Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention Supplementary Material

Angelos Katharopoulos¹² Apoorv Vyas¹² Nikolaos Pappas³ François Fleuret^{24*}

1. Gradient Derivation

In the first section of our supplementary material, we derive in detail the gradients for causally masked linear transformers and show that they can be computed in linear time and constant memory. In particular, we derive the gradients of a scalar loss with respect to the numerator of the following equation,

$$V_{i}' = \frac{\phi(Q_{i})^{T} \sum_{j=1}^{i} \phi(K_{j}) V_{j}^{T}}{\phi(Q_{i})^{T} \sum_{j=1}^{i} \phi(K_{j})}.$$
(1)

The gradient with respect to the denominator and the fraction are efficiently handled by autograd. Without loss of generality, we can assume that Q and K already contain the vectors mapped by $\phi(\cdot)$, hence given the numerator

$$\bar{V}_i = Q_i^T \sum_{j=1}^i K_j V_j^T,$$
 (2)

and $\nabla_{\bar{V}}\mathcal{L}$ we seek to compute $\nabla_Q\mathcal{L}$, $\nabla_K\mathcal{L}$ and $\nabla_V\mathcal{L}$. Note that $Q \in \mathbb{R}^{N \times D}$, $K \in \mathbb{R}^{N \times D}$ and $V \in \mathbb{R}^{N \times M}$. To derive the gradients, we first express the above equation for a single element without using vector notation,

$$\bar{V}_{ie} = \sum_{d=1}^{D} Q_{id} \sum_{j=1}^{i} K_{jd} V_{je} = \sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je}.$$
(3)

Subsequently we can start deriving the gradients for Q by taking the partial derivative for any Q_{lt} , as follows

$$\frac{\partial \mathcal{L}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \frac{\partial \bar{V}_{le}}{\partial Q_{lt}} = \sum_{e=1}^{M} \frac{\partial \mathcal{L}}{\partial \bar{V}_{le}} \left(\sum_{j=1}^{l} K_{jt} V_{je} \right).$$
(4)

If we write the above equation as a matrix product of gradients it becomes,

$$\nabla_{Q_i} \mathcal{L} = \nabla_{\bar{V}_i} \mathcal{L} \left(\sum_{j=1}^i K_j V_j^T \right)^T, \tag{5}$$

proving equation 13 from the main paper. In equation 4 we made use of the fact that Q_{lt} only affects \bar{V}_l hence we do not need to sum over *i* to compute the gradients. However, for *K* and *V* this is not the case. In particular, K_j affects all \bar{V}_i where $i \ge j$. Consequently, we can write the partial derivative of the loss with respect to K_{lt} as follows,

$$\frac{\partial \mathcal{L}}{\partial K_{lt}} = \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \bar{V}_{ie}}{\partial K_{lt}} = \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} \frac{\partial \left(\sum_{d=1}^{D} \sum_{j=1}^{i} Q_{id} K_{jd} V_{je}\right)}{\partial K_{lt}}$$

$$= \sum_{e=1}^{M} \sum_{i=l}^{N} \frac{\partial \mathcal{L}}{\partial \bar{V}_{ie}} Q_{it} V_{le}.$$
(6)

As for Q we can now write the gradient in vectorized form,

$$\nabla_{K_i} \mathcal{L} = \left(\sum_{j=i}^N Q_j \left(\nabla_{\bar{V}_j} \mathcal{L} \right)^T \right) V_i, \tag{7}$$

proving equation 14 from the paper. Following the same reasoning, we can compute the partial derivative of the loss with respect to V_{lt} and prove equation 15. Note that the cumulative sum matrices for the gradient with respect to Q and K have the same size, however one is computed in the forward direction (summing from 1 to N) similarly to the forward pass and the other is computed in the backwards direction (summing from N to 1) similar to backpropagation through time done in RNNs.

2. Training Evolution

In figure 1 we present the training evolution of all transformer models in our experiments. For the MNIST experiment (Fig. 1a) we train all methods for 250 epochs. The sequence length is small enough so that the training time does not vary significantly for all methods. We observe that our method converges on par with softmax attention outperforming significantly both reformer variants.

On the other hand, for CIFAR-10 (Fig. 1b) we train all methods for a fixed amount of time, namely 7 days. We observe that *lsh-1* and *linear* complete significantly more epochs than softmax and lsh-4 and achieve better performance. This gap is expected to increase with a further increase in sequence length.

Finally, in our last experiment on automatic speech recognition (Fig. 1c), softmax outperforms significantly both Reformer and linear in terms of convergence. Note that linear is $3 \times$ faster per epoch which means it has completed approximately 4 times more epochs in comparison to softmax. Even though softmax attention is better in this task, we observe that *linear transformers* significantly outperform Reformer both in terms of convergence and final performance.



Figure 1: Training evolution of transformers for all our experiments. It can be observed that *linear transformers* converge consistently faster than Reformer and in the autoregressive experiments on par with softmax. For MNIST all methods are trained for 250 epochs while for CIFAR we train for 7 days. In the speech recognition experiments all methods are trained to convergence. The details of the experiments can be found in § 4.2.1, § 4.2.2 and § 4.3 in the main paper.

