Exploring Fast Attention Mechanisms: Performer vs Vision Transformers for Image Classification

Simon Xu

2024/12/3

1. Abstract

In recent years, Transformer models have set new benchmarks in various domains, including natural language processing, computer vision, and multimodal tasks. Among these, Vision Transformers (ViT) and Performer models represent two distinct approaches to addressing the computational challenges associated with high-dimensional data and attention mechanisms. This paper explores the trade-offs between speed and accuracy of ViT and Performer models by leveraging various attention kernels: Performer-ReLU, Performer-exp, and a novel learnable kernel fθ designed to outperform the former two. Using image classification datasets, including CIFAR10 and Tiny ImageNet, I evaluate training and inference times alongside classification accuracy. My findings highlight the strengths and limitations of these architectures under different configurations, offering insights into the practical deployment of fast-attention mechanisms in real-world scenarios. Finally, I discuss the potential of the proposed learnable kernel in advancing efficient Transformer-based architectures.

2. Introduction

2.1. Motivation and Background

The advent of the transformer architecture, introduced in the seminal work "Attention is All You Need" by Vaswani et al. (2017), has revolutionized the field of deep learning. Transformers leverage the self-attention mechanism, which enables models to capture dependencies between input elements efficiently without relying on recurrent or convolutional layers. This mechanism, coupled with position-wise feed-forward layers, empowers transformers to process input data in parallel, achieving unprecedented performance across various domains, such as natural language processing and computer vision.

However, the vanilla transformer architecture comes with notable limitations, particularly in terms of computational and memory complexity. The quadratic time and memory demands of the self-attention mechanism hinder its scalability for large-scale inputs, such as high-resolution images or lengthy text sequences. These constraints have driven researchers to explore more efficient adaptations and alternatives to the original transformer model.

2.2. Advancing Vision Tasks: From Transformers to Vision Transformers (ViTs)

The success of transformers in natural language processing inspired their application to vision tasks, culminating in the Vision Transformer (ViT). ViTs demonstrated that transformers, when applied directly to image patches, could outperform convolutional neural networks (CNNs) in image classification tasks. By leveraging global attention mechanisms, ViTs capture long-range dependencies in visual data, offering a flexible and scalable approach to image understanding.

Nevertheless, the quadratic complexity of the self-attention mechanism in ViTs poses challenges for processing high-resolution images or deploying the model in resource-constrained environments. To address these limitations, various approaches have been proposed to optimize transformers for vision tasks, focusing on reducing the computational overhead while preserving the expressive power of the attention mechanism.

2.3. Performer: An Efficient Alternative

Among the innovations addressing the computational bottlenecks of transformers is the Performer, which introduces the concept of Fast Attention via Positive Orthogonal Random Features (FAVOR+). The Performer approximates the self-attention mechanism with linear complexity, making it feasible to handle larger input sizes without compromising accuracy significantly. This breakthrough has paved the way for scaling transformer-based models to previously infeasible scenarios, such as processing high-resolution visual data in real-time applications.

2.4. Research Motivation

Despite these advancements, challenges remain in balancing efficiency and accuracy in transformer-based architectures for vision tasks. The quadratic complexity of vanilla transformers and the trade-offs involved in linear attention mechanisms like Performers highlight the need for further research. This study aims to investigate and extend the applicability of efficient transformer variants, such as the Performer, to enhance vision-specific tasks while maintaining computational feasibility.

3. Architecture Review

3.1. Vision Transformer (ViT)

The Vision Transformer (ViT) represents a significant innovation in applying Transformer models to image data. Unlike traditional convolutional neural networks (CNNs), which utilize local receptive fields, ViT processes an image by dividing it into fixed-size patches. Each patch is treated as a token, similar to the tokenization in natural language processing.

- Patch Embeddings: Input images are divided into non-overlapping patches of a fixed size. Each patch is flattened and linearly projected into a fixed-dimensional vector. Positional embeddings are added to these patch embeddings to encode spatial information.
- Transformer Encoder: The core of ViT consists of a stack of Transformer encoder layers. Each encoder layer comprises multi-head self-attention mechanisms and feed-forward layers. These layers model long-range dependencies among image patches, making the model capable of capturing global context efficiently.
- Pooling Layer: Depending on the configuration, ViT utilizes either a classification (CLS) token or mean pooling to aggregate the output features of the encoder. These aggregated features are passed through an MLP head to produce the final classification results.

