

SeTformer Is What You Need for Vision and Language

Pourya Shamsolmoali¹, Masoumeh Zareapoor^{2*}, Eric Granger³, Michael Felsberg⁴

¹ School of Communication and Electronic Eng., East China Normal University

² Department of Electrical Eng., Shanghai Jiaotong University

³ LIVIA, Department of Systems Eng., ETS Montreal

⁴ Computer Vision Laboratory, Linköping University
mzarea@ieee.org

Abstract

The dot product self-attention (DPSA) is a fundamental component of transformers. However, scaling them to long sequences, like documents or high-resolution images, becomes prohibitively expensive due to the quadratic time and memory complexities arising from the softmax operation. Kernel methods are employed to simplify computations by approximating softmax but often lead to performance drops compared to softmax attention. We propose SeTformer, a novel transformer where DPSA is purely replaced by Self-optimal Transport (SeT) for achieving better performance and computational efficiency. SeT is based on two essential softmax properties: maintaining a non-negative attention matrix and using a nonlinear reweighting mechanism to emphasize important tokens in input sequences. By introducing a kernel cost function for optimal transport, SeTformer effectively satisfies these properties. In particular, with small and base-sized models, SeTformer achieves impressive top-1 accuracies of 84.7% and 86.2% on ImageNet-1K. In object detection, SeTformer-base outperforms the FocalNet counterpart by +2.2 mAP, using 38% fewer parameters and 29% fewer FLOPs. In semantic segmentation, our base-size model surpasses NAT by +3.5 mIoU with 33% fewer parameters. SeTformer also achieves state-of-the-art results in language modeling on the GLUE benchmark. These findings highlight SeTformer applicability for vision and language tasks.

Introduction

Transformers (Vaswani et al. 2017), initially introduced for natural language processing (NLP), have gained significant popularity in computer vision following the groundbreaking work of Vision Transformer (ViT) (Dosovitskiy et al. 2021). Its promise has been demonstrated in various vision tasks, including image classification, object detection, segmentation and beyond (Khan et al. 2022). Dot-product self-attention (DPSA) with softmax normalization plays a crucial role in transformers for capturing long-range dependencies. However, the computation of this model leads to quadratic time and memory complexities, making it challenging to train long sequence models. Therefore, research on “efficient” transformers has gained significant importance in the last two years. Several methods with the aim

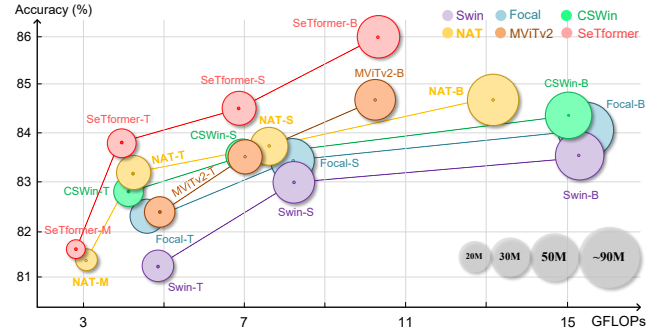


Figure 1: Top-1 classification accuracy vs. FLOPs on the ImageNet-1k, where the bubble size represents the number of parameters. SeTformer improves upon the baseline accuracy yet require fewer parameters and FLOPs.

of increasing the efficiency of transformers (Touvron et al. 2021; Liu et al. 2021; Huang et al. 2022; Yang et al. 2022; Liu et al. 2022; Wang et al. 2022; Hassani et al. 2023; Grainger et al. 2023) are proposed. Although these methods have achieved impressive results on ImageNet-1K (an area where ViT has struggled), the increased complexity makes these models slower overall (Figure 1). One main bottleneck in attention mechanisms is the softmax operator, which affects the efficiency (Zandieh et al. 2023). Thus, recent research has proposed kernel-based methods as a way to approximate the attention matrix and improve computational efficiency (Choromanski et al. 2021; Peng et al. 2021; Chowdhury et al. 2022; Choromanski et al. 2023; Zandieh et al. 2023; Reid et al. 2023). The approximate attention matrix is an unbiased estimate of the original attention matrix, encoding token similarities via a softmax-kernel without explicit construction. However, the improved efficiency achieved through kernelization often comes with additional, sometimes impractical assumptions on the attention matrix (Wang et al. 2020) or with approximations of the softmax operation under specific theoretical conditions (Choromanski et al. 2023). However, when these assumptions are not satisfied or approximation errors occur, the performance of these methods may not be consistently better than the vanilla transformer. For example, transformer variants with linear complexity, such as ScatterBrain (Chen et al. 2021), and

*Work done at Inria, CNRS-ENS.

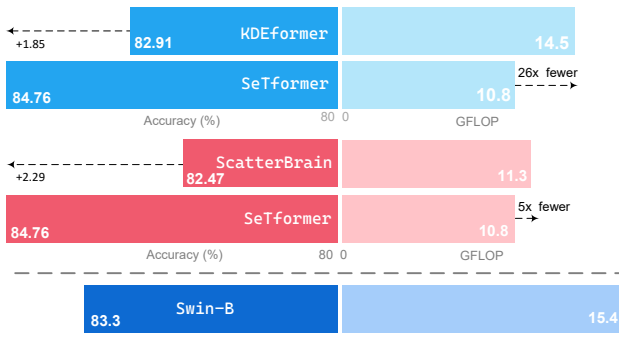


Figure 2: Top-1 Classification accuracy vs. FLOPs on the ImageNet-1k. We compare SeTformer with other kernel-based transformers while using Swin-B as the base.

KDEformer (Zandieh et al. 2023), show less satisfactory performance on ImageNet-1k compared to Swin (Liu et al. 2021), as observed in Figure 2. Here, we emphasize that while softmax is computationally expensive, but it is an accurate function for calculating attention weights. Given this challenge, a fundamental question arises: “*Is there an alternative to DPSA that maintains softmax’s properties while efficiently modeling long-range interactions?*”. In softmax attention, two factors play a vital role in its performance: non-negative elements within the attention matrix (Zandieh et al. 2023), and a nonlinear reweighting mechanism that enhances the stability of attention weights (Gao and Pavel 2017; Zhu et al. 2021). We propose a new variant of DPSA called Self-optimal Transport (SeT) that satisfies both of these properties by using principles from optimal transport (OT) and the kernel method. In SeT, input features are mapped into a Reproducing Kernel Hilbert Space (RKHS), in which point evaluation takes the form of a linear function (Bietti and Mairal 2019). Instead of using the dot-product to find the similarity score between elements, we use alignment scores achieved through the computation of OT between input and reference features using Sinkhorn’s method (Cuturi 2013). OT, is an effective computational technique for aligning distributions, that has found significant attention in many applications, including computer vision (De Plaen et al. 2023), and machine learning (Liu et al. 2023). As the name indicates, OT aims to find the most efficient way for transporting a mass from one location to another, while in our model, it efficiently computes similarity weights by aligning each feature in the input with a reference set.

