Training Strategies for Vision Transformers for Object Detection

Apoorv Singh
Motional
USA
apoorv.singh@motional.com

Abstract

Vision-based Transformer have shown huge application in the perception module of autonomous driving in terms of predicting accurate 3D bounding boxes, owing to their strong capability in modeling long-range dependencies between the visual features. However Transformers, initially designed for language models, have mostly focused on the performance accuracy, and not so much on the inference-time budget. For a safety critical system like autonomous driving, real-time inference at the on-board compute is an absolute necessity. This keeps our object detection algorithm under a very tight run-time budget. In this paper, we evaluated a variety of strategies to optimize on the inference-time of vision transformers based object detection methods keeping a close-watch on any performance variations. Our chosen metric for these strategies is accuracy-runtime joint optimization. Moreover, for actual inference-time analysis we profile our strategies with float32 and float16 precision with TensorRT module. This is the most common format used by the industry for deployment of their Machine Learning networks on the edge devices. We showed that our strategies are able to improve inference-time by 63% at the cost of performance drop of mere 3% for our problem-statement defined in Sec. 3. These strategies brings down Vision Transformers detectors [3, 15, 18, 19, 36] inference-time even less than traditional single-image based CNN detectors like FCOS [17, 25, 33]. We recommend practitioners use these techniques to deploy Transformers based hefty multi-view networks on a budge-constrained robotic platform.

1. Introduction

In the last decade, Convolution Neural Networks (CNNs) was driven by the model architectural updates [11–13, 32] in the field of computer vision. Moreover there have been a plethora of techniques proposed to improve training strategies of these CNN models [1, 6]. Recently, Vision Transformers, first introduced by ViT [5], and iteratively

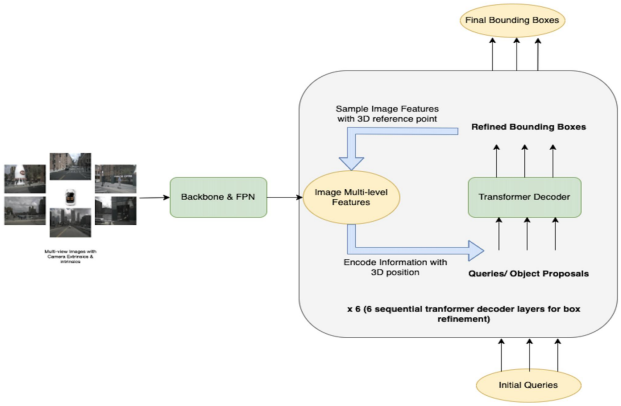

Figure 1. A standard multi-view vision based detector [36] with CNN based backbone and FPN [16] and transformers based detection head in gray-box. Inputs: Multi-view images, camera transformation matrices. Outputs: 3D bounding-boxes.

reformulated by DETR-based approaches [3, 15, 18, 36] has emerged as the better alternative architecture for object detection with images. However, the literature and leaderboards of the Transformers' object-detection community tends to focus most on the architectural impact of these hefty models. When these methods are to be used on an actual robotic-platform, runtime-accuracy joint-optimization is what that matters the most, owing to the fact that any edge-device would have a limited compute-budget. Moreover these ML algorithms have to be operated at high frequency as autonomous cars move fairly fast and need to update their road and dynamic agents' understanding at at-least 10hz (10 times a second). Any top-performing method from a detection leader-board [2, 31], would most likely be based out of ensemble approach, which is purely impractical on a compute-budget constrained on-board device. Performance of any machine learning model is dependent on three things: 1) Architecture; 2) Training Strategies; 3) Inference-time budget. This work, unlike others, focuses primarily on the later two: *Training strategies* and *Inference-time budget*. We conclude that with the

right network down-scaling strategies we can make Vision Transformers-based detectors practically capable to be fitted onto a deployment platform that operates within the inference-time budget and at high frequency. Transformers, which was initially inspired from language models, had less focus on inference-time bottlenecks early on as most of the processing of language models occurs on cloud the servers for example with ChatGPT. Autonomous car, an extremely safety critical system, has to perform its computation on-board device as we can not live-stream video data from cameras and laser sensors to a cloud servers; have it processed there; and sequentially download the processed predictions back to the device; in addition of doing all that within our real-time operation constraint.

