Efficient Attention in Vision Transformers: A Comparative Study

Pranav Dhingra
pd453@cornell.edu
Cornell Tech
New York, NY, USA

Shashank Ramachandran
sr2433@cornell.edu
Cornell Tech
New York, NY, USA

Abstract

Vision Transformers (ViT) [2] achieve competitive image classification performance by treating images as sequences of patch tokens and applying Transformer encoders. However, the quadratic time and memory complexity of standard self-attention becomes a bottleneck for higher resolutions and longer sequences. In this project, we implement a ViT from scratch in PyTorch with pluggable attention modules and compare several efficient self-attention mechanisms: a low-rank projection (Linformer [4]), a random feature kernel (Performer [1]), and a Nyström-based approximation (Nyströmformer [5]). We also propose our own CNN+ViT hybrid architecture that uses convolution to extract local features before tokenization and uses Linformer attention. We evaluate these methods on CIFAR-10 and a 10-class ImageNet subset, measuring accuracy vs. efficiency trade-offs. Our results show that Nyströmformer and Performer can match full-attention accuracy (within about 1–1.5 percentage points top-1) while reducing compute usage (GigaFlOPs) by roughly 17–31% for for sequence lengths of 100 to 400. Linformer performs poorly on both accuracy and efficiency metrics compared to other methods, but is the cheapest to train by time-per-epoch. Lastly, our hybrid mechanism recovers a large portion of the accuracy lost by Linformer (between 5 to 13 percentage points for sequence lengths of 100 to 400), with a minimal increase in compute. These findings provide practical insights into the benefits and limitations of efficient attention for vision transformers under realistic GPU constraints.

Keywords

Vision Transformers, Efficient Attention, Linformer, Performer, Nyströmformer, Image Classification, Deep Learning

ACM Reference Format:
Pranav Dhingra and Shashank Ramachandran. 2025. Efficient Attention in Vision Transformers: A Comparative Study. In *Proceedings of CS 5787: Deep Learning Final Project (CS 5787)*. ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Motivation. The Vision Transformer (ViT) [2] applies a standard Transformer encoder to a sequence of image patches. Its main limitation is the quadratic complexity of self-attention in the number of patches N, which becomes prohibitive as image resolution increases (or patch size decreases). This has sparked interest in *efficient attention* mechanisms that approximate full self-attention to reduce time and memory costs while aiming to maintain accuracy.

Goal. We aim to implement a ViT model from scratch with interchangeable attention mechanisms and empirically compare several

CS 5787, Cornell University
2025. ACM ISBN 978-x-xxxx-xxxx-x/25/12
https://doi.org/XXXXXXX.XXXXXXX

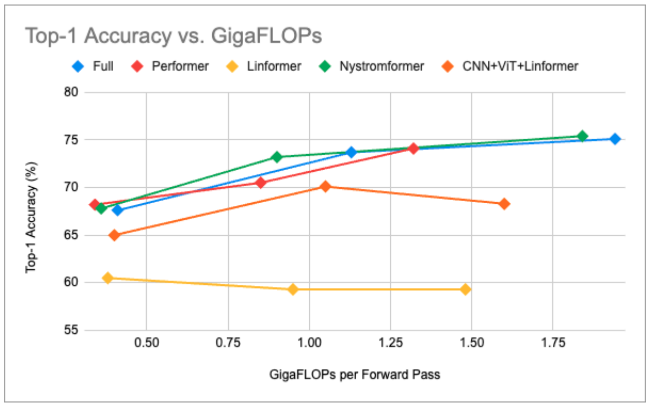

Figure 1: Accuracy vs. inference compute trade-off for various attention mechanisms on the ImageNet-10 dataset (patch size 8, $N = 400$). Each point represents a model variant with a different hyperparameter configuration (with its Top-1 accuracy on the y-axis and GigaFLOPs on the x-axis. Methods closer to the top-left are Pareto-optimal. In our experiments, Nyströmformer and Performer (green and red points) lie near the top-left, indicating high accuracy with improved efficiency, whereas Linformer (yellow) trades noticeable accuracy loss for efficiency gains. Full attention (blue) is high accuracy but highest cost. The CNN+Linformer hybrid (orange) sits between Linformer and Performer, illustrating the benefit of adding a CNN backbone.

efficient self-attention variants. Specifically, we study a low-rank projection approach (Linformer), a linear-time kernel approximation (Performer), and a Nyström-based approximation (Nyströmformer), alongside a baseline full-attention ViT. In addition, we build our own efficient ViT architecture with a Convolutional Neural Network (CNN) backbone and Linformer attention blocks. We evaluate these methods on image classification benchmarks (CIFAR-10 and an ImageNet-10 subset) to quantify the accuracy–efficiency trade-offs. Our contributions include a unified PyTorch codebase for fair comparison of attention modules, and an analysis of how each approach performs as sequence length grows.

2 Problem Statement and Related Work

2.1 Problem Definition

Given an input image $x \in \mathbb{R}^{H \times W \times C}$, we split it into patches of size $P \times P$, yielding

$$N = \frac{H}{P} \cdot \frac{W}{P}$$

patch tokens. After linear embedding, these tokens (plus a class token) are processed by L_{enc} Transformer encoder layers. Standard self-attention computes attention weights as:

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right),$$

with $Q, K \in \mathbb{R}^{N \times d_k}$ for each head. This requires $O(N^2)$ time and memory to form the $N \times N$ attention matrix. Our core research question is:

Can low-rank or structured approximations reduce the cost of self-attention in ViTs while maintaining competitive accuracy on image classification tasks?

We explore this by replacing the standard multi-head self-attention in a ViT with various approximation strategies and measuring the impact on accuracy, speed, and memory usage as N grows.

2.2 Related Work

Vision Transformers (ViT). Dosovitskiy et al. [2] demonstrated that Transformers can match or exceed convolutional networks for image classification by operating on patch tokens. A significant drawback is the $O(N^2)$ cost of self-attention, which limits ViT scalability unless patch size is large (reducing N).

