# Token Pruning using a Lightweight Background Aware Vision Transformer

**Sudhakar Sah, Ravish Kumar, Honnesh Rohmetra, Ehsan Saboori**
Deeplite, Inc.
Toronto, Canada
sudhakar@deeplite.ai

## Abstract

High runtime memory and high latency puts significant constraint on Vision Transformer training and inference, especially on edge devices. Token pruning reduces the number of input tokens to the ViT based on importance criteria of each token. We present a Background Aware Vision Transformer (BAViT) model, a pre-processing block to object detection models like DETR/YOLOS aimed to reduce runtime memory and increase throughput by using a novel approach to identify background tokens in the image. The background tokens can be pruned completely or partially before feeding to a ViT based object detector. We use the semantic information provided by segmentation map and/or bounding box annotation to train a few layers of ViT to classify tokens to either foreground or background. Using 2 layers and 10 layers of BAViT, background and foreground tokens can be separated with 75% and 88% accuracy on VOC dataset and 71% and 80% accuracy on COCO dataset respectively. We show a 2 layer BAViT-small model as pre-processor to YOLOS can increase the throughput by 30% - 40% with a mAP drop of 3% without any sparse fine-tuning and 2% with sparse fine-tuning. Our approach is specifically targeted for Edge AI use cases. Code and data are available at [Link].

## 1 Introduction

Transformers (31) have already demonstrated their ability to outperform traditional methods in Natural Language Processing (NLP) with models like BERT (5) and RoBERTa (19). They are now commonly used in modern vision-related tasks such as classification (20), object detection (2) (16) (9), segmentation (30), and pose estimation (33) as Vision Transformers(ViT). Despite the advantages of ViTs over traditional CNN-based approaches, their high computational requirements pose significant challenge in deployment of these models on edge devices with limited memory and computational power. The ViT accepts small image patches (typically $16 \times 16$ size) called *tokens* as input. As image resolution increases, more input tokens are generated, which increases the performance but reduces model throughput and latency.
ViT is also used for object detection by models like DETR (2) which uses learnable queries and encoder features to produce box predictions using decoder. Different variations of DETR-like models like (22)(16), (9) are proposed to create state of the art object detection models.

Zheng et.al (35) showed that the complexity of Deformable DETR (36) is $8.8\times$ compared to the decoder which suggests that focusing on efficiency of the encoder is very important. All the tokens do not have same importance and by reducing the number of tokens results in latency and throughput improvement. The technique to reduce the number of tokens by assessing the importance or relevance of each token is called *token pruning*. In this work, we aim to reduce the number of input tokens by introducing a novel token importance criteria for pruning with a minimal impact on performance. Our approach uses segmentation masks provided in the COCO (80 object categories) (15) and PASCAL

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

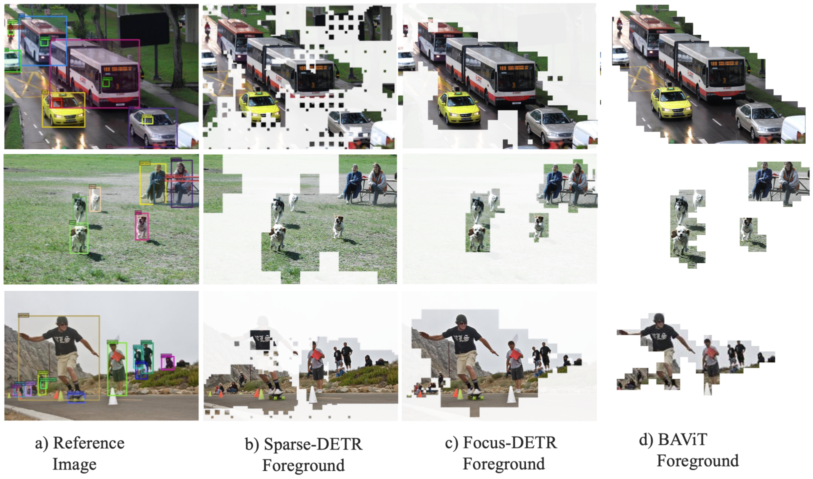

Figure 1: Comparison of background token identification results between Sparse DETR, Focus DETR and BAViT models