1.1. Motivation

Performance improvements not only come from the novel architectures [3, 5, 15, 18, 19, 36] but also from the modern scaling strategies of the Vision Transformers network. We noted that this is specially the case for inference-time improvements, as more researchers tend to focus on accuracy compared to inference-time optimizations if at all. Our work is to carefully analyze inference-time optimization strategies that may eventually lead to a Vision-based transformers models practically deploy-able on an autonomous vehicle platforms.

1.2. Contribution

Our contributions can be summarized as below:

- We identify modifications in Vision Transformer's architecture of the current State-of-the-art (SoTA) object detection algorithms that lead to better runtime-accuracy optimization. We also list findings on model-precision and training schedule involved in this analysis. As a secondary check of the model we perform MACs (#operations) analysis to add to the third dimension over inference-time and performance trade-off.
- We design post-processing and pre-processing strategies that lead to runtime-performance improvements for Vision Transformers models.
- We design an effective evaluation strategy for an on-car detectors that is able to cater all our accuracy needs and inference-time constraints. We extend inference-time analysis one step further and evaluate the inference-time numbers on a *float16* and *float32* model precision in *TensorRT* model format.
- We also discuss further extension ideas for this work that researchers may focus on to further optimize on runtime-accuracy optimization for these models. We link those ideas with the representative work too, bridging a relation from other fields.

Based on our empirical results we claim that carefully adjusting input-image resolution and Transformer-decoder layer embedding space and other parameters can bring down network inference-time down by 63% with just minimal drop in performance. We evaluate all the results on our in-house dataset which in terms of representation can be compared to any public dataset like nuScenes [2] or Waymo Open Dataset [31] but much larger with more diverse classes and scenarios, or in other words much more difficult one too. By adopting proposed strategies, we were able to get inference-time for Vision Transformers detector models close to or in-fact even better than a lot of traditional CNN-based detectors, keeping the detection performance at par with the superior Vision Transformers models. To the author's knowledge this is the first paper which does runtime-accuracy scaling strategies for Vision Transformers models. Moreover we do inference-time analysis on *TensorRT* model format along with MACs analysis.

1.3. Previous Work

[6] paper highlights interesting strategies on model-scaling for CNN models with respect to speed-accuracy Pareto curve. However this analysis lacked runtime analysis on the *TensorRT* model. Moreover this paper focused more on CNN based backbone and not the Transformer based [34] module which tend to be SoTA detectors in the current literature.

[1] focuses on the ResNet [11] based backbone. They only evaluated their models on a single-image based classification task, ImageNet [4], which is a fairly simple task compared to multi-view 3D object detection tasks [2, 31]. Hence a lot of learnings from here might not be valid for on-car deployment of these Transformers based networks.

[14] focuses on classification performance with [5] approach, which handled the problem end to end with transformers by dividing patches into 16x16 grids [5]. This has proven to be a slower and sub-optimal approach for the much more complex problem of multi-view object detection problems. Modern approaches [15, 36] have proved to better for this task as per the detection-leaderboard . In addition, this approach focuses on accuracy-training time pareto curve, but our focus is on accuracy-runtime or in other words accuracy-inference time of the network, which is a bigger problem statement in the autonomous vehicle industry.

While [24] focuses on Transformers but for the application of neural machine translation, which is a significantly different problem compared to the multi-view 3D object

detection problem that we are trying to solve with Vision Transformers.

[39] analysis is probably the most related to our studies however, it is yet different in multiple aspects. Firstly, this scaling studies again focuses on the single-image based classification task of ImageNet [4] and does not have any mobile robot application. This paper also focuses on ViT [5] based approach, which is an end-to-end transformers based network and not DETR based approaches [3, 15, 18, 19, 36] that have recently proven better track record in doing 3D object detection for complex datasets of autonomous driving with multi-view cameras.