3. Image Generation Throughput Discussion

3.1. Stateful softmax attention

In § 4.2 of the main paper, we report the image generation throughput and we compare with **softmax** transformer and **lsh**. In this section we create another baseline, denoted as **stateful-softmax**, that implements a softmax autoregressive transformer as a recurrent model. Namely, all the keys and values are saved and then passed to the model again when predicting the next element of the sequence. The state of this recurrent model is the set of keys and values which has size proportional to the sequence length. This is qualitatively different to our proposed model that has a state with fixed dimensions and computing the *i*-th state given the previous one has fixed computational cost regardless of *i*.

Transformers are RNNs

Method	Bits/dim	Images/sec			Method	Bits/dim	Images/sec		
Softmax	0.621	0.45	(1×)		Softmax	3.47	0.004	(1×)	
Stateful-softmax	0.621	7.56	(16.8×)		Stateful-softmax	3.47	0.32	(80×)	
LSH-1	0.745	0.68	$(1.5\times)$		LSH-1	3.39	0.015	(3.75×)	
LSH-4	0.676	0.27	(0.6×)		LSH-4	3.51	0.005	(1.25×)	
Linear (ours)	0.644	142.8	(317 ×)	_	Linear (ours)	3.40	17.85	(4,462 ×)	
(a) Image generation on MNIST					(b) Image generation on CIFAR-10				

Table 1: Comparison of autoregressive image generation throughput of MNIST and CIFAR-10 images. The experiment can be found in \S 4.2 in the main paper. For stateful-softmax we save the keys and values and reuse them for predicting the next element. A detailed description of this extra baseline can be found in \S 3.1.

Table 1 summarizes the results. We observe that stateful-softmax is significantly faster than vanilla transformers. However, its complexity is still quadratic with respect to the sequence length and our forumlation is more than $50 \times$ faster for CIFAR-10. Moreover, we would like to point out that implementing a similar stateful attention for Reformer is not a trivial task as the sorting and chunking operations need to be performed each time a new input is provided.

3.2. Equalizing the batch size

In the previous sections we evaluate the throughput of all transformer variants for the task of autoregressive image generation. However, another important factor to consider is latency, namely the total time required to produce a single image. To this end, we use a batch size of 1 and measure the time required by all methods to generate a single image. In addition to running the inference on the GPU, we also evaluate the time required on CPU. The results are reported in table 2.

Method	Seconds (CPU)		Seconds (GPU)		Method	Seconds (CPU)		Seconds (GPU)	
Softmax	72.6	(13.2×)	10.2	(1.4×)	Softmax	8651.4	(191.8×)	300.1	(4.9×)
Stateful-softmax	7.4	$(1.3 \times)$	10.4	$(1.42 \times)$	Stateful-softmax	71.9	(1.6×)	70.4	$(1.14\times)$
LSH-1	46.0	(8.3×)	19.2	$(2.6 \times)$	LSH-1	2318.9	(51.4×)	221.6	(3.6×)
LSH-4	112.0	(20×)	55.8	(7.6×)	LSH-4	5263.7	(116.7×)	683.9	(11.1×)
Linear (ours)	5.5	(1×)	7.3	(1×)	Linear (ours)	45.1	$(1 \times)$	61.3	(1×)

(a) Image generation on MNIST

(b) Image generation on CIFAR-10

Table 2: Comparison of the time required to generate a single image with autoregressive transformers on MNIST and CIFAR-10. We run all methods with a batch size of 1 both on CPU and GPU and report the total time in seconds. For all numbers in the table, lower is better.

We observe that all methods underutilize the GPU and achieve significantly smaller image generation throughput than the one shown in table 1. The proposed linear transformer is faster than all the methods and in particular it is almost $6.6 \times$ faster than softmax transformers for generating an image on CIFAR-10. Note that our linear autoregressive transformer is the only method that is faster on the CPU than on the GPU in every case. This is due to the fact that computing the attention as an RNN has such a low cost that the main computational bottleneck becomes the inevitable outer loop over the sequence.

4. Qualitative Results on Image Generation

In this section we provide qualitative results for our image generation experiments. Since the perplexity of all models is approximately the same, as expected, the qualitative differences are not significant. A rather interesting observation however is that the Reformer models provide significantly fewer variations in their unconditional samples. Moreover, we observe that image completion is a significantly easier task than unconditional generation as all models perform significantly better.



Figure 2: Unconditional samples from the transformer models trained with MNIST. See § 4.2.1 in the main paper.



Figure 3: MNIST digit completion from all trained models. See § 4.2.1 in the main paper.



Figure 4: Unconditional samples from the transformer models trained with CIFAR-10. See § 4.2.2 in the main paper.



Figure 5: CIFAR-10 image completions from all trained transformer models. See § 4.2.2 in the main paper.