4

Opening the black box: analyzing the learned music representation

A common criticism of deep learning methods are their lack of interpretability, an area where symbolic rule-based methods particularly excel. In this section, we argue the opposite viewpoint and demonstrate that characterization of the concepts learned by the model can be surprisingly insightful. The benefits of cautiously avoiding prior assumptions pay off as we discover the model itself learns musically meaningful concepts without any supervision.

4.1 Investigation of neuron activation responses to applied stimulus

Inspired by stimulus-response studies performed in neuroscience, we choose to characterize the internals of our sequence model by applying an analyzed music score as a stimulus and measuring the resulting neuron activations. Our aim is to see if any of the neurons have learned to specialize to detect musically meaningful concepts.

We use as stimulus the music score shown in fig. 4.1, which has already been preprocessed as described in section 3.1.1 on page 18. To aid in relating neuron activities back to music theory, chords are annotated with Roman numerals obtained using `music21`'s automated analysis.

Note that Roman numeral analysis involves subjectivity, and the results of automated analyses should be carefully interpreted.

4.1.1 Pooling over frames

In order to align and compare the activation profiles with the original score, all the activations occuring in between two chord boundary delimiters must be combined. This aggregation of neuron activations from higher resolution (*e.g.* note-by-note) to lower resolution (*e.g.* frame-by-frame) is reminiscent of pooling operations in convolutional neural networks [3]. Motivated by this observation, we introduce a method for pooling an arbitrary number of token-level activations into a single frame-level activation.

Let $\boldsymbol{y}^{(l)}_{t_m:t_n}$ denote the **activations** (*e.g.* outputs) of layer l from the t_mth input token $\boldsymbol{x}_{t_m}$ to the t_nth input token $\boldsymbol{x}_{t_n}$. Suppose that $\boldsymbol{x}_{t_m}$ and $\boldsymbol{x}_{t_n}$ are respectively the mth and nth chord boundary delimiters within the input sequence. Define the **max-pooled frame-level activations** $\tilde{\boldsymbol{y}}^{(l)}_n$ to be the element-wise maximum of $\boldsymbol{y}^{(l)}_{t_m:t_n}$, that is:

$$\tilde{\boldsymbol{y}}^{(l)}_n \coloneqq \left[\max_{t_m<t<t_n} \boldsymbol{y}^{(l)}_{t,1}, \quad \max_{t_m<t<t_n} \boldsymbol{y}^{(l)}_{t,2}, \quad \cdots, \quad \max_{t_m<t<t_n} \boldsymbol{y}^{(l)}_{t,N^{(l)}}\right]^\intercal \tag{4.1}$$

where $\boldsymbol{y}^{(l)}_{t,i}$ is the activation of neuron i in layer l at time t and $N^{(l)}$ is the number of neurons in layer l. Notice that the pooled sequence $\tilde{\boldsymbol{y}}$ is now indexed by frames rather than by tokens and hence corresponds to time-steps.

We choose to perform max pooling because it preserves the maximum activations of each neuron over the frame. While pooling methods (*e.g.* sum pooling, average pooling) are possible, we did not find significant differences in the visualizations produced.

The max-pooled frame-level activations are shown in fig. 4.2 As a result of pooling, the horizontal axis can be aligned and compared against the stimulus fig. 4.1. Notice the appearance of vertical bands corresponding to when a chord/rest is held for multiple frames. In particular, the vector embedding corresponding to rests (*e.g.* near frames 30 and 90 in fig. 4.2 top) are sparse, showing up as white smears not only in the embedding layer but on all LSTM memory cells. (the unpooled token-level activations are deferred to fig. A.4 on page 51)

4.1.2 Probabilistic piano roll: likely variations of the stimulus

fliang: Revise: the piano roll reference is removed

The bottom panel in fig. 4.2 shows the model's predictions for tokens in the next frames, where the tokens are arranged according to (arbitrary) index within the vocabulary. As the to-