3.2. Performer

Performers are designed to address the computational bottlenecks of traditional Transformers, particularly the quadratic complexity associated with self-attention. They introduce kernel-based approximations that scale linearly with the input size, making them highly efficient for handling large datasets and high-dimensional data.

Fast Attention via Orthogonal Random Features (FAVOR+): Performers replace the traditional softmax attention mechanism with a linear approximation based on kernel functions. The key idea is to approximate the softmax kernel using a feature map $\phi(x)$, such that:

$K(Q,K) \approx \phi(Q)\phi(K)^T K(Q, K)$

This reduces the time complexity from $O(N^2)$ to $O(N)$ while maintaining comparable accuracy.

Kernel Variants: Performers allow flexibility in kernel function selection, including deterministic functions like ReLU, exponential functions, and learnable functions (as in my Performer-fθ variant).

Generalized Attention: By leveraging custom kernel functions, Performers extend the attention mechanism to tasks beyond softmax-based approaches. This generalization is particularly useful in tasks requiring non-linear feature transformations.

3.3. Key Architectural Differences

Self-Attention Complexity: ViT uses the traditional quadratic self-attention mechanism, making it resource-intensive for large datasets. Performers significantly mitigate this issue by introducing linear attention.

Positional Encoding: Both models employ positional encoding, but Performers can handle positional embeddings more flexibly due to their kernel-based design.

Application Scope: While ViT excels in accuracy for small datasets, Performers demonstrate superior scalability and efficiency for large-scale tasks.

This architecture review establishes the foundation for my comparative experiments, aiming to measure speed-accuracy trade-offs and assess whether my Performer-fθ variant surpasses the standard Performer implementations.

4. Experiment Methodology

4.1. Datasets

To evaluate the performance of Vision Transformer (ViT) and Performer architectures, I conducted experiments on two widely recognized image classification datasets: CIFAR10 and Tiny ImageNet. These datasets were chosen to provide a balance between accessibility, computational feasibility, and diversity in classification challenges.

CIFAR10

Overview: CIFAR10 is a widely-used benchmark dataset in computer vision, consisting of 60,000 color images with a resolution of 32×32 pixels.

Classes: The dataset is evenly divided into 10 mutually exclusive classes, including airplanes, cars, birds, and more.

Dataset Split:

Training set: 50,000 images.

Test set: 10,000 images.

Preprocessing:

Images were resized and divided into patches of 4×4 pixels for input into the Vision Transformer (ViT) and Performer models.

Standard data augmentation techniques were applied, such as random cropping, horizontal flipping, and normalization using the dataset's mean and standard deviation.

Tiny ImageNet

Overview: Tiny ImageNet is a reduced-scale version of the ImageNet dataset, specifically designed for faster training and evaluation of machine learning models. It contains images of 64×64 pixels across a large number of classes.

Classes: The dataset includes 200 classes, each with a unique semantic label.

Dataset Split:

Training set: 500 images per class, resulting in 100,000 training images.

Validation set: 50 images per class (10,000 validation images in total).

Test set: 10,000 images without class labels, intended for blind testing.

Preprocessing:

Experiments were conducted using patch sizes of both 4×4 and 8×8 pixels, enabling an analysis of how patch size affects model performance.

Similar data augmentation techniques as CIFAR10 were applied, along with resizing to the appropriate resolution for patch embedding.

Purpose of Dataset Selection

The use of CIFAR10 and Tiny ImageNet ensures that my experiments capture performance across varying levels of task complexity:

CIFAR10 focuses on evaluating how models perform on low-resolution images with fewer classes.

Tiny ImageNet pushes the models to generalize across a more diverse set of classes and higher-resolution images, making it suitable for assessing scalability and robustness.

This diverse setup enables a comprehensive understanding of the speed-accuracy tradeoff for ViT and Performer architectures under different task complexities and patch configurations.

4.2. Experimental Setup

Training Parameters: Lots of hypamarameters were chosen to see how I can enhance the performance on validation set. The parameters include embedding dimension, number of transformer layers, number of attention heads, MLP dimension, pooling strategy(using CLS or average), dimension of each attention head, dropout ratio, embedding dropout ratio, batch_size. These hyperparameters were modified and changed between experiments.

Learning Rate: Because there are costs related to using a A10 GPU, only limited experiments were conducted. The initial learning rate was set from 0.0008 to 0.0025 in different experiments and adjusted dynamically using a cosine annealing learning rate scheduler.

