Term Paper Presentation

RWKV: Reinventing RNNs for the transformer era

Presented by:
Jainisha Choksi (202211019)
Hinal Desai (202211035)
Man Desai (202211040)
Rutvik Prajapati (202211053)

01

MOTIVATION

MOTIVATION

- RNNs are hard to massively parallelize across many GPUs. This motivated using "attention mechanism" which was introduced in transformers.
- However, the attention mechanism scales quadratically with the length of sequence to be processed.
- Transformers require much more memory than RNNs.
- RWKV combines the efficient parallelizable training of transformers with efficient inference of RNNs.
- The RWKV approach leverages a linear attention mechanism and allows to formulate the model as either a transformer or RNN.
- It is the first non-transformer architecture to be scaled to tens of billions of parameters.
- The motivation behind RWKV is to bridge the gap between computational efficiency and expressive capacity in neural network architectures.

The problem

Drawbacks of models with sequential processing tasks:

- **RNNs:** suffer from vanishing gradient problem, making them difficult to train for long sequences. They cannot be parallelized in time dimension during training.
- **CNNs:** captures only local patterns, which limits their capacity to deal with long-range dependencies.
- **Transformers:** The self-attention mechanism poses challenge due to its quadratic complexity making the architecture computationally expensive and memory-intensive.

Hence, RWKV was introduced to combine the strengths of RNNs and Transformers and eliminating drawbacks.

02

PROBLEM FORMULATION

Problem formulation

- The current implementation for RWKV is trained to perform the language modeling task.
- **Objective:** Develop a language model to perform the Question-Answering task as well as fill-in-the blank tasks based on different datasets.
- **Input:** Textual content
- **Output:** Predicted answer from given choices
- **Loss function:** cross-entropy loss

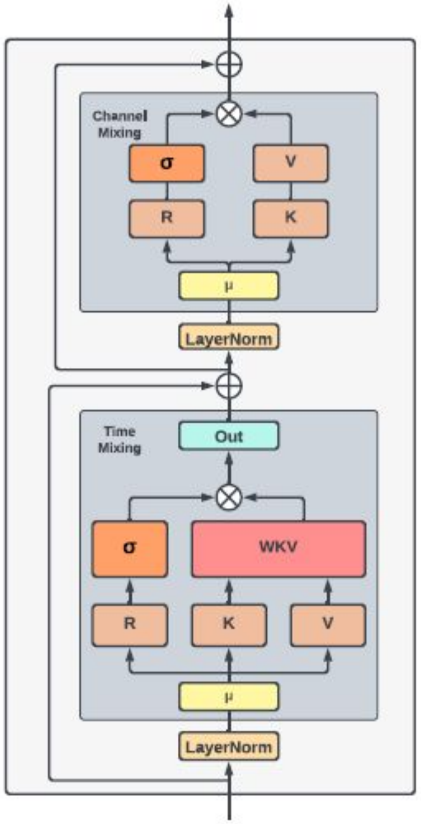
03

APPROACH

RWKV ARCHITECTURE

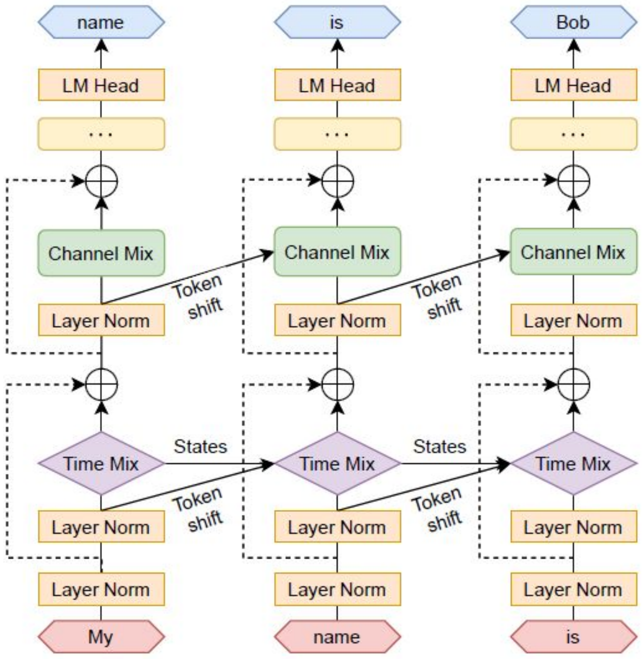
The RWKV architecture is comprised of a series of stacked residual blocks, each formed by a **time-mixing** and **channel-mixing** sub-blocks with recurrent structures. The four primary elements are:

- **R: Receptance** vector acting as the acceptance of past information.
- **W: Weight** is the positional weight decay vector. (trainable model parameter)
- **K: Key** is a vector analogous to K in traditional attention.
- **V : Value** is a vector analogous to V in traditional attention.



RWKV ARCHITECTURE

- **Time-mixing:** The recurrence is formulated both as linear interpolation between current input and input at previous time step. This is called time-shift mixing or token-shift
- **Channel-mixing:** W is a channel-wise vector multiplied by relative position.
- The WKV computation, plays the role of Attn(Q,K,V) in transformers without the quadratic cost as the interactions are between scalars.
- **Squared-ReLU activation** used in channel-mixing component
- **Sigmoid** of receptance is taken in both time-mixing and channel-mixing components, works like a "forget-gate" to eliminate unnecessary historical information.



RWKV = RNN + Transformers

- **RNN-like Sequential Decoding (time-sequential mode):** . Each output token is dependent only on the latest state, which is of constant size irrespective of sequence length. RWKV behaves as an RNN decoder in this case, yielding constant speed and memory footprint with respect to sequence length.
- **Transformer-like Parallelization (time-parallel mode):** The element-wise WKV computation is time-dependent but can be parallelized along the other two dimensions. The token-shift is implemented as a simple offset in temporal dimension at each block.

