#02456: DEEP LEARNING

MUSIC GENERATION USING RECURRENT NEURAL NETWORKS

Christopher Heje Grønbech and Maximillian Fornitz Vording

DTU Compute, Technical University of Denmark

19 December 2016

ABSTRACT

We present two models both using a recurrent neural network to reconstruct music and for the latter model generate music. We use these models on a large data set of folk tunes from which melodies are extracted for the models to recreate. We test these models using both L_2 regularisation and the dropout technique. Both models achieve accuracies of around 37 % for predicting pitches and of around 75 % for predicting durations. We also take a closer look at reconstructions from the best models, and find that they keep the musical structure, like rhythmical motifs and musical scales, two of the cornerstones in musical theory.

1. INTRODUCTION

Making great music comes natural to some, but is near impossible for others. To create music, rules have been set up for what sounds or *tones* go great together, and this is called musical theory. Music is normally experienced as raw audio, but is simpler to represent using musical notation. This is analogous to the relationship between speech and text. In musical notation, a tone is represented by a note which has a frequency called a *pitch* as well as a *duration*. The absence of tones is also important and is called a *rest*. Putting several notes and rests into a sequence makes a *melody*, and it helps using musical theory to do this.

We have examined whether a recurrent neural network (RNN) can learn musical theory from several collections of folk music and recreate complete melodies and create new ones. To represent the melodies in the network, we were inspired by an article by Choi et al. on electronic health records [1]. Herein, a patient's hospitalisation history is represented by a sequence of hospitalisations with a multi-hot encoding of diagnosis codes and medication codes for each hospitalisation as well as a sequence of durations for time between hospitalisations. In an RNN, the predicted hospitalisation codes and durations are then conditioned separately on both the earlier hospitalisation codes and the earlier durations. Based on this idea, notes and rests could be represented as a sequence of one-hot encoded pitches (including no pitch for rests) as well as a sequence of durations of the notes and rests.

Another approach has been done by Zimmerman [2] by predicting pitch and possibly a chord every nth part of a beat using an RNN. Using the Nottingham Music Database [3], accuracies of about 77 % are achieved. Chung et al. have also used the same data set in comparing RNNs with long short-term memory (LSTM) and gated recurrent units (GRU) as well as to a simple RNN using a tanh activation function [4], but only using the pitch values. They find that the simple RNN performs the best.

In this report, we present three models using RNN: one similar to the one used by Choi et al. [1], one also using the previous output, and one only using the previous output. We use these models on the Nottingham Music Database to reconstruct the melodies therein. We also examine using different kinds of regularisation on the models.

2. DATA

The Nottingham Music Database [3] is encoded in ABC notation, which is musical notation in text format [5]. An example of ABC notation can be seen at the top of Figure 1. It includes a total of 1034 folk tunes divided into nine different collections

based genres or composers. The folk tunes are separated into a training set of 659 tunes, validation set of 165 tunes, and a test set of 207 tunes. Three tunes were omitted due to bad formatting.

To use the tunes in the models, the tunes are first converted from the ABC format to melodies with notes and rests. Then the pitches and duration are extracted from the sequence of notes and rests and put into a pitch sequence and a duration sequence, respectively. Rests are represented in the pitch sequence by a 0. Finally, both sequences are one-hot encoded, which results in 35 pitch values and 14 duration values. An end-of-sequence character has also been included in both sequences. This preprocessing is illustrated in Figure 1.

3. METHODS

Both models can be seen in Figure 2. In the following, the three models we have developed, the networks they use as well as their cost functions are described.

3.1. Conditional Distribution

The pitch $\mathbf{x}_{\mathrm{p}}^{(t)}$ and duration classes $\mathbf{x}_{\mathrm{d}}^{(t)}$ at step t in a melody sequence, can be denoted by the combined feature vector $\mathbf{x}^{(t)} = \{\mathbf{x}_{\mathrm{p}}^{(t)}, \mathbf{x}_{\mathrm{d}}^{(t)}\}$. The architecture of the full network model is based on the assumption of each note in the melody conditioning on all previous, so the next-step prediction of the network

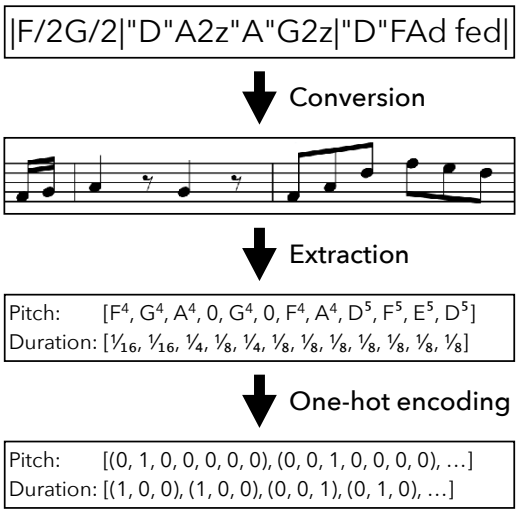


Figure 1 Extraction of pitch and duration sequences from ABC notation.

can be written as two conditional distribution:

$$p\left(\mathbf{x}_{\mathrm{p}}^{(t+1)} | \mathbf{x}^{(t' \leq t)}\right) = p\left(\mathbf{x}_{\mathrm{p}}^{(0)}\right) \prod_{t'=1}^{t+1} p\left(\mathbf{x}_{\mathrm{p}}^{(t')} | \mathbf{x}^{(t'-1)}\right) \tag{1}$$

where the conditional distribution is similar for the duration classes.