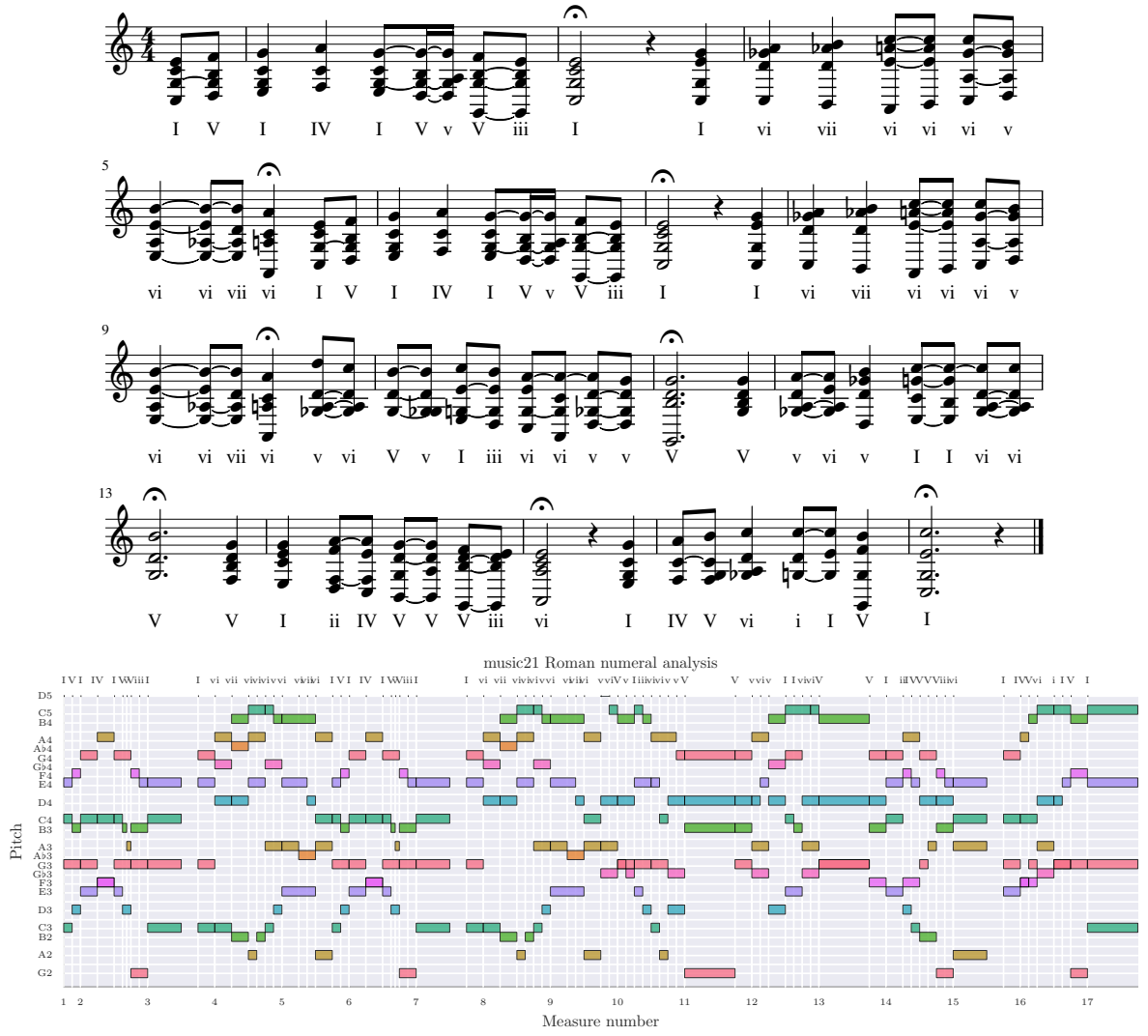


Fig. 4.1 *Top*: The preprocessed score (BWV 133.6) used as input stimulus with Roman numeral analysis annotations obtained from `music21`; *Bottom*: The same stimulus represented on a piano roll

