Music generation with GAN and RL

Ivan Anokhin, Egor Nuzhin

2018

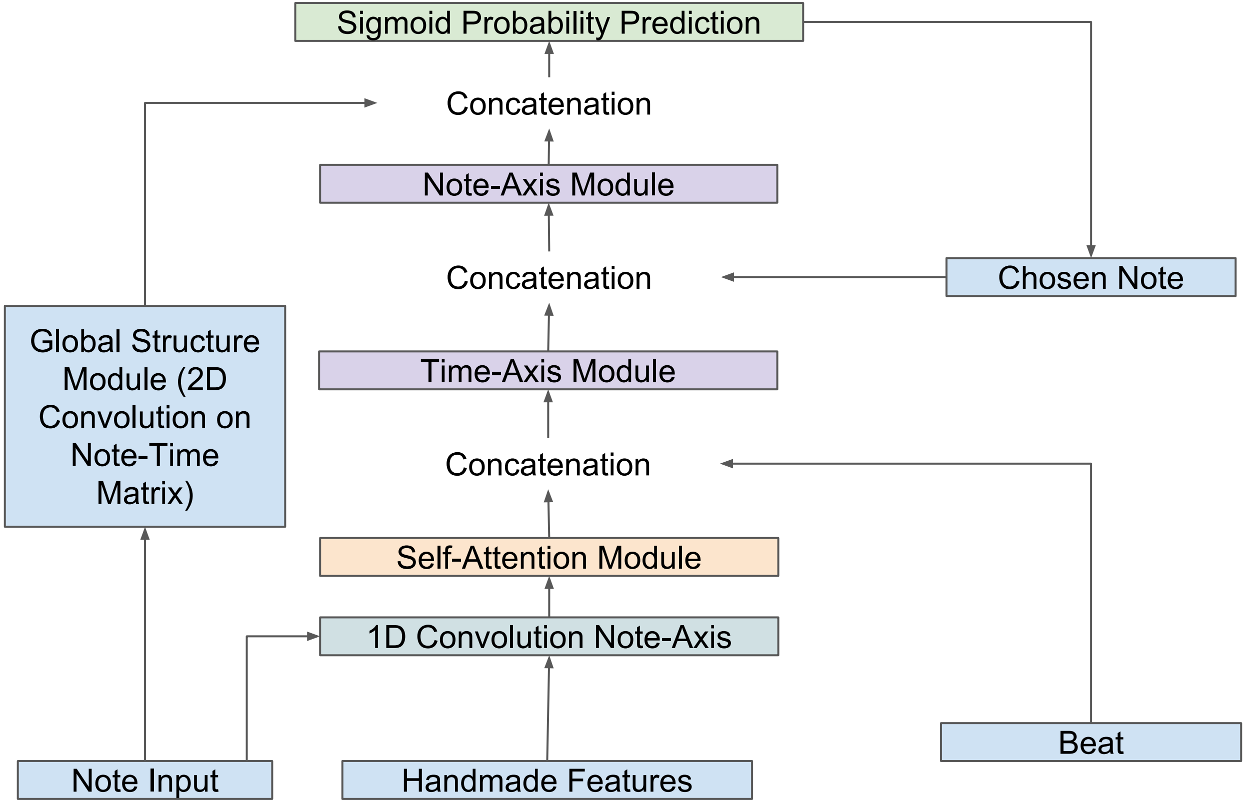
Introduction

- Listening to music is one of important part of our life
- Music is a simplest way to relax
- Music industry is one of the biggest market place it the world
- Musical composition are required in many different areas: cinema, games, ets.

