Proceedings of Machine Learning Research vol 145:1–23, 2022 3rd Annual Conference on Mathematical and Scientific Machine Learning

Momentum Transformer: Closing the Performance Gap Between Self-attention and Its Linearization

author names withheld

Editor: Under Review for MSML 2022

Abstract

Transformers have achieved remarkable success in sequence modeling and beyond but suffer from quadratic computational and memory complexities with respect to the length of the input sequence. Leveraging techniques include sparse and linear attention and hashing tricks; efficient transformers have been proposed to reduce the quadratic complexity of transformers but significantly degrade the accuracy. In response, we first interpret the linear attention and residual connections in computing the attention map as gradient descent steps. We then introduce momentum into these components and propose the *momentum transformer*, which utilizes momentum to improve the accuracy of linear transformers while maintaining linear memory and computational complexities. Furthermore, we develop an adaptive strategy to compute the momentum value for our model based on the optimal momentum for quadratic optimization. This adaptive momentum eliminates the need to search for the optimal momentum value and further enhances the performance of the momentum transformer. A range of experiments on both autoregressive and non-autoregressive tasks, including image generation and machine translation, demonstrate that the momentum transformer outperforms popular linear transformers in training efficiency and accuracy.

Keywords: transformer, adaptive momentum, efficient attention

1. Introduction

The self-attention mechanism plays a fundamental role in building transformers (Vaswani et al., 2017; Kim et al., 2017). Given an input sequence $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D_x}$ of N feature vectors, the self-attention transforms it into another sequence $\hat{\boldsymbol{V}} = [\hat{\boldsymbol{v}}_1, \ldots, \hat{\boldsymbol{v}}_N]^\top \in \mathbb{R}^{N \times D_v}$ as follows

$$\hat{\boldsymbol{v}}_i = \sum_{j=1}^{N} \operatorname{softmax}\Big(\frac{\boldsymbol{q}_i^\top \boldsymbol{k}_j}{\sqrt{D}}\Big)\boldsymbol{v}_j, \text{ for } i = 1, \ldots, N, \tag{1}$$

where the scalar $\operatorname{softmax}((\boldsymbol{q}_i^\top \boldsymbol{k}_j)/\sqrt{D})$ can be understood as the attention $\hat{\boldsymbol{v}}_i$ pays to the input feature $\boldsymbol{x}_j$. The vectors $\boldsymbol{q}_i, \boldsymbol{k}_j$, and $\boldsymbol{v}_j$ are called the query, key, and value vectors, respectively; these vectors are computed as follows

$$\begin{aligned}
[\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_N]^\top &:= \boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q^\top \in \mathbb{R}^{N \times D}, \\
[\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_N]^\top &:= \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K^\top \in \mathbb{R}^{N \times D}, \\
[\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N]^\top &:= \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V^\top \in \mathbb{R}^{N \times D_v},
\end{aligned} \tag{2}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{D \times D_x}$, and $\boldsymbol{W}_V \in \mathbb{R}^{D_v \times D_x}$ are the weight matrices. We can further write (1) into the following compact form

$$\hat{\boldsymbol{V}} = \operatorname{softmax}\Big(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\Big)\boldsymbol{V}, \tag{3}$$

where the softmax function is applied to each row of the matrix $(\boldsymbol{Q}\boldsymbol{K}^\top)/\sqrt{D}$. Equation (3) is also called the "scaled dot-product attention" or "softmax attention". Each transformer layer $T_\ell(\cdot)$ is defined via the following residual connection,

$$T_\ell(\boldsymbol{X}) = f_\ell(\hat{\boldsymbol{V}} + \boldsymbol{X}), \tag{4}$$

where $f_\ell(\cdot)$ is a function that transforms each feature vector independently and usually chosen to be a feedforward network. In this paper, we call a transformer built with softmax attention standard transformer or transformer. It is easy to see that both memory and computational complexity of (3) are $\mathcal{O}(N^2)$ with N being the length of the input sequence. We can further introduce causal masking into (3) for autoregressive applications (Vaswani et al., 2017).

Transformers have become the state-of-the-art model for solving many challenging problems in natural language processing (Vaswani et al., 2017; Al-Rfou et al., 2019; Dai et al., 2019; Williams et al., 2018; Devlin et al., 2018; Brown and et al., 2020; Howard and Ruder, 2018; Rajpurkar et al., 2016) and computer vision (Dehghani et al., 2018; So et al., 2019; Dosovitskiy et al., 2020; Touvron et al., 2020). Nevertheless, the quadratic memory and computational cost of computing the softmax attention (3) is a major bottleneck for applying transformers to large-scale applications that involve very long sequences, such as those in (Liu et al., 2018; Huang et al., 2018; Parmar et al., 2018). Thus, much recent research on transformers has been focusing on developing efficient transformers, aiming to reduce the memory and computational complexities of the model (Qiu et al., 2019; Child et al., 2019; Ho et al., 2019; Parmar et al., 2018; Beltagy et al., 2020; Ainslie et al., 2020; Wang et al., 2020b; Tay et al., 2020a,b; Kitaev et al., 2020; Roy et al., 2021; Vyas et al., 2020; Zaheer et al., 2021; Wang et al., 2020b; Katharopoulos et al., 2020; Choromanski and et al., 2021; Shen et al., 2021; Schlag et al., 2021; Blanc and Rendle, 2018; Rawat et al., 2019; Song et al., 2021; Peng et al., 2021; Xiong et al., 2021; Nguyen et al., 2021). A thorough survey of recent advances in efficient transformers is available at (Tay et al., 2020c). These efficient transformers have better memory and/or computational efficiency at the cost of a significant reduction in accuracy.

1.1. Motivation

Katharopoulos et al. (2020) have established a connection between transformers and recurrent neural networks (RNNs) through the kernel trick. They proposed the linear transformer, which can be considered a rank-one approximation of the softmax transformer. Linear transformers have computational advantages in training, test, and inference: the RNN formulation (see Equation (8) below) enjoys fast inference, especially for autoregressive tasks, and the unrolled RNN formulation (see Equation (6) below) is efficient for fast training. See section 2 for a detailed review of the linear transformer and its advantages. Nguyen et al. (2020) proposes integrating momentum into RNNs to accelerate training RNNs and improve learning long-term dependencies. We notice that MomentumRNN also enjoys a closed unrolling form, which is quite unique among existing techniques for improving RNNs, enabling fast training, test, and inference; see section 3 for details. As such, in this paper we study *how does momentum improves linear transformers?*

1.2. Contribution

We propose *momentum transformers* by integrating two new momentum-related ingredients into the recently proposed linear transformers (Katharopoulos et al., 2020) and its RNN formulation to improve the model's accuracy and efficiency. Our contributions include: 1) Similar to (Nguyen

et al., 2020), we make an analogy between the RNN formulation of causal linear attention, which is the linear attention with causal masking for auto-regressive applications (Katharopoulos et al., 2020), and a gradient descent step. We then integrate the heavy ball-style momentum into this RNN formulation and result in the causal momentum attention. We extend this causal momentum attention into momentum attention for both autoregressive and non-autoregressive applications. We name the transformer with the new momentum attention the *momentum transformer*. 2) We further introduce another momentum into the residual connection between the attention $\hat{\boldsymbol{V}}$ and the input $\boldsymbol{X}$ in (4) to enhance the model's performance. 3) We develop a new adaptive strategy to compute the momentum value and eliminate the burden of tuning momentum hyperparameters in our model. We name the momentum transformer with adaptive momentum the *adaptive momentum transformer*. The major advantages of momentum-based transformers include:

- Momentum and adaptive momentum transformers inherit memory and computational efficiency from the linear transformers while achieving better accuracy.
- The training of momentum-based transformers converges remarkably faster than the training of linear transformers.
- The design principle of momentum transformers is rooted in momentum-based optimization algorithms, enabling us to design more general and advanced momentum transformers for a wide range of applications.

