On Efficient Object-Detection NAS for ADAS on Edge devices

Diksha Gupta
IBM Research
Singapore
diksha.g@ibm.com

Rhui Dih Lee
IBM Research
Singapore
rhui.dih.lee@ibm.com

Laura Wynter
IBM Research
Singapore
lwynter@sg.ibm.com

***Abstract*—Object detection is a crucial building block for Advanced Driving Assistance Systems (ADAS). These systems require real-time accurate detection on resource-constrained edge devices. Deep learning models are emerging as popular techniques over traditional methods with superior performance. A hurdle in deploying these models is the inference time and computational cost of these models, in addition to training challenges for specialized tasks.We address this using supernet training-based neural architecture search (NAS) to obtain a variety of object detection models at a scale specific to the ADAS application. To this end, we consider a convolutional neural network-based object detection model. We produce a palette of CNN models using the CityScapes, and BDD10K datasets, catering to diverse parameters and accuracy tradeoffs. Our resulting models range between 1.8M to 2.6M parameters with an mAP score within in 29.7% to 33.60% on the CityScapes dataset, and 20.08% to 23.35% on BDD10K. Inspired by the popularity of Large Vision Models, we further develop cost-effective transformer-based ADAS Object Detection models. We obtain a palette of transformer models ranging from 69.1M to 113M parameters with mAP score within 28.58% and 32.43% on CityScapes and between 24.31% to 26.51% on the BDD10K dataset.**

***Index Terms*—Object Detection, Supernet Training, Weight-Sharing NAS, ADAS**

I. Introduction

Object detection is one of the fundamental tasks in computer vision, forming the basis for numerous downstream tasks such as instance segmentation, image captioning, object tracking, and many more. It is a two-fold process that creates a bounding box around objects and then assigns labels to them. It finds applications in several diverse domains, such as video surveillance, autonomous driving, and healthcare monitoring.

Object detection is foundational to Advanced Driving Assistance Systems (ADAS). It improves transportation system safety through human-machine interfacing, encompassing pedestrian detection, parking assist, driver drowsiness detection, lane detection, and many other tasks that assist drivers with safety-critical functionality.

With the recent advances in deep learning, convolutional neural networks (CNN) models have emerged as achieving unprecedented improvement in accuracy for object detection tasks over traditional handcrafted approaches [1]–[3]. Specifically, multiple research works have studied the problem of object detection using CNNs for different ADAS tasks [4]–[6].

In conjunction with the transformer-based architectures redefining the field of NLP, Vision Transformers have emerged as a competitive replacement to state-of-the-art (SOTA) CNN models on image classification tasks [7]. Motivated by this, recent works have attempted transformer-based models for ADAS object detection [8], [9].

Despite such enhancements in the capabilities of deep learning models for object detection, there is a lack of rampant adoption of these techniques in the real world. The major contributing factors to this are:

1) Due to the time-critical nature of ADAS tasks, these systems require deployment locally on resource-constrained edge devices. Such devices lack expensive inference hardware and have limited memory to store the models. Existing solutions fail to meet these performance requirements. Therefore, developing models that cater to diverse deployment scenarios is a critical problem.
2) Determining the best architecture for a given set of deployment constraints is challenging as well as computationally expensive. Supernet-based Neural Architectural Search (NAS) is an efficient technique addressing this issue in the image classification domain [10], [11]. A large amount of training data is required for the supernet training. This may not always be available for different ADAS tasks. Hence, NAS in the presence of limited training data for ADAS-specific applications is an open problem.
3) Training transformer-based object detection models is computationally expensive. For instance, the training of a single ViTDet on the COCO dataset requires at least 50 epochs using 64 A100 GPUs [12]. This cost compounds as we move to supernet training. Hence, there is a need for less resource-intensive training strategies.

