PIVOT- Input-aware Path Selection for Energy-efficient ViT Inference

Abhishek Moitra
abhishek.moitra@yale.edu
Yale University, New Haven, CT, USA

Abhiroop Bhattacharjee
abhiroop.bhattacharjee@yale.edu
Yale University, New Haven, CT, USA

Priyadarshini Panda
priya.panda@yale.edu
Yale University, New Haven, CT, USA

ABSTRACT

The attention module in vision transformers(ViTs) performs intricate spatial correlations, contributing significantly to accuracy and delay. It is thereby important to modulate the number of attentions according to the input feature complexity for optimal delay-accuracy tradeoffs. To this end, we propose PIVOT - a co-optimization framework which selectively performs attention skipping based on the input difficulty. For this, PIVOT employs a hardware-in-loop co-search to obtain optimal attention skip configurations. Evaluations on the ZCU102 MPSoC FPGA show that PIVOT achieves 2.7× lower EDP at 0.2% accuracy reduction compared to LVViT-S ViT. PIVOT also achieves 1.3% and 1.8× higher accuracy and throughput than prior works on traditional CPUs and GPUs. The PIVOT project can be found at this Github link.

KEYWORDS

Vision Transformers, Systolic Array Accelerators, Energy-efficiency

1 INTRODUCTION

Vision Transformers (ViT) have demonstrated remarkable accuracy in large-scale image classification tasks [3, 5, 6]. The success of ViTs can be attributed to the attention module shown in Fig. 1a which utilizes the self-attention mechanism to perform sophisticated spatial correlation operations [6]. However, the attention module, involves computationally intensive operations, including matrix multiplications and non-linear functions like softmax [3, 6]. Hence, as seen in Fig. 1b, the attention module (QKV+QK^T+SM+(SMxV)+Proj combined) contributes 77.5% to 81.9% of the total ViT inference delay.

Recently, there have been several ViT inference optimization works that focus on reducing the attention delay overhead. These mainly fall under two categories 1) Attention sparsification [8, 17] 2) Token pruning techniques [4, 11, 15]. Attention sparsification techniques exploit the sparsity in the QK^T and (SMxV) layers [8, 17] (Fig. 1a). In [8], the authors algorithmically investigate the effect of structured sparsity in the attention heads on ViT accuracy. In a more recent work [17], the authors propose an accelerator co-design framework that performs sparse-dense attention decomposition and develop a sparse accelerator to exploit the attention sparsity. The objective of token pruning is to selectively reduce the number

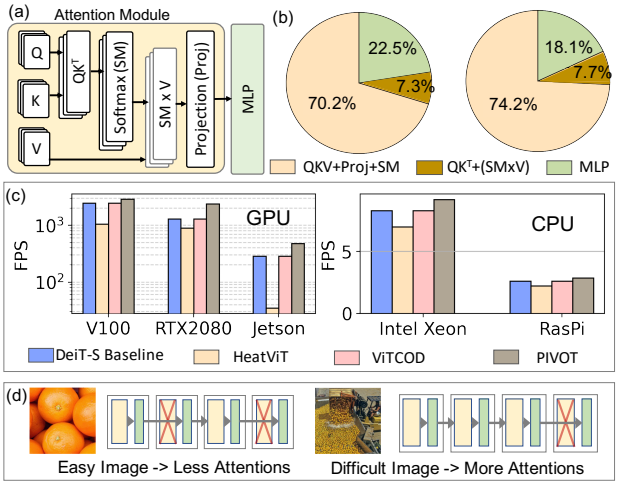

Figure 1: (a) Figure showing the encoder architecture of a vision transformer. Q-Query, K-Key and V-Value. (b) Delay distribution across different ViT modules for DeiT-S (left) and LVViT-S (right) ViTs. Note, Attention delay is QKV+SM+QK^T+(SMxV)+Proj. (c) Throughput of PIVOT compared with DeiT-S Baseline (a standard DeiT-S [14] ViT), prior token pruning (HeatViT [4]) and attention sparsification (ViTCOD [17]) techniques implemented on GPUs- Nvidia V100, RTX2080ti, Jetson Orin Nano and CPUs- Intel Xeon and Raspberry Pi 4. (d) PIVOT's input difficulty-aware inference

of tokens in the ViT. In [11, 15], the authors use predictor networks to compute the global-local token importance to eliminate redundant tokens. In HeatViT [4], the authors use predictor networks to score the token importance based on the information in each attention head. Along with the predictor networks, the authors use a token packaging technique wherein unimportant tokens are combined into one token to maintain a good accuracy-efficiency tradeoff. Although, attention sparsification and token pruning works [4, 8, 11, 17] achieve good accuracy at reduced computation, they have two major problems. Firstly, the portion of delay optimized by these works is small. For example, attention sparsification works are only able to optimize 7.3-7.7% of the overall delay since they target the QK^T and (SMxV) layers as shown in Fig. 1b. The second problem is that attention sparsification and token pruning approaches require nuanced hardware support to achieve optimal efficiency. For example, attention sparsification works require sparse matrix multiplication hardware to fully exploit sparse computations. Similarly, token pruning works require custom hardware design to efficiently implement the token score predictor modules. Thus, as shown in Fig. 1c, when implemented on general purpose platforms (GPPs) such as CPUs and GPUs, they do not achieve any inference delay benefits and, in fact, result in lower throughput compared to a dense baseline.

