EcoFormer: Energy-Saving Attention with Linear Complexity

Anonymous Author(s)
Affiliation
Address
email

Abstract

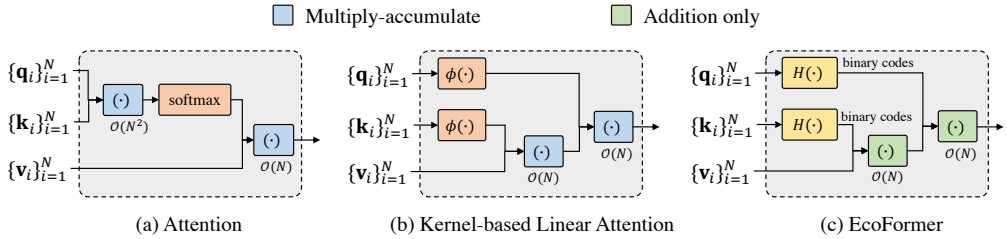
Transformer is a transformative framework for deep learning which models sequential data and has achieved remarkable performance on a wide range of tasks, but with high computational and energy cost. To improve its efficiency, a popular choice is to compress the models via binarization which constrains the floating-point values into binary ones to significantly save resource consumption owing to cheap bitwise operations. However, existing binarization methods only aim at minimizing the information loss for the input distribution statistically, while ignoring the pairwise similarity modeling at the core of the attention mechanism. To this end, we propose a new binarization paradigm customized to high-dimensional softmax attention via kernelized hashing, called EcoFormer, to map the original queries and keys into low-dimensional binary codes in Hamming space. The kernelized hash functions are learned to match the ground-truth similarity relations extracted from the attention map in a self-supervised way. Based on the equivalence between the inner product of binary codes and the Hamming distance as well as the associative property of matrix multiplication, we can approximate the attention in linear complexity by expressing it as a dot-product of binary codes. Moreover, the compact binary representations of queries and keys in EcoFormer enable us to replace most of the expensive multiply-accumulate operations in attention with simple accumulations to save considerable on-chip energy footprint on edge devices. Extensive experiments on both vision and language tasks show that EcoFormer consistently achieves comparable performance with standard attentions while consuming much less resources. For example, based on PVTv2-B0 and ImageNet-1K, EcoFormer achieves 73% reduction in energy footprint with only a slight performance drop of 0.33% compared to the standard attention.

1 Introduction

Recently, Transformers [58] have shown rapid and exciting progress in natural language processing (NLP) [15, 14] and computer vision (CV) [18, 57] due to its extraordinary representational power. Compared with convolutional neural networks (CNNs) [31], Transformer models are generally more scalable to massive amounts of data and better at capturing long-dependency global information with less inductive bias, thus achieving better performance in many tasks [27, 39]. However, the efficiency bottlenecks, especially the high energy consumption, greatly hamper the massive deployment of Transformer models to resource-constrained edge devices, such as mobile phones and unmanned aerial vehicles, for solving a variety of real-world applications.

To reduce the energy consumption, quantization has been actively studied to lower the bit-width representation of network weights [11, 53, 67] and/or activations [26, 69, 68]. With the most aggressive bit-width, binary quantization [50, 2, 47] has attracted much attention since it enables

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Attention (b) Kernel-based Linear Attention (c) EcoFormer

Figure 1: Computational graphs for standard attention (left), kernel-based linear attention (middle) and the proposed EcoFormer based on kernelized hashing (right).

Table 1: Energy cost for different operations (on 45nm CMOS technology) [22, 23, 33].

Operation	16-bit FP Add	16-bit FP Mult	32-bit FP Add	32-bit FP Mult
Energy (pJ)	0.4	1.1	0.9	3.7
Area (μm^2)	1,360	1,640	4,184	7,700

efficient bit-wise operations by representing values with a single bit (*e.g.*, +1 or -1). When we only binarize weights in analogous to BinaryConnect [11], as shown in Table 1, it brings great benefits to dedicated hardware by replacing a large number of energy-hungry multiply-accumulate operations with simple energy-efficient accumulations, which saves significant on-chip area and energy required to run inference with Transformers, making them feasible to be deployed on mobile platforms with limited resources. However, the conventional binarization process typically targets at minimizing quantization errors between the original full-precision data distribution and the quantized Bernoulli distribution statistically. In other words, each token is binarized separately, where the binary representations may not well preserve the original similarity relations among tokens. This motivates us to customize the binarization process to softmax attention, the core mechanism in Transformer that encodes the pairwise similarity between tokens. To this end, we can adapt the well-established hashing methods, to map the high-dimensional queries and keys into compact binary codes (*e.g.*, 16-bits) that are able to preserve the similarity relations in Hamming space. A simple solution is to use the locality-sensitive hashing (LSH) [1] to substitute the binary quantization counterparts.

Nevertheless, another energy bottleneck exists in Transformers. Specially, given a sequence of tokens, the softmax attention obtains the attention weights by computing the inner product between a query token and all key tokens, leading to the quadratic time complexity $\mathcal{O}\left(N^2\right)$ regarding the number of tokens N, as shown in Figure1 (a). This problem is even worse for a long sequence length N, especially for high resolution images in dense prediction tasks. To reduce the complexity of the softmax attention, some prior works propose to express the attention as a linear dot product of kernelized feature embeddings [46, 9]. With the associative property of matrix multiplication, the attention operation can be approximated in linear complexity $\mathcal{O}(N)$, as illustrated by Figure 1 (b).