In contrast to the other work, we focus on down-scaling of model with the close-watch on model's performance degradation focused on the Vision Transformers models that are based on CNNs for extracting image features and transformers for detection head to predict boxes in the Bird's Eye View space. Other transformers work [3, 40] primarily focuses on the architectural impact of their approach, however, we focus on training strategies that lead to joint optimization of performance and run-time thereby making these networks deploy-able on an autonomous vehicle under run-time and compute constraint.

2. Object Detection Methods

In this section we will cover different strategies used in object detection starting off with *Single-image Based Detection* and then focusing majorly on *Multi-image Based Detection*.

2.1. Single-image Based Detection

Single-image based object detection can be divided into *two-stage*, *single-stage* and *set-based* detectors in terms of chronological invention of these detectors. Two-stage detectors [10, 26, 38] are a class of detectors that are divided into two stages. First stage is to predict arbitrary number of object proposals, and then in second stage they predict boxes by classifying and localizing those object proposals. However, these models suffers through high inference-time bottleneck because of the two-stage nature with redundant computations. Then came the single-stage detectors [7, 17, 25, 33] which brings down the inference-time of two-stage detectors. These models either uses heuristics based anchor boxes [17, 25] or center-based heatmap on features [7, 33, 35] to make predictions. However these approaches still relies on limited receptive field associated with the CNNs and fail to develop long-range relations between the image features. This becomes especially the concern for multi-image detection problem in autonomous driving where 6-8 cameras are used to capture the entire 360° surrounding scene. Next, researchers became interested in bringing transformer-based architecture leanings from language models to the computer vision task. [5] reformulated the entire CNN-based problem to transformers based problem. This architecture has transformers based encoder a.k.a backbone as well as transformers based detection head. It suffered through inference-time constraints as attention module was applied to even very low-level pixels information and hence shooting up the computation requirement. However, some strategies [39] have been explored on accuracy-runtime pareto curve, but it only focused on relatively easier image classification problem. Subsequently most recent detection work has been focused on CNN-based backbone and transformers based detection head in DETR-based approaches [3, 18, 19, 40]. These approaches were claimed to be much superior in making sparse predictions in the scene by leveraging self-reasoning of the object-proposals in self-attention layer [34]. These object-proposals are later refined as a set of predictions of boxes. This approach has also extended its way to the multi-view object detection problem as discussed in the next section.

2.2. Multi-image Based Detection

For a lot of robotics applications including autonomous driving, perception module needs to make prediction for the entire 360° scene [29], and that too in the Bird's eye view (BEV) so that it can be easily consumed by the other autonomy's software module like path-prediction and path-planning. Until the last few years, multi-view based detection problem was handled using per-view Machine learning based methods and then all the detections were consolidated using heuristic methods of data association using IoU (Intersection over Union). This post-processing step merges duplicate detections coming from the adjacent camera-views. Recently a lot of work have been explored in learning end-to-end differentiable network for multi-view detection. These approaches can be broadly classified as per [29] by (1) *Geometry based view transformers* and (2) *Cross-attention based vision-transformer*. Geometry based models are typically based out of [23] approach, which as an intermediate step creates BEV pseudo point-cloud by discretizing depth for each pixel and then runs a LiDAR based detector [7] on the pseudo point-cloud frustum from all the cameras. Due to the inherent two-stage process these models tend to have huge inference-time, despite being entirely based out of CNNs. Cross-attention based vision-transformer based approaches [15, 36] have shown to be SoTA in the leader-boards [2, 31]. These approaches typically have CNN based backbone and transformers based head which feeds in queries and CNN features to predict 3D bounding boxes [22, 27, 28]. [18, 19, 36] have focused their work on sparse-queries that are either learned from the training-dataset or constructed using input-data, whereas [15] focuses on dense queries in the BEV map. For the