In this work, we solve the problem of efficient training of object detection model for the ADAS task deployable on diverse edge devices. To this end, we make the following contributions:

1) We propose supernet-based NAS techniques for ADAS CNN-based as well as transformer-based Object Detection to obtain a palette of models catering to diverse deployment scenarios. We demonstrate its performance on

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00183

a CNN-based object detection model and a transformer-based object detection model. We produce a palette of models for class of architectures meeting diverse device constraints: a) CNN-based in the range of 1.8M to 2.6M parameters, and b) Transformer-based in the range of 69.1M to 113M parameters, without severe performance drop.

2) To overcome the problem of limited task-specific training data, we propose first pre-training the supernet on a large generic object-detection dataset, followed by training on ADAS-specific data. We empirically illustrate the efficacy of our strategy on two small ADAS datasets consisting of at most 10,000 samples each, namely: the CityScapes and BDD10K datasets.
3) We propose an efficient transformer supernet-training strategy, even with limited training data. Instead of supernet pretraining on a large object detection dataset, we load pretrained weights of the static model to the supernet's largest subnet (maxnet). This significantly reduces computational requirements from 64 A100 GPUs to a single V100 GPU for our training strategy. However, this approach leads to suboptimal performance in smaller subnets. To mitigate this performance degradation, we incorporate in-place knowledge distillation from the maxnet to the remaining subnets during the transformer supernet training, yielding better-performing smaller subnets but falling short of the desired level. To further boost the performance of these smaller subnets, we explore various knowledge distillation loss functions and empirically determine that the Pearson Correlation Coefficient-based Knowledge Distillation (PKD) method from [13] is the most effective.

We organize the rest of the paper as follows: In Section II, we provide the related work on the state of object detection in ADAS, and supernet-based NAS techniques. Then, we discuss our methodology in Section III, followed by the experimental setup and results in Section IV, and Conclusion & Discussion in Section V.

II. Related Work

In this section, we review the existing literature on object detection for ADAS, supernet training-based Neural Architectural Search, and object detection transformers. Several other works propose different techniques for inference on edge devices such as pruning [14], [15], quantization [16], image partitioning [17]. These techniques independently result in optimised models catering to specific deployment constraints, which require additional computational effort per set of constraints. On the other hand, our approach only requires a one-time training computational cost and produces a palette of models for diverse deployment constraints. Despite these shortcomings, the aforementioned techniques can further enhance the performance of our output models by incorporating them on top of our proposed technique.

A. Object Detection in Advanced Driving Assistance Systems

Object detection serves as a fundamental building block in ADAS, critical to the automobile industry. Several works propose CNN-based object detection solutions, but their application to ADAS is limited. For instance, object scale variation and occlusion are challenges faced by existing ADAS solutions. [4] addresses these issues; however, the resulting models depend on GPU for inference. Given the time-critical nature of the ADAS tasks and the limited computational capacity of edge devices, there is a need to eliminate GPU dependence for inference. [18] proposes EdgeYOLO, a modified YOLOv4 model that eliminates the redundancies in the network backbone and the feature extraction stage. Despite these optimizations to fit on edge devices, the resulting model still requires a GPU for inference. Generalizing to diverse resource constraint devices is identified as future work. [19] presents a generic strategy for the YOLOv5 model, utilizing the RepNAS strategy from [20] to determine an efficient architecture. The NAS strategy requires retraining for every distinct device constraint, which is orthogonal to our training approach. We propose a one-time training strategy to accommodate diverse device constraints.

B. Supernet-based Neural Architectural Search

Current one-shot supernet-training-based algorithms for neural architectural search primarily concentrate on CNN models intended for image classification [21], [22]. As an extension of this, Chen et al. introduce DetNAS in [23] to enhance the design of CNN backbones for object detection. They first pretrain the backbone supernet on the ImageNet dataset, followed by supernet training incorporating the detector neck and head on the detection COCO dataset. However, this method is computationally demanding and performs poorly when fine-tuning for detection on smaller datasets. Another approach, SPNAS [24], proposes model backbone supernet training, requiring multiple stages of training on both the Imagenet dataset and object detection dataset, resulting in computational inefficiency. It is important to note that while these techniques work well on large object detection datasets, they are less suitable for ADAS application. In our work, we aim to overcome this challenge.

