Giovanni Sorice - 606915

# An application of Compressive Transformer for music generation


Intelligent Systems for Pattern Recognition (651AA), A.Y. 2019-2020

# Table of content


- Introduction
- Transformers model
- Library
- Experiments
- Results

# Music Generation

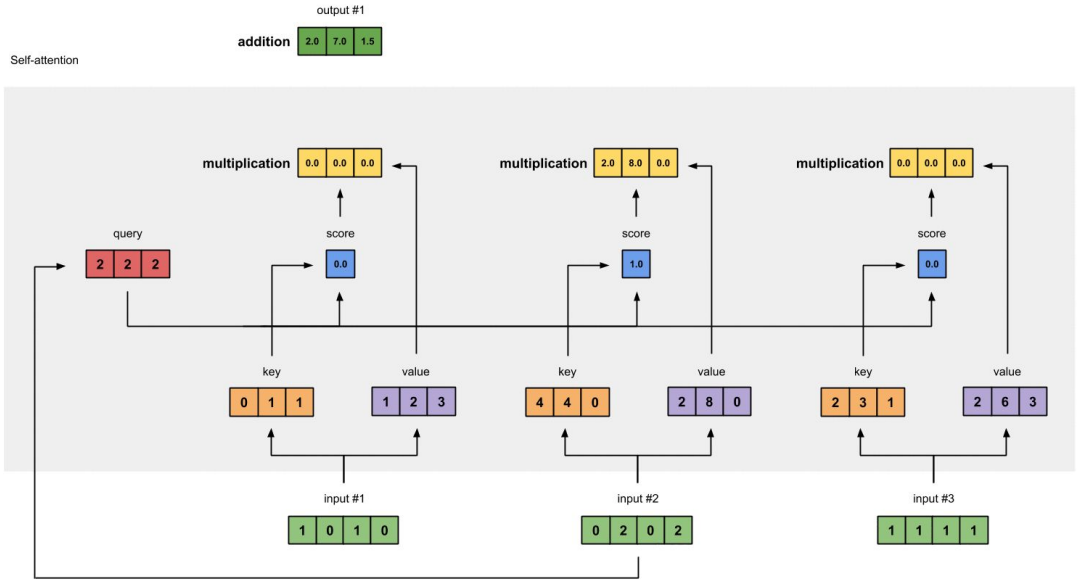
- Starting from the early 1960s with Iannis Xenakis with stochastic composition;
- Deep learning vs Markov models:
  - complexity of the models;
  - long-term temporal structures;
  - relations, contexts and regularities in the music.
- Autonomous music-making systems vs Assistance during the composition;

