What do these 5,599,881 parameters mean? An analysis of a specific LSTM music transcription model, starting with the 70,281 parameters of its softmax layer

Bob L. Sturm
Centre for Digital Music, Queen Mary University of London UK

Abstract

A *folk-rnn* model is a long short-term memory network (LSTM) that generates music transcriptions. We have evaluated these models in a variety of ways – from statistical analyses of generated transcriptions, to their use in music practice – but have yet to understand how their behaviours precipitate from their parameters. This knowledge is essential for improving such models, calibrating them, and broadening their applicability. In this paper, we analyse the parameters of the softmax output layer of a specific model realisation. We discover some key aspects of the model's local and global behaviours, for instance, that its ability to construct a melody is highly reliant on a few symbols. We also derive a way to adjust the output of the last hidden layer of the model to attenuate its probability of producing specific outputs.

Introduction

The *folk-rnn* software[1] builds a long short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997) that models music transcription sequences expressed in a vocabulary derived from "ABC notation" (Sturm et al. 2016).[2] This notation finds much use in the on-line sharing of folk tunes, e.g., the website `http://thesession.org`. We have trained several *folk-rnn* models on a dataset extracted from that website, and have evaluated them in a variety of ways (Sturm and Ben-Tal 2017). However, what we have yet to understand is *how* the behaviours of a *folk-rnn* model precipitate from its parameters.

Consider the following transcription generated by the *folk-rnn* model "v2" (Sturm et al. 2016), notated in Fig. 1:

```
<s> M:6/8 K:Cmaj |: C D E G, 2 A, | C D E A,
2 G, | C D E G, 2 G, | C 3 A, 3 | C D E G, 2
A, | C D E G 3 | A A G A 2 B | c 3 C 3 :|
|: C D E G 2 A | G 3 E 2 D | C D E C 2 D |
E C A, G, 2 G, | C D E G 2 A | G 3 A 3 |
G C 2 D E D | C 3 C 3 :| </s>
```

In this transcription, each space separates a token produced by the model. The first and last tokens mark the beginning and end of a transcription. The second and third tokens denote meter and mode, respectively. These are followed by a repeat token, and then pitch, duration, and measure tokens.

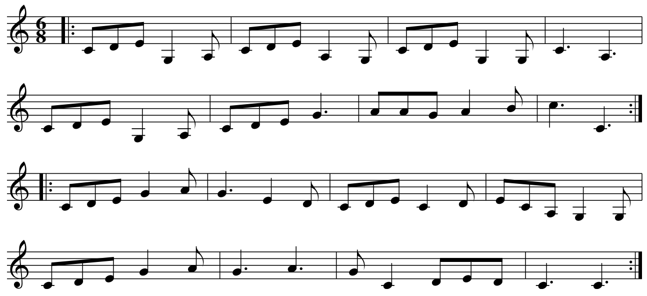
Figure 1: A transcription generated by folk-rnn model v2. How do the 5,599,881 numerical parameters of v2 relate to the local and global characteristics of this transcription?

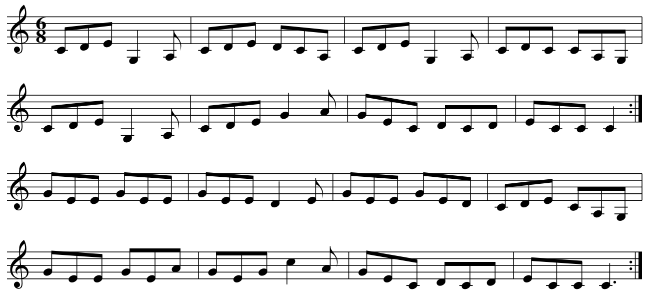
Figure 2: Transcription "Páidín Ó Raifeartaigh" from the training data of folk-rnn model v2. (This also appears in the training data as "The Quaker's Wife". The counting mistake in last bar of the A part is in the original.) Compare with the transcription generated by v2 in Fig. 1.

Many local and global characteristics of this transcription are common in the training transcriptions. It has an AABB form, with 8 correctly-counted measures in each part. Both parts start and end on the tonic, have a cadence, and share a strong resemblance by repetition and variation of a melodic idea. The melody features step-wise motion, arpeggiation, and rhythmic consistency. There is also an ambiguity between C major and its relative minor. Has v2 merely reproduced something from its training material? Figure 2 shows

[1]Available here: https://github.com/IraKorshunova/folk-rnn
[2]http://abcnotation.com/

the closest transcription we find among the 23,636 transcriptions in the training data. This is clearly a different tune. So how do the local and global characteristics we see in Fig. 1 precipitate from the 5,599,881 parameters of v2?

The analysis of RNN sequence models, in terms of explaining the roles of their parameters (and hyperparameters) for the sequences they are modelling, rarely appears in the literature. Researchers often build models using a range of hyperparameters (e.g., architecture and training decisions) and compare them according to how well each fits validation sequences. Many researchers have applied LSTM to music sequences, e.g., Franklin (2006); Eck and Lapamle (2008); Chung et al. (2014); Greff et al. (2016); Colombo, Seeholzer, and Gerstner (2017), but they do not analyse the parameters of the resulting models. This is in contrast to work in computer vision that aims to understand content identification systems in terms of the functions of their elements, e.g., specific groups neurons that activate for dog faces or flowers (Zeiler and Fergus 2013; Szegedy et al. 2014; Yosinski et al. 2015).

One notable exception in research applying LSTM to sequence modelling is that of Karpathy, Johnson, and Li (2015). They inspect the internal dynamics of LSTM models of written text (the novel "War and Peace" and the Linux kernel) and find some interpretable components, e.g., gates that turn on and off with open and closed quotes or "if" statements. They find that many of the components in the models are not so clearly interpretable. They do not, however, analyse the parameters themselves.

In this paper, we analyse the parameters of the softmax layer of v2 using multivariate analysis. This helps us relate its parameters to local and global characteristics of its output.[3] How is v2 generating music transcription sequences that by and large appear correctly counted? Can we isolate in its parameters its latent "notions" of time, repetition, variation, melodic contour, structure, and cadence? How can we modify its parameters to encourage or discourage particular behaviours without retraining, e.g., doubling the likelihood of it producing specific output? Such knowledge is essential to improve such models, calibrate them for particular users, and broaden their applicability.