Based on the hashing mechanism and kernel-based formulation of attention, we devise a simple yet effective energy-saving attention, called EcoFormer, which is shown in Figure 1 (c). In particular, we propose to use kernelized hashing with RBF kernel to map the queries and keys to compact binary codes. The resulting codes are valid for similarity preserving based on the good property that the codes' inner product (*i.e.*, Hamming affinity) and Hamming distance have one-to-one correspondence [37]. Thanks to the associative property of the linear dot-product between the binary codes, the kernelized hashing attention is in *linear complexity* with significant energy saving. Moreover, the pairwise similarity matrix in attention can be directly used to obtain the supervision labels for hash function learning, delivering a novel *self-supervised* hashing paradigm. By maximizing the Hamming affinity on the similar pairs of tokens and simultaneously minimizing on the dissimilar pairs of tokens, the pairwise similarity relations between tokens can be preserved. With *low-dimensional binary* queries and keys, we can replace most of the energy-hungry floating-point multiplications in attention with simple additions, which greatly saves the on-chip energy footprint.

To sum up, we make three main contributions: 1) We propose a new binarization paradigm to better preserve the pairwise similarity in softmax attention. In particular, we present EcoFormer, an energy-saving attention with linear complexity powered by kernelized hashing to map the queries and keys into compact binary codes. 2) We learn the kernelized hash functions based on the ground-truth Hamming affinity extracted from the attention scores in a self-supervised way. 3) Extensive experiments on CIFAR-100, ImageNet-1K and Long Range Arena show that EcoFormer is able to significantly reduce the energy cost while preserving the accuracy. For example, based on PVTv2-B0

and ImageNet-1K, EcoFormer achieves 73% reduction in energy footprint with only a marginal performance drop of 0.33% compared to the standard attention.

2 Related Work

Efficient attention mechanisms. To alleviate the quadratic computational cost for vanilla attention with respect to the number of tokens, much work has endeavored developing efficient attentions. One line of research performs attention only on the part of the tokens [39, 62, 4, 30, 61, 12, 59, 55]. For instance, Reformer [30], SMYRF [12], Fast Transformers [59] and LHA [55] restrict the attention to the most similar token pairs via hashing and reduces the computational complexity to $\mathcal{O}(N \log N)$. Linformer [61] approximates the attention with low-rank factorization that reduces the length of the key and value. However, the computational complexity is dependent on the design for reducing tokens. Another line of research speeds up the vanilla attention with kernel-based methods [9, 65, 42, 29, 48]. For example, Performer [9] approximates the softmax operation with orthogonal random features. Nyströmformer [65] and SOFT [42] approximate the full self-attention matrix via matrix decomposition. Although impressive achievements have been achieved, how to develop attention that is highly energy-efficient remains under-explored, as multiplications dominate the energy consumption. AdderNet variants [8, 54] replace the energy-hungry multiplication-based similarity measurement with the energy-efficient addition-based L1 distance and argue that additions can also provide powerful feature representations. Nevertheless, this heuristic approach brings a drastic performance drop. In contrast, we are from the perspective of binarization and propose to learn kernel-based hash functions using attention scores to map the original features into compact similarity-preserving binary codes in Hamming distance, which is energy-efficient and in linear complexity $\mathcal{O}(N)$. With the low-dimensional binary queries and keys, our EcoFormer is able to replace most of the multiplications with simple accumulations.

Hashing. Hashing is an efficient nearest neighbor search method by embedding the high-dimensional data into a similarity preserving low-dimensional binary codes, based on the intuition that highly similar data should be assigned the same hash key. Hashing methods can be roughly categorized into data-independent and data-dependent schemes. The former focuses on building random hash functions and locally sensitive hashing (LSH) [20, 7] is arguably the most representative one, which guarantees the sub-linear time similarity search and is followed by non-linear extensions such as hashing with kernels [32] or on manifolds [64]. The latter can be further divided into unsupervised [38, 21, 36], semi-supervised [45, 60] and supervised hashing [37, 72, 66]. When it comes to Transformers, as self-attention encodes the pairwise similarity among tokens, hashing is thus a natural choice to efficiently retrieve similar keys given a query. Reformer is such a pioneering work, which proposes to group similar tokens in a single hash bucket to form sparse self-attention. Our Ecoformer is fundamentally different from Reformer in three aspects: 1) Reformer relies on local attention lookups to reduce the complexity while our Ecoformer is designed from a numerical perspective, where the low-dimensional binary codes are used to save the multiplications; 2) Reformer is built upon LSH with linear mapping, which cannot deal with the kernel-based formulation of attention to scale linearly with the sequence length; 3) Our hash functions are self-supervised by the pairwise affinity labels in attention, which are optimized in conjunction with network parameters and more accurate than unsupervised random projections.

Binary quantization. Binarization, an extreme quantization scheme, seeks to represent the vectors by binary codes. As a result, the computationally heavy matrix multiplications become light-weight bitwise operations (*i.e.*, xnor and $\mathrm{popcount}$), yielding promising memory saving and acceleration. In general, to make binary neural networks [25] reliable in accuracy, current research targets at tackling two main challenges. The first challenge is to minimize the quantization error, basically based on learning the scaling factors [50, 6], parameterizing the quantization range and/or intervals [28, 19], and ensembling multiple binary bases [35, 73], *etc.* Another category of studies focus on solving the non-differentiable optimization problem due to the discretization process via training with regularization [17], knowledge distillation [44, 47], relaxed optimization [24, 3], appending full-precision branches [40, 43] and so on. Apart from CNNs, there are some recent pioneering attempts targeting on binarizing Transformers. For example, BinaryBERT [2] proposes to push Transformer quantization to the limit by weight binarization. BiBERT [47] quantizes both weights, embeddings and activations of BERT [16] to 1-bit and achieves considerable savings on FLOPs and model size, but still has obvious performance drop. In contrast, we propose to customize the

binarization paradigm to softmax attention from the hashing perspective, preserving high-fidelity pairwise similarity information in compact binary codes which are used to deliver linear-complexity, energy-efficient self-attention.

