Ouroboros: On Accelerating Training of Transformer-Based Language Models

Qian Yang[1*], Zhouyuan Huo[2], Wenlin Wang[1], Heng Huang[2], Lawrence Carin[1]

[1] ECE, Duke University [2] ECE, University of Pittsburgh

[*] qian.yang@duke.edu

Code

Problem

- **Language models** are essential for natural language processing (NLP) tasks, such as machine translation and text summarization.
- Remarkable performance has been demonstrated recently across many NLP domains via a **Transformer-based language model** with over a billion parameters, verifying the benefits of model size.
  - BERT [Devlin' 18]: 340M parameters
  - OpenAI GPT-2 [Radford' 19]: 1.5B parameters
- **Model parallelism** is required if a model is too large to fit in a single computing device.
- Current methods for model parallelism suffer from
  - either **backward locking** in backpropagation
  - or are not applicable to language models because the shared embeddings make the networks **non-separable**

Preliminaries

Consider training a Transformer-based language model with L layers. We may represent the computations in the network as follows:

$$h_1 = F_1(h_0; w_1, V_i), \quad (1)$$
$$h_l = F_l(h_{l-1}; w_l), \qquad \forall l \in \{2, ..., L-1\}, \quad (2)$$
$$h_L = F_L(h_{L-1}; w_L, V_o), \quad (3)$$

where h_{l-1} denotes the input of layer l, $F_l(\cdot; w_l)$ denotes the computation of layer l with weight w_l, V_i is the input embedding, and V_o is the output projection. In particular, h_0 denotes the input data x, and $h_L = F(x; \tilde{w})$ represents the output of the network. For the sake of performance, V_i and V_o are typically tied in language modeling or machine translation tasks, so that $V = V_i = V_o$

Defining network weights $w = [w_1, w_2, ..., w_L]$, embedding layer V and $\tilde{w} = [w, V]$, the loss function for language modeling can be represented as:

$$\min_{\tilde{w}} f(F(x; \tilde{w}), y), \quad (4)$$

where y denotes the target. In the following context, we use $f(\tilde{w})$ for simplicity.

Gradient-Based Method

With stochastic gradient descent (SGD) [Robbins' 51], the weights of the network are updated as:

$$w_l^{t+1} = w_l^t - \gamma_t \nabla f_{l,x_{i(t)}}(\tilde{w}^t) \quad \text{and} \quad V^{t+1} = V^t - \gamma_t \nabla f_{V,x_{i(t)}}(\tilde{w}^t), \quad (5)$$

for any $l \in \{1, ..., L\}$, where γ_t is the stepsize, $i(t)$ represents data index at iteration t, and $\nabla f_{l,x_{i(t)}}(\tilde{w}^t)$ is the gradient of the loss function (4) with respect to the weights at layer l and data sample $x_{i(t)}$.

Backpropagation

In the backward computation, we apply the chain rule:

$$\frac{\partial f(\tilde{w}^t)}{\partial w_l^t} = \frac{\partial f(\tilde{w}^t)}{\partial h_l^t} \times \frac{\partial h_l^t}{\partial w_l^t} \quad \text{and} \quad \frac{\partial f(\tilde{w}^t)}{\partial h_{l-1}^t} = \frac{\partial f(\tilde{w}^t)}{\partial h_l^t} \times \frac{\partial h_l^t}{\partial h_{l-1}^t}, \quad (6)$$

where $\tilde{w} = [w, V]$, and $\nabla f_{l,x_{i(t)}}(\tilde{w}^t) = \frac{\partial f(\tilde{w}^t)}{\partial w_l^t}$. For Transformer-based language models, the gradient with respect to the input embedding and output projection layer are computed as:

$$\frac{\partial f(\tilde{w}^t)}{\partial V_i} = \frac{\partial f(\tilde{w}^t)}{\partial h_1^t} \times \frac{\partial h_1^t}{\partial V_i} \quad \text{and} \quad \frac{\partial f(\tilde{w}^t)}{\partial V_o} = \frac{\partial f(\tilde{w}^t)}{\partial h_L^t} \times \frac{\partial h_L^t}{\partial V_o}. \quad (7)$$

