PopH
Music generation via NN

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.

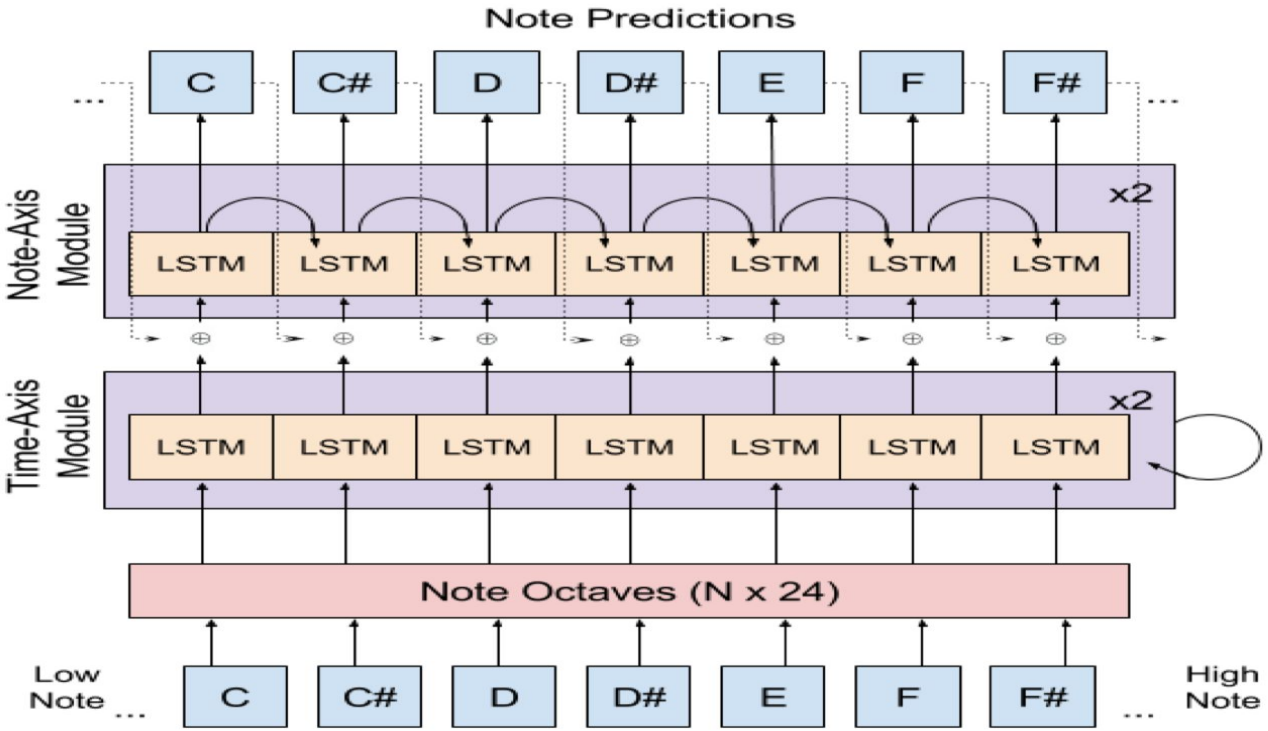
Related works

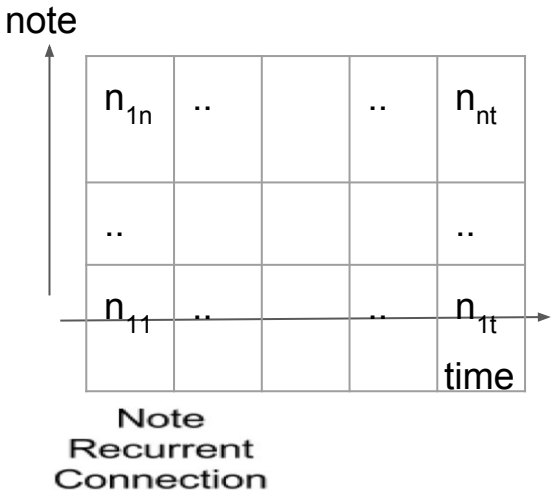
- For the basic algorithm to implement we decided to take the one from the recent papers: DeepJ: Style-Specific Music Generation (2018), Generating Polyphonic Music Using Tied Parallel Networks (2016).
- After implementation we used self-attention mechanism as described in article: Attention Is All You Need

