COMPARATIVE STUDY OF DEEP LEARNING MODELS FOR AUTOMATIC MUSIC TRANSCRIPTION

Nino Gotoshia

ABSTRACT

Automatic Music Transcription (AMT) is a fundamental problem in Music Information Retrieval (MIR). In this paper I explore early dual-model architecture consisting of acoustic and language models represented by a convolutional neural network with a goal of estimating probabilities of pitch in a frame of audio combined with a recurrent neural network capturing correlations of pitches overtime. I will be modifying said model with newer and more robust model architectures such as GRU and LSTM and comparing the results. I will also be analyzing and comparing these models to more advanced dual-objective architectures.

I. Introduction

Automatic music transcription (AMT) aims to create a symbolic representation from raw audio. This makes many tasks in musical information retrieval (MIR), such as searching for common chord progressions or categorizing musical motifs much easier, thus opening up opportunities for possible computational musicology studies. Automatic Piano Transcription for single note progression is considered to be a solved problem, it is for polyphonic musical pieces where we come across many challenges. Though polyphonic music transcription is a tedious task for both humans and computers, a well-trained human ear outperforms state-of-the-art models available today. However, these models show a lot of promise and potential for improvement as we continue to accumulate a large amount of training data, making them quite sufficient for the MIR tasks mentioned above.

Using deep learning for AMT was first proposed in [1], describing an architecture analogous to speech recognition systems. Convolutional Neural Networks (CNN), following their success with Image classification problems, have been used with two-dimentional time-frequency representation of audio. Though CNN alone has been used for AMT, as authors of [1] show, it is best utilized combined with a Recurrent Neural Network (RNN) language model as is the common approach for speech recognition. The authors also combine the recurrent neural network with Neural Autogressive Distribution Estimator (NADE) which is a distribution estimator for high dimensional binary data [2].

Though CNN-RNN-NADE is able to outperform any of its consisting models alone or combined, It is possible to base upon it a different model with improved performance while keeping the core acoustic-language architecture intact. I propose using a more advanced recurrent neural models as a language model such as Long Short-Term Memory (LSTM) [3] or a Gated Recurrent Unit (GRU)[4] . As shown later, This architecture has yielded better results compared to previous RNN or RNN-NADE architectures.

Finally, I will be comparing my model to a more advanced architecture for AMT. The dual objective model proposed in [5], combines two CNN-LSTM based networks, one of them detecting onsets of a piano note as its amplitude is in its peak. Results of Onset detection are fed into frame-wise note detection network. The two are trained together. See Fig 2.

II. RNN models

A recurrent neural network (RNN) is an extension of a conventional feedforward neural network, which is able to handle sequencial input by having a recurrent hidden state whose activation at each time is dependent on that of the previous time. Formally at time t RNN performs an operation:

$$h_t = f\left(W_{hx} h_l^t + W_{hh} h^{t-1} + b_l\right) \tag{1}$$

$$y_t = W_{hy} h_t \tag{2}$$

In 1 W_{hx} is weight matrix from input to hidden units and W_{hh} is weight matrix for recurrent connection and W_{hy} is weight for the output. As is with all neural networks, the parameters for RNN $\Theta = \{W_{hh}, Whx, Why\}$ are calculated using backpropagation which would involve Multiplications by many times, which could result in either very large, or very small results depending on the Largest significant value of W, called exploding or vanishing gradients. Though exploding gradients can be solved by establishing Thresholds during backpropagation, there is not such easy fix for vanishing gradients. Thus a new models were proposed.

A. LSTM

In short LSTM can be described as[1]:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-2} \\ x_t \end{pmatrix} \tag{3}$$

1 Note that this is not matrix multiplication. It describes the sequence of operations required to get the four gates of the unit.

$$c_i = f \cdot c_{t-1} + i \cdot g \quad (4)$$

$$h_t = o \cdot \tanh(c_t) \quad (5)$$

Thus in (3) we have a new state c_t, which we will call internal cell state and we use $ifog$ gates to update the internal cell state. By this we are only exposing part of the internal state to the hidden state h_t. On a really high level, these specific gates serve certain purposes.

f – forget gate (whether to erase the cell, using sigmoid between (0,1)) $f_t = \sigma(W_f x + t + U_t h_{t-1} + V_f c_{t-1})$

I – input gate (Whether to write the cell, using sigmoid between (0,1)) $i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})$

g – gate gate (new cell state content, using tanh between (-1, 1)) $g_t = \tanh(W_c x_t + U_c h_{t-1})$

o – output gate (how much to reveal the cell, using sigmoid between (0,1)) $o_t = \sigma(W_o x_t + U_o h_t - 1 + V_o c_t)$

In figure 1.a considering the backwards pass, it is visible that from c_t to c_{t-1} we no longer multiply by W^t each time, we only have element-wise multiplication by f with different f gates at each time step. Thus eliminating the problem of exploding/vanishing gradient.