1.3. Related Work

In this part, we briefly review three lines of recent research that are most related to our work: 1) momentum in optimization and sampling, 2) momentum in deep neural network (DNN) design and 3) algorithms for efficient transformers.

Momentum in optimization and sampling Momentum has been a popular technique for accelerating (stochastic) gradient-based optimization (Polyak, 1964; Goh, 2017; Sutskever et al., 2013; Kingma and Ba, 2014; Paszke et al., 2019) and sampling algorithms (Duane et al., 1987; Neal et al.; Chen et al., 2014; Betancourt, 2017). A particularly interesting momentum is the iteration-dependent one in NAG (Nesterov, 1983; Nemirovskii and Nesterov, 1985; Beck and Teboulle, 2009), which has a significantly better convergence rate than constant momentum for convex optimization. The stochastic gradient NAG that employs a scheduled restart can also be used to accelerate DNN training with better accuracy and faster convergence (Wang et al., 2020a).

DNNs with momentum. Momentum has been used for designing DNN architectures. He et al. (2019) employ momentum to build large and consistent dictionaries for unsupervised learning with a contrastive loss. At the core of this approach is a momentum-based moving average of the queue encoder. Many DNN-based methods for sparse coding are designed by unfolding the classical optimization algorithms (Moreau and Bruna, 2017), e.g., FISTA (Beck and Teboulle, 2009), in which momentum can be used in the underpinning optimizer. MomentumRNNs (Nguyen et al., 2020) are a class of RNN models that are designed based on momentum accelerated first-order optimization algorithms. MomentumRNNs can effectively resolve the vanishing gradient issue in training RNNs and obtain faster training and better performance over traditional RNNs. Momentum has also been integrated into ResNets (Li et al., 2018).

Efficient transformers. Existing efficient transformers can be roughly classified into several categories, as summarized in (Roy et al., 2021). Among these categories are models with fixed patterns, which sparsify the attention matrix (Parmar et al., 2018; Liu et al., 2018; Qiu et al., 2019; Child et al., 2019; Beltagy et al., 2020). Another category includes models that integrate two or more distinct access patterns to improve the coverage (Child et al., 2019; Ho et al., 2019). Learnable patterns are also leveraged to learn the access pattern in a data-driven fashion (Kitaev et al., 2020; Roy et al., 2021; Tay et al., 2020b). Some other efficient transformers take advantage of a side memory module to access multiple tokens at once (Lee et al., 2019; Sukhbaatar et al., 2019; Asai and Choi, 2020; Beltagy et al., 2020). Finally, low-rank and kernelization approximation are employed to improve the memory and computational efficiency of computing self-attention, see e.g., (Tsai et al., 2019; Wang et al., 2020b; Katharopoulos et al., 2020; Choromanski and et al., 2021; Shen et al., 2021).

1.4. Notations

We denote scalars by lower- or upper-case letters. We also denote vectors and matrices by lower- and upper-case boldface letters, respectively. For a vector $\boldsymbol{x} = (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$, where $(x_1, \ldots, x_d)^\top$ denotes the transpose of the vector $(x_1, \ldots, x_d)$, we use $\|\boldsymbol{x}\| = (\sum_{i=1}^d |x_i|^2)^{1/2}$ to denote its ℓ_2 norm. We denote the vector whose entries are all 0s as $\mathbf{0}$. For a matrix $\boldsymbol{A}$, we use $\boldsymbol{A}^\top$, $\boldsymbol{A}^{-1}$, and $\|\boldsymbol{A}\|$ to denote its transpose, inverse, and spectral norm, respectively. We use $\boldsymbol{I}$ to denote the identity matrix, whose dimension can be determined in its context. For a function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, we denote its gradient as $\nabla f(\boldsymbol{x})$. Given two sequences $\{a_n\}$ and $\{b_n\}$, we write $a_n = \mathcal{O}(b_n)$ if there exists a positive constant $0 < C < +\infty$ such that $a_n \leq C b_n$.

1.5. Organization

We organize this paper as follows: In section 2, we review the kernelization trick used to linearize the softmax attention and the RNN formulation of the linear transformer with causal masking. In section 3, we present the momentum transformer and adaptive momentum transformer, providing the motivation and detailed derivation. We verify the efficiency of our momentum-based transformers on various applications, including both autoregressive and non-autoregressive tasks in section 4. The paper ends up with concluding remarks.

2. Linear Transformer

Transformers learn long-term dependencies in sequences effectively and concurrently through the self-attention mechanism. By denoting $k(\boldsymbol{q}_i, \boldsymbol{k}_j) := \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sqrt{D})$, we can rewrite (1) as

$$\hat{\boldsymbol{v}}_i = (\sum_{j=1}^N k(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j)/(\sum_{j=1}^N k(\boldsymbol{q}_i, \boldsymbol{k}_j)).$$

In linear transformers (Wang et al., 2020b; Katharopoulos et al., 2020; Choromanski and et al., 2021; Shen et al., 2021), the feature map $k(\boldsymbol{q}_i, \boldsymbol{k}_j)$ is linearized as the product of feature maps $\phi(\cdot)$ on the vectors $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$, i.e., $k(\boldsymbol{q}_i, \boldsymbol{k}_j) = \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)$. The associative property of matrix multiplication is then utilized to derive the following efficient computation of the attention map

$$\hat{\boldsymbol{v}}_i = \frac{\sum_{j=1}^N k(\boldsymbol{q}_i, \boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^N k(\boldsymbol{q}_i, \boldsymbol{k}_j)} = \frac{\sum_{j=1}^N \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)\boldsymbol{v}_j}{\sum_{j=1}^N \phi(\boldsymbol{q}_i)^\top \phi(\boldsymbol{k}_j)} = \frac{\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^N \phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top}{\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^N \phi(\boldsymbol{k}_j)}. \tag{5}$$

In the matrix-product form, we can further write (5) as follows

$$\hat{\boldsymbol{V}} = \frac{\phi(\boldsymbol{Q})(\phi(\boldsymbol{K})^\top \boldsymbol{V})}{\phi(\boldsymbol{Q})\phi(\boldsymbol{K})^\top}. \tag{6}$$

Replacing $(\phi(\boldsymbol{Q})\phi(\boldsymbol{K}^\top))\boldsymbol{V}$ with $\phi(\boldsymbol{Q})(\phi(\boldsymbol{K}^\top)\boldsymbol{V})$ reduces the memory and computational cost of computing the attention map from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$, making linear transformers scalable to very long sequences.

Furthermore, causal masking can be easily implemented in the linearized attention by truncating the summation term in the last equation of (5), resulting in

$$\hat{\boldsymbol{v}}_i = \frac{\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^{i} \phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top}{\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^{i} \phi(\boldsymbol{k}_j)} := \frac{\phi(\boldsymbol{q}_i)^\top \boldsymbol{s}_i}{\phi(\boldsymbol{q}_i)^\top \boldsymbol{z}_i}, \tag{7}$$

where $\boldsymbol{s}_i = \sum_{j=1}^{i} \phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top$ and $\boldsymbol{z}_i = \sum_{j=1}^{i} \phi(\boldsymbol{k}_j)$. The states $\boldsymbol{s}_i$ and $\boldsymbol{z}_i$ can be computed in a recurrent fashion.