Overview of the folk-rnn model v2

Architecture and parameters

The folk-rnn model v2 consists of three LSTM layers (Hochreiter and Schmidhuber 1997) of 512 units each, and an output softmax layer (choices made for no particular reason). Its input and output dimensions relate to an indexed vocabulary (Sturm et al. 2016), $\mathcal{V}$, which consists of $|\mathcal{V}| = 137$ *tokens* of seven types: transcription (2); meter (7); mode (4); measure (5); pitch (85); grouping (9); and duration (25). Examples of these tokens can be seen in the transcription of the previous section.

v2 operates in the following way. Let s be a sequence of elements of $\mathcal{V}$, and let $s(t)$ be the tth one. v2 transforms $s(t)$ through a sequence of non-linear operations: $\mathcal{V} \rightarrow \{0,1\}^{137} \rightarrow [-1,1]^{512} \rightarrow [-1,1]^{512} \rightarrow [-1,1]^{512} \rightarrow \mathcal{V}$. After encoding $s(t)$ as a binary (one-hot) vector, $\mathbf{x}_t$, the first LSTM layer transforms it to a vector $\mathbf{h}_t^{(1)} \in [-1,1]^{512}$. The second LSTM layer maps $\mathbf{h}_t^{(1)}$ to a vector $\mathbf{h}_t^{(2)} \in [-1,1]^{512}$. The third LSTM layer maps $\mathbf{h}_t^{(2)}$ to a vector $\mathbf{h}_t^{(3)} \in [-1,1]^{512}$. And the softmax layer "decodes" $\mathbf{h}_t^{(3)}$ to produce a categorical probability distribution over $\mathcal{V}$. The model samples from that distribution to produce $s(t+1)$.

Denote the input to the ith LSTM layer of v2 by $\mathbf{y}_t^{(i)}$. This layer transforms $\mathbf{y}_t^{(i)}$ by (Graves 2013):

$$\mathbf{i}_t^{(i)} \leftarrow g(\mathbf{W}_{xi}^{(i)}\mathbf{y}_t^{(i)} + \mathbf{W}_{hi}^{(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b}_i^{(i)}) \tag{1}$$

$$\mathbf{f}_t^{(i)} \leftarrow g(\mathbf{W}_{xf}^{(i)}\mathbf{y}_t^{(i)} + \mathbf{W}_{hf}^{(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b}_f^{(i)}) \tag{2}$$

$$\mathbf{o}_t^{(i)} \leftarrow g(\mathbf{W}_{xo}^{(i)}\mathbf{y}_t^{(i)} + \mathbf{W}_{ho}^{(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b}_o^{(i)}) \tag{3}$$

$$\begin{aligned}\mathbf{c}_t^{(i)} \leftarrow \tanh(\mathbf{W}_{xc}^{(i)}\mathbf{y}_t^{(i)} + \mathbf{W}_{hc}^{(i)}\mathbf{h}_{t-1}^{(i)} + \mathbf{b}_c^{(i)}) \odot \mathbf{i}_t^{(i)} \\ + \mathbf{f}_t^{(i)} \odot \mathbf{c}_{t-1}^{(i)}\end{aligned} \tag{4}$$

$$\mathbf{h}_t^{(i)} \leftarrow \tanh(\mathbf{c}_t^{(i)}) \odot \mathbf{o}_t^{(i)}. \tag{5}$$

where g is the sigmoid function applied element-wise to its argument, and $\odot$ is an element-wise multiplication. $\mathbf{i}_t^{(i)}$ is the output of the *input gates*; $\mathbf{f}_t^{(i)}$ is the output of the *forget gates*; $\mathbf{o}_t^{(i)}$ is the output of the *output gates*; $\mathbf{c}_t^{(i)}$ is the output of the *cell gates*; and $\mathbf{h}_t^{(i)}$ is the *hidden state* of the layer. The size of each matrix and bias vector of a layer are commensurate with the following connections between layers: $\mathbf{y}_t^{(1)} \leftarrow \mathbf{x}_t \in \{0,1\}^{137}$; $\mathbf{y}_t^{(2)} \leftarrow \mathbf{h}_t^{(1)} \in [-1,1]^{512}$; and $\mathbf{y}_t^{(3)} \leftarrow \mathbf{h}_t^{(2)} \in [-1,1]^{512}$.

v2 produces $s(t+1)$ by sampling from $\mathcal{V}$ according to a categorical distribution with parameters

$$\mathbf{p}_t = \text{softmax}\left(\frac{1}{T_s}\left[\mathbf{W}_s\mathbf{h}_t^{(3)} + \mathbf{b}_s\right]\right) \tag{6}$$

where $T_s \in \mathbb{R}_+$ is a user-specified *sampling temperature* setting. (In this work, $T_s \leftarrow 1$.) The dimensions of $\mathbf{p}_t \in [0,1]^{137}$ are ordered corresponding to $\mathcal{V}$. For example the 87th dimension, $[\mathbf{p}_t]_{87}$, corresponds to the token "K:Cmaj", which denotes the C major mode.

Training

In total, v2 has 5,599,881 numerical parameters: 8 matrices and 6 vectors in each LSTM layer, and a matrix and vector in the softmax layer. These parameters are found via training on a dataset, which aims to optimise a cost function that describes how well the model fits the dataset. The cost function for v2 is the negative mean log probability, or *sequence loss*:

$$L(s) := -\frac{1}{|s|}\sum_{t=1}^{|s|} \log[\mathbf{p}_t]_{s(t)} \tag{7}$$

for a transcription s. This shows that training v2 attempts to maximise each "ground truth" dimension of $\mathbf{p}_t$ computed by the model over the sequence.

