IEOR 4540 Final Project: Linear-Attention of Linformers & Performers Architectures

Yige Yang, Tiantian Chen, Tong Chen, Yuyang Xu, Jiewan Yang, Ningzhou Gu

December 11, 2023

Abstract

This study provides a comprehensive comparison of various Transformer architectures, specifically focusing on regular Transformers, Performers, and Linformers, for small image classification tasks using the CIFAR-10 dataset. By changing self-attention of Tranformer to linear-attention of Linformer and Transformer, we found they have different performance in speed (Transformer has time complexity $O(n^2)$, while Linformer and Performer has $O(n)$. We explored different variants of the Performer architecture, including those with positive random features and a deterministic mechanism with ReLU and EXP nonlinearities. Two training strategies, starting from scratch and from pre-trained checkpoints, are employed for both Performers and Linformers. The analysis includes speed tests for all trained models, providing a holistic evaluation of their performance.

The regular Transformer attains a testing accuracy of 20.65% with an epoch training time of around 935 seconds. Linformers show a significant 50% reduction in training time, coupled with modest accuracy gains. Leveraging pre-trained checkpoints enhances efficiency and maintains or improves accuracy across architectures.

Performers achieve substantial accuracy enhancements, displaying training efficiency and consistent processing speed. Variants, including ReLU and EXP nonlinearities, demonstrate impressive learning aptitude. This study offers practitioners valuable insights into the trade-offs and strengths of each architecture for small image classification tasks, aiding informed model selection. Additionally, Linformers are highlighted as more memory- and time-efficient alternatives, contributing to the broader landscape of Transformer models.

1 Introduction

The transformer architecture was originally designed for language processing tasks and it has demonstrated remarkable versatility and applicability across various domains, including computer vision. Our study focuses on unraveling the nuanced performance of distinct variants of Transformer models, specifically focusing on Performers architectures, and comparing their performance against regular Transformers and Linformers architectures on small image classification tasks.

The Performers' architecture introduces innovations that address the computational complexities associated with self-attention mechanisms. We scrutinize two pivotal variants of Performers: the deployment of positive random features, and a deterministic mechanism featuring ReLU and EXP nonlinearities. These variants are designed with computational efficiency and are set to undergo a rigorous examination to unveil their efficacy in image classification tasks. To have a more comprehensive analysis, we evaluate the variants with regular Transformers and Linformers. Linformers represent a distinct class of models that deviate from the conventional Transformer structure. Linformers leverage a linear attention mechanism, demonstrating potential advantages in terms of parameter efficiency and training dynamics.

Our study aims to conduct a meticulous investigation and comparative analysis of different Transformer architectures, specifically focusing on small-scale image classification tasks. The dataset CIFAR-10 serves as a benchmark for evaluating the efficacy of different models.

The training strategies we adopted and utilized in our study for both Performer and Linformers give a nuanced dimension. We explored two distinct approaches: training from scratch, and leveraging pre-trained checkpoints derived from conventional Transformer training. By doing this, our goal is to generate insights their convergence dynamics, accuracy landscapes, and efficiency trade-offs. In aspects of transparency and reproducibility, we recorded the key design decisions, spanning the number of heads, query-key dimensions per head, and other pertinent parameters. Besides, our study extended to the realm of computational efficiency as speed tests were also conducted to offer a comprehensive evaluation of each architecture.

For this study, we aim to contribute nuanced perspectives to the discussion about Transformer-based architectures in computer vision and provide a deeper understanding of the intricacies that underpin these models to help with decision-making in navigating the complex issues of small image classification tasks.

2 Data Preprocessing

2.1 Data Source

The CIFAR-10 dataset is a widely used benchmark in the field of computer vision, comprising a collection of 60,000 32x32 color images distributed across 10 distinct classes. Each image is labeled with one of the following categories: 'plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', and 'truck'. The dataset is designed for multi-class object recognition tasks and is used to evaluate the performance of machine learning models. Each image in CIFAR-10 is categorized by its RGB color representation with three channels capturing the red, green, and blue intensities.

2.2 Data Preprocessing

To prepare the data for analysis, we did preprocessing initially. We resized the image and normalized it for the RGB image. Subsequent to preprocessing, the dataset is divided into training and testing sets. The sample size is 50,000 for the training dataset and 10,000 for the test dataset. The data preprocessing ensures a standardized and coherent input space, fostering fair evaluations of model performance on the CIFAR-10 dataset.

Refer to Figure 1 and Figure 2 for the first 10 images in training and test datasets.

Figure 1: First 10 images in Training Dataset

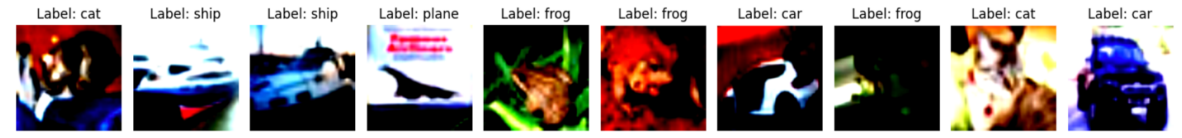

Figure 2: First 10 images in Test Dataset

3 Methodology

3.1 Transformer

The transformer model represents a groundbreaking advancement in the field of machine learning, specifically designed to process sequential data with unparalleled efficiency. Introduced in the paper "Attention is All You Need" by Vaswani et al., the transformer architecture has since become a cornerstone in various natural language processing tasks and has been adapted to other domains.

Unlike traditional sequential models, transformers rely on self-attention mechanisms to weigh the importance of different parts of the input sequence, allowing them to capture long-range dependencies effectively. This innovative approach not only enhances performance but also facilitates parallelization, leading to faster training times.

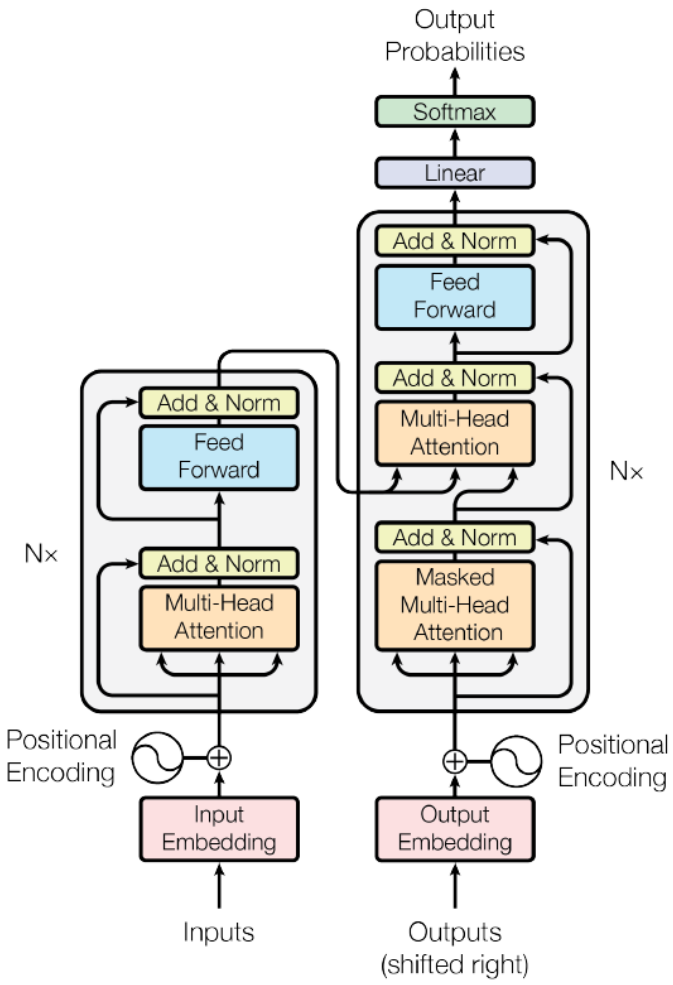

Figure 3: Transformer Model Architecture

The transformer's modular design consists of an encoder-decoder structure, each comprising multiple layers. The encoder processes the input data, while the decoder generates the output. Each layer contains multi-head self-attention mechanisms and feedforward neural

networks, enabling the model to capture intricate patterns and relationships within the data.

One of the key strengths of transformers is their ability to handle variable-length input sequences, making them well-suited for tasks such as machine translation, text summarization, and sentiment analysis. The attention mechanism enables the model to focus on relevant information, contributing to its robustness in understanding context and nuances.

Since its introduction, the transformer model has transcended language-related tasks and found applications in diverse domains, including computer vision (e.g., Vision Transformer or ViT) and beyond. Its versatility, efficiency, and scalability have positioned transformers as a foundational technology in modern machine learning, influencing the development of subsequent models and architectures.