3.2. Input network

The input network merges the two one-hot-encoded class vectors into the two-hot-encoded vector, $\mathbf{x}^{(t)}$. The input network also provides the binary mask for each melody for the network, so empty predictions are made when reaching the end of each melody, when the values in the mask becomes zero.

3.3. Recurrent neural network

The main part of the models is the recurrent neural network in the centre transferring temporal information from the input network to the output network. To enhance the long term memory, gated recurrent units (GRUs) are used. These are receiving $\mathbf{x}^{(t)}$ for each step, t, in the melody. Each GRU run through the melody step by step as Chung et al. [4], calculating the hidden activations recursively by the activation functions:

$$\text{Reset:} \quad \mathbf{r}^{(t)} = \sigma_r\left(\mathbf{x}^{(t)}\mathbf{W}_{xr} + \mathbf{h}^{(t-1)}\mathbf{W}_{hr} + \mathbf{b}_r\right) \tag{2}$$

$$\text{Update:} \quad \mathbf{u}^{(t)} = \sigma_u\left(\mathbf{x}^{(t)}\mathbf{W}_{xu} + \mathbf{h}^{(t-1)}\mathbf{W}_{hu} + \mathbf{b}_u\right) \tag{3}$$

$$\text{Candidate:} \quad \mathbf{c}^{(t)} = \sigma_c\left(\mathbf{x}^{(t)}\mathbf{W}_{xc} + \mathbf{r}^{(t)} \odot \left(\mathbf{h}^{(t-1)}\mathbf{W}_{hc}\right) + \mathbf{b}_c\right) \tag{4}$$

$$\text{Activation:} \quad \mathbf{h}^{(t)} = \left(1 - \mathbf{u}^{(t)}\right) \odot \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \odot \mathbf{c}^{(t)} \tag{5}$$

where $\mathbf{W}_x$ are the vertical weights transforming the current input $\mathbf{x}^{(t)}$ linearly and the $\mathbf{W}_h$ are the horizontal weights transforming the previous activation linearly, so information about previous states can propagate horizontally through each step. The non-linear sigmoid functions are the logistic function ($\sigma(\mathbf{x}) = (1 + \mathrm{e}^{-\mathbf{x}})^{-1}$) for the reset gate in equation (2) and update gate in equation (3) and the hyperbolic tangent function for the candidate activation in equation (4). The reset gate controls how much of the signal from the previous activation, $\mathbf{h}^{(t-1)}$, will be a part of the current candidate

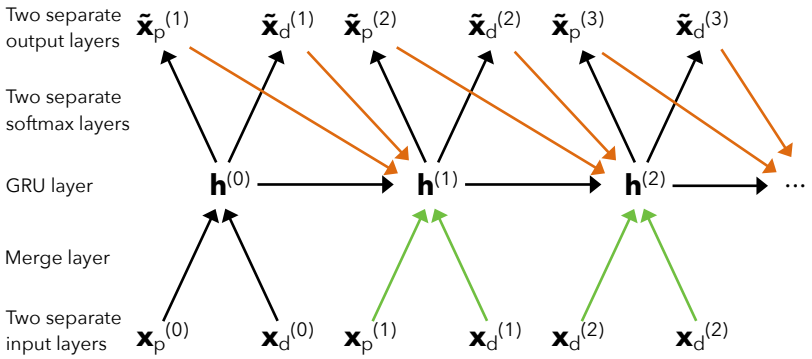


Figure 2 The setup for both models. $\mathbf{x}_p^{(t)}$ and $\mathbf{x}_d^{(t)}$ are the one-hot encoded pitch and duration, respectively, at position $t+1$ in each sequence. The black arrows represent connections in all models, orange arrows represent connections present in models 2 and 3, and green arrows represent connections which are ignored in model 3.

activation, $\mathbf{c}^{(t)}$, in equation (4) and the update gate then controls the proportional mixing of $\mathbf{h}^{(t-1)}$ and $\mathbf{c}^{(t)}$ in the new activation, $\mathbf{h}^{(t)}$, in equation (5). This means that $(\mathbf{r}^{(t)}, \mathbf{u}^{(t)}) = (0,1)$ will cut off the signal from the previous states and $(\mathbf{r}^{(t)}, \mathbf{u}^{(t)}) = (1,0)$ will copy the previous activation entirely. This makes it possible for the model to condition the next prediction over long terms, like the overall tonality and common motifs, and short terms, like previous pitch interval jumps and current harmony (chords). The models ability to perform this temporal conditioning in a musical manner is what will be investigated in the experiment section.

