AT4rAMT:
Adversarial Training for robust Automatic Music Transcription

Andreas Streich, Heman Tanos, Malte Toetzke, Sebastian Windeck
Deep Learning Lecture, Fall 2018

Abstract

Automatic Music Transcription (AMT) aims to transcribe acoustic signals into discrete tone values (pitches). AMT is a classic and difficult task within the area of Music Information Retrieval (MIR) and has attracted much attention, also from the machine learning community. In this project, we start from a previously published network architecture and train a base model on clean data. Afterwards, we use adversarial training to iteratively retrain the model on noisy data. Doing so, we adaptively choose a noise level such that the task becomes more difficult, but not too difficult for the model. To do so, we pre-evaluate the performance of the current classifier and modify the noise level until the noise-induced performance decrease lies in a pre-specified range. We then use that noise level to generate noisy data for the next training round.
With this type of adversarial training, we obtain a model that clearly outperforms the base model on noisy data, and also has a more steady performance as the noise increases. We observe that the transcription performance, measured by the F1 score, remains almost at the level achieved on clean sound, even as the signal-to-noise ratio decreases to conditions that render transcription very difficult for humans.

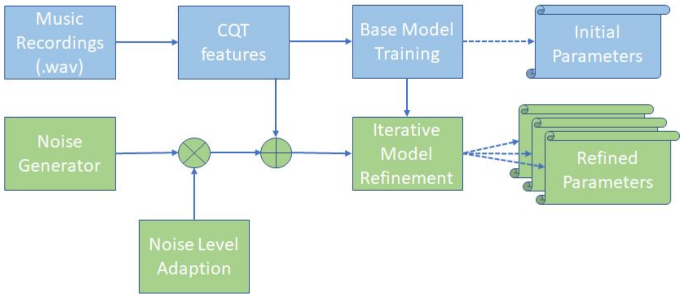
Introduction

Automatic Music Transcription (AMT) is a core discipline in the area of Music Information Retrieval (MIR). AMT aims to transcribe acoustic signals into discrete tone values (pitches). At this, tone values can be restricted to single pitches (monophonic) or entail multi pitches (polyphonic). Filtering and extracting sequences of multipitches makes AMT a complex problem for both human experts and machines [1].
A wide choice of machine learning methods has been applied to this task, including support vector machines (SVMs) [2] and non-negative matrix factorisation (NMF) [3]. Recently, neural networks were employed to make the classic step of feature engineering obsolete. Popular models are recurrent neural networks (RNNs) [4], deep belief networks (DBNs) [5], deep feed-forward neural networks (DNN) [6], possibly combined with additional classifiers on top [5], and convolutional neural networks (CNNs) [7].

AMT systems are typically highly sensible to noise comprised in the acoustic input signal (e.g. background noise during recording). We investigate whether a training under noisy conditions can improve the robustness of AMT systems to noise.
Training under adverse conditions, called adversarial training, is probably best known in image classification after the seminal paper by Goodfellow et al [8]. Szegedy et al. [9] have trained an adversary to fool an image object recognition system classifier by perturbing the input. Kereliuk et al. [10] have transferred this approach to music genre classification and show that the adversary network effectively inserts minimal perturbations that cause errors, but failed at training the classifier to become more resilient to these perturbations. While they used a fixed signal to noise ratio (SNR), we take a different approach here and add noise only gradually such that the AMT performance does decrease to some extent, but not too much.

