Supplementary Material

A. Additional Details on Melody Representation

For the melody representation (vocabulary), we followed (Waite, 2016) to encode the melody as a sequence of 92 unique tokens and quantized it to a 100ms grid. For the extraction procedure, we used the algorithm as outlined in the Magenta codebase (`https://github.com/tensorflow/magenta/blob/master/magenta/music/melody_inference.py`), where we use a heuristic to extract the note with the highest in a given performance. This heuristic is based on the assumption that all melodies coincide with actual notes played in the polyphonic performance. Specifically, we construct a transition matrix of melody pitches and use the Viterbi algorithm to infer the most likely sequence of melody events within a given frame.

B. NLL Evaluation for "Noisy" Model

Below, we provide the note-wise test NLL on the MAESTRO and YouTube datasets with melody conditioning, where the conditioning performance is perturbed by the procedure outlined in Section 3.

C. Model Architecture and Hyperparameter Configurations

We mostly use the default Transformer architecture as provided in the Tensor2Tensor framework, such as 8 self-attention heads as listed in the main text, and list the slight adjustments we made for each dataset below:

C.1. MAESTRO

For the MAESTRO dataset, we follow the hyperparameter setup of (Huang et al., 2019b):

1. num hidden layers = 6
2. hidden units = 384
3. filter size = 1024
4. maximum sequence length = 2048
5. maximum relative distance = half the hidden size
6. dropout = 0.1

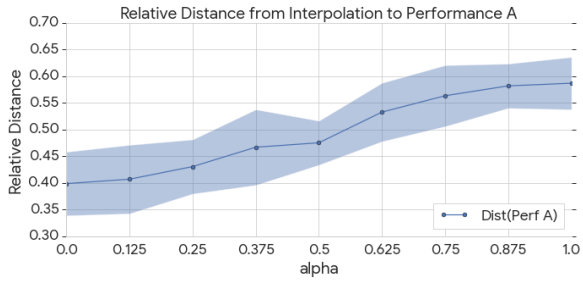
C.2. YouTube Dataset

For the YouTube dataset, we modify the number of hidden layers to 8 and slightly increase the level of dropout.

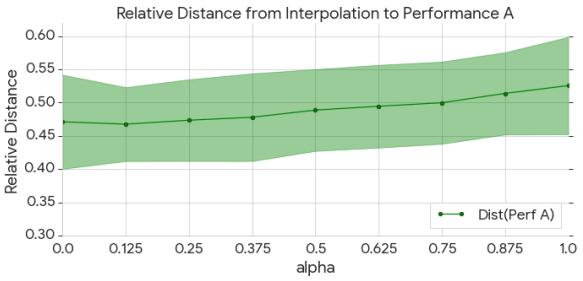
1. num hidden layers = 8
2. hidden units = 384
3. filter size = 1024
4. maximum sequence length = 2048
5. maximum relative distance = half the hidden size
6. dropout = 0.15

D. Additional Relative Distance Interpolations

In Figure 5, we show the interpolation relative distance results for the (a) performance and (b) melody & performance Transformer autoencoders for the MAESTRO dataset.



(a) Relative distance from interpolated sample to the original starting performance.



(b) Relative distance from the interpolated sample to the original melody, which is kept fixed.

Figure 5. The distance to the original performance increases as the value of α increases in (a), as expected. In (b), we see that there is a very slight increase in the relative distance to the original melody during the interpolation procedure.