B. GRU

Like LSTM, GRU has units modulating flow of information. It however does not have separate cell states therefore it uses the hidden state to capture dependencies of different time scales. At tome t, GRU does the following[4]:

$$h_t = (1 - z_t) h_{t-1} + z_t \tilde{h}_t \quad (6)$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \quad (7)$$

Where $\tilde{h}$ is the candidate activation and z_t is an update gate deciding how much the unit updates its activation. Candidate activation is computed similar to vanilla RNN unit:

$$\tilde{h}_t = \tanh(W x_t + U(r_t \circ h_{t-1})) \quad (8)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \quad (9)$$

Where r is a set of reset gates which are multiplied element-wise to the previous state

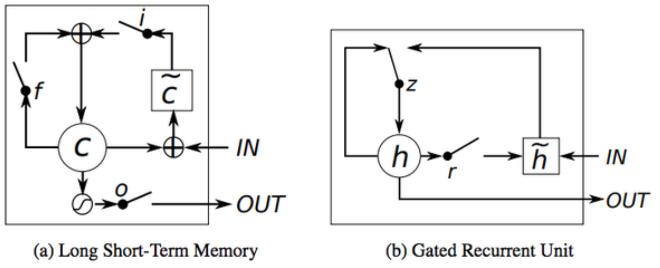

Fig. 1. LSTM and GRU units

C. Comparison

One obvious difference between the two models is the output gate in LSTM, which controls how much of memory is exposed to other units in the network. GRU unit exposes its full content. The LSTM unit also has separate input and forget gates, while the GRU performs both of these operations together via its reset gate.

It is difficult to conclude which of these models might perform better for a specific task on these differences alone, therefore it comes down to empirical observations to decide which will yield better results. It has been shown in [9] that with music modeling problems GRU tends to be faster in performance, but the overall outcome depends heavily on the dataset being used.

III. Dataset and metrics

A. Dataset

As described above, the main objective for a deeply learned AMT model, is to take as an input an audio file and output a symbolic musical representation (such as MIDI). Given this, it makes sense to have the Dataset be comprised of audio files with corresponding midi files to be used as ground truth labels. Only Such dataset available for a while was MAPS dataset consisting of over 200 classical pieces. This dataset was also used by both [1] and [5].

For my model I used a much larger MAESTRO dataset [6]. Though due to machinery constrains I was able to only utilize the part of data, more precisely 170 pieces for training and 32 pieces held out for testing. This matches the amount of data used by previous studies and could be argued to be fair while comparing the resulting models.

B. Metrics

Two types of metrics are used to evaluate proposed models: frame based and note based. F1, precision and recall are used for both frame and note based evaluation. These metrics for frame based evaluation are used as presented in [10] and can be defined as:

$$p = \frac{\sum_{t=1}^{T} TP[t]}{\sum_{t=1}^{T} TP[t] + \sum_{t=1}^{T} FP[t]}$$

$$R=\frac{\sum_{t=1}^{T} TP[t]}{\sum_{t=1}^{T} TP[t]+\sum_{t=1}^{T} FN[t]}$$

$$F1=\frac{2*P*R}{P+R}$$

where TP[t] is the number of true positives for the event at time t (e.i a single frame), FP is the number of false positives nd FN is the number of false negatives. Note-based metrics can be defined. A note event is assumed to be correct if its predicted pitch onset is within a ±50 ms range of the ground truth onset. For the latter metric I use the mir_eval library. Another note metrics is also used that considers note offsets within same ±50 ms range. As it is shown later this metric yields worst results for all the models explored in this paper.