When computing the back-propagation through time for adjusting the model parameters in recurrent neural networks, vanishing gradients can be a huge problem leading to premature convergence (stopped learning). This is due to neurones in the network being saturated, when the gradient of the activation function is close to zero near convergence along the tails. The gradients will therefore vanish after multiple multiplication in the back-propagation, where each gradient depends on the previous. Exploding gradients can also occur when training recurrent neural networks, so a gradient clipping of 3 is used for all gradients.

3.4. Output network

The output network consists of two standard feed-forward neural networks with softmax activation functions for class probabilities. A neurones response to the hidden GRU states at time t represents the probability of the next note at time $t+1$ belonging to a specific pitch class for the pitch network and duration class for the duration network. The conditional probability for pitch class p at time $t+1$ is then given by the output network as:

$$p\left(\tilde{\mathbf{x}}_p^{(t+1)} | \mathbf{x}^{(t' \leq t)}\right) = s\left(\mathbf{W}_p \mathbf{h}^{(t)} + \mathbf{b}\right) \tag{6}$$

where $s(\mathbf{x})$ is the softmax activation function and the same holds for duration classes.

3.5. Extended model

By plugging in the next-step prediction of the output network to the next step input of the GRU layer an extra loop is formed which transfers additional information about the previous step and penalises the model more for prediction errors. Adding the most probable prediction $\tilde{\mathbf{x}}^{(t)}$ from previous step to the original input, $\mathbf{x}^{(t)}$ after weighting and transforming both through a ReLU dense layer will combine the new input $\hat{\mathbf{x}}^{(t)}$, which will go into the activation functions in Equation (2)-(5) in the exact same manner. This model is also referred to as model 2 and the normal model as model 1.

The only difference between the two models is the usage of the previous output, so if a wrong prediction in the previous output leads to higher misclassification in the next step an avalanche of misclassifications roll along the sequence. So during training the model would gain the lowest cost by just lowering the connection throughput of the previous output and finally reduce model 2 to model 1. To avoid this a third model without any

original input, except the first note to kickstart it has also been implemented.

3.6. Cost function

The categorical cross entropy cost function for example n is:

$$L_n = -\sum_{t}^{T-1}\sum_{k}^{K} \mathbf{x}_{\mathrm{k}}^{(t+1)} \log p\left(\tilde{\mathbf{x}}_{\mathrm{k}}^{(t+1)} | \mathbf{x}^{(t'\leq t)}\right) \qquad (7)$$

summed over all K classes and time steps except the last one, T. All time steps are treated as separate data points, so the targets and outputs are unrolled and the loss for each are averaged over the examples, n, in the mini-batch, after masking out the steps outside the melodies.

4. EXPERIMENTS

The experiments were run for model 1, the normal RNN model using only original input and model 2, the extended RNN model using the previous prediction step output as input to the next step.

The models and the experiments have been implemented in Python with the Theano and Lasagne modules using mini-batch gradient descent with the Adam optimiser [6] with a batch size of 10 and learning rate of 0.005. The code is available on GitHub[1].

4.1. Regularisation

Dropout of 20% or 50% was applied right before the GRU layer in the input network. By dropping out notes along the melodies, a lossy noise and therefore a completion task is introduced to the models, so during training the next-step prediction will rely more on the previous GRU activations $\mathbf{h}^{(t-1)}$ and the horizontal connections will be enhanced to make up for the missing input. The stronger horizontal connections will hopefully combine some more general musical rules and therefore reduce overfitting.

An L_2-regularisation term is also added to the total cost during training to also reduce overfitting.

[1] https://github.com/maximillian91/recurrent-neural-notes.

4.2. Evaluation

The prediction accuracies for both pitches and durations are recorded for each epoch and their learning curves are shown in Figure 3.

To investigate the connections in the GRU layer, the mean, Frobenius norm and positive-value fractions of horizontal and vertical weights are recorded (and shown in Appendix b), which are expected to show the trend in the overall magnitude and sign of the weights.

The two first models have been tested with and without dropout as well as with and without regularisation for 10, 25, 50, 75, and 100 gated recurrent units, so scaling of performance can be investigated and the effect of the regularisation methods compared. The third model was tested with and without L_2 regularisation. The scaling of performance with the number of GRU are clearly seen, but not included here, so 100 GRU are used as a standard and for these regularisation methods and model architectures can be compared for the test set in Table 1.

All models were trained for 200 epochs, as they reach convergence after about 150 epochs in the learning curves seen in Figure 3.

4.3. Reconstructions

After training, the models were used to reconstruct several test melodies, and one was handpicked out of the unseen test set as an example for this report. The first four bars of this melody is shown in Figure 4 together with reconstructions using models 1 and 2. In Appendix a, the notes in the original melody are highlighted by colours

Table 1 The test evaluation accuracies.