Contributions. SeT involves a two-step process: constructing the kernel feature map and then applying OT to compute the alignment matrix. We begin by embedding input feature vectors into a RKHS, where a positive definite kernel ensures the non-negative property. This enables the model to avoid aggregating negatively-correlated information. The non-linear reweighting scheme is achieved by assigning different weights to tokens based on their relevance with the reference set. OT aligns input feature vectors with a reference set, enhancing local correlations and capturing complex dependencies. To maintain computational feasibility,

we use a kernel approach (Zhang, Wang, and Nehorai 2019) to obtain a finite-dimensional embedding, and multi-level hierarchical representations, similar to Swin (Liu et al. 2021), are constructed through kernel compositions, resulting in downsampled feature maps across levels. Our new transformer, SeTformer, demonstrates its effectiveness through extensive experiments on various tasks. This highlights the potential of content-based interactions to enhance transformer performance in vision and language applications.

Related Works

Vision Transformers. Transformer and self-attention mechanism have significantly impacted Natural Language Processing (Vaswani et al. 2017), and their application to vision tasks has been made possible by the pioneering Vision Transformer (ViT) (Dosovitskiy et al. 2021). Researchers have further extended ViT models in multiple directions, focusing on position encoding (Chu et al. 2023), data efficiency (Touvron et al. 2021), and optimization techniques (Li et al. 2022b). These advancements have led to significant progress in vision tasks (Khan et al. 2022). Several recent works have focused on enhancing ViT performance on downstream tasks by exploring pyramid structures, surpassing convolution-based methods. PVT (Wang et al. 2021, 2022) introduces sparse location sampling in the feature map to form key and value pairs. Swin (Liu et al. 2021) utilizes non-overlapping windows with window shifts between consecutive blocks. CSwin (Dong et al. 2022) extends this approach with cross-shape windows to enhance model capacity. PaCa-ViT (Grainger et al. 2023) introduces a new approach where queries start with patches, while keys and values are based on clustering, learned end-to-end. HaloNet (Vaswani et al. 2021) introduced a haloing mechanism that localizes self-attention for blocks of pixels instead of pixel-wise, aiming to address the lack of an efficient sliding window attention. Similarly, NAT (Hassani et al. 2023) adopts neighborhood attention, considering specific scenarios for corner pixels. FocalNets (Yang et al. 2022) replaces self-attention with a Focal module using gated aggregation technique for token interaction modeling in vision tasks. While these advances have been successful and achieved impressive results on vision tasks, they often come with higher complexity compared to vanilla ViT counterparts. With SeTformer, we’ve aimed to achieve a balance between performance and complexity, addressing this issue.

Kernel Methods for Transformers. When dealing with large-scale input, a more efficient approach is to directly reduce the complexity of the theoretical calculations. Kernelization accelerates self-attention by transforming the computation complexity from quadratic to linear. By utilizing kernel feature maps, we can bypass the computation of the full attention matrix, which is a major bottleneck in softmax. Recent advances in scalable transformers include (Choromanski et al. 2021; Peng et al. 2021; Chowdhury et al. 2022; Choromanski et al. 2023; Zandieh et al. 2023), where the self-attention matrix is approximated as a low-rank matrix for long sequences. These methods are either very simplistic (Chowdhury et al. 2022), while others are mathemati-

cally well explained but complex (Choromanski et al. 2021; Zandieh et al. 2023). Several of these methods rely on the Fourier transform, leading to sin and cos random features, which are unsuitable for transformers due to negative values in the attention matrix. To address this, Choromanski et al. (2021) proposed a solution using positive valued random features known as FAVOR+ for self-attention approximation. This approach was improved by Likhoshesterov et al. (2022), through carefully selecting linear combination parameters, allowing for one parameter set to be used across all approximated values. While all these methods achieved decent efficiency, they often falls behind popular vision transformers like Swin in terms of performance (Figure 2). However, we explore the concept of optimal transport and kernel learning to potentially overcome this performance gap while still maintaining the efficiency advantages of Transformers.

Our model’s corresponding kernel is a matching kernel (Tolias, Avrithis, and Jégou 2013), which uses a similarity function to compare pairs of feature (the input vector and reference set). Recent methods have also explored kernels based on matching features using wasserstein distance (Khamis et al. 2023; Kolouri et al. 2021). Prior studies by Skianis et al. (2020), Mialon et al. (2021) and Lee et al. (2019) analyze similarity costs between input and reference features in biological data. They employ dot-product operation (Vaswani et al. 2017) for element-wise comparison. In contrast, our model employs transport plans to compute attention weights, providing a new perspective on Transformer’s attention via kernel methods. Unlike other kernel-based methods like Performer that is not compatible with positional encoding techniques, our model incorporates this information, which are crucial in visual modeling tasks.

Proposed Method

From Self-Attention to Self-optimal Transport

Consider an input sequence $x=\{x_1, \dots, x_n\} \in R^d$ of n tokens. The DPSA is a mapping that inputs matrices $Q, K, V \in R^{n \times d}$. These matrices are interpreted as queries, keys, and values, respectively,

$$\text{Att}(Q, K, V) = D^{-1}AV \quad (1)$$

$$A = \exp\left(QK^T/\sqrt{d}\right); \quad D = \text{diag}(A\mathbf{1}_n^T)$$

where $\mathbf{1}_n$ is the ones vector, and $A, D \in R^{n \times n}$. The Softmax operation normalizes the attention weights A , allowing each token to be a weighted average of all token values. However, the quadratic complexity of the softmax becomes a bottleneck as the number of tokens increases. We aim to develop a powerful and efficient self-attention that is, above all, simple. We do not add any complex modules like convolution (Wu et al. 2021), shifted windows (Hassani et al. 2023), or attention bias (Li et al. 2022a) to improve the vision task’s performance. We indeed take a different strategy. SeT leverages the important properties of softmax, including non-negativity and reweighting mechanism (Gao and Pavel 2017), while also prioritizing efficiency in its design. The use of RKHS with a positive definite (PD) kernel avoids aggregating negative-correlated information. SeT incorporates

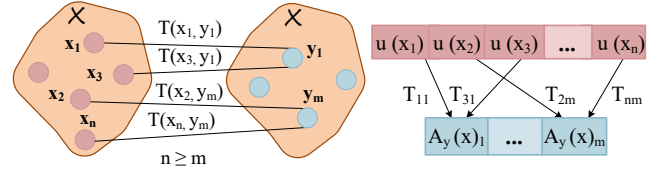


Figure 3: An input feature vector x is transported onto reference y via transport plan $T(x, y)$, that aggregates x features w.r.t. y , yielding $A_y(x)$. In DPSA, each x_i aggregates with all x features, forming a large sparse matrix. Our model aggregates based on best-matched x and y features through OT.

a nonlinear reweighting scheme through OT. This involves computing alignment scores between input and reference sets within RKHS. This process introduces nonlinearity to the alignment scores, assigning weights to elements to highlight their significance. This helps the model in capturing complex relationships and emphasizing local correlations.

