DualRes: Production-ready Dynamic Object Detection

Jibril El Hassani Thomas Verelst
Axelera AI
jibril.elhassani@student-cs.fr thomas.verelst@axelera.ai

Abstract

Dynamic Neural Networks (DNNs) have emerged as a promising solution to improve the computational efficiency of deep neural networks by adaptively adjusting inference complexity based on input characteristics. Despite their advantages, the deployment of dynamic networks in real-world applications remains challenging because most methods are hard to adapt for practical use cases such as object detection, in combination with the lacking support of inference infrastructure. In this work, we present a dynamic neural network architecture specifically designed for object detection. Using our method, we build a variety of Pareto-optimal models for object detection on COCO for models in the 7-10 GFLOPs range. Additionally, to measure the routing efficacy, we introduce an evaluation metric that facilitates standardized benchmarking across different dynamic network approaches. Finally, we introduce an evaluation of a deployment pipeline utilizing the ONNX format, thus building a DNN that shows speedup in a realistic deployment scenario. Experimental results demonstrate the performance and practical viability of our approach for efficient object detection in resource-constrained scenarios.

1. Introduction

Convolutional neural networks (CNNs) have been widely adopted across diverse applications. Yet, current architectures might still exhibit notable inefficiencies, particularly the uniform allocation of computational resources to all input samples irrespective of each sample's inherent complexity. 46% of ImageNet samples can be classified as having low difficulty [17], suggesting a potential for computational optimization through adaptive processing strategies. Dynamic Neural Networks (DNNs) have emerged as a paradigm to address this. These networks can dynamically adjust their computational complexity, potentially achiev-

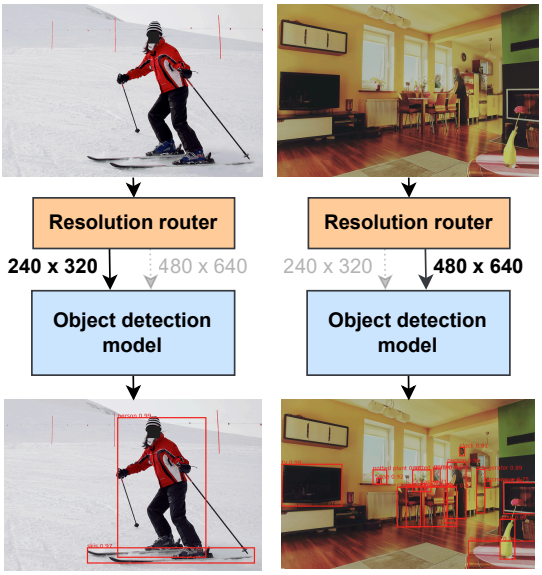


Figure 1. When presented with an easy image (left), the router chooses to run inference at a lower resolution. If the image is deemed harder (right), it is evaluated at high resolution.

ing significant speedups while maintaining or improving accuracy. However, the majority of existing dynamic network research has concentrated primarily on image classification tasks. Most dynamic methods are not directly applicable to object detection due to their architecture [36]. Today, dynamic neural networks have yet to achieve widespread adoption in real-world applications. This might partly stem from the additional complexity of dynamic methods when combined with production deployment pipelines.

To bridge this gap, we propose a simple yet effective method: we dynamically adjust the processing resolution based on the image complexity. When switching between a high-resolution and low-resolution processing, the same object detection network can be pre-compiled for the two resolutions, while sharing the same weight tensors. As such, during inference, the final memory consumption of the network is unchanged.

The processing resolution is chosen by a trainable resolution router, being a rather small neural network. The resolution router is trained end-to-end. In our experiments, we show that the trained router learns a universal policy, where the object detection network can be swapped for other architectures. This allows to plug-and-play combine routers with other object detection networks, without requiring any training. We demonstrate this by establishing a new state-of-the-art on COCO models between 7-10 GFLOPs by integrating our method into EfficientDet [24] models.

The accuracy-FLOPs trade-off serves as a standard benchmark metric to evaluate of the improvement achieved by using dynamic networks. However, this entangles architectural improvements with dynamic routing optimization, making it challenging to identify the contributions of better dynamic routing strategies. For this reason, we propose an evaluation metric that measures the efficacy of the dynamic routing, compared to random selection of the processing resolution.