# Transformers model

# Vanilla and XL transformers

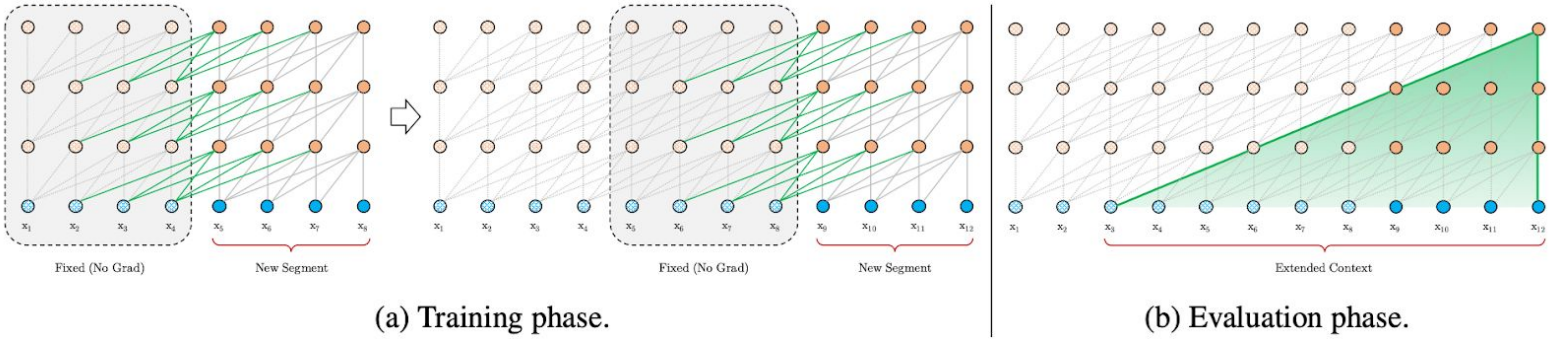
- Self-Attention

$$softmax(\frac{Q * K^T}{\sqrt{d^k}}) * V$$



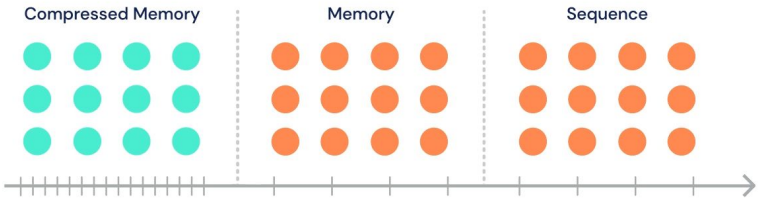
# Vanilla and XL transformers

- Recurrence Mechanism
- Relative Positional Encoding



# Compressive transformers

Memory compression




Compression functions:

- Max/mean pooling;
- 1D convolution;
- Dilated convolutions;
- Most-used.

# Library

# Compressive transformers library

```
from compressive_transformer_pytorch import CompressiveTransformer

model = CompressiveTransformer(
    num_tokens = 839,
    emb_dim = None,                # embedding dimensions
    dim = 64,
    heads = 8,
    depth = 6,
    seq_len = 64,
    mem_len = 64,                  # memory length
    cmem_len = 256,                # compressed memory buffer length
    cmem_ratio = 4,                # compressed memory ratio
    reconstruction_loss_weight = 1,# weight to place on compressed memory reconstruction loss
    attn_dropout = 0.1,            # dropout post-attention
    ff_dropout = 0.1,              # dropout in feedforward
    attn_layer_dropout = 0.1,      # dropout for attention layer output
    gru_gated_residual = False,    # whether to gate the residual intersection
    mogrify_gru = False,           # experimental feature that adds a mogrifier for the update and residual before gating by the GRU
    memory_layers = [5,6],         # specify which layers to use long-range memory
    one_head_kv = False,           # share one key/value head for all queries
    ff_glu = False                 # use GLU variant for feedforward
)
```

# Experiments

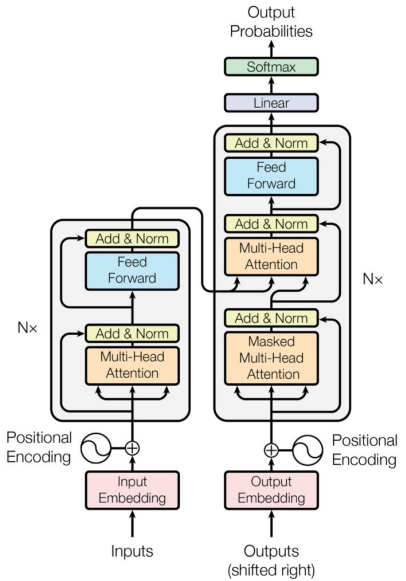
# Datasets

A subset of the Lakh MIDI Dataset.

1. 120 MIDI files are randomly and equally divided into three genres of different artist and all containing the piano instrument;
2. 120 MIDI files of only one genre of different artists and all containing the piano instrument;
3. 40 MIDI files are randomly and equally divided into three genres of different artist and all containing the piano instrument (this is a subset of the initial one);
4. (Piano) 120 MIDI files are randomly and equally divided into three genres of different artist and only containing the piano instrument;

# Architectures and tuning

- Six layer;
- Eight heads;
- Two memory layer;
- Length param in [32, 64, 128];
- Dropout [0, 0.1];
- Optimizer: Adam PyTorch;