Dataset description

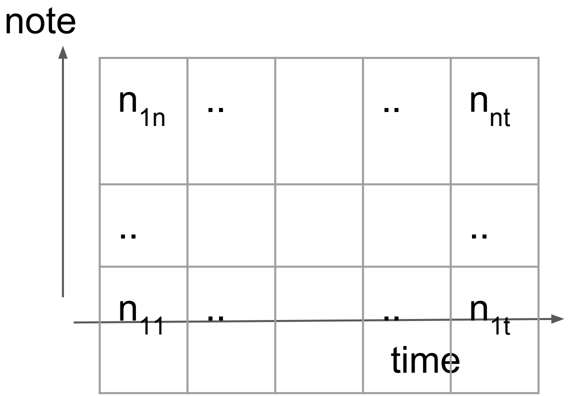
- We used open source dataset: The Lakh MIDI Dataset
- Transformed structure of midi file to use in model is:
  - Play matrix
  - Replay matrix
  - Volume matrix

Generator Architecture



Note input and Convolution Modules

1. Note Inpute - Note by Time Matrix with note embedding in each position



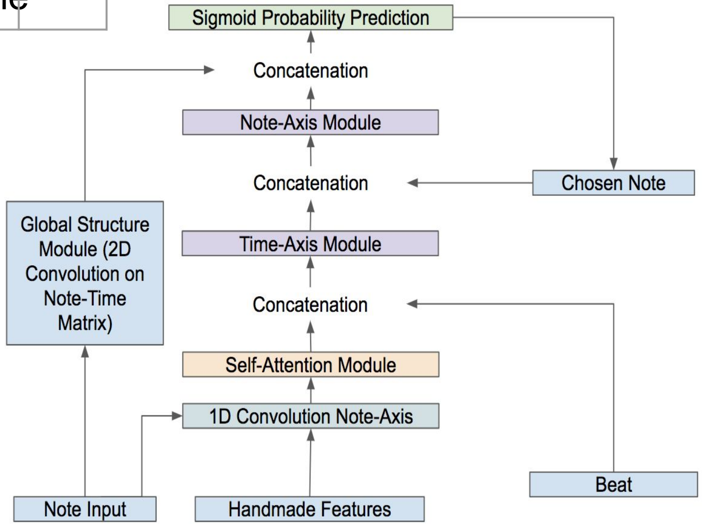
2. 1D Convolution Note-Axis (Kernel size = 24)

1D Convolution Note-Axis

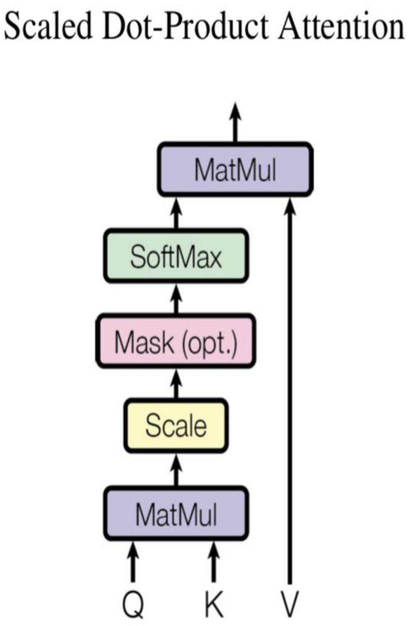
3. Global Structure Module Kernel size = (all note x 32 time steps), stride = 16

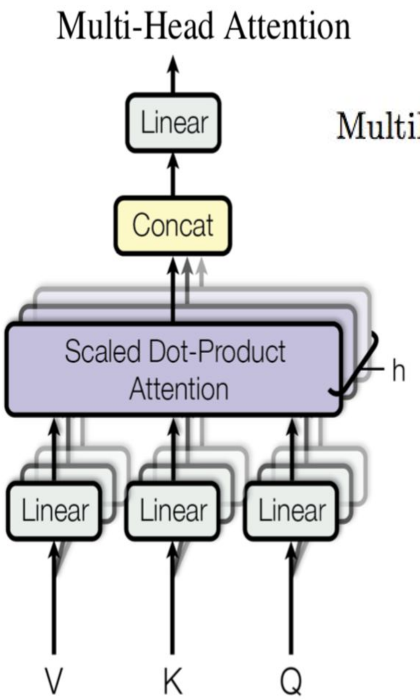
Global Structure Module (2D Convolution on Note-Time Matrix)