Batch Size: Batch size is 256 to ensure computational speed.

Optimizer: All models were optimized using the Adam optimizer with a weight decay of 3×10^{-5}.

Epochs: The number of training epochs was associated with the scheduler used. 70 and 84 epochs are used in this report to match a CosineAnnealingWarmRestarts scheduler with T_0 = 10 and T_0 = 12.

Hardware Environment:

GPU Model: NVIDIA A10

CUDA Version: 12.1

VRAM: 23GB (for each A10 GPU)

Driver Version: 530.30.02

The model was trained on a NVIDIA A10 GPU with 40GB VRAM across multiple processes, enabling efficient model training even with large datasets. I used PyTorch 2.1.1 and CUDA 12.1 for GPU acceleration.

CPU Model: Intel Xeon Platinum 8336C

CPU Cores: 28 physical cores, 56 logical cores

Clock Speed: 2.30 GHz

The training machine runs on a powerful Intel Xeon Platinum 8336C CPU with 28 physical cores, ensuring that pre-processing and other tasks do not become bottlenecks.

GPU: All experiments were conducted on a single NVIDIA A100 GPU with 40 GB of VRAM.

Framework: PyTorch 2.1.1

CUDA Version: 12.1.

4.3. Evaluation Metrics

The following metrics were used to evaluate the models:

Accuracy: Classification accuracy was measured on the validation set for each dataset.

Training Time: The time taken to complete one epoch and the total training time were recorded.

Inference Time: The time required to make predictions on the test dataset was measured.

Resource Utilization: GPU memory usage during training was monitored to assess the models' computational efficiency. The original VIT model was set to a score of 100.

4.4. Experiment Design

Baseline Models: I compared four Transformer architectures:

ViT: Using the standard self-attention mechanism with quadratic complexity.

Performer-ReLU: A Performer variant employing a ReLU kernel in the fast attention mechanism.

Performer-exp: A Performer variant with an exponential kernel for the softmax attention approximation.

fθ Variant: I designed a learnable kernel variant, Performer-fθ, where the kernel function fθ is parameterized as a neural network (e.g., a single linear layer followed by activation function ReLU). The objective was to test whether a learnable kernel could achieve better accuracy-resource trade-offs than Performer-ReLU and Performer-exp. Because a learnable kernel of course add complexity to the whole structure, and of course introduce extra parameters to the model. In order to beat Performer-relu and Performer-exp in speed, I need to make sure the neural network is not too deep so that the learnable kernel won't take too long to train and won't cost a long time to infer. So a learnable matrix nn.Parameter(torch.randn(input_dim, input_dim)) is introduced, followed by a layer of activation. Using this generalized kernel, I expect it to have a better speed-accuracy performance over Performer-relu and Performer-exp.

Patch Size Analysis: While CIFAR10 experiments were restricted to a patch size of 4×4, Tiny ImageNet experiments tested both 4×4 and 8×8 patch sizes to evaluate the models' sensitivity to sequence length.

Procedure:

Models were trained using identical training parameters to ensure fair comparison.

For each experiment, training, and testing were performed, and the metrics were logged. Within each experiment, all four architectures used the same set of training parameters, and between experiments, training parameters were modified to compare the performance under different parameter settings.

The evaluation emphasized speed-accuracy trade-offs, particularly the scalability of Performer models with larger datasets and patch sizes.

This methodology ensures a rigorous and comprehensive comparison between ViT, Performer variants, and my proposed Performer-fθ architecture.

5. Experiment Results

The first experiment (I will call this Experiment I) was on CIFAR10. Because CIFAR10 is really a small dataset with image size=32, we use patch size=4 and cut all images into 8*8 patches. The training parameters for the first experiment is listed below, and the four columns represents the original VIT, Performer-relu, Performer-exp and Performer-fθvariant.

Table 1 Training params for ExpI

model_type	vit	performer	performer	performer
image_size	32	32	32	32
patch_size	4	4	4	4
num_classes	10	10	10	10
dim	128	128	128	128
depth	6	6	6	6
heads	8	8	8	8
mlp_dim	200	200	200	200
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	32	32	32	32
dropout	0.15	0.15	0.15	0.15
emb_dropout	0.15	0.15	0.15	0.15
batch_size	256	256	256	256
num_epochs	70	70	70	70
learning_rate	0.0015	0.0015	0.0015	0.0015
epoch_step	5	5	5	5
nb_features		16	16	16
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

The results for Experiment I are shown in Table 2.