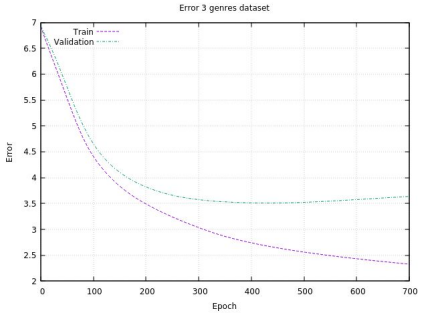
# Retrospective

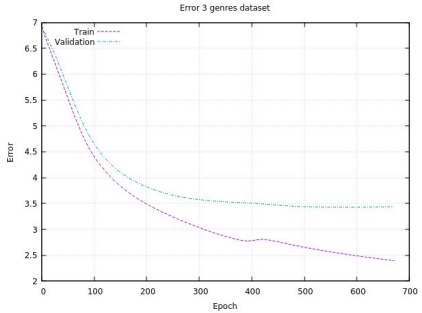
- What would I have changed?
  - Number of memory layer;
  - Dropout.
- Difficulties:
  - Undocumented library;
  - Preprocessing on the data;
  - Model requirement (time and computation).

# Results

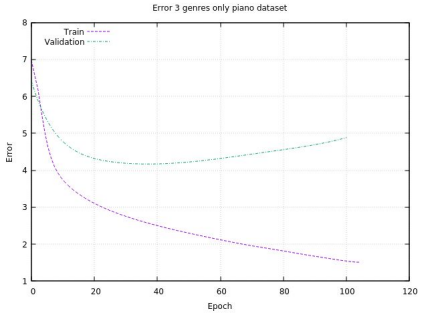
# Learning curves

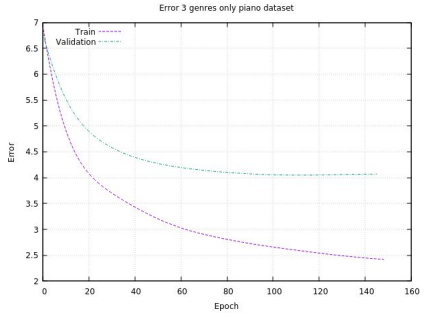
| Dataset | Epoch | Dropout | Error TR | Error VL | Error TS |
|---|---|---|---|---|---|
| 1 | 700 | 0 | 2.6694 | 3.5057 | 4.3238 |
| 1 | 671 | 0.1 | 2.5349 | 3.4274 | 4.2746 |
| Piano | 104 | 0 | 2.5028 | 4.1092 | 4.8609 |
| Piano | 148 | 0.1 | 2.6281 | 4.0401 | 4.7762 |
| 2 | 228 | 0 | 2.7016 | 3.2504 | 4.4305 |
| 2 | 525 | 0.1 | 2.7504 | 3.0382 | 4.1874 |
| 3 | 255 | 0 | 2.3218 | 4.3809 | 5.8341 |
| 3 | 187 | 0.1 | 2.5188 | 4.3612 | 5.4717 |




Learning curves with dataset 1 without an with dropout.




Learning curves with piano dataset without an with dropout.

# Further improvements

- Encode instruments with notes;
- Use and learn temporal parameters;
- Use and learn the melody of songs and genre;
- Try different hyperparameter configurations.

# Wrap up

- Transformers
  - Powerful architectures with a lot of possible applications;
  - Possible use of different compression functions;
  - Expensive in terms of time and computation (always remember “Do not shoot with a cannon”).
- Experience
  - Understand the data;
  - Understand the model;
  - Use documented libraries.

# Bibliography


- Iannis Xenakis. Formalized Music: Thought and Mathematics in Composition, Indiana University Press, 1963.
- A. Vaswan et al. Attention Is All You Need, NIPS 2017.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context, arXiv 2019.
- Rae, Jack W and Potapenko, Anna and Jayakumar, Siddhant M andHillier, Chloe and Lillicrap, Timothy P. Compressive Transformers for Long-Range Sequence Modelling, arXiv 2019.