3 Preliminaries

3.1 Attention Mechanism

Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ be the input sequence into a standard multi-head self-attention (MSA) layer, where N is the length of the input sequence and D is the number of hidden dimensions. A standard MSA layer calculates a sequence of query, key and value vectors with three learnable projection matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{D \times D_p}$, which can be formulated as

$$\{\mathbf{q}_t\}_{t=1}^N = \mathbf{X}\mathbf{W}_q, \{\mathbf{k}_t\}_{t=1}^N = \mathbf{X}\mathbf{W}_k, \{\mathbf{v}_t\}_{t=1}^N = \mathbf{X}\mathbf{W}_v, \tag{1}$$

where D_p refers to the number of dimensions for each head. For each query vector, the attention output is a weighted-sum over all value vectors as

$$\text{Attention}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) = \sum_i \frac{\exp(\mathbf{q}_t \cdot \mathbf{k}_i / \tau)}{\sum_j \exp(\mathbf{q}_t \cdot \mathbf{k}_j / \tau)} \mathbf{v}_i^\top, \tag{2}$$

where τ is the temperature for controlling the flatness of softmax and $\exp(\langle \cdot, \cdot \rangle)$ is an exponential function. The computation of Eq. (2) has a quadratic complexity of $O(N^2)$ in both space and time, which results in huge computational cost when dealing with long sequences.

3.2 Kernel-based Linear Attention

The idea behind kernel-based linear attention is to express the similarity measure in Eq. (2) as a linear dot-product of kernel embeddings, such as polynomial kernel, exponential or RBF kernel. A particular choice is to employ the finite random mapping [49] $\boldsymbol{\phi}(\cdot)$ to approximate the infinite dimensional RBF kernel. Then, according to the theorem from Rahimi [49], the inner product between a pair of transformed vectors $\mathbf{x}$ and $\mathbf{y}$ with $\boldsymbol{\phi}(\cdot)$ can approximate a Guassian RBF kernel. This gives rise to an unbiased estimation to $\exp(\langle \cdot, \cdot \rangle)$ in Eq. (2), which can be expressed as

$$\begin{aligned} \exp\left(\mathbf{x} \cdot \mathbf{y} / \sigma^2\right) &= \exp\left(\|\mathbf{x}\|^2 / 2\sigma^2 + \|\mathbf{y}\|^2 / 2\sigma^2\right) \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / 2\sigma^2\right) \\ &\approx \exp\left(\|\mathbf{x}\|^2 / 2\sigma^2 + \|\mathbf{y}\|^2 / 2\sigma^2\right) \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{y}). \end{aligned} \tag{3}$$

Assume that the queries and keys are unit vectors, then the attention computation in Eq. (2) can be approximated by

$$\text{Attention}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) \approx \sum_i \frac{\boldsymbol{\phi}(\mathbf{q}_t)^\top \boldsymbol{\phi}(\mathbf{k}_i) \mathbf{v}_i^\top}{\sum_j \boldsymbol{\phi}(\mathbf{q}_t)^\top \boldsymbol{\phi}(\mathbf{k}_j)} \tag{4a}$$

$$= \frac{\boldsymbol{\phi}(\mathbf{q}_t)^\top \sum_i \boldsymbol{\phi}(\mathbf{k}_i) \otimes \mathbf{v}_i}{\boldsymbol{\phi}(\mathbf{q}_t)^\top \sum_j \boldsymbol{\phi}(\mathbf{k}_j)}, \tag{4b}$$

where $\otimes$ refers to the outer product. Recent works have shown that kernel-based linear attentions perform favorably against the original softmax attention on machine translation [46] and protein sequence modeling [9]. However, although the complexity is reduced into linear, the intensive floating-point multiplications in Eq. (4b) still consume a large amount of energy, which can quickly drain the batteries on mobile/edge platforms.

3.3 Binary Quantization

Following [50], binary quantization typically estimates the full-precision $\mathbf{u} \in \mathbb{R}^n$ using a binary $\hat{\mathbf{u}} \in \{+1, -1\}^n$ and a scaling factor $\alpha \in \mathbb{R}^+$ such that $\mathbf{u} \approx \alpha \hat{\mathbf{u}}$ holds. To find an accurate estimation, existing methods [50, 43, 35, 2, 47] minimize the quantization error as

$$\alpha^*, \hat{\mathbf{u}}^* = \arg\min \|\mathbf{u} - \alpha \hat{\mathbf{u}}\|. \tag{5}$$

By solving Problem (5), we have $\hat{\mathbf{u}} = \text{sign}(\mathbf{u})$ and $\alpha = \frac{1}{n}\|\mathbf{u}\|_{\ell 1}$, where $\text{sign}(u)$ returns 1 if $u \geq 0$ and -1 if $u < 0$. Since the sign function is non-differentiable, the straight-through estimator (STE) [5] is applied to approximate the gradient such as using the gradient of hard $\tanh$ [26] or piecewise polynomial function [40].

4 Proposed Method

To reduce the energy consumption of self-attention, one may perform binary quantization [47, 2] on the queries $\{\mathbf{q}_t\}_{t=1}^N$ and keys $\{\mathbf{k}_t\}_{t=1}^N$. In this case, we can replace most of the energy-expensive multiplications with the energy-efficient bit-wise operations. However, existing binary quantization methods only focus on minimizing the quantization error between the original full-precision values and the binary ones as in Eq. (5), which fails to preserve the pairwise semantic similarity between different tokens in attention, leading to performance degradation.

Note that the attention can be seen as applying kernel smoother over pairwise tokens where the kernel scores denote the similarity of the token pairs, as mentioned in Section 3.2. Motivated by this, we propose a new binarization method that applies kernelized hashing with Gaussian RBF to map the original high-dimensional queries/keys to low-dimensional similarity-preserving binary codes in Hamming space. The proposed framework, which we dub *EcoFormer*, is depicted in Figure 1 (c). To maintain the semantic similarity in attention, we learn the hash functions in a self-supervised manner. By exploiting the associative property of the linear dot-product between binary codes and the equivalence between the code inner products (*i.e.*, Hamming affinity) and the Hamming distances, we are able to approximate the self-attention in linear time with low energy cost. In the following, we first introduce the kernelized hashing attention in Section 4.1 and then show how to learn the hash functions in a self-supervised way in Section 4.2.

