QT-ViT: Improving Linear Attention in ViT with Quadratic Taylor Expansion

Yixing Xu, Chao Li, Dong Li, Xiao Sheng, Fan Jiang, Lu Tian, Emad Barsoum
Advanced Micro Devices, Inc.

AMD

NEURAL INFORMATION PROCESSING SYSTEMS

Abstract

- The **time complexity** and **memory consumption** of Vision Transformers **increase quadratically** with the number of input patches.
- **Linear attention** is a way to mitigate the complexity of the original self-attention mechanism **at the expense of effectiveness**.
- To make up for the performance gap, previous methods necessitate knowledge distillation or high-order attention residuals that severely **increase GPU memory** consumption during training, making them unsuitable to train large models.
- We propose **QT-ViT** models that improve the previous linear self-attention using Quadratic Taylor expansion.
- We substitute the softmax-based attention with **second-order Taylor expansion**, and then accelerate the quadratic expansion by reducing the time complexity with a **fast approximation algorithm**.
- Extensive experiments demonstrate the efficiency and effectiveness of the proposed QT-ViTs, showcasing the state-of-the-art results. Particularly, the proposed QT-ViTs consistently surpass the previous SOTA EfficientViTs under different model sizes and achieve a **new Pareto-front in terms of accuracy and speed**.

Preliminaries

Softmax Self-Attention

- Given an input matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ where N is the number of patches and d is the dimension of each patch, the query, key and value matrices are:

$$\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q, \qquad \boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K, \qquad \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V.$$

- Then, the attention score is computed on each pair of patches:

$$\mathbf{O}_k = \sum_{i=1}^{N} \frac{\mathrm{Sim}(\mathbf{Q}_k, \mathbf{K}_i)}{\sum_{j=1}^{N}\mathrm{Sim}(\mathbf{Q}_k, \mathbf{K}_j)}\mathbf{V}_i = \sum_{i=1}^{N} \frac{\exp(\mathbf{Q}_k\mathbf{K}_i^\top/\sqrt{d})}{\sum_{j=1}^{N}\exp(\mathbf{Q}_k\mathbf{K}_j^\top/\sqrt{d})}\mathbf{V}_i$$

- The time complexity of softmax attention is $\mathcal{O}(N^2d)$.

Linear Self-Attention

$$\mathbf{O}_k = \sum_{i=1}^{N} \frac{\phi(\mathbf{Q}_k)\phi(\mathbf{K}_i)^\top}{\sum_{j=1}^{N}\phi(\mathbf{Q}_k)\phi(\mathbf{K}_j)^\top}\mathbf{V}_i = \frac{\phi(\mathbf{Q}_k)\left(\sum_{i=1}^{N}\phi(\mathbf{K}_i)^\top\mathbf{V}_i\right)}{\phi(\mathbf{Q}_k)\left(\sum_{j=1}^{N}\phi(\mathbf{K}_j)^\top\right)}$$

where the time complexity is changed from $\mathcal{O}(N^2d)$ to $\mathcal{O}(Nd^2)$.

- In order to losslessly decompose similarity function $\mathrm{Sim}(\boldsymbol{Q}_k, \boldsymbol{K}_i)$, the dimensionality of $\phi(\cdot)$ need to be infinite.
- A series of instantiations are proposed to compute $\phi(\cdot)$ efficiently while preserving as much information as possible.
- Such as $\phi(x) = \mathrm{elu}(x) + 1$, $\phi(x) = \mathrm{relu}(x)$, $\mathrm{Sim}(\boldsymbol{q}, \boldsymbol{k}) = 1/2 + 1/\pi \cdot (\boldsymbol{q}\boldsymbol{k}^T) + H_r$, etc.
- They need KD or the masked output of original softmax attention H_r to enhance the performance, causing high GPU mem cost.

QT-ViT

Decompose Quadratic Taylor Expansion

- The quadratic Taylor expansion of the similarity function is:

$$\mathrm{Sim}(\boldsymbol{q}, \boldsymbol{k}) = \exp\left(\frac{<\boldsymbol{q},\boldsymbol{k}>}{\sqrt{d}}\right) \approx 1 + \frac{<\boldsymbol{q},\boldsymbol{k}>}{\sqrt{d}} + \frac{<\boldsymbol{q},\boldsymbol{k}>^2}{\sqrt{d}} = \frac{\left(\frac{<\boldsymbol{q},\boldsymbol{k}>}{\sqrt{d}}+1\right)^2+1}{2} = \frac{<\phi(\boldsymbol{q}),\phi(\boldsymbol{k})>^2+1}{2},$$

where $<\cdot,\cdot>$ is the dot product and $\phi(x) = \left[\frac{x}{\sqrt[4]{d}}, 1\right]$ is used for vectors $\boldsymbol{q}$ and $\boldsymbol{k}$.