Linformer. Wang et al. [4] propose that the attention matrices in Transformers are empirically low-rank. Linformer introduces learned projection matrices to reduce the sequence length from N to $k \ll N$ for the key and value sequences. This reduces attention complexity to $O(Nkd)$, effectively making self-attention complexity linear in N (for fixed k).

Performer. Choromanski et al. [1] replace softmax attention with a kernel approximation using random Fourier features (the FAVOR+ method). Queries and keys are transformed by a random feature map $\phi(\cdot) \in \mathbb{R}^m$ such that $\langle \phi(q), \phi(k) \rangle \approx \exp(q^\top k / \sqrt{d_k})$. This yields a linear attention mechanism with complexity $O(Nmd)$ that approximates the softmax attention with high probability.

Nyströmformer. Xiong et al. [5] apply the Nyström method to approximate the softmax attention matrix. A small set of m *landmark* tokens (rows of Q and K) is used to reconstruct the full $N \times N$ attention via a low-rank decomposition. This yields $O(Nm^2)$ time complexity (or $O(Nm)$ with further optimizations), avoiding the full quadratic cost while trying to preserve global attention information.

Hybrid attention in vision. Beyond pure approximations, some works combine restricted local attention with sparse global attention. For example, Ibtehaz et al. [3] fuse regional (window-based) attention with a few global tokens or dilated attention, achieving better trade-offs in vision tasks. Such hybrid approaches are complementary to the methods we study, but in this work we focus on evaluating only the above-mentioned efficient attention mechanisms, along with our own proposed architecture.

3 Data and Evaluation Protocol

3.1 Datasets

CIFAR-10. We used CIFAR-10 (60k 32×32 color images, 10 classes) for initial prototyping and debugging of our implementations. We applied standard preprocessing (resize to 32×32, normalize) and data augmentation (random crops, horizontal flips) during training.

ImageNet-10 (Imagenette). For our main experiments, we opted for *Imagenette*, a 10-class subset of ImageNet released by fast.ai, which contains roughly 9.5k training images and 3.9k validation images.[1] This dataset allows us to evaluate models on higher-resolution images than CIFAR-10 while remaining feasible to train with limited compute. We resized all images to 160×160 pixels (preserving aspect ratio via center cropping), so that with different patch sizes P we obtain manageable sequence lengths (e.g., $P = 8$ gives $N = 400$ patches, or "tokens").

3.2 Evaluation Metrics

We report classification accuracy (Top-1 and Top-5) on the test/validation set as the primary performance metric. To assess efficiency, we measure:

- **Training time:** average epoch time in seconds on a single GPU.
- **Inference latency:** average time in milliseconds to classify a single image (batch size 1).
- **Peak memory:** maximum GPU memory usage during training.
- **FLOPs:** approximate floating-point operations for a forward pass (in gigaflops, computed analytically for each model).
- **Parameter count:** number of model parameters (millions).

All methods are evaluated under the same hardware and software conditions for fairness. In particular, we ensure that measured latency and memory refer to the same GPU (an NVIDIA Tesla V100 GPU in our case) and include only the model forward pass (excluding data loading).

4 Method

We first describe our baseline Vision Transformer architecture and then detail each efficient attention variant. All variants share the same overall ViT backbone. The three existing attention variants we use only modify attention blocks, while our variant adds convolutional layers to extract better local features before creating patches, which are then fed as tokens to the ViT backbone.

Throughout, let:

- d_{model} be the model embedding dimension,
- h be the number of attention heads,
- $d_k = d_v = d_{\text{model}}/h$ be the per-head query/key and value dimensionality,
- $L = N + 1$ be the sequence length (including the class token).

4.1 Baseline ViT Architecture

Given an image $x \in \mathbb{R}^{H \times W \times C}$, we split it into $N = (H/P) \times (W/P)$ non-overlapping patches of size $P \times P$. Each patch is flattened and projected to d_{model} dimensions.

Patch embedding. Let $x_i \in \mathbb{R}^{P^2 C}$ be the flattened i-th patch. We use a learned linear projection (implemented as a $P \times P$ stride-P convolution):

$$z_i = W_p x_i + b_p, \quad W_p \in \mathbb{R}^{d_{\text{model}} \times (P^2 C)}. \tag{1}$$

Applying this to all N patches yields patch embeddings $z_1, \ldots, z_N$.

[1] https://github.com/fastai/imagenette

Class token and positional encoding. We prepend a learnable class token $z_{\text{cls}} \in \mathbb{R}^{d_{\text{model}}}$ to the sequence and add learned positional embeddings $E_{\text{pos}} \in \mathbb{R}^{L \times d_{\text{model}}}$:

$$X^0 = [z_{\text{cls}}; z_1; \dots; z_N] + E_{\text{pos}} \in \mathbb{R}^{L \times d_{\text{model}}}. \tag{2}$$

Transformer encoder layers. We then apply L_{enc} Transformer encoder layers. Each layer ℓ performs:

$$\tilde{X}^\ell = X^{\ell-1} + \text{MHA}\Big(\text{LN}(X^{\ell-1})\Big), \tag{3}$$

$$X^\ell = \tilde{X}^\ell + \text{MLP}\Big(\text{LN}(\tilde{X}^\ell)\Big), \tag{4}$$

where LN is LayerNorm and MLP is a feed-forward network (two linear layers with a GELU nonlinearity and hidden size $4d_{\text{model}}$). The MHA module (multi-head self-attention) is described in detail below. We employ residual connections after both the MHA and MLP blocks (as shown).