Representing Local Image Neighborhoods in an RKHS.

In order to maintain a linear computation, we embed the input feature vector into a RKHS, in which point evaluation takes the form of linear function. Kernel methods (Schölkopf et al. 2002) enable us to map data from its original space \mathcal{X} to a higher-dimensional Hilbert space (feature space) \mathcal{F} through a positive definite (PD) kernel \mathcal{K} . For a function $u: \mathcal{X} \rightarrow \mathcal{F}$ (feature map), the PD kernel is denoted as $\mathcal{K}(x, x') = \langle u(x), u(x') \rangle_{\mathcal{F}}$. Given that $u(x)$ can be an infinite-dimensional, the kernel technique allows to derive a finite-dimensional representation $v(x)$ in R^k , with an inner product $\langle v(x_i), v(x'_j) \rangle$ denoting $\mathcal{K}(x, x')$. As shown by (Fukumizu 2008), if \mathcal{K} is positive definite, for any x, x' , we have $\mathcal{K}(x, x') \geq 0$, which aligns with the non-negativity property of softmax operator.

Optimal Transport (OT). A fundamental role in our model is to aggregate related tokens by learning a mapping between them. Our weighted aggregation relies on the transport plan between elements x and x' treated as distinct measures or weighted point clouds. OT has found extensive use in alignment problems (Shamsolmoali et al. 2024), and has an impressive capacity to capture the geometry of the data (Liu et al. 2023). We focus throughout this paper on the Kantorovich form of OT (Peyré and Cuturi 2019) with entropic regularization for smoothing transportation plans. Let g in μ_n and h in $\mu_{n'}$ denote the weights of discrete measures $\sum_i g_i \delta_{x_i}$ and $\sum_j h_j \delta_{x'_j}$ for the elements x and x' . The cost matrix $C \in R^{n \times n'}$ has entries $c(x_i, x'_j)$ for the (i, j) pairs. In our model, the weights g and h are determined by the transport plan that aligns the input set with the reference sets. Instead of computing a pairwise dot product between distributions, OT finds the minimal effort based on the ground cost to shift the mass from one distribution to another,

$$OT_{\text{kant}} = \min_{T \in U(g, h)} \sum_{ij} C_{ij} T_{ij} + \epsilon H(T) \quad (2)$$

where the negative entropy function is defined as $H(T) = \sum_{i,j} T_{ij}(\log(T_{ij}) - 1)$ with regularization parameter ϵ . The

transport plan T_{ij} describes the amount of mass flowing from location i to location j with minimal cost, and the constraint $U(g, h) = \{T \in R_+^{n \times n'} : T1_n = g; T^T 1_{n'} = h\}$ represents the uniform transport polytope. The computation of OT in (2) is efficiently done using a matrix scaling procedure derived from Sinkhorn’s algorithm (Cuturi 2013). OT assigns different weights to individual elements/tokens based on their significance within the input, similar to the reweighting scheme in softmax attention. Furthermore, OT enforces non-negativity by optimizing alignments between input elements (Sinkhorn and Knopp 1967; Cuturi 2013), preserving the non-negative nature of attention weights, as in softmax attention. Indeed, kernels capture the nonlinear transformation of the input, while OT finds optimal alignments between sets of features with fast computation.

Self-optimal Transport (SeT)

With an input feature vector x and a reference y_m in \mathcal{X} , we perform the following steps: (i) representing the feature vectors x and y in RKHS \mathcal{F} , (ii) aligning the elements of x with y using OT, (iii) performing a weighted aggregation of the elements x into m clusters, resulting in an alignment matrix A , as detailed in Figure 3. We use a reference y for efficient element aggregation. Each element in the reference set serves as an “alignment cell”, and input features are aggregated within these cells through a weighted sum. These weights indicate the correspondence between the input and references, computed using OT. Suppose we have an input feature vectors $x = \{x_1, \dots, x_n\}$ living in $\mathcal{X} \in R^d$, randomly extracted from input images. In the context of the Nyström approximation (Xiong et al. 2021), the samples of y is the centroids obtained by conducting K-means clustering to feature vectors in the training set \mathcal{X} , such that we obtain $y = \{y_1, \dots, y_m\}$ with $m \leq n$. The use of the reference set helps optimize the computation process and enables the model to scale effectively for longer input sequences (Skianis et al. 2020; Mialon et al. 2021). Let k be a positive-definite kernel, like the Gaussian kernel that defined on RKHS along with mapping $u : R^d \rightarrow \mathcal{F}$. We create a matrix k of size $n \times m$ that stores the comparisons $k(x_i, y_j)$. Next, we compute the transport plan between x and y based on (2), resulting in the $n \times m$ matrix $T(x, y)$. The transport plan finds the best way to align the input features with the reference elements while minimizing the alignment cost. Our $A_y(x)$ is now defined as

$$m^{1/2} \left[\sum_{i=1}^n T(x, y)_{i1} u(x_i), \dots, \sum_{i=1}^n T(x, y)_{im} u(x_i) \right] = m^{1/2} T(x, y)^T u(x) \text{ where } u(x) = [u(x_1), \dots, u(x_n)]^T. \quad (3)$$

This aggregation relies on a positive-definite kernel such that

$$\mathcal{K}_y(x, x') = \sum_{i, i'=1}^n T_y(x, x')_{ii'} k(x_i, x'_{i'}) \quad (4)$$

This formulation with $T_y(x, x') = m \cdot T(x, y)T(x', y)^T$, can be seen as a kernel that assigns weights to match between pairs of elements. The weights are determined by OT in the feature space (\mathcal{F}). Our attention process involves aggregating the non-linearly embedded elements of x into

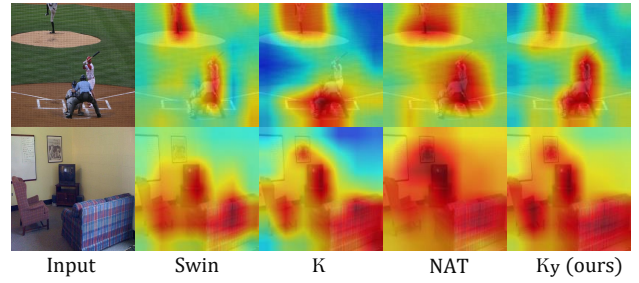


Figure 4: Visualization of our model on COCO dataset using \mathcal{K} , and \mathcal{K}_y . They are compared with the Swin (Liu et al. 2021) and NAT (Hassani et al. 2023). While \mathcal{K} focuses solely on large objects, \mathcal{K}_y accurately captures multi-scale objects without including sparse areas.

