Melody-to-Chord using paired model and multi-task learning language modeling

Mu-Heng Yang
Department of Music
Stanford University
mhyang@stanford.edu

Wei-Ting Hsu
Department of Computer Science
Stanford University
wthsu@stanford.edu

Nicholas Huang
Department of Computer Science
Stanford University
nykh@stanford.edu

Abstract

In music composition, melody is relatively easy to generate: a piece composed even by an amateur can be pleasing at times. However, chords composition typically requires some musical background to have them be acoustically in harmony. In this report, we designed a system to generate chords matching a given melody using several models and draw similarities between natural language and music. We mainly focused on two models: a proposed paired model and a multi-task language model, each with a few extensions. Both models were able to generate chords that match to melody in harmony, and we found that proposed paired model performance was able to generate more pleasing music. [1] [2]

1 Music as a Language

Music and natural language are very similar. They are both in the form of sound (musical notes and speech) which can be transcribed into symbols (musical sheet and articles). Therefore, acoustic model and language model are extensively used on both of them respectively. Moreover, one can view music theory as the grammar in natural language and preprocess their hierarchical structure. The pitches, notes, bars, musical phrases, and chorus in music are the counterpart of phonemes, morphemes, words, phrases, and articles in natural language. Given so many similarities, many researchers deem music as a universal languageCohen [2008], Besson and Schön [2001]. In our project, we want to generate chord progression given melody, which is a sequence-to-sequence problem just like the language translation in NLP. There are still some difference between natural language and music. For example, music is represented in time units in music, which allows us to stretch our data to twice the length or compress it into half the length for data augmentation and take average of predictions for more stable results. Also, music theory can be formulated using mathematics and thereby have multiple representations. Based on these advantages that only music possess, we propose paired model and language modeling using multi-task learning and merging units methods for melody-to-chord in this report.

[1] Link to generated music https://goo.gl/Mzflpl
[2] Github: https://github.com/hsuwt/cs224n-project

2 Introduction

Melody and chords are main elements of music. Melody is a succession of musical notes that provide the foreground rhythm, while each chord is composed of multiple notes and the sequence of chords provides the harmonic background for melody. In music composition, we often have a melody line firstly and then need to find a chord progression to accompany it.

There are mainly two approaches to automatic music composition. The first approach exploits music theory and domain knowledge to formulate hand-crafted rules for composition. Such a model is usually explicitly coded, so it is difficult to extend the model to different genres or applications. This is generally not an issue for the second approach, which is based on machine learning and therefore data-driven. Some of the attempts at the second approach used simplistic models such as a finite state machine (Markov chain) [Forsyth and Bello, 2013, Musick, and Bello, 2015] or Hidden Markov Model [Simon, and Basu, 2008, Simon, et al., 2008].

A major limitation of both the FSM and HMM models is that they consider only the 1st-order relations between adjacent short-time frames of music; that is, they make the Markov assumption about chord progression. This proved to be too naive for actual music pieces, where long term correlation often happen across time frames. As a result, these methods tend to choose only the most common and conservative chords combinations.

Another limitation of HMM and FSM is the need to define the state spaces corresponding to chord types for melody-to-chord and musical notes for chord-to-melody. However, it might be difficult to adequately set the number of states. For example, for melody-to-chord, using too few chord types may limit the diversity and aesthetic quality of the generated chord sequence, whereas using too many chord types may lead to data imbalanced problems, since some of the chord types can be under-represented in the training data. It is better to get rid of the states and use sequential representations of melody and chords.

The popularity of the Internet and data sharing platforms has made available a large amount of data (e.g. musical scores) in digital format suitable for training deep neural networks [Bengio, and Hinton, 2015] to generate artworks. This has been attempted, for example, on visual arts [Ecker, and Bethge, 2015] and on generating full music piece given the first few notes (i.e. automatic music generation) [Eck and Schmidhuber, 2002, Bengio, and Vincent, 2012].