We summarize our contributions as follows:

1. We present DualRes, a novel dynamic resolution method to accelerate inference in object detection models. Our simple approach achieves state-of-the-art mAP while reducing computations. We evaluate the method on COCO [31] object detection, logo detection [11] and license plate detection [27]. Using this method, we establish new Pareto-optimal COCO object detection models between 7 to 10 GFLOPs.

2. We propose an evaluation metric that quantifies the effectiveness of dynamic routing policies independently of underlying architectural performance. We analyze the results on numerous DNN architectures.

3. We introduce a comprehensive deployment pipeline for dynamic neural networks using ONNX and build a DNN showcasing speedup in a realistic deployment scenario.

2. Related work

2.1. Dynamic neural networks

The early work on dynamic neural networks focused on reducing computational cost by early-exiting easy samples. This technique consists of stopping the computation at an intermediate layer depending on a routing condition. While [34] adds classifiers after different layers in a ResNet [16] model, [9] introduces a multi-scale architecture to decouple coarseness of the feature and earliness of classification. While those methods have been adapted to object detection [19, 35, 38], we argue that the overhead incurred by adding an object detection head at each exit is suboptimal as it adds more parameters and is harder to deploy.

2.2. Dynamic resolution

Changing the resolution of the processed image depending on some routing criterion also arose as a way to reduce compute with minimal loss in accuracy. Indeed, [28] shows that only a small subset of pixels are relevant when it comes to classifying an image. To reduce the number of pixels in a hardware-friendly way, one can downsample the image: [18] mixes dynamic resolution and early-exiting as a way to exploit the spatial redundancy of the easiest images. Moreover, [23] introduces

the architecture of router and classifier, the router choosing a resolution and the classifier running a normal forward pass. This architecture has also been investigated for dynamic pose detection [33]. We build upon this architecture and show that good policies can be achieved without an end-to-end retraining of the whole network. Dynamic resolution processing has also proven its efficiency on video object detection [21, 30]. Elastic-DETR [7] uses dynamic routing to gain mAP on COCO object detection using an attention-based resolution predictor in combination with a DETR [25] model.

3. Methodology

3.1. Dynamic resolution model

We develop a trainable model that is capable of choosing the processing resolution dynamically, where a router block is trained to select the optimal resolution, as pictured in Figure 1. Let $X = (x_1, ..., x_n)$ be a dataset, and $Y = (y_1, ..., y_n)$ the corresponding labels. We define our dual-resolution router as: $R_\alpha : X \to \{0, 1\}$. This router receives an input image and returns 1 if the sample should be processed at high resolution and 0 otherwise. We parametrize this router with $\alpha \in [0, 1]$, defining the target percentage of samples routed to high resolution. We then define the downsampler as:

$$D_\alpha : x \mapsto \begin{cases} x & \text{if } R_\alpha(x) = 1 \\ Dsample(x) & \text{otherwise} \end{cases} \quad (1)$$

Where $Dsample$ is the down sampling operation that we implemented using bilinear interpolation. Let M be our downstream model, e.g. YOLOv6 [20] model. We run inference on a sample x_i:

$$\hat{y}_i = M(D_\alpha(x_i)) \quad (2)$$

3.1.1 Resolution router network

The resolution router needs to be computationally efficient to avoid overhead. To quickly obtain features suitable for object detection, we leverage pretrained network architectures. We use the MobileNetV3 [6] backbone from torchvision's Faster-RCNN [26] model as a feature extractor. We add a final linear layer for binary classification.

