Piano music generation using GAN with LSTM units

Arvid Peterson, Linn Lyster

Chalmers University of Technology
Electrical Engineering Department

CHALMERS UNIVERSITY OF TECHNOLOGY

Introduction

Creating qualititative music is hard, even for humans. Can computers yet again come to our aid?

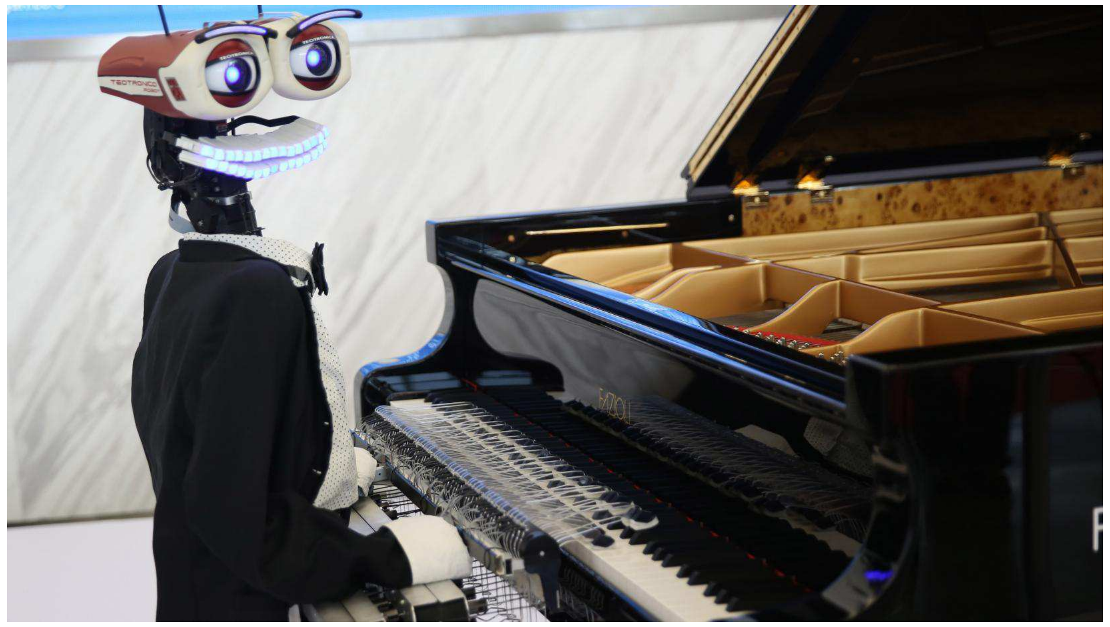

Figure 1: Digital composer replacing humans.

- Music quality hard to quantify
  - Needs knowledge of music rules and concepts
  - No recipe for pleasingly sounding music
- Generate music can be seen as sequential modelling
  - Intuition: Use recurrent neural nets
  - Problem: Hard to quantify similarity to real music
- Generative Adversarial Networks (GANs) used for generating new content
  - Uses generator and discriminator to find increased similarity with real data

Project goal: Combine RNNs to GANs in order to create pleasantly sounding music.

Implementation

Only pitch is taken into account. Assumed constant volume and offset between notes.

Network structure

What is GAN?

- Generative model *G* trained to transform a random noise *z* to follow distributions similar to the real data in order to trick a discriminating model *D*[1].
- The discriminator is trained to distinguish between generated data and real data.
- Goals:
  - Generator: Maximise the probability for the discriminator to make a mistake.
  - Discriminator: Minimise the classification error and successfully separate between generated and real data.

This adversarial conflict is what improves the network performance - both models try to beat each other, which drives the progress for each of them in getting better and better.

