectrogram can be produced. This approach is later extended to generate speech waveforms.

3.1 Spectrogram Generation

As an additional preprocessing step, the first and last 50 timesteps of each audio sample were removed to exclude potentially noisy parts of the speech signal. Input features were normalized to have zero mean and unit variance. To enforce the same fixed length for each input sequence, each sample was divided into instances spanning 100 timesteps with an overlap of 10 timesteps. During training, the last element of each sequence was used as the label.



(a) LSTM-L (b) LSTM-MDN

Figure 2: Generated spectrograms from *mcp* vector sequences. The first 99 timesteps were provided as the initial input sequence, whereas the following 81 timesteps were iteratively generated by the model.

Our model consists of two 128-dimensional LSTM layers, followed by a GMM layer with $M = 5$ Gaussian mixture components (LSTM-MDN). It takes a sequence of 40-dimensional *mcp* vectors as input and produces a *mcp* vector as its output. Dropout with $p = 0.2$ was interspersed after every LSTM layer to regularize the network.

For comparison, another model was trained where the GMM layer was replaced by a fully-connected layer with linear activation function (LSTM-L). Both networks were optimized with RMSProp (Tieleman and Hinton, 2012), where gradient clipping was used to keep values within $[-10, 10]$, thereby mitigating numerical instabilities during training. Finally, spectrograms were generated from the produced network outputs.

3.2 Speech Generation

In addition to *mcp* parameters, logarithmic fundamental frequency ($\log F_0$) and maximum voiced frequency values (*mfv*) were obtained from the audio samples using AHOCoder (Erro et al., 2011). Explicit voicing modeling (Yu and Young, 2011) was applied in order to eliminate discontinuities of the $\log F_0$ sequence, i.e. $\log F_0$ values during unvoiced speech parts were linearly interpolated and a binary value (*vuv*) was added to the output features. $\log F_0$ output values were later replaced again in case the speech part is classified as unvoiced. The first and last 50 frames of each sample were removed and input features were normalized. Samples were cut into fixed-length training instances as before, where the respective last timestep serves as a label.

Again, two models LSTM-MDN and LSTM-L were trained in a similar fashion as before, now

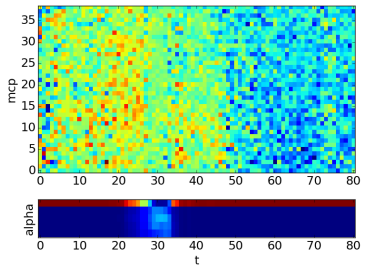


Figure 3: Spectrogram generated by LSTM-MDN model (*top*) and corresponding activations of GMM weights at each timestep (*bottom*).

taking 43 input features (40 *mcp* + 1 $\log F_0$ + 1 *mfv* + 1 *vuv*). Modeling the voiced/unvoiced bit x_{vuv} as a Bernoulli random variable extends Eq. 3, yielding:

$$\mathcal{L}(x) = \sum_{t=1}^{T} -\log\left(\sum_{j=1}^{M} \alpha_t^j \, \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j)\right) - \begin{cases} \log e_t & \text{if } x_{\text{vuv}} = 1 \\ \log(1 - e_t) & \text{otherwise} \end{cases}$$

To ensure a smooth speech output signal, the speech parameter generation algorithm (Tokuda et al., 2000) is usually applied as a post-processing step before the speech parameters are synthesized; see e.g. Zen and Senior (2014). However, this did not prove to be helpful. One possible explanation might be that it requires to model delta and delta-delta features of the speech parameters (i.e. first and second derivatives), which further complicates the learning problem. Therefore, we directly used the generated speech parameters to synthesize speech waveforms based on the harmonics plus noise model of AHOCoder.

