PDVS-RT-DETR: A Compact and Efficient Model for Aerial Vehicle Detection

Daniel Yang, Hiranya Udagedara, Fisayo Olofin, Mahdis Bisheban, Samira Ebrahimi Kahou

***Abstract*—Autonomous quadrotor systems are increasingly utilized for dynamic tasks such as object tracking and surveillance. Effective operation in these domains requires robust visual servoing capabilities. Traditional geometric visual servoing methods are computationally lightweight but suffer from limited robustness to appearance variations, occlusions, and complex backgrounds compared to deep learning approaches. Transformer-based detectors offer superior robustness but incur high computational costs, limiting their deployment on resource-constrained onboard hardware. In this work, we present our pruned depth-reduced visual servoing real-time detection transformer (PDVS-RT-DETR), a compact and efficient RT-DETR model that uses knowledge distillation and pruning techniques tailored to the task of visual servoing. We evaluate our model on an overhead vehicle detection dataset representative of aerial surveillance applications.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly utilized for applications ranging from agricultural surveillance, safety monitoring, and dynamic cinematography. Autonomous tracking and monitoring of moving objects in these environments by UAVs, depends on accurate, real-time object detection under challenging conditions including partial occlusions, dynamic backgrounds, and motion blur. Visual servoing systems that control UAV's motion require this detection capability to maintain continuous visual feedback. While traditional geometric visual servoing methods are computationally lightweight, they lack robust detection mechanisms resilient to these disturbances [1], [2]. Conversely, transformer-based object detectors, such as real time detection transformer (RT-DETR), offer superior global reasoning and robustness but are computationally slow [1]. To resolve the trade-off between tracking robustness and computation speed, this paper proposes a compact visual servoing framework based on knowledge distillation (KD) [3].

A critical challenge in deploying these detectors for robotics is the mismatch between model training and deployment contexts. Widely-used computer vision frameworks such as Ultralytics primarily develop and optimize models on the general-purpose microsoft common objects in context (MS COCO) dataset containing 80 object classes [4]. However, specialized robotic applications such as autonomous car surveillance require detection of only a single class, making these general-purpose models inefficient.

We define a new RT-DETR architecture, employing multiple "Teacher-Student" frameworks to bridge the gap between high-accuracy transformer detectors and the limited processing power available on-board. Our approach contributes a compressed "Student" model (denoted as PDVS-RT-DETR) derived from a larger "Teacher" baseline (real time detection transformer-large (RT-DETR-L)) by: (1) reducing the depth of the encoder backbone by 50%, and (2) structured pruning unimportant parameters. We contribute custom KD training scripts for RT-DETR, currently unavailable in the Ultralytics framework [4], which we release through the IDCL project Github repository[1]. Our experiments demonstrate that the compressed model retains high detection accuracy while achieving real-time performance, suitable for real-time deployment on embedded UAV platforms.

II. Related Work

Classical approaches for object detection such as Scale-Invariant Feature Transform (SIFT) [5] and Histogram of Oriented Gradients (HOG) [6], rely on manually designed features. As the complexity of the detection task increases, these classical methods struggle to maintain accuracy while remaining computationally effective. Therefore, most of the modern research tasks rely on deep-learning-based algorithms for improved accuracy and robustness [7].

Deep-learning-based algorithms could be further categorized into two-stage and single-stage detectors, based on their architecture [8]. A two-stage detector generates proposal regions first, and classifies the regions based on location afterwards. [9] uses Faster R-CNN, a two-stage object detector, for the detection of different traffic signs from images captured by a drone. The two-stage algorithm displayed high accuracy at the expense of long inference times, making it not viable option for real-time applications [9], [10]. Alternatively, one-stage algorithms generate class probabilities and locations directly, providing faster detections. The single-stage category includes CNN-based detectors, such as YOLO (You Only Look Once), and transformer-based detectors, such as RT-DETR [7], [10]. Transformer-based detectors provide stronger

Daniel Yang, Hiranya Udagedara, Fisayo Olofin, Mahdis Bisheban are with Dept. of Mechanical and Manufacturing Eng., and Samira Ebrahimi Kahou is with Dept. of Electrical and Software Eng, all at the Schulich School of Engineering, University of Calgary, Alberta, Canada. Emails: {daniel.yang2, hiranya.udagedara, fisayo.olofin, mahdis.bisheban, samira.ebrahimikahou}@ucalgary.ca.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Government of Alberta, Alberta Innovates, and the Schulich School of Engineering at the University of Calgary. Funding was awarded to Dr. Mahdis Bisheban, Director of the Intelligent Dynamics and Control Lab and Assistant Professor at the University of Calgary.

[1]Source code available at https://github.com/IDCL-UCalgary/Visual-Servoing-/tree/main

global reasoning through self-attention, compared to CNN-based detectors. Through this feature, [11] claims that they are more effective even in complex scenarios such as occluded and crowded environments. To justify this claim, we apply object detection inference on a test image from our dataset [12] containing multiple spaced detections occluded by trees across RTDETR-L, PDVS-RT-DETR, and YOLO11 models.