VOC (20 object categories) (6) datasets to annotate each individual patch as foreground (FG) or background (BG). This annotation serves as a guide for ViT models in object detection tasks to determine the importance of each token. Sparse DETR (27) and Focus DETR (35) are two most impressive and state of the art techniques for token pruning. As show in Figure 1, sparse DETR uses the token importance score by computing cross-attention map in the decoder which reduces the number of tokens by 70%. Focus DETR (35) on the other hand detects background tokens and prunes them to increase the throughput. We propose a method that uses background token detection similar to focus DETR but we our target usecase is Edge devices so we avoid using heavy CNN backbones, as proposed in Focus DETR, to detect background tokens which could be computationally very expensive.

We summarize our contributions as follows:

- We introduce a novel Background Aware Vision Transformer (*BAViT*) capable of separating FG and BG tokens
- We introduce a modified Accumulative Cross Entropy Loss function for BG/FG classification.
- We demonstrate that integration of *BAViT* as pre-processing block of DETR/YOLOS like object detection model provides a good latency/accuracy trade-off and increased throughput of the model.

## 2 Related Work

### 2.1 Vision Transformers

Transformers (31) have emerged as a dominant architecture in NLP (5) (19) as well as vision-related tasks (22) (9) and these models (21) have achieved state-of-the-art performance for vision tasks including object detection such as DETR (2), RT-SETR(22) and YOLOS(9). DETR (2) employs a combination of CNN-based backbones followed by transformers to address object detection tasks. Swin-transformers (20) introduced new ViTs that can serve as general-purpose backbones for computer vision tasks. WBDetr(16) replaced the CNN-based backbones in DETR (2) with a transformer-based backbone for object detection. Similarly, innovations continue to enhance ViT capabilities, such as (12), which introduces a K-dimensional score map to provide localized information about image patches. Recent work by Fang et al. (9) proposes end to end object detection as sequence-to-sequence task. Our BAViT proposes an additional information about of these image tokens as BG and FG, which can be integrated as the pre-processing stage to filter out unnecessary patches.

### 2.2 Runtime Memory Improvement

ViTs (21) require substantial runtime memory, which limits their use on smaller devices. Many research efforts, including (3) (29) (34), propose methods to optimize the performance of vision transformers. Reformer (14) introduces architectural changes to the residual layers, replacing them with reversible residual layers to make the model more efficient. Sparse attention(3) proposes an alternative attention formulation through sparse factorization of the attention matrix, which is one of the most computationally expensive components in ViTs. Sparse Detr(27) enhances the efficiency of DETR (2)-like models by substituting dense attention with deformable attention. Other works, such as (7), replace standard dropout layers with structured dropout layers to improve the efficiency and robustness of transformers. Few methods focus on pruning heads by ranking them based on their estimated importance (32). Additionally, quantization approaches (1) (28) (8) have been explored to further improve the efficiency of ViTs.

### 2.3 Token Pruning

The number of tokens contribute to quadratic complexity in ViTs during inference. However, all the tokens generated from the input image are not equally important; many primarily contain background information. Several research efforts, including (24) (18) (17) (27) (35), propose efficient approaches to remove unnecessary tokens, thereby improving the inference time. (35) introduces a technique that efficiently scores the importance of tokens, discards background queries, and enhances the semantic interaction of fine-grained object queries based on these scores. (17) proposes an adaptive method to hierarchically discard useless tokens and adjust computational costs for different input instances. (18) suggests reusing pruned tokens at later stages of the model. Our work is very close to Focus DETR (35) as both approaches focus on classifying tokens into FG and BG. However, Focus DETR uses a heavy backbone from DETR (like ResNet50, ResNet101 (10)) which is not suitable for edge devices. Also, Focus DETR proposes many modifications to the existing DETR model which requires model retraining or fine-tuning for a long time. Therefore, although the technique produces SOTA results, it is not feasible approach for edge devices. Our work proposes a simpler strategy for background token identification using a learnable small ViT model using 2 layers. Also, our approach produces foreground images which visibly looks very similar to Focus DETR produced foreground images but our approach uses a very small model, compared to Focus DETR, to achieve this. BAViT can be used as a separate module and integrated with other models at the pre-processing data stage, enabling faster performance and making the models suitable for smaller devices. Our target use case is small ViTs for edge devices, therefore it is difficult to compare our method with Focus DETR mAP/latency numbers which uses very large model and performs latency experiments on larger GPUs.

