Aspekty techniczne pracy

Generacja muzyki przy pomocy metod uczenia maszynowego

Problemy napotkane podczas pracy

- Tokenizacja plików MIDI
- Trening modelu – adaptacja w RL
- Niestabilność oraz skalowalność modelu
- Nowa nadzieja

Tokenizacja

MidiTok: A Python package for MIDI file tokenization

Nathan Fradet[1,2], **Jean-Pierre Briot**[1], **Fabien Chhel**[4,3],
Amal El Fallah Seghrouchni[1], **Nicolas Gutowski**[3]
[1]Sorbonne University, CNRS, LIP6, F-75005 Paris
[2]Aubay, Boulogne-Billancourt, France
[3]University of Angers, LERIA, 49000 Angers, France
[4]ESEO, ERIS, 49100 Angers, France

https://github.com/Natooz/MidiTok

Paczka zawiera w sobie wiele algorytmów tokenizacji z różnych artykułów naukowych, które są zebrane pod wspólny interface. Paczka również umożliwiła integrację z tokenizerami Hugging Face, przez co możliwe jest tworzenie pipeline-ów.

Kolejną zaletą, jest *out of the box* tworzenie *datasetów* w bibliotece Torch, przez co implementacja *training loopa* jest o wiele prostrza.

Trening modelu – adaptacja w RL

$$V = \{v_1, v_2, \ldots, v_n\} \qquad \text{Model}(S_t) \rightarrow p(V)$$

$$\begin{aligned} S_{t_0} \quad & v_8, v_5, v_{12} \rightarrow v_{t_0} \qquad v_{t_0} \sim p(\text{Model}(S_{t_0})) \\ S_{t_1} \quad & v_8, v_5, v_{12}, v_3 \rightarrow v_{t_1} \\ S_{t_2} \quad & v_8, v_5, v_{12}, v_3, v_{21} \rightarrow v_{t_2} \end{aligned}$$

Model generuje na wyjściu rozkład prawdopodobieństwa nad wszystkimi tokenami, z którego następnie **losujemy**, kolejny token w sekwencji. Operacja losowania nie jest różniczkowalna, więc nie można po niej policzyć gradientu, co sprawia, że nie można z zachowanymi gradientami jej przekazać dalej (w moim podejściu do dyskryminatora modelu GAN), co uniemożliwia trening. Rozwiązaniem jest potraktowanie treningu jako problemu RL, gdzie stan w którym znajduje się agent to obecna sekwencja, a akcją, którą należy podjąć jest wylosowanie tokenu. Model jest uczony algorytmem *gradient policy* REINFORCE. Podejście to zostało zaprezentowane w artykule, gdzie wprowadzono model SeqGAN.

Niestabilność oraz skalowalność modelu

Jak każdy model GAN, trening modelu jest nietrywialny, na co składa się różnica w zaawansowaniu generatora i dyskryminatora. Problemem modelu jest również możliwość nieosiągnięcia zbieżności dla pewnych zbiorów danych.