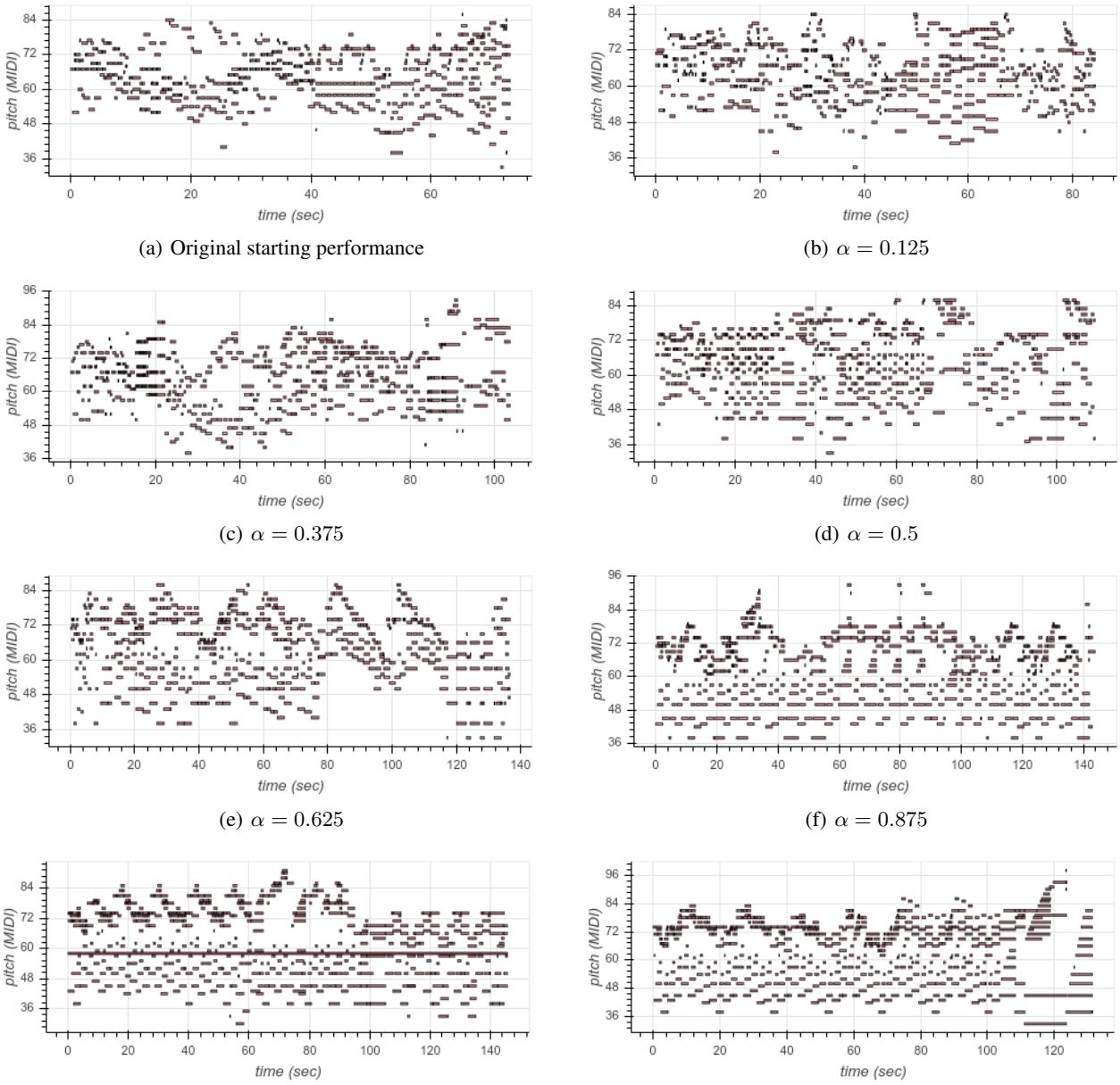
We find consistent results in these interpolations as provided in the main text.

E. Internal Dataset Performance Interpolations

In Figures 6 and 7, we provide piano rolls demonstrating the effects of latent-space interpolation for the YouTube dataset, for both the (a) performance and (b) melody & performance Transformer autoencoder respectively. For similar results in MAESTRO as well as additional listening samples, we refer the reader to the online supplement: `https://goo.gl/magenta/music-transformer-autoencoder-examples`.

Model variation	MAESTRO	YouTube Dataset
Noisy Melody TF autoencoder with relative attention, sum	1.721	1.248
Noisy Melody TF autoencoder with relative attention, concat	1.719	1.249
Noisy Melody TF autoencoder with relative attention, tile	1.728	1.253

Table 5. Note-wise test NLL on the MAESTRO and YouTube piano performance datasets with melody conditioning, with event-based representations of lengths $L = 2048$.