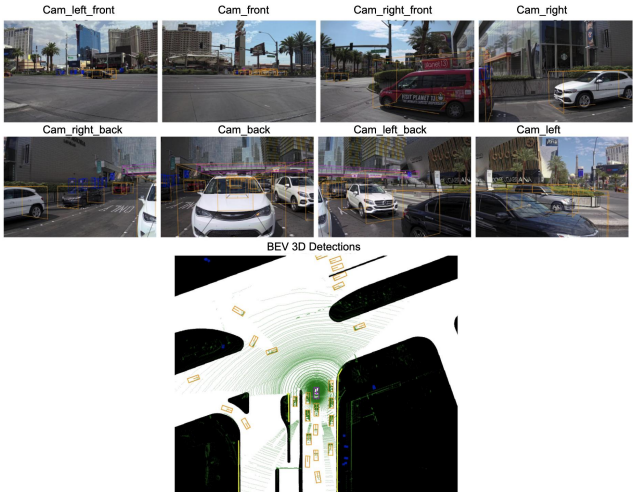

Figure 2. Input-Output of the Problem Statement. Surround-view 8 camera images (top); LiDAR Point Cloud overlayed over an HD Map (bottom). Key: Green points: LiDAR point cloud; Pink box: Autonomous vehicle; Black-white map: Pre-computed HD map.

scope of this paper we focus our methods on sparse-query based approaches which are much more optimal in terms of inference-time. Swin-transformer [20] has explored in converting CNN based backbone to transformers based one using hierarchical Transformer whose representation is computed with shifted windows.

3. Evaluation Criteria

In this section we discuss different fronts used for comparative analysis for different models viz., *Accuracy*, *Inference-time a.k.a run-time* and few less used ones: *Multiply–accumulate operation (MACs)* and Number of Parameters.

3.1. Model Accuracy

3D object detectors use multiple criteria to measure performance of the detectors viz., precision and recall. However, mean Average Precision (mAP) is the most common evaluation metric. Intersection over Union (IoU) is the ratio of the area of overlap and area of the union between the predicted box and ground-truth box. An IoU threshold value (generally 0.5) is used to judge if a prediction box matches with any particular ground-truth box. If IoU is greater than the threshold, then that prediction is treated as a True Positive (TP) else it is a False Positive (FP). A ground-truth object which fails to detect with any prediction box, is treated as a False Negative (FN). Precision is the fraction of relevant instances among the retrieved instances; while recall is the fraction of relevant instances that were retrieved.

$$Precision = TP/(TP + FP) \quad (1)$$

$$Recall = TP/(TP + FN) \quad (2)$$

Based on the above equations, average precision is computed separately for each class. To compare performance between different detectors (mAP) is used. It is a weighted mean based on the number of ground-truths per class. Alternatively F1 score is the second most common detection metric, which is defined as weighted average of the precision and recall. Higher *AP* detectors gives better performance when the model is deployed at varied confidence threshold, however higher *max-F1* score detector is used when the model is to be deployed at a known fixed optimal-confidence threshold score.

$$F1 = 2 * Precision * Recall/(Precision + Recall) \quad (3)$$

For all our evaluations we restrict our Ground-truths and predictions up-to *60meters* in the Bird's eye view space from the ego-vehicle for comparison at common-grounds.

3.2. Inference-time

In-terms of practical application, another equally important metric for detectors is the inference-time, defined as run-time during deployment of the model. We compare our models on *PyTorch* model at standard precision *float32* and also at optimized precision of *float16* in Sec. 4.4. In most of the robotic applications deployment platforms is *Nvidia GPU*, for which *TensorRT* is used as the deployment library. *TensorRT* runtimes and *PyTorch* module inference-time aren not always correlated. So it is not a fair comparison of inference-time of the model until we do this analysis with the actual *TensorRT* format. In addition to *TensorRT float32* model precision we show run-time comparison in *TensorRT float16* format a.k.a *half-precision* as well.