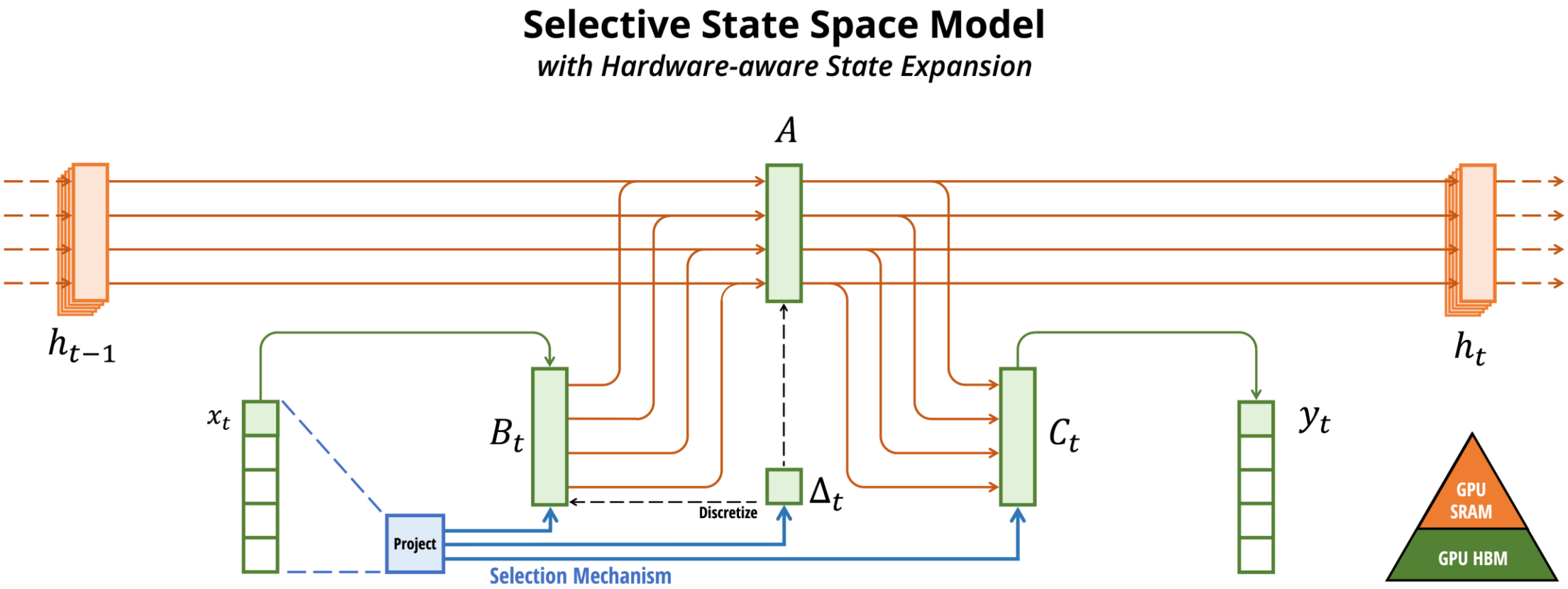
Problemem również okazał się trening na sprzęcie prywatnym (GPU z 12GB VRAM), gdzie implementacja „zapychała" pamięć już dla sekwencji o długości 40 tokenów. Zapotrzebowanie na pamięć rosło liniowo z ilością tokenów, więc dla dłuższych sekwencji, nawet na lepszym sprzęcie trening byłby mało opłacalny.

Algorithm 1 Sequence Generative Adversarial Nets

Require: generator policy G_θ; roll-out policy G_β; discriminator D_ϕ; a sequence dataset $\mathcal{S} = \{X_{1:T}\}$

Initialize G_θ, D_ϕ with random weights θ, ϕ.
Pre-train G_θ using MLE on $\mathcal{S}$
$\beta \leftarrow \theta$
Generate negative samples using G_θ for training D_ϕ
Pre-train D_ϕ via minimizing the cross entropy
repeat
  for g-steps **do**
    Generate a sequence $Y_{1:T} = (y_1, \ldots, y_T) \sim G_\theta$
    for t in $1 : T$ **do**
      Compute $Q(a = y_t; s = Y_{1:t-1})$ by Eq. (4)
    end for
    Update generator parameters via policy gradient Eq. (8)
  end for
  for d-steps **do**
    Use current G_θ to generate negative examples and combine with given positive examples $\mathcal{S}$
    Train discriminator D_ϕ for k epochs by Eq. (5)
  end for
  $\beta \leftarrow \theta$
until SeqGAN converges

Nowa nadzieja - Mamaba



Dość nowe rozwiązanie (z grudnia 2023), które rozwiązuje problem złożoności obliczeniowej architektury transformera (kwadratowa), i zamienia ją na złożoność liniową, przy okazji dobrze integrując się z obecnie używanym hardware-em. Okno *look-back* jest zdecydowanie większe, oraz model charakteryzuje się zdecydowanie większą możliwością odwzorowania danych przeszłych. Obecnie pokazane możliwości na ziorach danych z tekstem i DNA, dają lepsze wyniki niż najlepsze modele transformerowe oparte o attention.