The Vision Transformer (ViT) is a powerful architecture that applies the transformer model originally designed for natural language processing to computer vision tasks. Its mathematical foundation stems from the transformer model originally designed for natural language processing. This part provides a detailed methodology for the implementation of ViT, highlighting key components and their roles.

3.1.1 Patch Embedding

The Patch Embedding process involves breaking down the input image into smaller patches and creating embeddings for each patch. This step aims to capture local information from different parts of the image.

Methodology:

1. Divide the input image into patches of a specified size.
2. Apply a convolutional layer to each patch, transforming them into embeddings.
3. Add positional embeddings to the patch embeddings to provide spatial information.

Mathematically, given an input image I of size $H \times W \times C$ (height, width, channels), the patch embedding can be expressed as:

$$X_i = \text{Linear}\left(\text{Flatten}(\text{Conv}(P_i))\right)$$

where P_i is the i-th patch extracted from the image, Conv represents the convolution operation applied to each patch, Flatten reshapes the patch, and Linear applies a linear transformation.

3.1.2 Multi-Head Self-Attention

Multi-Head Self-Attention is a crucial component that allows the model to focus on different parts of the input sequence, facilitating the capture of long-range dependencies.

The time complexity of the transformer model can be understood through the self-attention mechanism. In a standard transformer, the self-attention mechanism computes the attention

scores for each position in the input sequence concerning all other positions. Let n be the sequence length.

The self-attention mechanism involves the computation of attention scores using a dot product between the query, key, and value vectors. This operation requires $O(n)$ time for each position, and since there are n positions, the overall time complexity is $O(n^2)$.

Therefore, the quadratic time complexity arises from the pairwise comparisons between all positions in the sequence, making transformers computationally expensive for long input sequences.

Methodology:

1. Projecting the input embeddings into query, key, and value vectors.
2. Calculating attention scores between different positions in the sequence.
3. Combining information from different heads to obtain a weighted representation.

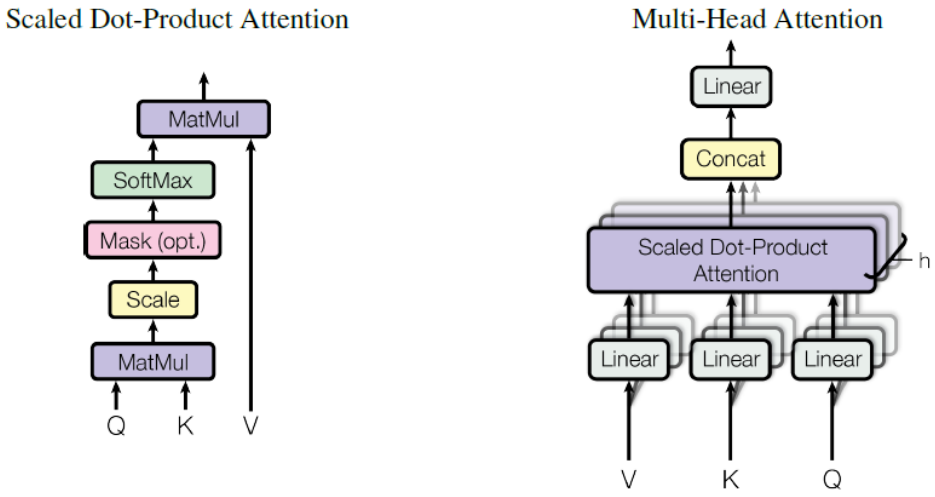

Figure 4: Multi-Head Self-Attention

Given an input sequence X of embeddings, the attention mechanism computes new embeddings using the following formulas:

$$Q_i = XW_Q \quad K_i = XW_K \quad V_i = XW_V$$

where W_Q, W_K, and W_V are learnable weight matrices. The attention scores ($\text{Attention}(Q_i, K_j)$) are calculated, normalized, and used to weigh the values (V_j), resulting in the attended output:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where d_k is the dimension of the key vectors.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W_O$$

where $\text{head}_i = \text{Attention}(QW_Q, KW_K, VW_V)$

3.1.3 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron processes information obtained from the attention mechanism, enabling the model to capture complex patterns and relationships.

Methodology:

1. Applying Layer Normalization to the input.
2. Passing the normalized input through a fully connected layer with GELU activation.
3. Introducing dropout for regularization.

For each position i, the output of the MLP is given by:

$$\text{MLP}(X_i) = \text{Linear}_2(\text{GELU}(\text{Linear}_1(X_i)))$$

where Linear_1 and Linear_2 are linear transformations, and GELU is the Gaussian Error Linear Unit activation function.

3.1.4 Encoder

The Encoder consists of the Multi-Head Self-Attention mechanism and the Multi-Layer Perceptron. It processes and refines the information obtained from the input.

Methodology:

1. Applying Multi-Head Self-Attention to capture relationships within the sequence.
2. Adding a skip connection and normalization.
3. Passing the output through an MLP for further feature extraction.

The output of the encoder for position i is given by:

$$\text{Encoder}(X_i) = X_i + \text{MLP}(\text{Multi-Head Attention}(X))$$

This introduces a skip connection, allowing the model to capture both local and global information.

3.1.5 Vision Transformer (ViT)

The Vision Transformer combines all the aforementioned components to create a holistic model for computer vision tasks.

Methodology:

1. Utilizing the Patch Embedding layer to transform images into manageable patches.
2. Employing multiple Encoder blocks to process and refine the information.
3. Concluding with a classification head that utilizes the token corresponding to the class.

The final classification head utilizes the token corresponding to the class:

$$\text{ViT}(I) = \text{Linear}_3(\text{Encoder}(\text{Patch Embedding}(I)[0]))$$

where Linear_3 is the final linear transformation for classification.

3.2 Linformer

Linformer is an efficient variant of self-attention with low-rank approximation. The linear-time approximation is achieved by projecting the original key and value layers into a lower-dimensional space, significantly reducing the computational cost. Therefore, Linformer's time complexity becomes $O(n)$, making it more efficient for long input sequences.

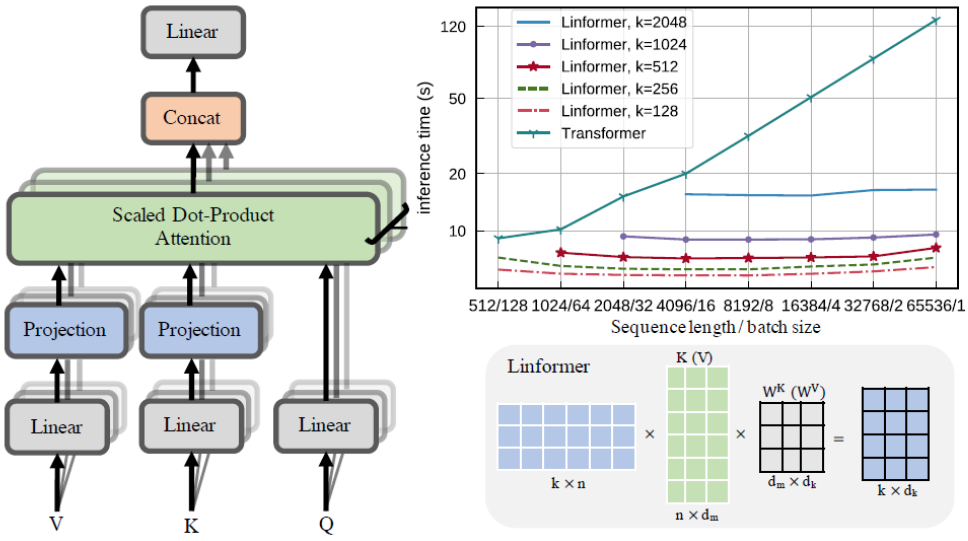

Figure 5: Linformer Model Architecture

Notably, Linformer uses a low-rank approximation for the key and value projections in the attention mechanism. Instead of using full-rank matrices, Linformer employs linear projections with reduced dimensions (parameter k). This approximation significantly reduces the computational complexity and memory requirements for handling long sequences.

Unlike traditional multi-head attention, where linear projections are applied to the query (Q), Linformer applies linear projections to both the key (K) and value (V) tensors. This approach enhances the computational efficiency of the attention mechanism.

3.2.1 Linformer Attention

In the Linformer's multi-head attention mechanism, the key distinction lies in the efficient handling of long sequences through low-rank approximation, making Linformer more scalable for sequence lengths. Linformer introduces an efficient self-attention mechanism that reduces the time complexity to linear, $O(n)$, by approximating self-attention in a novel way.