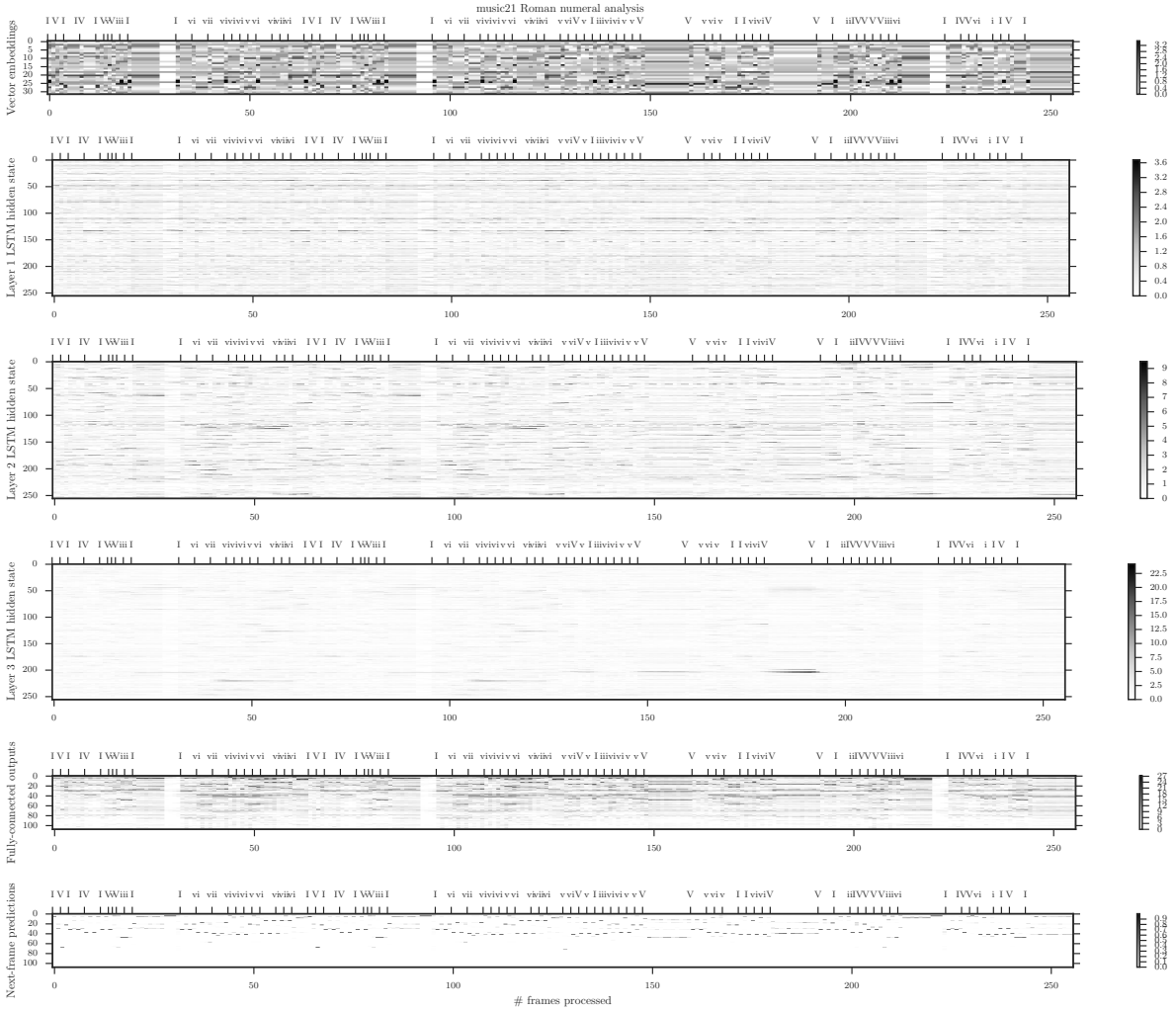


Fig. 4.2 Neuron activations, pooled over frames

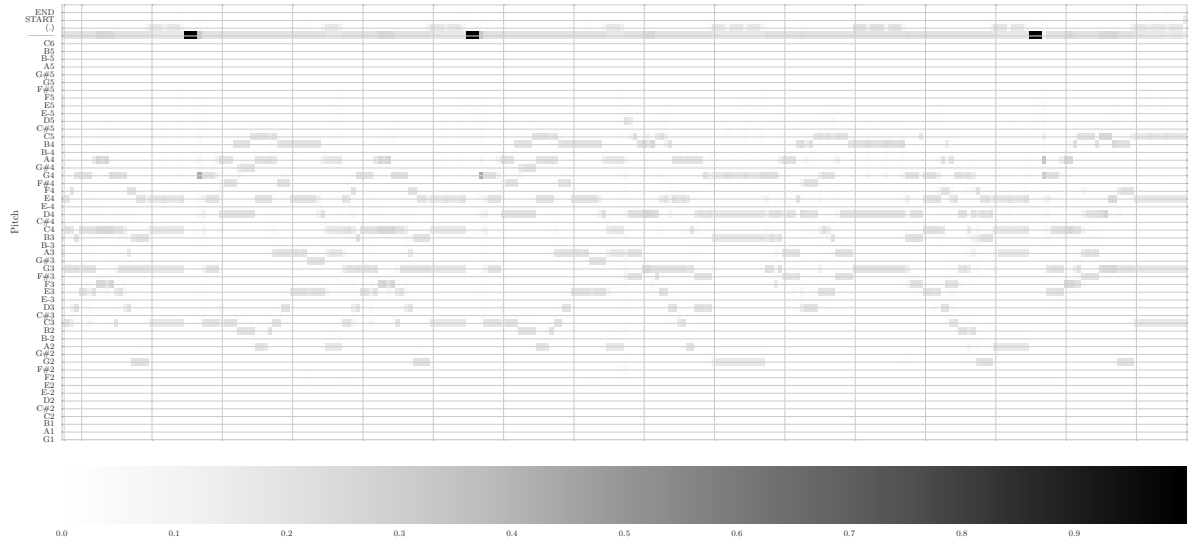


Fig. 4.3 Probabilistic piano roll of next note predictions. Note the strong predictions for fermatas near ends of phrases and the uncertain predictions immediately after long periods of rest.

kens correspond to pitches, they can be sorted according to pitch to reconstruct a **probabilistic piano roll**[1] consisting of the model's sequence of next-frame predictions as it processes the input.

fliang: Revise: the piano roll reference is removed

Notice that the probabilistic piano roll in fig. 4.3 closely resembles the stimulus. This is unsurprising because the recurrent inputs are taken from the stimulus rather than sampled from the model's predictions (a.k.a. [4]), so a model which predicts to only continue holding its input would produce a probabilistic piano roll identical to the stimulus delayed by one frame.

Two interesting rows of fig. 4.3 are the rows corresponding to frame delimiters (fourth from top, "|||") and fermatas (third from top "(.)"). Notice that the predictions for chord delimiters are particularly strong during rests. This is because rests are encoded as empty frames, so the large probability values indicate that the model has learned to prolong periods of rests. At the end of rest periods, the model tends to assign probability across a wide range of notes, consistent with the intuition that the possible notes occuring directly after a rest is less constrained than

fliang: cite the intuition?

those occuring in the middle of a phrase. Finally, notice that the probability assigned to fermatas is larger near the ends of phrases, suggesting that the model has successfully learned the concept of phrasing within music.

The probabilistic piano roll can be interpreted as variations on the stimulus which the model finds likely and may serve as a useful computational tool for generating likely chorale variations.

4.1.3 Neurons specific to musical concepts

Research in convolutional networks has shown that individual neurons within the network oftentimes specialize and specifically detect certain high-level visual features

fliang: Cite deconvolution

. Extending the analogy to musical data, we might expect certain neurons within our learned model to act as specific detectors to certain musical concepts.

To investigate this further, we look at the activations over time of individual neurons within the LSTM memory cells. Our results confirm our hypothesis: we discover certain neurons whose activities are correlated to specific motifs, chord progression, and phrase structures. The activity profiles of these neurons are shown in fig. 4.4.

For notational clarity, we will use the ordered tuple (l, i) to refer to the ith neuron in layer l.

The first three neurons ((1, 64), (1, 138), (1, 207)) shown in the 2nd to 4th panel from top of fig. 4.4 effectively behave like cadence detectors. While they all exhibit activity when the stimulus contains V chords (*i.e.* G-major). (1, 64) and (1, 138) are both specific to perfect cadences (*i.e.* $V--I$ chord progressions) used to conclude phrases and differ only in the chord inversions which they are most sensitive to. In contrast, (1, 207) only exhibits activity for the V chord associated with the imperfect cadences near frames 150 and 180.

The next two neurons in fig. 4.4, (1, 87) and (1, 151), act as motif detectors. Activity in (1, 151) peaks when a $vi - vii - vi$ progression is present in the stimulus. (1, 87) exhibits large spikes on $I--V--I$,

(2, 37) exhibits less specificity, but has large spikes right before the IV chord in $I--IV$ chord progressions.

(2, 82) peaks at the top of an ascending harmonic progressions, right before a descending major scale is to follow.

(2, 243) is specific to $v--vi$ progressions, with large spikes occuring at the $v--vi$ progressions near frames 55, 120, 130, and a lower intensity spike at 170. Some activity is also observed for the $V--vi$ around frame 230 despite the first chord being a major mode V rather than minor v.

fliang: Add mark's analysis and credit him

fliang: This refutes criticisms of black boxness of approach, play it up

Fig. 4.4 Activation profiles of neurons within our model which have learned high-level musical concepts

References

[1] Douglas Eck and Jasmin Lapalme. "Learning musical structure directly from sequences of music". In: **University of Montreal, Department of Computer Science, CP** 6128 (2008).

[2] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: **Proceedings of The 30th International Conference on Machine Learning** 2 (2012), pp. 1310–1318. ISSN: 1045-9227. DOI: 10.1109/72.279181. arXiv: arXiv:1211.5063v2. URL: http://jmlr.org/proceedings/papers/v28/pascanu13.pdf.

[3] Dominik Scherer, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". In: **International Conference on Artificial Neural Networks**. Springer. 2010, pp. 92–101.

[4] Ronald J Williams and David Zipser. "A learning algorithm for continually running fully recurrent neural networks". In: **Neural computation** 1.2 (1989), pp. 270–280.

A

Appendix C: Additional Proofs, Figures, and Tables

This section contains additional proofs, figures, and tables referenced from the body of this work. It is intended for readers who wish to examine our claims in greater detail.

A.1 Sufficient conditions for vanishing gradients

Following Pascanu, Mikolov, and Bengio [2], let $\|\cdot\|$ be any submultiplicative matrix norm (*e.g.* Frobenius, spectral, nuclear, Shatten p-norms). Without loss of generality, we will use the **operator norm** defined as

$$\|A\| = \sup_{x \in \mathbb{R}^n; x \neq 0} \frac{|Ax|}{|x|} \tag{A.1}$$

where $|\cdot|$ is the standard Euclidian norm.