For automatic music composition, recurrent neural network (RNN) Goller and Kuchler [1996] represents a promising deep learning architecture, because it takes into account information from all the frames, allowing the model to analyze the songs as a whole. Intuitively, we can build a sequence-to-sequence (seq2seq) model to achieve either melody-to-chord conversion, by using melody sequence as the input and chord sequence as the output.

Even though seq2seq is widely used in NLP especially language translation, its architecture is best known for its ability to transform sequences with different length. Since melody and chord progression have the same length, we may modified language modeling Ponte and Croft [1998] to build a simpler architecture. Instead of predicting next frame like normal RNNLM Mikolov et al. [2010], we can predict the chord progression at each time-step given the current melody.

For most music composers, melody and chord are composed in parallel with multiple iterations, rather than separately. This motivates us to develop a unified model that can examine melody and chord at the same time and generate the correction of the chord to perform the music accompaniment at the same time, just like a human composer naturally does.

In this report, we proposed a new framework called the proposed paired model, or paired model for short. The paired model takes a pair of melody and chord lines as input, and output how good a match the incoming melody-chord pair constitute and additionally how to modify the inputs chord to get a better match.

By jointly learning their relationships, our paired model can better capture their dependency using information from all pairs of inputs, and then find the best melody-chord pair at test time.

3 Models

We experimented with three fundamentally different learning strategies or task definition for melody-to-chord: seq2seq, language modeling, and paired model. As discussed above, paired model takes a pair of melody and chord lines as input and tries to learn what constitute a good match between melody and chord, which establishes a commutative relation; language model, on the other hand, takes the sequence of melody as input and tries to learn and produce the best chord to accompany the input melody. The two models have their own merits which are discussed below.

We experimented with a variety of models combined with each strategy where applicable: basic RNN cells, Gated Recurrent Unit[Chung et al., 2014], the Long Short-term Memory network [Hochreiter and Schmidhuber, 1997], Bidirectional-RNN [Schuster and Paliwal, 1997], and seq2seq model[Cho et al., 2014]. In particular, the basic RNN can be combined with both learning tasks.

3.1 RNN variants

A standard RNN suffers from vanishing gradient problem, documented in Hochreiter and Schmidhuber [1997], and is prevented from learning long-term dependencies. Since each timestep in our model represents a 16th note on the music score, the length of sequence our RNN can be easily in the order of hundreds for a 10-second music clip. Gated cells such as GRUs are therefore used in our model to ameliorate the problem of vanishing gradients.

To learn even longer time dependency, we adopted LSTM model as network units. While RNNs have the so-called vanishing gradient problems, LSTM is free from such an issue and is best-known for classifying time-series data with long lags between events. For music, we can treat beats, bars, and musical phrases as musical events. The time signature also plays an important role in music composition. Therefore, we expect that the LSTM architecture can perform better than RNN.

While LSTM has more parameters and may be more capable of modeling complex data, GRUs takes less memory and are faster to train. With our dataset not being tremendously large, GRUs may be more suited for our application.

For musicians, the process of composing music is rarely unidirectional. They often modify the previous notes based on the subsequent notes, and vice versa. Therefore, we also tested *bidirectional RNN* (BRNN) Schuster and Paliwal [1997], which consists of two recurrent models that learn forward and backward dependency respectively. The prediction is based on both directions. Similarly, we can have *bidirectional LSTM* (BLSTM).

3.2 Proposed Paired strategy

To capture the complex relationship between chords and melody, we want our model to be a music connoisseur who can differentiate between good and bad music. Given a melody/chord pair, the task is to determine whether the melody and chords are compatible or a good match.

Specifically, we adopt a simple strategy in generating data pairs with different goodness-of-match for model training: With 1,000 songs, we cross product the melody and chord into 1,000,000 pairs and assume melody and chords from different songs are mismatch, and those from the same song are perfect match. During test time, we feed all chord progressions with the input melody to see which chord candidate get the highest score.