For higher accuracy in detection, larger models can be utilized, but often at the expense of increased inference times. To improve the accuracy while maintaining an operable inference speed, various approaches have been investigated. [13] showed that pre-processing of the images can improve the detection accuracy. [5], [14] showed that through applying a Median Enhanced Weiner Filter greater object detection accuracy was achievable at the expense of an increase in the pre-processing workload. [15] has replaced the original standard convolution modules of YOLOv8n with conditionally parameterized convolutions for high accuracy and faster inference. Although this approach provides faster inference, further performance gain in terms of accuracy can be achieved by using knowledge distillation (KD) and pruning [16]. KD enables a large-scale teacher model with high accuracy to train a similar or smaller-scale student model through the comparison of feature maps [17]. Another use of KD with RT-DETR framework was performed by [18], who customized RT-DETR backbone modules and ensured training convergence through KD. To prevent bias and improve generalization in KD, multiple teachers can be used in-place of the widely used teacher-student framework to allow for generalization in learning [19].

Pruning is primarily used to reduce the model size and can be performed in an unstructured or structured manner. Unstructured pruning eliminates individual weights with negligible effects on the model output. Structured pruning follows a more systematic approach, where pruning is achieved by removing channels or layers [16], [20]. For transformer-based detectors, attention head pruning has also been explored, where heads with low relative importance are eliminated in multi-head self attention layers [21].

In order to properly train YOLO or RT-DETR models, an adequate dataset consisting of training, validation, and testing splits is required. Examples of popular datasets used in developing pretrained weights for foundational backbone models consist of Microsoft Common Objects in Context (MS COCO) [22] and Open Images [23] due to the variety of unique features across detection classes. Because of so, models pretrained on these datasets are able to quickly adapt to custom detection tasks due to the likeliness that they already store some representation of the identifiable object features within their backbone. However, if the user desired detection task differs too greatly from these popular datasets, a pre-trained backbone may be less effective in generalizing data [24].

III. Methodology

A. Overall RT-DETR Architecture and Dataset

In RT-DETR, the input image is first processed through a lightweight Cross Stage Partial (CSP) Darknet backbone, where the visual features are extracted [2]. The extracted features are then processed globally through a transformer encoder. This is done through self-attention, allowing each pixel to observe the entire image, which enables greater performance in identifying overlapping or partially visible objects. The extracted features are then processed through the transformer decoder which matches queries with keys to generate and refine object predictions. These predictions are then passed through the detection heads of the architecture to generate bounding boxes with class probabilities across the image.

This study focuses on the RT-DETR-L model as it is the smallest available architecture with publicly available pretrained weights in [2]. All presented models within this study are trained using an overhead vehicle detection dataset provided by [12] consisting of 4,679 labeled frames with 17,040 annotated object instances across 10 classes. Such classes are; boat, camping car, car, motorcycle, pickup, plane, tractor, van, truck, or other. Input images are of size 640 x 640 pixels, which is consistent with the standard MS COCO image sizes that RT-DETR is benchmarked on [4]. All models obtained in this paper and tested with an unseen partition of 702 images from the dataset [12] as well as unseen overhead drone traffic monitoring footage [25] to achieve a visual interpretation of model performance.

B. Teacher Model Training

We divide the Overhead Vehicle Detection dataset [12] into training and testing sets comprising 85% and 15% of the data, respectively. We select a hyperparameter of five folds, a common practice, to split our 3,977 image training dataset into for cross-validation as it allows for a reasonable validation split size while minimizing model variance. [26] suggests that model variance levels off around five or ten fold splits depending on dataset size such that increasing the value of this hyperparameter yields little to no benefit. Following [2], stochastic gradient descent (SGD) is used for training RT-DETR-L on large datasets due to its simplicity and computational efficiency, while AdamW is adopted for teacher models to improve convergence behavior and reduce sensitivity to overfitting through decoupled weight decay. This training is conducted over a 100-epoch training schedule to match with default training lengths from pre-trained weights presented by [4].

When training transformer-based detectors such as RT-DETR-L, we apply a learning-rate warm-up schedule to stabilize optimization, even when pretrained weights are used. [2] report stable convergence by linearly increasing the learning rate from 1×10^{-7} to 1×10^{-4} over the first 2000 forward-pass steps. In this work, the most stable and accurate results were obtained using a warm-up schedule spanning 25 epochs, with