C. Transformer-based Object Detection

Motivated by the SOTA performance of ViT in image classification tasks [7], [25] et al. propose replacing the classifier head with a detection head on the ViT backbone for COCO object detection. Building upon this idea, Li et al. [12] further enhance the approach by integrating a simple feature pyramid with the ViT backbone. These models exhibit competitiveness with SOTA CNN models but rely on large target datasets. Moreover, their non-edge-friendly nature, relying on GPUs for inference, poses limitations. To address these shortcomings for ADAS application, we extend the work by [12] to obtain a palette of models. In another line of research, a pure transformer-based object detection model, the DEtection TRansformer (DETR), is explored [26]. However, we focus

on plain ViT backbone architecture to develop efficient NAS for ADAS object detection, which is orthogonal to DETR.

III. Methodology

Our aim is to design an efficient NAS strategy for object detection backbones, resulting in tailored subnets suitable for deployment on diverse devices. A major constraint in our quest is the limited ADAS-specific training data, which leads to the supernet rapidly overfitting, thereby producing subnets that under-perform. Hence, we propose an alternative approach.

A. Supernet Training Strategy

We define our supernet as a set of dynamic layers with elastic dimensions, using the weight entanglement strategy similar to [10]. The basic idea is to share weights among different layers for overlapping parts in each layer. The elastic dimensions define our neural architectural search space $\mathcal{A}$, where the supernet is the largest network defined by it. Additionally, for a given architecture $\alpha \in \mathcal{A}$, we denote a subnet by $\mathcal{S}^{\alpha}$, consisting of a subset of layers, each containing blocks as specified by α. Note that weights $\mathbf{w}$ are shared among the various subnets and updated once during each training iteration.

We present our pseudo-code in Algorithm 1. In a given iteration, for each mini-batch of data from the dataset D, we employ the sandwich rule from [27] for the subnet sampling in our NAS space $\mathcal{A}$. This includes uniformly sampling a subnet from $\mathcal{A}$ along with the smallest and largest configurations. We refer to the latter two as the minnet and the maxnet, respectively. This enables pushing the performance of both the smaller as well as the larger subnets together in each iteration. For each of these subnets, we compute and update the weights $\mathbf{w}$ using aggregate gradient on the loss function $\mathcal{L}$ as:

$$\mathcal{L} = \mathcal{L}^{maxnet} + \mathcal{L}^{uniform} + \mathcal{L}^{minnet} \tag{1}$$

where $\mathcal{L}^{maxnet}$ is the loss value for the maxnet, $\mathcal{L}^{uniform}$ is the loss value for the uniformly sampled subnet in the current iteration, and $\mathcal{L}^{minnet}$ is the loss value for the minnet.

B. CNN-Supernet Training

As mentioned earlier, we have limited training data for ADAS applications. Thus, we first perform supernet training on the large COCO dataset, followed by supernet training on the ADAS dataset (Refer to Step (2) & (3) of Part II of Algorithm 1). We use the CE loss function, $\mathcal{L}_{CE}$ for training our supernet. For the i^{th} mini-batch of data consisting of C classes, suppose $\hat{y}_i$ is the logits of subnet s and y_i be the one hot vector encoding of the ground truth label, then:

$$\mathcal{L}^{s}_{CE}\left(\hat{y}^{s}_{i}, y_i\right) = \mathcal{H}\left(softmax\left(\hat{y}^{s}_{i}\right), y_i\right)$$

where $\mathcal{H}(x, z) = -\sum_{c=1}^{C} x_c \log z_c$ is the cross entropy. Substituting this in Eq.1 for Step 8, we get our aggregated loss function.