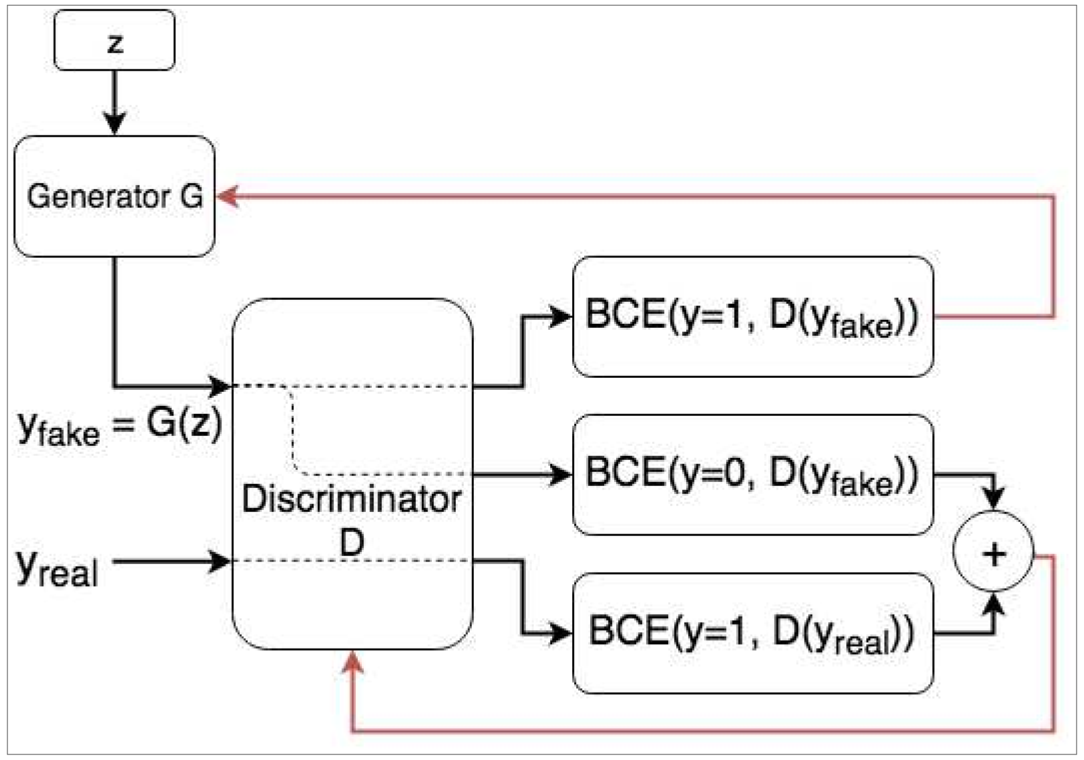

Figure 4: Schematics over how the GAN network is build up. Black arrows indicates transfer of data, while red arrows indicates how the loss is backpropagated. BCE stands for binary cross entropy loss.

Results

- ▶ Dataset: MAESTRO dataset with 200 h piano music in MIDI files
- ▶ Preprocessing:
  - ▶ Data is split into sequences of length 100
  - ▶ Then converted from notes to numerical representation
  - ▶ Data normalised to $(-1, 1)$
- ▶ Input to generator:
  - ▶ Random gaussian noise $z \in (-1, 1)$
- ▶ Structure of generator:

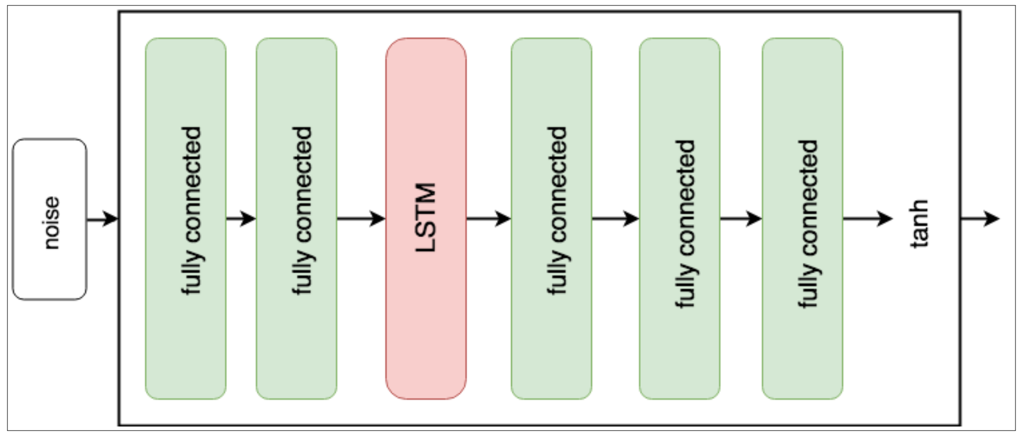

Figure 2: Generator *G*

- ▶ Input to discriminator:
  1. Output from generator: Fake sequence
  2. Real sequence from data
- ▶ Structure of discriminator:

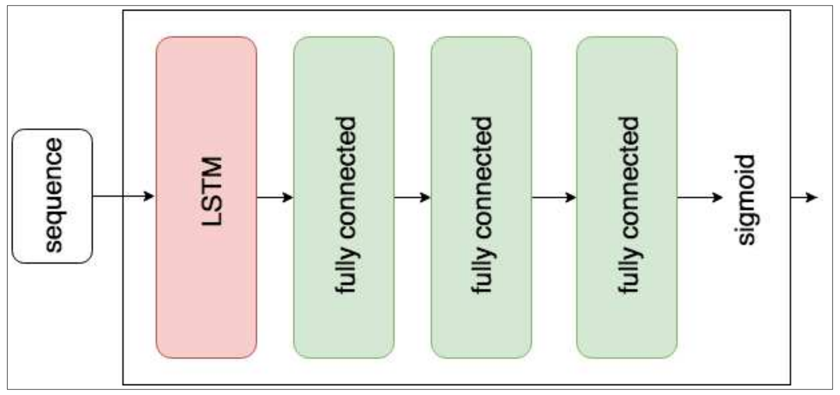

Figure 3: Discriminator *D*

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio.
Generative Adversarial Nets.
In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

- ▶ Code and generated MIDI files avalable at GitHub: `https://github.com/SysterLyster/DeepLearningProject-LSTM-GAN`
- ▶ Trained network can generate sequences with repeated musical figures and patterns.
- ▶ More work needed to create music that could be mistaken for a human composed music.