Another missing consideration in prior ViT optimization literature is the input difficulty awareness. Interestingly, different images have different feature complexity. For example, an easy image will

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3655679

contain simple, low-level features compared to a difficult image with intricate feature representations [16]. Since attention modules are responsible for capturing different levels of feature representations in the image, it is therefore imperative to modulate the number of attentions in a ViT according to the input difficulty (Fig. 1d). Modulating the number of attentions according to input difficulty will ensure minimal attention activation to achieve high accuracy at low inference delay. There have been several input difficulty-aware network optimization works in the CNN literature [1, 10, 16]. However, there are no works that analyze the co-dependency between the number of attentions and input difficulty from the perspective of accuracy and ViT inference delay.

To this end, we propose PIVOT, a hardware-algorithm co-design framework that modulates the number of attentions in the ViT according to the input difficulty. The goal of PIVOT is to achieve high classification accuracy by using the minimum number of attentions in the ViT. As shown in Fig. 1d during inference, PIVOT uses two kinds of ViTs - 1) Low Effort and 2) High Effort ViT. The low effort ViT entails more attention skips compared to the high effort and classifies the easy images. While the high effort ViT is used for classifying the difficult images. An iterative hardware-in-the-loop co-search is applied to obtain the optimal low and high effort ViTs according to the user-provided delay constraints. For evaluation, we implement PIVOT on various GPPs such as CPUs and GPUs. Additionally, we also evaluate PIVOT on Xilinx ZCU102-implemented systolic array accelerator [12]. Unlike token pruning and attention sparsification works, PIVOT does not require any application-specific hardware and can achieve 1.3×-2× higher throughput than baseline across various GPPs as shown in Fig. 1c.

In summary, the key contributions of our work are:

(1) We propose PIVOT- a hardware-algorithm co-optimization framework that leverages input difficulty-aware attention skipping in ViTs to overcome the high inference delay overhead of the attention module. During attention optimization, PIVOT uses PIVOT-Sim, a cycle-accurate simulator for ViT implemented on a Xilinx ZCU102 FPGA-based systolic array accelerator. PIVOT-Sim will be made open-source and can benchmark different state-of-the-art ViTs.
(2) Using PIVOT-Sim, we find that PIVOT achieves 1.73× (2.7×) lower energy-delay-product (EDP) at merely 0.4% (0.2%) accuracy reduction compared to DeiT-S [14] (LVViT-S [7]) baselines. End-to-end evaluations using PIVOT-Sim show that PIVOT is able to achieve more than 1.7× energy reduction across different resources in the Xilinx ZCU102 FPGA such as the ZynQ MPSoC PS, systolic array, on-chip buffers, and communication/memory controller circuits.
(3) Through extensive experiments we show the overheads introduced by prior ViT co-optimization works [4, 17] when implemented on GPPs such as GPUs and CPUs. As PIVOT does not require nuanced hardware support, when implemented on GPPs, it achieves 1.8× higher throughput at 0.4-1.3% higher accuracy compared to prior works.

2 BACKGROUND ON VISION TRANSFORMER

A Vision Transformer (ViT) comprises multiple cascaded encoders, and each encoder follows the architecture depicted in Fig. 1a. In each encoder, the inputs of dimensions $t \times d$ undergo QKV operations wherein, weights W_Q, W_K and W_V are multiplied with the input to generate the Query (Q), Key (K) and Value (V) matrices. The attention module uses the multi-head self-attention (MHSA) mechanism, that captures close relationships between different image features [7, 11, 14]. For this, the Q, K and V outputs are partitioned into multiple smaller attention heads (Q_i, K_i, V_i), where i denotes a head of MHSA.

The attention is computed using Equation 1. In each head matrix multiplications between Q_i, K_i^T (QK^T) is performed followed by the softmax (SM) and matrix multiplication with V_i (SM×V) operations [13]. The softmax is computed using Equation 2.

$$Attention(Q_i, K_i, V_i) = Softmax(\frac{Q_i K_i^T}{\sqrt{d}})V_i, \tag{1}$$

$$Softmax(x_i) = \frac{e^{x_i - x_{max}}}{\sum_i e^{x_i - x_{max}}}. \tag{2}$$

Next, the attention outputs are concatenated resulting in a $t \times d$ output attention matrix. Following this, the projection and MLP layers project the information into a higher dimension feature space. Each encoder outputs a $t \times d$ vector that is forwarded to the subsequent encoder.