4.1 Kernelized Hashing Attention

Before applying hash functions, we let the queries $\{\mathbf{q}_t\}_{t=1}^N$ and keys $\{\mathbf{k}_t\}_{t=1}^N$ to be identical following [30, 42]. In this way, we can then apply kernelized hash functions $H : \mathbb{R}^{D_p} \mapsto \{1, -1\}^b$ without explicitly applying transformation $\phi(\cdot)$ mentioned in Section 3.2 to map $\mathbf{q}_i$ and $\mathbf{k}_j$ into b-bit binary codes $H(\mathbf{q}_i)$ and $H(\mathbf{k}_j)$, respectively (see Section 4.2). In this case, the Hamming distance between them can be defined as

$$\mathcal{D}\left(H(\mathbf{q}_i), H(\mathbf{k}_j)\right) = \sum_{r=1}^{b} \mathbb{1}\{H_r(\mathbf{q}_i) \neq H_r(\mathbf{k}_j)\}, \tag{6}$$

where $H_r(\cdot)$ is the r-th bit of the binary codes; $\mathbb{1}\{A\}$ is an indicator function that returns 1 if A is satisfied and otherwise returns 0. With $\mathcal{D}\left(H(\mathbf{q}_i), H(\mathbf{k}_j)\right)$, the codes inner product between $H(\mathbf{q}_i)$ and $H(\mathbf{k}_j)$ can be formulated as

$$\begin{aligned} H(\mathbf{q}_i)^\top H(\mathbf{k}_j) &= \sum_{r=1}^{b} \mathbb{1}\{H_r(\mathbf{q}_i) = H_r(\mathbf{k}_j)\} - \sum_{r=1}^{b} \mathbb{1}\{H_r(\mathbf{q}_i) \neq H_r(\mathbf{k}_j)\} \\ &= b - 2\sum_{r=1}^{b} \mathbb{1}\{H_r(\mathbf{q}_i) \neq H_r(\mathbf{k}_j)\} = b - 2\mathcal{D}\left(H(\mathbf{q}_i), H(\mathbf{k}_j)\right). \end{aligned} \tag{7}$$

Importantly, Eq. (7) shows the equivalence between the Hamming distance and the codes inner product since there is a one-to-one correspondence. By substituting with the hashed queries and keys in Eq. (4a), we can approximate the self-attention as

$$\mathrm{Attention}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) \approx \sum_i \frac{H(\mathbf{q}_t)^\top H(\mathbf{k}_i)\mathbf{v}_i^\top}{\sum_j H(\mathbf{q}_t)^\top H(\mathbf{k}_j)}. \tag{8}$$

Note that $H(\mathbf{q}_t)^\top H(\mathbf{k}_j) \in [-b, b]$. To avoid zero in denominator, we introduce a bias term 2^c to each inner product so that $H(\mathbf{q}_t)^\top H(\mathbf{k}_j) + 2^c > 0$, having no effect on the similarity measure. Here, we can simply set c to $\lceil \log_2(b+1) \rceil$ where $\lceil u \rceil$ returns the least integer greater than or equal to u. Using the associative property of matrix multiplication, we approximate the self-attention as

$$\begin{aligned} \mathrm{Attention}(\mathbf{q}_t, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) &\approx \sum_i \frac{\left(H(\mathbf{q}_t)^\top H(\mathbf{k}_i) + 2^c\right)\mathbf{v}_i^\top}{\sum_j \left(H(\mathbf{q}_t)^\top H(\mathbf{k}_j) + 2^c\right)} \\ &= \frac{H(\mathbf{q}_t)^\top \sum_i H(\mathbf{k}_i) \otimes \mathbf{v}_i + \sum_i 2^c \mathbf{v}_i^\top}{H(\mathbf{q}_t)^\top \sum_j H(\mathbf{k}_j) + 2^c N}. \end{aligned} \tag{9}$$

In practice, the multiplications between the binary codes and the full-precision values in Eq. (9) can be replaced by simple additions and subtractions, which greatly reduce the computational overhead in terms of on-chip energy footprint. Moreover, the multiplications with a powers-of-two 2^c can also be implemented by efficient *bit-shift* operations. As a result, the only multiplications come from the element-wise divisions between the numerator and denominator.

4.2 Self-supervised Hash Function Learning

Given queries $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\} \subset \mathbb{R}^{D_p}$, we seek to learn a group of hash functions $h : \mathbb{R}^{D_p} \mapsto \{1, -1\}$. Instead of explicitly applying the transformation function $\phi(\cdot)$ mentioned in Section 3.2, we compute the hash functions with a kernel function $\kappa(\mathbf{q}_i, \mathbf{q}_j) : \mathbb{R}^{D_p} \times \mathbb{R}^{D_p} \mapsto \mathbb{R}$. Given $\mathbf{Q} = [\mathbf{q}_1, \cdots, \mathbf{q}_N]^\top \in \mathbb{R}^{N \times D_p}$, we randomly sample m queries $\mathbf{q}_{(1)}, \ldots, \mathbf{q}_{(m)}$ from $\mathcal{Q}$ as support samples following the kernel-based supervised hashing (KSH) [37] and define a hash function h as

$$h(\mathbf{Q}) = \operatorname{sign}\left(\sum_{j=1}^{m} \left(\kappa\left(\mathbf{q}_{(j)}, \mathbf{Q}\right) - \mu_j\right) a_j\right) = \operatorname{sign}\left(\mathbf{g}(\mathbf{Q})\mathbf{a}\right), \tag{10}$$