Methodology:

1. Transforming the input tensor X with a linear transformation to obtain Q, K and V.
2. Projecting K and V to a lower-dimensional space k using linear projections.
3. Computing attention scores scaled by the square root of the head dimension, followed by a softmax operation.
4. Applying masking and dropout to the attention scores if a mask is provided.
5. Calculating the weighted sum of V based on attention scores and projecting the result back to the original dimension.
6. Combining the above steps in the Multi-Head Linformer Attention block.

The main idea of our proposed linear self-attention is to add two linear projection matrices $E_i, F_i \in \mathbb{R}^{n\times k}$ when computing key and value. We first project the original $(n\times d)$-dimensional key and value layers KW_{iK} and VW_{iV} into $(k\times d)$-dimensional projected key and value layers. We then compute an $(n\times k)$-dimensional context mapping matrix $\mathbf{P}$ using scaled dot-product attention.

$$\begin{aligned}\text{head}_i &= \text{Attention}(QW_{iQ}, E_iKW_{iK}, F_iVW_{iV}) \\ &= \text{softmax}\left(\frac{QW_{iQ}(E_iKW_{iK})^T}{\sqrt{d_k}}\right)F_iVW_{iV}\end{aligned}$$

Finally, we compute context embeddings for each head i using $\mathbf{P}\odot(F_iVW_{iV})$. Note the above operations only require $O(nk)$ time and space complexity. Thus, if we can choose a very small projected dimension k, such that $k \ll n$, then we can significantly reduce the memory and space consumption. The 'Linear self-attention' theorem states that when $k = O\left(\frac{d}{\varepsilon^2}\right)$ (independent of n), one can approximate $P\odot VW_{iV}$ using linear self-attention with ε error.

In brief, the Multi-Head Linformer Attention block can be represented mathematically as:

$$X = X + \text{Dropout}(\text{Linear}_{\text{proj}}(\text{Attention}(\text{LayerNorm}(X), \text{mask})))$$

where $\text{Linear}_{\text{proj}}$ represents the linear projection for the final output.

In the standard self-attention of transformers, the time complexity is $O(n^2)$. Linformer addresses this issue by incorporating linear projections for keys and values. The key idea is to use linear projections with matrices E_i and F_i when computing the key and value. This allows the self-attention to be computed in linear time.

The linear-time approximation is achieved by projecting the original key and value layers into a lower-dimensional space, significantly reducing the computational cost. Therefore, Linformer's time complexity becomes $O(n)$.

3.2.2 Linformer Encoder

The Linformer Encoder combines the Attention and MLP blocks, providing a refined representation of the input.

Methodology:

1. Applying the Attention block to the input tensor X within the Linformer Encoder.
2. Adding the result back to the original input.
3. Applying a multi-layer perceptron (MLP) block consisting of layer normalization, a feedforward neural network, and dropout.
4. Combining the Attention and MLP blocks for a refined representation.

The Linformer Encoder block can be expressed as:

$$X = X + \text{Dropout}(\text{MLP}(\text{LayerNorm}(\text{Attention}(X, \text{mask}))))$$

3.2.3 The ViL Model

The LinViT model combines Linformer attention with the Vision Transformer architecture.

Methodology:

1. Transforming input images into patches using the Patch Embedding layer.
2. Applying multiple Linformer Encoder blocks to hierarchically process and refine the information.
3. Utilizing layer normalization in the final output and employing a classification head that uses the token corresponding to the class.

The final classification head can be expressed as follows:

$$\text{LinViT}(I) = \text{Linear}(\text{LayerNorm}(\text{Encoder}(\text{Patch Embedding}(I)[0])))$$

where Linear represents the final linear transformation for classification.

3.3 Performer

The Performer is specifically engineered to optimize the attention mechanism by transforming its complexity from quadratic $O(n^2)$ to linear $O(n)$. This modification ensures that both the time and space complexity of the Performer are linear. As a variant of the Transformer, which has profoundly impacted the realm of deep learning, the Performer retains the model's powerful capacity for a range of tasks such as natural language processing and image recognition, while significantly enhancing computational efficiency.

3.3.1 Performer Attention

In our Performer Attention module, queries (q), keys (k), and values (v) are linearly projected from the input embeddings, followed by a scaling operation on the queries to control the variance. A distinctive aspect of this module is the application of an orthogonal random feature matrix to the queries and keys, transforming them into a space where the dot-product attention can be efficiently computed. This transformation is a crucial step in approximating the kernel of the attention mechanism with reduced computational overhead. Thus, Performer has a more efficient linear time complexity $O(n)$.

In general, by taking φ of the following form for functions $f_1, \ldots, f_l : \mathbb{R} \to \mathbb{R}$, function $g : \mathbb{R}^d \to \mathbb{R}$ and deterministic vectors ω_i or $\omega_1, \ldots, \omega_m \overset{\text{iid}}{\sim} \mathcal{D}$ for some distribution $\mathcal{D} \in \mathcal{P}(\mathbb{R}^d)$:

$$\varphi(x) = \frac{h(x)}{\sqrt{m}} \left(f_1(\omega_1^\top x), \ldots, f_1(\omega_m^\top x), \ldots, f_l(\omega_1^\top x), \ldots, f_l(\omega_m^\top x) \right) (Choromanski, 2020)$$

We can model most kernels by applying this. The softmax-kernel which defines regular attention matrix A is given as:

$$SM(x, y) \triangleq \exp(x^\top y).(Choromanski, 2020)$$

Methodology:

1. We employ softmax and Gaussian kernels with positive orthogonal random features in the FAVOR+ method to enable robust and unbiased estimation of regular (softmax) attention. The positive orthogonal random features facilitate a more efficient computation of these kernels, ensuring that the model remains scalable and precise, particularly in handling large-scale data or complex sequence modeling tasks. In our model, for $x, y \in \mathbb{R}^d$, $z = x + y$ we have:

$$\begin{aligned} SM(x, y) &= \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbb{I}_d)} \left[\exp\left(\omega^\top x - \frac{\|x\|^2}{2} \right) \exp\left(\omega^\top y - \frac{\|y\|^2}{2} \right) \right] \\ &= \Lambda \mathbb{E}_{\omega \sim \mathcal{N}(0, \mathbb{I}_d)} \cosh(\omega^\top z) (Choromanski, 2020) \end{aligned}$$

where $\Lambda = \exp\left(-\frac{\|x\|^2 + \|y\|^2}{2} \right)$ and cosh is the hyperbolic cosine. Consequently, the

softmax-kernel admits a positive random feature map unbiased approximation with:

$$h(x) = \frac{1}{\sqrt{2}} \exp\left(-\frac{\|x\|^2}{2}\right), l = 2, f_1(u) = \exp(u), f_2(u) = \exp(-u)(Choromanski, 2020)$$

The positive orthogonal random features also allow FAVOR+ to be adaptable for various other attention-based kernels, expanding its applicability in diverse neural network architectures. The Gaussian kernel is defined as

$$\mathrm{K}(x, y) = \mathrm{E}[\phi(x)^T \phi(y)].(Choromanski, 2020)$$

2. In this report, we explored the combination of ReLU activation function with multi-head attention.

   (1) Embedding: Our model starts by defining the dimensions for embedding and the number of attention heads. Each head operates on a portion of the embedding, with the dimension of each head determined by dividing the embedding dimension by the number of heads.

   (2) Scaling: The scaling factor (self.scale), based on the dimension of each head, is applied to the query vectors. This scaling is crucial in stabilizing gradients during training, particularly in models with large embedding sizes.

   (3) Linear Projections: Our class employs linear layers (self.qkv) to project input data into queries(q), keys(k), and values(v), which are fundamental components of the attention mechanism. These projections are reshaped and permuted to align with the multi-head attention framework

   (4) ReLU Activation: the ReLU activation function is applied to both the query and key vectors. The ReLU function, defined as

$$\mathrm{F}(x) = \max(0, x)$$

3.3.2 Performer Encoder

The Performer Encoder represents an advanced architecture in the realm of neural networks, specifically tailored for handling sequential data with efficiency. At our core, the Performer Encoder leverages the Performer Attention mechanism.

Our encoder structure further incorporates Layer Normalization and a Multi-Layer Perceptron (MLP), which are standard components in Transformer architectures. The Layer Normalization stabilizes the learning process by normalizing the inputs across the features. The MLP, configured with a hidden layer size determined by the "mlp ratio", adds depth and complexity to the model, allowing it to capture more intricate patterns in the data.