Table 2 Results for ExpI

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	9.25	1.01	83.63	83.63	100.00
Performer-relu	9.19	1.08	82.05	82.05	72.63
Performer-exp	10.96	1.24	78.40	78.40	93.51
Performer-fθ	9.55	1.11	82.38	82.38	80.42

For CIFAR, VIT has the highest accuracy as expected, followed by learnable kernel variant, relu kernel variant and exponential kernel variant. In this scenario, my designed learnable kernel beat relu and exp in terms of final accuracy. But regarding speed, only relu kernel ourperform the original VIT in terms of training time. And no performer variant beats VIT in terms of inference time. A possible reason may be that CIFAR is a small dataset with very small image sizes. Thus, the strength of Performer isn't revealed. But when we look at memory, Performer variants do save memory for us, with relu-kernel leading the three variants.

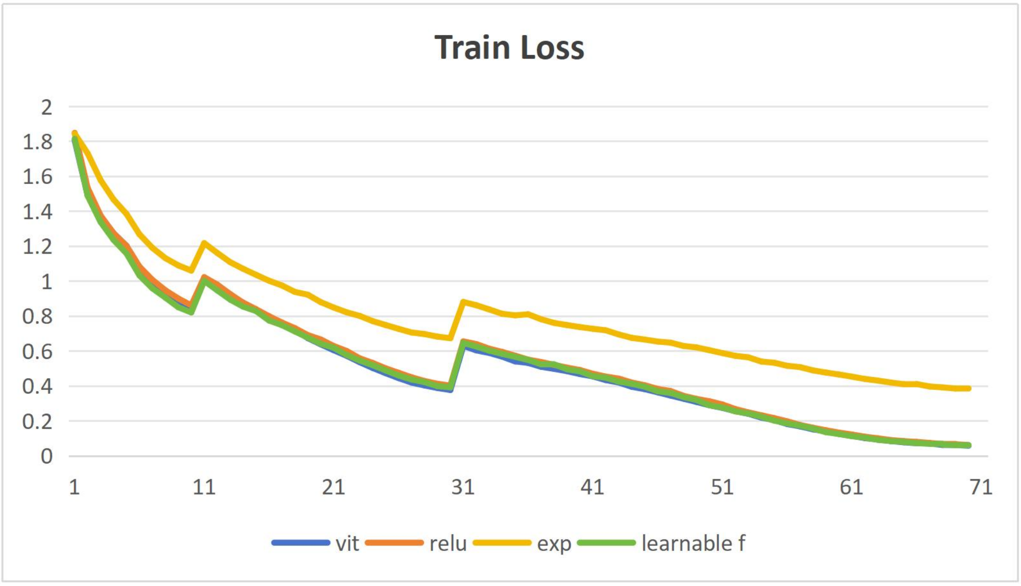

Figure 1 Train loss for Exp I

Experiment II uses the same sets of training parameters with Experiment I, but uses 64 random features to see the performance of performers.

Table 3 Training params for ExpII

model_type	vit	performer	performer	performer
image_size	32	32	32	32
patch_size	4	4	4	4
num_classes	10	10	10	10
dim	128	128	128	128
depth	6	6	6	6
heads	8	8	8	8
mlp_dim	200	200	200	200
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	32	32	32	32
dropout	0.15	0.15	0.15	0.15
emb_dropout	0.15	0.15	0.15	0.15
batch_size	256	256	256	256

num_epochs	70	70	70	70
learning_rate	0.0015	0.0015	0.0015	0.0015
epoch_step	5	5	5	5
nb_features		64	64	64
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

From the results we can see, when we elevate the number of random features, all the time cost dramatically increased, and the resourced used has also changed.

Table 4 Results for Exp II

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	9.25	1.01	83.63	83.63	100.00
Performer-relu	12.08	1.29	81.32	81.32	121.14
Performer-exp	15.29	1.61	79.59	79.59	138.13
Performer-fθ	13.28	1.37	81.65	81.65	151.36

When we use 64 random features, VIT seems to be the best in terms of both speed and accuracy. (This can be seen also from the test loss curve). But among the Performer variants, my learnable kernel still has the best accuracy, with slightly larger inference time than Performer-relu. However, the resource used for my learnable kernel has also dramatically increased, and it could be a problem when facing large datasets where we have to use more random features.