## 3 Methodology

### 3.1 Auxiliary Annotations

Transformers accept image patches (called tokens) of size ($k \times k$), created by dividing the input image into a sequence of square patches, as shown in Figure 2. ViTs use these patches to classify objects in the image through the attention mechanism. Popular datasets like Microsoft COCO (15) and Pascal VOC (6), used for object detection and segmentation tasks, contain annotations such as bounding boxes and instance segmentation maps. We create a M-dimensional patch annotation vector for every input image, where $M$ represents the total number of tokens formed by dividing the input image into $k \times k$ smaller non-overlapping patches as shown in Figure 2. We compare the Jaccard similarity coefficient (26) of each token with all the bounding boxes or segmentation map and it is labeled as one (Foreground - FG) if the overlap of a token with any of the bounding box is more than 0.5, otherwise it is labeled as zero (Background - BG) as shown in Equation 1 and Equation 2. Figure 2 shows a sample Pascal VOC image (left), bounding boxes (center), and image patches with BG patches in gray and FG patches in red color. When using segmentation maps to create the annotation vector, any image patch with more than 10% overlapping pixel with any class of segmentation map is considered foreground; otherwise, it is considered background. We trained BAViT model both using bounding box annotations and segmentation maps but most of the results presented in this paper are

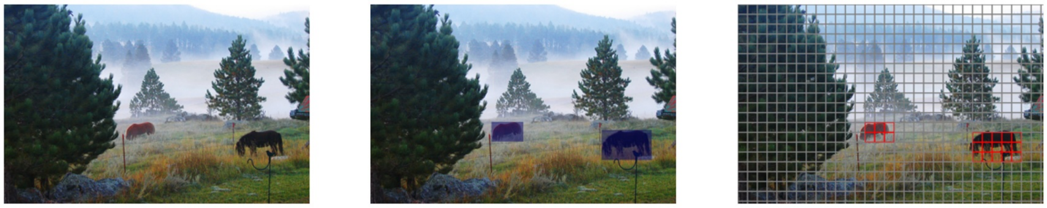

Figure 2: Three VOC images (left to right) a) original image b) foreground object area in transparency c)$16 \times 16$ grid with red grids bring foreground and gray being background

from annotated data using segmentation map.

$$\mathbf{L_i} = \begin{cases} 1 & \{\text{if } J(P_i, B_j) \geq \tau \\ 0 & \{\text{if } J(P_i, B_j) < \tau \end{cases} \tag{1}$$

$$\mathbf{J}(\mathbf{P}_i, \mathbf{B}_j) = \frac{|P_i \cap B_j|}{|P_i \cup B_j|} \tag{2}$$

where $P_i$ is patch and $B_i$ is bounding box, $L_i$ is assigned label for $i^{\text{th}}$ token, $\mathbf{J}(P_i, B_j)$ is Jaccard coefficient, $\tau$ is threshold for selecting the token as foreground or background.

### 3.2 BAViT Architecture

BAViT architecture is created by introducing few fundamental changes in the traditional ViT architecture as illustrated in Figure 3 (left). We remove the CLS token and introduce a linear layer with two output classes for each token. Traditional ViT uses CLS token to encapsulate knowledge from all tokens and it provides the score for each class. On the contrary, BAViT calculates classification score for FG and BG classes for each token. Therefore, we do not need a CLS token. Accumulative Cross Entropy Loss ($L_{acc}$) is calculated as defined in equation 3, and weights are updated via back propagation. This loss function can also be used with other loss functions targeting different vision tasks, such as object detection loss to help the model focus on important tokens. Since BAViT is supposed to be used as pre-processing step for token pruning, we decided to keep it light weight and used the model with only 2 layers (BAViT-small) to study the impact on YOLOS (9). However, we have provided BG/FG classification results with 10 layers as well (BAViT-large) as BAViT-small in result section to show the scalability and flexibility of this approach.