Algorithm 1 Object Detection NAS for ADAS

Variables
$\mathcal{S}$: Supernet with elastic dimensions.
$\mathcal{A}$: Neural Architectural Search space
$\mathcal{D}$: Training dataset
$\mathcal{L}$: Loss function

I. Supernet Training Strategy $(\mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{L})$

1: **for** $e = 1, ..., \texttt{epoch}$ **do**
2: **for** $i = 1, ..., d_{\texttt{iter}}$ **do**
3: $x_i, y_i \leftarrow \texttt{data}_i, \texttt{labels}_i$
4: $\texttt{S} \leftarrow$ **`sandwich_sample`**$(\mathcal{A})$
5: **for** each subnet s in S **do**:
6: $\hat{y}^{s}_{i} \leftarrow \mathcal{S}^{s}(x_i; \mathbf{w})$
7: **end for**
8: Compute gradient of loss using $\mathcal{L}$
9: Update weights $\mathbf{w}$ of the supernet $\mathcal{S}(\cdot; \mathbf{w})$.
10: **end for**
11: **end for**
12: **return** $\mathcal{S}(\cdot; \mathbf{w})$

II. ADAS CNN-Supernet Training Strategy

1: $\forall s \in \{maxnet, uniform, minnet\}, \mathcal{L}^{s} \leftarrow \mathcal{L}^{s}_{CE}$
2: $\mathcal{L} \leftarrow \mathcal{L}^{maxnet} + \mathcal{L}^{uniform} + \mathcal{L}^{minnet}$
3: $\mathcal{S}(\cdot; \mathbf{w}^{\texttt{P}}) \leftarrow$ Train the supernet $\mathcal{S}$ on pretrain dataset $\mathcal{D}^{\texttt{P}}$ using Supernet Training Strategy $(\mathcal{S}, \mathcal{D}^{\texttt{P}}, \mathcal{L})$.
4: $\mathcal{S}(\cdot; \mathbf{w}) \leftarrow$ Train the supernet $\mathcal{S}(\cdot; \mathbf{w}^{\texttt{P}})$ on ADAS dataset $\mathcal{D}^{\texttt{ADAS}}$ using Supernet Training Strategy $(\mathcal{S}, \mathcal{D}^{\texttt{ADAS}}, \mathcal{L})$.

III. ADAS Transformer-Supernet Training Strategy

1: $\mathcal{L}^{maxnet} \leftarrow \mathcal{L}^{maxnet}_{CE}$,
2: $\forall s \in \{uniform, minnet\}, \mathcal{L}^{s} \leftarrow (1-\lambda)\mathcal{L}^{s}_{CE} + \lambda\mathcal{L}^{s}_{IPD}$
3: $\mathcal{L} \leftarrow \mathcal{L}^{maxnet} + \mathcal{L}^{uniform} + \mathcal{L}^{minnet}$
4: $\mathcal{S}(\cdot; \mathbf{w}^{\texttt{P}}) \leftarrow$ Initialize the supernet $\mathcal{S}$ with maxnet pre-train weights $\mathbf{w}^{\texttt{P}}$
5: $\mathcal{S}(\cdot; \mathbf{w}) \leftarrow$ Train the supernet $\mathcal{S}(\cdot; \mathbf{w}^{\texttt{P}})$ on ADAS dataset $\mathcal{D}^{\texttt{ADAS}}$ using Supernet Training Strategy $(\mathcal{S}, \mathcal{D}^{\texttt{ADAS}}, \mathcal{L})$.

C. Enhanced Strategy for Transformers

Supernet training on the large corpus for transformer-based models is computationally expensive. For instance, training the plain VitDet from [12] on the COCO dataset requires 64 A100 GPUs for at least 50 epochs, processing 1 image per GPU. Supernet training on such a model would require greater computational resources. Hence, we replace Step 2 from CNN Supernet training with using pre-trained maxnet weights from a larger object detection dataset such as COCO for a warm start. This reduces the computational requirement to a single V100 GPU in our experiments for ADAS datasets (See Section IV). Using warm start in isolation does not result in well-performing subnets. We found that the minnets trained using Strategy III from Algorithm 1 with $\mathcal{L} = \sum_{s \in \{maxnet, uniform, minnet\}} \mathcal{L}^{s}_{CE}$ considerably under-