[3] 30,000 transcriptions generated by v2 can be found here: https://highnoongmt.wordpress.com/2018/01/05/volumes-1-20-of-folk-rnn-v1-transcriptions/

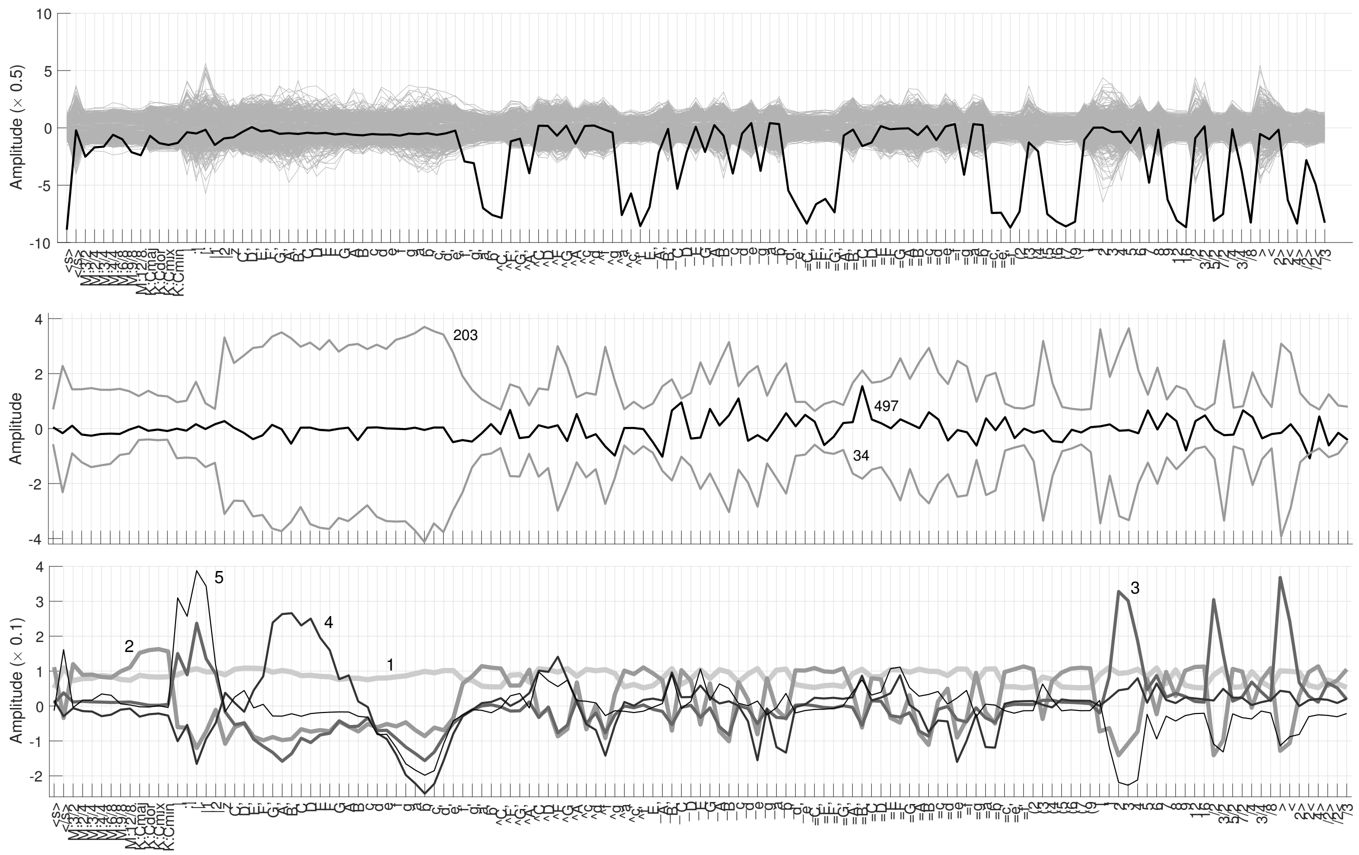
Figure 4: Top: Parameters $\mathbf{W}_s$ (columns, grey) and $\mathbf{b}_s$ (black) of the softmax layer (6). Middle: Three columns of $\mathbf{W}_s$ (numbered). Bottom: The five left singular vectors of $\mathbf{W}_s$ with the largest singular values. All plotted relative to the 137 transcription tokens of $\mathcal{V}$ ordered by type: transcription (2); meter (7); mode (4); measure (5); pitch (85); grouping (9); and duration (25).

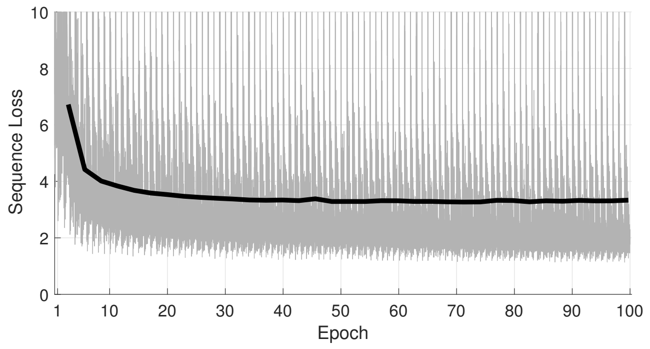

Figure 3: The sequence loss of the folk-rnn model v2 decays as its training proceeds. The black line is loss in the validation dataset. The grey line is loss in the training set. The erratic behaviour of the training loss comes from the minibatch training procedure: the model parameters are tuned on a new batch of 64 transcriptions every 351 iterations.

We train v2 using data from http://thesession.org. After much cleaning, we transpose all transcriptions to have a root of C. We then parse the transcriptions into tokens. The dataset contains 23,636 transcriptions, with a total of 4,032,490 tokens. The shortest transcription sequence is 46 tokens (titled, "The Ballintore Fancy"); the longest is 1952 tokens (titled, "The Mason's Apron"). The median transcription length is 150 tokens. More information on the training data is provided by Sturm and Ben-Tal (2017).

We train v2 using stochastic gradient descent with minibatches of 64 transcriptions selected randomly from 95% of the training transcriptions. We validate the model every 1000 iterations on the remainder. One *epoch* is defined as 351 steps, after which point the model has seen all training transcriptions. We use dropout at a rate of 0.5 between all layers. In adjusting the parameters, we use gradient clipping (to 5), a learning rate of 0.003, and a decay of 0.97 after 20 epochs. We do not truncate the number of steps for back-propagating the error. Figure 3 shows how the loss (7) decays for both the training and validation sets. If the model learns nothing, then $\mathbf{p}_t$ is iid uniform in the best case. For the v2 vocabulary this would be $L(s) = 4.92$. Hence, the training procedure appears to be working. v2 is the model after 100 training epochs, which we analyse below.

Analysis of the softmax layer

Since the rows of $\mathbf{p}_t$ (6) relate directly to $\mathcal{V}$, so do the rows of $\mathbf{W}_s \in \mathbb{R}^{|\mathcal{V}|\times 512}$ and $\mathbf{b}_s \in \mathbb{R}^{|\mathcal{V}|}$. The hidden state vector of the third LSTM layer (L3), $\mathbf{h}_t^{(3)}$, is the input to the softmax layer, and so we see that layer directly controls how the columns of $\mathbf{W}_s$ combine to displace the bias $\mathbf{b}_s$.