### 3.3 Accumulative Cross Entropy Loss

In contrast to the traditional ViT classifier training, which involves introducing an additional classification token (CLS) and calculating loss only for that token, we propose a new loss function that calculates the Cross Entropy Loss (23) for each token individually and then aggregates these losses. This aggregated loss is termed as Accumulative Cross Entropy Loss ($L_{acc}$), as defined in 3.

$$\mathbf{L_{acc}} = -\frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{c=1}^{C} y_{i,j,c} \log(\hat{y}_{i,j,c}) \tag{3}$$

where $N$ is the the number of image samples, $M$ is the number of tokens per sample, $C$ is the number of classes (background and foreground). $y_{i,j,c}$ is the variable indicating whether the $j$-th token in the $i$-th sample belongs to class $c$. It's value is one if the token belongs to class $c$, otherwise it is zero. $\hat{y}_{i,j,c}$ is the predicted probability of the $j$-th token in the $i$-th sample being in class $c$.

### 3.4 Model Training

We use both Pascal VOC (6) and COCO 2017 (15) to train BAViT and reported mAP (mean Average Precision) result on the validation dataset for both. Each training batch, denoted as ($B$, $M$, $S$),

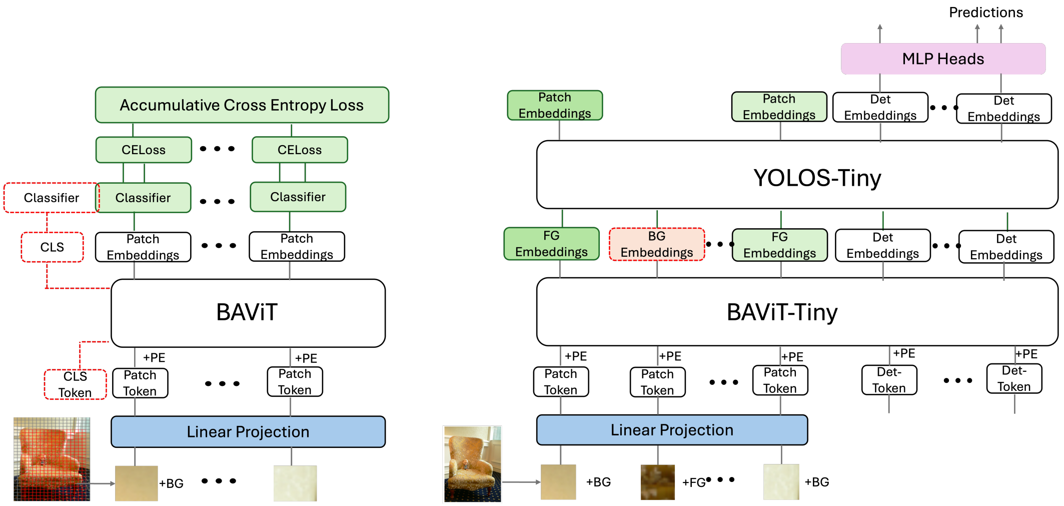

Figure 3: Background aware ViT architecture. (Left) 2 layers for BG and FG patch classification. (Right) BAViT attached as a pre-processing step to YOLOS (DETR type object detector) object detector

consists of $M$ tokens of size $16 \times 16$, each with an embedding size of $S = 192$, and labels for each token indicating either *BG* or *FG*. We employed the Adam (13) optimizer with a step learning rate scheduler and trained the model for 100 epochs until convergence. The initial weights for the ViT (25) model were loaded from ImageNet-1k (4) dataset pre-trained model.