Model	p_{dropout}	α_2	$A_{\mathrm{pitch}}(\%)$	$A_{\mathrm{duration}}(\%)$
1	0	0	36.3	73.8
1	0.5	0	37.6	75.3
1	0	1	35.5	73.3
1	0.5	1	37.2	74.6
2	0	0	37.6	74.6
2	0.5	0	37.5	74.9
2	0	1	36.7	73.7
2	0.5	1	37.3	75.2
3	0	0	20.8	65.2
3	0	1	19.0	64.0

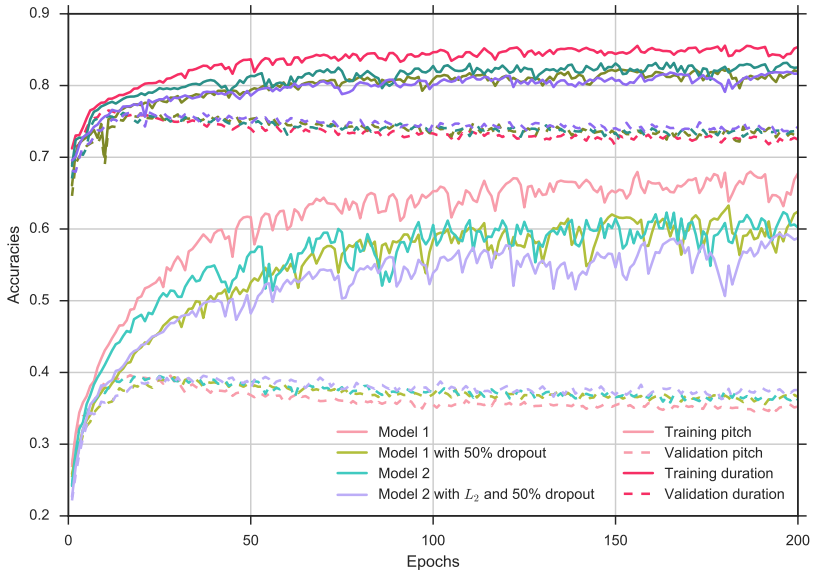


Figure 3 Learning curves over next-step prediction accuracies for both model types with and without regularisation. The models are evaluated on both training (solid lines) and validation sets (dashed lines) for pitch (turquoise) and duration classes (orange).

corresponding to the GRU activations, so patterns, like scale and rhythmical motifs, leading to higher activation can be localised and the functionality of the GRU identified.

To investigate the distribution of the pitch and duration classes, the frequencies of each class in the original data are represented in a histogram in Figure 5.

All reconstructions can be converted to a musical notation, written to a MIDI file, played, written as a musical score and compared against the original melody.

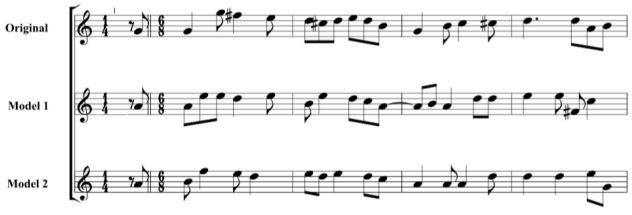


Figure 4 First 4 bars of notes in the melody, Fiddle Hill Jig, for original data and reconstructions produced by model 1 with dropout of 50% and by model 2 with dropout of 50% and L_2-regularisation.

5. DISCUSSION

By comparing the learning curves in Figure 3, all models converge after 150 epochs and starts overfitting after around 20 epochs of training even with regularisation. The regularisation does seem to lower overfitting somewhat, when validation curves move up and training curves down in accuracy, like e.g. for model 2 with L_2 regularisation and dropout, where the overall span and area between the (purple) curves is lower. So model 2 seems to generalise better than model 1, but the difference is so marginal, that they could almost be the same model as alluded to in Section 3. Model 2 could be forced to not reducing into model 1, by only providing the previous output as input, but following the test results for model 3 in Table 1, this seems to handicap this model a lot. Investigating the weights of the RNN, it is found that the Frobenius norm of the weight matrices keep on increasing and the mean varying gradually, so gradients are not vanishing and weights are adjusted during training. Plots of the weights can be seen in Appendix b.