3 PIVOT METHODOLOGY

3.1 PIVOT Inference with Low and High Efforts

During PIVOT's inference, we use the entropy metric to determine the number of attentions required to classify an input [9]. The entropy, $E(x)$, for an input x (belonging to a dataset with K classes) is calculated using Equation 3. Here, $\pi(\boldsymbol{y}|\boldsymbol{x})$ is the logit output of the ViT. The term $1/\log K$ normalizes the final entropy to $(0, 1]$.

$$E(\boldsymbol{x}) = -\frac{1}{\log K} \sum_{i=1}^{K} \pi(\boldsymbol{y}_i|\boldsymbol{x}) \log \pi(\boldsymbol{y}_i|\boldsymbol{x}). \tag{3}$$

The entropy measures the confidence of prediction. For example, if all classes have an equal probability of $\frac{1}{K}$, the entropy value will be 1, implying uncertainty in the prediction. Whereas, if one class's prediction probability reaches 1 while the other classes attain 0 probability, the entropy reaches 0 implying confident prediction.

As shown in Fig. 2a, during inference, PIVOT uses a combination of two efforts: 1) Low Effort and 2) High Effort. Here, *Effort* is defined as the number of active attention modules (attentions that are not skipped) in the ViT. First, all inputs are inferred with the low effort resulting in the logit outputs ($\pi(\boldsymbol{y}|\boldsymbol{x})$) and the entropy values ($E(x)$). For inputs with entropy values lower than the threshold (Th), the $\pi(\boldsymbol{y}|\boldsymbol{x})$ from the low effort ViT are used for class prediction. For inputs with $E(x) > Th$, an additional inference is performed with high effort and then, all inputs are inevitably classified. In Fig. 2a, F_L and F_H are defined as the fraction of inputs classified by low ($E(x) < Th$) and high effort ($E(x) > Th$), respectively. Additionally, the number of inputs correctly (incorrectly) classified with low and high efforts are denoted as C_L (I_L) and C_H (I_H), respectively. The C_L and C_H values are used to compute the accuracy.

Re-computation Overhead: During inference, some of the inputs that are unclassified with the low-effort are re-inferred with

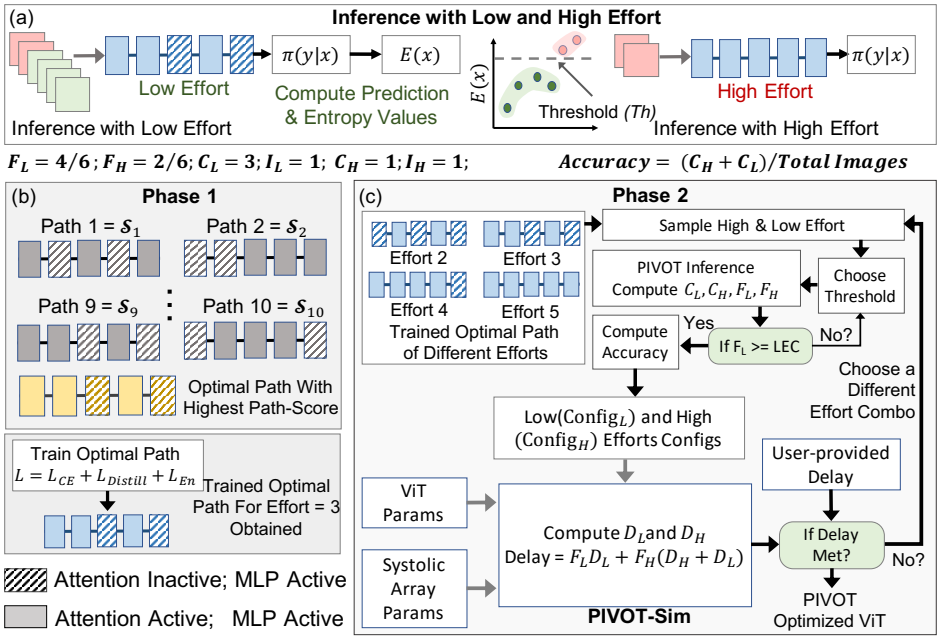

Figure 2: Figure showing (a) Input difficulty-aware inference procedure with PIVOT (b) PIVOT's Phase 1 (b) Phase2 Methodology. *LEC* denotes the user-provided low effort constraint which implies the fraction of inputs that must be classified by the low effort ViT. For PIVOT-Sim, ViT params include embedding dim size, mlp ratio etc. and systolic array params include array size, dataflow, etc.

the high effort which entails re-computation overhead that needs to be managed to obtain a tradeoff between accuracy vs. efficiency.