Applying the definition of submultiplicativity to the factors of the product in eq. (1.4), we have that for any k

$$\left\| \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{h}_{k-1}} \right\| \leq \|\boldsymbol{W}_{hh}^{\mathsf{T}}\| \| \operatorname{diag}\left(\sigma'_{hh}(\boldsymbol{h}_{k-1})\right) \| \leq \gamma_{\boldsymbol{W}} \gamma_\sigma \tag{A.2}$$

where we have defined $\gamma_{\boldsymbol{W}} = \|\boldsymbol{W}_{hh}^{\intercal}\|$ and

$$\gamma_\sigma \coloneqq \sup_{h\in\mathbb{R}^n} \| \operatorname{diag}\left(\sigma'_{hh}(\boldsymbol{h})\right) \| \tag{A.3}$$

$$= \sup_{h\in\mathbb{R}^n} \max_i \sigma'_{hh}(\boldsymbol{h})_i \qquad \text{Operator norm of diag} \tag{A.4}$$

$$= \sup_{x\in\mathbb{R}} \sigma'_{hh}(x) \qquad \sigma_{hh} \text{ acts elementwise} \tag{A.5}$$

Substituting back into eq. (1.4), we find that

$$\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k} \right\| = \left\| \prod_{t\geq i>k} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}} \right\| \leq \prod_{t\geq i>k} \left\| \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{h}_{i-1}} \right\| \leq (\gamma_{\boldsymbol{W}}\gamma_\sigma)^{t-k} \tag{A.6}$$

Hence, we see that a sufficient condition for vanishing gradients is for $\gamma_{\boldsymbol{W}}\gamma_\sigma < 1$, in which case $\left\| \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_k} \right\| \to 0$ exponentially for long timespans $t \gg k$.

If γ_σ is bounded, sufficient conditions for vanishing gradients to occur may be written as

$$\gamma_{\boldsymbol{W}} < \frac{1}{\gamma_\sigma} \tag{A.7}$$

This is true for commonly used activation functions (*e.g.* $\gamma_\sigma = 1$ for $\sigma_{hh} = \tanh$, $\gamma_\sigma = 0.25$ for $\sigma_{hh} = \text{sigmoid}$).

The converse of the proof implies that $\|\boldsymbol{W}_{hh}^{\intercal}\| \geq \frac{1}{\gamma_\sigma}$ are necessary conditions for $\gamma_{\boldsymbol{W}}\gamma_\sigma > 1$ and exploding gradients to occur.

A.1.1 Quantifying the effects of preprocessing

Related discussion is in section 3.1.1 on page 18.

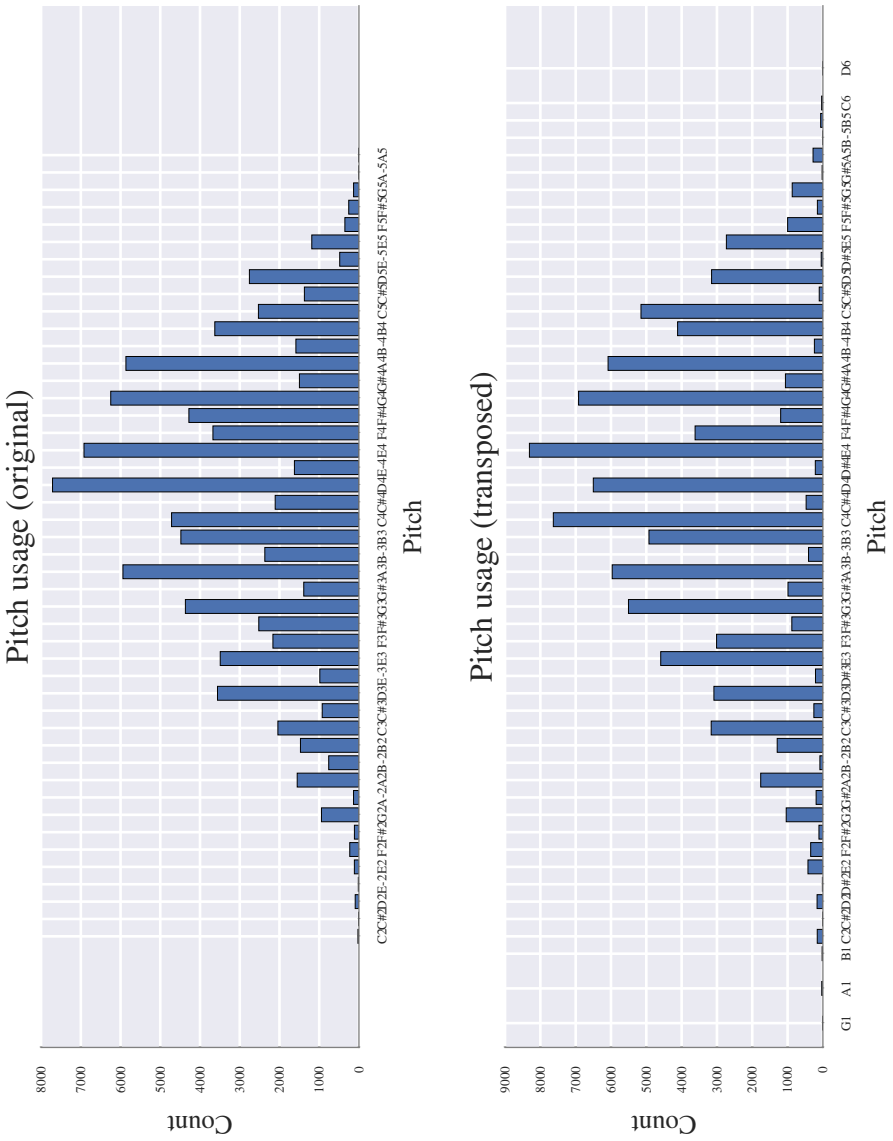


Fig. A.1 Distribution of pitches used over Bach chorales corpus. Transposition has resulted in an overall broader range of pitches and increased the counts of pitches which are in key.

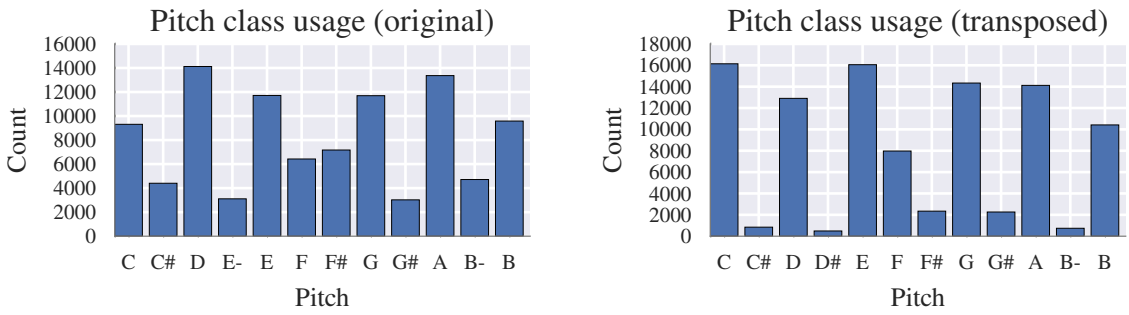


Fig. A.2 Distribution of pitch classes over Bach chorales corpus. Transposition has increased the counts for pitch classes within the C-major / A-minor scales.