An advantage of this strategy is that we do not need extra efforts in manually labeling the goodness-of-match of melody/chord pairs. To balance the positive and negative samples in the dataset. We shuffle the chord melody pairs as negative pairs and keep the number of positive data pairs the same as the negative ones.

Some may argue that randomly picking negative pairs might get a very great chord but labelled as negative samples. To deal with this problem, we extend our model to predict the "correction" from the input chord to ground truth chord. E.g. if our input chord is C Maj

[1,0,0,0,1,0,0,1,0,0,0,0] and output chord is C Min [1,0,0,1,0,0,0,1,0,0,0,0], Our correction ground truth would be their difference: [0,0,0,-1,1,0,0,0,0,0,0,0], indicating that we should add the E note and delete the D# note from the input chord progression. This model allow us to not only generate a great chord progression but also serve as a music composition teacher, telling us which notes we should correct to sound better.

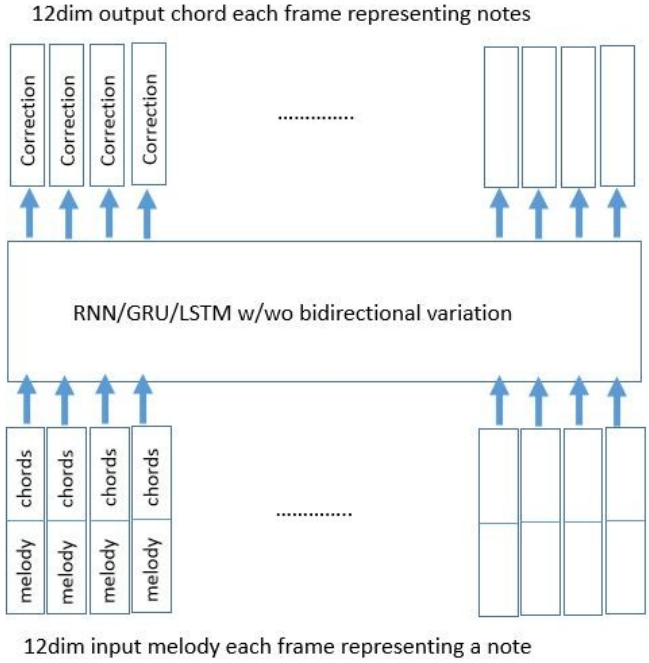

Figure 1: pair-model learning depiction

3.3 Language modeling strategy

Language model treats melody as input and chords as label at each timestep and attempts to predict the most suitable chords to an unseen melody. Musical chords, the ones our model will predict, can be described in both one-hot representation and chroma format. E.g. C major chord have notes C, E, and G, so its chroma vector is [1,0,0,0,1,0,0,1,0,0,0,0]. Both representation have their merits. The one-hot representation treats each chord as completely discrete entities and would not be able to predict chords unseen from the training set; whereas the chroma representation allows the model to learn similarities between any chords that have sharing notes and enable unseen chords prediction, but is prone to predict notes combinations that are not actual chords. The fact that chords can be expressed in chroma format an important property of music that natural languages don't have. Using this advantage, we build a one-to-one mapping between both representations and propose a multi-task learning framework to jointly learn the correlation of one-hot and chroma vectors during training and use ensemble from both prediction at test time. We experimented our models in one-hot representation, chroma representation, and multi-task learning to compare the performance in each of the model.

3.3.1 Stabilization of prediction

One difference between music and language learning is time notion: our model has to predict chords and the time to switch from one to another, which is irrelevant to an ordinary NLP tasks. With our language modeling approach to music chords matching, our model predicts a chord at each timestep (16th notes). Without any post-processing or augmentation, the model has a hard time deciding the best time for chord transition. As a result, we observe that the predictions are unstable at each beat and tends to switch at arbitrary 16th-note