3.2 PIVOT Phase1: Optimal Path Selection

PIVOT uses a two-phase hardware-in-the-loop search to design the multi-effort ViT. In Phase1, we select the optimal path for different efforts for a given ViT. Each effort contains multiple *Paths*. For example, as shown in Fig. 2b, a ViT with 5 encoders and *Effort*=3 entails $\binom{5}{3} = 10$ possible paths. Here, a *Path* is uniquely defined by the position of encoders with active and inactive attention modules. Having large number of paths for each effort increases the search space size in Phase2. Therefore, we define a *Path-Score* (shown in Algorithm 1) metric to single-out the *Optimal Path* (shown in yellow) corresponding to each effort. The path with the highest *Path-Score* ($\mathcal{S}$) is chosen as the *Optimal Path* and trained with the loss function shown in Fig 2b. The loss function contains cross-entropy loss L_{CE}, and the distillation loss $L_{Distill}$ between the final layer features of the teacher and student ViT. The L_{CE} and $L_{Distill}$ are commonly used in prior works to train high performance ViTs [7, 14]. In PIVOT, to improve the prediction confidence, we add the regularization term L_{En} that lowers the entropy for the correctly classified inputs. L_{En} is the mean of the entropy values for the correctly classified inputs. Lowering the entropy ensures increased confident classifications with low efforts and thereby improves the inference efficiency.

CKA Matrix Fig. 3a shows the center kernel alignment matrix (*CKA Matrix*) comprising of the CKA values computed between MLP outputs (MLP_i) and attention outputs (A_{i+1}) of ViT encoders $Encoder_i$ and $Encoder_{i+1}$, respectively. CKA measures the similarity between two matrices [2]. A high $CKA(MLP_i, A_{i+1})$ value implies high similarity in MLP_i and A_{i+1} outputs, thus suggesting that output MLP_i can be directly forwarded to MLP_{i+1} by skipping A_{i+1}

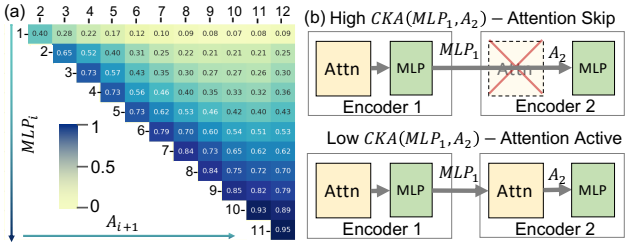

Figure 3: (a) *CKA Matrix* computed between the MLP output of $Encoder_i$ (MLP_i) and Attention output of $Encoder_{i+1}$ (A_{i+1}) for the DeiT-S ViT (b) Higher $CKA(MLP_i, A_{i+1})$ suggests data redundancy and the attention can be skipped.

as shown in Fig. 3b (top). Contrarily, for a low $CKA(MLP_i, A_{i+1})$ value, the attention cannot be skipped as shown in Fig. 3b (bottom).

Algorithm 1: Path-Score Computation Algorithm

Input: Effort Configuration (*Config*), #Encoders in ViT (*D*), *CKA Matrix*.
Output: Path-Score ($\mathcal{S}$)

```
S = 0;
for i ϵ Config do
    for j ϵ (i + 1, D) do
        if (A_j is Inactive) then
            S = S + CKA Matrix(i, j);
        else
            break;
```

Path-Score ($\mathcal{S}$): Algorithm 1 shows the methodology to compute $\mathcal{S}$. Algorithm 1 requires the *CKA Matrix* (shown in Fig. 3a) and the effort configuration (*Config*), containing encoder locations

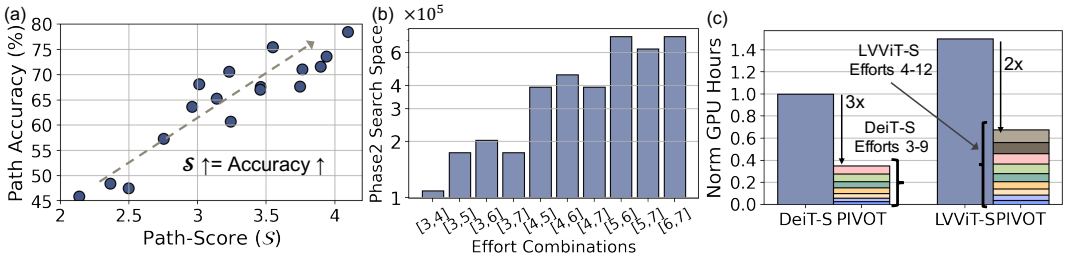

Figure 4: (a) Path Accuracy vs. *Path-Score* ($\mathcal{S}$) corresponding to Effort = 6 for DeiT-S ViT. (b) Design space size if random search is performed in Phase2, without selecting optimal path for each effort in Phase1 (size normalized to PIVOT's design space size) (c) GPU hours for training DeiT-S, LVViT-S and PIVOT Efforts (normalized to GPU hours required for training DeiT-S from scratch).

with active and inactive attention. The *CKA Matrix* is generated for a small batch of 256 images. For a given $Config$, $\mathcal{S}$ is computed by summing up the CKA values between the MLP outputs (MLP) of the encoders with active attention and the attention outputs (A) of the encoders with inactive attention. For example, $\mathcal{S}$ for $Config$ = [1,2,3,4,5,6,7,8,9,10,11,12], where encoder indices of inactive attentions are denoted by cyan can be computed as $CKA[MLP_2, A_3]$ + $CKA[MLP_2, A_4]$ + $CKA[MLP_8, A_9]$ + $CKA[MLP_8, A10]$. A high $\mathcal{S}$ signifies that the path contains highly redundant attentions that can be easily pruned out. Fig. 4a shows the positive correlation between $\mathcal{S}$ and path accuracy. As high S paths ensure pruning the most redundant attention blocks, they attain higher accuracy.