4 Results

4.1 Spectrogram Generation

Spectrograms were produced by randomly selecting an unseen sequence of *mcp* vectors and iteratively predicting the output at the next timestep. Figure 2 depicts generated spectrograms for both trained models. The LSTM-L model approximates the conditional mean of the target data and therefore technically does not generate novel sequences. While the produced spectrogram shows similarities to real speech, the model fails to reproduce the natural variability of the human voice. Furthermore, the *mcp* vectors converge to some low-energy state towards the end. This

is not untypical in real speech per se, but it is important to note that the model is particularly prone to such behavior. One reason is because it always treats its own predictions as the *true* input for the next timestep, making it hard to recover from previous decisions.

In contrast to that, the LSTM-MDN model produces outputs with high variability. The resulting spectrogram looks very noisy due to missing continuity between neighboring timesteps, which could be an indicator for overfitting. As a side effect, the higher range of values has diminished the visual variability of the initial input sequence.

As can be seen in Figure 3, the mixture model is clearly dominated by one component. This suggests that most of the variability of the model stems from the variance of this component. In other words, instead of learning multiple Gaussian components with moderate variances it has modeled a single Gaussian with high variance.

4.2 Speech Generation

To produce speech, we additionally generated *lf0*, *mfv* and *vuv* values from the model. Those parameters were used to synthesize speech using AHOCoder. Afterwards, spectrograms were extracted to visualize the results.

Figure 4a suggests that the LSTM-L model has learned a periodic function. After a short unvoiced speech part with high variability, the sequence converges towards some recurring pattern. Whereas a short excerpt thereof could resemble human speech, the regular pattern as a whole is unrealistic.

On the other hand, the spectrogram produced by the speech LSTM-MDN model depicted in Figure 4b shows a very noisy sequence. It uses only two mixture components, but from visual inspection it is unclear what those two modes correspond to. The predicted *vuv* sequence often quickly alternates between voiced and unvoiced, which is untypical for human speech. Another interesting behavior is that the model predicts lower values for the first few *mcp* components and increasingly higher values for subsequent components.

4.3 Discussion

The trained models were not able to generate realistically-looking speech sequences. This might be due to various reasons. As already stated by Chung et al. (2015), the main difficulty is to model the trade-off between generating a clean signal and sufficient variability within a single sequence as well as across different samples. This makes it particularly challenging to apply the right amount of regularization to prevent overfitting. For example, our experiments confirmed findings by Graves (2013) that adding a fixed amount of noise makes it harder to learn mixture model weights.

Furthermore, assuming isotropic Gaussians in case of the LSTM-MDN model is likely too simplistic. By explicitly modeling covariances, it is possible to capture interactions between variables at the cost of an increased number of parameters. Hyper-parameter optimization is expected to improve model performance as well. This includes parameters during pre- and post-processing, e.g. for encoding and decoding the speech signal.

5 Summary and Outlook

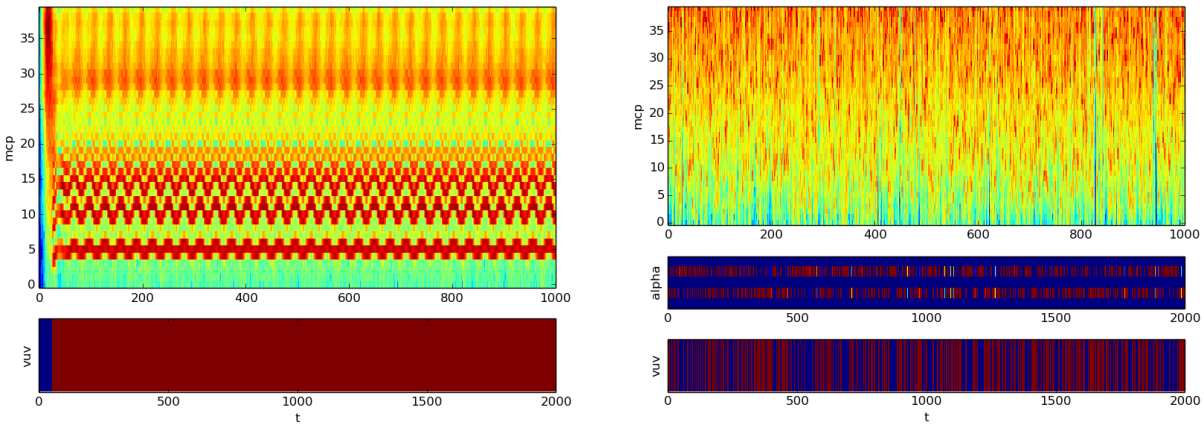
Producing realistic human speech is a particularly interesting yet hard sequence generation problem. In this paper, we used LSTMs as part of a MDN to generate speech signals. While the produced spectrograms share some aspects with real speech, they are still far from looking similar. This suggests we need to spend more time on training the model, especially in terms of regularization and hyper-parameter optimization. Experimenting with increased model complexity, e.g. through deeper architectures or by including covariances into mixture components, is left for future work as well. Another interesting direction is to change the input-output mapping, for example by predicting multiple timesteps at once.