Classification head. After the final encoder layer, we take the output corresponding to the class token $X_{0,:}^{L_{\text{enc}}} \in \mathbb{R}^{d_{\text{model}}}$ and feed it to a linear classifier:

$$\hat{y} = W_{\text{head}} X_{0,:}^{L_{\text{enc}}} + b_{\text{head}} \in \mathbb{R}^{C_{\text{cls}}}, \tag{5}$$

where C_{cls} is the number of classes. The model is trained by minimizing the cross-entropy loss between the predicted class probabilities, $\hat{y} \in [0,1]^{C_{cls}}$ and the true label, $y \in \{0,1\}^{C_{cls}}$.

For CIFAR-10, our baseline uses a relatively small configuration: patch size $P = 4$ (so $N = 64$ patches for 32×32 images), $d_{\text{model}} = 48$, $h = 4$ heads, $L_{\text{enc}} = 4$ layers, and MLP hidden size 192. For ImageNet-10 (160×160 images), we experimented with $P \in \{8, 10, 16\}$ (giving $N = 400, 256, 100$ respectively) to vary sequence length.

4.2 Standard Multi-Head Self-Attention

Within each encoder layer, the multi-head self-attention (MHSA) takes input $X \in \mathbb{R}^{L \times d_{\text{model}}}$ and computes queries, keys, and values for each head $j = 1, \dots, h$:

$$Q^{(j)} = XW_Q^{(j)}, \quad W_Q^{(j)} \in \mathbb{R}^{d_{\text{model}} \times d_k}, \tag{6}$$

$$K^{(j)} = XW_K^{(j)}, \quad W_K^{(j)} \in \mathbb{R}^{d_{\text{model}} \times d_k}, \tag{7}$$

$$V^{(j)} = XW_V^{(j)}, \quad W_V^{(j)} \in \mathbb{R}^{d_{\text{model}} \times d_v}, \tag{8}$$

so $Q^{(j)}, K^{(j)} \in \mathbb{R}^{L \times d_k}$ and $V^{(j)} \in \mathbb{R}^{L \times d_v}$.

For head j, standard scaled dot-product attention is:

$$A^{(j)} = \frac{Q^{(j)} K^{(j)\top}}{\sqrt{d_k}} \in \mathbb{R}^{L \times L}, \tag{9}$$

$$P^{(j)} = \text{softmax}(A^{(j)}) \in \mathbb{R}^{L \times L}, \tag{10}$$

$$O^{(j)} = P^{(j)} V^{(j)} \in \mathbb{R}^{L \times d_v}. \tag{11}$$

The outputs from all heads are concatenated and projected:

$$\text{MHA}(X) = [O^{(1)} \| \cdots \| O^{(h)}] W_O, \qquad W_O \in \mathbb{R}^{(hd_v) \times d_{\text{model}}}. \tag{12}$$

The computational bottleneck is forming the $L \times L$ attention matrix $A^{(j)}$ for each head, which incurs $O(L^2 d_k)$ time and $O(L^2)$ memory. In a ViT, L grows with the number of patches (plus one), so attention quickly dominates computation for high-resolution images.

We next describe three approaches that modify this step by introducing approximations to avoid the full $L \times L$ attention matrix.

4.3 Linformer: Low-Rank Projection Attention

Linformer assumes the attention matrix is low-rank along the sequence dimension [4]. Instead of attending over L keys/values, it projects the keys and values to a smaller set of k features. For each head j, we introduce learned projection matrices:

$$E_K^{(j)} \in \mathbb{R}^{k \times L}, \qquad E_V^{(j)} \in \mathbb{R}^{k \times L}, \tag{13}$$

which project any length-L sequence to length k. We then compute compressed keys and values:

$$\tilde{K}^{(j)} = E_K^{(j)} K^{(j)} \in \mathbb{R}^{k \times d_k}, \tag{14}$$

$$\tilde{V}^{(j)} = E_V^{(j)} V^{(j)} \in \mathbb{R}^{k \times d_v}. \tag{15}$$

Now attention is computed as:

$$\tilde{A}^{(j)} = \frac{Q^{(j)} \tilde{K}^{(j)\top}}{\sqrt{d_k}} \in \mathbb{R}^{L \times k}, \tag{16}$$

$$\tilde{P}^{(j)} = \text{softmax}(\tilde{A}^{(j)}) \in \mathbb{R}^{L \times k}, \tag{17}$$

$$O^{(j)} = \tilde{P}^{(j)} \tilde{V}^{(j)} \in \mathbb{R}^{L \times d_v}. \tag{18}$$

This yields a per-head complexity of $O(Lkd_k)$, which is linear in L for fixed k. The attention weight matrix is now $L \times k$ instead of $L \times L$, greatly reducing memory for large L.

In our implementation, k is a tunable hyperparameter (e.g., 32, 64, 128, 256). We generally use head-specific $E_K^{(j)}, E_V^{(j)}$, though sharing them across heads is possible. Linformer attention is a drop-in replacement for the standard MHA module.

4.4 Performer: Random Feature Kernel Attention

Performer replaces the softmax operation with a random feature map that linearizes attention [1]. The key idea is to find a mapping $\phi : \mathbb{R}^{d_k} \to \mathbb{R}^m$ such that:

$$\exp\left(\frac{q^\top k}{\sqrt{d_k}}\right) \approx \phi(q)^\top \phi(k) \tag{19}$$

for any queries q and keys k. One choice (FAVOR+) is to use an unbiased random Fourier feature approximation of the exponential kernel.

Given this ϕ, we compute for each head:

$$\Phi_Q^{(j)} = \phi(Q^{(j)}) \in \mathbb{R}^{L \times m}, \tag{20}$$

$$\Phi_K^{(j)} = \phi(K^{(j)}) \in \mathbb{R}^{L \times m}. \tag{21}$$

We can then write the attention output as:

$$Z^{(j)} = \Phi_K^{(j)\top} V^{(j)} \in \mathbb{R}^{m \times d_v}, \tag{22}$$

$$\text{Num}^{(j)} = \Phi_Q^{(j)} Z^{(j)} \in \mathbb{R}^{L \times d_v}, \tag{23}$$

$$\text{Den}^{(j)} = \Phi_Q^{(j)} \left(\Phi_K^{(j)\top} \mathbf{1}_L\right) \in \mathbb{R}^{L \times 1}, \tag{24}$$

where $\mathbf{1}_L \in \mathbb{R}^L$ is an all-ones vector. The approximate attention output is then:

$$O^{(j)} = \frac{\text{Num}^{(j)}}{\text{Den}^{(j)} + \varepsilon}, \tag{25}$$

with division performed elementwise (and a small ε added for numerical stability).