3.3 PIVOT Phase2: Selecting Optimal Effort Combinations

In Phase2, given a set of efforts with optimal paths (shown in blue in Fig. 2c), PIVOT determines the right effort combination to achieve optimal accuracy while meeting the user-provided delay requirement. 1) First, we start with a pair of low and high efforts (say, Effort 9 and Effort 12). 2) Next, the threshold values Th for the low effort inference is chosen. The Th values are iterated in an incremental manner. 3) A small batch of data (randomly sampled batch of 256 images from the training set) is inferred with the low and high efforts. This generates the C_L, C_H, F_L and F_H values. 4) Following this, the accuracy calculator uses C_L and C_H to compute the accuracy (Fig. 2a). The thresholds are iterated until the condition $F_L \geq LEC$ is met. Higher LEC value ensures more inputs classified by the low effort ViT. 5) The low ($Config_L$), high ($Config_H$) effort configurations, F_L and F_H values are passed to the PIVOT-Sim framework for delay computation. The PIVOT-Sim platform first computes the delays of low and high efforts (D_L and D_H, respectively) using $Config_L$, $Config_H$, ViT and systolic array parameters (Refer Section 3.4). Then, it computes the delay of the effort combination using D_L, D_H, F_L and F_H. If the delay lies within 5% of the user-provided delay constraint, the optimal effort combination is obtained. If the delay constraint is not met, a new effort combination (say, Effort 6 and Effort 9) is selected. In order to achieve high accuracy, the sampling starts with efforts containing maximum active attentions. In each iteration, a smaller effort combination is sampled than the previous iteration until the desired delay is obtained.

Benefit of CKA Score-based Optimal Path Selection In Fig. 4b, we compare the Phase2 design space size of random and PIVOT-based search. Since PIVOT uses the *Path-score* to single out the optimal path for each effort, there exists only one path for each effort combination. Whereas, Phase2 with random search entails multiple paths due to the absence of optimal path selection. For example, in random search as shown in Fig. 4b, effort combinations [3,6] can contain $\binom{12}{3} \times \binom{12}{6} = 2.03 \times 10^5$ possible paths for the DeiT-S ViT. For the DeiT-S ViT Phase2 with random search, the search space size is $\sim 10^5\times$ higher than PIVOT's search space size.

GPU Hours for Training all Efforts: Fig. 4c shows that the combined GPU hours required for training all efforts (see Section 4.1) for DeiT-S (LVViT-S) ViTs in PIVOT is 3× (2×) less compared to training the DeiT-S (LVViT-S) ViT from scratch. This is because, the training time reduces with reduction in the ViT effort.

3.4 PIVOT-Sim Platform

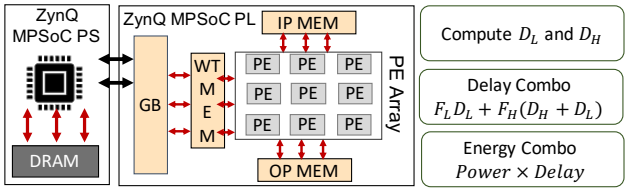

Figure 5: Figure showing the PIVOT-Sim Platform.

Fig. 5 shows the overall architecture of the PIVOT-Sim platform. PIVOT-Sim performs cycle-accurate delay estimation for a given ViT effort mapped on a Xilinx ZCU102 MPSoC FPGA-based systolic array accelerator. Like the ZynQ MPSOC FPGA, PIVOT-Sim contains two systems: 1) ZynQ MPSoC Processing System (PS) and 2) ZynQ MPSoC programmable logic (PL). All the linear matrix multiplication layers (QKV, QK^T, SMxV, Proj and MLP) are executed in the PL-implemented systolic accelerator. Inputs and weights are first loaded from the PS DRAM to the global SRAM buffer (GB) in the PL. Then the weights and inputs are fetched from GB to the Weight SRAM (WTMEM) and Input SRAM (IPMEM), respectively. Then, the weights from the WTMEM are loaded in to the PE array in a streaming fashion following which, the inputs are fetched from the IPMEM in a streaming fashion column by column. The multiply-and-accumulate (MAC) outputs are stored in the output SRAM (OPMEM). The outputs are pushed to the GB and finally to the DRAM. The non-linear operations such as softmax, entropy and GeLU are implemented using the ZynQ MPSoC PS.