3.1.2 Training

The resolution predictor outputs in a discrete space, choosing wether to downsample or not, which makes the output non-differentiable. To enable training of the resolution predictor, we use the Gumbel-Softmax trick [2, 8] as reparametrization in combination a modified loss function. During inference, we choose a threshold and route the samples accordingly:

$$R_\alpha(x) = \begin{cases} 1 & \text{if R}(x) > \alpha \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

With $R(x)$ being the forward pass of the resolution predictor. During training, we first add Gumbel noise to the resolution predictor's output:

$$R_\alpha(x) = \begin{cases} 1 & \text{if } \sigma\left(\dfrac{R(x) + g_1 - g_2}{\tau}\right) > \alpha, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Where σ is the sigmoid function and τ is the temperature hyperparameter. g_1 and g_2 are sampled from a Gumbel distribution:
$g_1 = -log(-log(u_1))$ and $g_2 = -log(-log(u_2))$, with u_1 and u_2 sampled from a uniform distribution $U(0, 1)$ independently. Here, we used the binary formulation of the Gumbel-Softmax trick [29]. τ is a temperature hyperparameter that regulates the learning process: a higher τ leads to a higher bias but helps the gradient flow, whereas a lower τ gives a better approximation of the hard decision but can lead to vanishing gradients [14].

Training the model in this configuration would not allow the router to learn a balanced policy: processing at high resolution typically yields lower loss and hence the network gets stuck in a state where it only uses the high-resolution routing. To address this, we add a term in the loss function in order to regulate this behavior:

$$L_{dynamic} = L_{YOLO} + \gamma * (p - \theta)^2 \quad (5)$$

where L_{YOLO} is the loss of the YOLOv6 model (consisting of class loss and IoU loss), p is the percentage of samples processed at high resolution in the current batch, θ is the desired percentage of high-resolution routing, and γ is a weighting hyperparameter balancing the loss terms. During training, we freeze the downstream object detection model and only train the router weights. This allows faster convergence as it avoids co-optimization of both networks, and allows faster training.

We train the router for 40 epochs using SGD with Nesterov momentum. The initial learning rate is $3.2e^{-4}$, and we use cosine scheduling. We use a momentum of 0.8 and a weight decay of 3.6e-4.

3.2. Quantifying the quality of a policy

To compare how well different policies perform in balancing compute and accuracy, we devise a metric that captures the quality of a dynamic mechanism. We define this metric in this section and conduct a benchmark across recent dynamic methods in Section 4.

3.2.1 The AUC-Policy metric

Let $X = (x_1, ..., x_n)$ be our validation dataset, and $Y = (y_1, ..., y_n)$ the corresponding labels. Our goal is to develop a neural network that maximizes an objective function while minimizing computational cost.

We define an objective metric as follows:

$$\text{Metric} : \hat{Y} \times Y \rightarrow [0, 1] \tag{6}$$

With $\hat{Y}$ being the set of predictions of a model. Examples include accuracy or F1 score for image classification, and mAP for object detection.

Let M_a be the dynamic model. We index the model using β, a parameter that allows us to trade between our objective metric and the amount of compute:

$$M_\beta : X \rightarrow \hat{Y} \tag{7}$$

This paradigm suits many dynamic methods, because β could be a FLOPs constraint parameter (as in [3]), a target percentage of samples routed to a given path (as in our method or in [38]), a confidence threshold (as in [9, 32]) or any other parameter that allows to trade an objective metric for compute. Without loss of generality, we assume $\beta \in [0, 1]$. Let $FLOPs(\beta)$ be the amount of compute in FLOPs per image on the dataset. We add the constraint that the compute is maximal at a 1, minimum at 0.

For a fixed model M_β, and a fixed dataset (X, Y), the corresponding metric depends only on β:

$$\text{Metric}(M_\beta(X), Y) = \frac{1}{n} \sum_{k=1}^{n} \text{Metric}(M_\beta(x_k), y_k) \tag{8}$$

For notation simplicity, in the rest of this section we fix the dataset (X, Y) and the model M, and note

$$\text{Metric}(\beta) = \text{Metric}(M_\beta(X), Y) \tag{9}$$

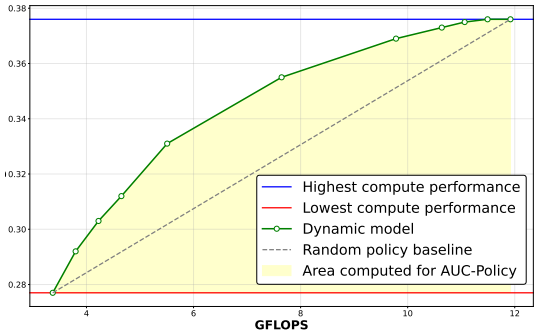
Our goal is to maximize $\text{Metric}(\beta)$ given a computational budget. We propose to use Area Under the Curve to measure the overall quality of a policy across all threshold values:

$$\text{AUC-Policy} = \frac{\int_0^1 \text{Metric}(\beta) - \text{Metric}(0)\, d\beta}{(\text{Metric}(1) - \text{Metric}(0))} \tag{10}$$