Efficient inference via the RNN formulation. Self-attention processes tokens of a sequence concurrently, enabling fast training of transformers via layerwise parallelism. However, during inference, the output for timestep i is the input for timestep $i+1$. As a result, the inference in standard transformers cannot be parallelized and is thus computationally inefficient. Linear transformers provide an elegant approach to fixing this issue by leveraging their RNN formulation. In particular, we can further write the linear attention with causal masking in (7) into the following RNN form[1]

$$\begin{aligned} \boldsymbol{s}_i &= \boldsymbol{s}_{i-1} + \phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top; \\ \boldsymbol{z}_i &= \boldsymbol{z}_{i-1} + \phi(\boldsymbol{k}_i); \\ \hat{\boldsymbol{v}}_i &= \frac{\phi(\boldsymbol{q}_i)^\top \boldsymbol{s}_i}{\phi(\boldsymbol{q}_i)^\top \boldsymbol{z}_i}, \end{aligned} \tag{8}$$

where $\boldsymbol{s}_0 = \boldsymbol{0}$ and $\boldsymbol{z}_0 = \boldsymbol{0}$. Note that this RNN formulation of linear transformers with causal masking contains two memory states $\boldsymbol{s}_i$ and $\boldsymbol{z}_i$.

3. Momentum Transformer

In this section, we present the *momentum transformer*. We start by integrating the heavy ball momentum into the RNN formulation of causal linear attention in (8), resulting in the causal momentum attention. Next, we generalize the causal momentum attention to momentum attention that can efficiently train the model. Moreover, we propose the *momentum connection* to replace residual connections between the attention $\hat{\boldsymbol{V}}$ and the input $\boldsymbol{X}$ in (4) to boost the model's performance. Finally, we derive the adaptive momentum attention from the theory of optimal choice of momentum for the heavy ball method.

1. For simplicity, we omit the nonlinearity (a two-layer feedforward network) compared to (Katharopoulos et al., 2020).

3.1. Momentum Transformer

Heavy ball momentum. Let us recall the heavy ball momentum for accelerating gradient descent in solving $\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x})$ (Polyak, 1964). Starting from $\boldsymbol{x}^0$ and $\boldsymbol{x}^1$, the heavy ball method iterates as follows

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma\nabla f(\boldsymbol{x}^k) + \beta(\boldsymbol{x}^k - \boldsymbol{x}^{k-1}), \tag{9}$$

where $\gamma > 0$ is the step size and $0 \le \beta < 1$ is the momentum parameter. By introducing the momentum state $\boldsymbol{m}$, we can rewrite the HB method as

$$\boldsymbol{m}^{k+1} = \beta\boldsymbol{m}^k + \nabla f(\boldsymbol{x}^k); \quad \boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma\boldsymbol{m}^{k+1}. \tag{10}$$

In contrast, gradient descent updates at each step as follows

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k - \gamma\nabla f(\boldsymbol{x}^k). \tag{11}$$

Integrating momentum into causal linear attention. Now we consider integrating the heavy ball momentum into causal linear attention. We integrate momentum into the state $\boldsymbol{s}_i$ in (8) only since the denominator in causal linear attention is simply a normalizing scalar. If we regard $-\phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top$ as the gradient vector in (11), then we can add momentum into the state $\boldsymbol{s}_i$ by following the heavy ball method in (10), resulting in the following RNN formulation of causal momentum attention,

$$\begin{aligned}
\boldsymbol{m}_i &= \beta\boldsymbol{m}_{i-1} - \phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top;\\
\boldsymbol{s}_i &= \boldsymbol{s}_{i-1} - \gamma\boldsymbol{m}_i;\\
\boldsymbol{z}_i &= \boldsymbol{z}_{i-1} + \phi(\boldsymbol{k}_i);\\
\hat{\boldsymbol{v}}_i &= \frac{\phi(\boldsymbol{q}_i)^\top\boldsymbol{s}_i}{\phi(\boldsymbol{q}_i)^\top\boldsymbol{z}_i},
\end{aligned} \tag{12}$$

where $\boldsymbol{m}_0 = \boldsymbol{0}$, and $\gamma > 0$ and $0 \le \beta < 1$ are two hyperparameters. The RNN formulation of causal momentum attention in (12) is efficient for autoregressive inference. For efficient training, we need to rewrite (12) into a form that is similar to the linear attention in (7). To this end, we need to eliminate the states $\boldsymbol{m}_i$, $\boldsymbol{s}_i$, and $\boldsymbol{z}_i$ from (12). Notice that

$$\boldsymbol{s}_i = \boldsymbol{s}_{i-1} - \underbrace{\gamma\boldsymbol{m}_i}_{:=\boldsymbol{p}_i} = \underbrace{\boldsymbol{s}_0}_{=\boldsymbol{0}} - \Big(\boldsymbol{p}_i + \boldsymbol{p}_{i-1} + \ldots + \boldsymbol{p}_1\Big),$$

since $\boldsymbol{m}_i = \beta\boldsymbol{m}_{i-1} - \phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top$, we have $\boldsymbol{p}_i = \beta\boldsymbol{p}_{i-1} - \gamma\phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top$. Therefore,

$$\begin{aligned}
\boldsymbol{s}_i &= -(\boldsymbol{p}_i + \boldsymbol{p}_{i-1} + \ldots + \boldsymbol{p}_1)\\
&= \gamma\phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top - \Big((1+\beta)\boldsymbol{p}_{i-1} + \boldsymbol{p}_{i-2} + \ldots + \boldsymbol{p}_1\Big)\\
&= \gamma\phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top + \gamma(1+\beta)\phi(\boldsymbol{k}_i)\boldsymbol{v}_i^\top\\
&\quad - \Big((1+\beta)^2\boldsymbol{p}_{i-2} + \ldots + \boldsymbol{p}_1\Big)\\
&= \ldots\\
&= \gamma\sum_{j=1}^{i}\frac{1-\beta^{i-j+1}}{1-\beta}\phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top \text{ for } i \ge 1.
\end{aligned}$$

We can then formulate the causal momentum attention as follows

$$\hat{\boldsymbol{v}}_i = \frac{\gamma\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^{i} \left(\frac{1-\beta^{i-j+1}}{1-\beta}\phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top\right)}{\phi(\boldsymbol{q}_i)^\top \boldsymbol{z}_i}. \tag{13}$$

Note that (13) is mathematically equivalent to (12), but it can be trained much more efficiently in a concurrent fashion via layerwise parallelism.

Remark 1 *Comparing (13) with (7) , we see that momentum plays a role in reweighting the terms* $\{\phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top\}_{j=1}^{i}$. *It is interesting to note that this reweighting is opposite to that used for reweighting the local attention (Dai et al., 2019). It has also been noticed that low-rank attention can complement local attention, resulting in improved performance (Nguyen et al., 2021). Often local attention behaves quite differently from low-rank attention, and different reweighting can be beneficial. We leave the study of reweighting local attention and low-rank attention differently as future work.*

Integrating momentum into linear attention. To obtain momentum attention without causal masking, we can simply take the sum from 1 to N instead of summing from 1 to i. Therefore, we obtain the following momentum attention

$$\hat{\boldsymbol{v}}_i = \frac{\gamma\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^{N} \left(\frac{1-\beta^{N-j+1}}{1-\beta}\phi(\boldsymbol{k}_j)\boldsymbol{v}_j^\top\right)}{\phi(\boldsymbol{q}_i)^\top \sum_{j=1}^{N} \phi(\boldsymbol{k}_j)}. \tag{14}$$

Memory and computational complexity. It is clear that training momentum transformers have the same memory and computational complexities of $\mathcal{O}(N)$ as the training of linear transformers. For test and inference, momentum transformers also have the same memory and computational complexities as linear transformers. However, in the RNN form, momentum transformers require slightly more memory than linear transformers to store the extra momentum state $\boldsymbol{m}_i$.