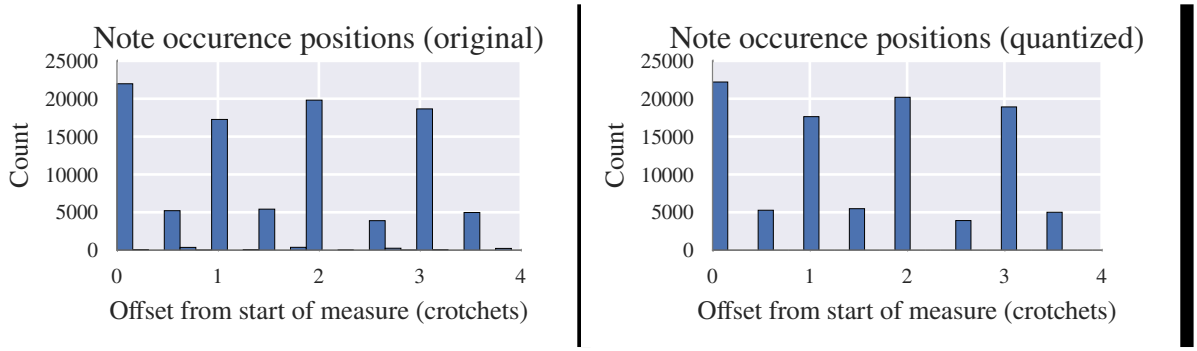


Fig. A.3 Meter is minimally affected by quantization due to the high resolution used for time quantization.

A.1.2 Discovering neurons specific to musical concepts

Related discussion is in section 4.1.3 on page 36.

A.1.3 Identifying and verifying local optimality of the overall best model

Related discussion is in section 3.2.5 on page 28.

Fig. A.5 Results of grid search (see Section 3.2.5) over LSTM sequence model hyperparameters

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
3.0	256.0	128.0	32.0	0.323781	0.477027

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
2.0	256.0	128.0	32.0	0.323668	0.479322
2.0	256.0	128.0	64.0	0.303158	0.482216
3.0	256.0	256.0	64.0	0.320361	0.484231
3.0	256.0	128.0	32.0	0.383811	0.484667
3.0	256.0	128.0	16.0	0.342955	0.484791
2.0	256.0	256.0	64.0	0.373641	0.485353
3.0	256.0	128.0	64.0	0.305290	0.486244
2.0	256.0	128.0	32.0	0.275125	0.486305
2.0	256.0	256.0	32.0	0.352257	0.486755
4.0	256.0	128.0	32.0	0.333133	0.487135
2.0	256.0	256.0	32.0	0.307188	0.487868
2.0	256.0	256.0	32.0	0.400955	0.489320
3.0	256.0	256.0	64.0	0.381868	0.489810
2.0	256.0	256.0	64.0	0.333356	0.491396
2.0	256.0	256.0	64.0	0.284248	0.491593
3.0	128.0	128.0	32.0	0.365171	0.492478
3.0	256.0	128.0	32.0	0.264723	0.492849
3.0	384.0	128.0	32.0	0.228556	0.495991
3.0	256.0	128.0	64.0	0.248987	0.496190
3.0	256.0	128.0	32.0	0.445840	0.498205
3.0	256.0	256.0	32.0	0.273567	0.499422
2.0	256.0	128.0	64.0	0.256022	0.500500
3.0	256.0	256.0	32.0	0.338776	0.501711
2.0	128.0	128.0	32.0	0.384075	0.501840
3.0	128.0	128.0	64.0	0.417780	0.501919
2.0	256.0	128.0	32.0	0.219939	0.502503
3.0	128.0	128.0	64.0	0.361381	0.503206
3.0	128.0	128.0	32.0	0.431771	0.503590
3.0	256.0	64.0	64.0	0.263001	0.503945
3.0	256.0	384.0	64.0	0.419091	0.504249
3.0	256.0	256.0	32.0	0.393463	0.506486
2.0	128.0	128.0	64.0	0.364640	0.506923
2.0	128.0	128.0	64.0	0.422178	0.507268

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
3.0	256.0	256.0	64.0	0.261563	0.507479
3.0	256.0	64.0	32.0	0.278916	0.507673
2.0	128.0	128.0	32.0	0.434552	0.508460
3.0	256.0	384.0	32.0	0.439684	0.514804
1.0	256.0	128.0	64.0	0.334873	0.517134
2.0	128.0	128.0	64.0	0.465061	0.520224
2.0	256.0	128.0	64.0	0.195905	0.521330
1.0	256.0	256.0	64.0	0.368281	0.522424
2.0	128.0	128.0	32.0	0.485346	0.522955
2.0	128.0	256.0	64.0	0.378280	0.525397
3.0	512.0	128.0	32.0	0.168366	0.525644
1.0	256.0	256.0	64.0	0.417803	0.525980
3.0	128.0	128.0	64.0	0.480340	0.526121
3.0	128.0	128.0	32.0	0.491876	0.527008
3.0	256.0	128.0	32.0	0.194120	0.528000
2.0	128.0	128.0	64.0	0.296537	0.528261
2.0	128.0	128.0	32.0	0.316390	0.529308
3.0	128.0	256.0	64.0	0.435649	0.529458
1.0	256.0	128.0	32.0	0.375717	0.529638
2.0	128.0	256.0	64.0	0.440450	0.529948
1.0	256.0	256.0	64.0	0.389651	0.531063
2.0	128.0	256.0	128.0	0.362561	0.533559
2.0	128.0	256.0	32.0	0.398919	0.533672
3.0	128.0	256.0	32.0	0.452009	0.536955
1.0	256.0	128.0	32.0	0.346140	0.538510
2.0	128.0	128.0	128.0	0.273516	0.539359
1.0	256.0	128.0	64.0	0.310597	0.539599
3.0	128.0	128.0	32.0	0.265842	0.539827
1.0	256.0	128.0	64.0	0.274568	0.541263
3.0	128.0	256.0	64.0	0.500697	0.544048
1.0	256.0	128.0	32.0	0.316189	0.545363
1.0	256.0	128.0	32.0	0.285714	0.546995
3.0	128.0	128.0	64.0	0.247192	0.549826

