Efficient Transformer Encoders for Mask2Former-style models

Manyi Yao[2], Abhishek Aich[1], Yumin Suh[1], Amit Roy-Chowdhury[2], Christian Shelton[2], and Manmohan Chandraker[1,3]

[1] NEC Laboratories, America, San Jose CA 95110, USA
[2] University of California, Riverside, CA 92521, USA
[3] University of California, San Diego, CA 92093, USA
Corresponding author: aaich@nec-labs.com

Abstract. Vision transformer based models bring significant improvements for image segmentation tasks. Although these architectures offer powerful capabilities irrespective of specific segmentation tasks, their use of computational resources can be taxing on deployed devices. One way to overcome this challenge is by adapting the computation level to the specific needs of the input image rather than the current one-size-fits-all approach. To this end, we introduce `ECO-M2F` or **E**ffi**C**ient Transf**O**rmer Encoders for **M**ask**2F**ormer-style models. Noting that the encoder module of M2F-style models incur high resource-intensive computations, `ECO-M2F` provides a strategy to self-select the number of hidden layers in the encoder, conditioned on the input image. To enable this self-selection ability for providing a balance between performance and computational efficiency, we present a three step recipe. The *first* step is to train the parent architecture to enable early exiting from the encoder. The *second* step is to create an derived dataset of the ideal number of encoder layers required for each training example. The *third* step is to use the aforementioned derived dataset to train a gating network that predicts the number of encoder layers to be used, conditioned on input image. Additionally, to change the computational-accuracy trade-off, only steps two and three need to be repeated which significantly reduces retraining time. Experiments on the public datasets show that the proposed approach reduces expected encoder computational cost while maintaining performance, adapts to various user compute resources, is flexible in architecture configurations, and can be extended beyond the segmentation task to object detection.

1 Introduction

With the advent of powerful *universal* image segmentation architectures [5,6,11,16], it is highly desirable to prioritize the computational efficiency of these architectures for their enhanced scalability, *e.g.*, use on resource-limited edge devices. These architectures are extremely useful in tackling instance [12], semantic [29], and panoptic [19] segmentation tasks using one generalized architecture, owing to the transformer-based [31] modules. These universal architectures leverage

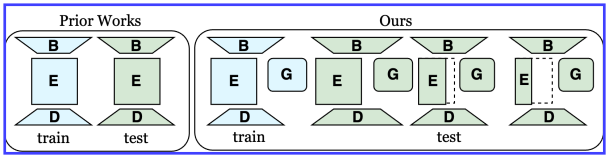

Fig. 1: Comparison to prior works. Instead of conventional M2F-style architecture that provides "one-size-fits-all" solution, our method ECO-M2F focuses on training such models in order to directly run at various resource encoder depths by leveraging a gating function. Here, **B**, **E**, **D**, and **G** denote the backbone, encoder, decoder, and (our proposed) gating network, respectively.

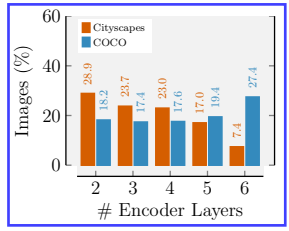

Fig. 2: Histogram of images achieving best panoptic segmentation by number of encoder layers.

DEtection TRansformers or DETR-style [2] modules and represent both *stuff* and *things* categories [19] using general feature tokens. This is an incredible advantage over preceding segmentation methods [15, 26, 40] in literature that require careful considerations in design specifications. Hence, these segmentation architectures reduce the need for task-specific choices which favor performance of one task over the other [5].

State-of-the-art models for universal segmentation like Mask2former (M2F) [5] are built on the key idea inspired from DETR: "mask" classification is versatile enough to address both semantic- and instance-level segmentation tasks. However, the problem of efficient M2F-style architectures have been under-explored. With backbone architectures (*e.g.*, Resnet-50 [13], SWIN-Tiny [22]), [20] showed that DETR-style models incur the highest computations from the transformer encoder due to maintaining full length token representations from multi-scale backbone features. While existing works like [20, 24] primarily focus on scaling the input token to improve efficiency, this approach often neglects other aspects of model optimization and leads to a "one-size-fits-all" solution (Figure 1). This limitation leaves significant room for further efficiency improvements.