### 3.5 BAViT Integration with ViT based Detection

The BAViT-small is added as a pre-processing block of the ViT based object detector as shown in Figure 3. We have used YOLOS (9) as the object detection model , an architecture similar to DETR(2) with an exception that YOLOS provides an option to use the detector without a CNN backbone. Our method works directly on image tokens, so it cannot be applied to a CNN backbone based ViT object detectors. The BAViT model works on $384 \times 384$ input and YOLOS (tiny) expects $512 \times 512$ inputs to achieve the benchmark mAP. BAViT outputs the classification of each token as BG or FG with a total of 576 tokens but the YOLOS model expects 1024 tokens so we upscale the tokens labels from 576 to 1024 keeping the relative BG/FG patch position same. After the label scaling step, each of 1024 token is classified as BG or FG token. The YOLOS model only computes the FG tokens from first to the final layer. We also modify YOLOS model slightly so that it does not compute anything for the BG tokens and return zeros as the final output token for these tokens. All the FG tokens are processed in the usual manner. So, the modified BAViT + YOLOS-tiny model contains 14 layers, first 2 layers of BAViT and the 12 layers of YOLOS-tiny.

## 4 Results

### 4.1 BG/FG Classification Model

The BAViT model was trained with both 2 layers (BAViT-small) and 10 layers (BAViT-large) depth. Table 1 displays the token classification accuracy of these models on different datasets. BAViT-small is used for integration with object detection model (YOLOS) but we also trained the BAViT-large model to assess the impact on model accuracy. We found that BAViT-small achieved 75.93% accuracy, which was reasonable compared to BAViT-large's 88.79% accuracy for the BG/FG classification task on VOC dataset given the difference in number of parameters for these two models. We also trained both models on COCO dataset as shown in Table 1 and used BAViT-small trained with COCO with mAP 70.88% as pre-processing block . YOLOS-tiny model has 6.5M parameters using 18.8 GFLOPS. Addition of BAViT-small over the native YOLOS-tiny marginally increases the total number of parameters (by 1.49M) and FLOP counts (+1.961 GFLOPS) but substantially reduced the amount of total number of tokens (25.63%) which has the quadratic impact over the computational

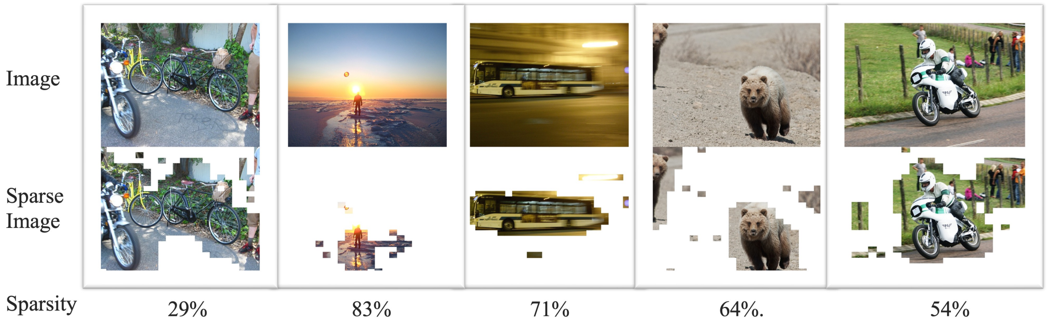

Figure 4: FG/BG token classification (16x16) on COCO images. Top- original image, bottom - sparse image generated from BAViT with sparsity percentage.

complexity of the ViT models. On the other hand focus DETR (35) models with ResNet50 backbone has 48M parameters using GFLOPS which is almost 8x times bigger and slower. Our results also suggest that it can be applied to different datasets with configurable number of layers based on latency and RAM constraints.

Table 1: Accuracy of FG/BG classification on different models trained on different datasets and with different number of layers.

| **Model** | **Depth** | **Dataset** | **Accuracy(%)** |
|---|---|---|---|
| BAViT-small | 2 | Pascal-VOC | 75.93 |
| BAViT-large | 10 | Pascal-VOC | 88.79 |
| BAViT-small | 2 | MS-COCO | 70.88 |
| BAViT-large | 10 | MS-COCO | 80.57 |

Figure 4 shows the BAViT-large model output for COCO images where top image is the original image and bottom image is sparse image with all background patches shown in white color. There are few misclassified tokens where background is classified as foreground and vice versa. Foreground being classified as background is concerning so we added additional post processing block to improve the classification accuracy as explained in Appendix section. It is evident that our model is able to separate FG/BG patches effectively even with multiple objects from different classes. It is also clear from these images that sparsity varies based on the image and it can be even more than 70% in many images. COCO images have an average of 40% of background tokens which means that only 60% tokens are important.