The PIVOT-Sim framework requires the ViT parameters (embedding dimension size, number of tokens, mlp ratio and attention head count) and systolic array parameters (array dimensions, dataflow, SRAM memory sizes, and the clock frequency) and the low (high) effort configurations $Config_L$ ($Config_H$) (discussed in Section 3.2) to compute the low (high) effort delays D_L (D_H). Additionally, it

also computes the delay of low-high effort combination using the F_L, F_H, D_L and D_H values as shown in Fig. 5. The $D_L \times F_H$ term in the delay computation accounts for the re-computation overhead. The energy is obtained by multiplying the power with the delay of the effort combination.

Entropy Computation Overhead We find that entropy computation (Equation 3) in the ZynQ MPSoC PS takes 0.03ms per image which is $< 0.05\%$ of the inference delay and thus, can be ignored.

4 EXPERIMENTS AND RESULTS

4.1 Experimental Setup

Datasets and ViTs: We benchmark all our results on the standard Imagenet-1K dataset using state-of-the-art efficient ViTs such as DeiT [14] and LV-ViT [7]. **Baseline:** For all experiments, the baseline is a ViT model without any effort modulation *i.e.,* all ViT attention modules will be activated irrespective of the input difficulty. **PIVOT-Optimized ViTs:** For ease of expression, throughout the text, we will refer to PIVOT-optimized DeiT-S and LVViT-S ViT as PVDS and PVLS, respectively.

Traning Details: In PIVOT, for the DeiT-S and LVViT-S ViTs, we create 7 (3, 4, 5, 6, 7, 8 and 9) and 9 (4, 5, 6, 7, 8, 9, 10, 11, 12) efforts, respectively. Each effort is finetuned for 30 epochs with the full training data. The ViTs are trained with 8-bit quantization. Training all the efforts is 3× (2×) more efficient than training a DeiT-S (LVViT-S) ViT from scratch (see Fig. 4c). For training we use Pytorch 1.3.1 with a single Nvidia V100 GPU backend.

Hardware Evaluation: All baselines and PIVOT-optimized ViTs (PVDS and PVLS) are evaluated using the PIVOT-Sim framework. The FPGA implementation parameters for PIVOT-Sim are shown in Table 1. The FPGA implementation requires 4566 LUTs, 20668 Registers, 48 Block RAMs and 2304 digital signal processing cores.

FPGA Board	Xilinx ZCU102
Global SRAM (GB) Size	16KB
IPMEM, WTMEM, OPMEM	64Kb, 64Kb, 64Kb
PE Array Size	64×36
Clock Frequency	125MHz
Dataflow	Input Stationary

Table 1: Table showing the FPGA implementation parameters.

4.2 Results on DeiT-S and LVViT-S ViTs

Table 2: Table comparing the performance of DeiT-S and PIVOT-optimized DeiT-S ViTs (PVDS-N) sampled at delay=N.

Model	Energy (J)	Delay (ms)	Power (W)	EDP (J×ms)	FPS/W	Accuracy (%)
DeiT-S	0.47	59.66	7.92	28.19	2.14(1×)	79.8
PVDS-50	0.38 (1.23×)	48.47 (1.23×)	7.92	16.21 (1.73×)	2.7(1.23×)	79.4
PVDS-35	0.292 (1.62×)	36.9 (1.61×)	7.92	10.5 (2.6×)	3.4(1.61×)	78.2

Table 3: Table comparing the performance of LVViT-S and PIVOT-optimized LVViT-S ViTs (PVLS-N) sampled at delay=N.

Model	Energy (J)	Delay (ms)	Power (W)	EDP (J×ms)	FPS/W	Accuracy (%)
LVViT-S	0.63	79.55	7.92	50.8	1.57(1×)	82.8
PVLS-50	0.410 (1.57×)	50 (1.6×)	7.92	20.13 (2.7×)	2.51(1.6×)	82.6
PVLS-35	0.312 (2.17×)	36.5 (2.17×)	7.92	10.57 (4.5×)	3.4(2.17×)	81.1

Table 2 and Table 3 compare the delay, energy-delay-product (EDP), energy efficiency (FPS/W) and the accuracy of different PVDS and PVLS ViTs searched at different target delays lesser than the baseline. Evidently, as seen in Table 2 the PVDS-50 (PVLS-50) ViTs achieve 1.73× (2.7×) EDP reduction, 1.23× (1.6×) higher FPS/W with merely 0.4% (0.2%) accuracy reduction compared to the baseline DeiT-S (LVViT-S). At a slightly higher accuracy reduction of 1.6% (1.7%) the PVDS-35 (PVLS-35) yields 2.6× (4.5×) lower EDP and 1.62× (2.17×) higher FPS/W compared to baseline.