Note Input

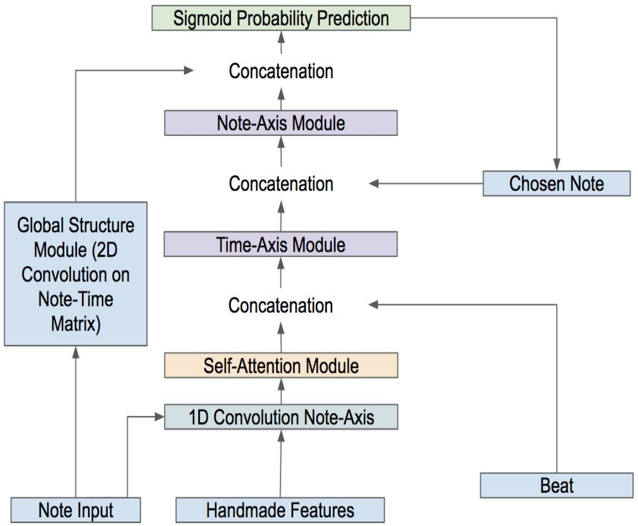


Self-Attention module





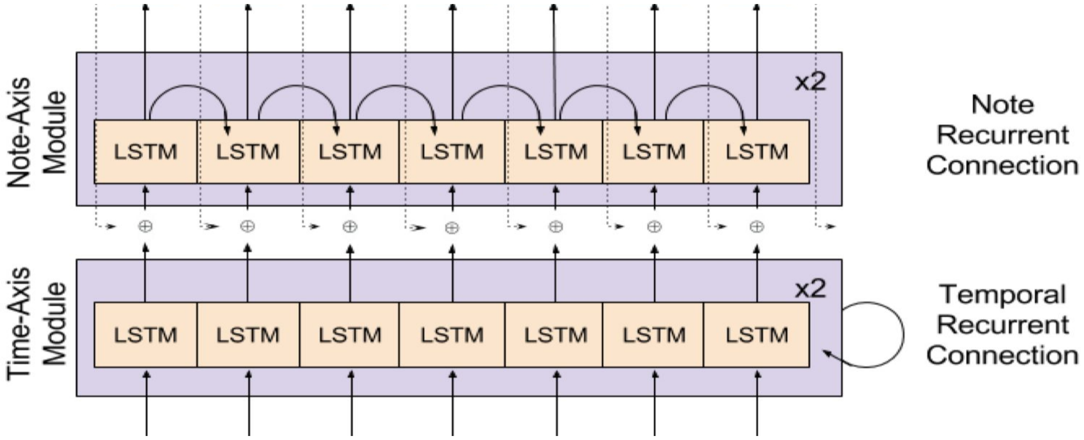
$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\ldots,\mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

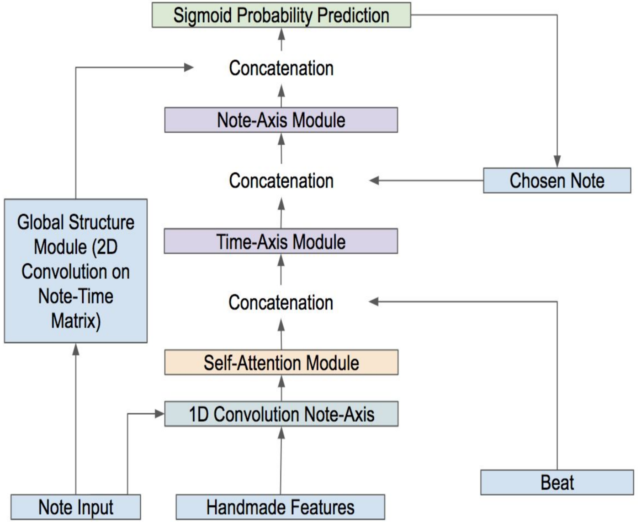


Self Attention Mechanism: When Q=K=V

Attention Is All You Need (2017)

