UG Research Project

Improving ViT Efficiency for MOT: Dynamic Tokenization

Benjamin Sanati

ECS, University of Southampton

September 30, 2022

Keywords
Deep Learning, Machine Learning, Computer Vision, Multiple Object Tracking, Transformer Architecture

1 Introduction

Transformers have recently begun to dominate the computer vision (CV) landscape. Beginning with [1], transformers were successfully used for CV tasks and consistently competed with/outperformed existing SOTA CV models. One of the most researched CV tasks is object detection, a task where many object detection transformer varieties dominate the leaderboards.

A major issue with transformers is the computational requirements of the model. Transformers are often very large models meaning large computational resources are required during inference. When it comes to vision transformers (ViTs), [2] find that the attention is sparse, meaning many of the tokens used in the model are not required and can be pruned. [2] use a prediction module that derives a halting score to determine the tokens to be pruned. Although these modules have been implemented on image data, they are yet to be investigated with video data.

This project will aim to implement a ViT-based object detector and deploy the model to a video feed. The performance of the model on the video feed will then be investigated.

As stretch goals, a dynamic token sparsification module will then be implemented into the ViT-based object detector and the performance improvements of the base object detectors would be investigated. This would then be integrated into a DBT (detection based tracker) to see the effect of the improvement of the object detector on a tracker as a whole.

2 Literary Review

2.1 Transformers

The transformer was initially devised as a model for NLP problems in [3]. It uses attention mechanisms to derive a representation of the sequence (self-attention) and allows the model to develop more subtle representations of the sequence by producing multiple attention scores for each word in the sequence (multi-headed attention). A transformer consists of an encoder and decoder stack, each of which contains multiple encoder/decoder layers, which create/use incorporated contextual information to generate an output sequence [4].

A transformer hinges on self-attention layers within its encoder and decoder stacks. Self-attention is a mechanism used to allow for the interaction between different inputs in an input sequence. This in-turn allows dependencies in the input sequence to be identified. A self-attention layer initially transforms the input into three different vectors: the query q, key k, and value v vectors. The attention function uses these vectors to acquire a weighted value matrix by translating normalized scores into probabilities and then matrix multiplying the probabilities with the value matrix, as shown in equation 1.

$$\text{Attention}(Q, K, V) = \sigma\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V \qquad (1)$$

This is a global process, therefore, the self-attention layer requires a positional encoding to be added to the input embedding. The positional encoding can be applied in one of two ways: (a) apply equation 2, with each element corresponding to a sinusoid, or (b) learn the positional encodings. Both methods allow the transformer to learn attention via relative positions, solving the model's positional invariance

issue.

$$\begin{aligned} \mathrm{PE}(pos, 2i) &= \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{model}}}}\right) \\ \mathrm{PE}(pos, 2i+1) &= \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{model}}}}\right) \end{aligned} \tag{2}$$

Each transformer block uses a multi-headed attention layer to improve the performance of the vanilla self-attention layer. Unlike RNNs [5] and LSTMs [6], transformers have the ability to take large-term dependencies into account. This is achieved via multi-headed attention as it processes entire sequences in parallel, instead of relying on past hidden states (like RNNs and LSTMs do). By using multiple attention heads, each input can achieve attention on different representation sub-spaces, thus both small and large-term dependencies are being taken into account.

Coupled with these attention mechanisms, feed-forward networks (FFN), layer normalization and residual connections are applied to the architecture, with the latter used to reduce the vanishing/exploding gradient effect.

2.2 Vision Transformers

Vision transformers [1] are transformers applied to image patches for CV tasks. A ViT splits up an image into patches of $p \times p$ dimensions and provides the sequence of linear embeddings of these patches as an input to a transformer encoder. Most ViTs only use the encoder stack, and discard the decoder stack. This removes the need for masked multi-headed self-attention, therefore, ViTs are non-autoregressive models. As ViTs are global mechanisms, ViTs have less inductive bias than CNNs. As such, they do not possess beneficial CV capabilities such as locality and translational equivariance, both of which are present in the CNNs convolution operation.