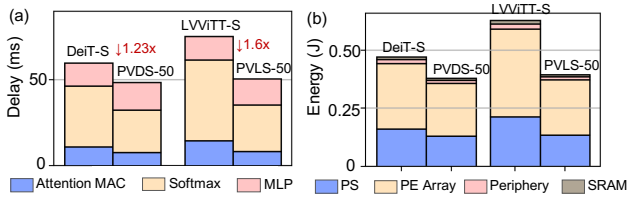

Figure 6: (a) Delay breakdown across encoder modules for different ViTs (b) Energy breakdown across the PE Array, Periphery and SRAM (part of the ZynQ MPSoC PL) and the PS (ZynQ MPSoC PS).

Fig. 6a shows the delay distributions across the Attention MAC (QKV, QK^T, (SMxV) and Proj), Softmax and MLP modules (refer Fig. 1a). Interestingly, the softmax module consumes 60% (63%) of the overall delay in the DeiT-S (LVViT-S) ViTs. With PIVOT, the softmax overhead reduces to 43% (48%) for the PVDS-50 (PVLS-50) ViTs. Similarly, the Attention MAC overhead reduces to 13% (14%) in the PVDS-50 (PVLS-50) ViTs compared to 18% (19%) in DeiT-S (LVViT-S) ViTs. Note, since PIVOT does not skip MLP modules, the delay overhead of MLP in PVDS-50 (PVLS-50) increase by 21% (19%) compared to the baselines due to the re-computation overhead (refer Section 3.1). However, due to high delay reduction in softmax and attention MAC modules, PIVOT achieves an overall delay reduction.

Energy Reduction across FPGA Resources: As seen in Fig. 6b, delay reduction in PVDS-50 and PVLS-50 ViTs lead to an energy reduction across the ZynQ MPSoC PS and PL systems. PVDS-50 and PVLS-50 ViTs achieve around 2× energy reduction in the PS and 1.6×, 1.7× and 1.8× energy reduction in the PE-Array, SRAM memories and peripheral circuits, respectively implemented on the ZynQ MPSoC PL (See Section. 3.4). The peripheral circuits (periphery) include PS-PL interconnects, reset and memory controllers.

4.3 Comparison with Prior Works

Table 4: Performance comparison of ViTCOD [17], HeatViT [4] and PVDS-50.

Work	ViTCOD [17]	HeatViT [4]	**PIVOT (Ours)**
ViT Backbone	DeiT-S	DeiT-S	**DeiT-S**
Effort Modulation	Constant	Constant	**Input-aware**
Prediction Mechanism	Norm Score	Head Level	**Entropy Metric**
Quantization	8-bits	8-bits	**8-bits**
Accuracy	78.1%	79.1%	**79.4%**
GPP Compatible	×	×	✓

Table 4 performs a holistic comparison between PIVOT and prior state-of-the-art algorithm-hardware co-design frameworks [4, 17]. Soft token pruning in HeatViT [4] achieves a high token pruning ratio of 40%, 74% and 87% in encoders 4-6, 7-9, and 10-12, respectively, while achieving 79.1% accuracy. ViTCOD [17] achieves 90% attention sparsity ratio at 78.1% accuracy. **Accuracy advantage in PIVOT:** HeatViT [4] and ViTCOD [17] do not modulate their efforts based on the input difficulty (token and attention sparsity ratios remain constant for all inputs). Therefore, at high token and attention pruning ratios, the accuracy suffers as difficult images are wrongly classified. Whereas, due to input-awareness, PIVOT (PVDS-50) achieves the highest accuracy of 79.4%.

Evaluation on GPPs: As HeatViT [4] and ViTCOD [17] require special hardware support for efficient implementation, we perform the delay comparison on GPPs such as CPUs- Intel Xeon, Raspberry

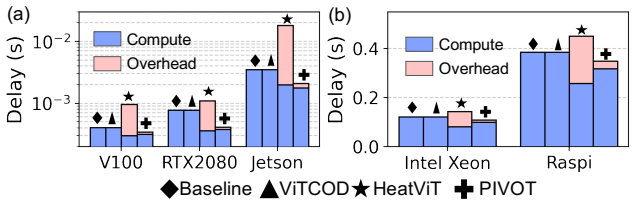

Figure 7: Compute and overhead delay breakdowns for DeiT-S baseline, HeatViT [4], ViTCOD [17] and PIVOT (PVDS-50) across (a) Nvidia V100, NVidia RTX2080ti and Nvidia Jetson Orin Nano (b) Intel Xeon and Raspberry Pi 4.

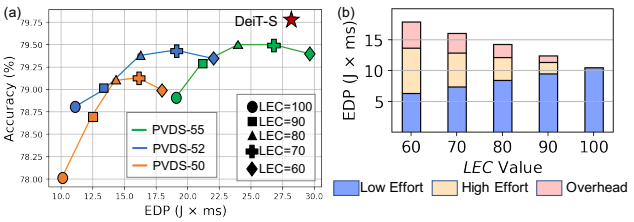

Figure 8: Figure analysing the effect of different *LEC* on the EDP and accuracy for different effort combinations. (b) EDP distribution between the low effort, high effort and the re-computation overhead (Overhead) for the PVDS-50 ViT.