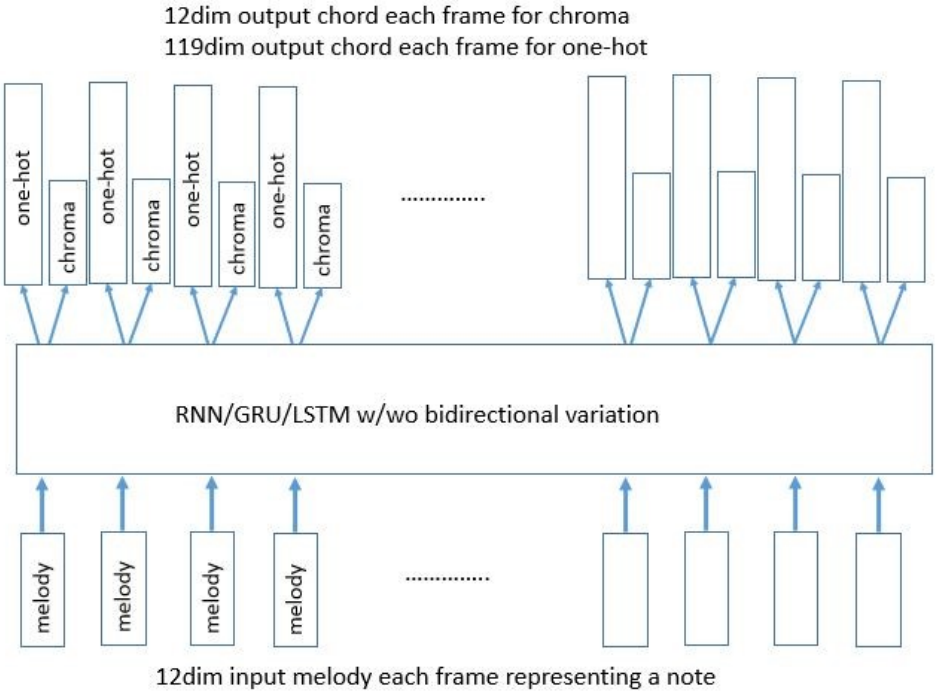

Figure 2: Multi-task learning depiction

timestep. To counteract such problems, we tried to augment our model by assigning different sample weight at each timesteps: we assign higher weight to the beginning of each beat and each measure, and let the error at these timestep contribute to a larger portion of training loss, in hope that our model picks up the importance of the time where chord transitions are likely to occur. Another technique is to post-process the prediction by merging every 8 timesteps by taking the average, and interpret the predictions to make a chord transition at every half note.

3.3.2 Multi-task Learning

We adopted a learning strategy to learn the relationship between melody and chord in both the chroma and one-hot representations. Intuitively this gives the system more information on how to improve the current chord selection.

Multi-task Learning is the learning strategy where the model learns at least two related tasks at the same time [Liu et al., 2015, Ando and Zhang, 2005, Evgeniou et al., 2005], where one task is the *primary* and the other tasks are the *secondary*. The model will be trained to perform both primary and the secondary task using a shared parameters. The idea is the model may generalize better to unseen data thanks to the shared structure of the tasks. Especially for cases where labeled data for the primary task is scarce, MTL can take advantage of the labeled data of the second task in improving performance for both[Yang et al., 2016].

By learning with multiple forms of output labels as the secondary tasks, MTL can possibly learn signal-level or music compositional level relationships among attributes, thereby improving the performance of the primary task [Yang et al., 2016]. Such an idea has been pursued in prior work, using for the example a multi-chain hidden Markov model (HMM) Ni et al. [2012] or a dynamic Bayesian network Mauch [2010].

In this project, we used a multi-task learning strategy where the primary task was learning from the one-hot representation and the secondary task was learning from the chroma representation of the chord progression.

3.4 Seq2seq

With language modeling approach we subscribe to an analogy of the melody to chord conversion to the machine translation problem. The seq2seq model was first proposed by Cho et al. [2014] and has proven very effective in tasks like machine translation, image caption, etc. The seq2seq model is handle sequence data of variable lengths, which makes it suitable for many tasks related to music.