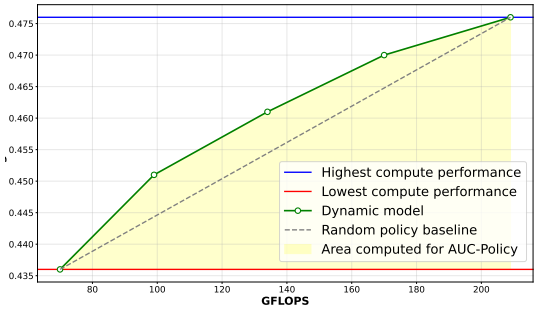
See Figure 2 for an illustration of the metric.

3.3. Deployment pipeline

Dynamic neural networks, while efficient across numerous tasks, have not been adopted widely: the "Gap between Theoretical & Practical Efficiency" was identified by [36] as one of the key challenges to the development of the field. Current deployment of dynamic models is done using standard PyTorch, whereas the deployment workflows for deep neural networks usually include ONNX or compilation. We benchmark the speed of a custom ONNX deployment pipeline against eager PyTorch deployment in Section 4. Our approach is one of the first dynamic neural networks for computer vision to demonstrate inference speedup in realistic deployment scenarios.



(a) mAP-compute trade-off for DualRes YOLO



(b) mAP-compute trade-off for Elastic DETR

Figure 2. Comparing DualRes YOLO and Elastic-DETR using the AUC-Policy metric on COCO dataset. We find an AUC-Policy of 0.700 for DualResYOLO and 0.616 for Elastic-DETR.

3.3.1 Graph mode and eager mode

For model deployment, graph mode is the production standard [15]. It consists of building a static computation graph of the model before runtime to evaluate operations quickly once the model is deployed. It is in contrast to eager mode where the operators are executed at runtime, which could be less efficient in a deployment scenario. Since graph mode relies on graph tracing, its design was until recently incompatible with dynamic neural networks.

3.3.2 Exporting a dynamic PyTorch model to ONNX

Since recently, dynamism in a graph can be handled using `torch.cond` [4]. This operator is a traceable 'if' statement and allows to trace the model using Torch Dynamo and `torch.onnx.export` [5]. Inference can then be performed using the ONNX runtime.

4. Results

4.1. Results on COCO

We benchmark our method against other methods on COCO's validation dataset. Results are presented in Figure 3. We also compare our method with other dynamic methods in Figure 4. Combining our trained router directly with EfficientDet models allows us to achieve the best performance across the entire 7-10 GFLOPs range. We refer to DualRes EfficientDet/X as our model, where X represents the percentage of samples processed by EfficientDet-D2 at high resolution. The remaining samples are processed by EfficientDet-D1 at lower resolution.

4.2. Results on License Plate Detection

License plate detection is a very suitable use case for our method. In fact, the object detection model is able to gain mAP thanks to a smart routing, as seen in Figure 5. A qualitative analysis of the dynamic policy is available in Figure 6. The performance on this task is most likely due to the specialized setting.

4.3. Evaluation of the AUC-Policy metric

We present the results of the proposed AUC-Policy applied to various dynamic methods. In order to compute the AUC-Policy, one needs at least three data points of accuracy-FLOPs. To the best of our knowledge, the methods featured in this study are the only state-of-the-art dynamic methods for which we have been able to find enough data. Table 1 shows the results on image classification and Table 2 shows the results of different object detection methods. Table 3 compares our method across different datasets.