The reconstructions using The model 1 and 2 can be compared to the original example in Figure 4, where it can be seen that the models can definitely catch the scale of the melody and mostly

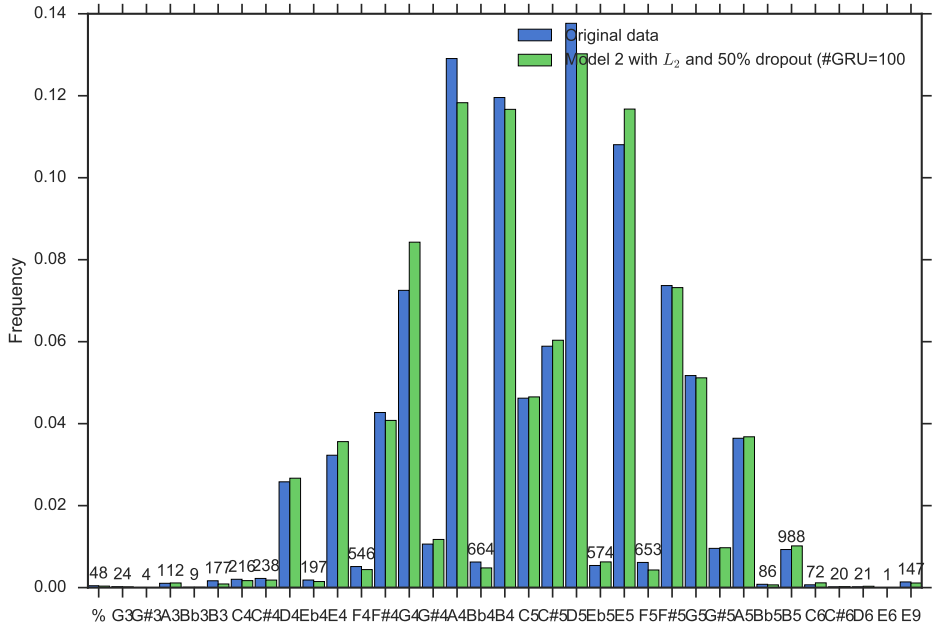


Figure 5 Histograms showing statistical frequency of pitch classes in the (blue) original data and in the (green) reconstructions produced by model type 2 with dropout of 50% and L_2-regularisation.

predict pitches belonging to this scale. The third note, F, made by model 2 is unfortunately outside the original melodies scale, D major, where it should have been F♯. Most melodies are in the G, D or A major scale, where a pitch like B♭ and F rarely appears only around 600 times, so the data set is highly biased towards some notes and a model would probably not be able to reconstruct melodies using scales outside the standard ones used in Irish folk music, like e.g. jazz standards.

Predicting durations should be much easier for the models, as the melodies mainly consists of 1/4 and 1/8 (a histogram for these is included in Appendix c). They seem to be produced with the same frequency as in the original melody, but mostly misplaced. This is a mere artefact of the feature formatting itself, as one wrong prediction of a duration will shift the whole melody and not align with the original melody. This could be solved by using continuous relative positions instead of duration classes.

Looking at the reconstructed melody in Figure 4, and by ignoring the duration shifts in the reconstructions, some rhythmical motifs becomes clear. Some of the jumps between durations of 1/4 (notes with no flag) and 1/8 (notes with one flag) are reconstructed correctly by model 2 in bar 3, where the sequence is (1/4, 1/8, 1/4, 1/8).

6. CONCLUSION

We have developed two new models to reconstruct melodies, maintaining important musical structures like scale, conjunct motion and rhythmical motifs. The best version of these models have accuracies of around 37 % for predicting pitches and of around 75 % for predicting durations. These versions use regularisation methods like dropout and L_2-penalty, and feeding in the previous output into the RNN for better error propagation, to enhance the generalisation performance.

There are several ways to possibly achieve better results. Instead of one-hot encoding the pitches and durations, we could instead use continuous encoding. The model would then be able to learn the differences in pitch, instead of learning where to go for each certain pitch. This would also make the model better at learning the scale of the tune and predicting consistent intervals. Chords could also be included as Zimmerman [2] did, since depending on which chord is playing, some pitches would be more probable than others. Another

thing to include are the positions in the each measure of the melody. This would essentially be a sum of the durations, and it would help with the rhythm, especially since some notes ought to fall on certain beats. Lastly, backwards conditioning the note on the following notes through a backwards GRU layer could also aid the model prediction.

On a longer timescale, it would be interesting to extend the model to a semi-supervised variational auto-encoder and include the collections in the data set to separate genres from the melodies. The musical notation could also be combined with sound (as with speech and text) to be able to synthesise different instruments playing.

7. ACKNOWLEDGEMENTS

We want to thank Ole Winther for supervising and for the lectures as well as Lars Maaløe for help and code snippets.

8. REFERENCES

[1] E. Choi, M. Taha Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor AI: Predicting Clinical Events via Recurrent Neural Networks," *ArXiv e-prints*, Nov. 2015.

[2] Y. Zimmerman, "A Dual Classification Approach to Music Language Modeling." http://yoavz.com/music_rnn_paper.pdf, Mar. 2016.