In our context, however, we assume an equal length fro all the songs for simplicity. Therefore we did not make use of the variable length advantage. On the other hand, seq2seq is much harder to train and require a lot of data, which is why we ended experimenting with other network models.

4 Experiment setting

4.1 Dataset

Music informatics is a data-poor area, where most music are protected by copy right, leading to only few dataset available for research. Even more, it requires expensive human-labor to precisely label the chords and melody from a particular song. Tse et al. uses algorithm to align and generate 130,000 MIDI files from audio files. However, it's still very difficult to extract melody and chord from MIDI files due to the lack of clear tracks and beat information. We wrote hundreds lines of code attempted to parse the 300,000 MIDI, but still failed to improve our model.

Thus, we collected another datasets of MIDI files, each with 8 measures of melody and chords. Most of the clips are extracted from Western pop songs. We randomly sampled 10% clips as testing data set, and used the remainders as the training data set.

We partitioned MIDI files into a melody track and chord track. Due to arbitrary practice by music creators, this process was not as straightforward as one's intuition would suggest. We follow the principle of preferring precision in the classification and simply reject those songs that cannot be easily partitioned.

We parsed the data into 12-bin chroma vectors specifying the activation of the 12 pitch classes (C, C#, D, D#,... and so on) to represent the notes in a melody line or chord progression, disregarding tone height (i.e. octave number).

We use a sixteenth note (semiquaver) as the smallest time unit. For simplicity, we deal with music clips of 8 measures (128 16th notes) in this work, although this can be easily extended in future study. Therefore, 8-measure melody line or chord progression can be represented as a Boolean matrix of size 128×12.

For the model to better learn the chord, we experimented with two different chord representations, which are the chroma and one-hot representation. In the chroma representation, the vectors of the involved notes are simply added. The advantage of the chroma representation is its transparency. It is easy to define a distance metric, such as L1, based on its components. The one-hot representation, on the other hand, uses a one-hot vector for the $2^{12} = 4096$ possible chords. With one-hot representation, our model is not able to distinguish the similarities between chords, however, it allows us to reduce the dimension of our output by eliminating notes combination that does not makes musical sense more easily, and provides a more intuitive approach in making a prediction. We experimented both and the result shows that , as we will show in the results section below.

4.2 Implementation Details

Our implementation of the neural networks is based on keras Chollet [2015], a modular neural network library. We use mini-batch learning with 128 neurons and ADAM for updating the learning rates. To prevent over-fitting, we set the dropout rate Srivastava et al. [2014] to 0.5. The activation function inside the RNN is set to the hyperbolic tangent.

For chroma prediction, we use the sigmoid activation function to generate a real value between zero and one. As for one-hot prediction, we use softmax activation to generate a probability distribution. We used binary cross-entropy cost evaluation for chroma representation and categorical cross-entropy for one-hot representation.

4.3 Evaluation Protocol

We evaluated the results by separating data into training and test set by around 9 : 1 ratio. After a model under evaluation is trained on the training set, we evaluated it by its prediction on the test set. In particular we measured its the L1 error per frame and number of unique predictions (for proposed pair strategy only).

The *number of unique predictions* is useful as a secondary evaluation metrics for the proposed pair model. It counts the number of unique chord predictions for about 100 melody lines in the test set. Essentially it measures the diversity of the model, and the higher the diversity is, the better the model should be able to generalize to melodies of diverse style and genres. The results is shown in Figure-6.

For multi-task model, we measure the L1 error per frame for both of subtasks and plot this measurement as a function of the ratio of contributions from the two subtasks in an attempt to determine the optimal ratio. The result is shown in Figure-5.

5 Results and Discussion

Fig. 3 shows the stabilization results before and after we implement the sample-biased and merging every 8 timesteps (from a frame of 16th notes to half notes). From the MIDI piano rolls, one can see the MIDI chords are much smoother across the entire song after stabilization.