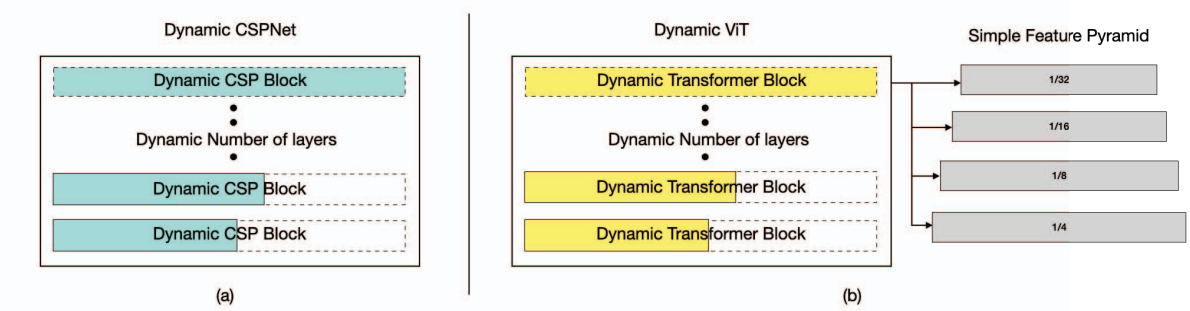

Fig. 1. Elasticity of Object Detection Models. We introduce elasticity into the backbones of our models: (a) For the CNN-based architecture, we introduce elasticity in the CSPNet backbone through the number of dynamic CSP blocks, each equipped with elastic number of channels and depth. (b) For the transformer-based architecture, we introduce the elasticity into the ViT backbone focusing only the ViT transformer blocks with dynamic number of layers, hidden dimension and intermediate dimension, and leave the simple feature pyramid untouched. For details of our elastic dimensions, refer to Tables I and II.

perform (See Section IV-C).

To combat the issue of under-performing subnets, we use Inplace Knowledge Distillation proposed by Yu et al. in [27]. It jointly trains all the subnets using supervised training, along with distilling logits from some subnets onto others. We use the following in our strategy: Train the maxnet in a supervised manner and distill its weighted soft logits to both the uniform subnet and the minnet. This corresponds to Step 1 and 2 of Strategy III of Algorithm 1. Suppose $\mathcal{L}_{IPD}$ denotes the in-place distillation loss, then using notation from Section III-B for the i^{th} minibatch of data, and for $s \in \{uniform, minnet\}, t = maxnet$:

$$\mathcal{L}^s_{IPD}(\hat{y}^s_i, \hat{y}^t_i) = \tau^2 \mathcal{D}_{KL}\big(softmax(\hat{y}^s_i s/\tau), softmax(\hat{y}^t_i/\tau)\big) \quad (2)$$

where $\mathcal{D}_{KL}$ is the Kullback-Leibler(KL) divergence, and τ is the temperature hyperparameter used to soften the logits from the output of the teacher [28]. However, we found in our experiments that utilizing KL divergence between the classification head outputs of the maxnet and the subnets for object detection task leads to limited performance gains of the subnets (see Section IV-C).

A recent work by Cao et al. [13] presents a novel knowledge distillation approach for object detection tasks focusing on the Feature Pyramid Network (FPN). They propose normalizing teacher and student features, followed by minimizing the MSE between normalized features, and call this Knowledge Distillation via Pearson Correlation Coefficient (PKD). Formally, for the i^{th} mini-batch consisting of d_{iter} data samples and for feature map of size $h \times w$, let $m = d_{iter} \times h \times w$ be the effective mini-batch size, and $\mathcal{L}_{FPN}$ denote the FPN loss function using PKD, defined as:

$$\mathcal{L}^s_{FPN} = 1 - \frac{\sum_{j=1}^{m}\big(\hat{y}^s_{i,j} - \mu^s_i\big)\big(\hat{y}^t_{i,j} - \mu^t_i\big)}{\sqrt{\sum_{j=1}^{m}\big(\hat{y}^s_{i,j} - \mu^s_i\big)^2 \sum_{j=1}^{m}\big(\hat{y}^t_{i,j} - \mu^t_i\big)^2}} \quad (3)$$

where, for subnet $k \in \{maxnet, uniform, minnet\}$, $\hat{y}^k_{i,j}$ is the soft logits of subnet k for the j^{th} effective mini-batch of mini-batch i and μ^k_i is the mean over the soft logits of subnet k for mini-batch i. This function focuses on the linear correlation between teacher and student features, while relaxing constraints on their magnitude.