Time-Axis and Note-Axis Modules

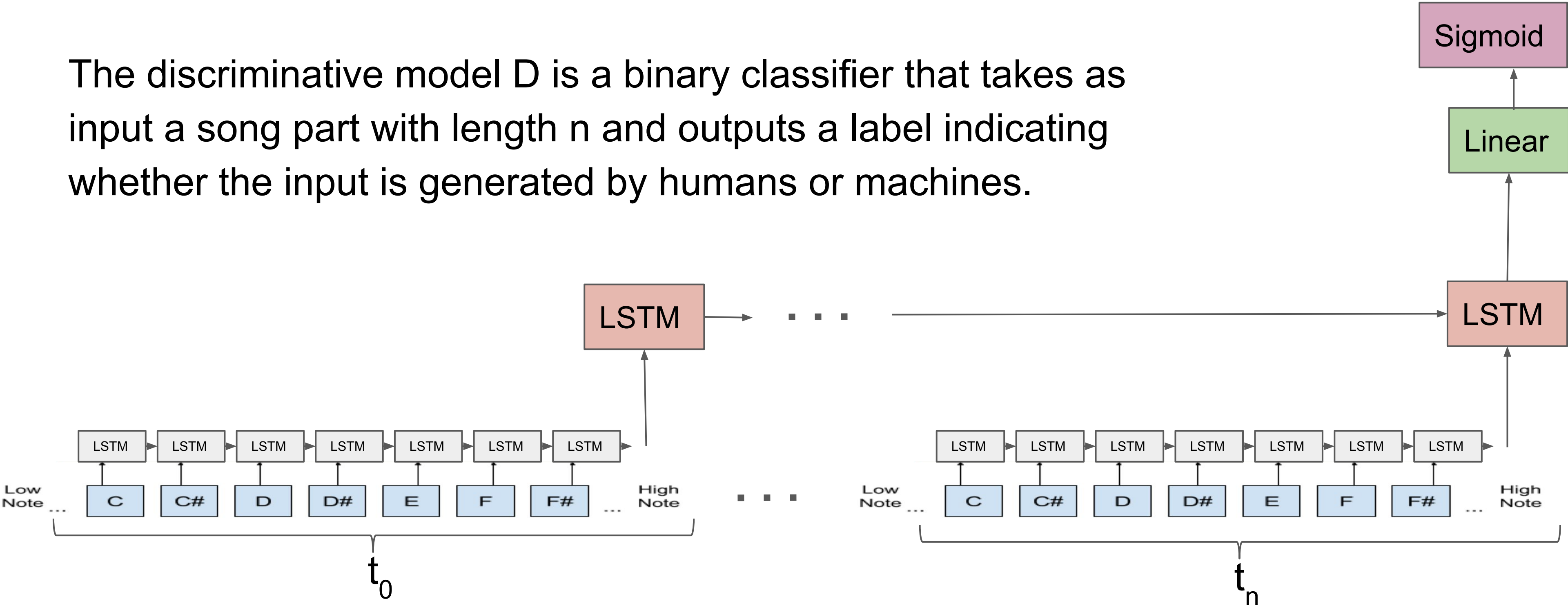




DeepJ: Style-Specific Music Generation (2018)

Discriminator hierarchical architecture

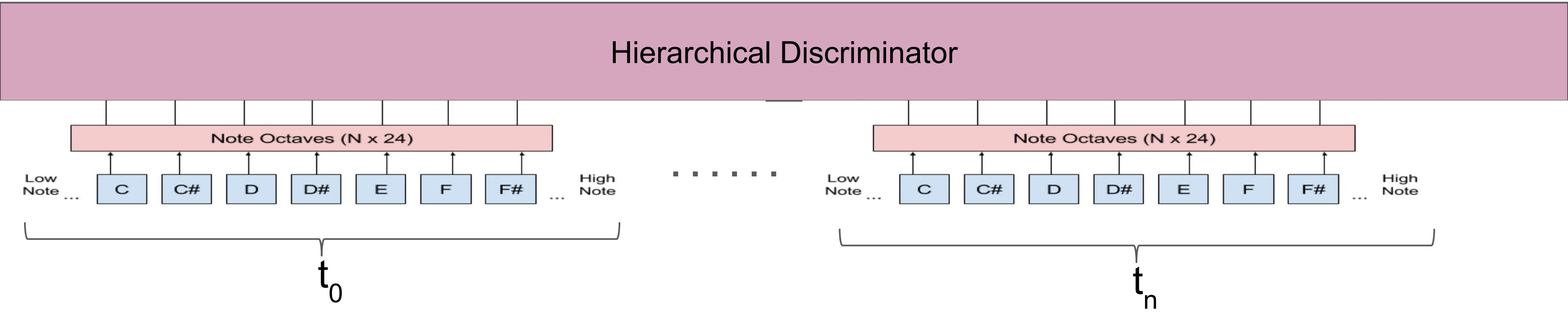
The discriminative model D is a binary classifier that takes as input a song part with length n and outputs a label indicating whether the input is generated by humans or machines.