Additionally, our architecture employs residual connections and an optional DropPath regularization. Residual connections, or skip connections, help in mitigating the vanishing gradient problem by allowing gradients to flow through the network more effectively.

3.3.3 The ViP Model

The ViP model is a variant of the Transformer architecture, specifically adapted for image processing tasks.

Methodology:

1. Patch Embedding: The model begins with a patch embedding layer, implemented as a 2D convolutional layer. This layer divides the input image into patches and linearly embeds each patch into a higher-embedding dimensional space. This approach effectively transforms the 2D image into a sequence of flattened patch embeddings, analogous to tokens in natural language processing, making it suitable for processing by subsequent Transformer-based layers.

2. Positional Embeddings: we added it to the patch embeddings to retain positional information, as the Transformer architecture is inherently permutation-invariant.

3. Performer Encoder Blocks: Each block applies self-attention mechanisms and feed-forward neural networks, but with a key modification for efficiency, we use of positive random features to approximate attention calculations.

4. Norm: After passing through the Performer Encoder blocks, the output is normalized using Layer Normalization.

4 Results

4.1 Hyperparameter

In our aforementioned model training, we selected the hyperparameters delineated in the following table.

Hyperparameter	Value
Batch size	64
Patch size	16
Number of heads	12
Layer	12
Embedding dimension	768
Image size	224
Query-key dimension per head	128
Number of classes	10
Optimizer	Adam
Epoch	10
Dropout rate	0.0

Table 1: Hyperparameters

GPU: NVIDIA GeForce RTX 3090 with 24 GB of GDDR6X memory

4.2 Transformer

After training and testing our Vision Transformer (ViT), we obtained the following results. As observed, the testing accuracy, while not exceptionally high, is deemed acceptable, thereby establishing a performance benchmark for the other two models. Additionally, it is noteworthy that training a ViT is a time-intensive process, requiring approximately 935 seconds for a single epoch.

Epoch	Accuracy	Loss	Speed
1	14.5	2.25031537	951.9983308
2	18.346	2.136161141	940.5604765
3	17.334	2.167745756	930.5257103
4	15.858	2.214945817	925.3346455
5	17.034	2.180209881	931.056514
6	16.91	2.169583863	934.8654416
7	19.144	2.11416513	931.0050581
8	19.156	2.113837396	934.2987816
9	18.786	2.130003051	925.3456297
10	19.372	2.11541461	930.7771175

Table 2: Training results of ViT

Key Findings:

- Accuracy: Testing accuracy of 20.65% establishes a performance baseline.

Test	Accuracy	Loss	Speed
1	20.65	2.08	69.94

Table 3: Testing results of ViT

- Benchmark: Provides a reference point for model comparison.
- Training Time: Approximately 935 seconds per epoch, indicating resource intensity.
- Epoch-wise Insights: Peak accuracy at epochs 7 and 8, with potential for improvement.
- Loss Values: Gradual decrease signifies converging training.
- Speed Metrics: Training and testing speeds for computational efficiency.

4.3 Linformer

4.3.1 From Scratch

The test results of the Linformer model demonstrate a substantial reduction in the training time, nearly halving in comparison to the standard transformer. This observation emphasizes the notable efficiency enhancements facilitated by the incorporation of linear-time approximation of self-attention in the model. Furthermore, a modest improvement in the accuracy of the Linformer model is observed. We have attached the predictive outcomes of Vision Transformer (ViL) trained from scratch.

Epoch	Accuracy	Loss	Speed
1	17.244	2.249688149	390.1846991
2	16.798	2.230605467	389.9359965
3	19.196	2.179559498	389.4486818
4	20.904	2.132000291	388.7528408
5	21.18	2.113001937	388.6713746
6	21.69	2.114410361	388.5506327
7	20.088	2.165493396	388.2285089
8	22.29	2.095652324	388.1278698
9	21.774	2.117289291	387.8603308
10	23.076	2.096329741	388.0836451

Table 4: Training results of ViL (From Scratch)

Test	Accuracy	Loss	Speed
1	24.17	2.0804	37.04

Table 5: Testing results of ViL (From Scratch)

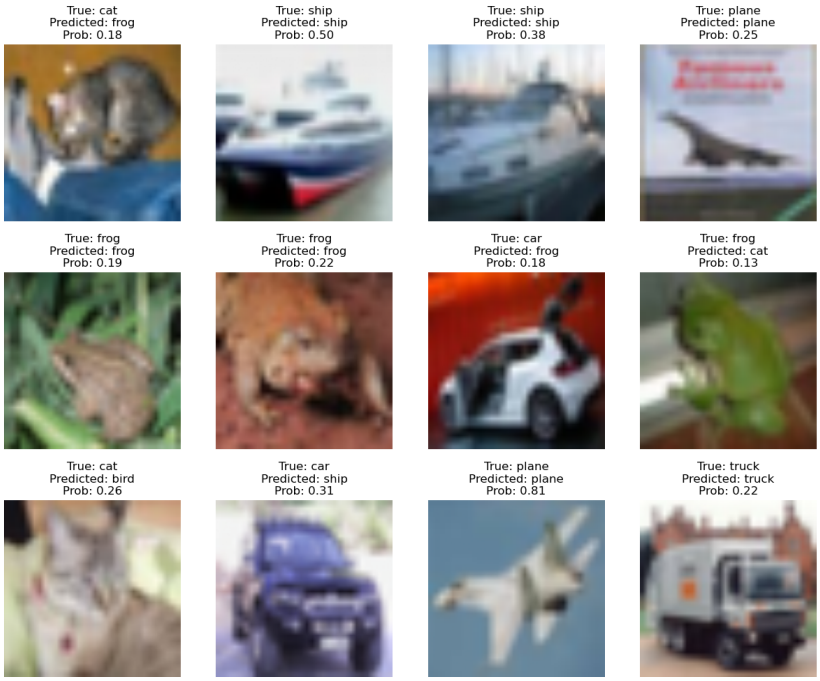

Figure 6: Predictions from ViL (From Scratch)

Key Findings:

- Training Efficiency: Linformer model exhibits significant training time reduction, almost 50% compared to standard transformers.
- Accuracy Enhancement: Modest accuracy improvement was observed in the Linformer model.
- Linear-Time Approximation: The incorporation of linear-time self-attention contributes to efficiency gains.
- Epoch-wise Insights (ViL): Peak accuracy at epoch 10 with continuous improvement.
- Speed Metrics (ViL): Testing speed of 37.04 indicates efficient model evaluation.

4.3.2 From Checkpoint

We preserved selected results from the preceding training of the Vision Transformer (ViT) in a checkpoint and subsequently leveraged them for the training of the Linformer (ViL). The acquired outcomes are presented below. Notably, the training duration exhibits a significant reduction compared to ViT, aligning with the inherent efficiency characteristics of Linformer.

Epoch	Accuracy	Loss	Speed
1	18.108	2.205288561	389.3295846
2	22.78	2.092265342	389.0710511
3	25.37	2.029607604	388.8777468
4	26.474	1.993428295	388.6639457
5	25.564	2.012228025	388.4068346
6	24.256	2.049263923	388.0371149
7	25.822	2.004438373	394.0403762
8	25.53	2.0105185	394.6265631
9	25.688	2.003103515	394.4441772
10	23.878	2.047121272	393.6331155

Table 6: Training results of ViL (From checkpoint)

Test	Accuracy	Loss	Speed
1	26.1	2.0078	37.97

Table 7: Testing results of ViL (From checkpoint)

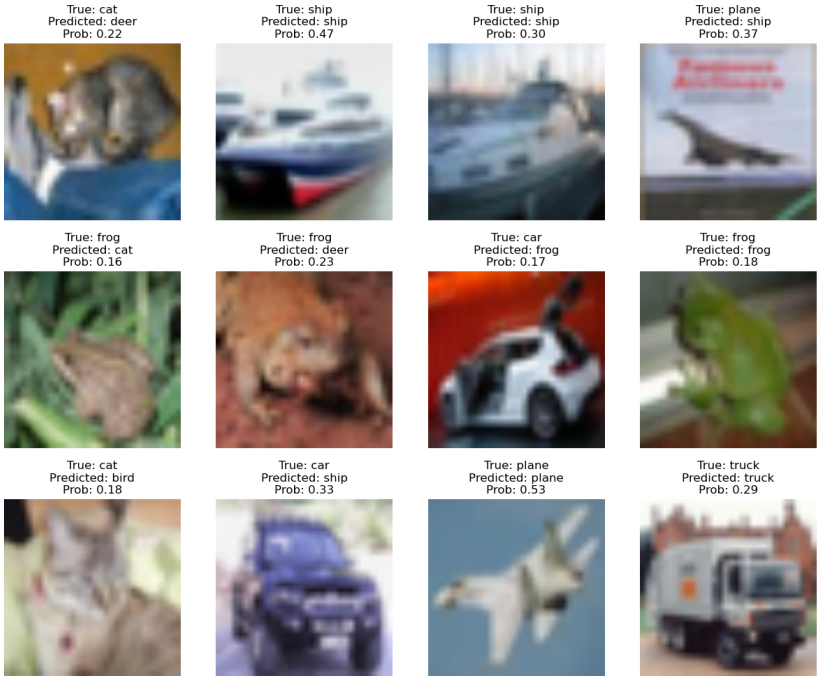

Figure 7: Predictions from ViL (From ViT Checkpoint)

Key Findings:

- Efficiency Gain: Training duration of ViL from the checkpoint shows a significant reduction compared to ViT, aligning with Linformer's efficiency.
- Accuracy Trends: ViL from the checkpoint maintains competitive accuracy across epochs, showcasing the effectiveness of leveraging pre-trained weights.