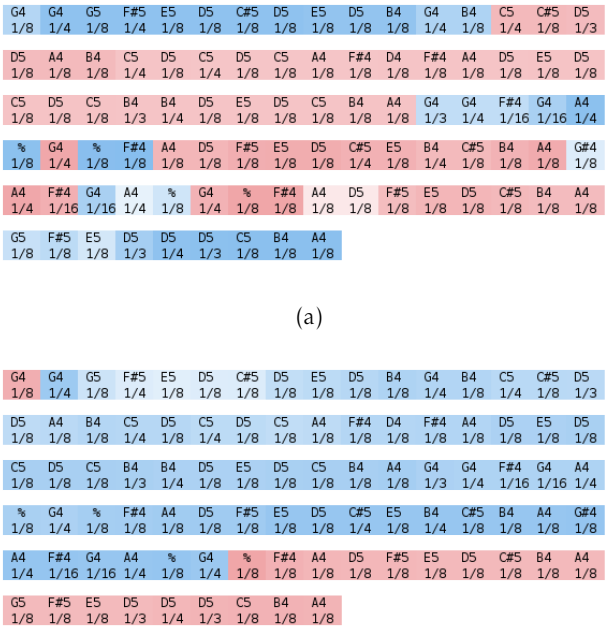
[3] J. Allwright, "ABC Version of the Nottingham Music Database." http://abc.sourceforge.net/NMD/, 2003.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *ArXiv e-prints*, Dec. 2014.

[5] C. Walshaw, "ABC Notation Home Page." http://abcnotation.com.

[6] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ArXiv e-prints*, Dec. 2014.

A. GRU ACTIVATIONS

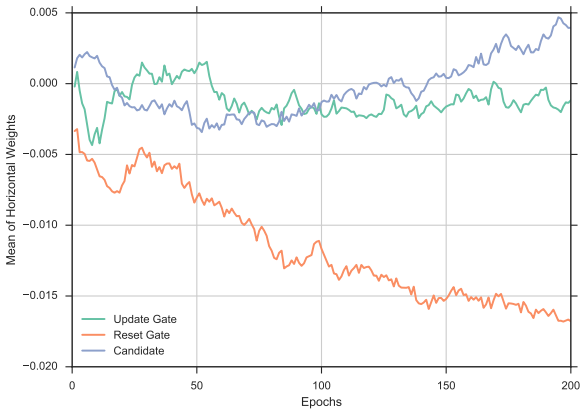


(a)

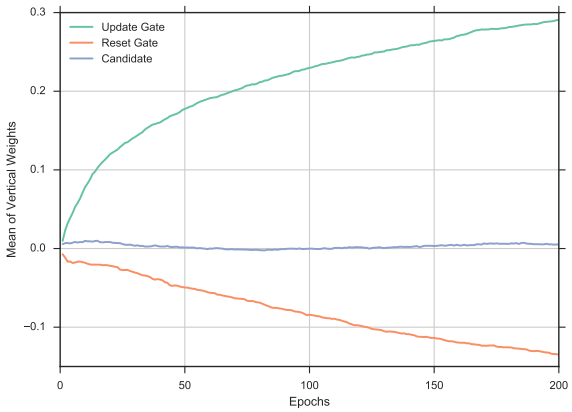
(b)

Figure 6 GRU Activations, $h_k^{(t)}$, for (a) unit $k = 26$, which shows a higher response to the combination of the pitch G and F#, and (b) unit $k = 50$, which shows a higher response in the beginning and end of the melody.)

B. GRU WEIGHTS

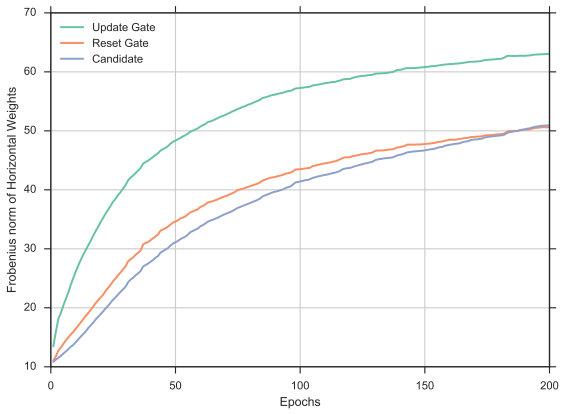


(a)

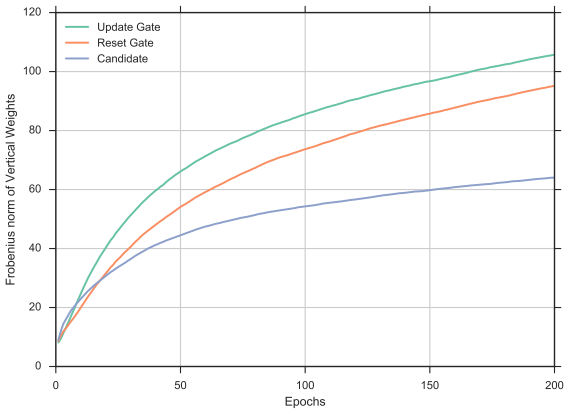


(b)

Figure 7 Mean over the three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.