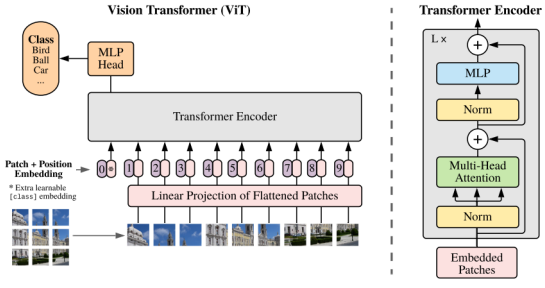

Figure 1: Source: [1]. ViT model architecture.

As a result, ViTs generalize to data much worse than CNNs for CV tasks, meaning huge pre-training tasks are required prior to training on downstream tasks. Although ViTs currently dominate major CV benchmarks, upon further inspection it can be observed that many of these models have been pre-trained on huge datasets (e.g. JFT-300M [7]) first, and as such have much worse training efficiency than equivalently performing CNNs. Additionally, papers like [8] show that ViTs do not outperform a CNNs inference efficiency when CNNs are modernized, especially in object detection and semantic segmentation tasks.

2.3 Dynamic Tokenization

Improving the inference efficiency of ViTs is a vital requirement for embedded ViT deployment. This is especially true for computationally difficult tasks like MOT. Adaptive computation techniques are used on many DNN (deep neural network) architectures to induce a desirable trade-off between the computational load of the model and the model's accuracy. The technique hinges on the idea that different inputs have different complexities, naturally progressing to the allocation of more computational resources for more complex inputs and the allocation of less computational resources for less complex inputs.

Dynamic tokenization is a technique that identifies spatial redundancy in images and prunes the tokens progressively based on the location of the spatial redundancies in the image and that the final predictions are only based on a small part of the input. This hinges on the idea that attention is sparse in ViTs, therefore, many tokens contribute little to the final prediction and are therefore not required.

[9] achieves this by pruning the redundant tokens in the input by using a lightweight prediction module to determine which tokens are redundant and should be pruned. The prediction module is placed after each encoder layer and provides a binary mask to the tokens, providing a sparse version of the current layers tokens as input to the next encoder layer. This pruning of tokens reduces the computational cost during inference for a small trade-off in prediction accuracy.

2.4 Multiple Object Tracking (MOT)

MOT is a computer vision task that includes object localization and the tracking of the trajectories of the objects within an input video [10]. A common paradigm used to perform an MOT task is Detection Based Tracking (DBT). Given a sequence of image frames from a video, a DBT

will first collect the sequential observations of all the objects from the first frame to the last frame with an object detector. Then an MOT tracker (SORT [11] or DeepSORT [12]) will link the objects to trajectories.

The premise of SORT is the positional tracking of objects with a Kalman filter in the image space and frame-by-frame object matching using the Hungarian algorithm with an association metric that measures bounding box overlap [11]. This has been shown to work well at high frame rates. The problem with SORT is that identity switches are common on a frame-by-frame basis due to their inefficiency with occlusions [12]. This problem is solved with DeepSORT by replacing the association metric with a metric that combines motion and appearance information. The additional appearance information reduces the number of identity switches, improving the tracking algorithm.

An issue with DBT is that the performance of the employed object detector highly depends on the performance of the employed object detector [10].

3 Project Limitations

Transformers work best as large models, and they require a large amount of data to be trained on to ensure a high-performing model is made. Unfortunately, video datasets are large too, meaning the computational requirements for the project are excessive and training the models from scratch, without pre-trained weights, is not an option if the desired high-performing model is to be made.

To put the computational requirements and the training time of the models into perspective, DETR [13] was trained on 16 GPUs for 3 days. To put the memory requirements of video datasets into perspective, 1 GPU on our cluster will run out of memory after putting 50 frames onto the GPU. The average video in the MOT17 [14] dataset has approximately 750 frames. For a 2 month project, training from scratch is not feasible, especially for large video datasets. To overcome these limitations, models are trained from checkpoints on videos whose memory requirements are reduced by the method explained in section 4.2.