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
1.0	128.0	128.0	64.0	0.458142	0.550102
1.0	128.0	128.0	128.0	0.360038	0.550509
2.0	128.0	256.0	32.0	0.465110	0.550995
1.0	256.0	256.0	32.0	0.444180	0.551894
3.0	256.0	128.0	64.0	0.184959	0.552200
2.0	128.0	256.0	64.0	0.490587	0.552217
2.0	128.0	256.0	32.0	0.514900	0.553092
1.0	128.0	128.0	64.0	0.487574	0.553498
1.0	256.0	256.0	32.0	0.471938	0.553586
1.0	128.0	128.0	64.0	0.384282	0.554990
1.0	128.0	128.0	64.0	0.425469	0.555312
1.0	256.0	256.0	32.0	0.411686	0.555955
1.0	256.0	128.0	64.0	0.238860	0.556672
3.0	64.0	128.0	64.0	0.420250	0.559336
3.0	64.0	64.0	128.0	0.345705	0.559549
3.0	128.0	128.0	128.0	0.238071	0.562603
2.0	256.0	128.0	32.0	0.143647	0.563866
1.0	128.0	128.0	32.0	0.489160	0.564304
3.0	128.0	256.0	32.0	0.521478	0.566153
2.0	128.0	128.0	64.0	0.584950	0.567093
2.0	64.0	128.0	64.0	0.443393	0.567754
2.0	128.0	256.0	64.0	0.549169	0.568419
1.0	128.0	64.0	32.0	0.359041	0.569011
3.0	128.0	256.0	64.0	0.573862	0.570873
1.0	128.0	128.0	32.0	0.525982	0.571859
3.0	64.0	128.0	128.0	0.408074	0.572306
1.0	128.0	128.0	32.0	0.467434	0.572480
1.0	128.0	128.0	32.0	0.417764	0.573797
2.0	64.0	64.0	32.0	0.413944	0.573993
3.0	64.0	64.0	64.0	0.355615	0.574236
1.0	256.0	128.0	128.0	0.204964	0.574585
1.0	128.0	64.0	64.0	0.328927	0.575464
2.0	64.0	64.0	64.0	0.390597	0.575592

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
2.0	64.0	128.0	128.0	0.424735	0.575868
2.0	64.0	32.0	32.0	0.399389	0.577974
2.0	64.0	64.0	128.0	0.372478	0.578856
2.0	128.0	64.0	32.0	0.240288	0.580802
3.0	64.0	64.0	32.0	0.375478	0.582072
1.0	128.0	64.0	128.0	0.304245	0.582897
3.0	64.0	128.0	32.0	0.430421	0.582991
3.0	128.0	256.0	32.0	0.590133	0.585245
3.0	64.0	32.0	32.0	0.348150	0.585800
2.0	64.0	32.0	64.0	0.387047	0.589173
1.0	128.0	256.0	64.0	0.501138	0.593823
3.0	64.0	32.0	128.0	0.339394	0.594401
1.0	128.0	32.0	32.0	0.348193	0.595001
2.0	64.0	128.0	32.0	0.470837	0.597005
3.0	64.0	32.0	64.0	0.344404	0.597406
2.0	128.0	64.0	64.0	0.224014	0.597418
1.0	64.0	32.0	64.0	0.462827	0.597437
1.0	64.0	32.0	32.0	0.500014	0.598521
2.0	64.0	32.0	128.0	0.376624	0.600570
1.0	64.0	32.0	128.0	0.453646	0.604043
1.0	128.0	256.0	64.0	0.539087	0.604710
2.0	256.0	128.0	64.0	0.122328	0.606237
1.0	64.0	128.0	128.0	0.489255	0.607122
1.0	128.0	32.0	64.0	0.319029	0.609441
1.0	128.0	256.0	64.0	0.566182	0.610409
1.0	128.0	32.0	128.0	0.294204	0.613838
1.0	64.0	64.0	128.0	0.436633	0.615036
1.0	64.0	64.0	64.0	0.461935	0.616265
2.0	128.0	64.0	128.0	0.206896	0.620845
1.0	128.0	256.0	32.0	0.550056	0.627652
2.0	256.0	128.0	128.0	0.106181	0.631364
3.0	128.0	64.0	32.0	0.185779	0.633145
1.0	128.0	256.0	32.0	0.591930	0.638022

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
1.0	256.0	64.0	32.0	0.200897	0.640652
1.0	64.0	64.0	32.0	0.487779	0.643943
1.0	128.0	256.0	32.0	0.621720	0.647467
2.0	128.0	32.0	32.0	0.209044	0.647553
3.0	256.0	128.0	32.0	0.100153	0.650138
1.0	64.0	128.0	64.0	0.515733	0.653191
1.0	256.0	64.0	64.0	0.171567	0.657626
3.0	256.0	128.0	64.0	0.087426	0.660995
3.0	128.0	64.0	128.0	0.169560	0.663409
3.0	128.0	64.0	64.0	0.172871	0.670402
1.0	64.0	128.0	32.0	0.561724	0.670482
1.0	256.0	64.0	128.0	0.149129	0.672432
2.0	128.0	32.0	64.0	0.193615	0.688310
2.0	128.0	128.0	64.0	0.802259	0.696580
2.0	128.0	256.0	32.0	0.907374	0.701893
3.0	256.0	128.0	128.0	0.076598	0.711632
2.0	256.0	64.0	32.0	0.081134	0.716840
2.0	128.0	32.0	128.0	0.173684	0.727354
2.0	256.0	64.0	64.0	0.073675	0.742250
1.0	256.0	32.0	32.0	0.161496	0.743529
3.0	128.0	32.0	32.0	0.146775	0.752404
1.0	256.0	32.0	64.0	0.138145	0.755407
1.0	256.0	32.0	128.0	0.125931	0.757801
3.0	128.0	32.0	64.0	0.134530	0.770094
2.0	256.0	64.0	128.0	0.063084	0.797383
3.0	128.0	32.0	128.0	0.129410	0.801131
3.0	256.0	64.0	64.0	0.048852	0.823713
3.0	256.0	64.0	32.0	0.052363	0.848516
2.0	256.0	32.0	32.0	0.058634	0.874037
3.0	256.0	64.0	128.0	0.044448	0.876398
2.0	256.0	32.0	128.0	0.049791	0.888397
2.0	256.0	32.0	64.0	0.050012	0.898488
3.0	256.0	32.0	32.0	0.037417	0.960396

Continued on next page

num_layers	rnn_size	seq_length	wordvec	train_metric	val_metric
3.0	256.0	32.0	64.0	0.034403	0.988554
3.0	256.0	32.0	128.0	0.036275	0.990457

A.1.4 Additional large-scale subjective evaluation results

Related discussion is in **??** on page ??.