On object detection (Table 2), results show that our method is the best at maintaining mAP when reducing compute. We also notice that specializing on easier use cases allows the model to develop

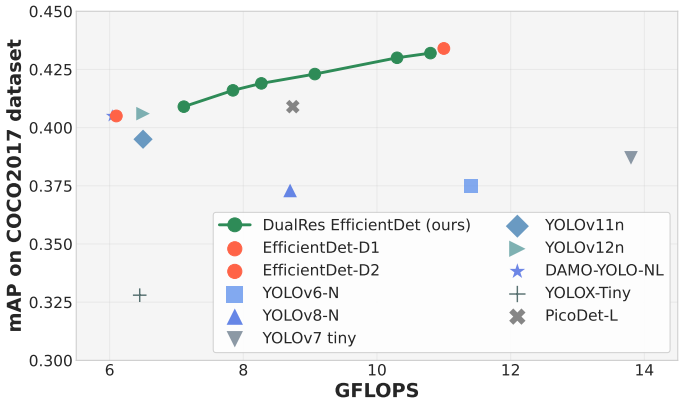


Figure 3. Comparison of state-of-the-art models on COCO object detection.

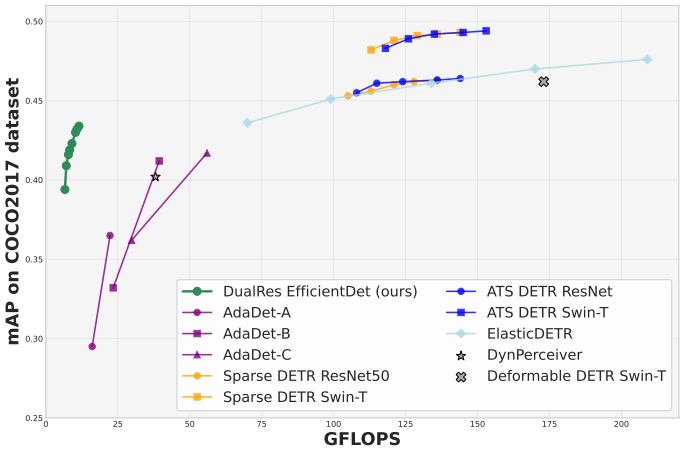


Figure 4. Comparison of our DualRes method to other dynamic models for COCO object detection.

better policies (Table 3): Indeed, on license plate detection, all the objects of interest are rectangles, the routing model can learn to look for rectangles in an image and, depending on their size and number, choose the correct routing. It can also learn to distinguish between cars, trucks and buses, which is a fairly easy task for today's CNNs. On the other hand, logos are often small and have very different shapes and colors. Creating a policy that is effective on all of the 352 brands' logo of the OpenLogo dataset seems to be more challenging.

4.4. Evaluation of the deployment pipeline

We export our dynamic DualRes YOLOv6 model to ONNX, and compare PyTorch eager inference to ONNX runtime. Table 5 shows that the dynamic network, using the `torch.cond` operator for the dynamic routing, runs properly in standard ONNX runtime, achieving inference time improvement over eager PyTorch.

To assess the viability of dynamic methods in production settings, we compare our ONNX deployment pipeline to the eager PyTorch model. To measure the overhead of dynamic routing, we study