Given this growing importance of M2F-style architectures and indispensable need for efficiency for real-world deployment, we introduce ECO-M2F or '**Effi**C**ient** Transf**O**rmer Encoders' for M2F-style architectures. Our key idea comes from our observation made on the training set of COCO [21] and Cityscapes [7] dataset demonstrated in Fig. 2. We plot a histogram of the number of hidden encoder layers that produces the best panoptic segmentation quality [19] for each image. It can be seen that not all images require the use of all K hidden layers of the transformer encoder in order to achieve the maximum panoptic segmentation quality [19]. With this insight, we propose to create a dynamic transformer encoder that economically uses the hidden layers, guided by a gating network that can select different depths for different images.

To achieve the aforementioned ability, ECO-M2F leverages the well-studied early exiting strategy [17, 23, 27, 28, 30, 33, 37, 41, 44] to create stochastic depths for the transformer encoder to improve inference efficiency. Previous exit mechanisms have primarily relied on confidence scores or uncertainty scores, typically

applied in classification tasks. However, implementing such mechanisms in our context would necessitate the inclusion of a decoder and a prediction head to generate a reliable confidence score. This additional complexity introduces a significant number of FLOPs, rendering it impractical for our purposes. By contrast, ECO-M2F provides a three-step training recipe that can be used to customize the transformer encoder on the fly given the input image. Step ***A*** involves training the parent model to *be dynamic* by allowing stochastic depths at the transformer encoder. Using the fact that the transformer encoder maintains the token length of the input throughout the hidden layers constant, Step ***B*** involves creating a *Derived* dataset from the training dataset whose each sample contains a pair of image and layer number that provides the highest segmentation quality. Finally, Step ***C*** involves training a *Gating Network* using the derived dataset, whose function is to decide the number of layers to be used given the input image.

The key contributions of ECO-M2F are multifold. *First*, given a trained M2F-style architecture, ECO-M2F fine-tunes the model into one that allows the ability to randomly exit from the encoder by leveraging the fact that the token length remains constant in the hidden layers. *Second*, it introduces an accessory to the parent architecture *via* the Gating network that provides the ability to smartly use the encoder layers. Using this module, ECO-M2F enables the parent architecture to decide the optimal amount of layers for the given input without any performance degradation as well as any confidence threshold (unlike prior early exiting strategies). *Third*, as a result of our Gating network module's training strategy, ECO-M2F can adapt the parent architecture to varying computational budgets using *only* Step ***C***. On COCO [21] dataset, the computational cost of Step ***B*** and ***C*** are only ∼6% and ∼2.5% of Step ***A*** cost, respectively. *Finally*, ECO-M2F can also incorporate recent advances [20] in making transformer encoder efficient using token length scaling, bringing best of the both methods in pushing the limits of the efficiency. To summarize, we make the following contributions:

- We present a dynamic transformer encoder ECO-M2F for M2F-style universal segmentation that maintains performance but reduces the computational load.
- ECO-M2F consists of a novel training recipe that leverages input image based early exiting (in Step ***A***), creating a derived dataset (based on training set segmentation performance in Step ***B***), which in turn is used to train a gating function (using Step ***C***) that allows adapting the number of hidden layers to reduce computations.
- Extensive experiments show that ECO-M2F improves the overall performance-efficiency trade-off, and adaptable to diverse architecture settings and can be extended beyond segmentation to the detection task.

2 Related Works

Efficient image segmentation. With the rise of transformers [31], researchers are increasingly interested in creating image segmentation models that work effectively in various settings, without requiring segmentation type specific modifications to the model itself. Building on DETR [2], multiple universal segmentation architectures were proposed [5, 6, 11, 16] that use transformer decoder to

predict masks for each entity in the input image. However, despite the significant progress in overall performance across various tasks, these models still face challenges in deployment on resource-constrained devices. Current emphasis [1,4,10,14,15,38,42,43] for efficiency for image segmentation has mostly been on specialized architectures tailored to a single segmentation task. Unlike these preceding works, ECO-M2F makes no such assumption on the segmentation task and addresses the limitation of inefficiency in M2F-style universal architectures that are task-agnostic.