A Hierarchical Neural Autoencoder for Paragraphs and Documents (2015)

Discriminator hierarchical architecture

As additional improvement we used Note Octave embedding



Discriminator inputs

Output of the generator after
creating full song sequence

Unconditional GAN

(0...0)

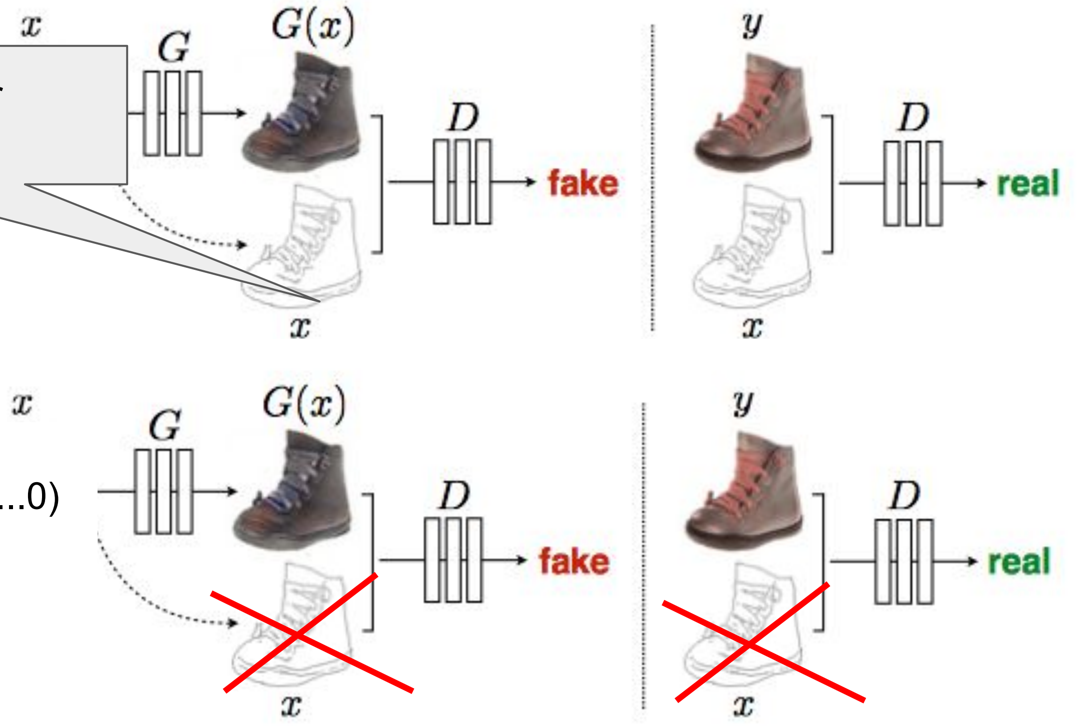


Image-to-Image Translation with Conditional Adversarial Networks (2017)

Learning technique. Algorithm

1. Update Discriminator n times
   a. Sample (X,Y) from real data
   b. Sample Y_G~G(·|X) from generator
   c. Update D using (X, Y) as positive examples and (X,Y_G) as negative examples.
2. Update Generator m times
   a. Sample (X,Y) from real data
   b. Sample Y_G~G(·|X) from real data
   c. Compute Reward r for (X, Y_G) using D.
   d. Update G on (X, Y_G) using reward r