- Speed Metrics: Testing speed of 37.97 indicates efficient model evaluation in the checkpointed ViL.

4.3.3 ViL Model Performance

- Accuracy Improvement: ViL from the checkpoint maintains or slightly improves accuracy, showcasing the efficacy of utilizing ViL's learned features.

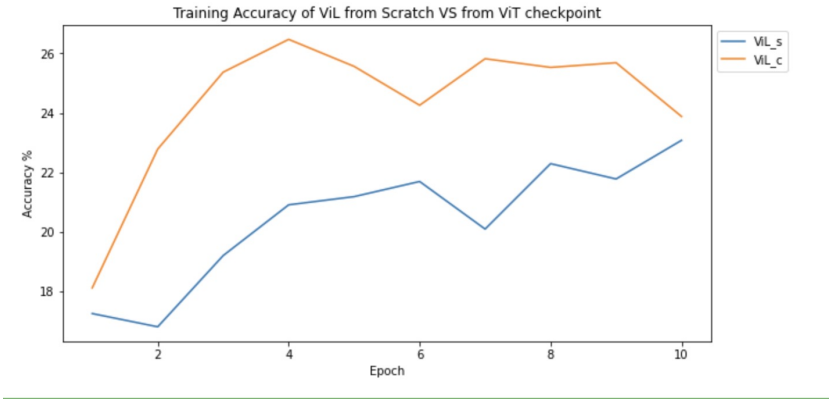

Figure 8: Training Accuracy of ViL from scratch VS from ViT checkpoint

- Training Time: ViL trained from scratch has a steadily decreasing training time, while ViL trained from the ViT checkpoint shows variable speed, with a spike around the 8th epoch. Overall, the ViL model trained from scratch demonstrates a more consistent and potentially more efficient training speed across epochs. Transfer learning from ViT might not improve the final model's speed because ViT has quadratic time complexity $O(n^2)$ which is slower than the linear time complexity $O(n)$ of ViL. Thus, the initial advantages of pre-trained weights might not translate to better efficiency in the long run.

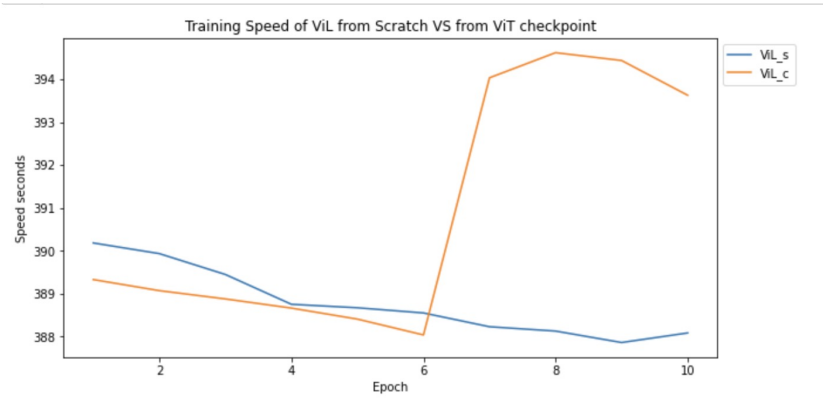

Figure 9: Training Speed of ViL from scratch VS from ViT checkpoint

4.4 Performer

4.4.1 Variant: positive random features

1. From Scratch

We found that the ViP-A model showed significant learning progress, beginning at a basic level and advancing to a high degree of accuracy. This gradual improvement is echoed in the loss metrics, which shrank notably from their starting point, suggesting that the model became more precise in its predictions as it processed more information. The model's computation speed stayed uniform, indicating consistent operational performance.

Moving to the testing phase, the ViP-A model largely maintained its effectiveness, reflecting a good ability to apply what it had learned to new data. The test loss points to some opportunities for refinement, indicating that the model can still be improved.

Overall, the results showcase the ViP-A model as a robust framework with steady computational performance and notable accuracy. This is especially significant when considered alongside the outcomes of other models like the Vision Transformer (ViL), contributing to ongoing advancements in transformer-based architectures.

Key Findings:

- Accuracy Enhancement: The model's accuracy improved significantly during training, starting at 24.208% and reaching 64.974% by the tenth epoch. In the testing phase, the model achieved an accuracy of 61.59%.
- Training Efficiency: The loss decreased markedly from its initial value to a low of 0.9807294621, indicating the model's increasing accuracy as it was exposed to more data. The testing phase loss was recorded at 1.0853, suggesting potential areas for further model optimization.
- Speed Consistency: The processing speed of the model remained consistent throughout the training period.

Epoch	Accuracy	Loss	Speed
1	24.208	2.063485971	425.8059635
2	36.858	1.744376165	420.0041111
3	42.866	1.586884411	419.2032294
4	47.438	1.466043505	419.820431
5	50.95	1.355951807	420.4903967
6	54.704	1.256543377	420.4405558
7	57.544	1.184021716	420.4295902
8	60.136	1.109944466	420.1699793
9	62.912	1.03922977	419.8954241
10	64.974	0.9807294621	420.1113605

Table 8: Training results of ViP-a (From Scratch)

Test	Accuracy	Loss	Speed
1	61.59	1.0853	38.02

Table 9: Testing results of ViP-a (From Scratch)

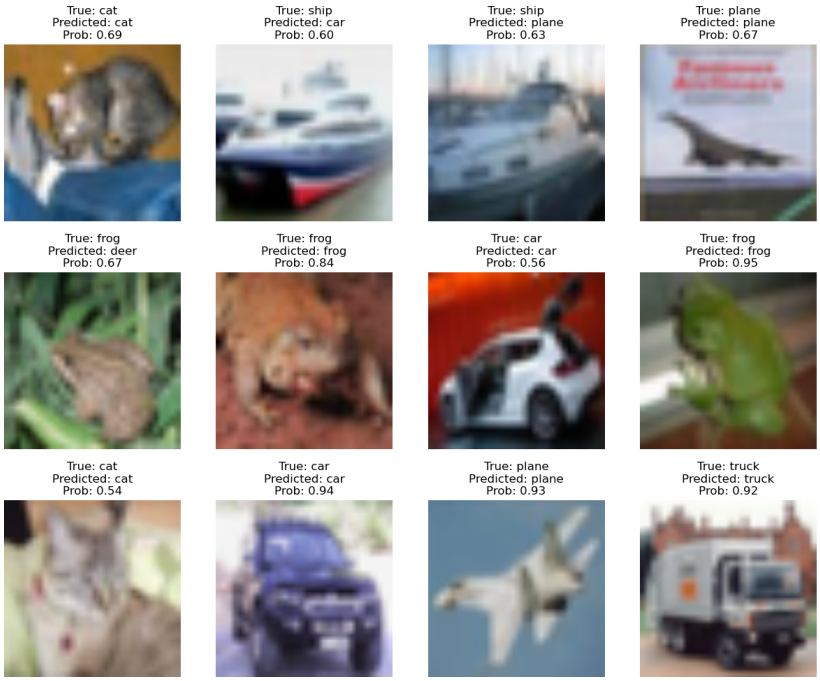

Figure 10: Predictions from ViP-a (From Scratch)

2. From Checkpoint

In this report, the ViP-A model's training performance from a checkpoint demonstrates a consistent upward trend in accuracy and a corresponding decrease in loss. This indicates that the model benefits from extended training, honing its predictive capabilities as it processes more data.

Key Findings:

- Accuracy: Achieves 58.24%, affirming the model's ability to generalize well to new data.
- Loss: Recorded at 1.1599, pointing to the model's precision in making predictions on the test set.
- Speed: Notably lower at 38.21 in the testing phase, indicating a significant change in processing time during testing compared to training.