- It is non-trivial to decompose the above equation into two separate kernel embeddings because of the quadratic term. In the following, we solve this problem by using the Kronecker product.
- Given two vectors $\mathbf{a} = \{a_i\}_{i=1}^d$ and $\mathbf{b} = \{b_i\}_{i=1}^d$, we have:

$$< \mathbf{a}, \mathbf{b} >^2 = \left(\sum_{i=1}^d a_i b_i\right)^2.$$

- This is equal to first compute the Kronecker product of each vector and then applying dot product. Given $K_r(\boldsymbol{x}) = \mathrm{vec}(\boldsymbol{x} \otimes \boldsymbol{x})$ where $\otimes$ is the Kronecker product and $\mathrm{vec}(\cdot)$ is the vectorized output, we have:

$$< K_r(\mathbf{a}), K_r(\mathbf{b}) > = [a_1\mathbf{a}, \cdots, a_d\mathbf{a}] \cdot [b_1\mathbf{b}, \cdots, b_d\mathbf{b}] = \left(\sum_{i=1}^d a_i b_i\right)^2 = < \mathbf{a}, \mathbf{b} >^2$$

- Thus, we have $\mathrm{Sim}(\boldsymbol{q}, \boldsymbol{k}) \approx \frac{<\phi(q),\phi(k)>^2+1}{2} = \frac{<K_r(\phi(q)),K_r(\phi(k))>+1}{2} = < \varphi(\boldsymbol{q}), \varphi(\boldsymbol{k}) >$, where

$$\varphi(\boldsymbol{x}) = \left[\frac{1}{\sqrt{2}}K_r(\phi(\boldsymbol{x})), \frac{1}{\sqrt{2}}\right] = \left[\frac{1}{\sqrt{2}}\mathrm{vec}(\phi(\boldsymbol{x}) \otimes \phi(\boldsymbol{x})), \frac{1}{\sqrt{2}}\right]$$

is the kernel function applied to the query and key vectors.

- Given $\boldsymbol{x} \in \mathbb{R}^d$, we have $K_r(\boldsymbol{x}) \in \mathbb{R}^{d^2}$. Thus, the time complexity of this method is $\mathcal{O}(Nd^3)$, which is bad.

Reduce the Time Complexity from $\mathcal{O}(Nd^3)$ to $\mathcal{O}(Nd^2)$

- By rewriting the definition of $K_r(\phi(\boldsymbol{x}))$ in its element-wise form, we can get:

$$K_r(\phi(\boldsymbol{x})) = K_r\left(\left[\frac{x}{\sqrt[4]{d}}, 1\right]\right) = \left[\frac{x_1}{\sqrt[4]{d}} \cdot \left[\frac{x}{\sqrt[4]{d}}, 1\right], \cdots, \frac{x_d}{\sqrt[4]{d}} \cdot \left[\frac{x}{\sqrt[4]{d}}, 1\right], \left[\frac{x}{\sqrt[4]{d}}, 1\right]\right]$$

$$= \left[\left\{\frac{x_1x_1}{\sqrt{d}}, \cdots, \frac{x_1x_d}{\sqrt{d}}, \frac{x_1}{\sqrt[4]{d}}\right\}, \cdots, \left\{\frac{x_dx_1}{\sqrt{d}}, \cdots, \frac{x_dx_d}{\sqrt{d}}, \frac{x_d}{\sqrt[4]{d}}\right\}, \left\{\frac{x_1}{\sqrt[4]{d}}, \cdots, \frac{x_d}{\sqrt[4]{d}}, 1\right\}\right]$$

- Since the order of the elements in the above equation does not influence the result of the inner product $< K_r(\phi(\boldsymbol{q})), K_r(\phi(\boldsymbol{k})) >$ as long as they change the order of their elements in the same manner.
- Thus, the above equation can be written as:

$$\hat{K}_r(\phi(\boldsymbol{x})) = \mathrm{concat}\left(\frac{\{x_ix_j\}_{i,j=1}^d}{\sqrt{d}}, \frac{\{x_i\}_{i=1}^d}{\sqrt[4]{d}}, \frac{\{x_i\}_{i=1}^d}{\sqrt[4]{d}}, 1\right).$$