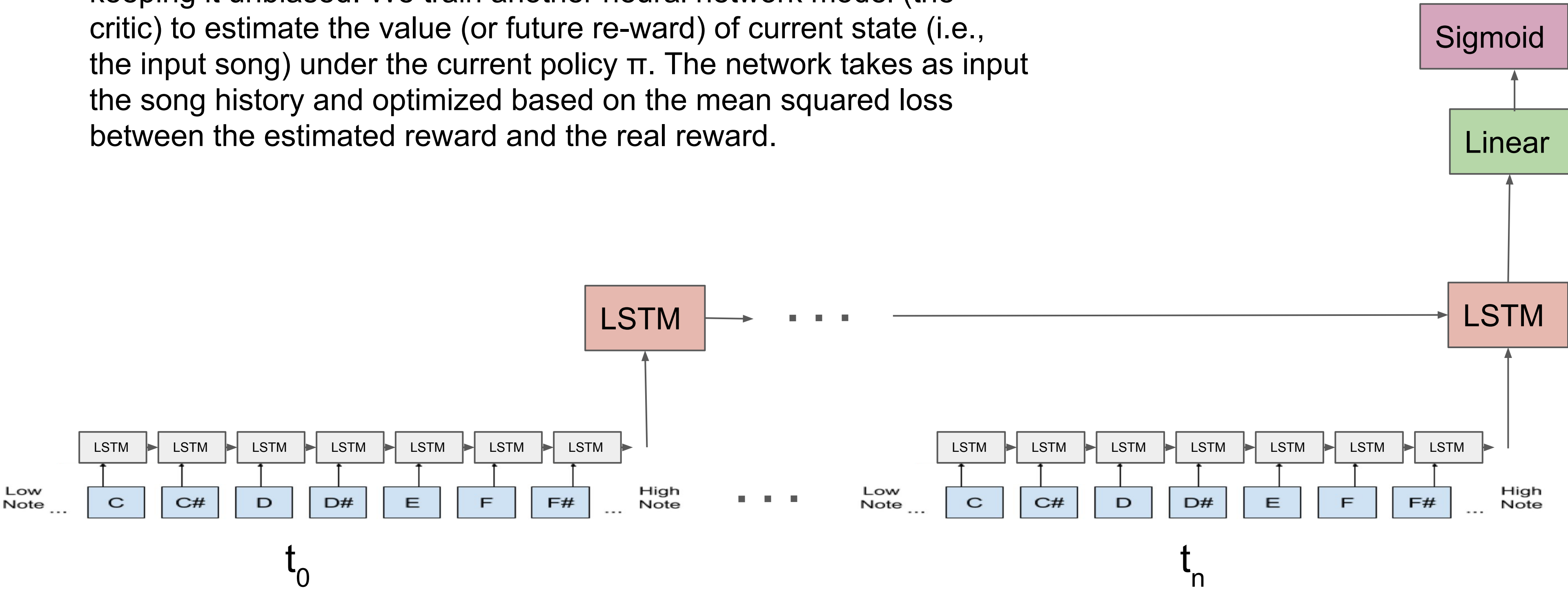
Policy Gradient of generator:

$$\nabla J(\theta) \approx [Q_+(\{x,y\}) - b(\{x,y\})] \nabla \log \pi(y|x) = [Q_+(\{x,y\}) - b(\{x,y\})] \nabla \sum_t \log p(y_t|x, y_{1:t-1})$$