Epoch	Accuracy	Loss	Speed
1	24.974	1.990530719	431.4721203
2	36.136	1.734897944	431.8672299
3	42.086	1.576472757	419.1184974
4	46.614	1.462155847	419.4114168
5	49.782	1.379387935	418.9704666
6	52.382	1.313594199	418.9738593
7	54.492	1.262495603	418.7389169
8	56.814	1.196094408	418.9284375
9	58.842	1.141721397	418.6392863
10	60.782	1.089657385	418.5650532

Table 10: Training results of ViP-a (From Checkpoint)

Test	Accuracy	Loss	Speed
1	58.24	1.1599	38.21

Table 11: Testing results of ViP-a (From Checkpoint)

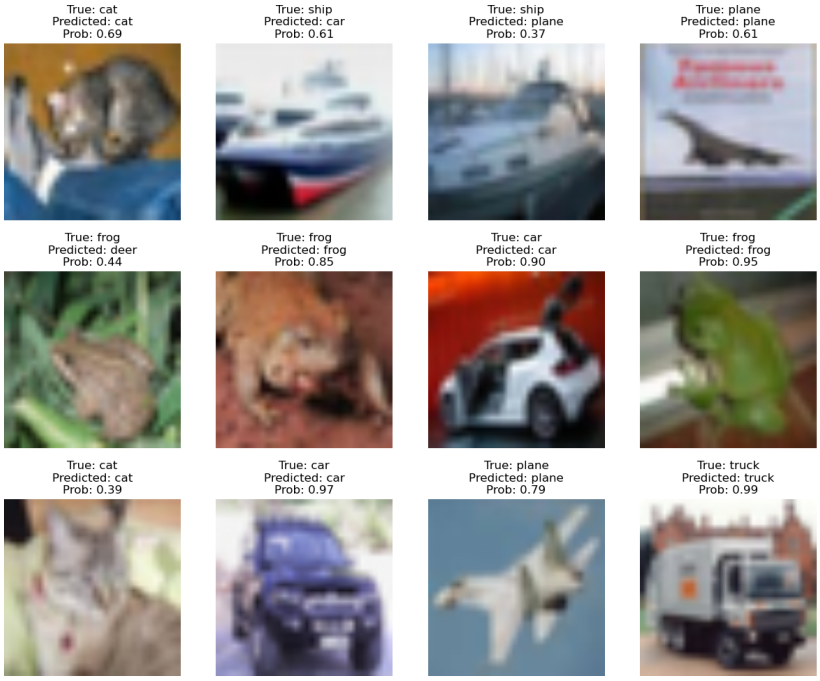

Figure 11: Predictions from ViP-a (From ViT Checkpoint)

4.4.2 Variant: ReLU nonlinearities

1. From Scratch

We observed that the ViP-ReLU model also showcases a notable progression in learning aptitude, reflected in the significant enhancement of its predictive accuracy as training progresses. Such advancement is accompanied by a consistent processing speed, underscoring the model's dependable computational performance throughout its training phase.

Key Findings:

- Accuracy: The model registers a test accuracy of 62.06%, which, while lower than the final training accuracy, demonstrates the model's ability to generalize to unseen data.
- Loss: With a recorded test loss of 1.4101, there is an indication of the model's predictive performance on the test dataset.
- Speed: A notable speed of 35.95 is observed during testing, differing from the training speed, potentially due to different test conditions or computational demands.

Epoch	Accuracy	Loss	Speed
1	37.136	1.760887852	360.6570535
2	51.348	1.359258212	361.0038655
3	57.83	1.183797797	361.1678824
4	62.22	1.05835126	364.0599923
5	66.204	0.9511057969	361.2791054
6	69.872	0.8460774732	361.0880096
7	73.716	0.7355195257	366.3072016
8	77.532	0.6232154186	361.2057261
9	81.844	0.5002579771	361.191222
10	85.748	0.3939506338	361.4748709

Table 12: Training results of ViP-ReLU (From Scratch)

Test	Accuracy	Loss	Speed
1	62.06	1.4101	35.95

Table 13: Testing results of ViP-ReLU (From Scratch)

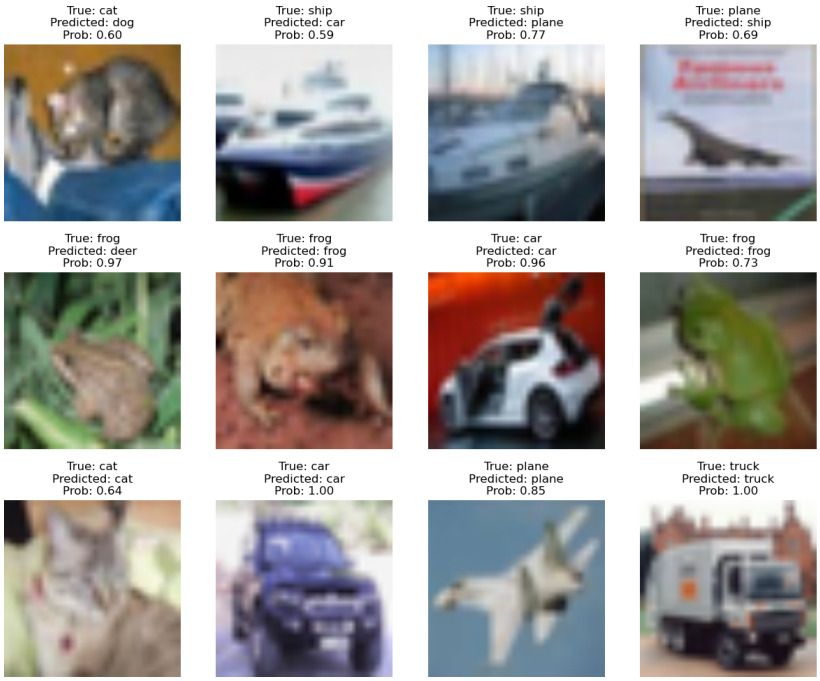

Figure 12: Predictions from ViP-ReLU (From Scratch)

2. From Checkpoint

Continuing from a checkpoint, the ViP-ReLU model has shown impressive learning dynamics, as characterized by its training performance. The model's accuracy improved markedly across epochs, reflecting a successful adaptation and deeper comprehension of the training data. This improvement in accuracy corresponds with a substantial decline in loss, indicating that the model's predictions became increasingly precise over time. Similar to the results obtained from previous models, the model's processing speed remained fairly consistent, which is an encouraging sign of the model's computational stability.

In the testing phase, the ViP-ReLU model displayed a good level of accuracy, confirming the model's ability to effectively generalize the knowledge acquired during training to unseen data. The model's test speed varied from the training phase, possibly reflecting different computational loads or testing configurations.

Key Findings:

- Accuracy: The model's accuracy showed a significant increase from 38.664% at epoch 1 to 86.692% by epoch 10, which is also the highest accuracy we have achieved in this report.
- Loss: There was a considerable decrease in loss in the training period.

Epoch	Accuracy	Loss	Speed
1	38.664	1.682479757	362.3932302
2	52.994	1.302921862	362.2847588
3	59.932	1.118216716	362.063529
4	64.448	0.9991495006	362.535078
5	67.918	0.9037227175	362.5165827
6	71.544	0.7996329907	362.433341
7	75.118	0.6971924024	361.9886413
8	78.942	0.5890665943	362.324204
9	83.012	0.4762933264	361.7517765
10	86.692	0.3700677932	362.2385216

Table 14: Training results of ViP-ReLU (From Checkpoint)

Test	Accuracy	Loss	Speed
1	63.71	1.4715	35.28

Table 15: Testing results of ViP-ReLU (From Checkpoint)