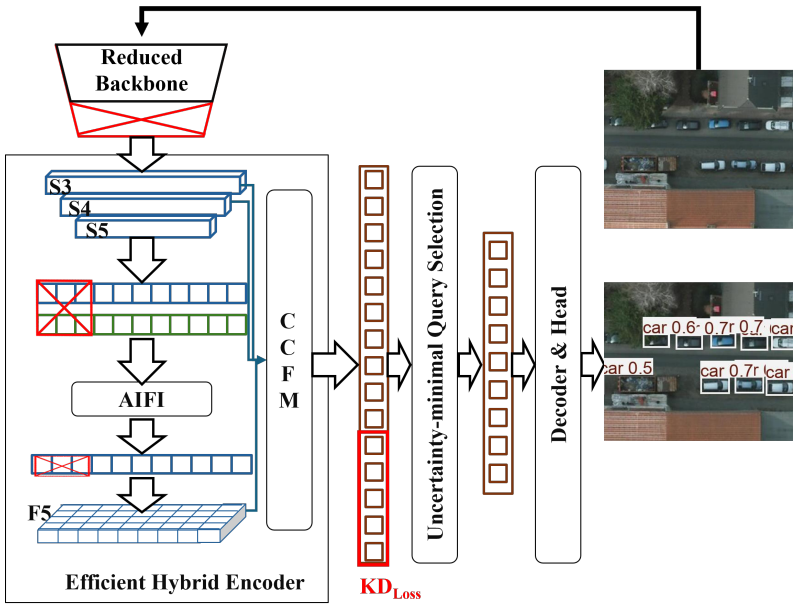


Fig. 1: Student Architecture - PDVS-RT-DETR with Backbone Reduction, Pruning, and Knowledge Distillation

an initial learning rate of 1×10^{-6} and a final learning rate of 1×10^{-4}. Unlike the single-sequence image streams considered in [2], the overhead vehicle dataset contains multiple vehicle classes across diverse environments. Thus, we define the warm-up schedule at the epoch level rather than per forward pass.

From the standard Ultralytics trainer, the detection loss is calculated as follows,

$$L_{total} = \lambda_{cls} L_{cls} + \lambda_{box} L_{box} + \lambda_{GIoU}(1 - GIoU), \quad (1)$$

where λ_{GIoU}, λ_{cls}, and λ_{box} are coefficients of Generalized Intersection of Unions loss $GIoU$, cross-entropy classification loss L_{cls}, and box loss L_{box}. They are defined as $\lambda_{GIoU} = 1.5$, $\lambda_{cls} = 0.5$, and $\lambda_{box} = 7.5$ from the standard weighting distribution [4].

C. Knowledge Distillation Loss and Weighted Sum

A feature-based distillation strategy is applied using PyTorch forward hooks to access and align hidden layer data [27]. When applying forward pass computations across a batch sequence of 8 training images, which is the largest batch size allowable given the GPU memory of our setup, the total loss is calculated for each teacher model obtained from cross-validation training. The teacher model with the lowest loss for this sequence is then selected for distillation in a method known as Reinforced Teacher Selection for Knowledge Distillation (RT-KD) [28]. By dynamically aligning each Student layer to the selected Teacher model's layers in the Cooperative Convolutional Feature Mapping (CCFM) portion of the network, the least squares knowledge distillation loss, shown by Loss_{KD}, is then calculated and summed with the batch loss, as follows,

$$\begin{aligned}\text{Training Loss} = {} & w_{KD}\,\text{Loss}_{KD} + w_{Class}\,\text{Loss}_{Class} \\ & + w_{GIoU}\,\text{Loss}_{GIoU} + w_{L1}\,\text{Loss}_{L1}, \end{aligned} \quad (2)$$

where w represents assigned loss category weights, Loss_{Class} represents cross-entropy classification loss, Loss_{L1} represents L1 bounding box coordinate loss, and Loss_{GIoU} represents bounding box intersection loss.

$$\text{KD Loss} = \frac{\sum_{i=1}^{n}(feature_{student} - feature_{teacher})^2}{n}, \quad (3)$$

where n is the total number of elements in the feature map output by a hidden convolutional layer, and feature represents a single element in the feature map. Following [4], these layers are selected as they represent large-scale feature fusions as opposed to individual fine details.

As KD training is currently unsupported for RT-DETR architecture in the standard Ultralytics library [4], we developed a custom training script for our student model training. In this custom training pipeline, KD is conducted in two stages. First, feature-map knowledge is transferred from a teacher ensemble to a depth-reduced student model. In the second stage, these distilled feature representations are further transferred from the student model to its pruned counterpart. To accommodate the additional learning required when bridging pretrained models with randomly initialized weights, our custom training script is executed over a 500-epoch schedule [29]. An overview of the proposed KD framework, is illustrated in Figure 2.

During student model training, $GIoU$ and $L1$ bounding box losses are doubled to ensure that the student model is capable of learning relevant features across the image space from scratch during its warm-up schedule. Classification loss

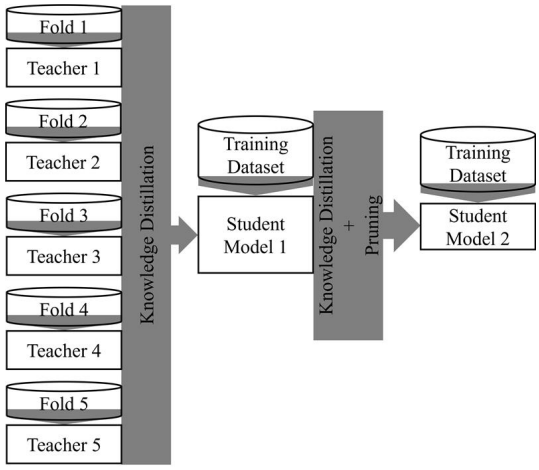


Fig. 2: Proposed Knowledge Distillation Framework within the Selected Training Pipeline

weighting is specified at the same value applied in teacher model training due to an observed tendency of the model to over-predict the background class whenever this weighting is increased. Lastly, knowledge distillation weight is set to a value of 1.0 to match with [28]'s methodology with only one teacher for distillation. When fine tuning the pruned student model, the same classification loss and knowledge distillation weights are applied while $GIoU$ and $L1$ bounding box loss weights are restored to their default training values of 1.5 and 7.5, respectively. To prevent overfitting of certain image batches, the training loss across each image batch is summed to determine a total loss value and the model with the lowest total loss across the training schedule is saved.