IMPLEMENTATION

- Pytorch library
- Custom CUDA kernel for WKV computation
- Current implementation focuses on the task of language modeling (RWKV-LM)
- Model architecture comprises of **embedding layer** and several **residual blocks** applied sequentially. After the last block, **Layer Normalization** and **linear projection** is used to obtain the logits to be used in next-token prediction task and calculate the **cross-entropy loss** during training.
- Training is performed in time-parallel mode while autoregressive inference and potential chat interface leverage the time-sequential mode.

04

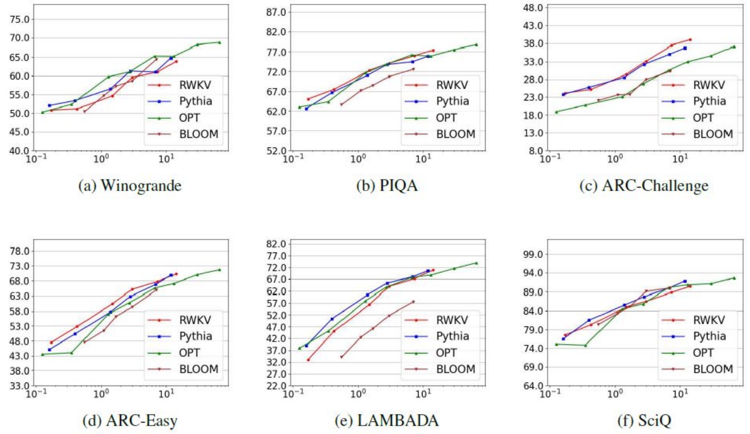
EVALUATION

Research questions

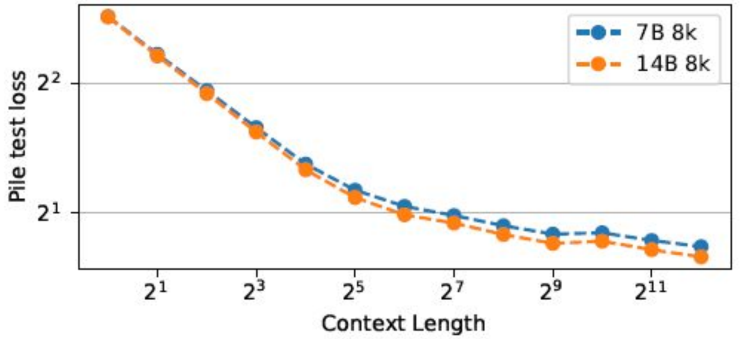
- **RQ1:** Is RWKV competitive against quadratic transformer architectures with equal number of parameters and training tokens?
- **RQ2:** When increasing the number of parameters, does RWKV remain competitive against quadratic transformer architectures?
- **RQ3:** Does increasing parameters of RWKV yield better language modeling loss, when RWKV models are trained for context lengths that most open-sourced quadratic transformers cannot efficiently process?

BENCHMARKS

Addressing the **RQ1 and RQ2**, RWKV demonstrated competitiveness on six benchmarks (**Winogrande, PIQA, ARC-C, ARC-E, LAMBADA, and SciQ**) against Pythia, OPT, and BLOOM—major open-source quadratic complexity transformer models.

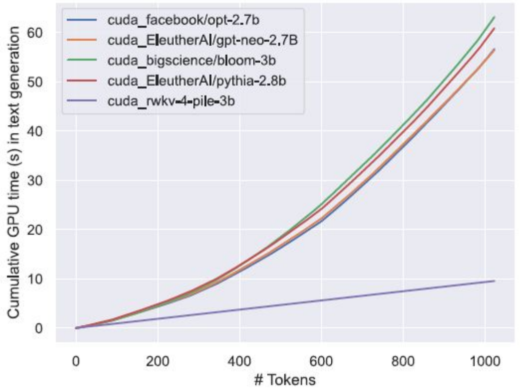


For **RQ3,** the figure below shows that increasing context length leads to lower test loss on Pile. This indicates that RWKV can make effective use of long contextual information.



Inference experiments

- Benchmarked the inference experiments according to size and family.
- Evaluated the text generation speed and memory requirements on typical compute platforms including GPU and CPU.
- Comparative studies done on RWKV-4 and ChatGPT/GPT-4 revealed that RWKV-4 is very sensitive to prompt engineering. F1-performance increased from 44.2% to 74.8% when prompts adjusted from the ones used for GPT to be more suitable to RWKV.



THANKS!

REFERENCES

[1] Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., GV, K.K. and He, X., 2023. RWKV: Reinventing RNNs for the Transformer Era. arXiv preprint arXiv:2305.13048.

APPENDIX

$$
\begin{aligned}
r_t &= W_r \cdot (\mu_r x_t + (1-\mu_r) x_{t-1}), \\
k_t &= W_k \cdot (\mu_k x_t + (1-\mu_k) x_{t-1}), \\
v_t &= W_v \cdot (\mu_v x_t + (1-\mu_v) x_{t-1}), \\
wkv_t &= \frac{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i=1}^{t-1} e^{-(t-1-i)w+k_i} + e^{u+k_t}}, \\
o_t &= W_o \cdot (\sigma(r_t) \odot wkv_t),
\end{aligned}
$$

Time-mixing block

$$
\begin{aligned}
r_t &= W_r \cdot (\mu_r x_t + (1-\mu_r) x_{t-1}), \\
k_t &= W_k \cdot (\mu_k x_t + (1-\mu_k) x_{t-1}), \\
o_t &= \sigma(r_t) \odot (W_v \cdot \max(k_t, 0)^2),
\end{aligned}
$$

Channel-mixing block