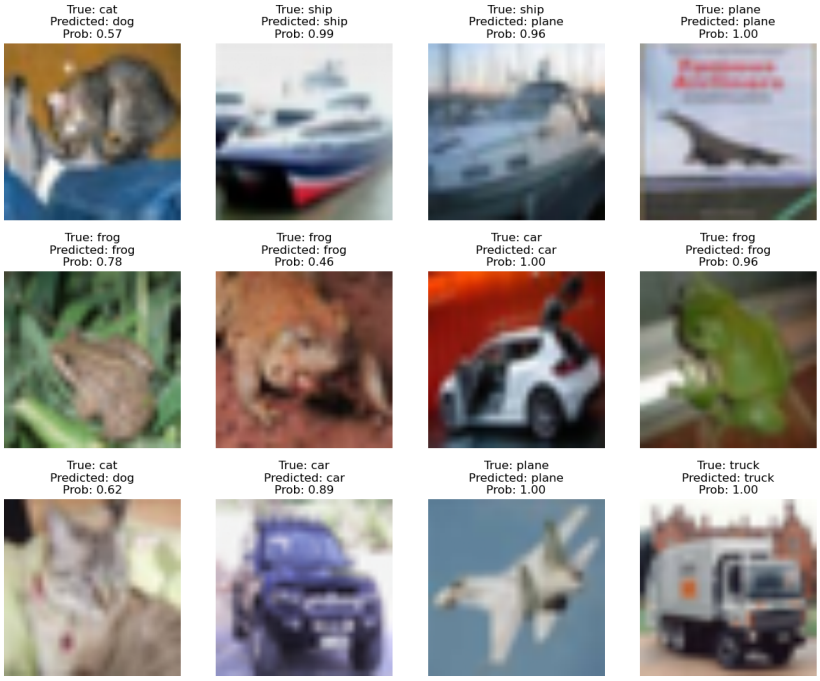

Figure 13: Predictions from ViP-ReLU (From ViT Checkpoint)

4.4.3 Variant: EXP

1. From Scratch

The application of the EXP variant potentially provides an effective approach to managing the model's complexity and facilitating its learning process. The enhancement in performance metrics suggests that the model is successfully capturing the underlying patterns within the training dataset. As the epochs progress, the model's ability to reduce error rates and enhance

prediction accuracy becomes evident. Additionally, the stability in processing speed throughout the training epochs further indicates that the model is efficiently utilizing computational resources without compromising performance.

However, there is slight discrepancy in loss points to the model's potential overfitting to the training data and indicates room for improvement in model generalization.

Key Findings:

- Testing Accuracy and Loss: The model achieves a testing accuracy of 60.8% with a loss of 1.3866. While the testing accuracy is lower than the final training accuracy.

Epoch	Accuracy	Loss	Speed
1	37.734	1.740057835	367.4902542
2	50.688	1.374043438	367.7788591
3	56.876	1.205672385	367.501714
4	61.012	1.088795159	367.2354243
5	64.866	0.9866908412	368.0222888
6	68.246	0.8900541047	367.8052654
7	72.002	0.7858314775	367.6575601
8	76	0.6731833156	367.4836648
9	80.178	0.555473798	367.7747083
10	84.286	0.4356426591	367.4339218

Table 16: Training results of ViP-EXP (From Scratch)

Test	Accuracy	Loss	Speed
1	60.8	1.3866	35.65

Table 17: Testing results of ViP-EXP (From Scratch)

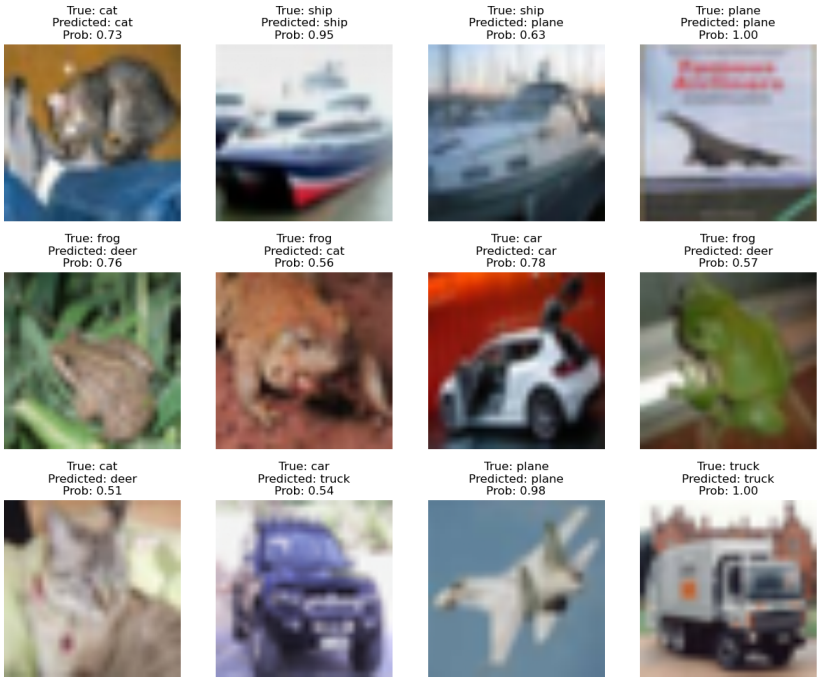

Figure 14: Predictions from ViP-EXP (From Scratch)

2. From Checkpoint

After resuming from a checkpoint, the ViP-EXP model displays a significant advancement in its training performance, evidenced by a notable improvement in accuracy and a consistent decline in loss. The model's learning curve suggests effective further training from the checkpoint, with a positive impact on its ability to make accurate predictions.

Key Findings:

- Training Accuracy: There's an upward trend from the first epoch to the tenth, reflecting the model's capacity to enhance its performance as it processes additional training data.
- Testing Performance: The model achieves a solid accuracy in the testing phase, which points to its effectiveness in generalizing from the training to the testing dataset. The slightly lower testing speed compared to the training speed may be due to a range of factors, including the different nature of the testing data or the computational environment.

Epoch	Accuracy	Loss	Speed
1	39.384	1.669344079	367.2127366
2	52.846	1.308861423	367.1001122
3	58.898	1.152107128	367.5314858
4	62.634	1.04867035	367.311569
5	65.696	0.9553475635	367.3653035
6	69.496	0.8610032282	367.4367776
7	72.454	0.7725070097	367.1451221
8	75.782	0.6753412035	367.337985
9	79.812	0.5643623685	367.5919876
10	83.608	0.4563867958	367.3902822

Table 18: Training results of ViP-EXP (From Checkpoint)

Test	Accuracy	Loss	Speed
1	63.05	1.3686	35.43

Table 19: Testing results of ViP-EXP (From Checkpoint)

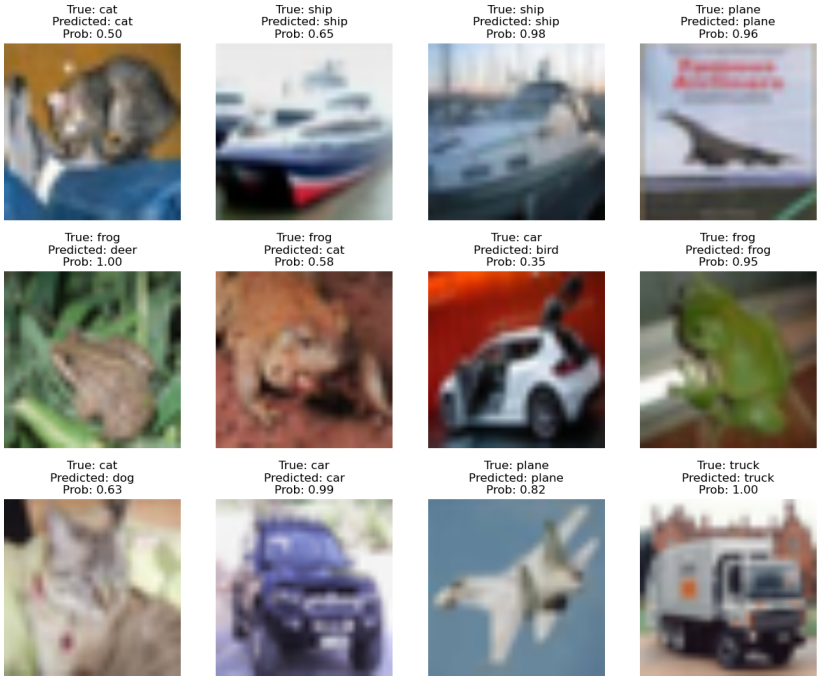

Figure 15: Predictions from ViP-EXP (From ViT Checkpoint)

4.4.4 ViP Model Performance

Overall, the training speed among different models is relatively stable across increasing epochs, with the ViP-ReLU model training from checkpoints having the quickest speed and the highest accuracy, while the ViP with positive random features model training from checkpoints has the relatively slowest speed and lowest accuracy.