To the best of our knowledge, we are the first to apply this loss for in-place knowledge distillation, and to a transformer backbone for object detection. Using Eq. 3, we modify Step 2 in Strategy III of Algorithm 1 as:

$$\mathcal{L}^s \leftarrow (1 - \lambda)\mathcal{L}^s_{CE} + \lambda \mathcal{L}^s_{FPN}$$

where λ is a hyperparameter for controlling the training and the distillation loss influence on the supernet training. This function bridges the gap between the maxnet and the other subnet activation patterns, resulting in better performing subnets.

We use Evolutionary Search to obtain optimal subnets in our NAS space from the supernet.

IV. Experiments

In this section, we first discuss our implementation details and performance of the supernet training, and ablation study on the transformer supernet training strategy.

A. Implementation Details

Datasets. We consider two datasets relevant to the ADAS object detection task, namely: CityScapes with 5,000 samples; and BDD10K with 10,000 samples. Both these datasets consists of data from 8 classes, namely: person, rider, car, truck, bus, train, motorcycle, and bicycle. Additionally, we use the COCO dataset for CNN supernet pretraining, which consists of 200,000 images and 80 classes of data.

CNN Supernet Training. We perform Supernet training on a CNN-based architecture inspired from [29] with a dynamic CSPNet backbone. The neck and head of our model are static, and we only search for the optimal backbone as illustrated in Figure III-B(a). Table I describes the elasticity of our NAS space. We introduce a dynamic number of channels and depth at Stage II, III, and IV of the CSPNet backbone. We first pretrain the model on COCO Dataset, then replace the detection head and train on the ADAS dataset using Algorithm 1. We perform a one-time supernet pretraining for 250 epochs.

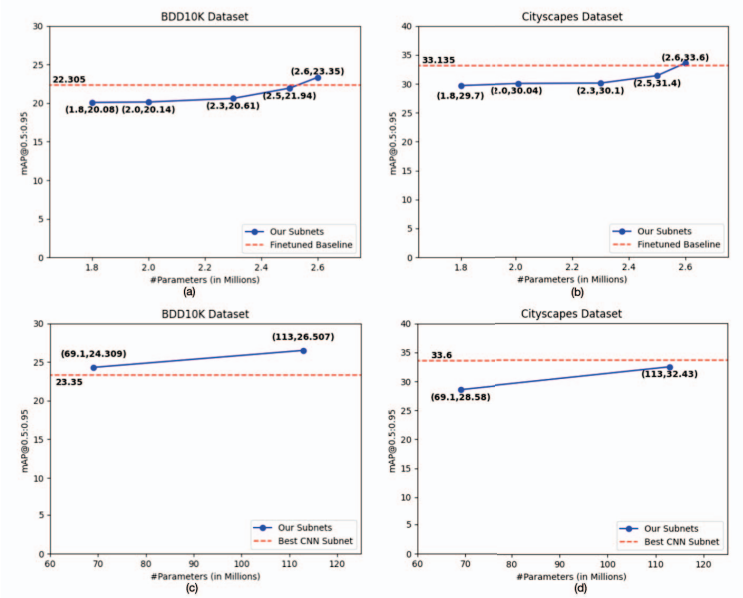

Fig. 2. Results for Object Detection NAS on ADAS datasets. Plots (a) and (b) report the performance of CNN subnets obtained using our supernet training strategy, and plots (c) and (d) report the performance of our transformer maxnet and minnet. Note that the dashed line in (a) and (b) correspond to the finetuned maxnet on the ADAS dataset, whereas for (c) and (d) they are the best performing CNN subnets we obtained from our CNN supernets.