Figure 4(top) shows the "shapes" of the parameters in $\mathbf{W}_s$ and $\mathbf{b}_s$ relative to $\mathcal{V}$. This shows $\mathbf{b}_s$ has small values for the tokens: `<s>, =f', 16, (7, ^f', =C,`. We find that these tokens are rarely generated by v2, and in fact they are themselves rare in the training data (with the exception of `<s>`, which is the "start transcription" token used to initiate generation).

Figure 4(middle) shows three columns of $\mathbf{W}_s$. These are controlled by L3 units 34, 203 and 497. We see that 497 pushes up probability mass for tokens `=B,`, `_C` and `_c` and pushes down probability mass for tokens `_A,` and `^g` (or vice versa). This suggests that in some instances v2 is treating enharmonic pitch classes in similar ways. We also see that L3 units 34 and 203 are controlling columns of $\mathbf{W}_s$ that appear quite similar, save polarity. If these two units both saturate in the positive direction, these two columns of $\mathbf{W}_s$ will add to increase the probability mass of the four modes. We find several such instances in $\mathbf{W}_s$. This shows it is important to interpret the columns of $\mathbf{W}_s$, and likewise the units of L3, as groups contributing to the distribution over $\mathcal{V}$. This motivates applying singular value decomposition.

Singular value decomposition of $\mathbf{W}_s$

The singular value decomposition (SVD) of $\mathbf{W}_s$ shows how the hidden state of L3 affects the probability distribution at the softmax layer (6). The SVD of $\mathbf{W}_s$ expresses it as a sum of rank-1 orthogonal projection matrices:

$$\mathbf{W}_s = \sum_{j=1}^{|\mathcal{V}|} \sigma_j \mathbf{u}_j \mathbf{v}_j^T \tag{8}$$

where $\{\mathbf{u}_j \in \mathbb{R}^{|\mathcal{V}|}\}$ are the "left singular vectors" (lsv), $\{\mathbf{v}_j \in \mathbb{R}^{512}\}$ are the "right singular vectors" (rsv), and $\{\sigma_j\}$ are the singular values ordered such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{|\mathcal{V}|} > 0$. Since all singular vectors are orthonormal, then

$$\mathbf{W}_s \mathbf{v}_j = \sigma_i \mathbf{u}_j. \tag{9}$$

We see from this how $\mathbf{h}_t^{(3)}$ can point along $\mathbf{v}_j$ to displace the probability distribution along $\mathbf{u}_j$.

Figure 5 shows the singular values of $\mathbf{W}_s$. We find the largest singular value to be more than 7.5 times all the others. Figure 4(bottom) shows the five lsv with the greatest singular values. An lsv amplifies the probability of tokens for which it is positive, and attenuates the probability of tokens for which it is negative (or vice versa since the range of the hidden state of L3 is $[-1, 1]$). Here we see the first lsv increases the probability mass of all tokens, with the two largest changes being for the end transcription token `</s>` and the end chord token `]`. The second lsv amplifies the probability of the four mode tokens, and attenuates the probability of three octaves of pitch tokens. The third lsv amplifies the probability of several measure tokens, as well as particular duration tokens (e.g., `2`, `/2` and `>`), and attenuates the probability of the same three octaves of pitch tokens as the second left singular vector. The fourth lsv amplifies

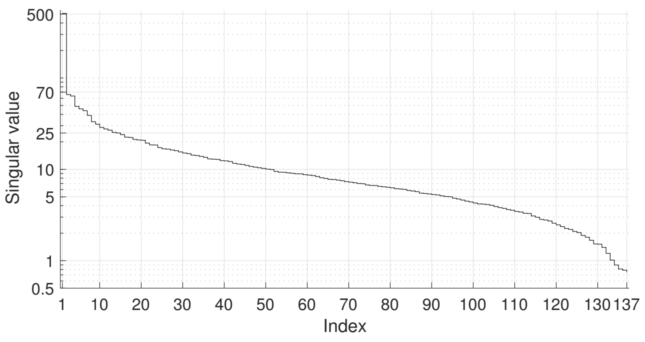

Figure 5: The singular values of $\mathbf{W}_s$.

some of the low-pitched tokens. The fifth lsv greatly amplifies the measure tokens and attenuates specific duration tokens.

Projections of layer 3 hidden state sequences

Returning to the transcription produced by v2 in Fig. 1, we can now see how its L3 hidden state sequence $\{\mathbf{h}_t^{(3)}\}$ points along the orthogonal basis defined by the rsv of $\mathbf{W}_s$. Figure 6(top) shows the hidden states of 200 L3 units over the generated tokens. We see a variety of behaviours of the units: some saturate after each measure token is generated; some activate just before a repeat measure token appears; many have periodicities of about 6-7 steps, which is about the period of the measure tokens; and a few units stay on for the majority of the transcription. We also see that the hidden states of L3 are sparse.

Figure 6(bottom) shows the magnitude projections of $\{\mathbf{h}_t^{(3)}\}$ onto the 30 rsv of $\mathbf{W}_s$ that have the greatest singular values ($\sigma_j > 15$, Fig. 5). We see for this transcription that every hidden state vector points in large part along the first rsv. The second-step hidden state vector, $\mathbf{h}_2^{(3)}$ produces the mode token `K:Cmaj`, and is the only one to point in large part along rsv 9 and 11. We see the hidden state vector points along rsv 5 and 15 in the steps where v2 produces a measure token `|` or `:|` — which is no surprise given how much lsv 5 points along measure tokens in Fig. 4(bottom).

To explore the meaning of rsv 5 and 15 for measure tokens, we have v2 generate 10 different transcriptions (including that in Fig. 1), and project all L3 hidden state vectors onto these two rsv. Figure 7 shows in this subspace measure tokens appear concentrated in a region separate from the others. We now explore the significance of this subspace of the column space of $\mathbf{W}_s$.