Early-exiting in vision transformers. Recent works on early exiting [17, 23, 27, 28, 30, 32, 33, 37, 39, 41, 44] aim to boost inference efficiency for large transformers. Some works [23, 28, 37] used early exiting for classification tasks along with manually chosen confidence threshold in vision transformers. For example, [37] proposed an early exiting framework for classification task ViTs combining heterogeneous task heads. Similarly, [28] proposed an early exiting strategy for vision-language models by measuring layer-wise similarities by checking multiple times to exit early. Applying early exiting solely to the encoder (like [37]) is infeasible due to the dependency on separate decoders, leading to an unacceptable optimization load. In contrast, methods like [28] suffer from redundant computations for exit decisions at all possible choices, hindering efficient resource allocation. In contrast, ECO-M2F only trains one decoder for all possible exit routes, as well as uses a gating module to decide the number of encoder layers required for the model depending on the input image.

3 Proposed Methodology: ECO-M2F

Model preliminaries. We first review the meta-architecture of M2F [5] upon which ECO-M2F is based, along with the notation. This class of models contains

- a *backbone* $b(\cdot)$ which takes the i-th image $\boldsymbol{x}^{(i)}$ as input to generate multi-scale feature maps $b(\boldsymbol{x}^{(i)})$, represented as $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3, \boldsymbol{s}_4$. These multi-scale feature maps correspond to spatial resolutions typically set at $1/32$, $1/16$, $1/8$, and $1/4$ of the original image size, respectively.
- a *transformer encoder* (called the "pixel decoder" [5]), which is composed of multiple layers of transformer encoders. The function of this module is to generate rich token representation from $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3\}$ and generate per-pixel embeddings from $\boldsymbol{s}_4$. Each layer in the transformer encoder, denoted as $f_k(\cdot)$ (where $k \in \{1, 2, \ldots, K\}$) is successively applied to $b(\boldsymbol{x}^{(i)})$, with $f_K(\cdot)$ being the last layer in the transformer encoder.
- a *transformer decoder* (along with a segmentation head) that takes two inputs: the output of the transformer encoder and the object queries. The object queries are decoded to output a binary mask along with the corresponding class label.

For brevity, we collectively refer to the operations in the transformer decoder and segmentation head together as $h(\cdot)$. Thus, the output of the meta-architecture

with K encoder layers (a predicted mask $\hat{\boldsymbol{y}}_K^{(i)}$ and corresponding label $\hat{\ell}_K^{(i)}$) can be written as

$$\{\hat{\boldsymbol{y}}_K^{(i)}, \hat{\ell}_K^{(i)}\} = h \circ f_K \circ \cdots \circ f_2 \circ f_1 \circ b(\boldsymbol{x}^{(i)})\,. \tag{1}$$

Here, the operation $\circ$ represents function composition, *e.g.*, $g \circ f(x) = g(f(x))$ and subscript denotes output predicted using K encoder layers. With $\{\boldsymbol{y}^{(i)}, \ell^{(i)}\}$ as the pair of ground truth segmentation map and corresponding label of image $\boldsymbol{x}^{(i)}$, the final loss [5] is computed as

$$\mathcal{L}_K = \lambda_{\text{mask}}\mathcal{L}_{\text{mask}}(\hat{\boldsymbol{y}}_K^{(i)}, \boldsymbol{y}^{(i)}) + \lambda_{\text{class}}\mathcal{L}_{\text{class}}(\hat{\ell}_K^{(i)}, \ell^{(i)})\,, \tag{2}$$

where $\mathcal{L}_{\text{mask}}(\cdot,\cdot)$ is a binary mask loss and $\mathcal{L}_{\text{class}}(\cdot,\cdot)$ is the corresponding classification loss. λ_{mask} and λ_{class} represent the associated loss weights.