3.3. MACs and Parameters

MACs (Multiply–accumulate operation) and number of parameters in any ML model are other less used metrics to compare two models size along with the inference-time. While inference-time of the network is dependent on the hardware but MACs and number of parameters are hardware-agnostic parameters. MACs defined number of computations happen during inference of the model. These computations typically include number of *multiply* and *add* operations on the float numbers. These metrics are generally only used when deployment hardware is not known while development of the ML network. MACs and parameters might not always correlate with inference-time, which is more predictive measure of latency of the network.

4. Methodology and Experiments

In this section we will present series of network down-scaling methods. We train our model on our in-house au-

Module	Inference-time	MACs	Parameters
Entire Network	447	832B	37M
Backbone + FPN	380	829B	34M
Detection Head	67	2B	3M

Table 1. Profiling different stages of the network in terms of inference-time, MACs and number of parameters.

tonomous driving dataset with 3D BEV boxes as annotations and evaluated as per *mAP* and *max-F1* score mentioned in Sec. 3 and inference time calculated on a *T4 Nvidia GPU* in different model formats. All the models are trained and evaluated on the same GPU as well. One sample of our dataset is shown in Fig. 2. Our training strategies can be categorized as: *Pre-training Strategies*, *Training Strategies*, *Post-training Strategies*, *Model Formats and Precision*. For profiling measures: inference-time is measure in milliseconds, MACs are measured as number of operations and Paramters are measured as number of parameters.

4.1. Pre-training Strategies

4.1.1 Input Resolution

To understand the over-head of backbone and detection head as shown in Fig. 1, we ran profiling with inference-time, parameters count and mac operations on it. Results from Tab. 1 shows that most of the inference-time over-head lies in the backbone. We deduced that image being a 2D matrix; total inference-time can be quadratically reduced if we can reduce the input resolution keeping a close watch on accuracy performance change. We trained and evaluated our model by aggressively reducing the input resolution as shown in Tab. 2. We noticed that by reducing image-resolution smaller objects like pedestrians is affected more adversely compared to larger object vehicles. We concluded that we can safely reduce down the resolution to $(598, 394)$ i.e. $3/5^{th}$ of the original $(996, 656)$ with 52.5% inference-time improvement at the cost of mere 1.4% drop in the pedestrian AP (Average precision) performance and the 0.5% drop in the vehicle AP performance. Also note that the number of parameters don't change with input-resolution as it is based out of fully-convolutional based backbone [13] and an FPN [16]. For further experiments we will focus more on the inference-time metric rather than MACs and Parameters, as inference-time is a more representative metric for the given hardware. For further experiments we will change our baseline to $(598, 394)$ input-resolution to show accumulative effect of our strategies.

4.1.2 Image Pre-croppers

In object detection, image Pre-cropping refers to cropping of the image from certain edges of the camera which represents least informative pixels to squeeze out model inference-time from it by saving onto those pixels' computations. To select optimal pre-cropping strategy for our current network we used two methods, firstly we visually analyzed images as shown in Fig. 2 and tried to find the least informative part of the image in-terms of our problem statement i.e. detect all the safety critical agents on the road. We noticed that generally top few-tens of pixels in the image are the least important as they mostly include top of a tree/ top of a building/ top of a pillar etc. as shown in Fig. 2. Our second methods was quantitative analysis in which we gathered all the boxes that lies in those cropped top few-tens pixels to confirm that we are not pruning information of any of our objects of interest with those cropped pixels. We found that less than 0.25% of our ground-truth in training dataset is actually in the top-50 pixels for input image-resolution of $(598, 394)$. Our empirical findings are shown in Tab. 3. We noticed that removing pixels from the top has no/ positive effect on pedestrians and only minor effect on vehicles, as some of the features of vehicles might get missed specially for a very close-range vehicles which occupy the entire image. We concluded that removing 50 pixels from top is an optimal strategy for inference-time and accuracy joint-optimization with 10% improvement in inference-time at the cost of only 2% accuracy drop in vehicles and no drop in pedestrian.