Shrinking the column space of $\mathbf{W}_s$

We find that $\mathbf{W}_s \in \mathbb{R}^{137 \times 512}$ is full rank, which means its column space is $\mathbb{R}^{137}$. We can shrink the dimension of its column space by truncating the sum in (8). If we run v2 with the same random seed as for Fig. 1, but approximating $\mathbf{W}_s$ using only the 30 pairs of singular vectors having the largest singular values, it produces the transcription shown in Fig. 8. Compared with that in Fig. 1, we see the transcription still has an AABB form, with 8 bars in each part correctly counted. There is now a second ending for A. Both parts start

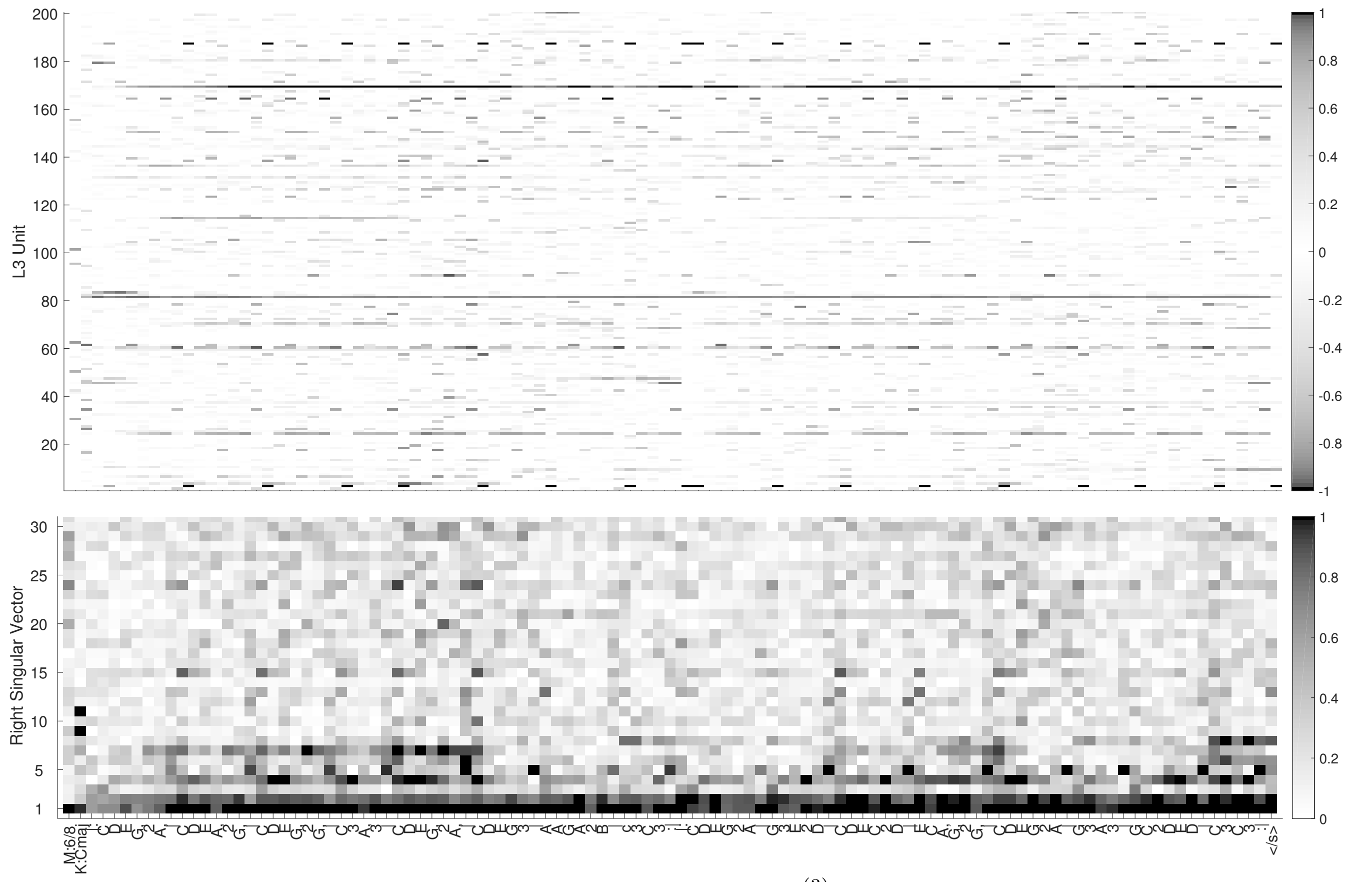
Figure 6: Top: the sequence of layer three LSTM hidden state vectors $\{\mathbf{h}_t^{(3)}\}$ for the transcription generation shown in Fig. 1 (only 200 of 512 units shown). Bottom: magnitude projections of $\{\mathbf{h}_t^{(3)}\}$ onto the 30 right singular vectors of $\mathbf{W}_s$ with the largest singular values. Generated transcription tokens shown along x-axis.

and end on the tonic with a cadence. We still see repetition and variation, but it seems more varied in this transcription. Unlike for the transcription in Fig. 1, there is not a strong resemblance between the parts here. It seems then that many of these behaviours of v2 are not affected by excising this 107-dimensional subspace from the column space of $\mathbf{W}_s$.

If we instead reduce the dimension of the column space of $\mathbf{W}_s$ by only one, setting $\mathbf{W}_s \leftarrow \mathbf{W}_s - \sigma_5 \mathbf{u}_5 \mathbf{v}_5^T$, then the modified model generates a transcription that we must end manually after thousands of steps. Figure 9 shows the beginning of this transcription. While we do see stepwise motion, nothing is counted correctly, and the melody lacks direction. Hence it seems this slight modification to the column space of $\mathbf{W}_s$ has severely affected global behaviours of the model.