where $\mathbf{a} = [a_1, \cdots, a_m]^\top$ is the weight of h, $\mu_j = \frac{1}{n}\sum_{i=1}^{N} \kappa\left(\mathbf{q}_{(j)}, \mathbf{q}_i\right)$ is to normalize the kernel function to zero-mean, and $\mathbf{g} : \mathbb{R}^{D_p} \mapsto \mathbb{R}^m$ is a mapping defined by $\mathbf{g}(\mathbf{Q}) = \left[\kappa\left(\mathbf{q}_{(1)}, \mathbf{Q}\right) - \mu_1, \ldots, \kappa(\mathbf{q}_{(m)}, \mathbf{Q}) - \mu_m\right] \in \mathbb{R}^{N \times m}$. Then, we define the kernelized hash function $H(\cdot)$ as

$$H(\mathbf{Q}) = [h_1(\mathbf{Q}), \cdots, h_b(\mathbf{Q})] = \begin{bmatrix} h_1(\mathbf{q}_1), \cdots, h_b(\mathbf{q}_1) \\ \cdots\cdots \\ h_1(\mathbf{q}_N), \cdots, h_b(\mathbf{q}_N) \end{bmatrix} = \operatorname{sign}\left(\mathbf{g}(\mathbf{Q})\mathbf{A}\right), \tag{11}$$

where $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_b] \in \mathbb{R}^{m \times b}$, and $h_r(\mathbf{Q}) = \operatorname{sign}\left(\mathbf{g}(\mathbf{Q})\mathbf{a}_r\right)$ is the hash function for the r-th bit.

To guide the learning of the binary codes, we hope that similar token pairs will have the minimal Hamming distance while dissimilar token pairs will have the maximal distance. Nevertheless, directly optimizing the Hamming distance is difficult due to the non-convex and non-smooth formulation in Eq. (6). Utilizing the equivalence between the code inner products and the Hamming distances in Eq. (7), we instead conduct optimization based on the Hamming affinity to minimize the reconstruction error as

$$\min_{\mathbf{A}} \left\|H(\mathbf{Q})H(\mathbf{Q})^\top - b\mathbf{Y}\right\|_F^2 = \min_{\mathbf{A}} \left\|\sum_{r=1}^{b} h_r(\mathbf{Q})h_r(\mathbf{Q})^\top - b\mathbf{Y}\right\|_F^2, \tag{12}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathbf{Y} \in \mathbb{R}^{N \times N}$ is the target Hamming affinity matrix. To preserve the similarity relations between queries and keys, we use the attention scores as the *self-supervised* information to construct $\mathbf{Y}$. Let $\mathcal{S}$ and $\mathcal{U}$ be the similar and dissimilar pairs of tokens. We obtain $\mathcal{S}$ and $\mathcal{U}$ by selecting the token pairs with the Top-l largest and smallest attention scores. We then construct pairwise labels $\mathbf{Y}$ as

$$\mathbf{Y}_{ij} = \begin{cases} 1, & (\mathbf{q}_i, \mathbf{q}_j) \in \mathcal{S} \\ -1, & (\mathbf{q}_i, \mathbf{q}_j) \in \mathcal{U} \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

However, Problem (12) is NP-hard. To solve it efficiently, we adapt discrete cyclic coordinate descent to learn binary codes sequentially. Specifically, we only solve $\mathbf{a}_r$ once the previous $\mathbf{a}_1, \cdots, \mathbf{a}_{r-1}$ have been optimized. Let $\hat{\mathbf{Y}}_{r-1} = b\mathbf{Y} - \sum_{t=1}^{r-1} h_t(\mathbf{Q})h_t(\mathbf{Q})^\top$ be the residual matrix, where $\hat{\mathbf{Y}}_0 = b\mathbf{Y}$. Then, we can minimize the following objective to obtain $\mathbf{a}_r$

$$\min_{\mathbf{a}_r} \quad \left\|h_r(\mathbf{Q})h_r(\mathbf{Q})^\top - \hat{\mathbf{Y}}_{r-1}\right\|_F^2 = \min_{\mathbf{a}_r} -2h_r(\mathbf{Q})^\top \hat{\mathbf{Y}}_{r-1} h_r(\mathbf{Q}) + C, \tag{14}$$

where $C = \left(h_r(\mathbf{Q})^\top h_r(\mathbf{Q})\right)^2 + \operatorname{tr}\left(\hat{\mathbf{Y}}_{r-1}\right)$ is a constant. Note that $\hat{\mathbf{Y}}_{r-1}$ is a symmetric matrix. Therefore, Problem (14) is a standard binary quadratic programming problem, which can be efficiently solved by many existing methods, such as the LBFGS-B solver [70] and block graph cuts [34]. To learn $\mathbf{a}_r$ in conjunction with network parameters, we propose to solve Problem (14) using the gradient-based methods. For the non-differentiable sign function, we use STE [5] to approximate the gradient using hard $\tanh$ as mentioned in Section 3.3. Note that learning the hash functions for each epoch is computationally expensive yet unnecessary. We only learn the hash functions per τ epoch.

5 Experiments

5.1 Comparisons on ImageNet-1K

Table 2: Main results on ImageNet-1K. The number of multiplications, additions, as well as energy consumption are calculated based on an image of resolution 224×224. The throughput is measured with a mini-batch size of 32 on a single NVIDIA RTX 3090 GPU.

Model	Method	#Mul. (B)	#Add. (B)	Energy (B pJ)	Throughput (images/s)	Top-1 Acc. (%)
PVTv2-B0 [63]	MSA	2.02	1.99	9.3	850	70.77
	Ours	**0.54**	**0.56**	**2.5**	**1379**	**70.44**
PVTv2-B1	MSA	5.02	5.00	23.1	621	78.83
	Ours	**2.03**	**2.09**	**9.4**	**874**	**78.38**
PVTv2-B2	MSA	8.64	8.60	39.7	404	81.82
	Ours	**3.85**	**3.97**	**17.8**	**483**	**81.28**
Twins-SVT-S [10]	MSA	5.96	5.91	27.4	426	81.66
	Ours	**2.72**	**2.81**	**12.6**	**576**	**80.22**

To investigate the effectiveness of the proposed method, we conduct experiments on ImageNet-1K [31], a large-scale image classification dataset that contains $\sim$1.2M training images from 1K categories and 50K validation images. We compare our kernelized hashing attention with standard MSA by adapting the two attention approaches into two popular vision Transformer frameworks PVTv2 [63] and Twins [10]. We measure model performance by the Top-1 accuracy. Furthermore, as FLOPs cannot accurately reflect the computational cost in our proposed method, we measure the model complexity by the number of multiplications and additions, separately, as done in [54]. Moreover, we report the energy consumption according to Table 1 and the throughput with a mini-batch size of 32 on a single NVIDIA RTX 3090 GPU.