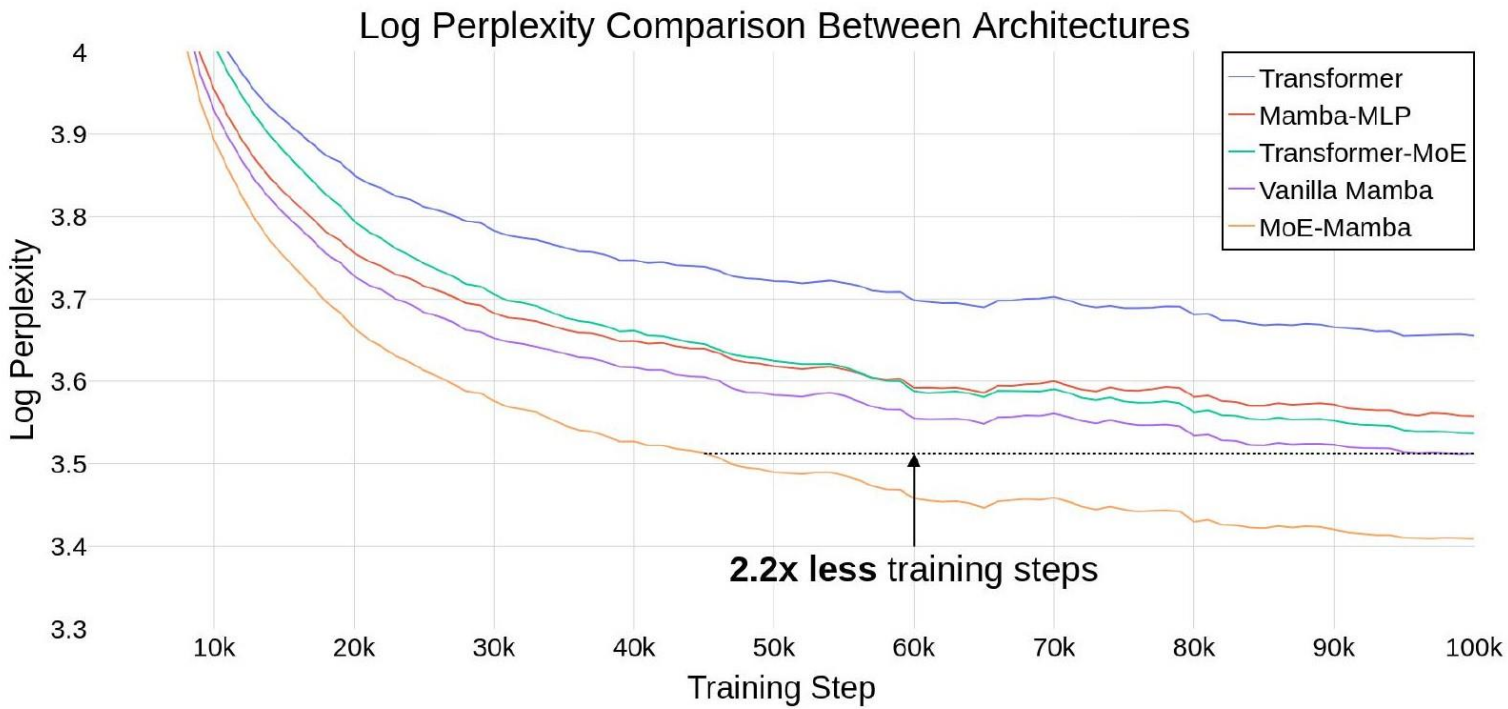
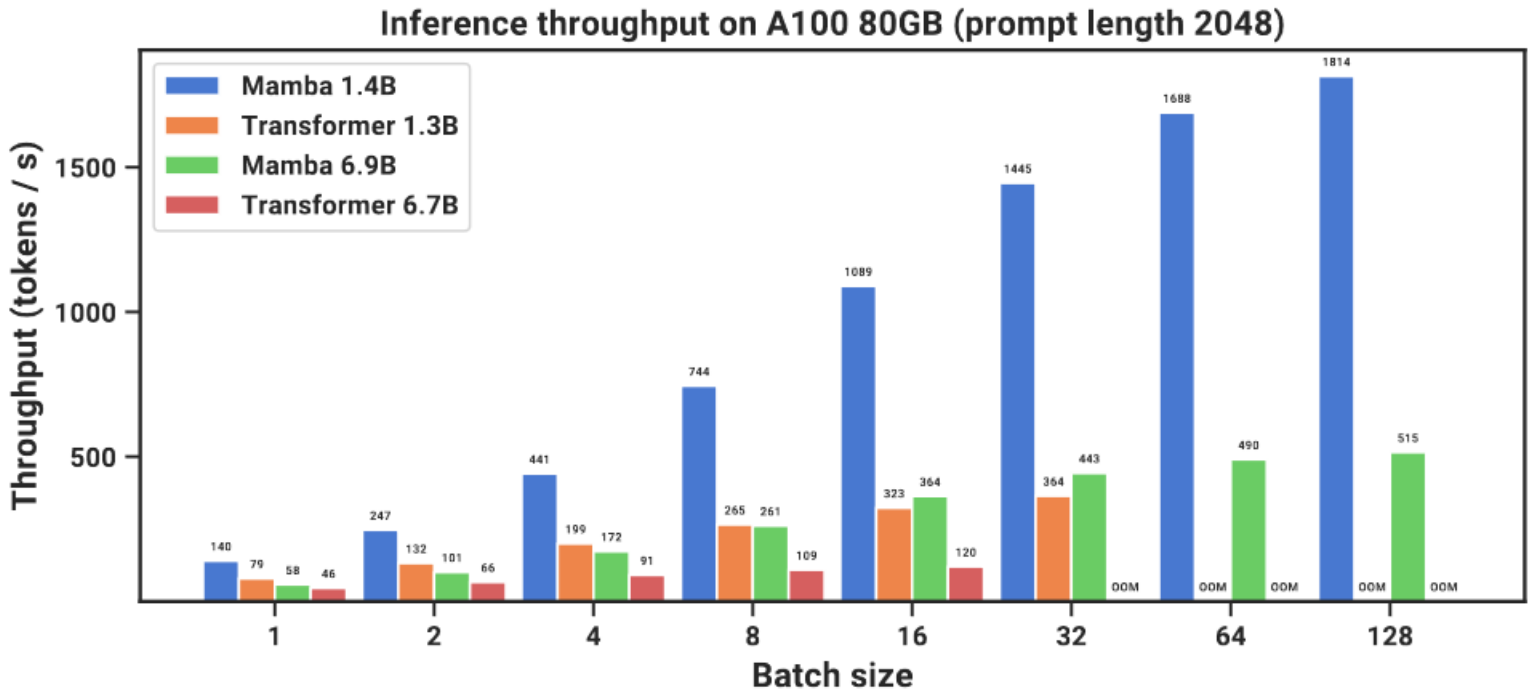


Figure 1. Log perplexity throughout the training of different methods. From top to bottom: Transformer; Mamba interleaved with feed-forward layers (Mamba-MLP); Transformer-Moe; Vanilla Mamba; MoE-Mamba.


Inference throughput on A100 80GB (prompt length 2048)
Mamba 1.4B
Transformer 1.3B
Mamba 6.9B
Transformer 6.7B
Throughput (tokens / s)
1500
1000
500
140
79
58
46
247
132
101
66
441
199
172
91
744
265
261
109
1089
323
364
120
1445
364
443
OOM
1688
OOM
490
OOM
1814
OOM
515
OOM
1
2
4
8
16
32
64
128
Batch size