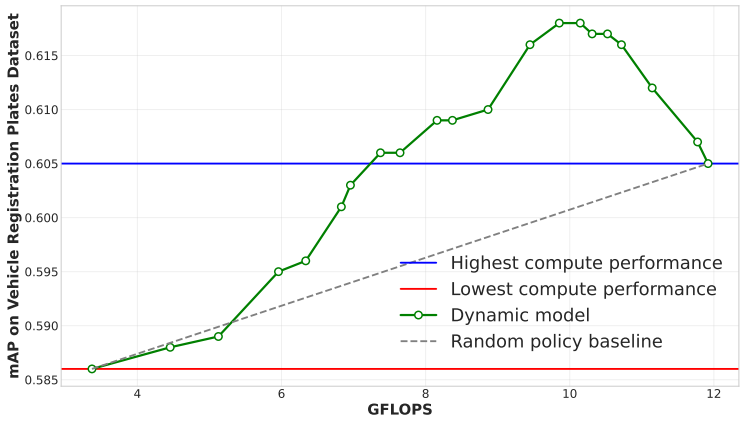
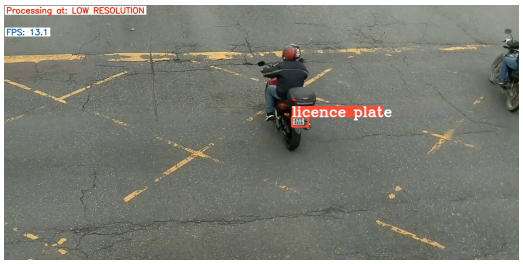
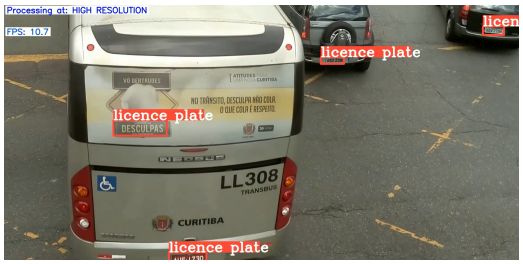


Figure 5. 'mAP vs. compute' trade-off on the Vehicle Registration Plates Dataset. Our method is able to gain up to 1.3 points of mAP simply by downsampling the right images. The AUC-policy is 0.894 which is the best for any object detection model.



(a) Example of an image processed at low resolution : the street is not busy therefore the router recognizes the image as an easy sample



(b) Example of an image processed at high resolution : the bus and cars are not fully on the frame, thus, the router identifies the frame as a hard image.

Figure 6. Qualitative analysis of the DualRes YOLO license plate recognition model : the router identifies an easy (6a) and a hard (6b) sample. Data from [13]

.

Table 1. AUC-Policy of various ImageNet image classification dynamic methods

Model	AUC-Policy
Convnet - AIG [1]	0.493
MSDNet [9]	0.663
IMTA-MSDNet [12]	0.630
RANet [18]	0.683
DVT [37]	0.684
MS-GFNET [10]	0.748
CF-ViT DeiT-S [22]	**0.768**
CF-ViT LV-ViT-S [22]	0.683
OTD-Net (Resnet 18) [39]	0.603

an early-exit network with multiple conditional operations (VGG-SDN [34]). We compare the runtime in eager PyTorch and ONNX. Each experiment is repeated 20 times and we report the mean of the inference times. The routing policy is identical in ONNX and eager PyTorch, ensuring the samples followed identical routes regardless of the type of deployment. All the times reported are in milliseconds.

Table 2. AUC-Policy of various object detection dynamic methods on COCO. AUCP refers to our AUC-Policy metric.

Model	AUCP
Dy-Yolov7-W6 [38]	0.566
Dy-Yolov7-X [38]	0.621
Dy-YOLOv7 [38]	0.614
Elastic DETR [7]	0.616
DualRes Efficientdet (ours)	**0.636**
DualRes YOLOv6 (ours)	**0.700**

Table 3. AUC-Policy of Dual Res YOLOv6 across multiple datasets. OL stands for the OpenLogo dataset and VRPD for the Vehicle Registration Plates Dataset. Increasing the number of classes makes the policy less efficient.

	VRPD	COCO	OL
Number of classes	1	80	352
AUC-Policy	0.894	0.700	0.641

Table 4. CPU inference time of VGG-SDN (in ms) on 100 CIFAR-10 images. The early-exit threshold indicates the confidence threshold above which we can stop computations. A lower threshold means faster exiting, and therefore less compute.

Early-exit threshold	PyTorch CPU time	ONNX CPU time
1.01 (no exiting)	44.9	20.2
0.9	12.9	11.6

4.4.1 Overhead of the early-exit scheme

Deploying the dynamic VGG-SDN on ONNX instead of eager PyTorch a does not provide a significant speedup. Indeed, in this model, there are 6 nested early-exit stages. The successive use of `torch.cond` creates an overhead: in ONNX, the dynamic method is 1.7x faster than its static counterpart. Whereas in eager PyTorch, the dynamic method is 3.5x faster. This suggests that early-exit methods with many exits are less viable in production given the current state of deployment software. This is not the case for DualRes YOLO (see Table 5) as we only use a single `torch.cond` operator.