These interventions on $\mathbf{W}_s$ show that the L3 layer of v2 in the case of Fig. 1 is working by and large in a subspace of $\mathbb{R}^{512}$ that has a dimension far less than 137. The model seems to rely heavily on one specific direction, $\mathbf{u}_5$, for not only generating measure tokens at the correct positions, but building melodies, repetition and variation to create parts, and ending a transcription. This suggests that the "counting" ability of v2 is not so easily isolated from its other abilities, i.e., it is not clear if we can preserve its "sense" of melody while excising its ability to correctly output measure tokens. The local behaviours of the model, e.g., stepwise motion,

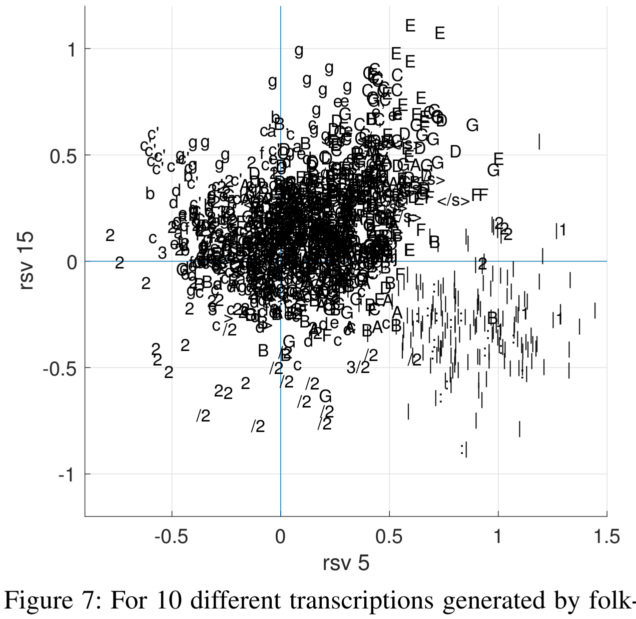
Figure 7: For 10 different transcriptions generated by folk-rnn model v2, including that in Fig. 1, the projection of each $\{\mathbf{h}_t^{(3)}\}$ onto right singular vectors (rsv) 5 and 15 of $\mathbf{W}_s$.

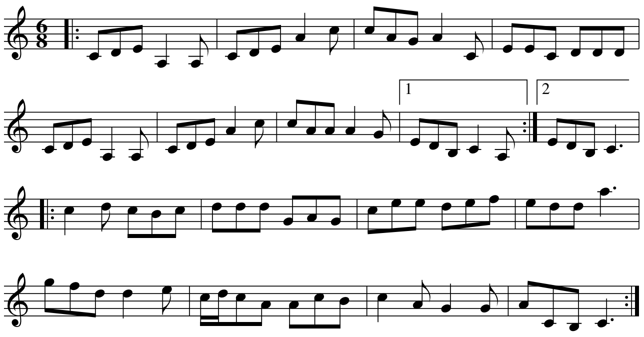

Figure 8: Transcription generated by folk-rnn model v2 with the same random seed as Fig. 1, but using a softmax matrix approximated with the first 30 singular vectors of $\mathbf{W}_s$ (6).

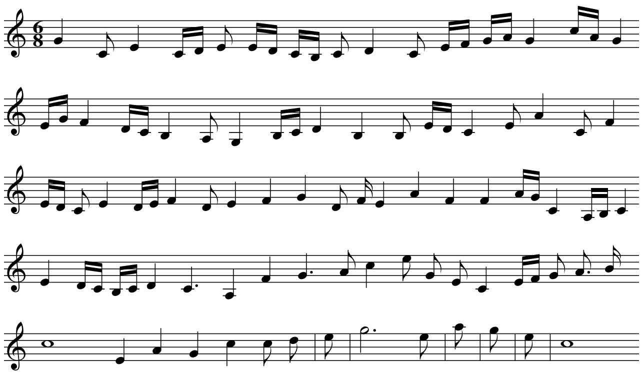

Figure 9: The beginning of the transcription generated by folk-rnn model v2 with the same random seed as Fig. 1, but with the softmax matrix $\mathbf{W}_s - \sigma_5 \mathbf{u}_5 \mathbf{v}_5^T$ (6).

do not seem to be so severely affected by this "lobotomy", however.

Melodic "ability" of v2 relies on measure tokens

To test the hypothesis that v2 relies heavily on measure tokens for building melodies, we suppress its sampling of measure tokens at the softmax layer. Starting with the same random initialisation as Fig. 1, we force v2 to sample repeatedly until it produces a token in each step that is not `|`. Figure 10 shows the resulting transcription, and Fig. 11(top) shows the probability of sampling `|` in each step. Compared with Fig. 1, we see the first measure is the same up to the A semiquaver. At this point, the model samples `|` but is forced to sample again. It eventually produces the token `/2`, and then produces another semiquaver note, and then tries again to sample `|`. When forced to resample, it produces the first-ending measure token `|1`. The system finishes the transcription after 9 bars, with each correctly counted save the first. There does appear to be some repetition and variation of the rhythm of the first six notes, and the melody does feature stepwise motion and some arpeggiation. The fourth bar features a V-I cadence, with a leading tone pickup into the next bar, but otherwise after the 4th bar the melody is wandering. Figure 11(top) shows that the model never loses interest in producing the measure line token throughout the generation.

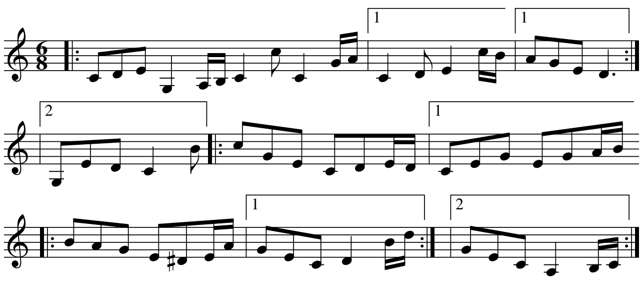

Figure 10: The transcription generated by folk-rnn model v2 with the same random seed as Fig. 1, but suppressing its output of measure token `|`. Figure 11(top) shows the probability of `|`, and the tokens that were sampled instead.

Figure 11(bottom) and Fig. 12 show the results when we suppress the sampling of any measure token. After the suppression of the start repeat token `|:` the transcription differs from Fig. 1 and 10. We see stepwise motion and arpeggiation in the melody, some repetition and variation of the rhythm, and something that sounds like a place for a repeat symbol after 8 bars, but otherwise the melody is just wandering as in Fig. 10. Figure 11(bottom) shows that the model gradually loses interest in sampling measure tokens until step 186, when it tries to sample `|` with a probability mass of 0.9.

From these experiments then it appears that the "ability" of v2 to build melodies having the long term structure we see in the training data — which is apparent in Fig. 1 and 8 — heavily depends on measure tokens.