“cells”, with each cell corresponding to a reference element in y , guided by the transport plan $T(x, y)$. The reweighting property in $A_y(x)$ arises from using OT to align the non-linearly embedded input x with the reference y . The weights assigned during aggregation, driven by this alignment, are guided by the OT process. In other words, each value in $T_y(x, x')$ represents the weight or importance of x aligned with the elements of x' . Since the OT matrix is constructed from non-negative values (Cuturi 2013), the aggregation process will involve non-negative weights, satisfying the non-negativity property. However, if we exclude the use of a reference for cost calculation, we will have $\mathcal{K}(x, x') = \sum_{i, i'=1}^n T(x, x')_{ii'} k(x_i, x'_{i'})$, which is computationally costly due to the quadratic number of transport plans required. To differentiate between this kernel and \mathcal{K}_y , we need to find the relationship between $T_y(x, x')$ and $T(x, x')$. Consider x, x', y with lengths n, n' , and m respectively. The weighted wasserstein distance $\mathcal{W}_y(x, x')$ based on reference y is $\langle T_y(x, x'), d_k^2(x, x') \rangle^{1/2}$, where $d_k^2(x, x')$ is a distance metric induced by the kernel. The inequality $|\mathcal{W}(x, x') - \mathcal{W}_y(x, x')| \leq 2\min(\mathcal{W}(x, y), \mathcal{W}(x', y))$, shows that the weighted wasserstein distance $\mathcal{W}_y(x, x')$ is a valid approximation of the actual wasserstein distance $\mathcal{W}(x, x')$. The difference between these two distances is bounded by a factor of 2 times the minimum of the wasserstein distances between the input sets x and y ; x' and y . This efficient computation is highlighted in Figure 4 where \mathcal{K}_y effectively captures objects of different scales, emphasizing essential features of each object rather than the surrounding region. \mathcal{K} mainly focuses on larger objects (e.g., sofa), and \mathcal{K}_y accurately captures smaller objects like baseball bat (see Suppl. for more results). NAT excels at capturing multi-scale objects but also includes the sparse region around each object.

Projecting Onto a Linear Subspace. When dealing with finite-dimensional $u(x)$, the $A_y(x)$ can be directly computed without incurring significant computational overhead. In the case of infinite or high-dimensional $u(x)$, the Nyström algorithm (Xiong et al. 2021) provides an efficient approximation for the embedding $v : R^d \rightarrow R^k$. The Nyström algorithm approximates the transport plan by sampling a subset of columns and rows, and projecting input from the

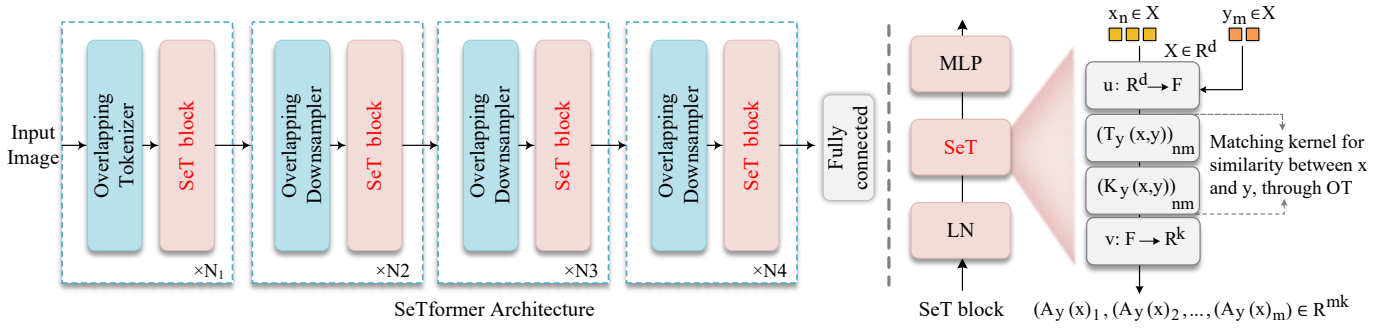


Figure 5: SeTformer Architecture (Left). It starts with a convolutional downsampler, followed by four sequence stages containing multiple SeT blocks. Consecutive stages are bridged by downsampler layers that reduce the spatial size while doubling the depth. On the right, we illustrate our attention formulation: mapping x and y elements to the RKHS, then aggregating x features as they align well with the corresponding reference through OT computation between x and y .

feature space \mathcal{F} onto a linear subspace \mathcal{F}_1 , which results in an embedding $\langle v(x_i), v(x'_j) \rangle_{\mathcal{F}_1}$. The subspace \mathcal{F}_1 is spanned by k centroids $u(z_1), \dots, u(z_k)$. The explicit formula $v(x_i) = k(z, z)^{-1/2} k(z, x_i)$ represents a new embedding with $z = z_1, \dots, z_k$ as centroids. This efficient method only requires performing K-means clustering and computing the inverse square root matrix. To implement this approximation, replace each x_i with its corresponding embedding $v(x_i)$, updating the attention model in (3),

$$m^{1/2} \left[\sum_{i=1}^n T(v(x), y)_{i1} v(x_i), \dots, \sum_{i=1}^n T(v(x), y)_{im} v(x_i) \right] \\ = m^{1/2} T(v(x), y)^T v(x). \quad (5)$$

here, $v(x) \in \mathbb{R}^{m \times k}$ with m denoting the features in y .

Linear Positional Encoding. In our model, to integrate positional information, we adopt the approach from Mairal (2016). This involves an exponential penalty based on the positional distance between input and reference sets. Specifically, we multiply $T(v(x), y)$ with a distance matrix \mathcal{M} , defined as $\mathcal{M}_{ij} = e^{(-\frac{1}{\tau^2})(\alpha-\beta)^2}$, where $\alpha = i/n$, $\beta = j/m$, and τ representing the smoothing parameter. The resulting value is then inserted into (5). Our similarity weights that consider both content and positional information achieve superior performance compared to other positional encoding methods, as detailed in Table 7.