### 4.2 Token reduction using BAViT

As explained in section 3.5, we added BAViT-tiny to pre-process the image and classify each patch as FG or BG tokens before passing to YOLOS model. Using FG patches for all computation and ignoring all the BG patches, we can reduce the number of tokens in YOLOS-tiny model drastically. Equation 4 shows the calculation used to calculate the average reduction in tokens for 5000 COCO validation images. Table 2 shows BAViT model used with different level of sparsity for token pruning and the impact on mAP due to the same. The sparsity is controlled by modifying the confidence threshold of background tokens. Our BAViT model adds extra complexity to the overall model but since this model has very low complexity and it works at much lower resolution , the overall number of token is less than the original model. for eg. the model with 34% sparsity reduces total tokens by 24% with an accuracy drop of 2.6% on COCO dataset. Please note that we are not demonstrating the results of fine-tuning for most of these models. However, we have finetuned one of the model with 35% sparsity and could improve the accuracy by 2 mAP points. It is important to note that we fine-tuned the model only for 30 epochs to improve the accuracy.

Although our method suffers a drop in mAP due to sparsification, it is still applicable to edge use cases whereas solution proposed in methods like Sparse DETR (27) and Focus DETR (35) can't be used. Focus DETR, being the SOTA in token pruning field, uses ResNet50 and ResNet101 backbones to detect background tokens, which makes it impractical for edge use cases with very limited memory and computational capabilities. Also, Focus DETR proposes significant changes in the model architecture which necessitates the model to be retrained which is very expensive. BAViT on the other hand does not need model retraining, whereas to compensate the drop in mAP due to sparsification, it can be fine-tuned for lesser number epochs (30 epochs is used for our experiments).

$$\text{Token Reduction} = \sum_{i=1}^{n} \frac{Ty_i - (Tb_i + Ty_i \cdot s)}{N} \tag{4}$$

where $Ty_i$ is total YOLOS tokens for $i^{\text{th}}$ image, $Tb_i$ is total BAViT tokens for $i^{\text{th}}$ image, s is sparsity percentage in $i^{\text{th}}$ image and N is total number of images.

Table 2: Token reduction using BAViT as a pre-processing block to YOLOS-tiny model. Total number of tokens for 2 layers of BAViT is 1152 (576 tokens per layer for $384 \times 384$ input ) and total number of tokens for 12 layers of BAViT is 12288 (1024 tokens per layer for $512 \times 512$ input). BAViT+YOLOS-F is the fine-tuned YOLOS model using only Foreground tokens (30 epochs)

| | | | **Number of Tokens** | | | | |
|---|---|---|---|---|---|---|---|
| **Model** | **Sparsity %age** | **mAP** (COCO) | **BAViT** | **YOLOS** | **YOLOS Pruned** | **YOLOS +BAViT** | **%Reduction** |
| BAViT+YOLOS | 46% | 20.00 | 1152 | 12288 | 6635 | 7787 | 36.63% |
| BAViT+YOLOS | 43% | 21.50 | 1152 | 12288 | 7004 | 8156 | 33.63% |
| BAViT+YOLOS | 40% | 22.50 | 1152 | 12288 | 7372 | 8524 | 30.63% |
| BAViT+YOLOS | 39% | 22.70 | 1152 | 12288 | 7495 | 8647 | 29.63% |
| BAViT+YOLOS | 37% | 23.80 | 1152 | 12288 | 7741 | 8893 | 27.63% |
| **BAViT+YOLOS** | 35% | **24.40** | 1152 | 12288 | 7987 | 9139 | 25.63% |
| **BAViT+YOLOS-F** | 35% | **26.60** | 1152 | 12288 | 7987 | 9139 | 25.63% |
| BAViT+YOLOS | 32% | 25.00 | 1152 | 12288 | 8355 | 9507 | 22.60% |
| BAViT+YOLOS | 29% | 25.90 | 1152 | 12288 | 8724 | 9876 | 19.60% |
| BAViT+YOLOS | 5% | 27.70 | 1152 | 12288 | 11673 | 12825 | -4.37% |
| BAViT+YOLOS | 2% | 28.60 | 1152 | 12288 | 12042 | 13194 | -7.38% |
| BAViT+YOLOS | 0% | 28.80 | 1152 | 12288 | 12288 | 13440 | -9.40% |