Implementation details. All training images are resized to 256×256, and 224×224 patches are randomly cropped from an image or its horizontal flip, with the per-pixel mean subtracted. To obtain the MSA baselines, we first replace the original attention layers in PVTv2 [63] and Twins [10] with standard MSAs and initialize the models with the pretrained weights on ImageNet-1K. Next, we finetune each model on ImageNet-1K with 100 epochs. Based the pretrained MSA weights, we then apply our approach to each model and finetune on ImageNet-1K with 30 epochs. All models in this experiment are trained on 8 V100 GPUs with a total batch size of 256. The initial learning rate is set to 2.5×10^{-5}. We use AdamW optimizer [41] with a cosine decay learning rate scheduler. All other hyperparameters are the same as in PVTv2. Also note that recent hierarchical ViTs [63, 39, 10] have multiple stages to incorporate pyramid feature maps. At the last stage, they usually apply standard MSAs due to the comparably low-resolution feature maps. This design is also adopted in PVTv2 and Twins. Therefore, we follow the common practice and do not modify the attention layers at the last stage. For the hash functions learning, we set the number of support samples m and update interval τ to 25 and 30, respectively. The hyper-parameter l in constructing pairwise labels $\mathbf{Y}$ is set to 10. We set the hash bit b to 16.

Results analysis. We report the results in Table 2. In general, compared to MSA, our method achieves lower computational complexity, less energy consumption and higher throughput with comparable performance. For example, based on PVTv2-B0, our method saves around 73% multiplications and 72% additions, as well as reducing 73% energy consumption, which demonstrates the energy-efficiency of our approach. With more efficient accumulation implementation, the throughput of our EcoFormer can be further improved. Besides, note that a larger model comes with more multiplications and additions, especially in other types of layers (*e.g.*, feed-forward layers, as shown in the supplementary material). In this case, as our approach focuses on the attention layers, the energy-saving from larger models is comparably less than smaller models (*e.g.*, PVTv2-B0 vs. PVTv2-B2). Even though, we still reduce significant computational cost and energy consumption. For example, on PVTv2-B2, we save around 55% energy consumption. Compared with PVTv2, the performance drop of our method is slightly larger on Twins. One possible reason is that our method may be sensitive to the conditional positional encodings in Twins.

5.2 Comparisons on Long Range Arena

To evaluate the performance of different efficient attentions under long-context scenarios, we train our EcoFormer on two tasks, **Text** and **Retrieval** from the Long Range Arena (LRA) benchmark [56]

following the settings of [71]. Our implementations are based on the released code of [65]. We use the same hyper-parameters m, τ and b as in ImageNet-1K experiments. We show the results in Table 3. From the table, our EcoFormer achieves comparable performance with much lower energy consumption. For example, on **Text**, compared with MSA, our method saves around 94.6% multiplications and 93.7% additions as well as 94.5% energy consumption, which is more efficient than existing attention mechanisms.

Table 3: Comparisons of different methods on Long Range Arena (LRA). We report the classification accuracy (%) for **Text** as well as **Retrieval** and average accuracy across two tasks. * denotes that we obtain the results from the original paper.

Method	#Mul. (B)	#Add. (B)	Energy (B pJ)	Text (4K)	Retrieval (4K)	Average
Transformer	4.63	4.57	21.25	64.87	79.62	72.25
Performer [9]	0.83	0.84	3.83	64.82	79.08	71.95
Linformer [61]	0.81	0.81	3.74	57.03	78.11	67.57
Reformer [30]	0.54	0.54	2.49	65.19	79.46	72.33
Combiner* [51]	0.51	0.51	2.34	64.36	56.10	60.23
EcoFormer	**0.25**	**0.29**	**1.17**	64.79	78.67	71.73

5.3 Ablation Study

In this section, we evaluate the effectiveness of our EcoFormer by comparing it with different binarization approaches and efficient attention mechanisms. By default, we train each model from scratch on CIFAR-100 with 2 GPUs. The total batch size is 64. The initial learning rate is 6.25×10^{-6}. For the hash functions learning, we set update interval τ to 300. All the other hyperparameters are the same as in ImageNet-1K experiments.

Table 4: Performance comparisons with different binarization methods on CIFAR-100.

Model	Method	#Mul. (B)	#Add. (B)	Energy (B pJ)	Top-1 Acc. (%)
PVTv2-B0	FP-EcoFormer	0.94	0.94	4.33	70.78
	Bi-EcoFormer	0.63	0.83	3.09	70.06
	EcoFormer	**0.54**	**0.56**	**2.49**	**71.23**
Twins-SVT-S	FP-EcoFormer	5.96	5.91	27.36	80.04
	Bi-EcoFormer	3.01	3.59	14.38	80.04
	EcoFormer	**2.72**	**2.81**	**12.58**	**80.31**

Quantization vs. hashing. To investigate the effect of different binarization methods, we compare our EcoFormer with the following methods: **FP-EcoFormer**: Based on EcoFormer, we do not binarize queries and keys in attentions. **Bi-EcoFormer**: Relying on EcoFormer, we use the same binary quantization [26] as BinaryBERT [2] and BiBERT [47] to obtain binarized queries and keys instead of our proposed hash functions. For fair comparisons, the attention operations in the compared method are in linear complexity. We apply different methods to PVTv2-B0 and Twins-SVT-S and report the results in Table 4. We observe that our EcoFormer consistently outperforms Bi-EcoFormer on different frameworks. For example, based on PVTv2-B0, our EcoFormer surpasses Bi-EcoFormer by 1.17% in terms of the Top-1 accuracy. Compared with binary quantization, our proposed self-supervised hash functions preserve the pairwise similarity of attention, leading to better performance. Moreover, our EcoFormer does not need to explicitly compute transformation $\phi(\cdot)$ as in Eq. (4b). Therefore, the energy cost of our EcoFormer is lower than Bi-EcoFormer.