4.4.2 Inference speed on COCO object detection

Table 5. CPU inference time of 128 COCO images, for a given percentage of high-resolution image, in PyTorch eager mode and ONNX runtime. We report the average inference time per image.

High-res percentage	PyTorch	ONNX
100%	179 ms	161 ms
50%	131 ms	109 ms
0%	73 ms	56 ms

While this aspect has received little attention in past research, we notice that in practice, very intricate computation graph with multiple branches are less efficient. In our method, the overhead caused by dynamism is not as heavy as with VGG-SDN, making it more suitable for production use cases.

5. Conclusion

In this paper we introduced a new dynamic resolution method for object detection, a comprehensive evaluation metric tailored to dynamic neural networks and an evaluation of a deployment pipeline based on ONNX. Our DualRes method achieves state of the art trade-off between accuracy and compute and our AUC-Policy metric highlights the role of the dynamic routing in this result. On the deployment side, our pipeline bridges the gap between research and development and shows the applicability of dynamic neural networks in real-life deployment scenarios.

6. Acknowledgments

This work is funded by EU's Horizon Europe research and innovation program under grant agreement No. 101070374.

References

[1] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European conference on computer vision (ECCV)*, 2018. 7

[2] Chris J. Maddison and Andriy Mnih and Yee Whye Teh. The concrete distribution: a continuous relaxation of discrete random variables. In *arXiv preprint arXiv:1611.00712*, 2017. 3

[3] Chuyi Li and Lulu Li and Hongliang Jiang and Kaiheng Weng and Yifei Geng and Liang Li and Zaidan Ke and Qingyuan Li and Meng Cheng and Weiqiang Nie and Yiduo Li and Bo Zhang and Yufei Liang and Linyuan Zhou and Xiaoming Xu and Xiangxiang Chu and Xiaoming Wei and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications. In *arXiv preprint arXiv:2209.02976*, 2022. 4

[4] Pytorch contributors. torch.cond. In *Documentation article*, 2022. 5

[5] Pytorch contributors. torch.onnx. In *Documentation article*, 2022. 5

[6] Torch contributors. Faster r-cnn. In *Documentation article*, 2017. 3

[7] Daeun Seo and Hoeseok Yang and Sihyeong Park and Hyungshin Kim. Elastic-detr: Making image resolution learnable with content-specific network prediction. In *arXiv preprint arXiv:2412.06341*, 2024. 3, 8

[8] Eric Jang and Shixiang Gu and Ben Poole. Categorical reparametrization with gumbel-softmax. In *Proceedings of 5th International Conference on Learning Representations*, 2017. 3

[9] Gao Huang and Danlu Chen and Tianhong Li and Felix Wu and Laurens van der Maaten and Kilian Weinberger. Multi-scale dense networks for ressource efficient image classification. In *International Conference on Learning Representations*, 2018. 2, 4, 7

[10] Gao Huang and Yulin Wang and Kangchen Lv and Haojun Jiang and Wenhui Huang and Pengfei Qi and Shiji Song. Glance and focus networks for dynamic visual recognition. In *IEEE transactions on pattern analysis and machine intelligence*, 2022. 7

[11] Hang Su and Xiatian Zhu and Shaogang Gong. Open logo detection challenge. In *BMVC*, 2018. 2

[12] Hao Li and Hong Zhang and Xiaojuan Qi and Ruigang Yang and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2019. 7

[13] Icaro O. de Oliveira, Rayson Laroca, David Menotti, Keiko V. O. Fonseca and Rodrigo Minetto. Vehicle-rear: A new dataset to explore feature fusion for vehicle identification using convolutional neural networks. In *arXiv preprint https://arxiv.org/abs/1911.05541*, 2021. 7

[14] Iris A. M. Huijben and Wouter Kool and Max B. Paulus and Ruud J. G. van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. In *IEEE transactions on pattern analysis and machine intelligence*, 2022. 3