SeT via DPSA. The proposed A_y performs sequence alignment w.r.t. a reference y using OT, while the self-attention in transformers relies on dot product similarities for feature aggregation. This provides an alternative approach for computing attention-like weights, addressing recent questions about the use of DSPA (Tay et al. 2021; Yang et al. 2022). We obtain our attention weight by computing $T(x, y_i)T(x, y_i)^T$ for each reference y_i . In comparing SeT with DPSA, the differences are evident in the elements of aggregation and alignment schemes. For DPSA, the attention matrix is computed as $\mathbf{W} = W^T Q$, where W and Q are learned parameters, with an aggregation based on x , size of $O(n^2)$ (n is the number of input x), and a position information function $P(t, t')$. On the other hand, SeT uses T

for the attention matrix, with aggregation based on y , size of $O(nm)$ (m is the number of elements in reference y), leverages y as the learnable parameter, and employs the exponential function $e^{(-\frac{1}{\tau^2})(\alpha-\beta)^2}$ for position information, where α and β are normalized positions in the input and reference sets, respectively. This distinct framework of SeT allows for more efficient computation of attention weights.

Model	Layers	# Param.	FLOPs
SeTformer-Miny	3, 4, 6, 5	16M	2.8G
SeTformer-Tiny	3, 4, 18, 5	22M	4.1G
SeTformer-Small	3, 4, 18, 5	35M	6.9G
SeTformer-Base	3, 4, 18, 5	57M	10.2G

Table 1: Different variants of the SeTformer architecture.

SeTformer. We use Swin as our baseline model, replacing its self-attention with our SeT module. Our model consists of four stages, each with a different spatial resolution, resulting in a spatial size 1/4th of the input image. Inputs are embedded using two-layer 3×3 convolutions with 2×2 strides. After each stage, except the last, there's a down-sampler via 3×3 convolutions with 2×2 strides. This is different from Swin, which uses non-overlapping 2×2 convolutions. The overall network architecture is presented in Figure 5, and different variants of our model are reported in Table 1.

Experimental Validation

We conduct experiments on both image and language domains, including ImageNet, COCO, and ADE20K, as well as the GLUE to demonstrate the impact of our model. We fine-tuned hyper-parameters like the number of references (m), entropic regularization ϵ in OT, and τ in position embedding. We observed that ϵ and τ showed stability across tasks, while careful selection is required for the value m . In the supplementary materials, we provided additional visualization results and the source code.

Classification on ImageNet-1K. The ImageNet-1K (Deng et al. 2009) has $\sim 1.28M$ images in 1000 classes. Following Swin's training setting, we utilize an AdamW (Kingma and Ba 2014) for 300 iterations, with 20 for

Model	# Param.	FL.	Thro.	Top-1
SeTformer-M	16.2M	2.8G	793/s	81.7
Swin-T (Liu et al. 2021)	28.2M	4.5G	755/s	81.3
ConvNeXt-T (Liu et al. 2022)	28.4M	4.5G	775/s	82.1
FocalNet-T (Yang et al. 2022)	28.6M	4.5G	696/s	82.3
CSWin-T (Dong et al. 2022)	22.8M	4.3G	701/s	82.7
MViTv2-T (Li et al. 2022b)	24.0M	4.7G	776/s	82.3
NAT-T (Hassani et al. 2023)	28.0M	4.3G	752/s	83.2
SeTformer-T	22.3M	4.1G	785/s	83.9 (+0.7)
Swin-S (Liu et al. 2021)	49.6M	8.7G	437/s	83.0
ConvNeXt-S (Liu et al. 2022)	49.5M	8.7G	448/s	83.1
FocalNet-S (Yang et al. 2022)	50.3M	8.7G	406/s	83.5
CSWin-S (Dong et al. 2022)	35.1M	6.9G	437/s	83.6
MViTv2-S (Li et al. 2022b)	35.0M	7.0G	341/s	83.6
NAT-S (Hassani et al. 2023)	51.0M	7.8G	435/s	83.7
SeTformer-S	35.0M	6.9G	451/s	84.7 (+1.0)
Swin-B (Liu et al. 2021)	87.4M	15.4G	278/s	83.3
ConvNeXt-B (Liu et al. 2022)	88.9M	15.4G	297/s	83.5
FocalNet-B (Yang et al. 2022)	88.7M	15.4G	296/s	83.9
CSWin-B (Dong et al. 2022)	78.2M	15.2G	250/s	84.2
MViTv2-B (Li et al. 2022b)	52.0M	10.2G	253/s	84.4
NAT-B (Hassani et al. 2023)	90.0M	13.7G	294/s	84.4
SeTformer-B	56.7M	10.2G	298/s	86.2 (+1.8)

Table 2: Classification accuracy on the ImageNet-1K at 224×224 resolution.

Model	Top-1 acc.	GFLOPs
Baseline (Yuan et al. 2021)	82.55	161.10
Performer (Dosovitskiy et al. 2021)	80.50	5.06 (31.87 \times)
Reformer (Kitaev et al. 2020)	81.44	11.71 (13.75 \times)
ScatterBrain (Chen et al. 2021)	81.95	7.18 (22.43 \times)
KDEformer (Zandieh et al. 2023)	82.08	8.80 (18.30 \times)
SeTformer	83.17	6.9 (23.35\times)

Table 3: Classification results on the ImageNet-1K adopting T2T-ViT, with the kernel-based methods.

warm-up of the learning rate, followed by gradual decay, and then perform ten cool down epochs. The results of top-1 accuracy are reported in Table 2, and Figure 1. In our ablation study we showed that the optimal reference size for this task is 500. Raising m can enhance performance until reaching a saturation point. We also set ϵ to 0.1 and τ to 0.5. SeTformers consistently outperform ConvNeXt with smaller model size, Flops, and throughput. Our mini variant exceeds Swin-T by +0.4%, using 40% fewer parameters (28M \rightarrow 16M) and 37% fewer Flops. Our Tiny model (83.9%) surpasses CSWin by +1.2% in performance with a similar model size, resulting in a 12% speedup (from 701/s to 785/s). It also outperforms FocalNet-T by +1.6% while being lighter. With larger models, we achieve state-of-the-art performance with fewer parameters and lower computational costs. For example, SeTformer-B outperforms NAT-B (84.4%) by a margin of +1.8%, while having over 24% and 36% fewer Flops and parameters. We also note that, the throughputs are measured on a V100 GPU. We also evaluate the kernel-attention models on ImageNet classification, and ensuring fairness by using the same T2T-ViT backbone for all models. With a pre-trained 24-layer model, we apply our method to 2 attention layers in the T2T module. On the ImageNet validation set, we measure top-1 accuracy and Flops of the first attention layer, which is the most resource-intensive. The results (Table