4.2. Training Strategies

4.2.1 Training Schedule

For our baseline experiments we used one-cycle learning rate (L.R.) scheduler [30] with maximum learning rate as $5e^{-5}$. We experimented with three values associated with one-cycle learning rate viz., Start L.R; Max L.R. and End L.R. Empirical results are shown in Tab. 4. We concluded that there's a value in using one-cycle learning rate with its limits by using higher learning rate in the middle of training and relatively much smaller learning rate at the beginning and end of the training. We also observed if we try to increase the ending learning rate, we suffer through training-divergence.

4.2.2 Number of Decoders

Transformers decoders are sequential transformer layers that iteratively refines object-proposals so that they eventually converge to predictions that match the ground-truth as defined in [3]. In this section, we try to build a relation on how many number of sequential decoder layers are optimal. Similar analysis has also been done in [36], but we

Input-resolution	Vehicle (AP)	Pedestrian (AP)	Inference-time (ms)	MACs	Parameters
996, 656	64.9	71.8	447	832B	37M
664, 437	64.5	71.5	242	372B	37M
598, 394	64.0	71.4	212	305B	37M
498, 328	60.4	70.6	166	212B	37M

Table 2. Effects of reducing input-resolution on accuracy and inference-time. Input-resolution is represented as pixels in width and height dimension respectively.

Input-resolution	Cropped pixels (Top)	Pedestrian (AP)	Vehicle (AP)	Inference-time
598, 394	0	64.0	71.4	212
598, 394	25	64.8	70.3	200
598, 394	50	64.8	69.6	191
598, 394	100	62.8	65.4	175

Table 3. Empirical experiment with Pre-croppers.

L.R. Star	L.R. Max.	L.R. End	Pedestrian	Vehicle
$5e^{-6}$	$5e^{-5}$	$5e^{-6}$	60.7	67.0
$1e^{-5}$	$1e^{-4}$	$1e^{-5}$	63.1	72.4
$1e^{-6}$	$1e^{-4}$	$1e^{-5}$	63.4	70.8
$5e^{-6}$	$5e^{-4}$	$5e^{-6}$	64.0	72.9

Table 4. Empirical experiment with the one-cycle learning rate scheduler. Pedestrian and vehicle performance is measured in AP (Average Precision).

#Decoders	Pedestrian	Vehicle	Inference-time
1	40.6	49.7	134
2	57.9	64.3	144
3	60.1	66.1	155
5	61.9	67.4	179
6	64.8	69.6	191
7	65.0	70.1	202

Table 5. Empirical experiment with number of transformers decoders. Pedestrian and vehicle performance is measured in AP (Average Precision).

wanted to confirm the findings based on our own dataset and other training conditions. With our empirical experiments as shown in Tab. 5, we concluded that our baseline number of decoder layers i.e. **6**, is indeed the most optimal number of decoders for the problem statement.

4.2.3 Embedding Dimensions

In this section we try to formulate the relevance of the long embedding dimension of the queries. Queries are latent representation of the object-proposals which are refined using

Embedding Dimension	Pedestrian	Vehicle	Inference-time
256	64.8	69.6	211
128	64.7	69.9	191
96	64.2	68.4	190
64	61.4	66.7	183

Table 6. Empirical experiment with the varied embedding dimensions of the decoder. Note: FPN [16] and transformer queries dimensions are changed together in these experiments. Pedestrian and vehicle performance is measured in AP (Average Precision).

transformers-decoders for making final predictions. Visualizations of the query in image-space can be referred in [3]. Baseline taken from [3, 36] has embedding dimension of 256, but based on our results shown in Tab. 6, 128 dimension gave 9.5% better inference-time with no performance drop at all. It is also worth noting that due to internal optimization of the GPU, embedding dimension has to be of the power of 2 for inference-time optimization. This can be empirically seen in the row with 96 embedding dimension where inference-time is close to the one with 128 dimension-size despite having less computations.