# Figures

Slide 5: https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

Slide6: Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Slide 7: https://deepmind.com/blog/article/A_new_model_and_dataset_for_long-range_memory

Slide 12: Attention Is All You Need

# Appendix

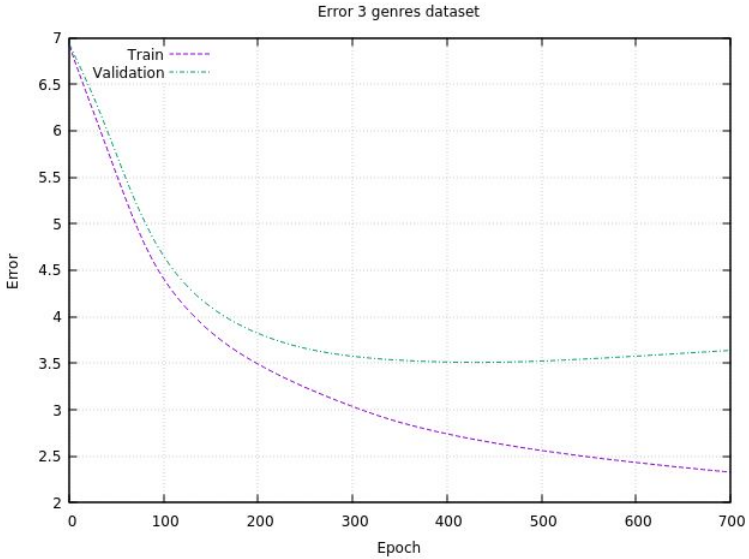
# Model parameters

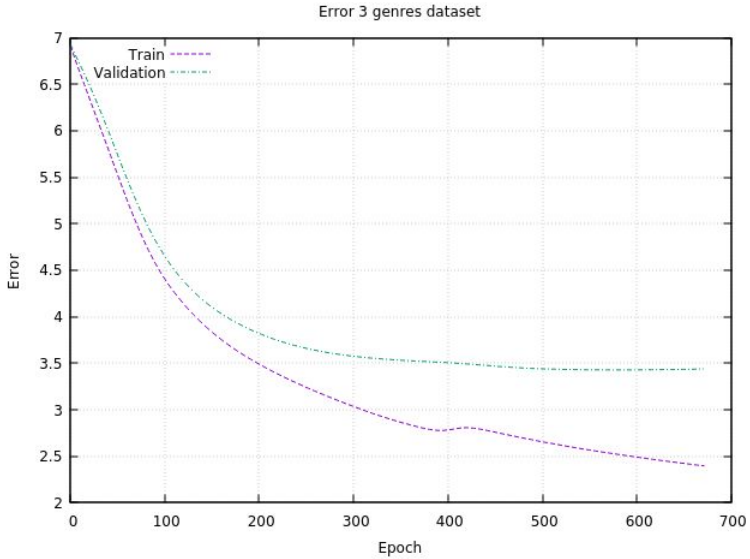
Lower bound from the vanilla transformers:

- Encoder $\boldsymbol{m} \times [1 \times \underbrace{(8 \times [(\boldsymbol{x} \times \boldsymbol{x}) + \boldsymbol{x}])}_{\text{MHDPA}} + 2 \times \underbrace{(\boldsymbol{x} + \boldsymbol{x})}_{\text{Layer Norm}} + 1 \times \underbrace{(\boldsymbol{x} \times \boldsymbol{x} + \boldsymbol{x})}_{\text{FFNN}}]$
- Decoder $\boldsymbol{n} \times [2 \times \underbrace{(8 \times [(\boldsymbol{x} \times \boldsymbol{x}) + \boldsymbol{x}])}_{\text{MHDPA}} + 3 \times \underbrace{(\boldsymbol{x} + \boldsymbol{x})}_{\text{Layer Norm}} + 1 \times \underbrace{(\boldsymbol{x} \times \boldsymbol{x} + \boldsymbol{x})}_{\text{FFNN}}]$

- Where n are the decoder layer and m the encoder layer.
- Approx: 104256+204096 = 308352 params

# Learning curves (dataset 1)



# Learning curves (dataset 2)