Mask R-CNN - 3x schedule							
Model	AP ^b	AP ^b ₅₀	AP ^b ₇₅	AP ^m	AP ^m ₅₀	AP ^m ₇₅	Param. FL.
ResNet-50	41.2	61.8	44.9	37.2	58.4	40.1	44M 260
ResNet-101	42.7	63.2	47.5	38.4	60.3	41.5	63M 335
SeTformer-M	46.7	68.2	51.7	41.9	65.1	44.7	32M 190
Swin-T	46.0	68.2	50.4	41.6	65.2	44.8	48M 264
FocalNet-T	48.0	69.7	53.0	42.9	66.5	46.1	49M 268
CSWin-T	49.0	70.7	53.7	43.6	67.9	46.6	42M 279
NAT-T	47.7	69.0	52.6	42.6	66.1	45.9	48M 258
MViTv2-T	48.2	70.9	53.3	43.8	67.9	47.2	44M 279
SeTformer-T	49.3	71.5	53.9	44.0	68.3	47.8	40M 245
Swin-S	48.5	70.3	53.5	43.3	67.4	46.5	69M 359
FocalNet-S	49.3	70.7	54.2	43.8	68.0	47.4	72M 365
CSWin-S	50.0	71.3	54.7	44.5	68.4	47.7	54M 342
NAT-S	48.4	69.8	53.2	43.2	66.9	46.5	70M 330
MViTv2-S	49.9	72.0	55.0	45.1	69.5	48.5	54M 326
SeTformer-S	51.3	73.1	55.6	45.9	71.2	49.4	52M 312
FocalNet-B	49.8	70.9	54.6	44.1	68.2	47.2	111M 507
CSWin-B	50.8	72.1	55.8	44.9	69.1	48.3	97M 526
MViTv2-B	51.0	72.7	56.3	45.7	69.9	49.6	71M 392
SeTformer-B	51.9	72.8	57.2	46.3	70.3	51.2	68M 351

Table 4: Object detection using Mask R-CNN on the COCO. FLOPs (FL) are measured on the input resolution of 1280×800 . All backbones are pre-trained on ImageNet-1K.

Model	# Param.	FLOPs	SS/MS mIoU
SeTformer-M	40M	758G	45.8 / 46.9
Swin-T (Liu et al. 2021)	60M	946G	44.5 / 45.8
ConvNeXt-T (Liu et al. 2022)	60M	939G	46.1 / 46.7
NAT-T (Hassani et al. 2023)	58M	934G	47.1 / 48.4
CSWin-T (Dong et al. 2022)	60M	959G	49.3 / 50.7
SeTformer-T	51M	873G	50.6 / 51.4
Swin-S (Liu et al. 2021)	81M	1040G	47.6 / 49.5
ConvNeXt-S (Liu et al. 2022)	82M	1024G	48.7 / 49.6
NAT-S (Hassani et al. 2023)	82M	1010G	48.0 / 49.5
CSWin-S (Dong et al. 2022)	65M	1027G	50.4 / 51.5
SeTformer-S	59M	986G	51.1 / 51.9
Swin-B (Liu et al. 2021)	121M	1188G	48.1 / 49.7
ConvNeXt-B (Liu et al. 2022)	122M	1170G	49.1 / 49.9
NAT-B (Hassani et al. 2023)	123M	1137G	48.5 / 49.7
CSWin-B (Dong et al. 2022)	109M	1222G	51.1 / 52.2
SeTformer-B	81M	1008G	52.0 / 52.8

Table 5: Semantic segmentation on the ADE20K. FLOPs are computed based on an input resolution (2048, 512).

3) highlight that our model achieves the highest accuracy at 83.17%, while demanding 23 times fewer operations. The Performer, although the fastest, exhibits a significant -2.05% accuracy drop compared to the vanilla transformer.

Object Detection on COCO. We perform object detection on MS-COCO (Lin et al. 2014). Models are trained on 118K training images and evaluated on 5K validation set using Mask R-CNN (He et al. 2017). We pre-train the backbone on ImageNet-1k, fine-tune on COCO for 36 epochs ($3 \times$ schedule). We choose the number of references m as 750, and set ϵ to 0.3 and τ to 0.8. Table 4 presents our results. SeTformer outperforms CNN (e.g., ResNet) and Transformer backbones (e.g., CSWin, NAT, MViTv2). For example, SeTformer-T outperforms NAT-T by +1.6/+1.4 in AP^b/AP^m, while requiring less computation and having a smaller model size. When scaling up, SeTformer-B (51.9) improves over CSWin-B (50.8) by +1.1 AP^b, while using 28% fewer parameters and 33% fewer Flops.

Model	QQP	SST-2	MNLI
Roberta-base (Liu et al. 2019)	88.41	92.31	79.15
Performer (Dosovitskiy et al. 2021)	69.92	50.91	35.37
Reformer (Kitaev et al. 2020)	63.18	50.92	35.47
Linear Trans. (Katharopoulos et al. 2020)	74.85	84.63	66.56
Longformer (Beltagy et al. 2020)	85.51	88.65	77.22
FAVOR++ (Likhoshesterov et al. 2022)	88.43	92.23	78.91
SeTformer	88.76	92.28	79.16

Table 6: classification results on GLUE for different kernel-based transformers using (Liu et al. 2019) as baseline.

Model / datasets	ImageNet	COCO	
	Top/1	AP ^b	Train(iter/s)
1) no pos.	83.78	49.63	3.2
2) APE (Dosovitskiy et al. 2021)	83.96	50.41	2.9
3) RPE (Liu et al. 2021)	84.35	50.89	1.5
4) LPE (ours)	84.72	51.32	3.1
Model / datasets	GLUE		
	QQP	SST-2	MNLI
5) dot product instead of OT	87.85	89.26	78.03
6) Sliced-OT (Kolouri et al. 2021)	86.74	88.96	76.51
7) \mathcal{K}	88.92	91.73	78.60
8) \mathcal{K} (w/o position encoding)	86.32	88.25	77.38
9) \mathcal{K}_y	89.36	93.54	79.81

Table 7: Ablation study. Rows [1-4] explore positional bias; Rows [5-6] examine the effect of using OT; Rows[7-8] analyze the reference impact in the SeTformer-S.