Fig. 4 shows the results for multi-task learning in language modeling, we only train chroma output when multi-task ratio is 0.0, and only train one-hot output when multi-task ratio is 1.0. We expect to see the U-curve loss plot. However, because one-hot loss and dimension is way larger than chroma ones, it's very difficult to balance the losses, and allow model to learn their correlation in shared weights. We also experimented with late-fusion ensemble learning on both output. Nonetheless, one-hot representation still dominate the whole prediction and thereby make no improvement.

6 Conclusion and future work

We surveyed a number of strategies to generate chords to a given input melody, including pair-model, language model, multitask-leaning, and sequence-to-sequence. Each of which has their own advantage over others and we implemented techniques to augment the strength and to remedy the weakness of each of the strategies. Pair model utilizes a lot of domain knowledge to restrict the chord progression to an existing song, and is less capable of generalization; language model provides freedom to generate any chord at any given timestep but lacks domain knowledge. Even though music and languages share a lot of similarities and the language model generates the lowest L1 training loss, the music generated by the pair model sounds more acoustically pleasing. This is because L1 is a close, but not absolute indication of how good a chord match to melody. For a single melody line, there can be many great chord progression for accompaniment. A simple L1 loss similarity against the original chord progression does not fully capture the quality of match by a predicted chord progression and can reject many perfectly good ones. Some chord progressions may sound great but are very different from the ground truth chord progression, which then suffers a poor L1 loss.

To further study this applications, we should first collect and sanitize more melody and chord progression data.

More research should also be performed on balancing the contributions of the subtasks in multi-task learning such that one does not dominate another and can learn from each other.

References

R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Machine Learning Research*, (6):1817–1853, 2005.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *arXiv preprint arXiv:1206.6392*, 2012.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

Mireille Besson and Daniele Schön. Comparison between language and music. *Annals of the New York Academy of Sciences*, 930(1):232–258, 2001.

Kyunghyun Cho, Bart van Merriënboer, and Dzmitry Bahdanau. On the properties of neural machine translation: Encoder-decoder approaches. October 2014. URL `https://arxiv.org/pdf/1409.1259.pdf`.

Franois Chollet. keras. `https://github.com/fchollet/keras`, 2015.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL `http://arxiv.org/abs/1412.3555`.

Cynthia Cohen. Music: A universal language. *Music and conflict transformation. Harmonies and dissonances in geopolitics*, pages 26–39, 2008.

Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, 103, 2002.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

T. Evgeniou, C. A. Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *J. Machine Learning Research*, (6):615–637, 2005.

Jonathan P. Forsyth and Juan P. Bello. Generating musical accompaniment using finite state transducers. In *16th International Conference on Digital Audio Effects (DAFx-13)*, 2013.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997. doi: 10.1162/neco.1997.9.8.1735.

X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proc. Conf. the North American Chapter of the Association for Computational Linguistics-Human Language Technologies*, 2015.

M. Mauch. *Automatic Chord Transcription from Audio Using Computational Models of Musical Context*. PhD thesis, University of London, Queen Mary, 2010.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Jonathan P. Forsyth, Rachel M. Bittner, Michael Musick, and Juan P. Bello. *Improving and Adapting Finite State Transducer Methods for Musical Accompaniment*. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 2015.

Y. Ni, M. Mcvicar, R. Santos-Rodriguez, and T. De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Trans. Audio, Speech, Lang. Proc.*, 20(6): 1771–1782, 2012.

Jay M Ponte and W Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281. ACM, 1998.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks - signal processing, ieee transactions on. 1997.

Dan Morris, Ian Simon, and Sumit Basu. Exposing parameters of a trained dynamic model for interactive music creation. In *Association for the Advancement of Artificial Intelligence*, pages 784–791, 2008.

Ian Simon,, Dan Morris,, and Sumit Basu. Mysong: automatic accompaniment generation for vocal melodies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 725–734. ACM, 2008.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

Tim Tse, Justin Salamon, Alex Williams, Helga Jiang, and Edith Law. Ensemble: A hybrid human-machine system for generating melody scores from audio.