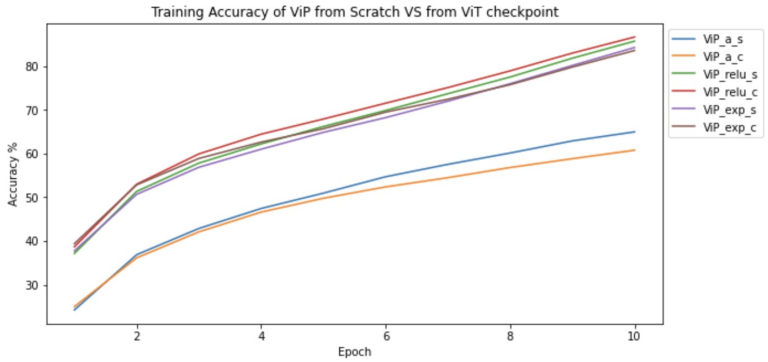

Figure 16: Training Accuracy of ViP from scratch vs from ViT checkpoint

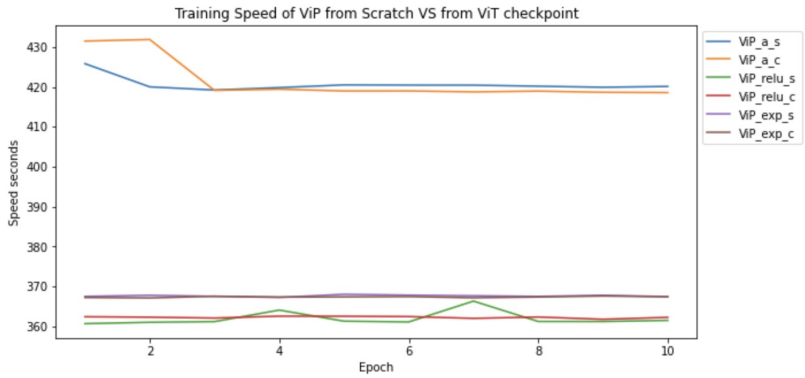

Figure 17: Training Speed of ViP from scratch vs from ViT checkpoint

4.5 Model Comparison

From the graph, it appears that the ViT model has a significantly slower training time compared to the other models. This difference is consistent across all epochs. The Linformer and Performer, whether trained from scratch or ViT checkpoint, maintain a relatively consistent and much faster training time over epochs when compared to the ViT model. Thus, the ViL and ViP models are more efficient in training speed than the ViT model due to changing self-attention time complexity from $O(n^2)$ to $O(n)$.

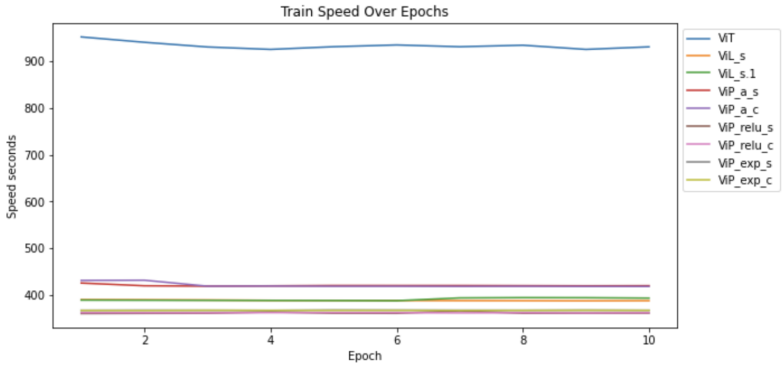

Figure 18: Training Speed Over Epochs

The result of the test speed per model is the same as the training speed over epochs. Transfomers are much slower than Linformers and Performers. The test speed of Transformers is 69.9 seconds, while other models' speeds are near 35 seconds. Specifically, Performers using a deterministic mechanism with ReLU is the fastest one.

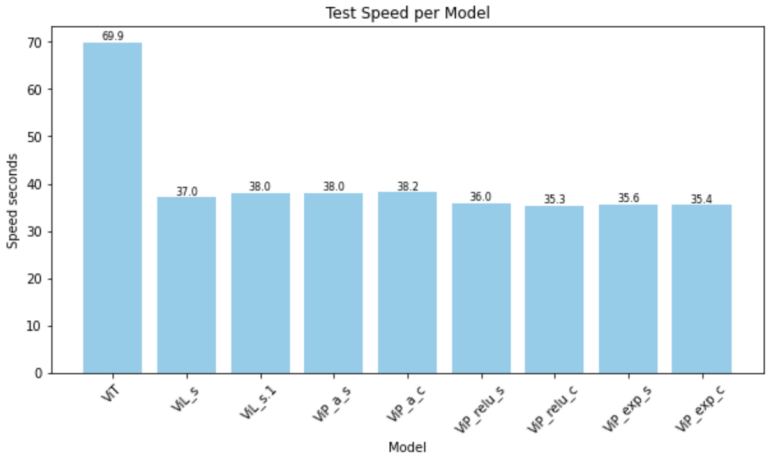

Figure 19: Test Speed per Model

5 Conclusion

This study provides valuable insights into the comparative performance of various Transformer architectures for small image classification tasks. Regular Transformers, Performers, and Linformers each exhibit unique characteristics and trade-offs. Performers, particularly with positive random features and deterministic mechanisms, demonstrate significant accuracy enhancements and consistent training efficiency. Speed in transformer with self-attention has time complexity of $O(n^2)$, which is slower than that of Linformer and Performer with linear-attention $O(n)$.

Regular Transformers, serving as our benchmark, revealed a resource-intensive nature with a testing accuracy of 20.65%. While providing a reference point, their training time of approximately 935 seconds per epoch signals a computational intensity that may not be ideal for resource-constrained environments. This sets the stage for investigating alternative architectures that balance efficiency and accuracy.

Linformers emerged as compelling alternatives, exhibiting a substantial training time reduction of nearly 50% compared to regular Transformers. The incorporation of linear-time self-attention contributed to efficiency gains, complemented by a modest accuracy improvement. Leveraging pre-trained checkpoints further validated the efficacy of Linformers, showcasing consistent accuracy across epochs and efficient model evaluation with a testing speed of 37.04.

Performers, with their unique attention mechanisms, demonstrated remarkable accuracy enhancements during training, reaching a peak accuracy of 64.974% by the tenth epoch. The training efficiency of Performers was evident in the consistent decrease in loss, indicating the model's increasing accuracy as it processed more data. Leveraging pre-trained checkpoints not only maintained or slightly improved accuracy but also showcased consistent processing speed, emphasizing the robustness of the models.

The investigation reveals that Performer models exhibit significant accuracy improvements during training, with varying mechanisms showcasing distinct learning aptitudes. Linformers, known for their efficiency, demonstrate substantial training time reduction and competitive accuracy. Leveraging pre-trained checkpoints, both for Performers and Linformers, highlights the benefits of knowledge transfer, emphasizing the efficacy of learned features. The study also introduces Performer variants with ReLU and EXP nonlinearities, showcasing their predictive accuracy and computational stability.

Notably, the ViP-ReLU model displays consistent learning dynamics, achieving high accuracy and computational stability during training and testing phases. The ViP-EXP model, after resuming from a checkpoint, exhibits notable advancements in training performance, emphasizing the effectiveness of further training. Additionally, the proposed linear attention mechanism, Linformer, is introduced as a memory- and time-efficient alternative to standard Transformers, maintaining competitive performance.

Exploring different variants of Performers, including those with positive random features and deterministic mechanisms employing ReLU and EXP nonlinearities, added depth to our analysis. The findings underscored the adaptability of Performers to diverse configurations,

showing notable progression in learning aptitude and dependable computational performance.

In conclusion, our study contributes to the broader understanding of Transformer architectures in small image classification tasks, offering a comprehensive comparison and paving the way for further exploration and optimization. As the demand for efficient yet accurate models persists, the insights gleaned from this research are poised to guide future developments in the realm of deep learning and artificial intelligence.

References

[1] Choromanski, Krzysztof Marcin, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, et al. "Rethinking Attention with Performers," 2020. https://openreview.net/forum?id=Ua6zuk0WRH.

[2] Dgvv4. "(10) Vision Transformer Animal Image Three-Class Classification with Complete Pytorch Code." CSDN, 2022. Available at: https://blog.csdn.net/dgvv4/article/details/125184340.

[3] Fymwin. "Vision Transformer." Kaggle, 2023. Available at: https://www.kaggle.com/code/fymwin/visio transformer.

[4] Google Research. "Google Research." GitHub repository. (2023). Available at https://github.com/google research/google-research.

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020.

[6] Lucidrains. "Performer-Pytorch." GitHub repository. (2003). Available at https://github.com/lucidrains pytorch.

[7] Qq_37937847. "Title of the Blog Post." CSDN, 2021. Available at: https://blog.csdn.net/qq_37937847/art

[8] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N., Kaiser Łukasz, and Polosukhin Illia. 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30 (2017).

[9] Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. (2020). Linformer: Self-attention with linear complexity. CoRR, abs/2006.04768.