4.2.4 Number of Queries

In this section we experimented with the number of object-proposals a.k.a queries in transformers head. This parameter is specially very sensitive to Vision transformer head because we natively don't perform Non-maximum suppression on the detection output, owing to the fact of using set-based loss. Set-based loss theoretically forces to make only single prediction per ground-truth object, in-comparison to dense-prediction methods [15, 17, 25, 33] which makes mul-

#Queries	Pedestrian	Vehicle	Inference-time
900	64.8	69.6	191
700	64.6	69.5	187
500	65.0	69.9	183
400	64.8	70.0	181
300	61.5	66.7	181

Table 7. Empirical experiment with the number of queries. Pedestrian and vehicle performance is measured in AP (Average Precision).

tiple prediction per ground-truth. Theoretical way to get the optimal number for queries is by analyzing the training dataset and finding average number of ground-truths present per training sample. Then we may add $10 - 20\%$ of that number as a buffer for negative class mining. In addition to these analysis we empirically show results in Tab. 7. On an average our training dataset has 300 agents per 360° view. We concluded that using 400 queries are optimal for this problem-statement, with a 5% inference-time improvement at the cost of no performance. Moreover we actually noticed that precision has increased with 400 query model when compared to 900 queries because of more relevant predictions.

4.3. Post-training Strategies

In this study we go through different post-processing techniques we can use to maximize our performance once the training has completed. These post-processing strategies' inference-time overhead can be treated as negligent compared to the model inference-time.

4.3.1 Pick Top-k boxes

In baseline paper [36] we noted that the authors have made use of top-k boxes to filter predictions based on confidence values. We experimented by removing this top-k logic with a parameter sweep, but we concluded that having top-k boxes as 90% of the boxes a.k.a number of queries seemed to be the optimal number with vision transformer detectors.

4.3.2 Non-Maximum Suppression (NMS)

Seminal paper [3, 18, 36], claimed against using NMS approach with transformers based network. Their rationale behind this claim was that self-attention layer and set-based loss during training of the network are enough to not let the network make duplicate detections for an object. However, even after tuning down number of queries to just 400, we noticed that there are some overlap bounding boxes for a single object. This is a clear application of the NMS method. We went ahead and implemented a simple version of class-agnostic NMS to see performance difference with the baseline in Tab. 8. We concluded that NMS has clear gain in vehicle class, as these boxes are generally bigger and transformers may try to associate a ground-truth with multiple queries. However, performance improvement wasn't very noticeable in case of smaller sized class i.e. pedestrians.

NMS	IoU-threshold	Pedestrian	Vehicle
$\times$	-	64.0	72.9
$\surd$	0.5	63.1	72.5
$\surd$	0.3	63.5	72.6
$\surd$	0.2	64.2	73.6

Table 8. Empirical experiment with NMS post-processing. Pedestrian and vehicle performance is measured in AP (Average Precision).

	PyTorch	TensorRT
float32	187	148
float16	98	62

Table 9. Model precision and model-format inference-time comparison. All numbers are measured on the same T4 Nvidia GPU.

4.4. Model Formats and Precision

In this section we will highlight inference-time improvements we can get with the hardware accelerators after reducing the precision of the models. We will cover two formats: *PyTorch* and *TensorRT* with two different model precision: *float32* and *float16* in Tab. 9. For *PyTorch* model format we noted there's approximate 50% inference-time boost just using half-precision a.k.a *float16* model precision. In addition to that *TensorRT* has further speedups by merging layers together and hence reducing number of operations. *int8* model precision wasn't studied in this analysis. It may provide further boost in the inference-time of these models.

5. Discussions

5.1. Societal Impact

We ran a series of scaling experiments to derive these training strategies on multiple-multi GPU machines for multiple days. Eventually we were able to prove our hypothesis on down-scaling strategies for hefty vision transformers models. We are hopeful that the cost and environmental effect associated with these experiments would be much lesser compared to the long term benefits these learning would have for future researchers and deployment engineers.