Mu-Heng Yang, Li Su, and Yi-Hsuan Yang. Highlighting root notes in chord recognition using cepstral features and multi-task learning. *Asia-Pacific Signal and Information Processing Association*, 2016.

Appendix

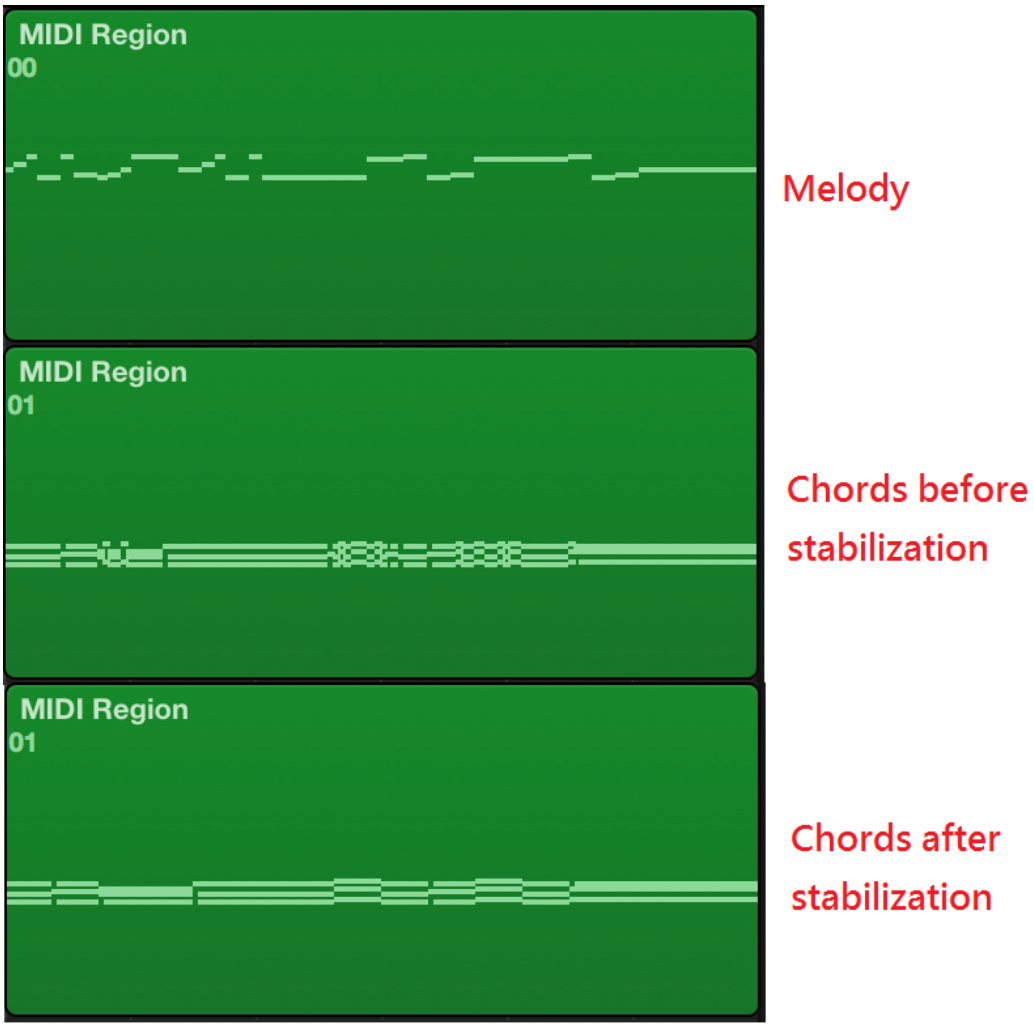

Figure 3: melody and chords prediction before and after stabilization

Figure 4: Multi-task Learning Performance plot. Where train1 and train12 refer to the training loss of one-hot (1) and chroma (12) subtask. err1 and err12 refer to the test error of the two subtasks.