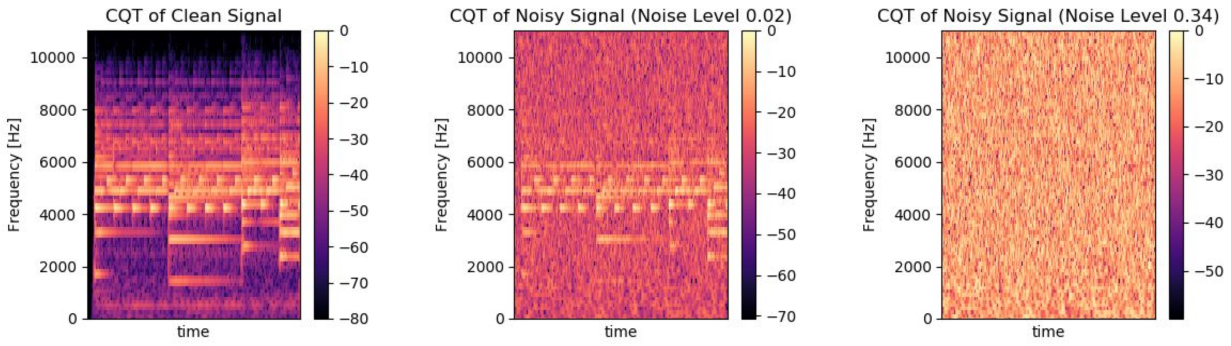
Models and Methods

Our model is implemented in python, the neural network is modeled with the `keras` package. The overarching structure of our model is depicted in the figure on the right. In the following, we go through the main building blocks.



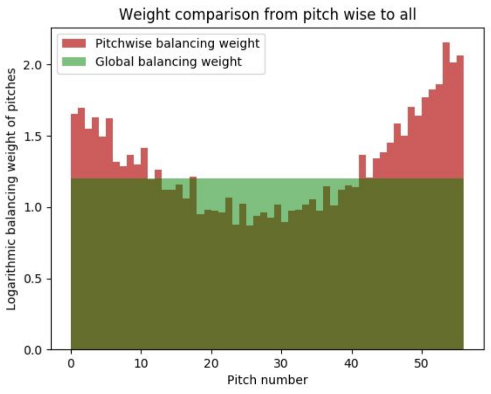
Data set: We use the open available MAPS dataset [11] for our experiments. This dataset consists of audio and corresponding transcriptions for isolated sounds, chords and complete pieces of piano music sampled at a rate of 44'100Hz. The files represent approximately 65 hours of music in stereo quality and its transcription with high temporal precision. For the analysis we use the complete pieces only.

Feature extraction: We use the constant Q transform (CQT) as input representation. CQT is well suited for time-frequency representation, as its frequency spacing is linear in the pitch. Furthermore, it results in a rather low-dimensional representation. We use the CQT implementation provided by the `librosa` package [13]. We use a sampling rate of 16 kHz and a hop length of 512, resulting in a feature vector every 32 ms, or a *frame rate* of approx. 31 Hz.
We allocate 3 bins per pitch and limit ourselves to the frequencies between approx. 70 Hz (corresponds to the tone C#2) and 1700 Hz (G#6). This allows us to reduce the amount of training data to a vector of length 168 per frame, and yields a more balanced data set, as the very low and very high frequencies are played much less frequent that the once taken into consideration here. Furthermore, we have introduced a threshold on the minimal volume, and we ignore frames in which no true labels are given (i.e. silence). Finally, as subsequent frames are highly correlated, and in order to reduce the data amount to a quantity that can be stored on the Leonhard cluster, we take only every second frame. A representation of the CQT features as a heat map over time is given on the right.
The feature extraction is done at the beginning; noise is later added to the obtained CQT values. Heatmaps of the CQT spectrum of a clean signal (approx. 15 seconds) and noisy versions (with noise parameters 0.02 and 0.34) are shown below.



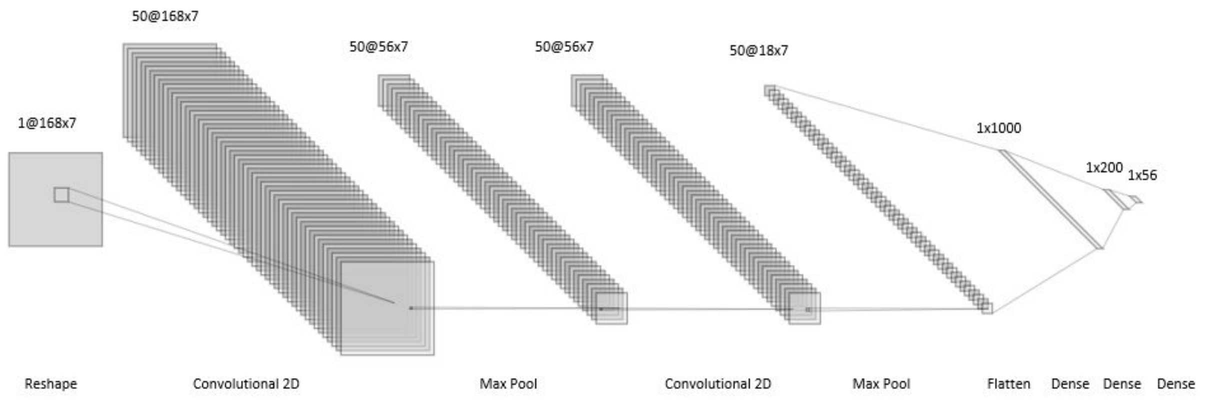
Target values: With the given frequency range, we get 56 notes that are considered independently here. Hence, we have effectively 56 independent binary classification tasks: at every time step, every note can either be played or not. Note that the classification problem is significantly imbalanced in that most of the notes are not played most of the time. This is taken into account by a specific loss function considering the individual balance of the pitch.