Semantic Segmentation on ADE20K. We also evaluate SeTformer on ADE20K (Zhou et al. 2019) for semantic segmentation using UperNet (Xiao et al. 2018). Methods are trained for 160K epochs using batch size 16, following (Liu et al. 2021). Our optimal result is achieved with $m = 800$, ϵ and τ set to 0.3 and 0.8, respectively. In Table 5, we report the mIoU and MS mIoU results of different methods. SeTformer-T and SeTformer-S outperform CSWin counterparts by +1.3/+0.7 and +0.7/+0.4 mIoU (SS/MS), respectively, while being lighter with lower complexity.

Experiments on Language Modeling. In this section, we investigate SeTformer’s efficacy on the GLUE benchmark (Wang et al. 2018) across QQP, SST-2, and MNLI tasks. To ensure fairness, all transformers, including SeTformer, are pre-trained for 50K iterations on the WikiText-103 (Merity et al. 2017). We fine-tune the models using training parameters from RoBERTa (Liu et al. 2019). By comparing SeTformer with other scalable transformers in Table 6, we observe that SeTformer consistently outperforms the baseline (Liu et al. 2019). It achieves competitive or superior performance across various datasets in comparison to other transformers. The GLUE benchmark is a diverse set of language tasks that rely on context and word relationships. SeTformer’s unique attention mechanism based on optimal transport could effectively model these relationships, potentially leading to improved performance.

Ablation Studies

Positional Encoding Comparison: In this section, we evaluate the important components of the SeTformer using ImageNet, COCO, and GLUE datasets. Table 7, Rows [1-4] ab-

Model	No. of ref.	ImageNet	COCO	ADE20K
SeTformer-T	100	79.81	45.76	46.93
	300	82.54	46.59	48.21
	500	83.91	48.55	49.11
	750/800	83.25	49.34/49.37	50.62
	1000	81.74	48.19	50.08

Table 8: Number of m vs. performance: performance improves with increasing m , but saturates at high values.

Model	ImageNet-1k		COCO	
	Acc.	Test (im/s)	AP ^{box}	Train(iter/s) GFLOPs
MViTv2-B	84.4	253	51.0	2.1 392
SeTformer-S	84.7	451	51.3	3.1 245
SeTformer-B	86.2	298	51.9	2.4 371

Table 9: Runtime comparison on ImageNet-1K and COCO.

lates the different position embedding techniques. We find: (i) Absolute positions (2) only slightly improve over no position (1), as pooling operators already encode position. (ii) Relative positions (3) boost performance by introducing shift-invariance. Notably, our linear positional information proves remarkably efficient, particularly in COCO, where it accelerates training by $2.1\times$ versus relative positions. **Replacing OT with other variants:** Rows [5-6] utilize dot product (Vaswani et al. 2017) and sliced-OT instead of OT. The results are less favorable compared to our model. **Effect of using references:** Rows [7-8] explore the impact of references. (7) employs \mathcal{K} without the reference concept, while (8) indicates using \mathcal{K} without adding positional encoding. Comparing Rows (9) with (7), the presence of references enhances performance. We analyze the model’s sensitivity to the number of elements m in Table 8. For example, in the COCO scenario, using $m = 800$ yields slightly improved results (49.37), while $m = 750$ also provides good performance (49.34) and it has the advantage of being lighter. While increasing m can enhance performance, it becomes less impactful for very large values. **Runtime comparison:** The runtime comparison between SeTformer, NAT, and Swin is presented in Table 9. SeTformer-S surpasses MViT-B in both ImageNet (+0.3%) and COCO (+1.2%) datasets, while having higher throughput (451 im/s vs. 253 im/s) on ImageNet and faster training (3.1 iter/s vs. 2.1 iter/s) on COCO. SeTformer-B is slightly slower but remarkably more accurate (+1.8% on ImageNet-1K and +0.9% on COCO).

Conclusion

We proposed SeTformer, an efficient transformer based on Self-optimal Transport (SeT). Our SeTformer benefits from the properties of kernel methods and optimal transport effectively captures complex relationships and ensures that attention scores are both non-negative and follow a reweighting scheme similar to the softmax mechanism. By leveraging this unique structure, we have created a series of SeTformer-X architectures that consistently surpass existing transformers across vision and language domains. Our work can provide another perspective on attention modulation to explore better content-based interactions, which can benefit visual recognition models.

Acknowledgments

In this paper, P. Shamsolmoali and M. Zareapoor have made equal contributions. This work was partly supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP), funded by the Knut and Alice Wallenberg Foundation, and the Swedish Research Council grant 2022-04266.