D. Student Model Training and Pruning

In order to train a student model with reduced backbone depth (Denoted as DVS-RT-DETR), the YAML file of the RT-DETR-L teacher model is adjusted such that the depth scale is set to a value of 0.5 to balance feature representation accuracy and inference speed. From this architecture, parameters are initially randomized using He initialization [29] to ensure stability in backpropagation. Due to this random initialization, the model is more sensitive in its early training stages and requires a longer training schedule to achieve desired performance. This is because the backbone responsible for detecting edges and relational patterns must be completely relearned as opposed to adapting from set of pretrained parameters, as reported by [30]. Additionally, image augmentations presented within the Albumentations library [31] are chosen over traditional ultralytics mosaic augmentations due to an observed benefit in RT-DETR training stability.

Based on the RT-DETR-L training outline of [2], we define a warm-up schedule of 100 epochs where the model learning rate increases from an initial rate of from 1×10^{-7} to 1×10^{-4}. This is to allow for a lower initial learning rate, which was proven by [2] to be beneficial in RT-DETR training stability. During this warm-up period, knowledge distillation is disabled due to it potentially providing an adversarial effect on the backbone's ability to learn key features and instead overfit CCFM layer parameters [32]. A tunable KD loss weight and variable learning rate presented by [32] is implemented to ensure stability while training DVS-RT-DETR. At epoch 101, KD is applied with an initial weighting factor of 0.05. This factor grows at a fixed rate until it reaches the final set value of 1.0 on epoch 120. Training is then carried out for an additional 380 epochs resulting in a 500 epoch long student training schedule.

Through applying pruning, additional increases in model efficiency are obtained, resulting in decreases of model inference times. Although previous pruning methodologies such as random pruning and weight based pruning accomplish this, they also have the tendency to degrade model performance [33].

We adopt the Taylor Importance-based pruning approach proposed by [33] which prunes parameters based on global Taylor Importance, where channels with a calculated importance value below a specified threshold are pruned. For each channel, the importance is calculated after the first forward pass as follows,

$$\mathcal{I}_m^{(2)}(\mathbf{W}) = (g_m w_m)^2, \tag{4}$$

where $g_m = \frac{\partial E}{\partial w_m}$ contains the elements of the loss gradient computed from a single forward pass and w_m includes channel weights.

RT-DETR-L's backbone, based on the standard CSP-Darknet architecture requires image inputs to be split across red, green, and blue (RGB) channels [2]. Since blue coloration in images are highly sensitive to object shadows and lighting, the channels across the backbone associated with this input may provide an adversarial effect in overhead vehicle detection [34]. Additionally, due to heavy class imbalances and poor characterization of certain object classes within the original overhead vehicle detection dataset [12], labels pertaining to any class other than cars are deleted and relabeled under a single category. To minimize the possible influence of shadow and lighting effects on the model predictions and account for this simplification in the detection task, a 33% the channels with the least importance across the entirety of the model backbone are zeroed out using the Pytorch pruning library [27].

Consequently, the downstream architecture must also account for this backbone channel reduction. We account for such through applying a forward pass using the pruned model, computing the Taylor importance metric across all attention heads within the Attention-based Intra-scale Feature Interaction (AIFI) module, and pruning a third of the least important attention heads to match with the channel pruning ratio of 33%. After pruning, 50 epochs of training are applied with a 10 epoch warm-up schedule using the same learning rates as the teacher models to stabilize and fine tune pruned model parameters. The PDVS-RT-DETR student architecture obtained is illustrated in Figure 1.

IV. Results

Commonly applied metrics in evaluating prediction accuracy of object detectors across a testing dataset are model mean Average Precision (mAP) and Recall. mAP provides a measure of how capable a model is of making true positive predictions across all predicted object instances. When reported as $mAP@0.5$ or $mAP@[0.5:0.95]$, the numerical value specifies the IoU threshold required for a detection to be counted as a true positive. On the other hand, Recall provides a measure of how capable a model is of detecting all object instances within an image. Formulation used in calculating these metrics is presented in the following,

$$Recall = \frac{TP}{TP + FN}, \tag{5}$$

$$mAP@0.5 = \frac{1}{n}\sum_{i=1}^{n}\frac{TP}{TP + FP}, \tag{6}$$

where TP are the true positives, FN are the false negatives, FP are the false positives, and n represents the number of classes. Ten classes are observed in [12], but due to the dataset skew around car objects, only 1 class provides meaningful Precision and Recall metrics. Thus, mAP and Average Precision (AP) take on the same values for this analysis.