References

C. Bishop (1994). Mixture density networks. Technical Report.

A. Black, H. Zen and K. Tokuda (2007). Statistical Parametric Speech Synthesis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 1229–1232. Hawaii, USA.

N. Boulanger-Lewandowski, Y. Bengio and P. Vincent (2012). Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In Proceedings of the 29th International Conference on Machine Learning, 1159–1166. Edinburgh, Scotland.



(a) LSTM-L

(b) LSTM-MDN

Figure 4: Spectrograms extracted from generated sequences that were produced by speech models (*top*). Predicted *vuv* flag (*bottom*) at each timestep indicates whether speech was classified as voiced (*red*) or unvoiced (*blue*). For the LSTM-MDN speech model, corresponding activations of GMM weights are shown as well (*middle*). Note the different time scales due to post-processing of the generated speech parameter vectors.

K. Cho et al. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1724–1734. Doha, Qatar.

J. Chung et al. (2015). A Recurrent Latent Variable Model for Sequential Data. In *Advances in Neural Information Processing Systems 28*, 2962–70. Montreal, Canada.

D. Erro, et al. (2011). Improved HNM-based Vocoder for Statistical Synthesizers. In *Proceedings of InterSpeech 2011*, 1809–1812. Florence, Italy.

F. Gers et al. (1999). Learning to forget: continual prediction with LSTM. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, 850–855 (vol. 2). Edinburgh, Scotland.

A. Graves (2013). Generating Sequences With Recurrent Neural Networks. arXiv preprint arXiv:1308.0850.

S. Hochreiter and J. Schmidhuber (1997). Long Short-Term Memory. *Neural Computation* **9(8)**: 1735–1780.

M. Schuster. Better Generative Models for Sequential Data Problems: Bidirectional Recurrent Mixture Density Networks. In *Advances in Neural Information Processing Systems 2012*, 589–595. Lake Tahoe, USA.

A. Stan et al. (2013). TUNDRA: a multilingual corpus of found data for TTS research created with light supervision. In *14th Annual Conference of the International Speech Communication Association*, 2331–2335. Lyon, France.

I. Sutskever, L. Martens and G. Hinton (2011). Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning*, 1017–1024. Bellevue, USA.

T. Tieleman and G. Hinton (2012). Lecture 6.5 - RMSProp, Coursera: Neural Networks for Machine Learning. Technical report.

K. Tokuda et al. (2000). Speech parameter generation algorithms for HMM-based speech synthesis. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1315–1318 (vol. 3). Istanbul, Turkey.

R. Williams and D. Zipser (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Backpropagation: Theory, Architectures and Applications*, 433–486.

K. Yu and S. Young (2011). Continuous F0 Modelling for HMM based Statistical Parametric Speech Synthesis. In *IEEE Transactions on Audio, Speech, and Language Processing* **19(5)**: 1071–1079.

H. Zen, A. Senior and M. Schuster (2013). Statistical parametric speech synthesis using deep neural networks. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 7962-7966. Vancouver, Canada.

H. Zen and A. Senior (2014). Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 3872-3876. Florence, Italy.