Table 5: Performance comparisons with different hash functions regarding PVTv2-B0 on CIFAR-100.

Method	#Mul. (B)	#Add. (B)	Energy (B pJ)	Top-1 Acc. (%)
LSH-EcoFormer	0.68	0.69	3.12	70.18
KLSH-EcoFormer	0.54	0.56	2.49	70.66
EcoFormer	**0.54**	**0.56**	**2.49**	**71.23**

Effect of different hash functions. To investigate the effect of different hash functions, we include the following methods for comparisons: **LSH-EcoFormer**: Relying on EcoFormer, we use Locality-Sensitive Hashing (LSH) [13] rather than our proposed kernelized hash function. **KLSH-EcoFormer**: Based on EcoFormer, we replace the proposed hash function with Kernelized Locality-Sensitive Hashing (KLSH) [32]. We report the results in Table 5. We can observe that KLSH-EcoFormer outperforms LSH-EcoFormer by 0.48% in terms of the Top-1 accuracy with less energy consumption. The reason can be attributed to that LSH is based on random linear projection, which can not deal with

the non-linear softmax attention well. Critically, our EcoFormer further improves the performance by 0.57% on the Top-1 accuracy. Compared with random hashing in KLSH, our EcoFormer learns the hash functions with additional self-supervised information. Therefore, our learned binary codes are better at preserving the token similarity.

Table 6: Comparison with other efficient attention methods regarding PVTv2-B0 [63] on CIFAR-100.

Method	#Mul. (B)	#Add. (B)	Energy (B pJ)	Top-1 Acc. (%)
Transformer	2.02	1.99	9.25	71.44
Performer [9]	0.94	0.94	4.33	70.78
Linformer [61]	0.69	0.69	3.18	71.17
Reformer [30]	1.62	1.63	7.44	70.56
EcoFormer	**0.54**	**0.56**	**2.49**	**71.23**

Comparing with other efficient attention mechanisms. To compare Ecoformer with different attention mechanisms, we conduct experiments on CIFAR-100 based on PVTv2-B0 in Table 6. In our experiments, we directly replace the attention layers with each compared method in PVTv2-B0 [63]. In general, compared to other efficient attention mechanisms, EcoFormer saves more computations and reduces more energy consumption while achieving better performance. Particularly, benefiting from the multiplication-saving operations and low-dimensional binary queries and keys, EcoFormer saves more energy than Performer. Also note that since the proposed kernelized hash function $H(\cdot)$ does not need to explicitly apply transformation $\phi(\cdot)$ to the queries and keys as in Eq. (4b), EcoFormer simultaneously reduces more multiplications and additions than Performer. Besides, Linformer achieves competitive results. However, as the size of the learnable low-rank projection parameters depends on the length of the input sequence, Linformer is not scalable to different image resolutions, whereas EcoFormer with sufficient bits is agnostic to the sequence length.

Latency and Energy on BitFusion [52]. To show the actual energy consumption and latency, we test different methods on a simulator of BitFusion, a bit-flexible microarchitecture synthesized in 45 nm technology. From Table 7, EcoFormer shows much lower latency and energy than the other efficient attention methods, which further verifies the advantage of EcoFormer.

Table 7: Latency and energy comparisons with different attention methods. We measure the latency and energy of an attention layer with a batch size of 16, a sequence length of 3,136 and an embedding dimension of 32 on a BitFusion [52] simulator.

Method	Latency (ms)	Energy (pJ)
Transformer	0.0036	85,692.18
Performer [9]	0.0019	41,113.64
Linformer [61]	0.0018	45,770.61
Reformer [30]	0.0024	57,305.47
EcoFormer	**0.0010**	**24,990.75**

6 Conclusion and Future Work

In this paper, we have presented a novel energy-saving attention mechanism with linear complexity to save the vast majority of multiplications from a new binarization perspective, making the deployment of Transformer models at scale feasible on edge devices. We are inspired by the fact that conventional binarization methods are built upon statistical quantization error minimization without considering to preserve the pairwise similarity relations between tokens. To this end, we customize binarization to softmax attention by mapping the original token features into compact binary codes in Hamming space using a set of kernel-based hash functions, where the similarity can be measured by codes dot product. The hash functions for queries/keys are learned to encourage the Hamming affinity of a token pair to be close to the target obtained from the attention scores, in a self-supervised way. Extensive experiments have demonstrated that EcoFormer saves significant energy footprint while achieving comparable performance with standard attentions on ImageNet-1K, Long Range Arena and CIFAR-100. In terms of the future work, we can further binarize the value vectors in attention, multi-layer perceptrons and non-linearities in Transformer to make it fully binarized for more significant energy-saving. We may also extend EcoFormer to other NLP tasks such as machine translation and speech analysis tasks to make it more impactful to wider communities.

Limitations and societal impact. We have shown that EcoFormer is more energy-efficient compared to the standard attention. However, in practice, the addition operations between binary codes and floating-point numbers will require specialized GPU kernels (*e.g.*, customized CUDA operators) for further acceleration. Our work potentially brings some negative societal impacts that training large Transformer models requires extensive computations, resulting in financial and environmental costs. A promising solution is to jointly optimize training and inference efficiency.

References

[1] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *NeurIPS*, volume 28, 2015.

[2] H. Bai, W. Zhang, L. Hou, L. Shang, J. Jin, X. Jiang, Q. Liu, M. Lyu, and I. King. Binarybert: Pushing the limit of bert quantization. In *ACL*, pages 4334–4348, 2021.