5.2. Further Extensions

Here we will cover the aspects which were not in the scope of the paper but authors believe would be worthwhile to explore for future Vision Transformers strategies. Whole lot of work has happened over CNN-based Neural Architecture Search (NAS) [8, 9] but we haven't seen any of such work associated with Vision Transformer models yet. It could be a great direction to explore and effectively convenient too as transformers tend to have less hyper-parameters compared to the CNN networks to search for. In addition we can develop model pruning strategies for Vision Transformers models, looking at inference-time improvements with CNN based Vision models [21, 37], we might see some low hanging fruits here. Lastly we think for papers with dense queries [15], researchers could develop on smart ways to automatically block some of the non-practical queries based on the prior HD-map or physics modeling etc. to get gain back the low-latency model.

6. Conclusion

In this work, we analyze runtime-accuracy performance numbers on different network related and pre-processing and post-processing related strategies on Vision Transformers model. We took an extra step of also comparing model runtimes on *TensorRT* modules in *float32* and *float16* precision which is more of a standard practice in the autonomous industry. In addition we also did MACs and #parameters analysis with these techniques to understand differences in the model as per the operations number. We concluded that depending on the problem statement, bigger and deeper models do not always lead to better results; despite wasting resources and environmental impact. With all our strategies, we are able to improve inference-time by 63% at the cost of performance drop of mere 3% for our problem-statement. We hope researchers and engineers community would benefit from these deployment-time analysis of Vision transformers models on a robotic platforms and can unblock some of the concerns we are facing for the path to deployment for these models. Also we hope to get some environmental positive impact towards scaling down energy-hungry compute resources for beefier object detection algorithms.

References

[1] Irwan Bello, William Fedus, Xianzhi Du, Ekin Dogus Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 34:22614–22627, 2021. 1, 2

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 2, 3

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 213–229. Springer, 2020. 1, 2, 3, 5, 6, 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2, 3

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3

[6] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. *arXiv preprint arXiv:2107.00057*, 2021. 1, 2

[7] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6569–6578, 2019. 3

[8] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pages 2943–2952. PMLR, 2020. 8

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 8

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1, 5

[14] Saghar Irandoust, Thibaut Durand, Yunduz Rakhmangulova, Wenjie Zi, and Hossein Hajimirsadeghi. Training a vision transformer from scratch in less than 24 hours with 1 gpu. *arXiv preprint arXiv:2211.05187*, 2022. 2

[15] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel,*

October 23–27, 2022, Proceedings, Part IX, pages 1–18. Springer, 2022. 1, 2, 3, 6, 8

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 5, 6

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 1, 3, 6

[18] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII*, pages 531–548. Springer, 2022. 1, 2, 3, 7

[19] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images. *arXiv preprint arXiv:2206.01256*, 2022. 1, 2, 3

[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 4

[21] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 8

[22] Jongwoo Park, Apoorv Singh, and Varun Bankiti. 3m3d: Multi-view, multi-path, multi-representation for 3d object detection. *arXiv preprint arXiv:2302.08231*, 2023. 3

[23] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pages 194–210. Springer, 2020. 3

[24] Martin Popel and Ondřej Bojar. Training tips for the transformer model. *arXiv preprint arXiv:1804.00247*, 2018. 2

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 3, 6

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 3

[27] Apoorv Singh. Transformer-based sensor fusion for autonomous driving: A survey. *arXiv preprint arXiv:2302.11481*, 2023. 3

[28] Apoorv Singh. Vision-radar fusion for robotics bev detections: A survey. *arXiv preprint arXiv:2302.06643*, 2023. 3

[29] Apoorv Singh and Varun Bankiti. Surround-view vision-based 3d detection for autonomous driving: A survey. *arXiv preprint arXiv:2302.06650*, 2023. 3

[30] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 5

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1, 2, 3

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1

[33] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 1, 3, 6

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3

[35] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021. 3

[36] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022. 1, 2, 3, 5, 6, 7

[37] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14913–14922, 2021. 8

[38] Sergi Adipraja Widjaja, Venice Erin Baylon Liong, Zhuang Jie Chong, and Apoorv Singh. Machine learning-based framework for drivable surface annotation, Jan. 19 2023. US Patent App. 17/836,974. 3

[39] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022. 3

[40] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 3