Method motivation. Our motivation stems from the observation that layers within the transformer encoder of M2F exhibit non-uniform contributions to Panoptic Quality (PQ) [19], as discussed in Sec. 1. This prompts us to question the necessity of all $K = 6$ layers for every image and target minimizing layer usage according to the user's computational constraints while ensuring that overall performance remains within acceptable bounds. Hence, we adopt an adaptive early exiting approach driven by three critical components:

1. *Model suitability for early exiting.* Traditional early exiting techniques [17, 23, 27, 28, 30, 33, 37, 41, 44] often face challenges in maintaining satisfactory performance levels at potential exit points throughout the neural network. We recognize the importance of a model architecture that not only allows for early exiting but also ensures that the performance remains consistently high. Therefore, we aim to develop a model that not only permits early exits but also for which the accuracy steadily improves as the network delves deeper into its architecture. By prioritizing this aspect, we seek to establish a framework where early exiting does not compromise the overall performance of the model.
2. *Efficient and effective gating network for optimal exit decision making.* The efficacy of an early exiting strategy heavily depends on the ability to make informed exit decisions. A gating network must strike a delicate balance, minimizing computational overhead while effectively identifying components that can be bypassed without compromising accuracy. Our objective is to design a lightweight yet powerful gating mechanism capable of discerning optimal exit points within the model architecture.
3. *Dynamic control mechanism for cost-performance trade-off.* We require a mechanism with the ability to adaptively regulate the balance between computational cost and performance according to user-defined priorities. Such a mechanism empowers the model to exit at the optimal layer based on specific needs and desired outcomes, ensuring efficient resource allocation and maximizing utility in various application scenarios, particularly in resource-constrained environments like edge computing or real-time applications.

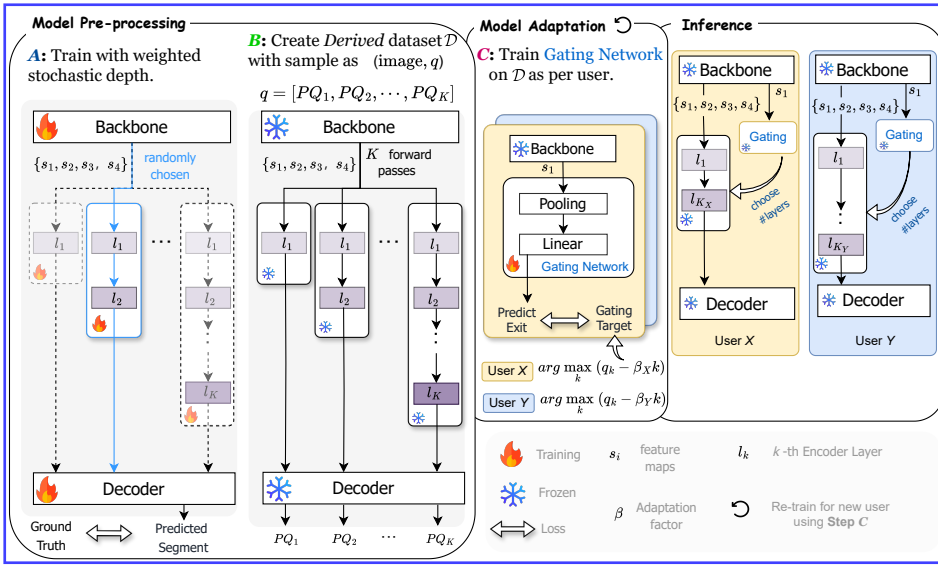

Fig. 3: ECO-M2F framework. During the *model pre-processing* phase, we train the model to exit stochastically at K potential exits using Step ***A***. This is followed by Step ***B***, where we use this model to perform inference on the training images at each exit to create a dataset $\mathcal{D}$. In the *model adaptation* phase, we perform Step ***C*** to establish a gating target based on the computational budget and train a lightweight gating network. During *inference*, the network adheres to the gating network's output and exits at its designated output layer.

Driven by these considerations, ECO-M2F offers a novel training process that enables an adaptive early exiting mechanism designed to bolster computational efficiency while preserving satisfactory model accuracy. For better understanding, we'll begin with a general overview of model training and inference before diving into the specific details of our training process.