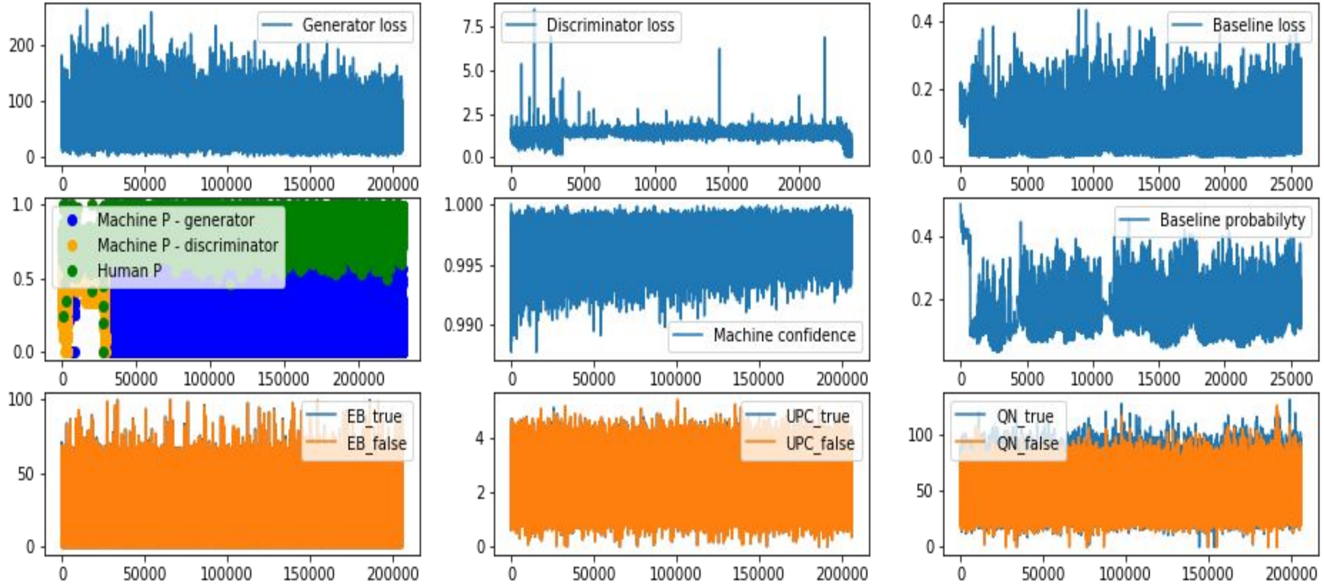
SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient (2017), *Adversarial Learning for Neural Dialogue Generation (2018)*

Baseline

Baseline value use to reduce the variance of the estimate while keeping it unbiased. We train another neural network model (the critic) to estimate the value (or future re-ward) of current state (i.e., the input song) under the current policy π. The network takes as input the song history and optimized based on the mean squared loss between the estimated reward and the real reward.



Typical scenario of learning



EB - ratio of empty bars (in %).

UPC - number of used pitch classes per bar (from 0 to 11)

QN - ratio of “qualified” notes (in %)

Loss of Discriminator:

$$Loss_d = -log(1-Q_{+_{false}}) - \log(Q_{+_{true}})$$

Loss of generator:

$$Loss_g = RL_{loss} + Classic_{loss}$$

Loss of Baseline:

$$Loss_b = \sum_i \left(Q_+(x, y_i) - b(x)\right)^2$$

EB, UPC, QN metrics were proposed in MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment (2018)

Learning technique. Comments on Methods

- Teacher Forcing - Use true sample as generator prediction
  - Helps to reach EB, UPC, QN metrics
- Monte Carlo - Sample negative examples several times from generator
  - Does not help much
- Baseline Value
  - Speeds up generator training
- Pretrain generator
- Pretrain discriminator
- Use noise as negative sample to pretrain discriminator
  - Speed up discriminator training
- Use classical cross entropy loss along with RL loss for generator
  - Generated music stay harmonic after a long training

References

1. Generator Architecture:
   a. DeepJ: Style-Specific Music Generation (2018 **arXiv:1801.00887**)
   b. Generating Polyphonic Music Using Tied Parallel Networks (2016).
   c. Attention Is All You Need (2017 **arXiv:1706.03762**)
2. Discriminator Architecture:
   a. A Hierarchical Neural Autoencoder for Paragraphs and Documents (2015 **arXiv:1506.01057v2**)
3. Learning technique:
   a. Adversarial Learning for Neural Dialogue Generation (**arXiv:1701.06547**)
   b. Simple statistical gradient-following algorithms for connectionist reinforcement learning (1992)
   c. Image-to-Image Translation with Conditional Adversarial Networks (2017 **arXiv:1611.07004v2**)
   d. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient (**arXiv:1609.05473**)
4. Metrics:
   a. MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment (**arXiv:1709.06298**)

Thank you for your attention!

Ivan.Anokhin@skoltech.ru

Egor.Nuzhin@skoltech.ru