Pi, and GPUs- Nvidia V100, Nvidia RTX2080ti and Nvidia Jetson Orin Nano for a fair comparison. As seen in Fig. 7a and Fig. 7b, the PIVOT (PVDS-50) achieves around 1.2-1.5× lower delay compared to the baseline across all GPPs. Since ViTCOD requires sparse matrix multiplication support, the delay on GPP is similar to the baseline. Due to hefty predictor networks and token packaging modules for soft token pruning, HeatViT [4] entails significant delay overhead when implemented on GPPs. PIVOT is general purpose and entails a small overhead of 6% in the delay. This delay is majorly contributed by the re-computation overhead. The contribution of entropy computation (Equation 3) is negligibly small (< 0.05%).

4.4 Analysis with *LEC* Constraints

From Fig. 8 we find that $LEC = 70$ and $LEC = 80$ attain the best EDP and accuracy tradeoff across different PVDS ViTs. At low $LEC = 60$, the EDP is high as merely 60% of the inputs are classified by the low effort. Additionally, $LEC = 90$ entails 90% of the inference with low effort but this leads to a significant accuracy degradation.

The EDP is contributed by the low effort and high effort inference, and the re-computation overhead (Section 3.1). At low *LEC* values, both high-effort and re-computation EDPs are high while the low effort EDP is less. As the *LEC* value increases, the low effort EDP increases marginally while the high effort and re-computation EDP reduce significantly leading to overall low EDPs.

Need for Input difficulty awareness As seen in Fig. 8a for $LEC = 100$, all inputs are inferred by the low effort. This leads to low EDP at the cost of accuracy since the efforts are not modulated for difficult inputs. Therefore, PIVOT's input-aware effort modulation achieves optimal accuracy-efficiency tradeoffs.

4.5 Efforts Combinations for Different Delays

Figure 9: Different PVDS ViTs sampled by PIVOT at different delay constraints.

As seen in Fig. 9, reduction in the delay requirement lowers the number of active attentions in the ViT. The efforts shown here represent the optimal path with the highest *Path-score* for each effort. Interestingly, we observe that across all efforts, attentions skipping is preferred in the deeper layers as the $CKA(MLP, A)$ value is higher in the latter layers.

5 CONCLUSION

PIVOT motivates ViT attention optimization in an input difficulty-aware manner. PIVOT's input-awareness yields 0.4%-1.3% higher accuracy compared to prior token pruning and attention sparsification works. Unlike prior works, PIVOT is GPP compatible and yields 1.2-1.5× higher throughput compared to baseline ViT across different CPU/GPU platforms. Additionally, PIVOT-Sim- an end-to-end open source FPGA-based evaluation platform is developed that will motivate future ViT-hardware co-optimization works.

ACKNOWLEDGEMENT

This work was supported in part by CoCoSys, a JUMP2.0 center sponsored by DARPA and SRC, the National Science Foundation (CAREER Award, Grant #2312366, Grant #2318152), and the DoE MMICC center SEA-CROGS (Award #DE-SC0023198)

REFERENCES

[1] Bhattacharjee et al. 2022. MIME: adapting a single neural network for multi-task inference with memory-efficient dynamic pruning. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 499–504.
[2] Cortes et al. 2012. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research* 13, 1 (2012), 795–828.
[3] Dehghani et al. 2023. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*. PMLR, 7480–7512.
[4] Dong et al. 2023. Heatvit: Hardware-efficient adaptive token pruning for vision transformers. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 442–455.
[5] Dosovitskiy et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[6] Han et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.
[7] Jiang et al. 2021. All tokens matter: Token labeling for training better vision transformers. *Advances in neural information processing systems* 34 (2021), 18590–18602.
[8] Kim et al. 2021. Rethinking the self-attention in vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3071–3075.
[9] Li et al. 2023. Input-aware dynamic timestep spiking neural networks for efficient in-memory computing. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[10] Panda et al. 2016. Conditional deep learning for energy-efficient and enhanced pattern recognition. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 475–480.
[11] Rao et al. 2021. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems* 34 (2021), 13937–13949.
[12] Samajdar et al. 2018. Scale-sim: Systolic cnn accelerator simulator. *arXiv preprint arXiv:1811.02883* (2018).
[13] Stevens et al. 2021. Softermax: Hardware/software co-design of an efficient softmax for transformers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 469–474.
[14] Touvron et al. 2021. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*. PMLR, 10347–10357.
[15] Wang et al. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 97–110.
[16] Wu et al. 2018. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8817–8826.
[17] You et al. 2023. Vitcod: Vision transformer acceleration via dedicated algorithm and accelerator co-design. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 273–286.