Loss function: We run experiments with two different loss functions. The first is the binary cross-entropy as provided by the standard implementation of keras with a global balance weight. The second one is a custom-made loss ratio, that takes into account the vast differences in how often the individual pitches occur. The weighting per pitch is chosen such that the frames with and without that pitch being played have equal weight over the entire training data set.



Metric: We use several metrics to assess the quality of the transcription results. As a single-number metric, we use the f1-score, given by twice the product of the precision and the recall, divided by the sum of precision and recall. Precision is the ratio between the true positive count and the count of all samples identified as true, while recall is the ratio between the true positive count and the number of samples with ground label true. We compute the precision and recall over all pitch values and over all frames (macro-averaging), and then determine the F1 value based on the overall precision and recall.
To provide more detailed insights on to the frequency-dependent performance, we use a confusion matrix per pitch value. Confusions are evaluated per frame, and frames are weighted by the ratio between the true number of pitches played at that time, and the estimated number.

Neural Network: The base model for our project is largely inspired by [7] and the code from [12] implementing that model. We use similar activations and regularizations to have a comparable result. The structure of the neural network is visualized below. The nomenclature of the convolutional layers describe the number of filters as well as the 2D array. For the input a sample has a CQT bandwidth and a certain number of windows created in the frequency extraction from the .wav files (sample@bandwidth x windows). The activation functions implemented in the convolutional layers are rectified linear units and sigmoid functions in the dense layers. Between each layer we use dropout layers to regularize the model, because of the reduced amount of data compared to the free parameters in the model. The base model is trained with 1000 epochs.



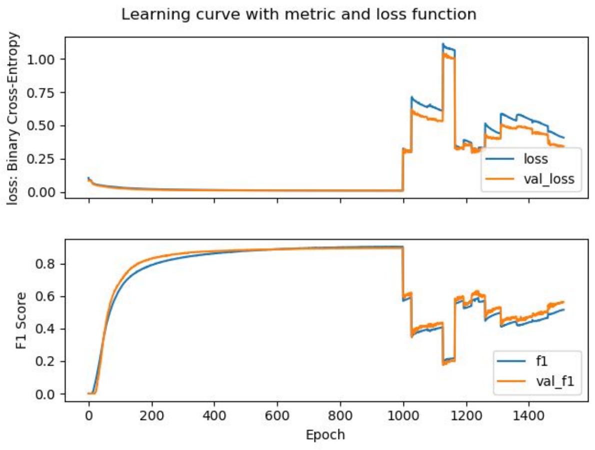
Noise Generation: We generate noise in the space of the CQT transform, which reduces the CQT transformation every round and thus allows us to conduct experiments more quickly. Also [12] apply perturbations directly in the feature domain (which is STFT in their case).

We have restricted ourselves to a simplistic noise generation consisting of uniform random numbers between 0 and 1. Other noise types, such as white (Gaussian), pink, brown and further coloured noises can easily be generated using the corresponding functions from the `acoustics` package.