Training and Inference overview. As shown in Fig. 3, the training phase of ECO-M2F involves following three main steps:

- Step ***A***: Train parent model for early exit via the transformer encoder.
- Step ***B***: Derive a dataset (which we call the Derived dataset) from the dynamic model obtained in Step ***A***.
- Step ***C***: Train the *Gating Network* to learn optimal exit points in the encoder tailored to users' requirements.

We refer to Step ***A*** and ***B*** together as *model pre-processing* and Step ***C*** as *model adaptation*. The former is required only once, whereas the latter is repeated as per user requirements. All these steps use the training data subset.

During inference, the gating network guides the parent model by selecting the optimal exit point based on features extracted from the backbone with just one forward pass for final predictions.

3.1 Step *A*: Training the Model with Weighted Stochastic Depth

In this step, we enable the model to allow exiting at the encoder. To maintain consistently high performance at each exit point, we input each stochastic depth's output to a shared transformer decoder. We then apply Eq. 2 to compute the loss $\mathcal{L}_k$ for each exit point k. However, we observe that direct training in this fashion does not encourage the model to use fewer layers to extract and prioritize informative representations, as shown in Tab. 5. To address this, we introduce a set of coefficients α_k to emphasize the quality of representations at later layers more, enabling earlier layers to also concentrate on producing effective intermediate representations. As the layer depth increases, the corresponding coefficient α_k grows, ensuring a progressively stricter standard for feature quality. The new loss function is then expressed as

$$\mathcal{L}_{\text{total}} = \frac{1}{N} \sum_{i}^{N} \sum_{k}^{K} \alpha_k \mathcal{L}_k, \text{ where } \forall k < k', \alpha_k < \alpha_{k'}, \tag{3}$$

where N is the number of images in the training set, and $\mathcal{L}_k$ is from Eq. 2.

3.2 Step *B*: Deriving the Gating Network Training Dataset

To facilitate informed exit decisions during inference, our approach is to train a gating network to learn optimal exit strategies. In this step, we facilitate this gating network training by first deriving an intermediate dataset.

To this end, we record the performance of the pre-trained stochastic depth model (obtained from Step ***A***) at all potential exit points for each image within the training dataset and create a *Derived* dataset $\mathcal{D}$. Specifically, we associate the i-th input image $\boldsymbol{x}^{(i)}$ with a vector $\boldsymbol{q}^{(i)}$ of length K. Each element $q_k^{(i)}$ of $\boldsymbol{q}^{(i)}$ represents the predicted panoptic quality [19] upon exiting at the encoder layer k. Hence, each sample of $\mathcal{D}$ can be represented as $(\boldsymbol{x}^{(i)}, \boldsymbol{q}^{(i)}) \in \mathcal{D}$.

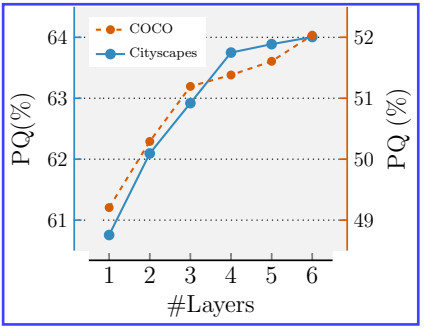

Fig. 4: Intuition for Eq. 4. This figure shows that prioritizing PQ would need more encoder layers, and conversely, prioritizing lesser layers would result in poorer PQ. (Backbone: SWIN-T; training set).

3.3 Step *C*: Training for Gating Network

In this step, we train the gating network on dataset $\mathcal{D}$ (obtained from Step ***B***) to self-select the number of encoder layers based on the input image. Ideally, this module should allow exiting at the encoder layer which would result in the highest quality segmentation map. With this in mind, we first establish the

target exit for the gating network. Note that the panoptic quality generally increases with increasing encoder layers (see Fig. 4). However, we would like the gating network to prioritize increasing the panoptic quality while also reducing the number of layers (to reduce the overall computations). Consequently, we introduce a utility function expressed as the linear combination of segmentation quality and the depth of the network. This function is formulated as