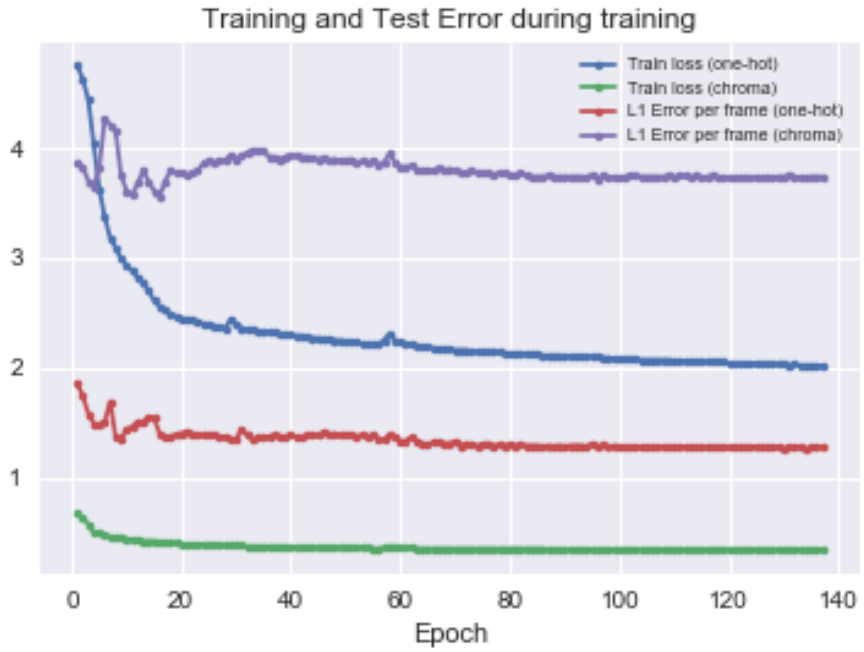

Figure 5: Loss value during training of multi-task learning model

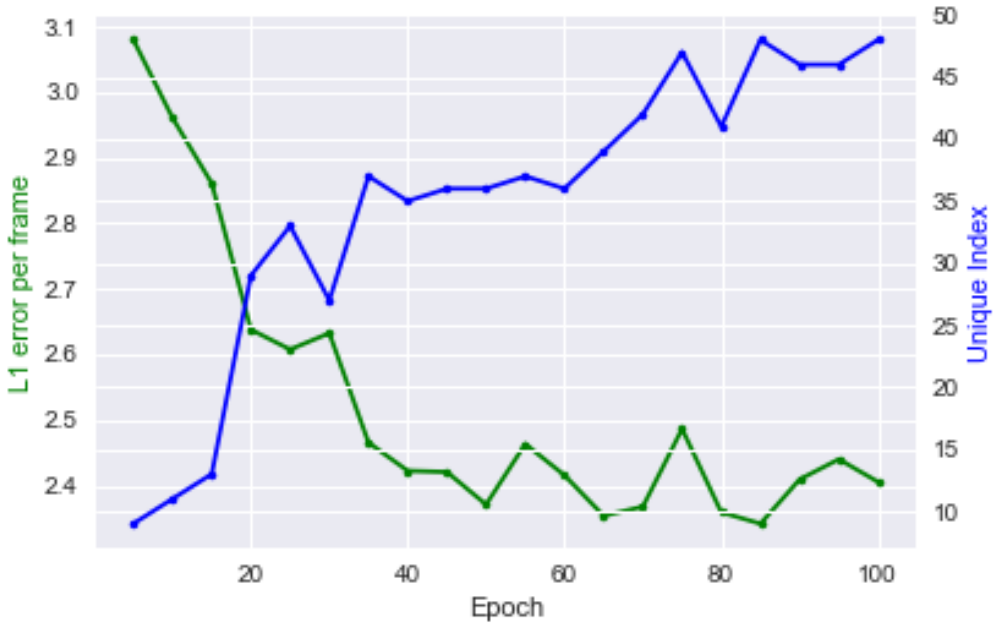

Figure 6: Proposed pair model norm and unique index