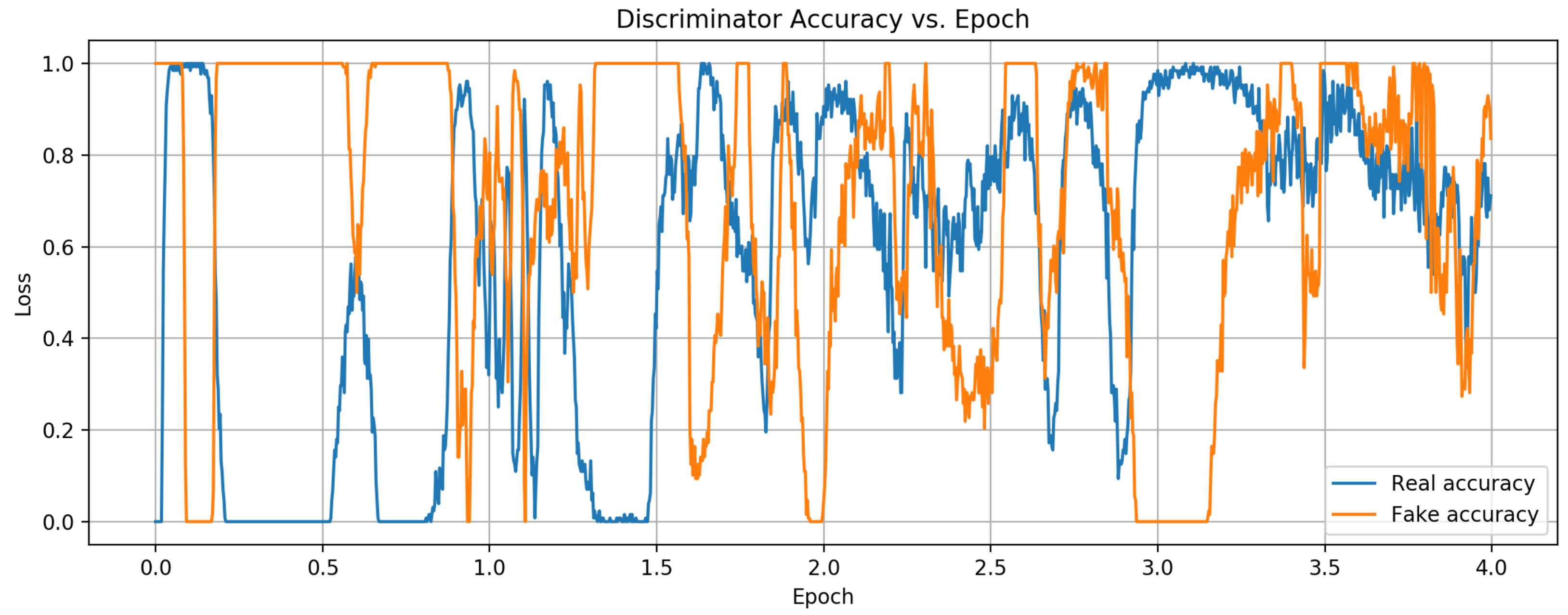

Figure 5: Discriminator accuracy on labeling generated (fake) and actual (real) piano MIDI music. We can see a clear correlation between the two classes. The discriminator being good at recognizing fake data leads to efficient training of the generator which in turn learns to fool the discriminator

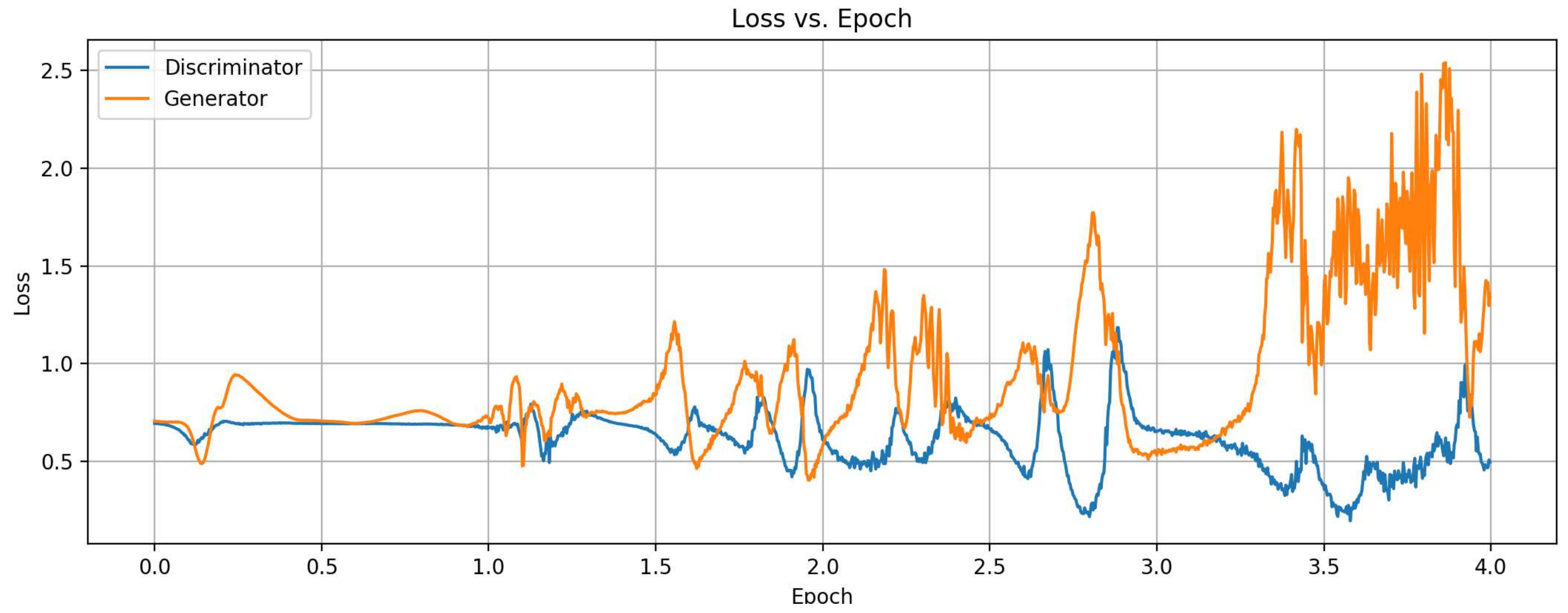

Figure 6: Training loss for both discriminator and generator networks. Similarly to figure 5 the loss is correlated as well for much the same reasons.

parvid@student.chalmers.se, lyster@student.chalmers.se