3.2. Momentum Connection

On top of the self-attention unit, each transformer layer has a residual connection between the self-attention output and the input as shown in (4). We further integrate momentum into (4) and derive the momentum connection as follows

$$T_\ell(\boldsymbol{X}) = f_\ell\big(\hat{\boldsymbol{V}} + \boldsymbol{X} + \tilde{\beta}(\boldsymbol{X} - T_{\ell-1}(\boldsymbol{X}))\big), \tag{15}$$

where $0 \leq \tilde{\beta} < 1$ is a hyperparameter.

3.2.1. ADAPTIVE MOMENTUM

Our momentum transformer introduces additional hyperparameters γ and β, as well as $\tilde{\beta}$, compared to the linear transformer. In section 4, we show that γ can be simply set to 1 in many experiments. However, tuning the momentum-related hyperparameters β and $\tilde{\beta}$ can introduce extra computational cost for training transformers. Moreover, using a constant momentum may not give us optimal performance. In this section, we will introduce an adaptive momentum formula for computing the momentum hyperparameter in momentum connection and thus eliminating the computational overhead for tuning $\tilde{\beta}$. Here, the adaptive momentum does not apply to β since it will break the closed unrolling form in (13).

Optimal choice of heavy ball momentum. We motivate our adaptive momentum from the optimal choice of the heavy ball momentum for solving the following quadratic minimization problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{b}. \tag{16}$$

The following theorem gives the optimal choice of β for (10) to solve (16).

Theorem 2 *Let $f(\boldsymbol{x})$ be the quadratic function (16) with $\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, where $\boldsymbol{A}$ is positive definite. Moreover, we denote the smallest ($\lambda_{\min}(\boldsymbol{A})$) and the largest eigenvalues ($\lambda_{\max}(\boldsymbol{A})$) of $\boldsymbol{A}$ as ν and L, respectively. Given any fixed step size $\gamma \leq 1/L$, the optimal choice for β is $\beta = (1 - \sqrt{\gamma\nu})^2$. In this case, the heavy ball method achieves a convergence rate of*

$$\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^*\| \leq (1 - \sqrt{\gamma\nu})\|\boldsymbol{x}^k - \boldsymbol{x}^*\|,$$

where $\boldsymbol{x}^*$ is the minimum of the quadratic function (16).

Theorem 2 shows that the optimal momentum for the heavy ball method should be $(1 - \sqrt{\gamma\nu})^2$ if $\gamma \leq 1/L$. However, the smallest eigenvalue ν is usually unknown. Therefore, we consider constructing the sequence $\{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|/\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|\}_{k\geq 1}$ to approximate ν. We have the following theoretical result to guarantee that $\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|/\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\| \to \nu$ as $k \to \infty$.

Proposition 3 *Assume that conditions in Theorem 2 hold and $\{\boldsymbol{x}^k\}_{k\geq 0}$ is generated by the heavy ball method (10). If $\gamma \leq 1/L$, for any fixed $0 \leq \beta < 1$, we have*

$$\lim_{k\to\infty} \frac{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|} = \nu.$$

We will prove Theorem 2 and Proposition 3 in Appendix A. In practice, for a given step size γ, we restrict the adaptive momentum to be in the range $[0, 1-\delta]$ with δ being the threshold parameter, and we choose it to be 10^{-3} in this work. Hence, we have the following adaptive momentum

$$\mathbf{proj}_{[0,1-\delta]}\left(1 - \sqrt{\gamma\frac{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|}}\right)^2, \tag{17}$$

where $\mathbf{proj}_{[0,1-\delta]}(\cdot) := \max(0, \min(1-\delta, \cdot))$. To simplify our computation, we apply the gradient descent update to approximate $\boldsymbol{x}^k - \boldsymbol{x}^{k-1}$, i.e., we approximate $\boldsymbol{x}^k - \boldsymbol{x}^{k-1}$ by $\gamma\nabla f(\boldsymbol{x}^{k-1})$, and we end up with

$$\beta_k := \mathbf{proj}_{[0,1-\delta]}\left(1 - \sqrt{\frac{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|}{\|\nabla f(\boldsymbol{x}^{k-1})\|}}\right)^2. \tag{18}$$

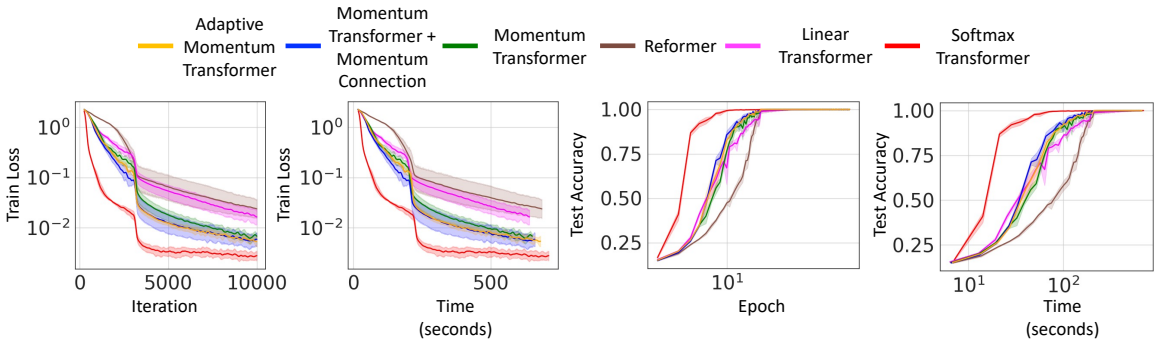

Figure 1: Convergence comparison of adaptive momentum, momentum, reformer, linear, and softmax transformer on the sequence copy task. Momentum and adaptive momentum transformers converge faster and achieve better training loss than both linear transformer and reformer. Softmax transformer converges the fastest but suffers from quadratic memory and computational complexity. Adaptive momentum transformer performs as well as momentum transformer without intensively searching for momentum values. The training time per epoch for softmax transformer, reformer, linear transformer, our momentum transformer without and with momentum connection, and our adaptive momentum transformer are 7.1, 6.9, 6.4, 6.5, 6.6, and 6.8 seconds, respectively. Here, the computational time per epoch does not reveal a significant advantage of efficient transformers over the softmax transformer because the sequence length is only 128.

4. Experimental Results

In this section, we evaluate the benefits of our momentum transformers in terms of convergence speed, efficiency, and accuracy. We compare the performance of momentum and adaptive momentum transformers with the baseline standard softmax transformer and several other efficient transformers in the following tasks: 1) the synthetic copy task, 2) the MNIST and CIFAR image generation task, 3) Long-Range Arena (Tay et al., 2021), and 4) the non-autoregressive machine translation task. These tasks are among standard benchmarks for measuring the performance of transformers and their efficiency. The tasks we choose also cover different data modalities - text and image - and a variety of model sizes. Our experimental results confirm that momentum and adaptive momentum transformers outperform many existing efficient transformers, including linear transformers and reformers, in accuracy and convergence. Furthermore, adaptive momentum transformer improves over momentum transformer without the need of searching for momentum hyperparameter. Values of momentum-related hyperparameters for experiments in our experiments are provided in Table 1 below.

