Replication of "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention"

Viet-Anh Do and Johannes Bergner

Version 1.0.2, 14.2.2022

1 Introduction

Transformers introduced by (Vaswani et al., 2017) are the state-of-the-art in tasks like language understanding and image processing. However their quadratic memory complexity within the self-attention mechanism weakens the efficiency when dealing with long sequences. Consequently a number of so called "X-$former$"- models have been proposed, that improve the original Transformer in terms of computational and memory efficiency.

Next to other approaches, like Performer (Choromanski et al., 2020), Linformer (Wang et al., 2020) and Big Bird (Zaheer et al., 2020) the authors of "Transformers are RNNs"developed a Linear Transformer (Katharopoulus et al., 2020) to show how the quadratic complexity $O(N^2)$ can be reduced to a linear complexity of $O(N)$ without reducing the accuracy.

Standard Transformers use the folllowing attention matrix:

$$Attention(Q, K, V){=}V{=}softmax\left(\frac{QK^T}{\sqrt{D_k}}\right)V, \tag{1}$$

with the queries Q $\in \mathbb{R}^{N\times D_k}$, keys K $\in \mathbb{R}^{M\times D_k}$ and values V $\in \mathbb{R}^{N\times D_v}$, where N and M represent the lenghts of queries and keys (or values) and D_k and D_v the dimensions of the keys (or queries) and values. The softmax function is applied rowwise to QK^T.

(Katharopoulus et al., 2020) simplify this attention mechanism by generalizing equation (1) and replacing the softmax with the kernel feature map $\phi(x){=}elu(x) + 1$:

$$V_i'{=}\frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)} \tag{2}$$

Because of that $\sum_{j=1}^{i} \phi(K_j)V_j^T$ and $\sum_{j=1}^{i} \phi(K_j)$ can be computed once and reused for every query, which leads to a time and memory complexity of $O(N)$. Furthermore by representing these cumulated sums as S_i and Z_i, they can be seen as states of an RNN for causal attention and computed from S_{i-1} and Z_{i-1} in constant time.

In tests the Linear Transformer of (Katharopoulus et al., 2020) performed an image generation task based on the CIFAR-10 dataset over 4,000 times faster than the normal transformer with

the same accuracy. These results seem extraordinary, which is why the Linear Transformer has been recognized in further scientific research and the paper "$Transformers\ are\ RNNs$" has been recited about 250 times in nearly two years.

Nevertheless papers that benchmark the improvements in the Transformer architecture, like (Yi et al., 2020) show, that Linear Transformer perform faster then the original Transformer but not as high as 4,000 times faster. In this replication we will try to replicate a Linear Transformer in Julia in order to examnie if five-digit speed improvements are really possible.

2 Scope of the replication

Due to their linear complexity, Linear Transformers are said to be highly efficient when the sequence length increases. (Katharopoulus et al., 2020) evaluate their Transformer on image generation and automatic speech recognition and conclude, that their Linear Transformer is up to three orders of magnitude faster than regular Transformers while reaching the same performance levels. In this Replication we want to investigate, if Linear Transformers perform tasks with long sequences as precise as the original Transformer, but significantly faster.

However, the official GitHub Repository of the paper does not contain experiments. The authors published their experiments in a seperate GitHub Repository, that contains evaluations on image generation and a copy task of number sequences. In order to examine speed and precision with growing sequence length, a copy task of number sequences is favourable because it is easy to increase the sequence length and to evaluate the results. Furthermore an example of a copy task is already given in the Julia package Transformers.jl. We will use this example as a benchmark, that represents the performance of the Transformer by (Vaswani et al., 2017). Based on that we are going to replace the softmax-based attention function by a linear attention function. Then we apply different sequence lengths and examine, if the Linear Transformer performs faster but with the same precision as the original Transformer especially with high sequence lenghts.

3 Methoden

3.1 Modellbeschreibung

For this replication a minimalistic model was developed to determine the convergence properties of Linear Transformers. The Transformer which was introduced by Vaswani et al. (2017) has been chosen as our baseline in this experiment. Unlike this aforementioned model which utilized softmax attention, the authors shifted the feature map into the module attention, computed the dot product of key and value instead of key and query so that the attention matrix never be explicitly computed and thus decrease the complexity from $O(N^2max(D, M))$ to $O(ND^2M)$, where $N > D^2$ in practice. Our target is to determine, whether Linear Attention converges faster than the softmax attention for given sequence length.

3.2 Datenbeschreibung

For this synthetic task, we use a sequence of maximum length 128 with 10 different symbols separated by a dedicated separator symbol. These sequences are randomly generated.