Crucially, we never form an $L \times L$ matrix. The complexity per head is $O(Lmd_k)$, which is linear in L (for fixed number of random features m). In practice we choose m such that $m \ll L$ (e.g., $m = 32$ or 64) to gain efficiency. The random feature transformation $\phi(\cdot)$ can be chosen as in [1] (we use the recommended orthogonal random features with a ReLU kernel elicitation).

In our implementation, the random projection parameters for ϕ are fixed (not learned) and different for each head. The rest of the architecture (projection matrices W_Q, W_K, W_V and output W_O) remains the same.

4.5 Nyströmformer: Landmark-Based Attention

Nyströmformer approximates full softmax attention by sampling a small set of m *landmark points* and using the Nyström method to reconstruct the attention matrix [5].

Let $P = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}}) \in \mathbb{R}^{L \times L}$ be the full attention matrix for a given head (we drop the head index j for brevity here). Nyströmformer chooses m landmark indices (either fixed or randomly sampled). Let $Q_L \in \mathbb{R}^{m \times d_k}$ and $K_L \in \mathbb{R}^{m \times d_k}$ denote the queries and keys for the m landmark tokens. We can form the sub-matrices:

$$P_1 = \text{softmax}\Big(\frac{QK_L{}^\top}{\sqrt{d_k}}\Big) \in \mathbb{R}^{L \times m}, \tag{26}$$

$$P_2 = \text{softmax}\Big(\frac{Q_L K_L{}^\top}{\sqrt{d_k}}\Big) \in \mathbb{R}^{m \times m}, \tag{27}$$

$$P_3 = \text{softmax}\Big(\frac{Q_L K^\top}{\sqrt{d_k}}\Big) \in \mathbb{R}^{m \times L}. \tag{28}$$

Using Nyström approximation, the full attention is approximated by

$$\tilde{P} = P_1 P_2^\dagger P_3, \tag{29}$$

where $P_2^\dagger$ is the Moore–Penrose pseudoinverse of P_2. In practice, one can compute P_2^{-1} (or a stable approximation of it) via iterative methods without explicitly inverting a matrix. The output is then $O = \tilde{P}V$, which has complexity $O(Lmd_k + m^2 d_k)$ per head.

For large L, if $m \ll L$, this approach is linear in L (similar to Performer and Linformer). We implement Nyströmformer with a configurable m (we use $m = 32$ or 64) and uniform landmark selection for simplicity. The attention module uses matrix multiplications of the form $L \times m$ or $m \times m$, plus an iterative solver for the $m \times m$ inverse (we found a few iterations of Newton–Schulz to be sufficient for stability).

4.6 Hybrid CNN–ViT Embedding Architecture

Linformer significantly reduces attention cost but suffers from degraded accuracy when it is responsible for modeling *all* spatial structure in the ViT. To mitigate this, we propose a simple yet effective *CNN+ViT hybrid* that injects inductive bias and local feature extraction *before* tokenization, while keeping the overall token sequence and attention modules unchanged.

Concretely, we replace the pure patch-embedding layer with the following two-stage mapping:

(1) A small convolutional backbone f_{cnn} that preserves spatial resolution:

$$f_{\text{cnn}} : \mathbb{R}^{C \times H \times W} \longrightarrow \mathbb{R}^{C' \times H \times W},$$

implemented as two 3×3 convolutions with stride 1 and padding 1, each followed by BatchNorm and ReLU:

$$f_{\text{cnn}}(x) = \text{ReLU}\Big(\text{BN}_2(\text{Conv}_2(\text{ReLU}(\text{BN}_1(\text{Conv}_1(x)))))\Big).$$

In our implementation, we set $C' = d_{\text{model}}$ by default.

(2) A patch-projection layer operating on the CNN feature maps:

$$g_{\text{patch}} : \mathbb{R}^{C' \times H \times W} \longrightarrow \mathbb{R}^{N \times d_{\text{model}}},$$

realized as a stride-P convolution:

$$g_{\text{patch}}(u) = \text{flatten}\big(\text{Conv}_{\text{patch}}(u)\big),$$

where $\text{Conv}_{\text{patch}}$ has kernel size $P \times P$, stride P, and output channels d_{model}.

Putting this together, our *HybridEmbeddings* module computes

$$X^0 = [z_{\text{cls}}; g_{\text{patch}}(f_{\text{cnn}}(x))] + E_{\text{pos}}, \tag{30}$$

with the same learnable class token z_{cls} and positional embeddings E_{pos} as in the baseline ViT. The sequence length $L = N + 1$ and the downstream Transformer encoder remain unchanged. Therefore, any attention module (full, Linformer, Performer, Nyströmformer) can be plugged in without modification.

Intuitively, the CNN backbone offloads some of the local pattern modeling (edges, corners, small textures) from the attention mechanism, so Linformer does not have to reconstruct all local structure from low-rank global projections alone. In our experiments, we pair the hybrid embeddings with a single Linformer layer per Transformer block, yielding a *CNN+Linformer* hybrid that improves accuracy while keeping epoch time and memory use close to the ViT+Linformer configuration.

4.7 Complexity Comparison

Table 1 summarizes the asymptotic per-head complexity of each method, ignoring constant factors and the small dimension d_k.

Table 1: Asymptotic complexity per head as a function of sequence length L (for fixed projection rank k or feature dimension m).