4.1. Copy Task

We train momentum transformers and baseline models on a synthetic copy task to analyze their convergence speed. In this task, the model has to duplicate a sequence of symbols. Each training and test sample has the form $0w0w$ where w is a sequence of symbols collected from the set $\{1, \ldots, N\}$. An example with the word w of length 3 is given below.

Table 1: Momentum and Stepsize Hyperparameteres for Momentum-based Transformers.

Model	Momentum	Stepsize	Momentum in Momentum Connection	Stepsize in Momentum Connection
Copy Task				
Momentum transformer	0.1	0.6		
Momentum transformer + Momentum connection	0.1	0.6	0.99	0.99
Adatptive momentum transformer	0.1	0.6		0.99
MNIST Generation				
Momentum transformer	0.6	0.9		
Momentum transformer + Momentum connection	0.6	0.9	0.1	0.99
Adatptive momentum transformer	0.6	0.9		0.99
CIFAR Generation				
Momentum transformer	0.1	0.9		
Momentum transformer + Momentum connection	0.1	0.9	0.1	0.9
Adatptive momentum transformer	0.1	0.9		0.9
Non-Autoregressive Machine Translation				
Momentum transformer	0.6	0.6		
Momentum transformer + Momentum connection	0.6	0.6	0.3	0.9
Adatptive momentum transformer	0.6	0.6		0.9
ListOps				
Momentum transformer	0.1	0.6		
Adatptive momentum transformer	0.1	0.6		0.4
Text				
Momentum transformer	0.6	2.0		
Adatptive momentum transformer	0.6	2.0		0.001
Retrieval				
Momentum transformer	0.6	1.0		
Adatptive momentum transformer	0.6	1.0		0.5
Image				
Momentum transformer	0.9	0.9		
Adatptive momentum transformer	0.9	0.9		0.001
Pathfinder				
Momentum transformer	0.3	0.1		
Adatptive momentum transformer	0.3	0.1		0.8

Example:	0	15	124	71	0	15	124	71

In our experiments, we follow the same experimental setting as that used by Katharopoulos et al. (2020). In particular, we use a sequence of maximum length 128 with 10 different symbols

Method	Bits/dim	Images/sec	
Standard softmax transformer	0.84	0.45	(1×)
Linear transformer	0.85	142.8	(317×)
Momentum transformer	0.84	139.7	(310×)
Momentum transformer + momentum connection	0.82	135.5	(301×)
Adaptive momentum transformer	0.80	134.9	(300×)

Table 2: Momentum transformers achieve better test bits/dim than both softmax and linear transformers on MNIST generation.

separated by a separator symbol. The baseline architecture for all methods is a 4-layer transformer with 8 attention heads and $D = 32$. The models are trained with the RAdam optimizer using a batch size of 64 and a learning rate of 10^{-3} which is reduced to 10^{-4} after 3000 iterations. Figure 1 shows the training loss and the test accuracy over epochs and over GPU time. Both the momentum and the adaptive momentum transformers converge much faster and achieve better training loss than the linear transformer. Notice that while the standard softmax transformer converges the fastest, it has quadratic complexity. Also, note that the adaptive momentum transformer has similar performance as the momentum transformer without the need of tuning for the momentum value.

4.2. Image Generation

Transformers have shown great promise in autoregressive generation applications (Radford et al., 2019; Child et al., 2019), such as autoregressive image generation (Ramesh et al., 2020). However, the training and sampling procedure using transformers are quite slow for these tasks due to the quadratic computational time complexity and the memory scaling with respect to the sequence length. In this section, we train our momentum-based transformers and the baselines with causal masking to predict images pixel by pixel and compare their performance. In particular, we demonstrate that, like linear transformers, both momentum and adaptive momentum transformers are able to generate images much faster than the standard softmax transformer. Furthermore, we show that momentum-based transformers converge much faster than linear transformers while achieving better bits per dimension (bits/dim). We compare the generated images by different models in the Appendix. Note that momentum and adaptive momentum transformers also generate images with constant memory per image like linear transformers.

MNIST. We first examine our momentum-based transformers on the MNIST image generation task. MNIST is a popular benchmark dataset used for image recognition and generation. For all methods, we train a 8-layer transformer with 8 attention heads and the embedding size of 256, which corresponds to 32 dimensions per head. The feedforward dimensions are 4 times larger than the embedding size. A mixture of 10 logistics is used to model the output as in (Salimans et al., 2017). For training, we use the RAdam optimizer with a learning rate of 10^{-4} and train all models for 250 epochs except for the adaptive momentum transformer.

We report the bits/dim and image generation throughput in Table 2. Compared to the linear transformer, all momentum-based transformers not only attain better bits/dim but also have comparable image generation throughput, justifying the linear complexity of our models. In addition,

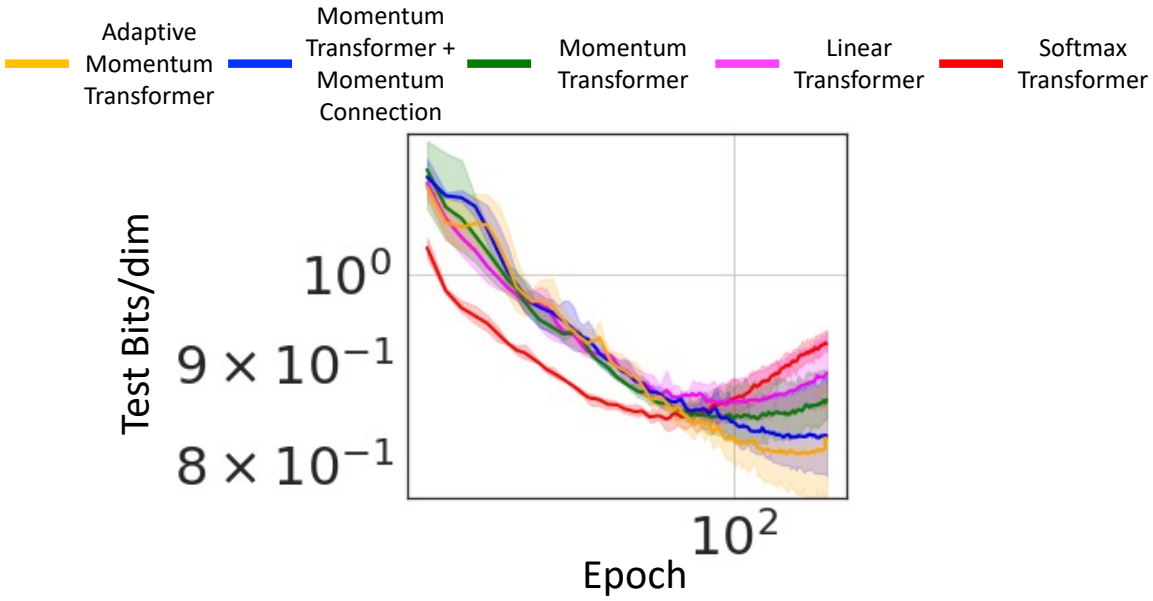

Figure 2: Momentum transformers outperform linear transformers on the MNIST image generation task. Adaptive momentum transformer achieves the best test bits/dim.

we demonstrate that the adaptive momentum transformer converges much faster than the baseline models in Figure 2. Momentum-based transformers even outperform softmax transformers in this task.