4 Accomplished Work

4.1 Object Detection

Object detection is a computer vision task where both object classification and object localization tasks are performed on an input image [15], with the task of identifying and labeling objects in an image with rectangular bounding boxes. In general, there are two primary techniques used to perform the object detection task: 1) One-stage detector; perform object detection as a regression or classification problem, adopting a more direct approach, and 2) Two-stage detector; generate region proposals then classify each proposal into object categories, adopting an indirect approach.

Prior milestones in one-stage detector networks include the YOLO (You Only Look Once) model [16] and the SSD (Single Shot Detector) model [17]. The YOLO model splits each image into regions and predicts bounding boxes and probabilities for each region simultaneously [18]. SSDs improved the YOLO model by using multi-reference and multi-resolution detection techniques to improve the detection accuracy while maintaining the same detection speed [18].

A one-stage detector approach is adopted in this project. This is due to their high inference speed and simple model implementation. The trade-off is that the one-stage detector models achieve lower accuracy than two-stage detector models.

4.2 Efficient Video Handling in PyTorch

Video handling in Pytorch is a tricky problem as it requires custom data handlers to be made depending on the dataset used. Luckily, the VideoFrameDataset class [19] allows for efficient data handling. The VideoFrameDataset class samples frames evenly from the video via **sparse temporal sampling** allowing for a representative, memory and compute efficient data loading of videos.

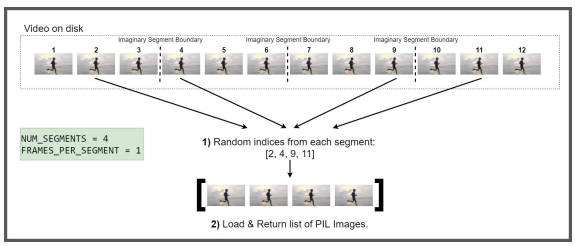

Figure 2: Source: [19]. A visualization of the sparse temporal sampling technique used by the VideoFrameDataset class.

The video frame sampling method divides the video into a number of segments. From each segment, a number of frames are selected and returned as images. This returns an $\mathbb{R}^{(N,F,C,H,W)}$

tensor of N batches and F frames that can be placed into a PyTorch DataLoader, ready for training.

The video labels are passed in the annotations file. This is a file that contains the file location for each frame of a video, followed by the label accompanying the video. Due to the high quantity of labels for an MOT task, all labels must be placed into a list of dimension (# frames × # objects in frame × 4), where the 4 represents the 4 coordinates of the bounding box.

4.3 ViTs for Object Detection

4.3.1 Feature Pyramid Networks

A major problem with identifying objects with vanilla ViT models is that objects smaller than the image patches cannot be consistently identified. This insight poses an important trade-off with regard to patch size for object detection. Larger patches reduce the resolution of the model, however, have high semantic value, high inference speed and reduce the computational requirements. In contrast, smaller patches increase the resolution of the model, however, have low semantic value, low inference speed and increased computational requirements.

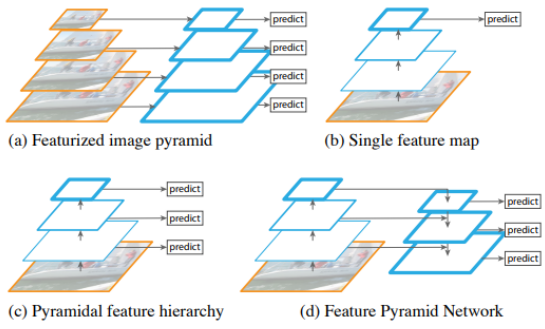

Figure 3: Source: [20]. A summary of the potential pyramidal/single feature hierarchy techniques. (a) This is an example of an image pyramid building a feature pyramid. Each feature layer is trained on one image layer, thus the computational requirements are large. (b) Most detection systems have single-scale systems, where an input image is reduced in scale to get image features. (c) The use of the feature hierarchy made by a CNN. (d) This is the FPN pipeline.