$$u(k) = q_k^{(i)} - \beta k \,, \tag{4}$$

where β serves as an *adaptation factor* governing the trade-off between segmentation quality and computational cost. Clearly, a higher value of β signifies a greater emphasis on efficiency over segmentation quality. Using Eq. 4, we determine a target exit point $t^{(i)}$ for each image $\boldsymbol{x}^{(i)}$ using

$$t^{(i)} = \arg\max_k (u(k)) \,. \tag{5}$$

With a target designated for each image using Eq. 5, the gating decision can be approached as a straightforward classification problem. The gating architecture consists of a pooling operation $\boldsymbol{z}(\cdot)$ on the token length dimension followed by a linear layer with weights $\boldsymbol{W}$. Its output logits can be represented as

$$\boldsymbol{g}^{(i)} = \boldsymbol{W}\boldsymbol{z}(\boldsymbol{s}_1^{(i)}) \,. \tag{6}$$

In consideration of having minimal impact on the computations due to the gating network, we use the output of the lowest resolution feature map $\boldsymbol{s}_1$ as input to the pooling operation. To optimize the gating network, we use the standard cross-entropy loss between the output logits $\boldsymbol{g}^{(i)}$ and the one-hot version of target exit $t^{(i)}$ as our training objective. During inference, the gating network identifies the layer with the highest predicted logits, *i.e.*, $\arg\max_k(g_k^{(i)})$, as the optimal exit layer for image $\boldsymbol{x}^{(i)}$. Note that while there can be more complex choices for the gating network, our simple linear layer in Eq. 6 works well in our experiments.

Saving training costs through Step *C*. `ECO-M2F` presents a distinct advantage in terms of its adaptability to varying computational constraints. In scenarios where a smaller model is desired, `ECO-M2F` necessitates training solely the gating network (*i.e.*, repeat Step *C*). Assuming that the computational load is proportional to the depth of the network, Eq. 4 enables us to weigh the performance gain against the computational overhead for each exit layer. We achieve this by setting the total number of layers K to a smaller number depending on user preferences. For instance, as illustrated in Fig. 3, User X preferring a smaller model compared to User Y may opt for a smaller K, *i.e.*, $K_X < K_Y$. Then, given the importance of segmentation quality, we choose β. With these two variables set in Eq. 5, we train the gating network. This capability shows that `ECO-M2F` is versatile and resource-efficient as it adapts to diverse needs and optimizes allocations.

3.4 Inference

In the inference phase, the gating network guides the parent model toward an optimal exit point tailored to each input image. Similar to the training phase, the gating mechanism receives low-resolution features from the backbone and produces a vector of length K for each image. The value of K remains consistent with that determined in Step C. Subsequently, the gating network identifies the layer with the highest predicted logits as the optimal exit layer for each image. The parent model adheres to this decision, exiting at the determined layer, and subsequently progresses through the subsequent components to make the final prediction. This dynamic process ensures that the model adaptively selects the most optimal layer for exit during inference, enhancing its efficiency in handling diverse input data.

4 Experiments

Datasets. Our study illustrates the adaptability of `ECO-M2F` in dynamically managing the trade-off between computation and performance based on M2F [5] meta-architecture. We do this on two widely used image segmentation datasets: COCO [21] and Cityscapes [7]. COCO comprises 80 "things" and 53 "stuff" categories, with 118k training images and 5k validation images. Cityscapes consists of 8 "things" and 11 "stuff" categories, with approximately 3k training images and 500 validation images. The evaluation is conducted over the union of "things" and "stuff" categories.

Evaluation metrics. We follow the evaluation setting of [5] for evaluation of "universal" segmentation, *i.e.*, we train the model solely with panoptic segmentation annotations but evaluate it for panoptic, semantic, and instance segmentation tasks. We use the standard **PQ** (Panoptic Quality [19]) metric to evaluate panoptic segmentation performance. We report $\mathbf{AP}_p$ (Average Precision [21]) computed across all categories for instance segmentation, and $\mathbf{mIOU}_p$(mean Intersection over Union [9]) for semantic segmentation by merging instance masks from the same category. The subscript p denotes that these metrics are computed for the model trained solely with panoptic segmentation annotations. In terms of computational cost, we use GFLOPs calculated as the average GFLOPs across all validation images. All models are trained on the *train* split and evaluated on the *validation* split.