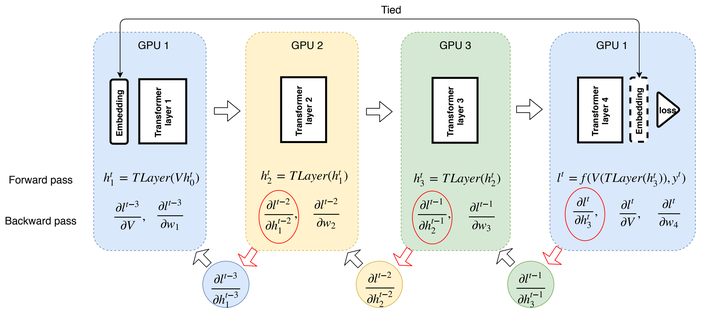
Contributions

(i) We present the first model-parallel algorithm to parallelize the training of Transformer-based language models, going beyond data parallelism.

(ii) The convergence rate of the proposed algorithm is analyzed, and it is proven that it is guaranteed to converge to critical points for non-convex problems.

(iii) We evaluate the proposed algorithm in training two Transformer-based language models, and experimental results verify our theoretical analysis, demonstrating convergence much faster than previous methods with comparable or better accuracy.

Method



We split an L-layer network into K modules so that the weights of the network are divided into K groups and each group is placed on a GPU. Our model is connected end-to-end and shrinks like a snake when grouping, so we name it "Ouroboros."

In our Ouroboros algorithm, at each iteration all modules are independent of each other, by using delayed gradients. Let $\tilde{w} = [w, V]$, the gradient of weights in $\mathcal{G}(k)$ is

$$\nabla f_{\mathcal{G}(k),x_{i(t-K+k)}}\left(\tilde{w}^{t-K+k}\right) = \sum_{l \in \mathcal{G}(k)} \frac{\partial f_{x_{i(t-K+k)}}(\tilde{w}^{t-K+k})}{\partial w_l^{t-K+k}}, \text{ if } t-K+k \geq 0, \quad (8)$$

or 0 otherwise for any $k \in \{1, 2, ..., K\}$. The gradient of V is the average of the gradients of output projection and input embedding:

$$\nabla f_{V,x_{i(t)}}(\tilde{w}^t) = \frac{1}{2}\nabla f_{V_o,x_{i(t)}}\left(\tilde{w}^t\right) + \frac{1}{2}\nabla f_{V_i,x_{i(t-K+1)}}\left(\tilde{w}^{t-K+1}\right) = \frac{1}{2}\left(\frac{\partial f(\tilde{w}^t)}{\partial V_o^t} + \frac{\partial f(\tilde{w}^{t-K+1})}{\partial V_i^{t-K+1}}\right), \quad (9)$$

otherwise 0 if $t-K+1 < 0$.

Gradient-Based Method with Ouroboros

$$w_{\mathcal{G}(k)}^{t+1} = w_{\mathcal{G}(k)}^t - \gamma_t \cdot g_k^t; \quad (10)$$
$$V_i^{t+1} = V_o^{t+1} = V_i^t - \gamma_t \cdot g_V^t, \quad (11)$$

Convergence Analysis

Theorem 1 *With Assumptions 1 and 2, and the fixed stepsize sequence $\{\gamma_t\}$ satisfying $\gamma_t = \gamma$ and $\gamma L \leq 1, \forall t \in \{0, 1, ..., T-1\}$, let w^* be the optimal solution to $f(w)$. The output of Algorithm 1 satisfies:*

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\left\|\nabla f(w^t)\right\|_2^2 \leq \frac{2\left(f(w^0) - f(w^*)\right)}{\gamma T} + 2\gamma L M_K, \quad (12)$$

and $M_K = (K + \frac{3}{4})M + (\frac{K^2}{2} + K^3)(K+4)M$.

According to Theorem 1, the average norm of gradients can converge to the neighborhood of critical points. As $T \to \infty$, it is also upper bounded by $2\gamma L M_K$.