References

- Beltagy, I.; Peters, M. E.; Cohan, and Arman. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Bietti, A.; and Mairal, J. 2019. Group invariance, stability to deformations, and complexity of deep convolutional representations. *J. Mach. Learn. Res.*, 20(1).
- Chen, B.; Dao, T.; Winsor, E.; Song, Z.; Rudra, A.; and Ré, C. 2021. Scatterbrain: Unifying sparse and low-rank attention. *Proc. NIPS*.
- Choromanski, K.; Likhoshesterov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlos, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; et al. 2021. Rethinking attention with performers. *Proc. ICLR*.
- Choromanski, K. M.; Li, S.; Likhoshesterov, V.; Dubey, K. A.; Luo, S.; He, D.; Yang, Y.; Sarlos, T.; Weingarten, T.; and Weller, A. 2023. Learning a Fourier Transform for Linear Relative Positional Encodings in Transformers. *arXiv preprint arXiv:2302.01925*.
- Chowdhury, S. P.; Solomou, A.; Dubey, A.; and Sachan, M. 2022. Learning the transformer kernel. *Trans. Mach. Learn. Res.*
- Chu, X.; Tian, Z.; Zhang, B.; Wang, X.; Wei, X.; Xia, H.; and Shen, C. 2023. Conditional positional encodings for vision transformers. In *Proc. ICLR*.
- Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Proc. NIPS*.
- De Plaen, H.; De Plaen, P.-F.; Suykens, J. A.; Proesmans, M.; Tuytelaars, T.; and Van Gool, L. 2023. Unbalanced Optimal Transport: A Unified Framework for Object Detection. In *Proc. CVPR*.
- Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*.
- Dong, X.; Bao, J.; Chen, D.; Zhang, W.; Yu, N.; Yuan, L.; Chen, D.; and Guo, B. 2022. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *Proc. CVPR*.
- Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.
- Fukumizu, K. 2008. Elements of positive definite kernel and reproducing kernel hilbert space. *Institute of Statistical Mathematics, ROIS*, 4.
- Gao, B.; and Pavel, L. 2017. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*.
- Grainger, R.; Paniagua, T.; Song, X.; Cuntoor, N.; Lee, M. W.; and Wu, T. 2023. PaCa-ViT: Learning Patch-to-Cluster Attention in Vision Transformers. In *Proc. CVPR*.
- Hassani, A.; Walton, S.; Li, J.; Li, S.; and Shi, H. 2023. Neighborhood attention transformer. In *Proc. CVPR*.
- He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proc. ICCV*.
- Huang, T.; Huang, L.; You, S.; Wang, F.; Qian, C.; and Xu, C. 2022. LightViT: Towards Light-Weight Convolution-Free Vision Transformers. *arXiv preprint arXiv:2207.05557*.
- Katharopoulos, A.; Vyas, A.; Pappas, N.; and Fleuret, F. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proc. ICML*.
- Khamis, A.; Tsuchida, R.; Tarek, M.; Rolland, V.; and Petersson, L. 2023. Earth Movers in The Big Data Era: A Review of Optimal Transport in Machine Learning. *IEEE Trans. Pattern Anal. Mach. Intell.*
- Khan, S.; Naseer, M.; Hayat, M.; Zamir, S. W.; Khan, F. S.; and Shah, M. 2022. Transformers in vision: A survey. *ACM Comput. Surveys*.
- Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kitaev, N.; Kaiser, L.; and Levskaya, A. 2020. Reformer: The efficient transformer. *Proc. ICLR*.
- Kolouri, S.; Naderializadeh, N.; Rohde, G. K.; and Hoffmann, H. 2021. Wasserstein embedding for graph learning. *Proc. ICLR*.
- Lee, J.; Lee, Y.; Kim, J.; Kosiorek, A.; Choi, S.; and Teh, Y. W. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proc. ICML*.
- Li, J.; Xia, X.; Li, W.; Li, H.; Wang, X.; Xiao, X.; Wang, R.; Zheng, M.; and Pan, X. 2022a. Next-vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios. *arXiv preprint arXiv:2207.05501*.
- Li, Y.; Wu, C.; Fan, H.; Mangalam, K.; Xiong, B.; Malik, J.; and Feichtenhofer, C. 2022b. Improved multiscale vision transformers for classification and detection. *arXiv 2021*. In *Proc. CVPR*.
- Likhoshesterov, V.; Choromanski, K. M.; Dubey, K. A.; Liu, F.; Sarlos, T.; and Weller, A. 2022. Chefs' Random Tables: Non-Trigonometric Random Features. *Proc. NIPS*.
- Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Proc. ECCV*.
- Liu, T.; Puigcerver, J.; and Blondel, M. 2023. Sparsity-Constrained Optimal Transport. *Proc. ICLR*.
- Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

- Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. ICCV*.
- Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A convnet for the 2020s. In *Proc. CVPR*.
- Mairal, J. 2016. End-to-end kernel learning with supervised convolutional kernel networks. *Proc. NIPS*.
- Merity, S.; Xiong, C.; Bradbury, J.; and Socher, R. 2017. Pointer sentinel mixture models. *Proc. ICLR*.
- Mialon, G.; Chen, D.; d'Aspremont, A.; and Mairal, J. 2021. A trainable optimal transport embedding for feature aggregation and its relationship to attention. *Proc. ICLR*.
- Peng, H.; Pappas, N.; Yogatama, D.; Schwartz, R.; Smith, N. A.; and Kong, L. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143*.
- Peyré, G.; and Cuturi, M. 2019. Computational optimal transport: With applications to data science. *Trends Mach. Learn.*, 11.
- Reid, I.; Choromanski, K. M.; Likhoshesterov, V.; and Weller, A. 2023. Simplex random features. In *Proc. ICML*.
- Schölkopf, B.; Smola, A. J.; Bach, F.; et al. 2002. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.
- Shamsolmoali, P.; Zareapoor, M.; Das, S.; Granger, E.; and García, S. 2024. Hybrid Gromov-Wasserstein Embedding for Capsule Learning. *IEEE Trans. Neural Net. Learn. Syst.*
- Sinkhorn, R.; and Knopp, P. 1967. Concerning nonnegative matrices and doubly stochastic matrices. *Pac. J. Math.*, 21(2).
- Skianis, K.; Nikolentzos, G.; Limnios, S.; and Vazirgiannis, M. 2020. Rep the set: Neural networks for learning set representations. In *Proc. AISTATS*.
- Tay, Y.; Bahri, D.; Metzler, D.; Juan, D.-C.; Zhao, Z.; and Zheng, C. 2021. Synthesizer: Rethinking self-attention for transformer models. In *Proc. ICML*.
- Tolias, G.; Avrithis, Y.; and Jégou, H. 2013. To aggregate or not to aggregate: Selective match kernels for image search. In *Proc. ICCV*.
- Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021. Training data-efficient image transformers & distillation through attention. In *Proc. ICML*.
- Vaswani, A.; Ramachandran, P.; Srinivas, A.; Parmar, N.; Hechtman, B.; and Shlens, J. 2021. Scaling local self-attention for parameter efficient visual backbones. In *Proc. CVPR*.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Proc. NIPS*.
- Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.
- Wang, S.; Li, B. Z.; Khabsa, M.; Fang, H.; and Ma, H. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.
- Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proc. ICCV*.
- Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2022. Pvt v2: Improved baselines with pyramid vision transformer. *Comput. Vis. Media*, 8(3).
- Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. Cvt: Introducing convolutions to vision transformers. In *Proc. ICCV*.
- Xiao, T.; Liu, Y.; Zhou, B.; Jiang, Y.; and Sun, J. 2018. Unified perceptual parsing for scene understanding. In *Proc. ECCV*.
- Xiong, Y.; Zeng, Z.; Chakraborty, R.; Tan, M.; Fung, G.; Li, Y.; and Singh, V. 2021. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proc. AAAI*.
- Yang, J.; Li, C.; Dai, X.; and Gao, J. 2022. Focal modulation networks. *Proc. NIPS*.
- Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.-H.; Tay, F. E.; Feng, J.; and Yan, S. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proc. ICCV*.
- Zandieh, A.; Han, I.; Daliri, M.; and Karbasi, A. 2023. Kde-former: Accelerating transformers via kernel density estimation.
- Zhang, Z.; Wang, M.; and Nehorai, A. 2019. Optimal transport in reproducing kernel hilbert spaces: Theory and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(7).
- Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S.; Barriuso, A.; and Torralba, A. 2019. Semantic understanding of scenes through the ade20k dataset. *I. J. Comput. Vis.*, 127.
- Zhu, C.; Ping, W.; Xiao, C.; Shoneybi, M.; Goldstein, T.; Anandkumar, A.; and Catanzaro, B. 2021. Long-short transformer: Efficient transformers for language and vision. *Proc. NIPS*.