Displacement at the softmax layer

Since the values are additive inside the softmax (6), we can derive what they mean in terms of the change in probability for any specific token. For the nth token with pre-softmax value $[\mathbf{w}]_n$ displaced by some $\delta \in \mathbb{R}$, we want to find α in the following:

$$\begin{aligned}[\mathbf{p}']_n &= [\text{softmax}(\mathbf{w} + \delta \mathbf{e}_n)]_n \\ &= \exp([\mathbf{w}]_n + \delta) / \sum_m \exp[\mathbf{w} + \delta \mathbf{e}_n]_m = \alpha [\mathbf{p}]_n \qquad (10)\end{aligned}$$

where $\mathbf{e}_n$ is the nth standard basis vector of $\mathbb{R}^{|\mathcal{V}|}$, the exponentiation is applied element-wise, and

$$[\mathbf{p}]_n = [\text{softmax}(\mathbf{w})]_n = \exp[\mathbf{w}]_n / \sum_m \exp[\mathbf{w}]_m. \qquad (11)$$

Using this we can rewrite the denominator of (10):

$$\begin{aligned}\sum_m \exp[\mathbf{w} + \delta \mathbf{e}_n]_m &= \left[\sum_m \exp[\mathbf{w}]_m\right] - \exp[\mathbf{w}]_n \\ &\quad + \exp([\mathbf{w}]_n + \delta) \\ &= \exp[\mathbf{w}]_n \frac{1 - [\mathbf{p}]_n}{[\mathbf{p}]_n} + \exp([\mathbf{w}]_n + \delta). \qquad (12)\end{aligned}$$

Substitution of this into (10) and solving for α finally gives

$$\alpha = \frac{1}{[\mathbf{p}]_n + e^{-\delta}(1 - [\mathbf{p}]_n)}, \delta \in \mathbb{R}. \qquad (13)$$

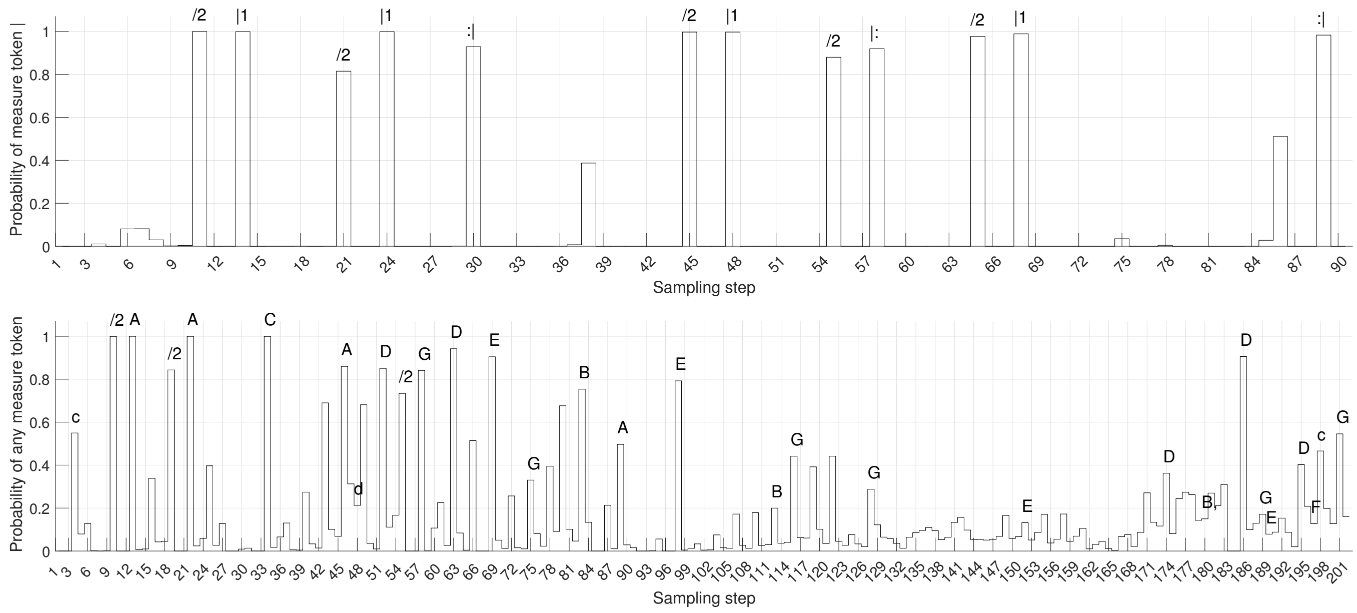

Figure 11: Top: Probability of folk-rnn model v2 sampling the measure token `|` during the generation of the transcription in Fig. 10 (with the same random seed as Fig. 1). Bottom: Probability of folk-rnn model v2 sampling any measure token during the generation of the transcription in Fig. 12 (with the same random seed as Fig. 1). In each we show the tokens sampled in place of the suppressed ones.

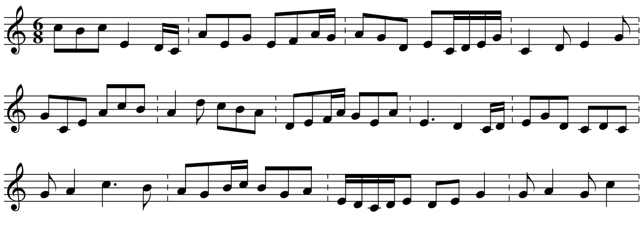
Figure 12: The first half of the transcription generated by folk-rnn model v2 with the same random seed as Fig. 1, but suppressing its output of any measure token. Dashed bar lines added manually to assist with grouping according to the meter. Figure 11(bottom) shows the probability of any measure token in this generation, and the tokens that were sampled instead.

Rearranging this expression, we can find what displacement δ is necessary to amplify or attenuate the probability of the nth token by a factor α:

$$\delta = \log \frac{1 - [\mathbf{p}]_n}{\frac{1}{\alpha} - [\mathbf{p}]_n}, 0 \leq \alpha \leq 1/[\mathbf{p}]_n \tag{14}$$

with the limits given by the axioms of probability. The sampling temperature T_s is implicit in these relationships, i.e., the computation of $\mathbf{p}$ in (6).

Applying the above to the hidden state of L3 unit 497, we can see from Fig. 4(middle) that if it were to saturate in the positive direction ($+1$), then, all other things remaining equal, it would almost quadruple ($\alpha = 3.94$) the probability mass of token index 130 (`=B,` and $\delta = 1.54$) if it were initially about 5% probable ($[\mathbf{p}]_{130} = 0.05$). If this same unit output -1, then, all other things remaining equal, it would decrease the probability of `=B,` by a factor of about 1/4 if it were about 5% probable. Figure 13 shows the transcription generated by v2 using the same random seed as Fig. 1, but with an attenuation of the probability of the token `A,` (dimension 106) by a factor $\alpha = 0.01$. We see the two transcriptions share many of the same features, but this one does not go to the relative minor.