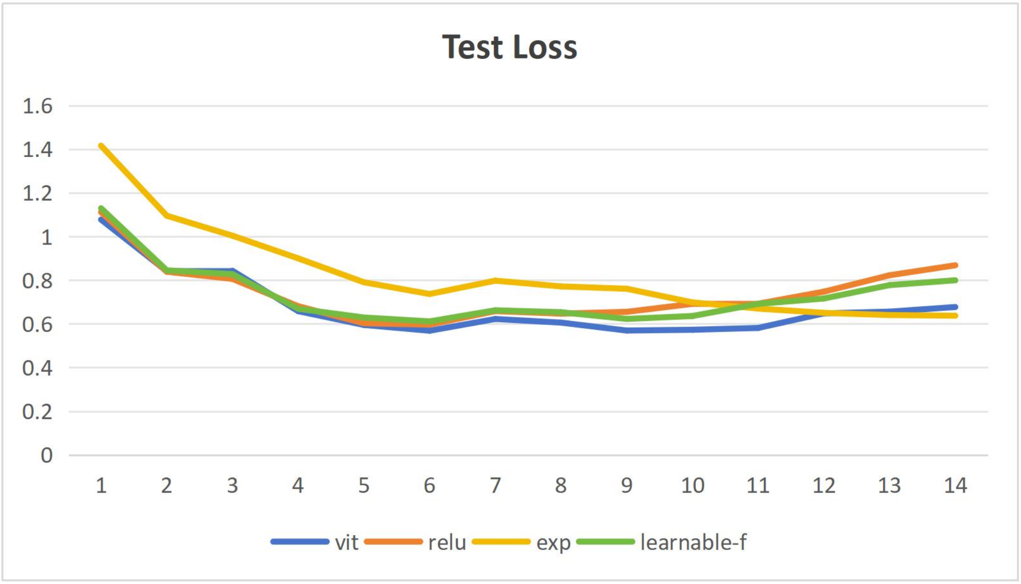

Figure 2 Test loss for Exp II

Experiment III uses the same sets of hyperparameters except for the nb_features. This time I use 32 random features and the results are as follows. When 32 random features were used, VIT still is the best one out of the four architectures in terms of speed and accuracy. And for GPU memory usage, only relu kernel beats the original VIT. The learnable f-variant outperform the other variant in terms of accuracy (which is for sure because it introduces more parameters), but can't reach the speed of relu.

Table 5 Results for Exp III

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	9.25	1.01	83.63	83.63	100.00
Performer-relu	10.38	1.15	80.96	80.96	91.57
Performer-exp	12.68	1.38	80.45	80.45	113.10
Performer-fθ	11.16	1.22	81.65	81.54	105.58

For now, we can have a brief summary of the first three experiments done.

Table 6 Summary for CIFAR Exps

Model	Random Features	Average Train Time	Average Inference Time	Test Accuracy	Resource Used
Performer-exp	16	10.96	1.24	78.40	93.51
	32	12.68	1.38	80.45	113.10
	64	15.29	1.61	79.59	138.13
Performer-fθ	16	9.55	1.11	82.38	80.42
	32	11.16	1.22	81.65	105.58
	64	13.28	1.37	81.65	151.36
Performer-relu	16	9.19	1.08	82.05	72.63
	32	10.38	1.15	80.96	91.57
	64	12.08	1.29	81.32	121.14
VIT		9.25	1.01	83.63	100.00

When we are facing a small dataset, the original VIT consistently outperforms all Performer variants in terms of both accuracy and inference speed. The Performer-exp variant shows lower accuracy across all random feature dimensions compared to both ViT and other Performer variants. The Performer-relu variant performs consistently well, with 82.05% accuracy using 16 random features. It strikes a balance between computational cost and performance. Moreover, relu-kernel beats VIT in terms of average training time. The Performer-fθ achieves competitive accuracy, particularly with 16 random features, where it reaches 82.38%, only slightly behind ViT. This suggests that learnable kernels can approximate attention well for small datasets.

However, as the number of random features increases, the training and inference times grow significantly, and resource usage also scales sharply, highlighting a tradeoff between learnable kernel complexity and computational efficiency.

Experiement IV alter the embedding dim to 200 and increase the MLP dim to 256. Number of random features in the gaussian orthogonal matrix is still 32, and all other arguments remain the same with Experiment I, II, III.