- Since the computational load mainly comes from the first quadratic term, we reduce the number of elements in this term by using the self-multiplication terms $\{x_i^2\}_{i=1}^d$ to represent all quadratic terms.
- Therefore, the Kronecker produce can finally be replaced with a compact version:

$$\tilde{K}_r(\phi(\boldsymbol{x})) = \mathrm{concat}\left(\alpha \cdot \sqrt{d}\frac{\{x_i^2\}_{i=1}^d}{\sqrt{d}}, \beta \cdot \sqrt{2}\frac{\{x_i\}_{i=1}^d}{\sqrt[4]{d}}, \gamma\right)$$

$$= \mathrm{concat}\left(\alpha \cdot \{x_i^2\}_{i=1}^d, \beta \cdot \sqrt{\frac{4}{d}}\{x_i\}_{i=1}^d, \gamma\right) \quad (\alpha, \beta, \gamma \text{ are learnable parameters})$$

Experiments

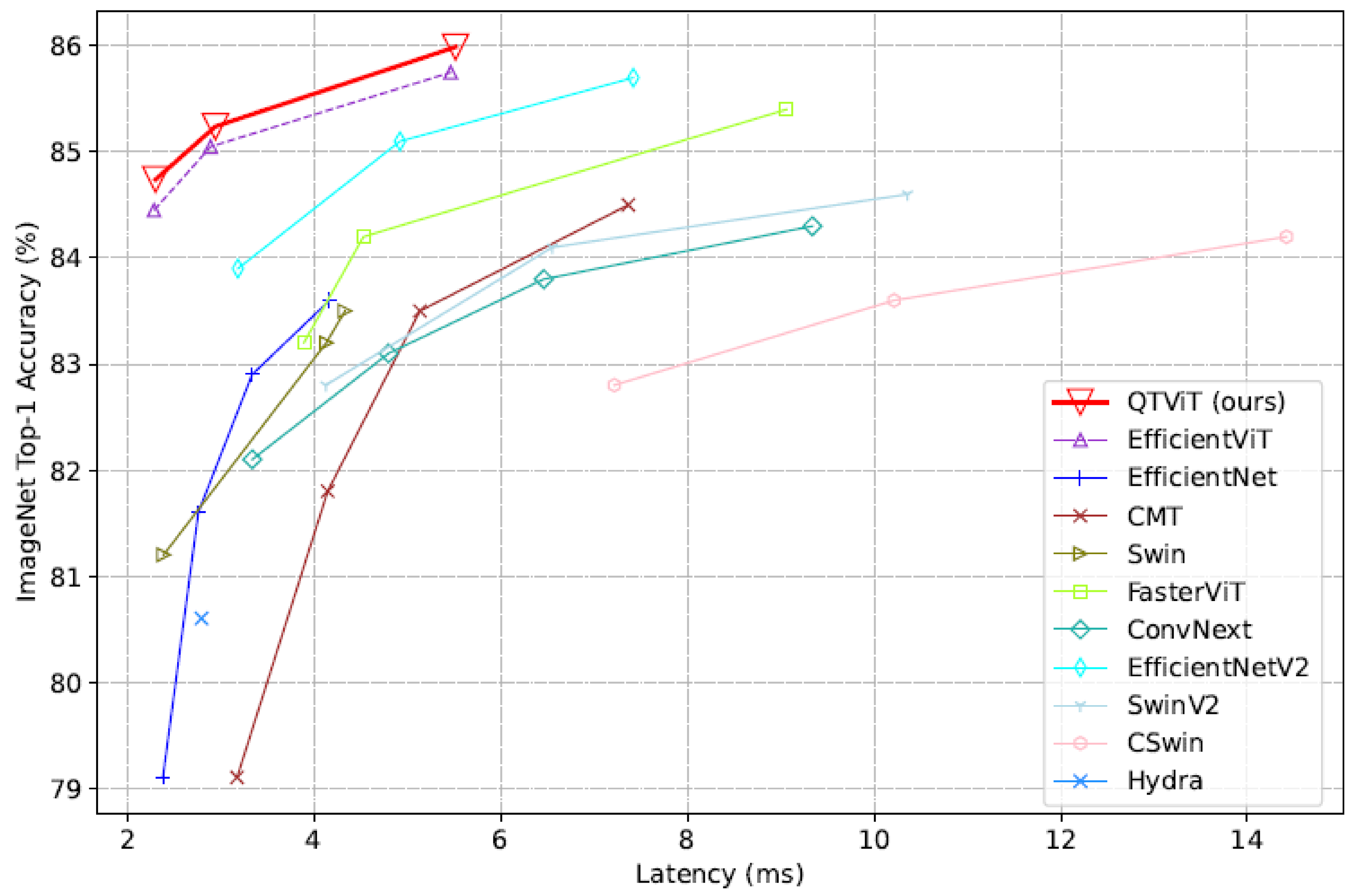

Figure 1: The accuracy-speed trade-offs of the proposed QT-ViTs and other state-of-the-art transformer models on the ImageNet dataset. Latencies are evaluated on the AMD Instinct MI250 GPU.