Object Detection with images of different sizes is challenging, particularly for smaller objects. A pyramid of the same image at different sizes can be used to detect objects, however, computationally expensive (especially when training).

A Feature Pyramid Network (FPN) [20] is a feature extractor designed for such a pyramid concept with accuracy and speed in mind. FPN exploits the pyramidal hierarchy of deep convolutional networks to construct feature pyramids with little computational cost [20]. This promotes the detection of objects regardless of size and optimizes the resolution/semantic value trade-off while minimizing any additional computational cost.

4.3.2 ViTDet

ViTDet [21] is an object detector that uses a ViT as a backbone for the object detection task. ViTDet draws inspiration from FPNs by creating a pyramid network, however, only uses the last feature map (FPN uses the feature map from each layer of the pyramid). The authors refer to this technique as a 'simple feature pyramid'.

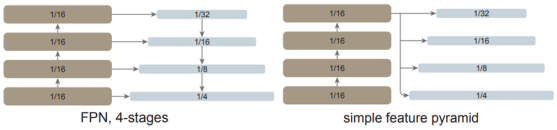

Figure 4: Source: [21]. A comparison of a normal FPN model compared to the simple feature pyramid used in ViTDet.

The information propagation strategy used in the last block of each subset is achieved via window attention. Given a high-resolution feature map, the map is divided it into non-overlapping windows, with self-attention computed between each window. The bounding box predictions are then acquired by passing the resulting heads in each subset to a mask-RCNN predictor.

4.3.3 Set-Based Matching: The Hungarian Algorithm

Set prediction tasks for object detection are difficult due to the requirement to avoid near bounding box duplicates. This problem is often overcome with postprocessing techniques, such as non-maximal suppression.

The optimal bipartite matching is achieved with the Hungarian algorithm. This is an optimization function that searches for a permutation, $\sigma \in G_o$, of o elements (in this case the number of objects) that gives the lowest matching loss. If $\hat{\sigma}$ is the ideal bipartite matching and $L_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is the pair-wise matching loss between the ground truth y_t and the prediction $\hat{y}$ of matching index $\sigma(i)$, the Hungarian algorithm efficiently identifies the optimal assignment that minimizes the matching loss, formalized by equation 3.

$$\hat{\sigma} = \arg\min_{\sigma \in G_o} \sum_{i}^{o} L_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) \quad (3)$$

This optimal assignment can be used later while finding the average IoU and mAP of the model on the dataset in question.

4.4 Model Training

Due to the high dependency on the performance of the object detector used for a DBT model, the model must be trained on sufficient data to ensure the high-performance requirement of the object detector is met. As such, the weights for the model are loaded from a pre-trained implementation of ViTDet on the MS COCO [22] dataset. Then the pre-trained model is further trained on the MOT17 [14] dataset.

The Adam optimizer [23] is used to optimize the set-based loss of the model. The Adam optimizer is a combination of the RMSprop and momentum [24] algorithms, where an adaptive learning rate is acquired with the aid of exponentially weighted averages of the past squared gradients and of past gradients. An issue with the Adam optimizer is that the changing step size of the function often rapidly decreases, resulting in issues with optimizer convergence. The AMSgrad [25] algorithm is used in unison with the Adam optimizer, however, it avoids large changes in the step size, aiming to improve the convergence of the algorithm with respect to using Adam on its own.

4.5 Dynamic Tokenization

The dynamic tokenization module used in [2] is embedded into the ViT module to dynamically reduce the computational requirements of the model during inference based on the complexity of the input image. A-ViT halts the computation for different tokens at different model depths. This is achieved with no extra learnable parameters, improving model throughput with minimal loss in performance. We see in figure 5 that the learnt halting tokens align well with image semantics, making the module an ideal candidate for object detection and tracking tasks.

This mechanism has thus far primarily been used for the image classification task, however, it can be seen from the albumation research performed by the authors that the technique is very good at removing computation representing background data while applying attention to the objects in the image. This makes A-ViT a good technique to use for object tracking tasks, reducing compute while maintaining the necessary information for object detection.