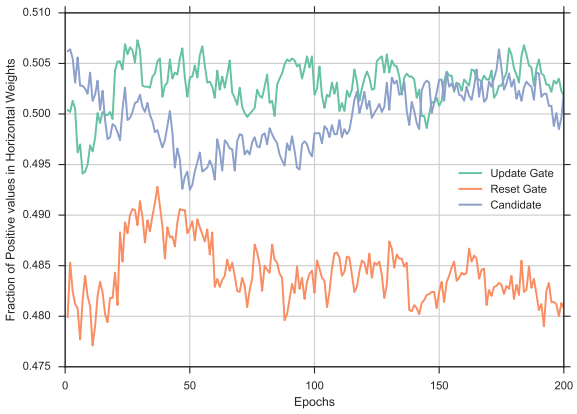


(a)

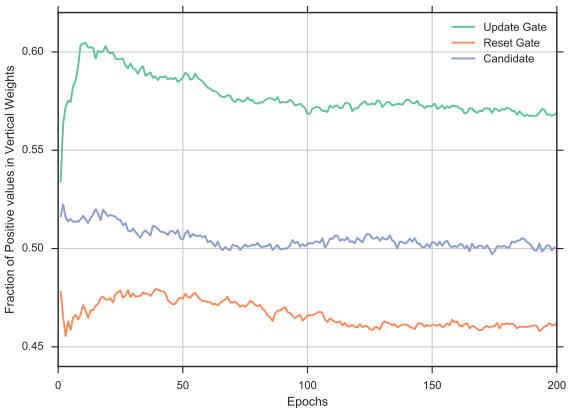


(b)

Figure 8 Frobenius norm over the three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.

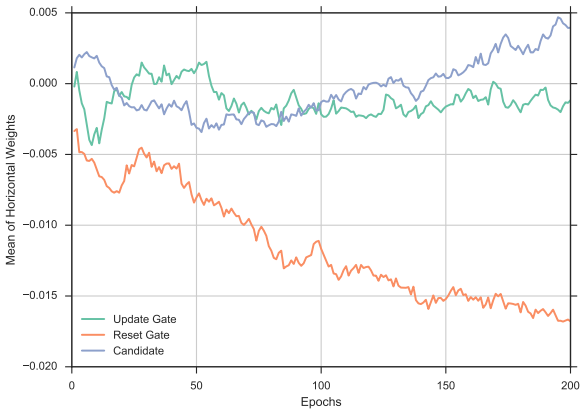


(a)

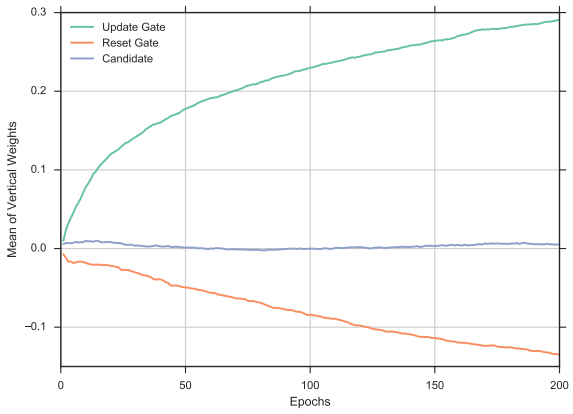


(b)

Figure 9 Fraction of positive values in three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.

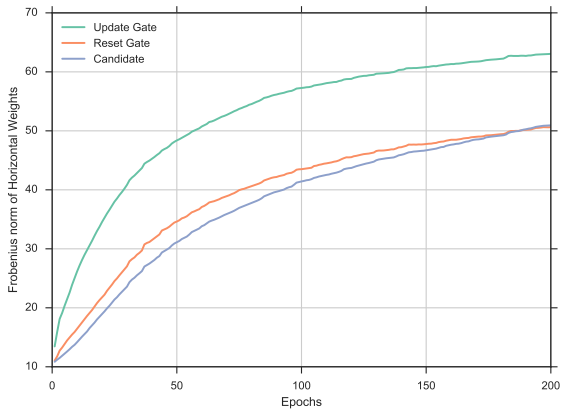


(a)

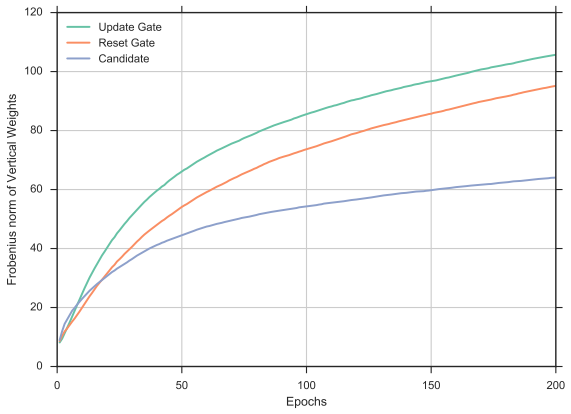


(b)

Figure 10 Mean over the three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.