Then, we replace the detection head for each ADAS dataset and train for 50 epochs on an NVIDIA V100 GPU. During training, we use batch size of 8, cosine learning rate scheduler with learning rate of 0.01 and 3 warm up epochs. Additionally, we employ a weight decay of 0.00001 for all model weights except bias, batch norm terms for detector head, and standard stochastic gradient descent (SGD) as optimizer.

Transformer Supernet Training. We perform Supernet training on the ViTDet-Base from [12] for the ADAS task. We illustrate the elasticity of the object detection backbone for our dynamic ViTDet in Figure III-B, where we focus on the ViT transformer blocks, while keeping the simple feature pyramid static. Table II details the elastic dimensions of the transformer backbone for our NAS space. Unlike the CNN model, we load the static pre-train weights for the ViTDet-Base on the maxnet, and only perform supernet training on the ADAS datasets. For a given dataset, we train the supernet for 30 epochs using batch size of 1 on an NVIDIA A100 GPU. We use a linear learning rate scheduler with lr of 0.0001 and 250 warmup iterations, weight decay of 0.1 and AdamW optimizer with epsilon 1e-8.

TABLE I
NAS SPACE OF CNN-BASED CSPNET BACKBONE

Stage	Number of Channels	Depth
I	{32}	{1}
II	{31,48,64}	{1,2}
III	{64,80,96,128}	{1,2}
IV	{128, 144, 160, 192, 224, 256}	{1}

TABLE II
NAS SPACE OF TRANSFORMER-BASED VIT BACKBONE

Dimension	Elasticity
Number of Layers	{10,11,12}
Hidden Dimension	{512,768}
Intermediate Dimension	{2560,3072}
Attention Heads	{12}

B. Results

Figure 2(a) and 2(b) presents our results for the CNN supernet training on BDD10K and Cityscapes datasets. Note that we demonstrate our approach on a small NAS space as proof of concept, with resulting subnets in parameter range of $[1.8M, 2.6M]$ parameters. Additionally, we compare the performance of our subnets against a finetuned CNN network. This was obtained by first training our CNN maxnet on the COCO dataset, followed by finetuning on the ADAS dataset. We report a palette of models that serve as a baseline for our transformer-supernet based approach. We observe that our smallest model is with 4.55 mAP score of the baseline with only half the number of parameters.

Next, we discuss our results for the transformer supernet-training strategy from Algorithm 1 using the PKD loss function from Equation 3 as $\mathcal{L}_{IPD}$. Figure 2(c) and 2(d) reports our final maxnet and minnet performance. We observe that our minnet is within 3.58 mAP score of the maxnet at 61% of the parameters.

Finally, we observe that our transformer model outperforms it's CNN counterparts with an improvement of 3.157 mAP points for the BDD10K dataset. However, our supernet does not perform as well for the Cityscapes dataset, as can been seen in Figure 2(d). We attribute this to the sizable difference in size of the two ADAS datasets, and leave it as a direction for future work.

C. Ablation Study

We compare the performance of transformer subnets obtained using the PKD loss function from Section III-C to basic supernet training loss function, CE loss, and distillation loss function, KL Divergence. Additionally, we study the impact of $\lambda = \{0.1, 0.5, 0.9, 0.95, 1.0\}$ from Eq.2 for both KL Divergence and PKD loss functions when employed using inplace knowledge distillation. For each of the loss functions, we load pretrain weights for the maxnet and then train the supernet only on the ADAS dataset. We run the ablation study on the Cityscapes dataset, and focus on the minnet and maxnet performance.

Table III presents the results of our ablation experiments. We observe that simply using CE loss for our supernet training on the Cityscapes dataset produces a under-performing networks at mAP 21.23 for the minnet, and mAP 25.95 for the maxnet. To improve the performance, we next experimented with inplace knowledge distillation using the the KL-divergence loss function from Eq.2. This improves the performance of our maxnet to become competitive with the CNN counterparts as well as the baseline, but the minnet still underperforms in comparison. Finally, to improve the performance of the minnet, we use the PKD loss function from Eq.3, which pushes the performance of both our maxnet as well as the minnet.