Noise Level Adjustments: Unlike [10], we scale the noise samples with a varying factor, which is chosen such that the relative loss increase is within a predefined interval. To do so, we choose a subsample of the training data, and generate random noise for each of these samples. Then, the noise factor is increased (if the loss difference is to small compared to previous training data) or decreased (if loss difference is to high) until the loss difference is in the desired range. This noise level adaptation is given as pseudocode:

```
X_clean, Y = Choose N samples from the training data X
this_noise = generate noise samples in the same shape as X_clean
Initialize noise_level
while true
        X_noisy = X_clean + noise_level * this_noise
        delta_loss = (loss(X_clean)-loss(X_noisy))/loss(X_clean)
        if delta_loss > max_delta_loss # X_noisy is too hard
                noise_level = noise_level / noise_decrease_factor
                continue # jump to next loop cycle
        else if delta_loss < min_delta_loss # X_noisy is too easy
                noise_level = noise_level * noise_increase_factor
                continue # jump to next loop cycle
        else # found appropriate noise level
                break
# use noise_level to train AMT
X_noisy_full = X + noise_level * [noise of size of X]
Train AMT on X_noisy_full
```

Training under Noise: The training with the noisy data is done in the same way as the initial training for the base model, except that we allow a maximum of 50 epochs for each noisy training data set. Then, the noise level is adapted again the next training starts. These steps are repeated 12 times, we call each of them *noise round* or simply *round*.

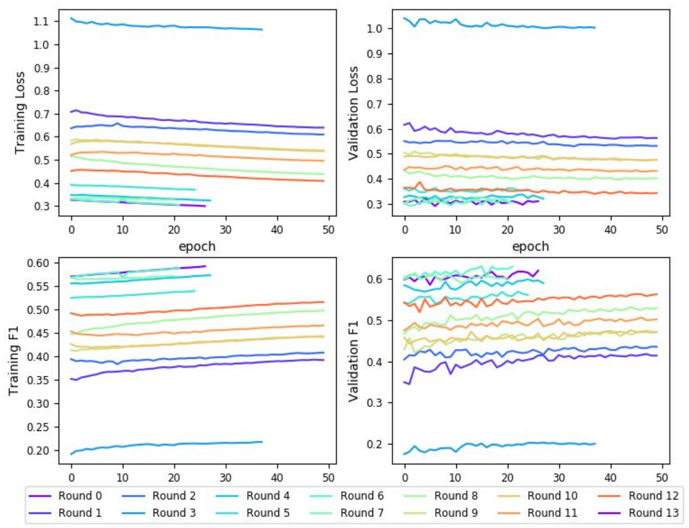


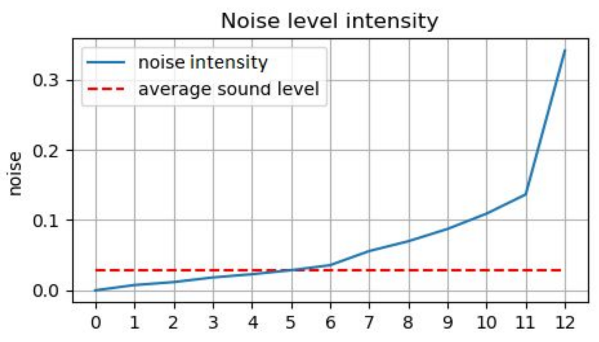
Results

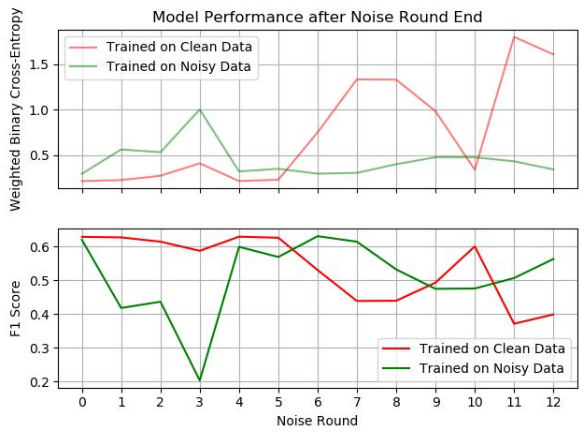
Learning Curve: The evolution of the loss (weighted binary cross-entropy) as well as of the f1 score over the epochs is plotted on the right, for both the training and the validation data set. The initial training of the base model on the clean data set shows nice, but somewhat slow convergence properties. We observe that the loss and the f1 score do not change significantly after the 400th epoch.