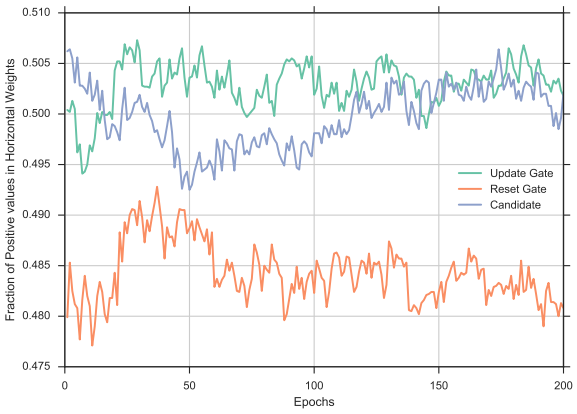


(a)

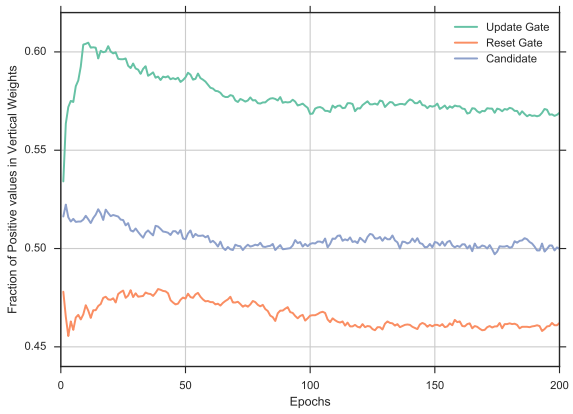


(b)

Figure 11 Frobenius norm over the three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.



(a)



(b)

Figure 12 Fraction of positive values in three kinds of GRU weights (Update, Reset and Candidate), for (a) horizontal weights $\mathbf{W}_h$ and (b) vertical weights $\mathbf{W}_x$ recorded after each epoch.

C. HISTOGRAM OVER NOTE DURATIONS

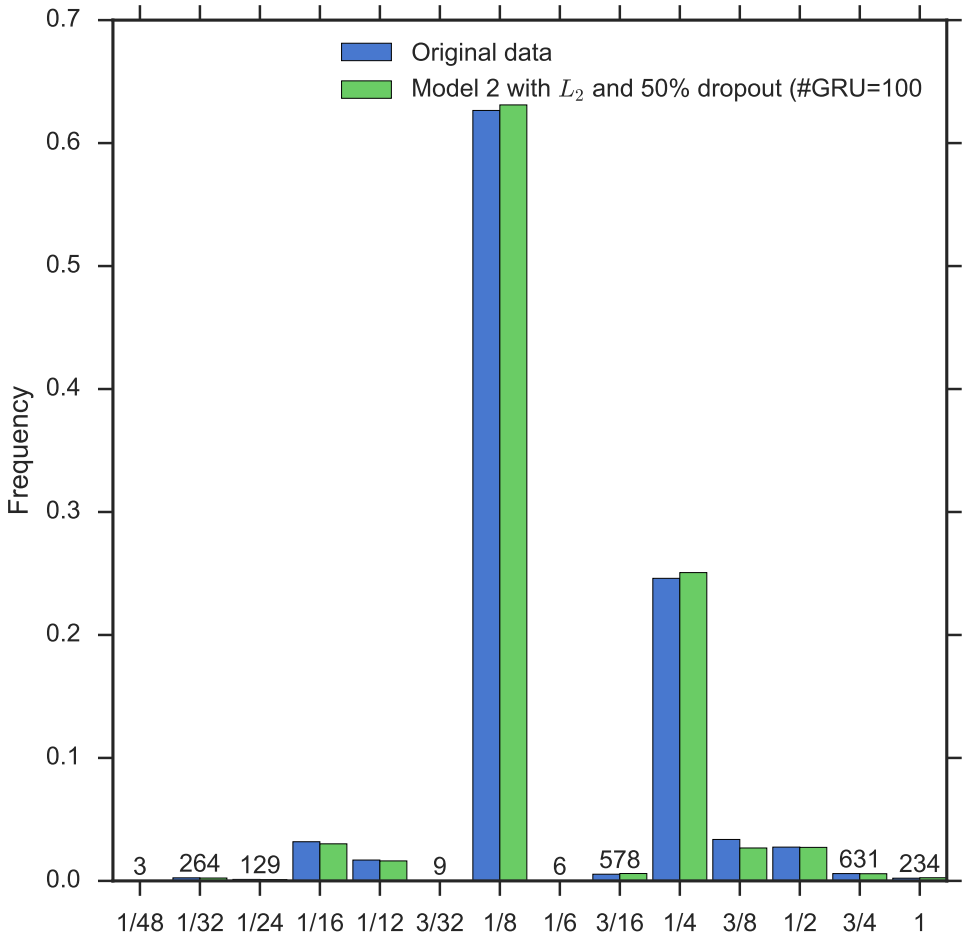


Figure 13 Histograms showing statistical frequency of duration classes in the (blue) original data and in the (green) reconstructions produced by model type 2 with dropout of 50% and L_2-regularization.