## 5 Conclusion

In this work, we introduced a novel method for separating BG/FG patches in images by leveraging existing annotations from bounding boxes and segmentation maps to create localized annotations. We applied these annotations within a token classification training strategy, achieving an accuracy of up to 88.79% on the Pascal VOC dataset and 80.57% on the COCO dataset using a 10-layer transformer model. Notably, even with just 2 transformer layers, we were able to achieve over 75% accuracy on Pascal VOC and 70% on COCO dataset respectively. We also used BAViT-small model for pre-processing step to prune tokens of a YOLOS-tiny model. Our approach could reduce the number of tokens by 25% with a mAP drop of 3% on COCO dataset. This drop is shown to be recovered (less than 2% mAP drop) by sparse token finetuning by using just 30 epochs. BAViT approach is a low cost and low complexity alternative to SOTA methods like Focus DETR (35) which works on large models not fitting on edge devices. Future work involves integrating our approach to YOLOS type of model to jointly train BG/FG classifier and object detector together to observe the accuracy-latency trade-off. Additionally, we also aim to achieve adaptive sparsity based on input image complexity, with a learnable threshold parameter similar to (17).

## References

[1] Bhandare, A., Sripathi, V., Karkada, D., Menon, V., Choi, S., Datta, K., Saletore, V.: Efficient 8-bit quantization of transformer neural machine language translation model. arXiv preprint arXiv:1906.00532 (2019)

[2] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)

[3] Child, R., Gray, S., Radford, A., Sutskever, I.: Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509 (2019)

[4] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)

[5] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[6] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results (2012), `http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html`

[7] Fan, A., Grave, E., Joulin, A.: Reducing transformer depth on demand with structured dropout. arXiv preprint arXiv:1909.11556 (2019)

[8] Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., Joulin, A.: Training with quantization noise for extreme model com- pression. arXiv preprint arXiv:2004.07320 (2020)

[9] Fang, Y., Liao, B., Wang, X., Fang: You only look at one sequence: Rethinking transformer in vision through object detection. Advances in Neural Information Processing Systems **34**, 26183–26197 (2021)

[10] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[11] He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., Chao, Y.: The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition **70**, 25–43 (2017)

[12] Jiang, Z.H., Hou, Q., Yuan, L., Zhou, D., Shi, Y., Jin, X., Wang, A., Feng, J.: All tokens matter: Token labeling for training better vision transformers. Advances in neural information processing systems **34**, 18590–18602 (2021)

[13] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)

[14] Kitaev, N., Kaiser, Ł., Levskaya, A.: Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451 (2020)

[15] Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft coco: Common objects in context (2014), cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list

[16] Liu, F., Wei, H., Zhao, W., Li, G., Peng, J., Li, Z.: Wb-detr: Transformer-based detector without backbone. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2979–2987 (2021)

[17] Liu, X., Wu, T., Guo, G.: Adaptive sparse vit: Towards learnable adaptive token pruning by fully exploiting self-attention. arXiv preprint arXiv:2209.13802 (2022)

[18] Liu, Y., Gehrig, M., Messikommer, N., Cannici, M., Scaramuzza, D.: Revisiting token pruning for object detection and instance segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2658–2668 (2024)

[19] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)

[20] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)

[21] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)

[22] Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., Liu, Y.: Detrs beat yolos on real-time object detection. arXiv preprint arXiv:2304.08069 (2023)

[23] Mao, A., Mohri, M., Zhong, Y.: Cross-entropy loss functions: Theoretical analysis and applications (2023)

[24] Rao, Y., Liu, Z., Zhao, W., Zhou, J., Lu, J.: Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(9), 10883–10897 (2023)