3.3 Hyperparameter

We have adjusted the hyperparameters of the model according to the original paper. The number of head per Transformer is 8, whose size is 64, which leads to the size of the whole Transformer of 512. The original length is kept the same as in the original paper at 128 whereas the learning rate will stay constantly at $10^{-}3$.

3.4 Implementierung

The programming language that is used in the replication task is Julia. Due to the restriction of time, we have mainly taken most of the code from the package Transformers.jl as a template for the implementation. Other packages which were also imported into the notebook include Flux.jl, Tullio.jl, NNlib.jl and so on. For the synthetic task, we use a version of copy task based on the experiment which was used by Kitaev et al. (2020). As mentioned above in 3.1, what we do to implement linear attention is to replace the attention function of Transformers.jl with our own code called LinearAttention, which has been translated from the original PyTorch code by (Katharopoulus et al., 2020) to Julia code. The original code requires in addition one more function called splitHeads to reshape the 3D tensor to 4D tensor, which we have also included in the script. Furthermore, as the authors used the activation function elu(x) + 1 in original paper instead of relu(x) as in the Transformers.jl, we also attempt to implement elu(x) + 1 (which was called nelu in our script) into the code to make sure that the performances of each model are not affected by different conditions and therefore they are comparable. Also we have trimmed the other unnecessary functions like *apply_mask*, ... for the sake of simplification. Lastly, instead of using Kullback-Leibler divergence as loss function, we apply cross entropy to calculate the loss for the gradient descent as given in the original work.

.....hier kommt noch code dazu

3.5 Aufbau der Experimente

The sequence for the copy task was first with start and end symbol, then embeded to become a 128x128 Matrix and goes through 4 layer of encoder as well as 4 layer of decoder. At the end, the loss can be computed for each iteration, so that we can monitor the improvement of loss over each iteration. Furthermore, we can also observe the execute time of each model after a certain amount of iteration. To do so, we use an Nvidia RTX ... with ... of memory.

3.6 Ressourcen für Berechnungen

Beschreiben sie die Anforderungen für die Berechnungen für jedes ihrer Experimente, z.B. die Anzahl der CPU/GPU-Stunden oder die Voraussetzungen für den Hauptspeicher und GPU-Speicher. Geben sie für Zeit und Speicher eigene Abschätzungen an, bevor die Experimente gelaufen sind und vergleichen sie dies mit den tatsächlich verbrauchten Ressourcen. Sie müssen vor den Experimenten einplanen, dass diese Informationen auch durch ihren Code gemessen und gespeichert werden.

4 Ergebnisse

Starten sie mit einem Überblick über die Ergebnisse. Bestätigen ihre Ergebnisse die aufgeführten Behauptungen? Dieser Abschnitt sollte hauptsächlich Fakten nennen und so präzise wie möglich geschrieben werden. Die Bewertung und Diskussion kann im späteren Kapitel “Diskussion” folgen.

Beschreiben sie dann detailliert jedes einzelne Ergebnis, das sie haben. Zeigen sie wie es mit einer oder mehreren Behauptungen in Beziehung steht. Erklären sie konkret was der Kern ihres Ergebnis ist. Gruppieren sie die Ergebnisse in logische Abschnitte. Beschreiben sie klar, wo sie über den Originalartikel hinausgegangen sind, wo sie zusätzliche Experimente durchgeführt haben und wie diese mit den ursprünglichen Behauptungen in Beziehung stehen.

Tipp 1: Drücken sie sich genau aus und verwenden sie eine klare und einfache Sprache, z.B.

> "we reproduced the accuracy to within 1% of reported value, that upholds the paper's conclusion that it performs much better than baselines." oder
>
> "We konnten die Klassifikationsrate bis auf 1% des angegebenen Werts reproduzieren. Dies unterstützt die Schlussfolgerung der Artikels, dass der Ansatz leistungsfähiger als die Baselines ist."

Oft kann man nicht die exakt gleiche numerische Zahl als Ergebnis bekommen. Deshalb müssen sie das Ergebnis bewerten, um zu entscheiden, ob ihr Ergebnis die Behauptung der Originalartikels unterstützt.

Tipp 2: Nutzen sie Tabellen und Abbildungen, um ihre Ergebnisse darzustellen.