Dataset description

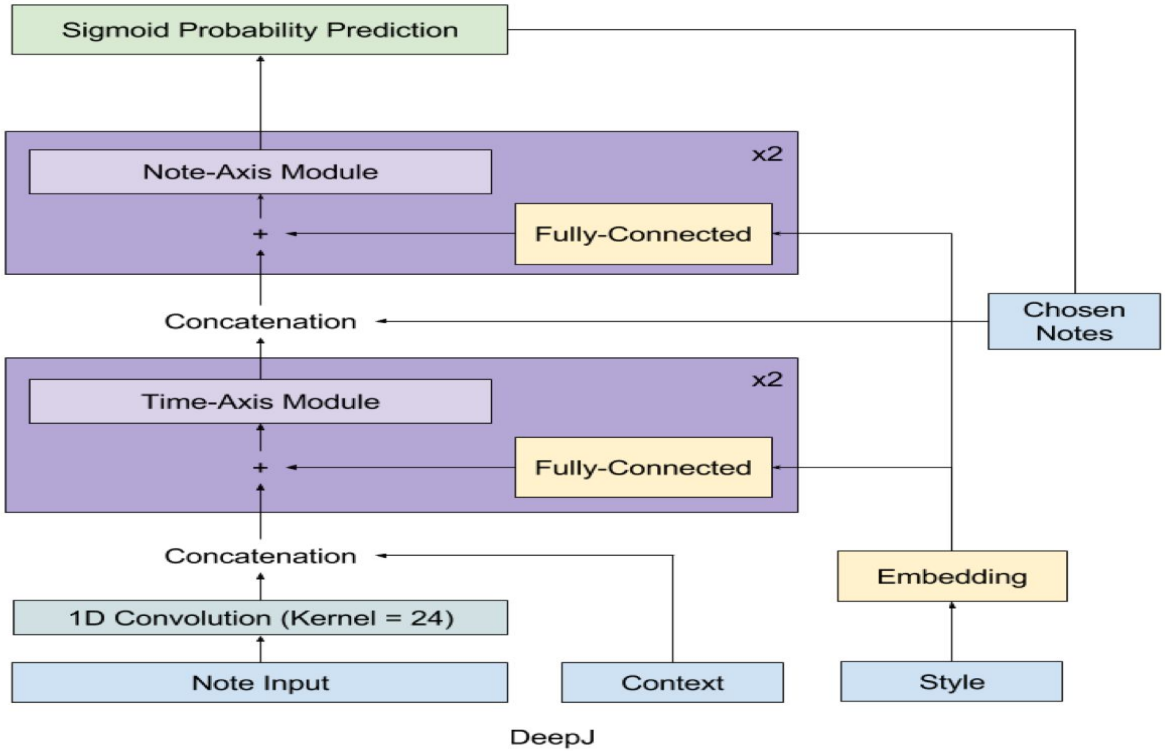
- We used open source dataset: The Lakh MIDI Dataset
- Transformed structure of midi file to use in model is:
  - Play matrix
  - Replay matrix
  - Volume matrix

ML methods: Biaxial LSTM Architecture



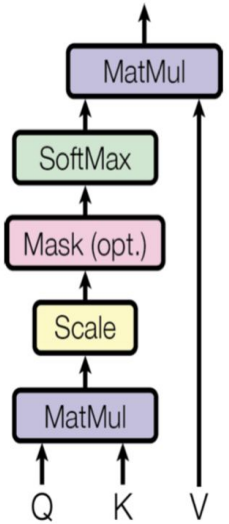


ML methods: DeepJ Architecture

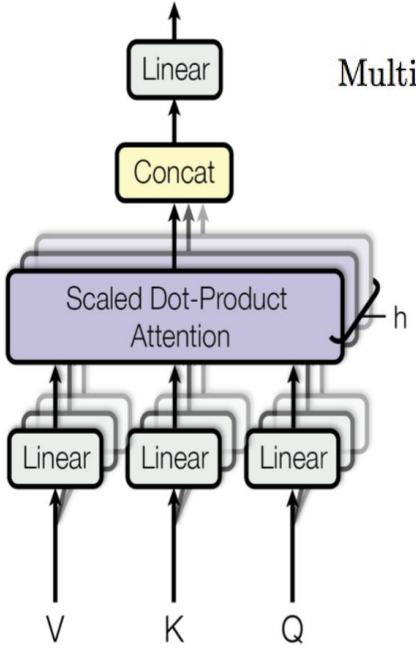


ML methods: Attention

Scaled Dot-Product Attention



Multi-Head Attention



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Self Attention Mechanism: When Q=K=V

ML methods: Our architecture

Output: t_{play}, t_{r}, $t_{dynamics}$

Note-Axis LSTM Module

Time-Axis LSTM Module

Self Attention Module

Note octaves (convolution 1d)

Input: t_{play}, t_{r}, $t_{dynamics}$

Experiments: DeepJ Losses

$$L_{play} = \sum t_{play} \log(y_{play}) + (1 - t_{play}) \log(1 - y_{play})$$

$$L_r = \sum t_{play}(t_r \log(y_r) + (1 - t_r) \log(1 - y_r))$$

$$L_{dynamics} = \sum t_{play}(t_{dynamics} - y_{dynamics})^2$$

$$L = L_{play} + L_r + L_{dynamic}$$

Experiments: our losses

$$L = L_{play} + L_r + L_{dynamic} + L_{harm}$$

$$L_{harm} = \min\left[A_{disharm} SoftMax\left(\tilde{y}_{play} \right) \right]$$

$$\tilde{y}_{play} = y_{play}[:12] + y_{play}[12:24] + y_{play}[24:36] + y_{play}[36:48]$$

$$A_{disharm} = \begin{pmatrix} C_{maj} \\ C_{\min} \end{pmatrix}$$

$$c_{maj} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$c_{\min} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Equipment:

3 GPU NVIDIA Tesla K80
(Microsoft Azure)

PopH Group

Thank you for your attention

Ivan.Anokhin@skoltech.ru

Egor.Nuzhin@skoltech.ru