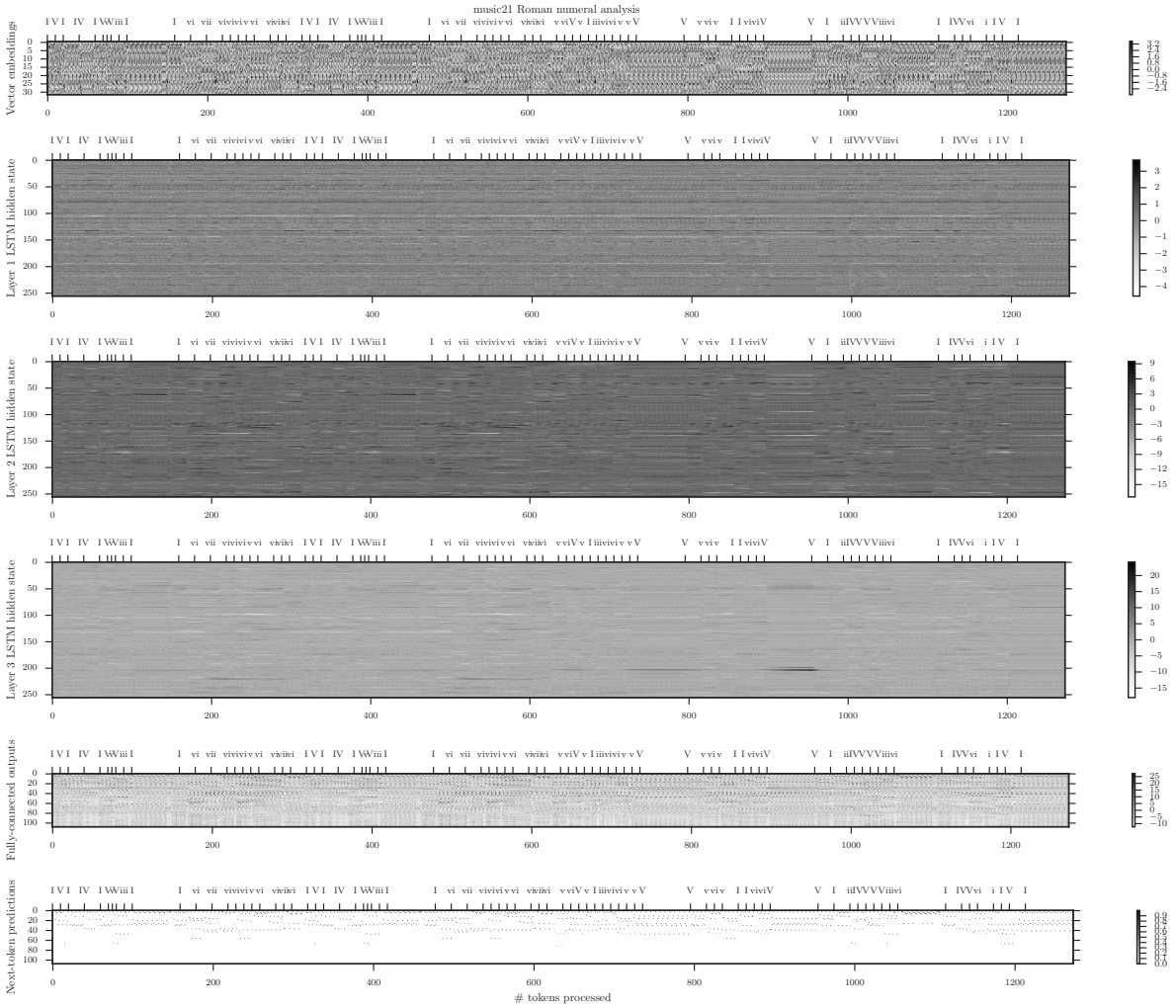


Fig. A.4 Neuron activations over time as the encoded stimulus is processed token-by-token

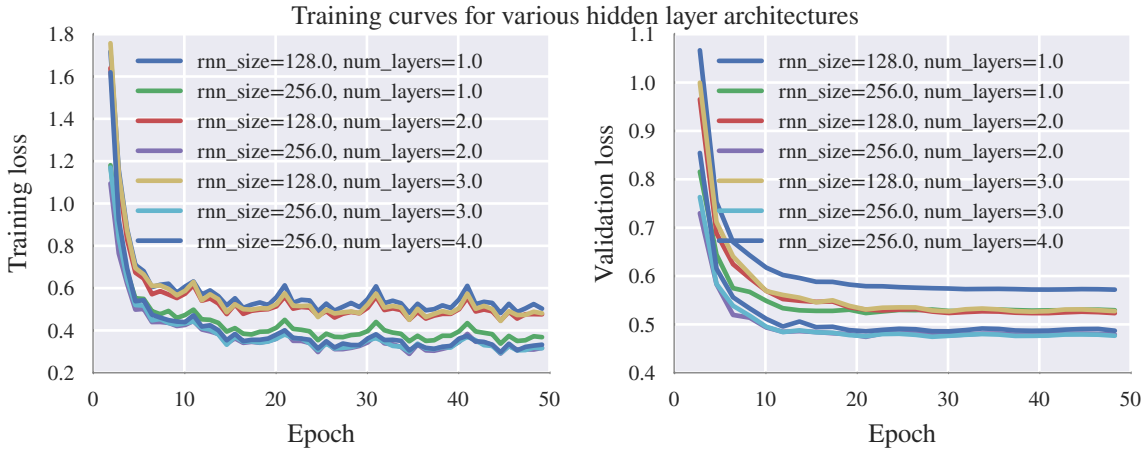


Fig. A.6 `rnn_size=256` and `num_layers=3` yields lowest validation loss.

Fig. A.7 Validation loss improves initially with increasing network depth but deteriorates after > 3 layers.