A. Teacher Model

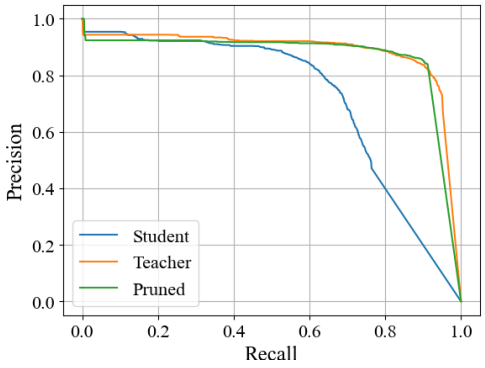
When evaluating prediction metrics across all five cross-validated teacher models on the test dataset of 702 images, the best performing model reached a $mAP@0.5$ of 88.6%, $mAP@[0.5:0.95]$ of 41.1%, and a Recall value of 95.2% for cars, indicating strong class-wise detection capability (Table I). The drop in detection when averaging across $mAP@[0.5:0.95]$ may be due to inconsistent dataset labeling procedures or perspective distortions.

The operational frame rate of the teacher, $Speed_{inf(T)}$, is calculated as the reciprocal of the mean inference time, $Time_{inf}$, observed across the validation set. From this calculation, the frame rate of the teacher model is calculated as 49.5 frames per second (FPS) on a NVIDIA RTX3060 GPU and the computational cost of this model is noted as 103.5 GFLOPs (Table I).

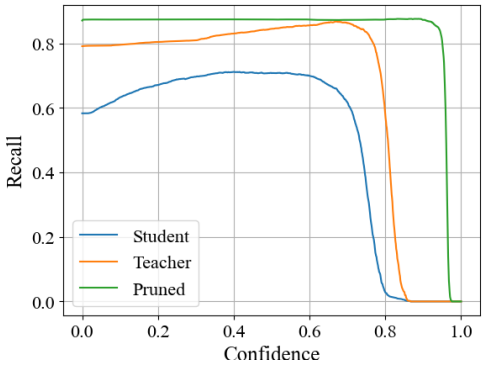
B. DVS-RT-DETR

DVS-RT-DETR retains a significant portion of teacher model performance, achieving a Recall of 76.8%, $mAP@0.5$ of 80.6%, and $mAP@[0.5:0.95]$ of 34.8% (Table I). This drop in performance is likely due to the selected training schedule length being inadequate for the reduced-depth backbone to learn important generalizations [35]. However, [36] found that early stoppage of teacher models in KD applications can provide more significant distillation parameters. The inference speed on a NVIDIA RTX3060 GPU is calculated as 54.3 FPS, which is a 9.8% decrease from the RT-DETR-L teacher model (Table I).

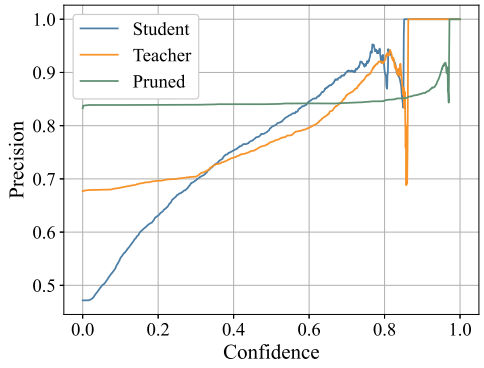
After pruning DVS-RT-DETRand tailoring its detection task to car objects, PDVS-RT-DETR achieved a Recall of 90.1% and a $mAP@0.5$ of 86.8% (Table I). Precision remains above 80% across most predictions, indicating accurate detection with minimal over- or under-prediction (Figure 3). The inference speed is then calculated as 61.7 FPS on a RTX3060 GPU with 78.5 GFLOPs (Table I). Overall comparisons of the RT-DETR-L model, the reduced-depth model, and the pruned reduced-depth model are presented in Table I.



(a) Precision vs Recall



(b) Recall vs Confidence



(c) Precision vs Confidence

Fig. 3: Comparison of the RT-DETR-L (Teacher), DVS-RT-DETR (Student), and PDVS-RT-DETR (Pruned Student) Detection Metrics on the Car Class.

C. Out of Distribution Performance

To qualitatively assess model performance, inference from all three models are applied on traffic monitoring footage taken from an aerial view with a UAV [25]. As this footage

TABLE I: Performance comparison of RT-DETR-L, DVS-RT-DETR, and PDVS-RT-DETR models on car detection.