IV. Representation And Proposed Models

A. Preprocessing

The first step is to transform audio into time-frequency representation. As is the common practice, this is done using Constant Q transform[7]. CQT is well suited for musical signals as it provides frequencies on a logarithmic scale (as are musical signals as well). It should be noted that CQT is a standard technique for AMT preprocessing, however the advanced dual-objective model in [5] uses log Mel-Spectogram which should be considered when comparing the results of the model to others. Following the steps done by other studies, audio is downsampled to 16kHz and CQT is computed over 7 octaves with 36 bins per octave and a hop size of 512 samples, resulting in a 252 dimensional input vector of real values, with a frame rate of 31.25 frames per second. In [1] resulting data was scaled by subtracting the mean and dividing by standard deviation across each dimension. Though this scaling method was tested for my model as well, significantly better results were given when each dimension was rescaled into range (0,1) which is calculated as follows:

$$z_i=\frac{x_i-min(x)}{max(x)-min(x)}$$

I sample the MIDI ground truth transcriptions of the training data at the same rate as the audio (32 ms) and obtain sequences of 88 dimensional binary vectors for training the models. The 88 outputs correspond to notes A0-C8 on a piano.

B. Models and Training

Preprocessed 2D frames were fed into CNN in a context window of size 7 with central frame being the target one. For this, beginning and end of the audio was zero padded. The objective is for CNN to output data that will be fed into LSTM. Since the LSTM has to be trained on a sequential data, we have to stack several CNNs for each input frame(in a context window). I use sequences of 100. the simulation for temporal input data to CNNs with adam optimizer was done using Kera's Time_Distrubuted wrapper for layers. The CNN consists of two Convolutional layers with 0.5 dropout and and MaxPooling layers in between, consisting of 64 and 128 units. Outputs of CNN should feed into a language model. As state preciously I constructed versions two language models, LSTM and GRU. Each of them bidirectorial consisting of 128 units, outputting to an 88 unit, sigmoid activated output later corresponding to the pitches taken at the particular frame.

Training was done on FloydHub platform on Tesla V100 GPU using Keras library. Both LSTM and GRU based models were trained on 70 pieces for 10 epochs. Both models took around 5 hours but GRU model was slightly faster, which is expected.

V. Results and comparison

For metrics we use precision, recall and F1 as described in Metrics. Test set consisted on 32 pieces not appearing in training data. It should be noted that each metric was calculated for a single piece and averaged out to get the final score. This is how it was done for [5] so it makes sense to do so for my models as I will be comparing the results of the two.

The scores for CNN-LSTM and CNN-GRU based models were the following:

CNN-LSTM	Precision	Recall	F1
Frame	75.52	75.14	75.00
Note	63.78	47.85	54.00
Note w/ offset	24.57	18.46	20.90

CNN-GRU	Precision	Recall	F1
Frame	80.87	66.05	70.67
Note	77.29	49.01	58.97
Note w/ offset	33.80	20.99	25.54

As seen above, GRU based model outperformed LSTM in all categories other than frame recall. Though exact reason behind why this is the case is hard to pinpoint, the results could provide intuition for the type of transcriptions that will result from the two models. Since precision is the measure of false positives, its is expected of GRU based transcription to be much "cleaner", containing fewer extra notes taken compared to the LSTM based model. However, due to its lower recall score, the models will probably miss more notes than its LSTM counterpart.

[5] Provides precision, recall and f1 scores for its proposed model as well as scores for several other models, including the CNN-RNN-NADE presented in [1] :

	Frame			Note			Note w/off		
	P	R	F1	P	R	F1	P	R	F1
Sigtia et al., 2016	71.99	73.32	72.22	44.97	49.55	46.58	17.64	19.71	18.38
Kelz et al., 2016	81.18	65.07	71.60	44.27	61.29	50.94	20.13	27.80	23.14
Melodyne	71.85	50.39	58.57	62.08	48.53	54.02	21.09	16.5	18.40
Onsets and Frames	88.53	70.89	78.30	84.24	80.67	82.29	51.32	49.31	50.22

As seen from this table, the Onsets and Frames dual-objective model severely outperforms rest of the models, except perhaps with frame recall score where it comes in a close second. It can be seen from this chart that both my models perform quite well compared to the remaining three architectures. On F1 score alone, LSTM based model beats out 3 out of 4 models for frame-wise evaluation, while the GRU model does the same for note-wise evaluation. Frame wise recall for LSTM model is in fact higher than all other models while the GRU model excels in both note and frame based precision and scores second after the onsets and frames model. It is also worth noting that in terms of note-with-offset evaluation GRU model has 3 out of for beat for every metric other than recall, which is shown to be relatively high for [11].