(a) Original starting performance

(b) $\alpha = 0.125$

(c) $\alpha = 0.375$

(d) $\alpha = 0.5$

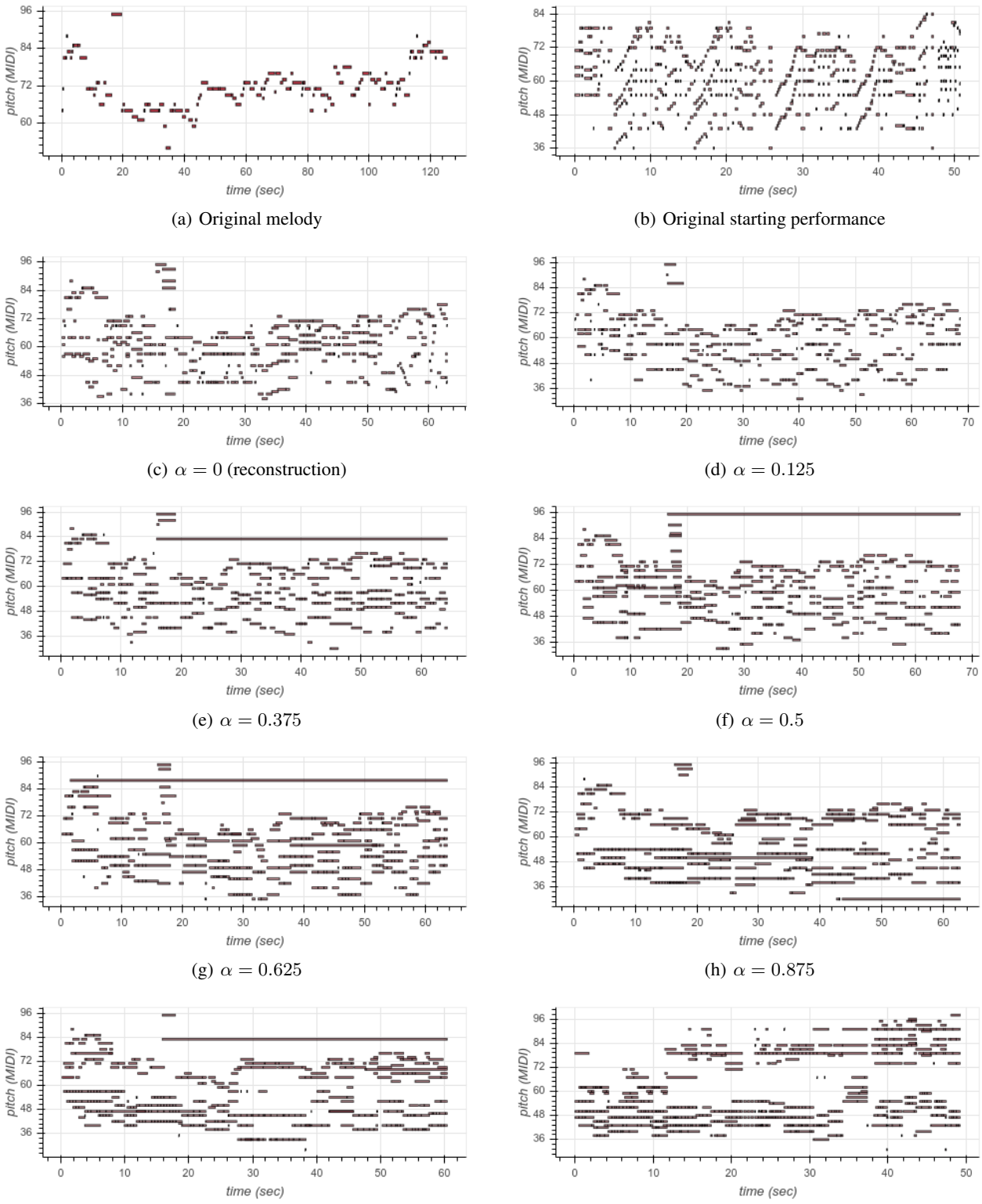
(e) $\alpha = 0.625$

(f) $\alpha = 0.875$

(g) $\alpha = 1.0$ (reconstruction)

(h) Original final performance

Figure 6. Interpolation of a starting performance (a) from the YouTube dataset to a final performance (h), with the coefficient α controlling the level of interpolation between the latent encodings between the two performances.



(a) Original melody

(b) Original starting performance

(c) $\alpha = 0$ (reconstruction)

(d) $\alpha = 0.125$

(e) $\alpha = 0.375$

(f) $\alpha = 0.5$

(g) $\alpha = 0.625$

(h) $\alpha = 0.875$

(i) $\alpha = 1.0$ (reconstruction)

(j) Original final performance

Figure 7. Interpolation of a starting performance (b) from the YouTube dataset to a final performance (j), with the coefficient α controlling the level of interpolation between the latent encodings between the two performances. The original conditioning melody (a) is kept fixed throughout the interpolation.