[3] Y. Bai, Y.-X. Wang, and E. Liberty. Proxquant: Quantized neural networks via proximal operators. In *ICLR*, 2019.

[4] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[5] Y. Bengio. Estimating or propagating gradients through stochastic neurons. *arXiv preprint arXiv:1305.2982*, 2013.

[6] A. Bulat and G. Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

[7] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[8] H. Chen, Y. Wang, C. Xu, B. Shi, C. Xu, Q. Tian, and C. Xu. Addernet: Do we really need multiplications in deep learning? In *CVPR*, pages 1468–1477, 2020.

[9] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *ICLR*, 2021.

[10] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021.

[11] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, volume 28, 2015.

[12] G. Daras, N. Kitaev, A. Odena, and A. G. Dimakis. Smyrf-efficient attention using asymmetric clustering. *NeurIPS*, 33:6476–6489, 2020.

[13] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG*, pages 253–262, 2004.

[14] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser. Universal transformers. In *ICLR*, 2019.

[15] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics, 2019.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[17] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu. Regularizing activation distribution for training binarized deep networks. In *CVPR*, pages 11408–11417, 2019.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[19] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha. Learned step size quantization. In *ICLR*, 2020.

[20] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[21] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *TPAMI*, 35(12):2916–2929, 2012.

[22] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016.

[23] M. Horowitz. 1.1 computing's energy problem (and what we can do about it). In *ISSCC*, pages 10–14. IEEE, 2014.

[24] L. Hou and J. T. Kwok. Loss-aware weight quantization of deep networks. In *ICLR*, 2018.

[25] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *NeurIPS*, pages 4107–4115, 2016.

[26] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR*, 18(1):6869–6898, 2017.

[27] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *ICML*, pages 4651–4664. PMLR, 2021.

[28] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *CVPR*, pages 4350–4359, 2019.

[29] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pages 5156–5165, 2020.

[30] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, volume 25, 2012.

[32] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *TPAMI*, 34(6):1092–1104, 2011.

[33] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, and X. Ji. High-performance fpga-based cnn accelerator with block-floating-point arithmetic. *TVLSI*, 27(8):1874–1885, 2019.

[34] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, pages 1963–1970, 2014.

[35] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *NeurIPS*, pages 344–352, 2017.

[36] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NeurIPS*, volume 27, 2014.

[37] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[38] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, 2011.

[39] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, pages 10012–10022, 2021.

[40] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *ECCV*, pages 722–737, 2018.

[41] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[42] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang. Soft: Softmax-free transformer with linear complexity. In *NeurIPS*, volume 34, 2021.

[43] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*, 2020.

[44] A. Mishra and D. Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*, 2018.

[45] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[46] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. Smith, and L. Kong. Random feature attention. In *ICLR*, 2020.

[47] H. Qin, Y. Ding, M. Zhang, Q. YAN, A. Liu, Q. Dang, Z. Liu, and X. Liu. BiBERT: Accurate fully binarized BERT. In *ICLR*, 2022.

[48] Z. Qin, W. Sun, H. Deng, D. Li, Y. Wei, B. Lv, J. Yan, L. Kong, and Y. Zhong. cosformer: Rethinking softmax in attention. In *ICLR*, 2022.

[49] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NeurIPS*, volume 20, 2007.

[50] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542, 2016.

[51] H. Ren, H. Dai, Z. Dai, M. Yang, J. Leskovec, D. Schuurmans, and B. Dai. Combiner: Full attention transformer with sparse computation cost. *NeurIPS*, 34:22470–22482, 2021.

[52] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh. Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network. In *ISCA*, pages 764–775. IEEE, 2018.

[53] S. Shen, Z. Dong, J. Ye, L. Ma, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *AAAI*, volume 34, pages 8815–8821, 2020.

[54] H. Shu, J. Wang, H. Chen, L. Li, Y. Yang, and Y. Wang. Adder attention for vision transformer. In *NeurIPS*, volume 34, 2021.

[55] Z. Sun, Y. Yang, and S. Yoo. Sparse attention with learning to hash. In *ICLR*, 2021.

[56] Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena: A benchmark for efficient transformers. In *ICLR*, 2020.

[57] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, volume 30, 2017.

[59] A. Vyas, A. Katharopoulos, and F. Fleuret. Fast transformers with clustered attention. *NeurIPS*, 33:21665–21674, 2020.

[60] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *TPAMI*, 34(12):2393–2406, 2012.

[61] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[62] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, pages 568–578, 2021.

[63] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pvtv2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10, 2022.

[64] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NeurIPS*, volume 21, 2008.

[65] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *AAAI*, volume 35, page 14138, 2021.

[66] C. Yan, B. Gong, Y. Wei, and Y. Gao. Deep multi-view enhancement hashing for image retrieval. *TPAMI*, 43(4):1445–1451, 2020.

[67] A. H. Zadeh, I. Edo, O. M. Awad, and A. Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *MICRO*, pages 811–824, 2020.

[68] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat. Q8bert: Quantized 8bit bert. In *EMC2-NIPS*, pages 36–39. IEEE, 2019.

[69] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[70] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

[71] C. Zhu, W. Ping, C. Xiao, M. Shoeybi, T. Goldstein, A. Anandkumar, and B. Catanzaro. Long-short transformer: Efficient transformers for language and vision. *NeurIPS*, 34:17723–17736, 2021.

[72] B. Zhuang, G. Lin, C. Shen, and I. Reid. Fast training of triplet-based deep binary embedding networks. In *CVPR*, pages 5955–5964, 2016.

[73] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid. Structured binary neural network for accurate image classification and semantic segmentation. In *CVPR*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See the abstract and introduction.
   (b) Did you describe the limitations of your work? [Yes] See the conclusion in Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See the conclusion in Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We have included the code and instructions needed in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See the implementation details in Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Error bars are not reported because it would be too computationally expensive. We use the same random seed as in recent works for fair comparison.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See the implementation details in Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code is included in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use public datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]