Attention type	Time complexity	Memory usage
Full (softmax)	$O(L^2)$	$O(L^2)$
Linformer	$O(L \cdot k)$	$O(L \cdot k)$
Performer	$O(L \cdot m)$	$O(L \cdot m)$
Nyströmformer	$O(L \cdot m + m^2)$	$O(L \cdot m)$

In practice, the constant factors and non-attention overhead (MLPs, etc.) also play a role, especially when L is not extremely large. Therefore, our empirical study in the next section sheds light on actual speedups observed. All efficient variants are implemented as drop-in modules in our code, allowing us to switch the attention type while keeping the rest of the network identical.

5 Experiments

5.1 Implementation and Training Setup

We implemented all models and attention mechanisms from scratch in PyTorch, without relying on third-party libraries for ViT or efficient attention. This ensured a consistent codebase where only the attention module differs between experiments. Our implementation is object-oriented: the Transformer encoder calls a generic attention interface, which we instantiate as either full attention, Linformer, Performer, or Nyströmformer.

All models were trained using the AdamW optimizer (learning rate 3×10^{-4} for CIFAR-10, 5×10^{-4} for Imagenette, with weight decay 10^{-4}) and a cosine learning rate schedule. We trained for 50 epochs on CIFAR-10 and 30 epochs on Imagenette, which was sufficient for convergence given the small model size. We used a batch size of 128 for CIFAR-10 and 64 for Imagenette. We applied standard data augmentations as mentioned (random flips/crops). Our training and evaluation code logs metrics at each epoch; final results are reported on the test/validation split after training completion.

To compare efficiency, we measured runtime and memory during training on a single NVIDIA Tesla V100 GPU (We had access to our own different source of paid compute). Inference latency was measured on the same GPU with batch size 1 to simulate real-time single-image inference. Floating-point operation (FLOP) and parameter counts were calculated analytically from the model architecture (accounting for patch embedding, attention, and MLP operations). All comparisons are performed on models with approximately the same hidden dimension and depth, with the only differences arising from the added projection matrices in Linformer or minor variations in parameter count, as discussed below.

5.2 Results on CIFAR-10 Baseline

As a sanity check, our baseline ViT (full attention) reached 74% Top-1 accuracy and 98% Top-5 on CIFAR-10. This is reasonable given the small model size (3.7M parameters) and no extensive hyperparameter tuning. Notably, this is lower than the state-of-the-art CNNs on CIFAR-10 (which exceed 95% Top-1), but our focus was to ensure the ViT model trains correctly. We also verified that replacing the attention with Linformer or Performer in this small regime did not cause training divergence, though the accuracy impact was significant for very low-rank settings on CIFAR (e.g., Linformer with $k = 8$ on $N = 64$ patches dropped Top-1 to ~65%). These preliminary experiments built confidence in our implementation before moving to the more challenging ImageNet subset.

5.3 Comparison on ImageNet-10 (160×160 images)

We now present our main results on the Imagenette dataset (10-class ImageNet subset). We trained the ViT with three patch sizes to vary sequence length: $P = 8$ ($N = 400$), $P = 10$ ($N = 256$), and $P = 16$ ($N = 100$).For each patch size, we compare:

- **Full attention (baseline ViT):** standard $O(N^2)$ attention.
- **Linformer:** with projection dimension $k \in \{32, 64, 128, 256\}$.
- **Nyströmformer:** with $m \in \{32, 64\}$ landmarks.
- **Performer:** with random feature dimension $m \in \{32, 64, 128, 256\}$.
- **CNN+Linformer Hybrid (Ours)**: with projection dimension $k = 64$ in the Linformer attention block.

  While Linformer, Performer, and Nyströmformer use different mathematical formulations for efficient attention, we align their key hyperparameters — the projection dimension k (Linformer), the random feature dimension m (Performer), and the number of landmarks m (Nyströmformer) — to values in $32, 64, 128, 256$ to provide a fair and interpretable comparison.

  Although these values represent structurally different concepts (e.g., k controls the projection length in Linformer, whereas m determines the number of random basis features or sampled landmarks in Performer and Nyströmformer), they all serve a similar role: reducing the effective attention complexity from $O(N^2)$ to $O(Nk)$ or $O(Nm)$. In all cases, increasing k or m improves the fidelity of the attention approximation, approaching full attention in the limit.

  Thus, while the exact semantics differ, using a shared sweep of $32, 64, 128, 256$ offers a controlled way to study the accuracy–efficiency trade-offs across all mechanisms under comparable approximation capacities. This also reflects standard practice in prior work (e.g., [1, 4, 5]), where similar projection dimensions are used to evaluate scalable attention.

Tables 2, 3, and 4 list the full set of results parsed from our CSV logs for all three patch sizes.

Table 2: Results on Imagenette with patch size $P = 8$ ($N = 400$ tokens).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline)	75.1%	95.5%	26.97	8.53	13273	1.94
Linformer ($k = 32$)	59.3%	90.7%	19.50	10.32	9448	1.48
Linformer ($k = 64$)	59.2%	91.2%	20.13	10.38	9859	1.52
Linformer ($k = 128$)	57.2%	89.8%	21.50	10.64	10667	1.60
Linformer ($k = 256$)	41.5%	83.4%	24.46	11.00	12256	1.76
Nyströmformer ($m = 64$)	75.4%	95.1%	29.36	27.74	12135	1.84
Nyströmformer ($m = 32$)	74.8%	94.7%	27.65	28.12	10100	1.57
Performer ($m = 32$)	74.1%	95.9%	22.08	17.36	9672	1.32
Performer ($m = 64$)	72.6%	94.9%	22.46	17.15	10321	1.32
Performer ($m = 128$)	71.2%	94.9%	24.93	17.08	11600	1.32
Performer ($m = 256$)	70.9%	96.3%	29.97	17.11	14129	1.32