Table 7 Training params for ExpIV

model_type	vit	performer	performer	performer
image_size	32	32	32	32
patch_size	4	4	4	4
num_classes	10	10	10	10
dim	200	200	200	200
depth	6	6	6	6
heads	8	8	8	8
mlp_dim	256	256	256	256
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	64	64	64	64
dropout	0.15	0.15	0.15	0.15
emb_dropout	0.15	0.15	0.15	0.15
batch_size	256	256	256	256
num_epochs	70	70	70	70
learning_rate	0.0015	0.0015	0.0015	0.0015
epoch_step	5	5	5	5
nb_features		32	32	32
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

The results are shown below. The original ViT maintains superior accuracy and computational efficiency, highlighting its suitability for small datasets even with increased embedding and MLP dimensions. However, with increased embedding dim and MLP dimensions, both the average train time and inference time dramatically increased across the four architectures, along with resource usage. Nonetheless, among Performers, Performer-relu stands out as the most efficient variant.

Table 8 Results for Exp IV

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	15.30	1.38	83.67	83.67	100.00
Performer-relu	16.61	1.60	80.6	80.53	88.46
Performer-exp	19.66	1.85	78.62	78.62	114.95
Performer-fθ	17.35	1.65	79.22	79.09	98.20

Experiment V through IX are conducted on TinyImageNet[1] dataset, which consists of larger images (64 × 64 resolution) compared to previous experiments. This dataset provides a more challenging evaluation setting, enabling us to assess the scalability and performance of the four architectures on a relatively larger and more complex dataset.

To comprehensively analyze the impact of key hyperparameters, the experiments were designed with a combination of patch sizes and the number of random features (nb_features) as follows:

Patch Size 4: Tested with nb_features = 16, 32, 64.

Patch Size 8: Tested with nb_features = 32, 64.

This experimental design allows us to evaluate not only the classification performance but also the computational efficiency and resource usage of each architecture under varying conditions, providing deeper insights into their behavior on larger datasets.

Experiment V has the following parameters:

Table 9 Training params for ExpV

model_type	vit	performer	performer	performer
image_size	64	64	64	64
patch_size	4	4	4	4
num_classes	200	200	200	200
dim	256	256	256	256
depth	6	6	6	6
heads	8	8	8	8
mlp_dim	256	256	256	256
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	32	32	32	32
dropout	0.18	0.18	0.18	0.18

[1] https://www.kaggle.com/datasets/akash2sharma/tiny-imagenet

emb_dropout	0.18	0.18	0.18	0.18
batch_size	256	256	256	256
num_epochs	84	84	84	84
learning_rate	0.0015	0.0015	0.0015	0.0015
epoch_step	5	5	5	5
nb_features		16	16	16
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

The results for Experiment V are in . ViT demonstrates significantly higher training (141.36 seconds) and inference times (5.36 seconds) compared to the Performer variants. This is consistent with the expectation that Performers optimize computational efficiency through the use of kernel-based approximations. Among the Performers, Performer-relu achieves the lowest training (82.90 seconds) and inference times (3.79 seconds), followed closely by Performer-fθ. Performer-exp is slightly slower than its counterparts, which may be attributed to the computational overhead of the exponential kernel. In terms of GPU memory utilization, Performer-relu is the most efficient, using only 45.46% of the resources compared to ViT. Performer-f θ and Performer-exp consume slightly more resources (48.24% and 52.20%, respectively), but still remain significantly lower than ViT.

Table 10 Results for Exp V

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	141.36	5.36	35.55	32.78	100.00
Performer-relu	82.90	3.79	31.43	29.49	45.46
Performer-exp	95.64	4.32	33.55	33.06	52.20
Performer-fθ	85.75	3.85	31.80	28.51	48.24

Experiment VI alters nb_features to 32 and maintains all the rest training parameters. The results are shown in Table 11.

Table 11 Results for Exp VI

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	141.36	5.36	35.55	32.78	100.00
Performer-relu	91.80	4.01	31.94	29.70	51.27
Performer-exp	108.39	4.81	33.80	33.31	59.20

Performer-fθ	97.71	4.23	31.61	29.98	56.39

We can see the accuracy for Performer architecture increases along with the training time cost and GPU memory allocated. But the trend remains the same, with relu kernel being the most efficient kernel, exp being the most accurate kernel and learnable fθ lies in between.

Experiment VII further increase the nb_features to 64 while remaining all other training parameters the same, namely patchsize = 4. The results are shown in Table 12.