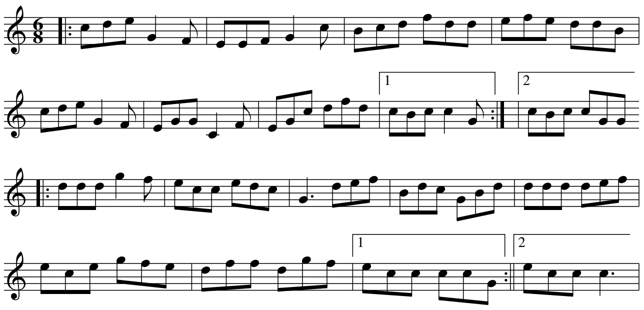

Figure 13: The transcription generated by folk-rnn model v2 with the same random seed as Fig. 1, but attenuating the softmax probability of token `A,` (dimension 106) by the factor $\alpha = 0.01$ (transposed up one octave for readability).

We can also find how to deviate the hidden state of L3 to produce a displacement of just the nth token in the softmax layer. We want to find $\Delta_h \in [-1, 1]^{512}$ such that $\mathbf{e}_n = \mathbf{W}_s \Delta_h$. Since $\mathbf{W}_s \in \mathbb{R}^{137 \times 512}$ and has full row rank, there can be an infinite number of solutions. We could use the psuedoinverse of $\mathbf{W}_s$ to find the Δ_h that has the minimum Euclidean norm. However, from Fig. 6 we see the L3 hidden state is typically sparse. Hence, we can look for a

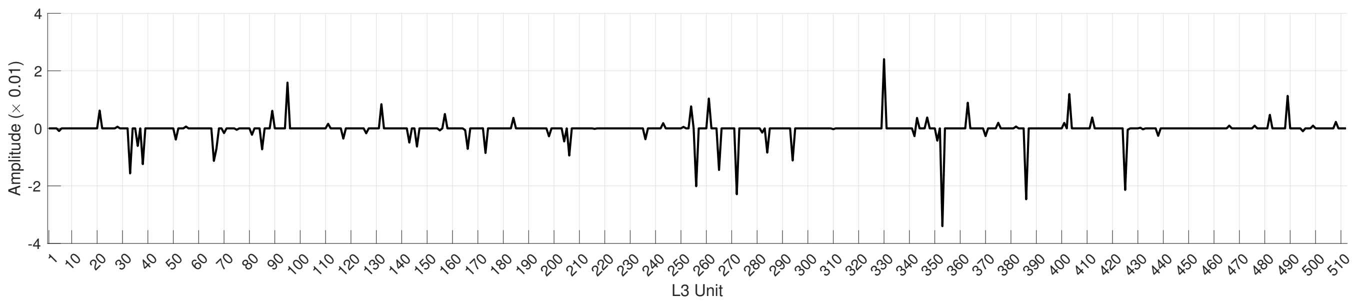

Figure 14: Deviation Δ_h of the layer three LSTM hidden state that attenuates the probability of token `A,` (dimension 106).

solution to the following:

$$\Delta_h = \arg \min_{\mathbf{h}\in[-1,1]^{512}} \|\mathbf{h}\|_0 \text{ subject to } \|\mathbf{W}_s\mathbf{h} - \mathbf{e}_n\|_2 < \epsilon, \|\mathbf{h}\|_\infty \leq 1 \quad (15)$$

for $\epsilon \geq 0$. Disregarding the infinity norm, this can be solved approximately by the LASSO (Tibshirani 1996), essentially replacing the ℓ_0-pseudo-norm with the ℓ_1-norm and then using convex optimisation. Figure 14 shows an L3 hidden state deviation Δ_h that amplifies or attenuates the softmax probability of the token `A,` (dimension 106).

Conclusion

We have analysed the parameters of the softmax layer of a specific folk-rnn LSTM model in order to gain insight into how it is working. Since the model is just a series of affine linear transformations and non-linearities, multivariate analysis can be used to uncover the effects of its parameters. We start with the softmax layer (6), which in turn illuminates the behaviour of the third LSTM layer. We use singular value decomposition to discover and understand the significance of subspaces of the column space of $\mathbf{W}_s$ in terms of the transcription vocabulary. This leads us to uncover the critical importance of measure tokens to the model's success in constructing transcriptions with plausible global characteristics. Our analysis also leads to ways to adjust the behaviours of the model, e.g., perturbing the L3 hidden state in order to attenuate the probability of particular tokens. Our next steps involve propagating our findings here to analyse the contributions of the LSTM layers.

References

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Proc. NIPS*.

Colombo, F.; Seeholzer, A.; and Gerstner, W. 2017. Deep artificial composer: A creative neural network model for automated melody generation. In *Proc. EvoMUSART*.

Eck, D., and Lapamle, J. 2008. Learning musical structure directly from sequences of music. Technical report, University of Montreal, Montreal, Canada.

Franklin, J. A. 2006. Recurrent neural networks for music computation. *Journal on Computing* 18(3):321–338.

Graves, A. 2013. Generating sequences with recurrent neural networks. *ArXiv e-prints* (1308.0850).

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; and Schmidhuber, J. 2016. LSTM: A search space odyssey. *IEEE Trans. Neural Networks and Learning Systems* 28(10):2222–2232.

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

Karpathy, A.; Johnson, J.; and Li, F.-F. 2015. Visualizing and Understanding Recurrent Networks. *ArXiv e-prints* (1506.02078v1).

Sturm, B. L., and Ben-Tal, O. 2017. Taking the models back to music practice: Evaluating generative transcription models built using deep learning. *J. Creative Music Systems* 2(1).

Sturm, B. L.; Santos, J. F.; Ben-Tal, O.; and Korshunova, I. 2016. Music transcription modelling and composition using deep learning. In *Proc. 1st Conference on Computer Simulation of Musical Creativity*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *Proc. ICLR*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. Royal Statist. Soc. B* 58(1):267–288.

Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T.; and Lipson, H. 2015. Understanding Neural Networks Through Deep Visualization. *ArXiv e-prints* (1506.06579).

Zeiler, M. D., and Fergus, R. 2013. Visualizing and Understanding Convolutional Networks. *ArXiv e-prints* (1311.2901).