We also compare our adaptive momentum transformer with the standard softmax and linear transformer qualitatively. In particular, we analyze the models trained for the MNIST image generation task and show the generated images from each model in Figure 3. We observe that the quality of images generated from the adaptive momentum transformer and linear transformer is as high as the quality of images generated from the softmax transformer while the first two models are much more computational and memory efficient.

CIFAR10. Next, we investigate the advantages of our momentum-based transformers when the sequence length and the number of layers in the model increase. We consider the CIFAR-10 image generation task, in which we train 16-layer transformers to generate CIFAR-10 images. The configuration for each layer is the same as in the MNIST experiment. For the linear transformer and our momentum-based transformer, we use a batch size of 4 while using a batch size of 1 for the standard softmax transformer due to the memory limit of the largest GPU available to us, i.e., NVIDIA V100. This is similar to the setting in (Katharopoulos et al., 2020). Like in the MNIST image generation task, our momentum-based transformers outperform the linear transformer in terms of bits/dim while maintaining comparable image generation throughput. This is a very expensive task, limiting us to perform a thorough hyperparameter search; we believe better results can be obtained with a more thorough hyperparameter search.

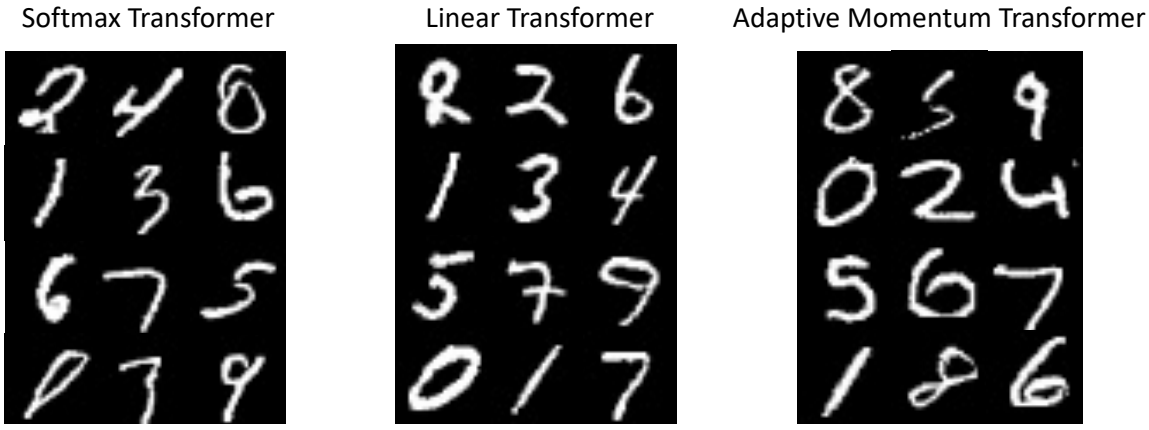

Figure 3: MNIST samples generated by the standard softmax transformer (left) (Vaswani et al., 2017), the linear transformer (middle) (Katharopoulos et al., 2020), and the adaptive momentum transformer (right).

Method	Bits/dim	Images/sec	
Standard softmax transformer	3.20	0.004	(1×)
Linear transformer	3.44	17.85	(4462×)
Momentum transformer	3.43	17.52	(4380×)
Momentum transformer + momentum connection	3.41	17.11	(4277×)
Adaptive momentum transformer	3.38	17.07	(4267×)

Table 3: Momentum-based transformers achieve better test bits/dim than linear transformer on CIFAR10 image generation task.

Model	ListOps (2K)	Text (4K)	Retrieval (4K)	Image (1K)	Pathfinder (1K)	Avg
Softmax	37.10 (37.10)	64.17 (65.02)	80.71 (79.35)	39.06 (38.20)	72.48 (74.16)	**58.70 (58.77)**
Linear	18.30	64.22	81.37	38.29	71.17	54.67
Performer	18.80	63.81	78.62	37.07	69.87	53.63
Reformer	19.05	64.88	78.64	43.29	69.36	55.04
Linformer	**37.25**	55.91	79.37	37.84	67.60	55.59
Momentum transformer	19.56	64.35	81.95	39.40	73.12	55.68
Adaptive momentum transformer	20.16	**64.45**	**82.07**	**39.53**	**74.00**	56.04

Table 4: Results on the LRA tasks. We report the test classification accuracy for each task and average accuracy across all tasks. The momentum-based transformers, in particular, the adaptive momentum transformer, outperforms all other transformers except on the ListOps. The numbers in the parenthesis are from the paper (Xiong et al., 2021). Unit: %.

Method	BLEU Score	Speed (tokens/s)
Standard softmax transformer	24.34	5104
Linear transformer	21.37	1382
Momentum transformer	22.11	1398
Momentum transformer + momentum connection	22.14	1403
Adaptive momentum transformer	22.20	1410

Table 5: BLEU scores and tokens per second from machine translation models trained on IWSLT show the advantages of our momentum-based transformers. The number of trainable parameters is almost the same for all models, up to the small difference introduced by the momentum mechanism in our models. Momentum-based transformers outperform the linear transformer in generation quality in terms of BLEU score and obtain comparable generation efficiency in terms of tokens per second.

4.3. Long-Range Arena

In this experiment, we evaluate our model on tasks that involve longer sequence lengths in the Long Range Arena (LRA) benchmark (Tay et al., 2021). We show that the momentum-based transformer outperforms the baseline linear transformer and other popular efficient transformers, including performer (Choromanski and et al., 2021), reformer (Kitaev et al., 2020), and linformer (Wang et al., 2020b) in all tasks. We also demonstrate that our momentum-based transformer yields better accuracy than the standard softmax transformer (Vaswani et al., 2017) in most tasks except the ListOps. These results justify the advantage of our momentum-based transformers in capturing long-term dependency.

Datasets and metrics We consider all five tasks in the LRA benchmark (Tay et al., 2021), including Listops, byte-level IMDb reviews text classification, byte-level document retrieval, CIFAR-10 image classification on sequences of pixels, and Pathfinder. These tasks involve long sequences of length $2K$, $4K$, $4K$, $1K$, and $1K$, respectively. We follow the setup/evaluation protocol in (Tay et al., 2021) and report the test accuracy for each individual task and the average result across all tasks.

Models and training All models have 2 layers, 64 embedding dimension, 128 hidden dimension, 2 attention heads. Mean pooling is applied in all models. Also, we use the nonlinear activation $elu(x) + 1$ for the linear transformer. Our implementation uses the public code by Xiong et al. (2021) as a starting point, and we follow their training procedures. The training setting and additional baseline model details are provided in the configuration file used in (Xiong et al., 2021) and can be found at https://github.com/mlpen/Nystromformer/blob/main/LRA/code/lra_config.py.

Results We summarize our results in Table 4. Both momentum-based transformers outperform linear transformers in all tasks and yield better accuracy than the standard softmax transformer in most tasks except the Listops. The adaptive momentum transformer performs the best on every task except the LipsOps, far behind the softmax transformer and Linformer.