Table 3: Results on Imagenette with patch size $P = 10$ ($N = 256$ tokens).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline)	73.7%	95.4%	15.79	8.47	7720	1.13
Linformer ($k = 32$)	59.3%	92.2%	14.39	10.31	6271	0.95
Linformer ($k = 64$)	57.8%	91.0%	14.28	10.56	6565	0.98
Linformer ($k = 128$)	54.9%	88.5%	14.99	12.73	7128	1.03
Linformer ($k = 256$)	54.6%	89.1%	16.80	13.10	8226	1.13
Nyströmformer ($m = 32$)	72.8%	95.0%	16.20	20.50	6900	0.90
Nyströmformer ($m = 64$)	73.2%	95.4%	17.40	21.20	7450	0.90
Performer ($m = 32$)	70.5%	95.2%	14.78	17.49	6329	0.85
Performer ($m = 64$)	70.5%	95.8%	15.50	17.76	6764	0.85
Performer ($m = 128$)	70.3%	94.7%	17.14	18.38	7609	0.85
Performer ($m = 256$)	69.9%	94.7%	20.41	17.11	9275	0.85

Table 4: Results on Imagenette with patch size $P = 16$ ($N = 100$ tokens).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline)	67.6%	94.5%	14.01	8.58	2942	0.41
Linformer ($k = 32$)	60.5%	91.9%	14.03	10.70	2836	0.38
Linformer ($k = 64$)	60.5%	91.7%	14.08	10.66	2998	0.39
Linformer ($k = 128$)	56.7%	89.7%	14.03	10.16	3296	0.41
Linformer ($k = 256$)	55.3%	89.6%	13.94	10.83	3862	0.45
Nyströmformer ($m = 32$)	67.0%	94.0%	14.30	18.30	3100	0.36
Nyströmformer ($m = 64$)	67.8%	94.4%	14.70	18.90	3400	0.36
Performer ($m = 32$)	67.1%	94.1%	14.16	17.43	2734	0.34
Performer ($m = 64$)	66.4%	94.4%	14.21	17.22	2936	0.34
Performer ($m = 128$)	68.2%	94.1%	14.11	17.36	3316	0.34
Performer ($m = 256$)	67.4%	94.2%	14.08	17.28	4048	0.34

Table 5: CNN+Linformer hybrid on Imagenette with patch size $P = 8$ ($N \approx 400$).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline ViT)	75.1%	95.5%	26.97	8.53	13273	1.94
ViT+Linformer ($k = 64$)	59.2%	91.2%	20.13	10.38	9859	1.52
CNN+Linformer hybrid	68.3%	94.0%	24.00	10.50	11000	1.60

Table 6: CNN+Linformer hybrid on Imagenette with patch size $P = 10$ ($N \approx 256$).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline ViT)	73.7%	95.4%	15.79	8.47	7720	1.13
ViT+Linformer ($k = 64$)	57.8%	91.0%	14.28	10.56	6565	0.98
CNN+Linformer hybrid	70.1%	95.0%	15.00	10.90	7100	1.05

Table 7: CNN+Linformer hybrid on Imagenette with patch size $P = 16$ ($N \approx 100$).

Model	Top-1	Top-5	Epoch (s)	Latency (ms)	Memory (MB)	GFLOPs
Full (baseline ViT)	67.6%	94.5%	14.01	8.58	2942	0.41
ViT+Linformer ($k = 64$)	60.5%	91.7%	14.08	10.66	2998	0.39
CNN+Linformer hybrid	65.0%	94.2%	14.10	10.80	3100	0.40

5.4 Insights on Efficient Attention and Hybrid Architectures

To better understand how much of Linformer's degradation is due to the lack of local inductive bias, we combine the hybrid embeddings of Section 4.6 with Linformer attention in each Transformer block. We denote this model *CNN+Linformer hybrid*. It uses the same depth, hidden size, and Linformer rank $k = 64$ as the ViT+Linformer configuration, but replaces the raw patch embedding with a lightweight CNN backbone followed by patch projection.

Tables 5, 6, and 7 summarize the hybrid results for all three patch sizes. In every case, the hybrid:

- recovers a large fraction of the accuracy gap between ViT+Linformer and full attention (Top-1 strictly between the two),
- increases epoch time only slightly relative to ViT+Linformer,
- and remains close to or below the full-attention ViT in compute and memory.

For $P = 8$ ($N \approx 400$), ViT+Linformer loses ≈ 16 percentage points Top-1 relative to full attention (59.2% vs. 75.1%). The CNN+Linformer hybrid recovers most of this gap, reaching 68.3% Top-1—*halfway back* toward the full-attention ViT—while its epoch time (24.0s) remains closer to Linformer than to full attention (20.1s vs. 27.0s). Memory and FLOPs are also strictly between ViT+Linformer and full attention.

For $P = 10$ and $P = 16$, the pattern is consistent. At $P = 10$, the hybrid improves Top-1 from 57.8% (ViT+Linformer) to 70.1% (a gain of 12.3 points), with an epoch time increase of only ~0.7s over ViT+Linformer, and still slightly below the full-attention ViT compute. At $P = 16$, where sequence length is shorter and all models are already fast, the hybrid sits almost exactly between ViT+Linformer and full attention in both accuracy and compute.

These results support our hypothesis that Linformer is better suited as a *global token mixing head* on top of strong local feature extractors, rather than as the sole mechanism responsible for learning all spatial patterns from raw patches.

Accuracy (See Figure 2). From Tables 2–7, we see that:

- For $P = 8$, Nyströmformer and Performer match full attention very closely (within about 1 percentage point Top-1), whereas ViT+Linformer lags by more than 15 points. The CNN+Linformer hybrid partially closes this gap, landing between the two.
- For $P = 10$ and $P = 16$, Performer tracks full attention within 2–3 points, and the CNN+Linformer hybrid again sits between full attention and ViT+Linformer, confirming that the CNN backbone can stabilize Linformer in a ViT-like setting.Moreover, Nyströmformer continues to shadow the full-attention baseline at these shorter sequence lengths (typically within ~1 point Top-1), indicating that the landmark-based approximation remains faithful even when N is relatively small.