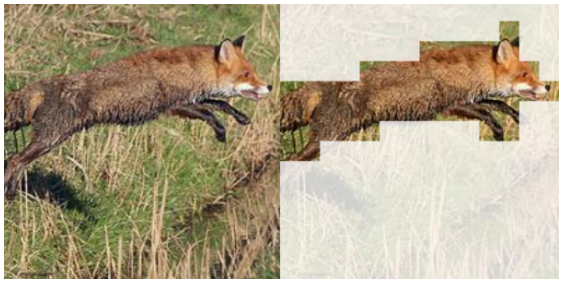

Figure 5: Source: [2]. We see that the non-pruned tokens align with the image semantics in the image.

5 Results

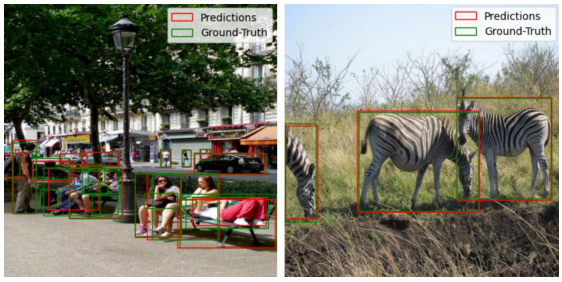

Figure 6: ViTDet COCO dataset predictions.

After training the pre-trained ViTDet object detector on the MOT17 training data, the model was tested on the MOT17 test data. The model achieved an average IoU of 0.77 on the test dataset and a mAP of 0.60. The model had an average frame inference speed of 0.277s/frame and it is suspected that the frames per second of the model will be improved once the object detector is integrated with Deep-SORT for a DBT.

Conclusion and Future Work

Given the time frame, this was a very ambitious project that accomplished many of its objectives and created the groundwork for future novel work to be completed. A plethora of material had to be learned to implement the tracker, from object detection and its metrics, transformers, ViTs, and its variants for object detection.

Subsequently, a backbone ViT called ViTDet was trained and implemented on the MOT17 and COCO datasets resulting in a high-quality object detector that was used in DBT. An A-ViT module from [2] was implemented into a normal ViT for classification, however, there was not sufficient time for implementation and training of the module into the ViTDet object detector.

For future work, the A-ViT module will be im-

Figure 7: ViTDet tracker predictions on MOT17 dataset. 6 frames shown in example.

plemented into the ViTDet object detector and then modified such that the module prunes redundant tokens based on frame-to-frame spatial similarity as well as spatial image complexity. Once this is completed, the object detector can then be implemented with Deep-SORT for improved tracking performance.

Acknowledgements

I would like to thank my supervisors, Doctor Asieh Salehi Fathabadi and Doctor Jagmohan Chauhan, for their continued support throughout the project.

I wish to also acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] H. Yin, A. Vahdat, J. M. Alvarez, A. Mallya, J. Kautz, and P. Molchanov, "A-vit: Adaptive tokens for efficient vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 809–10 818.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[4] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in

Proceedings of the IEEE international conference on computer vision, 2017, pp. 843–852.

[8] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[9] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," *Advances in neural information processing systems*, vol. 34, pp. 13 937–13 949, 2021.

[10] W. Luo, J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim, "Multiple object tracking: A literature review," *Artificial Intelligence*, vol. 293, p. 103448, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370220301958

[11] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[12] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[14] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831. [Online]. Available: http://arxiv.org/abs/1603.00831

[15] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[18] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint arXiv:1905.05055*, 2019.

[19] RaivoKoot, "Raivokoot/video-dataset-loading-pytorch: Generic pytorch dataset implementation to load and augment videos for deep learning training loops." [Online]. Available: https://github.com/RaivoKoot/Video-Dataset-Loading-Pytorch

[20] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[21] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection. arxiv 2022," *arXiv preprint arXiv:2203.16527*, 2022.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[25] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," *arXiv preprint arXiv:1904.09237*, 2019.