[15] Jade Nie and CK Luk and Xiaodong Wang and Jackie (Jiaqi) Xu. Optimizing production pytorch models' performance with graph transformations. In *Blog post*, 2022. 5

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

[17] Luca M. Schulze-Buschoff Robert Geirhos Kristof Meding and Felix A. Wichmann. Imagenet suffers from dichotomous data difficulty. In *NeurIPS*, 2021. 1

[18] Le Yang and Yizeng Han and Xi Chen and Shiji Song and Jifeng Dai and Gao Huang. Resolution adaptive networks for efficient inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 7

[19] Le Yang, Ziwei Zheng, Jian Wang, Shiji Song, Gao Huang, and Fan Li. Adadet: An adaptive object detection system based on early-exit neural networks. In *IEEE Transactions on Cognitive and Developmental Systems*, 2024. 2

[20] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022. 3

[21] Luca Bompani and Manuele Rusci and Daniele Palossi and Francesco Conti and Luca Benini. Multi-resolution rescored bytetrack for video object detection on ultra-low-power embedded systems. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition, 2024. 3

[22] Mengzhao Chen and Mingbao Lin and Ke Li and Yunhang Shen and Yongjian Wu and Fei Chao and Rongrong Ji. Cf-vit: A general coarse-to-fine method for vision transformer. In *Proceedings of the AAAI conference on artificial intelligence*, 2023. 7

[23] Mingjian Zhu and Kai Han and Enhua Wu and Qiulin Zhang and Ying Nie and Zhenzhong Lan and Yunhe Wang. Dynamic resolution network. In *NeurIPS*, 2021. 2

[24] Mingxing Tan and Ruoming Pang and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 2

[25] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov and Sergey Zagoruyko . End-to-end object detection with transformers. In *European conference on computer vision*, 2020. 3

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

[27] Augmented Startups. Vehicle registration plates dataset. In *Roboflow Universe*, 2022. 2

[28] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[29] Thomas Verelst and Tinne Tuytelaars. Spatially adaptive neural networks for computer vision. In *arXiv preprint*, 2023. 3

[30] Ting-Wu Chin and Ruizhou Ding and Diana Marculescu. Adascale: Towards real-time video object detection using adaptive scaling. In *Proceedings of machine learning and systems*, 2019. 3

[31] Tsung-Yi Lin and Michael Maire and Serge Belongie and Lubomir Bourdev and Ross Girshick and James Hays and Pietro Perona and Deva Ramanan and C. Lawrence Zitnick and Piotr Dollar. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, Proceedings*, 2014. 2

[32] Xin Wang and Yujia Luo and Daniel Crankshaw and Alexey Tumanov and Fisher Yu and Joseph E. Gonzalez. Idk cascades: Fast deep learning by learning not to overthink. In *arXiv preprint arXiv:1706.00885*, 2018. 4

[33] Yalong Xu and Lin Zhao and Chen Gong and Guangyu Li and Di Wang and Nannan Wang. Dynpose: Largely improving the efficiency of human pose estimation by a simple dynamic framework. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025. 3

[34] Yigitcan Kaya and Sanghyun Hong and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, 2019. 2, 7

[35] Yizeng Han and Dongchen Han and Zeyu Liu and Yulin Wang and Xuran Pan and Yifan Pu and Chao Deng and Junlan Feng and Shiji Song and Gao Huang. Dynamic perceiver for efficient visual recognition. In *International Conference on Computer Vision*, 2023. 2

[36] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang and Yulin Wang. Dynamic neural networks: A survey. In *IEEE TPAMI*, 2021. 2, 4

[37] Yulin Wang and Rui Huang and Shiji Song and Zeyi Huang and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In *Advances in neural information processing systems*, 2021. 7

[38] Zhihao Lin and Yongtao Wang and Jinhe Zhang and Xiaojie Chu. Dynamicdet: A unified dynamic architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 2, 4, 8

[39] Zhizhong Zhang and Shujun Li and Chenyang Zhang and Lizhuang Ma and Xin Tan and Yuan Xie. Optimal transport with arbitrary prior for dynamic resolution network. In *International Journal of Computer Vision*, 2025. 7