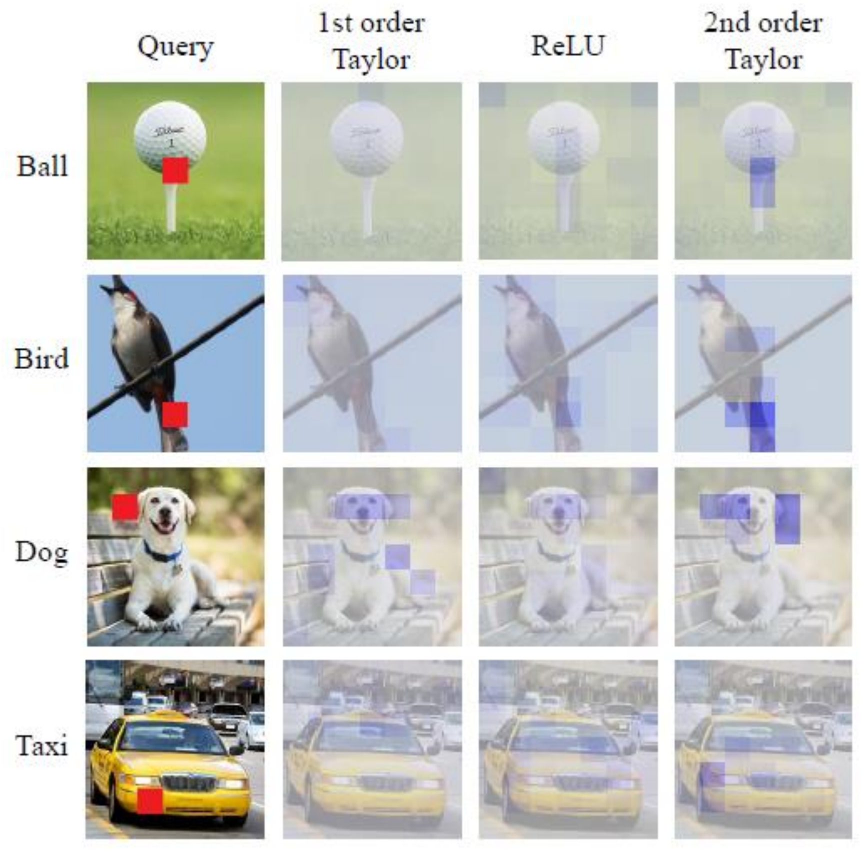

Figure 2: Attention maps from different linear attention methods including the first-order Taylor expansion, ReLU non-linearity function and the second-order Taylor expansion (ours).

Table 2: Experimental results on COCO 2017 dataset using different backbones.

Backbone	AP	AP_{50}	AP_{75}	Params (M)
EfficientViT-B1	39.1	58.0	41.8	57.6
QT-ViT-1	**39.3**	**58.2**	**42.1**	57.9
EfficientViT-B2	40.8	59.5	44.3	68.0
QT-ViT-2	**41.1**	**59.7**	**44.7**	68.5
EfficientViT-B3	42.3	60.6	45.5	92.1
QT-ViT-3	**42.6**	**60.9**	**45.9**	93.1

Table 3: Experimental results on COCO 2017 dataset using different backbones.

Backbone	AP	AP_{50}	AP_{75}
QT-ViT-1 w/ APE	39.3	58.2	42.1
QT-ViT-1 w/o APE	39.2	58.2	42.0
QT-ViT-2 w/ APE	41.1	59.7	44.7
QT-ViT-2 w/o APE	41.0	59.7	44.6
QT-ViT-3 w/ APE	42.6	60.9	45.9
QT-ViT-3 w/o APE	42.5	60.8	45.8