Theorem 2 *With Assumptions 1 and 2, and the diminishing stepsize sequence $\{\gamma_t\}$ satisfying $\gamma_t = \frac{\gamma_0}{1+t}$, $\gamma_t L \leq 1, \forall t \in \{0, 1, ..., T-1\}$, assume w^* to be the optimal solution to $f(w)$, and let $\sigma = K$ such that $M_K = (K + \frac{3}{4})M + (\frac{K^3}{2} + K^4)(K+4)M$. Setting $\Gamma_T = \sum_{t=0}^{T-1} \gamma_t$, then the output of Algorithm 1 satisfies*

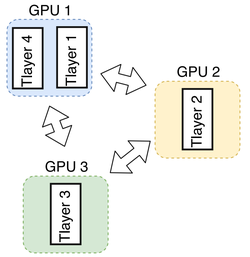
$$\frac{1}{\Gamma_T}\sum_{t=0}^{T-1} \gamma_t \mathbb{E}\left\|\nabla f(w^t)\right\|_2^2 \leq \frac{2\left(f(w^0) - f(w^*)\right)}{\Gamma_T} + \frac{2\sum_{t=0}^{T-1}\gamma_t^2 L M_K}{\Gamma_T}$$

Since $\gamma_t = \frac{\gamma_0}{t+1}$, the following inequalities are satisfied:

$$\lim_{T\to\infty}\sum_{t=0}^{T-1}\gamma_t = \infty \quad \text{and} \quad \lim_{T\to\infty}\sum_{t=0}^{T-1}\gamma_t^2 < \infty. \quad (13)$$

Therefore, according to Theorem 2, when $T \to \infty$, the right-hand side of (13) converges to 0.

Algorithm



Algorithm 1 Ouroboros + SGD

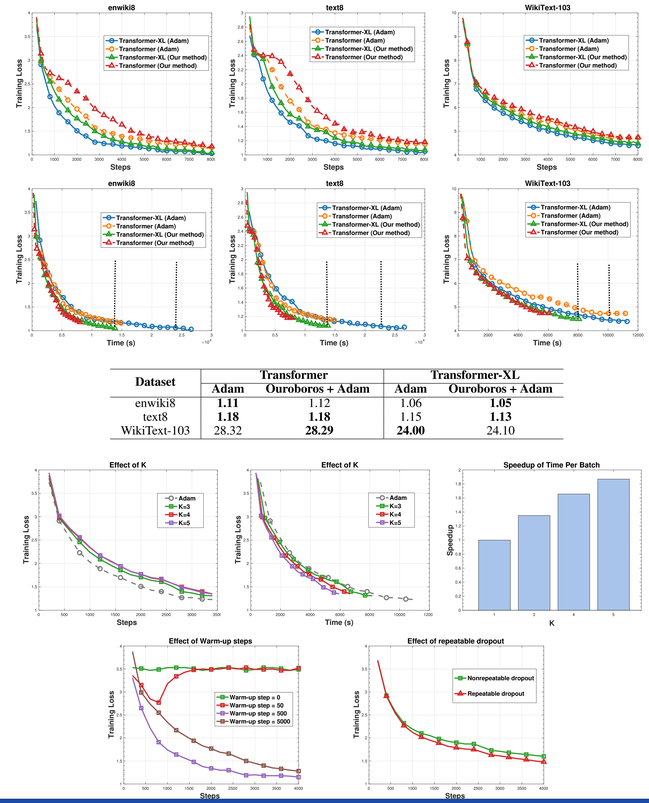
Require:
Initial weights $w^0 = [w_{\mathcal{G}(1)}^0, ..., w_{\mathcal{G}(K)}^0]$;
Initial word embedding $V_i^0 = V_o^0$;
Stepsize sequence $\{\gamma_t\}$;

1: **for** $t = 0, 1, 2, \ldots, T-1$ **do**
2: **for** $k = 1, \ldots, K$ **in parallel do**
3: Compute delayed gradient g_k^t for module k following (8);
4: Compute mixed gradient g_V^t for embedding layer following (9);
5: Update weights and embedding layer following SGD:

$$w_{\mathcal{G}(k)}^{t+1} = w_{\mathcal{G}(k)}^t - \gamma_t \cdot g_k^t;$$
$$V_i^{t+1} = V_o^{t+1} = V_i^t - \gamma_t \cdot g_V^t;$$

6: **end for**
7: **end for**
8: Output w^s, V_i^s and V_o^s randomly from $\{w^t\}_{t=0}^{T-1}$, $\{V_i^t\}_{t=0}^{T-1}$ and $\{V_o^t\}_{t=0}^{T-1}$.

Experimental Results



Dataset	Transformer		Transformer-XL	
	Adam	Ouroboros + Adam	Adam	Ouroboros + Adam
enwiki8	**1.11**	1.12	1.06	**1.05**
text8	**1.18**	**1.18**	1.15	**1.13**
WikiText-103	28.32	**28.29**	**24.00**	24.10

References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.
- Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.