4.4. Non-Autoregressive Machine Translation

All of the above experiments are for auto-regressive tasks. In this last experiment, we demonstrate that the benefits of our momentum-based transformers also hold for a non-autoregressive task. We consider a machine translation task on the popular IWSLT' 16 En-De dataset. We follow the setting in (Lee et al., 2018). In particular, we tokenize each sentence using a script from Moses (Koehn and et al., 2007) and segment each word into subword units using BPE (Sennrich et al., 2016). We also use $40K$ tokens from both source and target. Our baseline model is the small transformer-based network in (Lee et al., 2018) with $d_{model} = 278$, $d_{hidden} = 507$, $p_{dropout} = 0.1$, $n_{layer} = 5$, and $n_{head} = 2$. This model has 5 layers, and each layer has 2 attention heads. A depiction of this architecture is given in Figure 2 in (Lee et al., 2018). The block "Encoder" encodes the input X, the block "Decoder 1" computes the conditional $\log p(Y^0|X)$, and the block "Decoder 2" is shared across iterative refinement steps, calculating $\log p(Y^\ell|\hat{Y^{\ell-1}}, X)$. For the baseline standard softmax transformer model, we use the same architecture as in (Lee et al., 2018) with an additional positional attention and using the highway layer in the decoders. For the linear and our momentum-based transformer models, we replace the softmax attention with the linear attention and momentum-based attention, respectively. During training, we use linear annealing learning rate scheduling (from 3×10^{-4} to 10^{-5}). We do not use label smoothing nor average multiple check-pointed models.

Table 5 reports the results in terms of generation quality, measured by the BLEU score (Papineni et al., 2002), and generation efficiency, measured by the number of generated tokens per second. Consistent with other experiments above, our momentum-based transformers obtain better BLEU scores than the linear transformer in this non-autoregressive setting. Furthermore, in terms of generation efficiency, momentum-based models are comparable with the linear transformer and much more efficient than the standard softmax transformer.

5. Concluding Remarks

In this paper, we developed a new class of efficient transformers, i.e., momentum transformers, which have the same memory and computational complexity as the recently developed linear transformer. We developed momentum transformers based on an analogy between the RNN formulation of causal linear attention and gradient descent. Then we integrate the momentum into causal linear attention following the heavy ball method. Furthermore, we introduce an additional momentum into the residual connection between the attention $\hat{\boldsymbol{V}}$ and the input $\boldsymbol{X}$ in (4) to further improve the performance of the model. To eliminate the computational overhead for tuning the momentum-related hyperparameters and enhancing momentum transformers' performance, we developed the adaptive momentum transformer that can adaptively compute the momentum values based on the optimal momentum choice for the heavy ball method for quadratic optimization. An interesting observation is that the momentum attention can be understood as a reweighting between the product of the "key" and "value" in the standard attention model. There are numerous avenues for future work: 1) can we develop momentum transformers based on other popular optimization algorithms beyond the heavy ball method, e.g., Adam? And 2) can we design better weighting schemes to improve the performance of transformers?

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 268–284, November 2020. doi: 10.18653/v1/2020.emnlp-main.19. URL https://www.aclweb.org/anthology/2020.emnlp-main.19.

Rami Al-Rfou, DK Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. URL https://arxiv.org/abs/1808.04444.

Akari Asai and Eunsol Choi. Challenges in information seeking qa: Unanswerable questions and paragraph retrieval. *arXiv preprint arXiv:2010.11915*, 2020.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Michael Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

Guy Blanc and Steffen Rendle. Adaptive sampled softmax with kernel based sampling. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 590–599. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/blanc18a.html.

Tom Brown and et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691, 2014.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Marcin Choromanski and et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

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.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987.

Gabriel Goh. Why momentum really works. *Distill*, 2(4):e6, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer: Generating music with long-term structure. In *International Conference on Learning Representations*, 2018.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. Structured attention networks. *arXiv preprint arXiv:1702.00887*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Philipp Koehn and et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, June 2007. URL https://www.aclweb.org/anthology/P07-2045.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, October-November 2018. doi: 10.18653/v1/D18-1149. URL https://www.aclweb.org/anthology/D18-1149.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

Huan Li, Yibo Yang, Dongmin Chen, and Zhouchen Lin. Optimization algorithm inspired deep neural network structure design. In *Asian Conference on Machine Learning*, pages 614–629. PMLR, 2018.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

Thomas Moreau and Joan Bruna. Understanding the learned iterative soft thresholding algorithm with matrix factorization. *arXiv preprint arXiv:1706.01338*, 2017.

Radford M Neal et al. MCMC using Hamiltonian dynamics.

Arkaddii S Nemirovskii and Yu E Nesterov. Optimal methods of smooth convex minimization. *USSR Computational Mathematics and Mathematical Physics*, 25(2):21–30, 1985.

Yurii E Nesterov. A method for solving the convex programming problem with convergence rate o (1/kˆ 2). In *Dokl. Akad. Nauk Sssr*, volume 269, pages 543–547, 1983.

Tan Nguyen, Richard Baraniuk, Andrea Bertozzi, Stanley Osher, and Bao Wang. MomentumRNN: Integrating Momentum into Recurrent Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Tan Minh Nguyen, Vai Suliafu, Stanley Osher, Long Chen, and Bao Wang. FMMformer: Efficient and flexible transformer via decomposed near-field and far-field attention. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=YTKwvw7XI1.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/parmar18a.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=QtTKTdVrFBB.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Mark Chen, Rewon Child, Vedant Misra, Pamela Mishkin, Gretchen Krueger, Sandhini Agarwal, and Ilya Sutskever. Dall·e: Creating images from text. *OpenAI blog*, 2020.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/e43739bba7cdb577e9e3e4e42447f5a5-Paper.pdf.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. doi: 10.1162/tacl_a_00353. URL https://www.aclweb.org/anthology/2021.tacl-1.4.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight memory systems. *CoRR*, abs/2102.11174, 2021. URL https://arxiv.org/abs/2102.11174.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, August 2016. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.

David R So, Chen Liang, and Quoc V Le. The evolved transformer. *arXiv preprint arXiv:1901.11117*, 2019.

Kyungwoo Song, Yohan Jung, Dongjun Kim, and Il-Chul Moon. Implicit kernel attention. *arXiv preprint arXiv:2006.06147*, 2021.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020a.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse Sinkhorn attention. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9438–9447. PMLR, 13–18 Jul 2020b. URL http://proceedings.mlr.press/v119/tay20a.html.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020c.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qVyeW-grC2k.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for transformer's attention via the lens of kernel. *arXiv preprint arXiv:1908.11775*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33, 2020.

Bao Wang, Tan M Nguyen, Andrea L Bertozzi, Richard G Baraniuk, and Stanley J Osher. Scheduled restart momentum for accelerated stochastic gradient descent. *arXiv preprint arXiv:2002.10583*, 2020a.

Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020b.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, June 2018. doi: 10.18653/v1/N18-1101. URL https://www.aclweb.org/anthology/N18-1101.

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based Algorithm for Approximating Self-Attention. 2021.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2021.