[25] Research, G.: Vision transformer. `https://github.com/google-research/vision_transformer/` (2023)

[26] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 658–666 (2019)

[27] Roh, B., Shin, J., Shin, W., Kim, S.: Sparse detr: Efficient end-to-end object detection with learnable sparsity. arXiv preprint arXiv:2111.14330 (2021)

[28] Shen, S., Dong, Z., Ye, J., Ma, L., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Q-bert: Hessian based ultra low precision quantization of bert. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 8815–8821 (2020)

[29] Tay, Y., Bahri, D., Yang, L., Metzler, D., Juan, D.C.: Sparse sinkhorn attention. In: International Conference on Machine Learning. pp. 9438–9447. PMLR (2020)

[30] Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R., Herath, D.: Semantic segmentation using vision transformers: A survey. Engineering Applications of Artificial Intelligence **126**, 106669 (2023)

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems (2017)

[32] Voita, E., Talbot, D., Moiseev, F., Sennrich, R., Titov, I.: Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. arXiv preprint arXiv:1905.09418 (2019)

[33] Xu, Y., Zhang, J., Zhang, Q., Tao, D.: Vitpose: Simple vision transformer baselines for human pose estimation. Advances in Neural Information Processing Systems **35**, 38571–38584 (2022)

[34] Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A.: Big bird: Transformers for longer sequences. In: Advances in Neural Information Processing Systems (2020)

[35] Zheng, D., Dong, W., Hu, H., Chen, X., Wang, Y.: Less is more: Focus attention for efficient detr. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6674–6683 (2023)

[36] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)

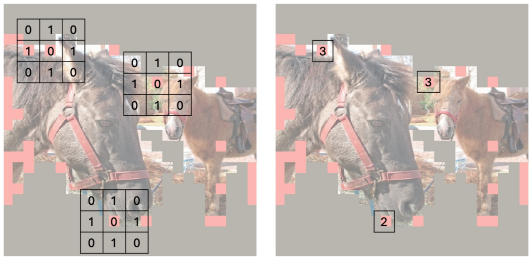

Figure 5: Converting BG tokens to FG tokens using a post processing convolution operation

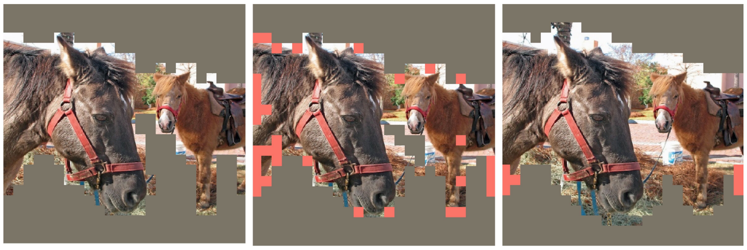

Figure 6: Left: Ground truth image (gray patch is background), Center : predicted BG/FG patches (gray patch is BG and orange patch is BG misclassified as FG, Right : Misclassified patches corrected by post-processing convolution operation

## Appendix

In this appendix, we provide additional details about post-processing algorithm applied to improve BG/FG classification results. Figure 6 shows the result of BAViT where orange patches are FG misclassified as BG and gray patches are correctly classified by the model. To minimize the error due to misclassified FG pixels, we use Connected Component Analysis (CCA) (11), the traditional graph analysis algorithm to connect nodes with connected neighbors. In this case, each patch is considered as a node of the graph and CCA is performed by applying a convolutional kernel (shown in Figure 5) on the graph (FG=1, BG=0) and converting the graph node from 0 to 1 for all pixels with convolution output greater than 2. The CCA algorithm is applied for few steps to minimize the classification error. More steps reduces classification error but it increases number of FG patches which were BG in the ground truth image. We found 3 steps to be optimal based on different experiments, impact on accuracy and efficiency. Right image in Figure 6 shows the result of our post processing convolution which brings the result very close to the ground truth. Please note that, we have not applied any post processing while reporting model's accuracy in 1 for fair evaluation. However,including the post processing convolution is expected to improve accuracy of the model significantly.