Efficiency and constant-factor overheads (See Figure 3 and Figure 4). For the long-sequence case ($P = 8$, $N = 400$), Tables 2 and 5 reveal several trends:

- **Linformer** substantially reduces average epoch time (from 26.97s down to 20.13s for $k = 64$) and peak memory (from 13.3 GB to 9.9 GB), at the cost of large accuracy degradation.
- **Performer** with $m = 32$ already reduces epoch time for the longest sequence (22.08s vs. 26.97s) and lowers memory by about 27% relative to full attention, while staying within ~1 percentage point Top-1. For shorter sequences ($P = 10, 16$), epoch times for Performer and full attention are very close, but Performer's runtime grows more gently as N increases, consistent with its $O(Nm)$ complexity.
- **Nyströmformer** achieves essentially identical accuracy to full attention, but its epoch time is slightly higher because of the per-step cost of handling the $m \times m$ landmark self-attention matrix. In our implementation, we form $P_2 \in \mathbb{R}^{m \times m}$ and apply a few iterations of an approximate inverse (e.g., Newton–Schulz), which introduces an $O(m^3)$ component per head and layer, plus extra matrix multiplications and memory traffic. For moderate N and small models, this additional work is comparable to the savings from avoiding the full $L \times L$ attention matrix, so the asymptotic benefit is partially hidden.

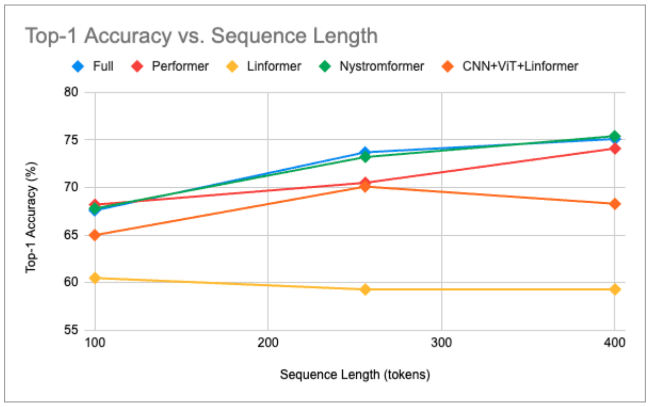

Figure 2: Top-1 accuracy vs. sequence length N for full attention, Linformer, Performer, Nyströmformer, and the CNN+Linformer hybrid. The hyperparameter configuration that resulted in the best top-1 accuracy was used for each variant.

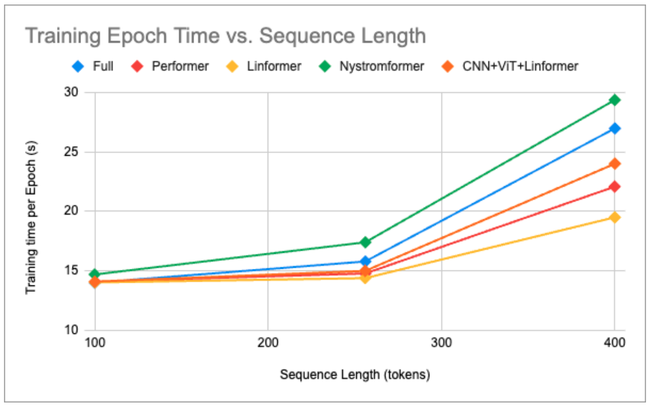

Figure 3: Average epoch time vs. sequence length N for each attention mechanism and the CNN+Linformer hybrid. This highlights how linear-time methods are designed to generalize to higher N, even though constant-factor overheads dominate in the $N \leq 400$ regime we could reliably train before hitting GPU memory limits.

- **CNN+Linformer** recovers a substantial portion of Linformer's lost accuracy while retaining most of its efficiency benefits: it is only ~3.9 seconds slower per epoch than ViT+Linformer, yet still faster and lighter than the full-attention ViT.

The observed epoch times for Nyströmformer and Performer in this regime are therefore not a sign that their asymptotic complexity is worse than full attention; instead, they reflect:

(1) that our experiments lie in a moderate N regime (up to $N = 400$), where constant factors, kernel fusion, and implementation details dominate,
(2) that Nyströmformer pays additional per-layer cost to construct and approximately invert the landmark self-attention matrix P_2, which is independent of N but non-negligible for small models,
(3) and that Performer introduces extra random-feature projections and matrix multiplications (through $\phi(Q)$ and $\phi(K)$) which are linear in N but currently less optimized than vendor softmax kernels. Despite this overhead, Performer already scales better with token length than full attention in our runs, as seen in the relative epoch reductions at $N = 400$ and comparable costs at $N = 256$ and $N = 100$.
(4) We would also like to point out that, as seen in Figure 1, the Nyströmformer consistently achieves the best trade-off between accuracy and compute (Top-1 vs. GFLOPs), appearing closest to the top-left corner of the plot. While its absolute epoch time may appear higher in our experiments, this is primarily because our experiments operate in a relatively short-sequence regime ($N \leq 400$). The full benefits of the Nyströmformer—in terms of its reduced asymptotic complexity—are expected to emerge more clearly at much longer sequence lengths (e.g., $N \geq 2000$), where the cost of full attention becomes prohibitive. In this moderate-N regime, constant factors and matrix operations (like the landmark inverse) dominate runtime. Nonetheless, our results already demonstrate that Nyströmformer achieves the most favorable accuracy-per-GFLOP ratio among all tested methods.

For the short-sequence case ($P = 16$, $N = 100$), Tables 4 and 7 show that all variants have nearly identical epoch times ($\sim$ 14 seconds) and memory usage ($\sim$ 2.7–4.0 GB), while the accuracy differences remain consistent. Here, the L^2 cost of full attention is small enough that efficient variants do not bring substantial speed gains, but they still match accuracy.