Appendix for

Momentum Transformer: Closing the Performance Gap Between Self-attention and Its Linearization

Appendix A. Proof of Theorem 2 and Proposition 3

A.1. Proof of Theorem 2

Let $\boldsymbol{y}^k := \begin{bmatrix} \boldsymbol{x}^k \\ \boldsymbol{x}^{k-1} \end{bmatrix}$, the heavy ball iteration can be rewritten as

$$\boldsymbol{y}^{k+1} = \boldsymbol{T}\boldsymbol{y}^k + \begin{bmatrix} -\gamma \boldsymbol{b} \\ \mathbf{0} \end{bmatrix},$$

where

$$\boldsymbol{T} = \begin{pmatrix} (1+\beta)\boldsymbol{I} - \gamma \boldsymbol{A} & -\beta \boldsymbol{I} \\ \boldsymbol{I} & \mathbf{0} \end{pmatrix} \text{ with } \boldsymbol{I} \text{ be the identity matrix.} \tag{19}$$

Let $\boldsymbol{x}^*$ be the minimizer of $f(\boldsymbol{x})$, which satisfies $\boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{b} = \mathbf{0}$. Then, $\boldsymbol{y}^* := \begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{x}^* \end{bmatrix}$ satisfies that

$$\boldsymbol{T}\boldsymbol{y}^* = \boldsymbol{y}^* + \begin{bmatrix} -\gamma \boldsymbol{b} \\ \mathbf{0} \end{bmatrix}.$$

Therefore, we have

$$\boldsymbol{y}^k - \boldsymbol{y}^* = \boldsymbol{T}(\boldsymbol{y}^{k-1} - \boldsymbol{y}^*) = \boldsymbol{T}^k(\boldsymbol{y}^0 - \boldsymbol{y}^*).$$

Then we turn to explore the eigenvalues of $\boldsymbol{T}$, i.e., the complex number λ satisfying

$$\det \begin{pmatrix} (\lambda - 1 - \beta)\mathbf{I} + \gamma \boldsymbol{A} & \beta \mathbf{I} \\ -\mathbf{I} & \lambda \mathbf{I} \end{pmatrix} = 0.$$

Note that $\boldsymbol{T}$ is non-singular, thus all eigenvalues of $\boldsymbol{T}$ are nonzero. Therefore, we can rewrite the above equation as

$$\det \begin{pmatrix} (\lambda + \frac{\beta}{\lambda} - 1 - \beta)\mathbf{I} + \gamma \boldsymbol{A} & \mathbf{0} \\ -\mathbf{I} & \lambda \mathbf{I} \end{pmatrix} = 0 \Longrightarrow \det((\lambda + \frac{\beta}{\lambda})\mathbf{I} - [(1+\beta)\mathbf{I} - \gamma \boldsymbol{A}]) = 0.$$

Hence for λ^* being any eigenvalue of $(1+\beta)\mathbf{I} - \gamma \boldsymbol{A}$, we just need to consider the following equation

$$\lambda + \frac{\beta}{\lambda} = \lambda^*. \tag{20}$$

To guarantee convergence of the heavy ball iteration, λ is required to satisfy $|\lambda| < 1$.

Optimal choice for β: For any fixed step size $\gamma \leq \frac{1}{L}$, $(1+\beta) - \gamma L \leq \lambda^* \leq (1+\beta) - \gamma \nu$.

a) If $(1+\beta) - \gamma L \geq 2\sqrt{\beta}$, (that is $\beta \leq (1 - \sqrt{rL})^2$), the larger root of (20) is

$$\frac{\lambda^* + \sqrt{(\lambda^*)^2 - 4\beta}}{2} \leq \lambda^* \leq (1+\beta) - \gamma \nu.$$

We want the right hand side to be as small as possible and we set $\beta = 0$, thus

$$\lambda_{\max} \leq 1 - \gamma\nu.$$

b) If $(1+\beta) - \gamma\nu \leq 2\sqrt{\beta}$, (that is $\beta \geq (1-\sqrt{\gamma\nu})^2$), the equation has complex roots whose norms are both β. The optimal choice is then $\beta = (1-\sqrt{\gamma\nu})^2$ and we have

$$\lambda_{\max} \leq 1 - \sqrt{\gamma\nu}.$$

It is easy to see that $1-\sqrt{\gamma\nu}$ is smaller than $1-\gamma\nu$.

A.2. Proof of Proposition 3

Recall the definition of the matrix (19), we have

$$(\boldsymbol{y}^{k+1} - \boldsymbol{y}^k) = \boldsymbol{T}(\boldsymbol{y}^k - \boldsymbol{y}^{k-1}).$$

For λ^* being the largest eigenvalue of $(1+\beta)\mathbf{I} - \gamma\boldsymbol{A}$, the larger root is $\frac{\lambda^* + \sqrt{(\lambda^*)^2 - 4\beta}}{2}$. Thus, the largest eigenvalue of $\boldsymbol{T}$ is

$$t^* := \frac{(1+\beta-\gamma\nu) + \sqrt{(1+\beta-\gamma\nu)^2 - 4\beta}}{2}.$$

Let $\boldsymbol{u}$ be the vector satisfying $[(1+\beta)\mathbf{I} - \gamma\boldsymbol{A}]\boldsymbol{u} = \lambda^*\boldsymbol{u}$, and $\boldsymbol{u}$ is the eigenvector corresponds the minimum eigenvalue of $\boldsymbol{A}$. We can check that

$$\boldsymbol{T}\begin{pmatrix} t^*\boldsymbol{u} \\ \boldsymbol{u} \end{pmatrix} = t^*\begin{pmatrix} t^*\boldsymbol{u} \\ \boldsymbol{u} \end{pmatrix}.$$

That means $\boldsymbol{v} := \begin{pmatrix} t^*\boldsymbol{u} \\ \boldsymbol{u} \end{pmatrix}$ is the eigenvector with t^*. Due to the fact that $\boldsymbol{A}$ has a unique minimum eigenvalue, and $\boldsymbol{T}$ has a unique maximum eigenvalue. The Jordan canonical form indicates that there exists $\mathbf{P} = [\boldsymbol{v}, ...]$ such that

$$\mathbf{P}\begin{pmatrix} t^* & \\ & \Lambda \end{pmatrix}[\mathbf{P}]^{-1} = T, \quad \lim_k (\Lambda/t^*)^k = \mathbf{0}$$

With the definition $\boldsymbol{z}^k := \begin{bmatrix} \boldsymbol{x}^k - \boldsymbol{x}^{k-1} \\ \boldsymbol{x}^{k-1} - \boldsymbol{x}^{k-2} \end{bmatrix}$, we have

$$\boldsymbol{z}^k = T^{k-3}\boldsymbol{z}^3 = \mathbf{P}\begin{pmatrix} (t^*)^{k-3} & \\ & [\Lambda]^{k-3} \end{pmatrix}[\mathbf{P}]^{-1}\boldsymbol{z}^3.$$

As k is large,

$$\boldsymbol{z}^k \approx \mathbf{P}\begin{pmatrix} (t^*)^{k-3} & \\ & 0 \end{pmatrix}[\mathbf{P}]^{-1}\boldsymbol{z}^3 = \{[\mathbf{P}]^{-1}\boldsymbol{z}^3\}_1 (t^*)^{k-3}\boldsymbol{v} \in \text{Range}(\boldsymbol{v}).$$

That is also $\lim_{k\to\infty} \frac{|\langle \boldsymbol{x}^k - \boldsymbol{x}^{k-1}, \boldsymbol{v}\rangle|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|\|\boldsymbol{v}\|} = 1$. Therefore, we are then led to

$$\lim_{k\to\infty} \frac{\|\nabla f(\boldsymbol{x}^k) - \nabla f(\boldsymbol{x}^{k-1})\|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|} = \lim_{k\to\infty} \frac{\|\boldsymbol{A}(\boldsymbol{x}^k - \boldsymbol{x}^{k-1})\|}{\|\boldsymbol{x}^k - \boldsymbol{x}^{k-1}\|} = \nu.$$