4.1 Ergebnis 1

4.2 Ergebnis 2

4.3 Zusätzliche Ergebnisse, die nicht im Originalartikel enthalten waren

Beschreiben sie alle zusätzlichen Experimente, die über den Originalartikel hinausgehen. Dies können Experimente zu weiteren Datenmengen sein oder sie probieren andere Methoden bzw. weitere Vereinfachungen des Modells aus oder passen die Hyperparameter an. Beschreiben sie für jedes zusätzliche Experiment, was sie genau durchgeführt haben, was die Ergebnisse sind und diskutieren sie was diese Ergebnisse zeigen.

5 Diskussion

Beschreiben sie die weiterführenden Implikationen der experimentellen Ergebnisse. War der Originalartikel replizierbar bzw. reproduzierbar. Falls nicht, welche Faktoren haben dazu geführt, dass die Experimente nicht reproduziert werden konnten.

Bewerten sie, ob sie die Evidenz, die sie durch das Durchführen der Experimente erhalten haben, auch überzeugt, dass die Behauptungen des Originalartikels dadurch gestützt werden. Diskutieren sie die Stärken und Schwächen ihres Ansatzes, vielleicht haben sie aus Zeitgründen nicht alle Experimente durchführen können, oder vielleicht haben zusätzliche Experimente durchgeführt, die den Originalartikel weiter stärken.

5.1 Was war einfach?

Beschreiben sie welche Teile der Replikation/Reproduktion sich leicht umsetzen ließen. Lief der Code der Autoren problemlos? War es aufgrund der Beschreibung im Originalartikel nicht aufwändig die Methoden zu reimplementieren? Dieser Abschnitt soll den Lesenden zeigen, welche Teile des Originalartikels sich leicht für eigene Ansätze verwenden lässt.

Tipp: Machen sie keine pauschalen Verallgemeinerungen. Was für sie leicht ist, muss für andere nicht leicht sein. Geben sie genügend Kontext und erklären sie warum manche Sachen leicht waren, z.B. der Code hatte eine umfangreiche Dokumentation der Schnittstellen und viele Beispiele aus der Dokumentation passten zu den Experimenten im Artikel.

5.2 Was war schwer?

Beschreiben sie welche Teile ihrer Replikation/Reproduktion aufwändig oder schwierig waren oder viel mehr Zeit in Anspruch genommen haben, als sie erwarteten. Vielleicht waren Daten nicht verfügbar, so dass sie einige Experimente nicht verifizieren konnten, oder der Code der Autoren funktionierte nicht und musste erst debugged werden. Vielleicht dauerten auch einige Experimente zu lange und sie konnten sie deshalb nicht verifizieren. Dieser Abschnitt soll den Lesenden zeigen, welche Teile des Originalartikels schwer wiederverwendbar sind, bzw. signifikante Zusatzarbeiten und Ressourcen erfordern.

Tipp: Setzen sie sorgfältig ihre Diskussion in den richtigen Kontext, z.B. sagen sie nicht " die Mathematik war schwer verständlich" sondern sagen sie " die Mathematik erfordert fortgeschrittene Kenntnisse in Analysis für das Verständnis".

5.3 Empfehlungen für die Replizierbarkeit / Reproduzierbarkeit

Geben sie Empfehlungen, wie die Autoren des Originalartikels oder andere Forschende in diesem Feld die Replizierbarkeit / Reproduzierbarkeit verbessern können.

6 Kommunikation mit den Autoren

Our contact to the authors was unanswered. However, the issues section in the GitHub Repository contains questions that are also interesting in our case.

The user gaceladri made his own implementation of a Linear Transformer and shared a plot that showed, that the loss of a Linear Transformer is higher than the loss of a softmax-based Transformer (in this case a BERT model) when the gradient steps are lower than 10,000. The user asked, if he has done something wrong regarding this discovery. Angelos Katharopoulos (username: angeloskath) answered, that this is possible, because the attention matrix of a Linear Transformer is low rank which leads to a harder learning process. The author further explains, that the whole point of Linear Transformers is an optimization of speed and memory. When these aspects are insignificant, using a softmax attention is probably better. https://github.com/idiap/fast-transformers/issues/84

The user Yogurt928 asked, why Einstein sums are used in the code, whereas the paper uses buzz signs. Katharopoulos responds:

> "The reason for using einsum instead of multiplication and summation is that it is faster since the large matrix does not need to ever be computed and kept in memory."

At last, the user burcehan asked, why $elu(x) + 1$ was used as the feature map. He recognized, that the convergence of his model is very slow using this function. Angelos Katharopoulos answers that the feature map needs to correspond to a non-negative similarity score. The question, whether other feature maps are better is an open research problem. In the tests for the paper, some feature maps achieved similar results, others not. All in all this depends on the analysed problem, but problems that require sparse attention patterns might be harder to learn using the $elu(x) + 1$ feature map.