Model	Accuracy Metrics			Efficiency Metrics		
	Recall (%)	mAP@0.5 (%)	mAP@[0.5:0.95] (%)	Inf. Time (ms)	FPS	GFLOPs
Teacher	95.2	88.6	41.1	20.2	49.5	103.5
Student	76.8	80.6	34.8	18.4	54.3	87.3
Pruned Student	90.1	86.8	38.5	16.2	61.7	78.5

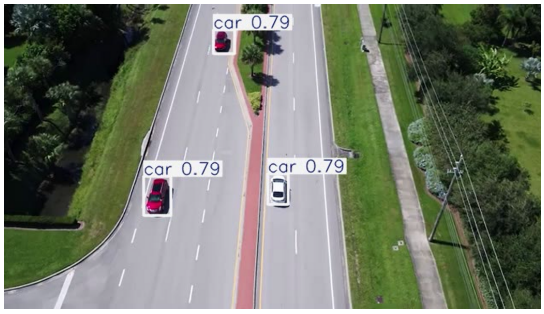
taken is completely outside of the overhead vehicle dataset utilized in model training and validation, it is unseen to the applied models and no bias or overfitting effects will influence prediction results.

Figure 4 illustrates model detection performance across the same sample frame for all three models. It is noted that the PDVS-RT-DETR model predicts vehicle objects with a similar accuracy as the full scale RT-DETR-L model, but with higher confidence values. Another important observation is that the DVS-RT-DETR model struggles with false positive shadow detections while PDVS-RT-DETR does not experience this issue. This is indicative that the Taylor Importance pruning method implemented is effective in rejecting irrelevant features captured in lighting sensitive channels.

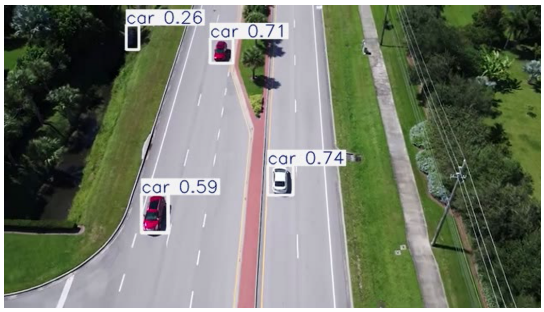
A comparison is also made between PDVS-RT-DETRand currenty state-of-the-art methods in YOLO11M and RTDETR-L for vehicle detection in complex and occluded environments. To justify the claim that self-attention applied across the image space improves performance in occluded and complex environments [11], model inference is applied on a test set image containing multiple detection instances occluded by tree branches (Figure 5).The YOLO11M model is trained across the entirety of the training set without cross-validation using a standard 100 Epoch schedule while the RTDETR-L and PDVS-RT-DETRmodels obtained from this study are reapplied.

V. Discussion

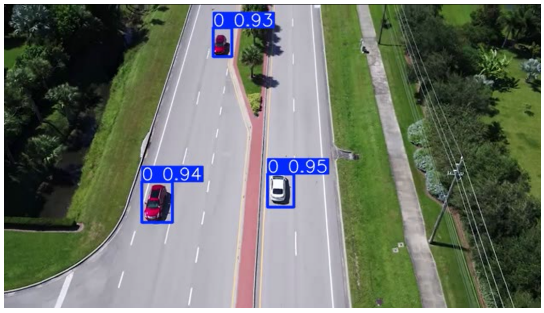
From the results in Table I, it can be seen that although the pruned student performs at a 5.1% lower Recall and a 1.8% lower mAP@0.5 than the teacher, it is capable of operating at a 24.7% faster frame rate (From 49.5 FPS to 61.7 FPS on a NVIDIA GTX3060). Figure 3 illustrates how full scale RT-DETR-L and PDVS-RT-DETR models possess similar object detection capabilities despite the drop in performance of the DVS-RT-DETR model. This aligns with the results presented by [36], while also confirming that the architectural reduction applied is valid for the given classification task. Furthermore, the pruning method implemented had a beneficial effect on the student model's performance in both prediction accuracy and inference speed, which indicates that an oversized model encodes adversarial features enabling the model to produce more false positive predictions. As the overhead vehicle dataset [12] applied contains a significant amount of truncated object instances, larger and unpruned models may encode finer features irrelevant to the generalization of vehicle classes and confuse out of distribution objects or shadows as false positives.



(a) RT-DETR-L Teacher Model



(b) DVS-RT-DETR



(c) PDVS-RT-DETR

Fig. 4: RT-DETR-L, DVS-RT-DETR, and PDVS-RT-DETRmodel performance on unseen video footage. The Pruned Student model maintains high accuracy and increased confidence.

This is further reflected in the test metrics of the overhead vehicle detection, where the Precision-Confidence curve (Figure 3c) displays both DVS-RT-DETR and RT-DETR-L models experiencing a decrease in precision as the confidence threshold is lowered. Simply put, if the specified confidence threshold is decreased to enable more object detections, RT-DETR-L and DVS-RT-DETR models detect more false positive classifications represented through a lowered model precision, while the PDVS-RT-DETR model does not and retains a precision value over 80%. This behavior is consistent with results from out-of-distribution video tests, where the pruned model avoided shadow false positives that were present in both the DVS-RT-DETR and RT-DETR-L models. Ultimately, by pruning the channels associated with these features and their respective attention heads, they are omitted

(a) YOLO11 Detection Result Across Validation Sample Image

(b) RTDETR-L Detection Result Across Validation Sample Image

(c) PDVS-RTDETR-L Detection Result Across Validation Sample Image

Fig. 5: Visual Comparison of YOLO11, PDVS-RT-DETR, and RTDETR Methods on Overhead Vehicle Detection.

from the CCFM feature fusion applied and unconsidered in the decoder's prediction task. The effect of this pruning is also limited in impact on the KD framework applied despite the distilled feature maps pertaining to the CCFM module. Since an importance based pruning method is applied, pruned features have limited effect on object detection loss. Thus, the applied loss weighting scheme can prevent additional KD loss introduced by feature pruning from influencing gradient updates towards a path that is harmful to the detection task.