Table 12 Results for Exp VII

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	141.36	5.36	35.55	32.78	100.00
Performer-relu	103.88	4.44	31.10	28.58	62.07
Performer-exp	127.98	5.66	32.86	32.51	69.96
Performer-fθ	113.44	4.78	31.70	29.84	73.86

ViT continues to exhibit the highest training time (141.36 seconds) and inference time (5.36 seconds), reflecting the computational intensity of full softmax attention. Performer-relu demonstrates the shortest training time (103.88 seconds) and inference time (4.44 seconds) among all models, showcasing the computational efficiency of the ReLU-based kernel. Performer-exp has the longest training time (127.98 seconds) among the Performer variants, approaching ViT’s training time, while Performer-fθ falls in between (113.44 seconds). The exponential kernel's computational cost and the complexity of learnable attention kernels contribute to this overhead.

But when we look at the Resource Used, Performer indeed helped a lot by reducing the memory allocated, thus can train a larger batch at the same time.

For Experiment VIII and IX, I changed the patch size to 8*8. The following table shows the training parameters and results.

Table 13 Training params for Exp VIII

model_type	vit	performer	performer	performer
image_size	64	64	64	64
patch_size	8	8	8	8
num_classes	200	200	200	200
dim	256	256	256	256

depth	6	6	6	6
heads	8	8	8	8
mlp_dim	256	256	256	256
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	32	32	32	32
dropout	0.18	0.18	0.18	0.18
emb_dropout	0.18	0.18	0.18	0.18
batch_size	256	256	256	256
num_epochs	80	80	80	80
learning_rate	0.0015	0.0015	0.0015	0.0015
epoch_step	5	5	5	5
nb_features		32	32	32
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

For Experiment VIII where we used 8*8 patch and 32 random features, VIT is back again and appears to be the most efficient architecture. This is in line with the intuition because the sequence length is shortened from 8*8 to 4*4. However, Performer-exp becomes the most accurate architecture this time. But as a trade off, it is also the slowest to train.

Table 14 Results for Exp VIII

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	24.28	2.04	31.66	30.01	100.00
Performer-relu	26.70	2.04	29.01	26.58	91.96
Performer-exp	31.22	2.07	32.66	31.17	106.19
Performer-fθ	28.20	2.05	28.21	25.78	103.09

Experiement IX just increased the number of random features to 64 based on Experiment VIII.

Table 15 Training params for ExpIX

model_type	vit	performer	performer	performer
image_size	64	64	64	64
patch_size	8	8	8	8
num_classes	200	200	200	200
dim	256	256	256	256
depth	6	6	6	6
heads	8	8	8	8

mlp_dim	256	256	256	256
pool	cls	cls	cls	cls
channels	3	3	3	3
dim_head	32	32	32	32
dropout	0.15	0.15	0.15	0.15
emb_dropout	0.15	0.15	0.15	0.15
batch_size	256	256	256	256
num_epochs	80	80	80	80
learning_rate	0.0016	0.0016	0.0016	0.0016
epoch_step	5	5	5	5
nb_features		64	64	64
generalized_attention		TRUE	FALSE	TRUE
kernel_fn		relu	exp	learnable
no_projection		FALSE	FALSE	FALSE
qkv_bias		TRUE	TRUE	TRUE

The results are quite similar. However, this time VIT is the best model in terms of speed and accuracy, as well as resource usage.

Table 16 Results for ExpIX

Model	Average Train Time	Average Inference Time	Max Test Accuracy	Final Accuracy	Resource Used
VIT	24.25	2.02	32.25	31.05	100.00
Performer-relu	30.03	2.09	28.73	26.86	119.84
Performer-exp	36.38	2.05	33.30	30.66	137.44
Performer-fθ	32.38	2.06	27.85	25.46	143.30

6. Discussion

The experimental results highlight distinct performance trends across ViT and Performer variants, offering insights into their relative strengths and limitations:

6.1. Why Some Methods Perform Better or Worse:

ViT's Accuracy Advantage: ViT achieves the highest accuracy in almost all experimental setups, which can be attributed to its use of the full softmax attention mechanism. This allows it to capture global dependencies with high fidelity, albeit at the cost of computational and memory efficiency.

Performer-exp's Robustness: Performer-exp consistently achieves competitive accuracy, often approaching ViT’s performance. The exponential kernel approximates the softmax function more closely than other kernels, explaining its robustness across different configurations.