With the addition of noise after 1000 epochs, the loss rises by a factor of at least 10, while the F1 score decreases to about ⅔ of its value on the clean data (both for training and validation data set). As the training on the noisy data moves on, the performance rises again, but the gains in performance are by trend larger for the loss than for the F1 score.

Evolution within Noise Rounds: The plot on the next page shows the performance for each of the epochs. Surprisingly, the progress of the model seems to be very consistent, as the lines on the training loss (upper left) all are very similar.







Transcription Performance on Noisy Music: In the figures on the right, we compare the performances of the two models in the AMT task, the base model trained uniquely on the clean data, and the model which iteratively is re-trained with noisy data. As the plot on the right shows, the model that was trained on noisy data has both a lower weighted cross-entropy and a higher F1 score than the model that only saw clean data. Also, we observe that the performance of the base model is much more varying as the noise increases, while the model that was trained also on the noise performs at a much more stable level.
Hence, we have shown that adversarial training is beneficial for AMT training and yields superior transcription performance even at high noise levels. With an average sound level of 0.03 in the data set and a noise intensity factor of 0.34 (corresponding to an average noise level of 0.17), the noise level in noise round 12 corresponds to a signal-to-noise ratio (SNR) of -7.7dB, a condition under which humans have significant difficulty to identify the signal.

Discussion

We observe that the performance as measured by the loss function is much more affected than the F1 score. We interpret this as a less certain (i.e. less clear) transcription; the probabilities are moving towards higher entropy (the maximum entropy is obtained at the value 0.5). However, the F1 score shows that the performance of the rounded AMT results is much less affected.

Summary

We show that adversarial training renders automatic music transcription more robust to noise. Further research is need to understand the effect of the training on noisy data on the parameters of the network. Also, the effect of training with additional noise should be compared to other approaches to increase stability, such as regularization and drop-out layers. Finally, we suggest to investigate whether the noise effects are local in the frequency space, or whether e.g. noise in low frequencies might influence the performance in higher frequencies.
As for the adversary, we have only considered a simplistic version here, focusing our attention on the transcription network. An extension of the work to train an adaptive adversary is also suggested as further research.

References

[1] A. Klapuri and M. Davy, *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[2] G. E. Poliner and D. P. Ellis. *A discriminative model for polyphonic piano transcription*. EURASIP Journal on Applied Signal Processing, vol. 2007, no. 1, pp. 154–154, 2007.

[3] P. Smaragdis and J. C. Brown. *Non-negative matrix factorization for polyphonic music transcription.* In 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. IEEE, 2003, pp. 177–180.

[4] S. Böck and M. Schedl. *Polyphonic piano note transcription with recurrent neural networks*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012, pp. 121–12.

[5] J. Nam, J. Ngiam, H. Lee, and M. Slaney. *A classification-based polyphonic piano transcription approach using learned feature representations.* In Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR), 2011, pp. 175–180.

[6] S. Sigtia, E. Benetos, N. Boulanger-Lewandowski, T. Weyde, A. S. d'Avila Garcez, and S. Dixon. *A Hybrid Recurrent Neural Network for Music Transcription.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Brisbane, Australia, April 2015, pp. 2061–2065.

[7] S. Sigtia, E. Benetos, S. Dixon. *An End-to-End Neural Network for Polyphonic Piano Music Transcription.* In IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2016), pp. 927-939.

[8] I. Goodfellow, P. McDaniel, N. Papernot. *Making Machine Learning Robust Against Adversarial Inputs.* Communications of the acm, July 2016.

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

[10] C. Kereliuk, B. Sturm, J. Larsen. *Deep Learning and Music Adversaries*. IEEE Transactions on Multimedia, 2015, pp. 2059-2071.

[11] E. Bertin, B. David, R. Badeau. *MAPS-A piano database for multipitch estimation and automatic transcription of music*. 2010. Data set available at http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music/

[12] Jonathan Sleep. *Automatic Music Transcription With Convolutional NEural Networks using Intuitive Filter Shapes*. Master thesis at California Polytechnic State University, San Luis Obispo, 2017. Code available at https://github.com/jsleep/wav2mid.

[13] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, O. Nieto. *librosa: Audio and music signal analysis in python.* In Proceedings of the 14th python in science conference, pages 18-25, 2015. Code available at https://librosa.github.io/.