In terms of pure ear observations, LSTM and GRU models do a good job capturing harmony and melody but seem to have issues with rhythm. I believe this is one of the problems, among others, that the onset detector in dual-objective model might tackle, as is expected from a separate model specifically targeted towards note onsets. This does in fact seem to be the case judging from the Onset and Frames precision, recall and F1 score and what is said of the ear observations in [5].

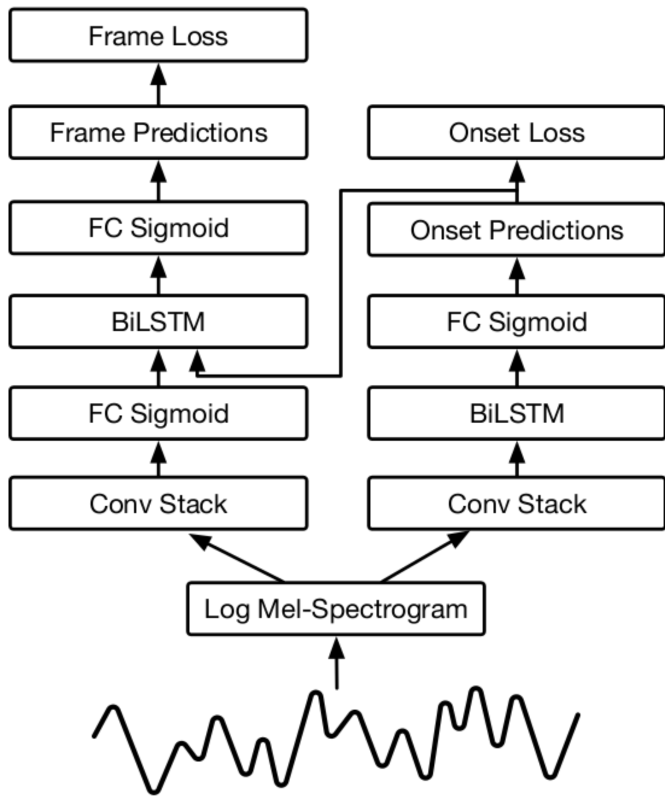

Fig. 2. Dual-objective “Onsets and Frames” model.

VI. Conclusions and future work

In conclusion, modifying the classical acoustic-language model consisting of CNN and vanilla RNN with more advanced models such as LSTM or GRU results in increase in performance even if the former RNN is combined with another non-neural model, such as NADE. Though the modified models fall behind the dual-objective Onset and Frames architecture, they seem to show great promise in terms of certain metrics such as both note and frame-wise precision, meaning that resulting transcription might be more modest but significantly cleaner in terms of extra notes taken. This is especially true for the GRU based language model.

The models presented by this paper were trained on around 170 audio files and tested on 32, reasons being machinery constraints for one, but also consistency with the amount of data other models were trained on. It would be interesting to see these models trained on full MAESTRO dataset to see if there is any improvement in their performance on pure dataset increase alone. It would also be interesting to see how these models perform with non-classical pieces, such as jazz as the two tend to differ on many aspects of harmony. Ideally this would require audio-midi dataset of jazz music so performance could be measured accurately with standard metrics.

VII. Referances

[1] Sigtia, S., Benetos, E. and Dixon, S. (2016). An End-to-End Neural Network for Polyphonic Piano Music Transcription. IEEE/ACM Transactions on Audio, Speech, and Language

[2] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” in International Conference on Artificial Intelligence and Statistics, 2011, pp. 29–37

[3] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735-1780

[4] Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.

[5] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, “Onsets and frames: Dual-objective piano transcription,” CoRR, vol. Abs/1710.11153, 2017.

[6] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset." In International Conference on Learning Representations, 2019

[7] Brown, J. (1991). Calculation of a constant Q spectral transform. The Journal of the Acoustical Society of America, 89(1), pp.425-434.

[8] E. H. Miller, “A note on reflector arrays,” *IEEE Trans. Antennas Propagat.,* to be published.

[9] Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. ArXiv, abs/1412.3555.

[10] Mert Bay, Andreas F Ehmann, and J Stephen Downie. Evaluation of multiple-f0 estimation and tracking sys- tems. In ISMIR, pages 315–320, 2009.

[11] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. arXiv preprint arXiv:1612.05153, 2016.