Performer-relu’s Efficiency: Performer-relu demonstrates the best computational efficiency in terms of training time and resource usage. However, its relatively simplistic kernel function may fail to approximate complex patterns as effectively as Performer-exp or fθ, resulting in lower accuracy.

Performer-fθ’s Trade-offs: The learnable kernel fθ shows a balance between accuracy and efficiency, offering better results than Performer-relu in most cases but failing to surpass Performer-exp. This may be due to insufficiently optimized kernel design or challenges in training the additional learnable parameters effectively.

6.2. Impact of Dataset Size and Sequence Length:

On smaller datasets like CIFAR10, ViT's overparameterization likely leads to overfitting, explaining its marginal improvement over Performers.

On larger datasets like TinyImageNet, Performers show efficiency benefits as the sequence length increases (e.g., smaller patch sizes), but their accuracy gap with ViT widens due to the approximation limits of kernel-based attention mechanisms.

6.3. Comparison Between Theoretical Predictions and Practical Results

While Performer models are theoretically designed to match or exceed the performance of softmax-based Transformers with improved efficiency, the practical results reveal some notable gaps:

Theoretical Efficiency: The Performer variants theoretically scale better with long sequences due to their linear complexity. This is evident in reduced training and inference times compared to ViT in experiments with smaller patches.

Practical Challenges: The theoretical benefits hinge on the quality of kernel approximations. Performer-exp often performs close to expectations, but Performer-relu and Performer-fθ occasionally underperform due to suboptimal kernel functions or training dynamics.

The results suggest that while kernel-based attention offers substantial efficiency gains, its ability to fully replicate softmax attention's performance remains dataset- and configuration-dependent.

And from my experiments, it can be shown too many random features cannot increase accuracy but will increase time cost to a large extent. And also for large patchsize, by dividing an image into 4*4 blocks actually cannot make full use of the performers.

6.4. Discussion on fθ Design and Future Optimization

Feasibility of Learnable Kernels:

The fθ variant demonstrates promising results, particularly on smaller datasets like CIFAR10, where its performance surpasses Performer-exp in some cases. This indicates that learnable kernels can adapt dynamically to data distributions, offering a flexible alternative to fixed kernels.

However, on larger datasets like TinyImageNet, fθ struggles to consistently outperform Performer-exp, suggesting that the current kernel design might not fully exploit the data's structure or the model's capacity.

Potential for Optimization:

Kernel Function Design: The current fθ implementation uses a simple learnable transformation. Exploring more sophisticated architectures, such as multi-layer perceptrons (MLPs) or attention-inspired parameterizations, could enhance its adaptability and performance.

Training Stability: Introducing regularization techniques or advanced optimizers could address potential issues with gradient explosion or vanishing, particularly for deeper networks or longer sequences.

Dynamic Feature Scaling: Adapting the number of random features dynamically during training might help balance efficiency and accuracy, particularly for datasets with varying complexity.

7. Conclusion & Future Work

This study compared the performance of Vision Transformers (ViT) with Performer variants, including Performer-relu, Performer-exp, and the newly introduced Performer-fθ, on image classification tasks across CIFAR10 and TinyImageNet datasets. The key findings include:

Performance Trade-offs:

ViT consistently achieved the highest accuracy but at the cost of significant computational resources and longer training times.

Performer-relu demonstrated remarkable efficiency in both training time and resource usage, but its accuracy lagged behind due to its simplistic kernel design.

Performer-exp balanced accuracy and efficiency well, proving to be a robust alternative to ViT in many scenarios.

Performer-fθ showcased promising adaptability with learnable kernels, outperforming other Performer variants in certain configurations while maintaining reasonable efficiency.

Efficacy of fθ:

The fθ variant demonstrated the feasibility of integrating learnable kernels into the Performer framework, achieving a trade-off between accuracy and efficiency. However, it fell short of consistently surpassing Performer-exp, indicating room for improvement in its design and implementation.

Future Work

The findings open up multiple avenues for further research and development:

Testing on Larger and More Diverse Datasets:

Future studies could extend experiments to larger datasets like full ImageNet or domain-specific datasets (e.g., medical imaging or remote sensing) to evaluate the scalability and generalization of Performer-fθ.

Advanced fθ Design:

Incorporating more sophisticated architectures for the learnable kernel fθ, such as multi-layer perceptrons (MLPs), convolutional layers, or graph-based representations, could enhance its adaptability and expressive power.