TABLE III
ABLATION RESULTS FOR TRANSFORMER-BASED SUPERNET TRAINING LOSS FUNCTION ON CITYSCAPES DATASET

Loss Function	IPD	λ	mAP_{minnet}	mAP_{maxnet}
CE Loss	×	0	21.23	25.95
KL Divergence	✓	0.10	24.96	30.81
	✓	0.50	26.08	32.17
	✓	0.90	23.33	31.68
	✓	0.95	21.09	30.82
	✓	1.00	6.113	31.39
PKD Loss	✓	0.10	**28.58**	**32.43**
	✓	0.50	27.87	32.10
	✓	0.90	27.22	31.33
	✓	0.95	27.23	33.00
	✓	1.00	23.80	32.37

REFERENCES

[1] 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.

[2] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, p. 103514, 2022.

[3] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[4] J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang, and Z. Xiong, "Enhanced object detection with deep convolutional neural networks for advanced driving assistance," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 4, pp. 1572–1583, 2019.

[5] J. Cao, Y. Pang, J. Xie, F. S. Khan, and L. Shao, "From handcrafted to deep features for pedestrian detection: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 4913–4934, 2021.

[6] J. S. Murthy, G. Siddesh, W.-C. Lai, B. Parameshachari, S. N. Patil, and K. Hemalatha, "Objectdetect: A real-time object detection framework for advanced driver assistant systems using yolov5," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[7] 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.

[8] K. Peng, A. Roitberg, K. Yang, J. Zhang, and R. Stiefelhagen, "Transdarc: Transformer-based driver activity recognition with latent space feature calibration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 278–285, IEEE, 2022.

[9] A. Balasubramaniam and S. Pasricha, "Object detection in autonomous vehicles: Status and open challenges," *arXiv preprint arXiv:2201.07706*, 2022.

[10] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12270–12280, 2021.

[11] K. T. Chitty-Venkata and A. K. Somani, "Neural architecture search survey: A hardware perspective," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.

[12] Y. Li, H. Mao, R. Girshick, and K. He, "Exploring plain vision transformer backbones for object detection," in *European Conference on Computer Vision*, pp. 280–296, Springer, 2022.

[13] W. Cao, Y. Zhang, J. Gao, A. Cheng, K. Cheng, and J. Cheng, "Pkd: General distillation framework for object detectors via pearson correlation coefficient," *Advances in Neural Information Processing Systems*, vol. 35, pp. 15394–15406, 2022.

[14] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, pp. 5058–5066, 2017.

[15] Y. Chen, R. Li, and R. Li, "Hrcp: High-ratio channel pruning for real-time object detection on resource-limited platform," *Neurocomputing*, vol. 463, pp. 155–167, 2021.

[16] P. Chen, J. Liu, B. Zhuang, M. Tan, and C. Shen, "Aqd: Towards accurate quantized object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 104–113, 2021.

[17] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "Flexible high-resolution object detection on edge devices with tunable latency," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pp. 559–572, 2021.

[18] S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, and K. Yu, "Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25345–25360, 2022.

[19] X. Jia, Y. Tong, H. Qiao, M. Li, J. Tong, and B. Liang, "Fast and accurate object detector for autonomous driving based on improved yolov5," *Scientific reports*, vol. 13, no. 1, pp. 1–13, 2023.

[20] M. Zhang, X. Yu, J. Rong, and L. Ou, "Repnas: Searching for efficient re-parameterizing blocks," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 270–275, IEEE, 2023.

[21] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[22] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.

[23] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[24] C. Jiang, H. Xu, W. Zhang, X. Liang, and Z. Li, "Sp-nas: Serial-to-parallel backbone search for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11863–11872, 2020.

[25] J. Beal, E. Kim, E. Tzeng, D. H. Park, A. Zhai, and D. Kislyuk, "Toward transformer-based object detection," *arXiv preprint arXiv:2012.09958*, 2020.

[26] 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*, pp. 213–229, Springer, 2020.

[27] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1803–1811, 2019.

[28] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[29] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023.