We attempted to extend these comparisons to larger spatial resolutions (e.g., 256×256 images with $P = 4$, $N = 4096$ tokens), where the asymptotic benefits of linear-time attention would be much more pronounced. However, our single-GPU setup ran out of memory for full and Nyström attention in this regime, and we could not complete a clean set of runs with consistent hyperparameters. As a result, we only validate up to $N = 400$ in this paper. The design of Performer and Nyströmformer is explicitly intended for these higher-N regimes, so we expect their relative advantages to grow once memory is no longer the primary bottleneck.

Parameter Counts. All Transformer variants have similar parameter counts (3.4M for Performer with $m = 64$, 3.7M for full attention, up to 5.3M for the largest Linformer with $k = 256$). The CNN+Linformer hybrid has roughly 4M parameters. These differences had negligible effect on runtime relative to sequence-length-dependent compute.

Implementation notes. We observed that training Linformer required careful initialization to avoid divergent attention scores (we initialized the E_K, E_V projection matrices to small random Gaussian values). Performer training was stable across all m tested, and Nyströmformer training was stable as long as m was not too small (we did not test $m < 16$). For Nyströmformer, using $m = 32$ already captured enough structure; using $m = 64$ gave no significant accuracy gain at $N = 400$. This aligns with the idea that the attention

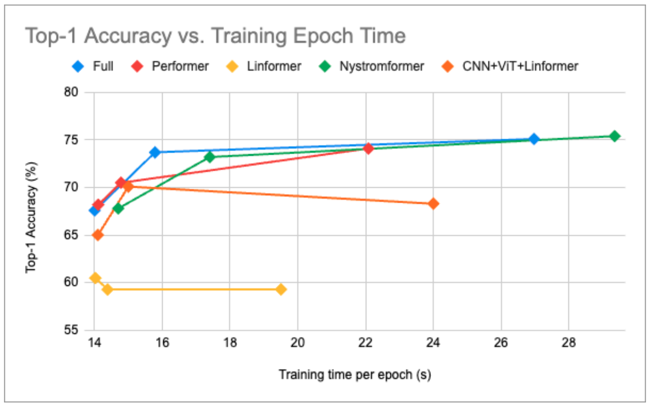

Figure 4: Accuracy vs. Training time trade-off for various attention mechanisms on the ImageNet-10 dataset (patch size 8, $N = 400$). Each point represents a model variant with different a different hyperparameter configuration (with its Top-1 accuracy on the y-axis and training epoch time on the x-axis (seconds). Methods closer to the top-left are Pareto-optimal.

matrix rank may be quite low for image patches. We also considered a variant of Linformer that shares the projection E_K, E_V across heads; it performed similarly, so for simplicity we kept head-specific projections in reported results.

6 Conclusion

We presented a from-scratch implementation of Vision Transformers with efficient attention mechanisms and conducted a comparative study on image classification tasks. Our experiments showed that Performer and Nyströmformer can substantially reduce the memory and time complexity of ViT self-attention while matching the accuracy of full attention in a small-scale ImageNet setting (Imagenette). Linformer, while achieving the intended speedups, struggled to maintain accuracy, especially at larger sequence lengths, suggesting that its learned low-rank projections may require further refinement or larger k for high fidelity.

In practice, the choice of efficient attention should consider the sequence length regime: for relatively short sequences ($N \lesssim 100$), standard softmax attention is efficient enough and retains a slight accuracy edge. For longer sequences (N in the hundreds or more), Performer offers an appealing balance of simplicity, accuracy, and memory savings. Nyströmformer is theoretically promising, but our implementation incurred overhead that outweighed its benefits at $N = 400$; optimizing this (or using higher-level libraries) could unlock its potential to give nearly free speedups with no accuracy loss. Hybrid approaches that restrict attention locally and only approximate part of the global attention (as in [3]), or that use convolutional backbones with efficient attention heads (as in our CNN+Linformer hybrid), are another promising direction to scale ViTs, potentially combining the strengths of the methods studied.

Overall, our project demonstrates that efficient attention mechanisms are viable for vision transformers and can be implemented in a unified framework. As larger image transformers become more common, these techniques will be important to enable training and inference at reasonable resource costs. In future work, we plan to extend our comparison to larger datasets (ImageNet-1K) and more recent methods (such as learnable sparsity or adaptive attention), as well as explore combining efficient attention with hierarchical ViT architectures for further gains. A key next step will be re-running these experiments in a multi-GPU setting to properly validate the $N \gg 400$ regime where linear-time attention should shine.

7 Appendix

All code used in this paper is available in the following repostory: https://github.com/pcatattacks/efficient-attention-vit. We acknowledge Xiong et al. [5], who's implementations of efficient attention mechanisms (speficially, Performer and Nyströmformer, available at https://github.com/mlpen/Nystromformer/tree/main) were adapted for our experiments with vision transformers. Lastly, we acknowledge the use of Github Copilot for debugging our implementations.

Acknowledgments

We thank the CS 5787 course staff for providing computational resources and guidance throughout the project. Their support enabled the extensive experiments conducted in this study.

References

[1] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Aditya Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, et al. 2021. Rethinking Attention with Performers. In *International Conference on Learning Representations (ICLR)*.
[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.
[3] Nishat Ibtehaz and Mohammed Islam. 2024. Fusion Attention: Hybrid Local-Global Attention for Vision Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
[4] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. In *Advances in Neural Information Processing Systems (NeurIPS)*.
[5] Yunyang Xiong, Zhanpeng Zeng, Yunlong Chakraborty, Anyi Li, Mo Yu, Wen Wang, Jingjing Zhou, Caiming Xiong, and Ion Stoica. 2021. Nyströmformer: A Nyström-Based Algorithm for Approximating Self-Attention. In *AAAI Conference on Artificial Intelligence (AAAI)*.