When comparing detection transformer models such as PDVS-RT-DETRand RTDETR-L with recent YOLO architecture, greater detection confidence and accuracy is observed across complex and occluded environments (Figure 5). This justifies the claim made by [11] and suggests that with enough reductions, transformer models may overtake YOLO architecture in the field of real-time object detection. Additionaly, from this figure the bounding box placements between RTDETR-L and PDVS-RT-DETRare observed to be consistent which is an indication that the KD framework applied was successful in adapting teacher performance to student performance.

VI. Conclusions and Future Work

In the original ultralytics documentation, the RT-DETR-L model is optimized based on an 80 class classification task as the author attempts to develop a universal recognition model from the MS COCO dataset [4]. Typically, applications of object detection require a fewer number of classes such that RT-DETR-L is no longer optimal. Through defining a training scheme utilizing KD with a multi-teachers ensemble in refining RT-DETR-L to develop problem specific models, the first half of the applied training illustrates the ability of RT-DETR-L predicting across a simplified detection task with a 50% reduction in encoder backbone depth. By then pruning this model, significant improvements in model accuracy and performance are observed at a much faster frame rate. The final weights of PDVS-RT-DETR may also serve as a foundational model for training vehicle detectors at a lower computational cost than RT-DETR-L. However, a limitation in the application of PDVS-RT-DETR is that lighting and weather effects are not accounted for in the overhead vehicle detection dataset [12]. Although the final distilled model showed improvements in shadow rejection through Taylor Importance pruning, degraded performance is expected in nighttime conditions and lighting due to the limited training exposure the model has in these environments. Additionally, occlusions from weather conditions, such as snow and rain, may negatively impact performance.

The ultimate goal of this object detector is to ultimately enable Unmanned Aerial Vehicles (UAVs) to track and interact with moving vehicles on a real-time basis through Visual Servoing processes. Because of so, performance time metrics on smaller light weight board GPUs such as Jetson Orin Nano are essential in understanding how this software may interface with UAV applications. Additional training of DVS-RT-DETR and PDVS-RT-DETR may also be performed with a more comprehensive dataset to allow for better object generalization through a process known as Grokking [35].

The KD framework could also be improved if the teacher is trained across a variety of datasets differing from the student model dataset. This allows the effects of the KD framework to be amplified, as the distilled features across image sequences can reflect a wider range of teacher-learned representations rather than being dominated by one or two high-performing teacher models. Additional reductions in decoder architecture may also be attempted when the number of classes in the detection task differ greatly from the 80 classes utilized in initializing RT-DETR-L framework.

REFERENCES

[1] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, "DETRs Beat YOLOs on real-time object detection," *arXiv preprint arXiv:2304.08069v3*, 2024.

[2] W. Lv, Y. Zhao, Q. Chang, K. Huang, G. Wang, and Y. Liu, "RTDETRv2: All-in-one detection transformer beats YOLO and DINO," *arXiv preprint arXiv:2407.17140v1*, 2024.

[3] X. He, Y. Zhang, and Q. Zhan, "AIN-YOLO: A lightweight YOLO network with attention-based inceptionnext and knowledge distillation for underwater object detection," *Advanced Engineering Informatics*, vol. 66, p. 103504, 07 2025.

[4] Ultralytics, "Ultralytics: YOLO and Vision Models," 2025, gitHub repository, version v8.3.209. [Online]. Available: https://github.com/ultralytics/ultralytics

[5] F. S. Alrayes, N. Ahmad, A. Alshuhail, M. Alshammeri, A. Alqazzaz, H. Alkhiri, J. S. Alqurni, and Y. Said, "Convolutional transform learning based fusion framework for scale invariant long term target detection and tracking in unmanned aerial vehicles," *Scientific Reports*, vol. 15, 12 2025.

[6] S. Badrloo, M. Varshosaz, S. Pirasteh, and J. Li, "Image-based obstacle detection methods for the safe navigation of unmanned vehicles: A review," *Remote Sensing*, vol. 14, no. 15, 2022.

[7] Y. Sun, Z. Sun, and W. Chen, "The evolution of object detection methods," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108458, 2024.

[8] S. Abdul-Khalil, S. Abdul-Rahman, S. Mutalib, S. I. Kamarudin, and S. S. Kamaruddin, "A review on object detection for autonomous mobile robot," *IAES International Journal of Artificial Intelligence*, vol. 12, pp. 1033–1043, 9 2023.

[9] M. Naranjo, D. Fuentes, E. Muelas, E. Díez, L. Ciruelo, C. Alonso, E. Abenza, R. Gómez-Espinosa, and I. Luengo, "Object detection-based system for traffic signs on drone-captured images," *Drones*, vol. 7, 2 2023.