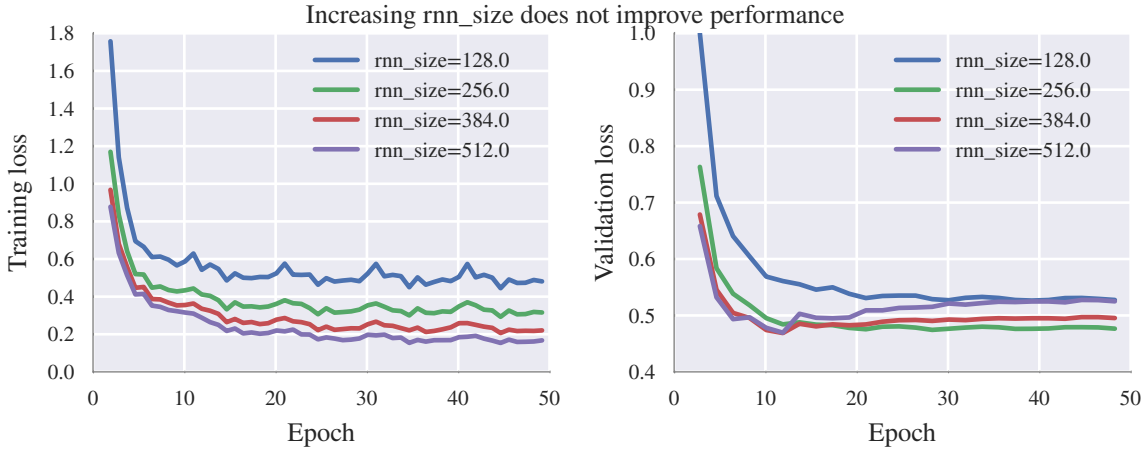


Fig. A.8 Validation loss improves initially with higher-dimensional hidden states but deteriorates after > 256 dimensions.

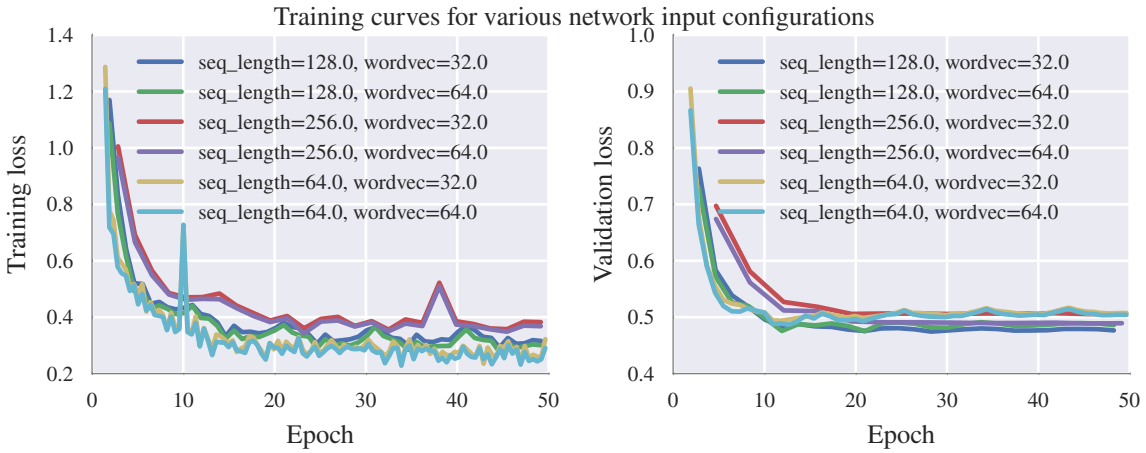


Fig. A.9 `seq_length=128` and `wordvec=32` yields lowest validation loss.

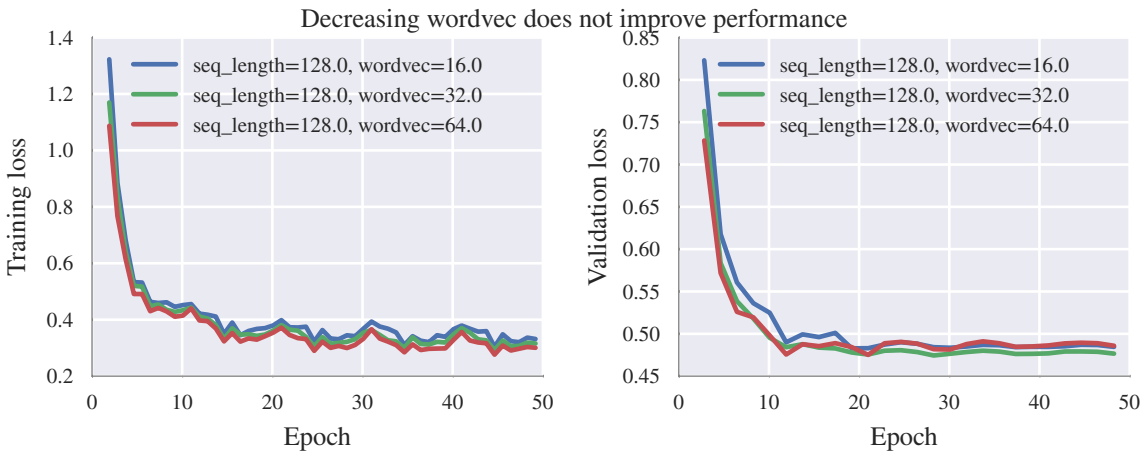
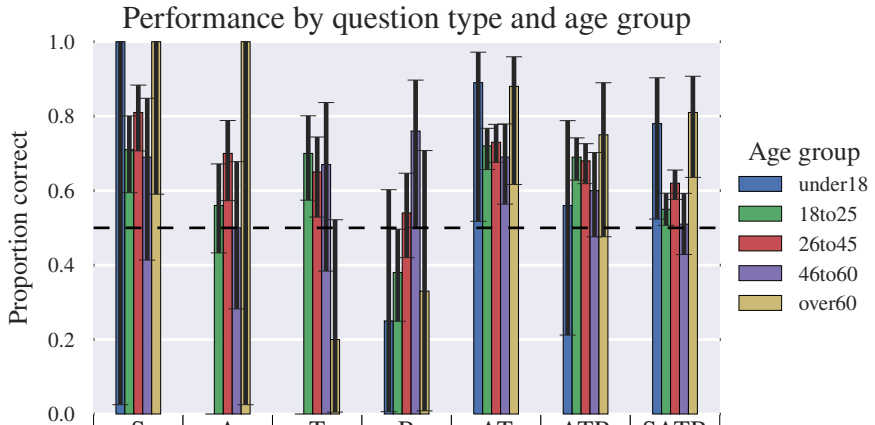


Fig. A.10 Perturbations about `wordvec=32` do not yield significant improvements.



	S	A	T	B	AT	ATB	SATB
under18	1.0	0.0	0.0	0.25	0.89	0.56	0.78
18to25	0.71	0.56	0.7	0.38	0.72	0.69	0.55
26to45	0.81	0.7	0.65	0.54	0.73	0.68	0.62
46to60	0.69	0.5	0.67	0.76	0.69	0.6	0.51
over60	1.0	1.0	0.2	0.33	0.88	0.75	0.81

Fig. A.11 Proportion of correct responses for each question type and age group.