[10] X. Chen, S. Li, Y. Yang, and Y. Wang, "DECO: Unleashing the potential of convnets for query-based detection and segmentation," in *The Thirteenth International Conference on Learning Representations*, 2025.

[11] Z. Rong, G. Wang, Q. Sun, J. Sun, R. Li, Y. Xu, and L. Yang, "A lightweight and efficient model for safety protection equipment detection based on improved RT-DETR," *IEEE Access*, vol. PP, pp. 1–1, 01 2025.

[12] Artsywelshman, "Overhead vehicle detection computer vision dataset," Roboflow Universe, 2025, https://universe.roboflow.com/artsywelshman/overhead-vehicle-detection-t1llw.

[13] V. Dewangan, A. Saxena, R. Thakur, and S. Tripathi, "Application of image processing techniques for uav detection using deep learning and distance-wise analysis," *Drones*, vol. 7, 3 2023.

[14] V. R. Yallamraju and S. Jana, "Dynamic object detection and tracking system on unmanned aerial vehicles for surveillance applications using regionvit-based adaptive multi-scale YOLOv8," *Computational Intelligence*, vol. 41, 8 2025.

[15] L. Y. Xu, Y. F. Zhao, Y. H. Zhai, L. M. Huang, and C. W. Ruan, "Small object detection in uav images based on YOLOv8n," *International Journal of Computational Intelligence Systems*, vol. 17, 12 2024.

[16] J. Liu, Y. Wang, C. Zhang, D. Wang, J. Zheng, J. Zhou, Y. Wang, X. Wang, and Y. Chai, "MiniSAM: A lightweight segmentation model with pruning and knowledge distillation," in *6th International Conference on Robotics, Intelligent Control and Artificial Intelligence, RICAI*, 2024, pp. 1330–1335.

[17] C. Liu, S. Peng, S. Liu, and J. Li, "RT-DETR-LGP: An effective defect detection method for light guide plates via multiscale feature fusion and knowledge distillation," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, 2025.

[18] G. Liang, S. Yu, and S. Han, "RT-DETR-FFD: A knowledge distillation-enhanced lightweight model for printed fabric defect detection," *Electronics*, vol. 14, 7 2025.

[19] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 6 2021.

[20] S. Y. Yang, H. Y. Cheng, and C. C. Yu, "Real-time object detection and tracking for unmanned aerial vehicles based on convolutional neural networks," *Electronics*, vol. 12, 12 2023.

[21] K. Shim, I. Choi, W. Sung, and J. Choi, "Layer-wise pruning of transformer attention heads for efficient language modeling," *arXiv preprint arXiv:2110.03252v1*, 2021.

[22] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common Objects in Context," 2015.

[23] Google LLC, "Open images dataset v7," https://storage.googleapis.com/openimages/web/index.html, 2022, accessed: 2026-02-02.

[24] H. Li, B. Singh, M. Najibi, Z. Wu, and L. S. Davis, "An analysis of pre-training on object detection," *arXiv preprint arXiv:1904.05871v1*, 2019.

[25] A. A. R. Experts, "Drone overhead video footage of a car accident," YouTube video, 2025, https://www.youtube.com/watch?v=NArNkuQsUD8.

[26] S. Arlot and M. Lerasle, "Choice of v for v-fold cross-validation in least-squares density estimation," *Journal of Machine Learning Research*, vol. 17, no. 208, pp. 1–50, 2016.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[28] F. Yuan, L. Shou, J. Pei, W. Lin, M. Gong, Y. Fu, and D. Jiang, "Reinforced multi-teacher selection for knowledge distillation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, pp. 14 284–14 291, May 2021.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[30] D. Weinshall, G. Cohen, and D. Amir, "Curriculum learning by transfer learning: Theory and experiments with deep networks," *arXiv preprint arXiv:1802.03796v4*, 2018.

[31] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," *Information*, vol. 11, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/125

[32] H. Peng, X. Lv, Y. Bai, Z. Yao, J. Zhang, L. Hou, and J. Li, "Pre-training distillation for large language models: A design space exploration," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, W. Che, J. Nabende, E. Shutova, and M. T. Pilehvar, Eds. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 3603–3618. [Online]. Available: https://aclanthology.org/2025.acl-long.181/

[33] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11 264–11 272.

[34] J. Tian, J. Sun, and Y. Tang, "Tricolor attenuation model for shadow detection," *IEEE Transactions on Image Processing*, vol. 18, no. 10, pp. 2355–2363, 2009.

[35] B. Wang, X. Yue, Y. Su, and H. Sun, "Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization," in *Advances in Neural Information Processing Systems*, vol. 37. Curran Associates, Inc., 2024, pp. 95 238–95 265.

[36] B. Dong, J. Hou, Y. Lu, and Z. Zhang, "Distillation $\approx$ early stopping? harvesting dark knowledge utilizing anisotropic information retrieval for overparameterized neural network," *arXiv preprint arXiv:1910.01255v1*, 2019.