Baseline models. We compare `ECO-M2F` with two sets of efficient segmentation methods. *First*, we compare with our baseline universal segmentation architecture M2F [5]. Further, we also integrate recently proposed transformer encoder designs (Lite-DETR [20] and RT-DETR [24]) for efficient object detection into M2F and named them Lite-M2F and RT-M2F, respectively. *Second*, we include comparisons with recent efficient architectures that proposed task-specific components, namely YOSO [15], RAP-SAM [40], and ReMax [26].

Table 1: COCO evaluation. Our method, Task-specific architectures

Model	Performance (↑) PQ	mIoU$_p$	AP$_p$	GFLOPs (↓) Total	Tx.	Enc.
Backbone: SWIN-T						
RT-M2F [24]	41.36	61.54	24.68	158.30	59.66	
Lite-M2F [20]	52.70	63.08	41.10	188.00	79.78	
M2F [5]	52.03	62.49	42.18	235.57	121.69	
ECO-M2F($\beta = 0.0005$)	52.06	62.76	41.51	202.39	88.47	
ECO-M2F($\beta = 0.02$)	50.79	62.25	39.71	181.64	67.71	
Lite-ECO-M2F	52.84	63.23	42.18	178.43	64.42	
Backbone: Res50						
M2F [5]	51.73	61.94	41.72	229.10	135.00	
MF [6]	46.50	57.80	33.00	181.00	–	
YOSO [15]	48.40	58.74	36.87	114.50	–	
RAP-SAM [40]	46.90	–	–	123.00	–	
ReMax [26]	53.50	–	–	–	–	
ECO-M2F	51.89	61.07	41.25	195.55	92.37	

Table 2: Cityscapes evaluation. Our method, Task-specific architectures

Model	Performance (↑) PQ	mIoU$_p$	AP$_p$	GFLOPs (↓) Total	Tx.	Enc.
Backbone: SWIN-T						
RT-M2F [24]	59.73	77.89	31.35	361.10	130.00	
Lite-M2F [20]	62.29	79.43	36.57	428.71	172.00	
M2F [5]	64.00	80.77	39.26	537.85	281.13	
ECO-M2F($\beta = 0.003$)	64.18	80.49	39.64	507.51	250.80	
ECO-M2F($\beta = 0.01$)	62.09	79.58	36.04	439.67	182.95	
Lite-ECO-M2F	62.64	79.99	36.52	412.88	156.17	
Backbone: Res50						
M2F	61.86	76.94	37.35	524.11	281.13	
YOSO [15]	59.70	76.05	33.76	265.1	–	
ReMax [26]	65.40	–	–	294.7	–	
ECO-M2F	62.20	77.34	37.21	453.50	220.59	

Architecture details. We focus on standard backbones Res50 [13] and SWIN-Tiny [22] pre-trained on ImageNet-1K [8], unless specified otherwise. We set the total number of encoder layers to be 6 following [5]. We consider layers 2 to 6 as potential exit points, unless stated otherwise. In our gating network, we use a straightforward 1D adaptive average pooling operation as our pooling function.

Training settings. The experimental setup closely mirrors that of M2F [5], with all model configurations and training specifics following identical procedures. We use Detectron2 [34] and PyTorch [25] for our implementation. For the stochastic depth training phase (Step ***A***), we initialize weights as provided by M2F and subsequently train 50 epochs for the COCO dataset and 90k iterations for Cityscapes, with a batch size of 16. For the training of the gating network (Step ***C***), we perform 2 epochs of training on the COCO dataset and 20k iterations on the Cityscapes dataset, employing the Adam optimizer [18]. The adaptation factor β in the utility function, as discussed in Sec. 3.3, is set to 0.0005 for COCO and 0.003 for Cityscapes, unless otherwise specified. Distributed training is performed using 8 A6000 GPUs. On the COCO dataset, the training time of Step ***A*** is 280 GPU hours, Step ***B*** is 17 GPU hours, and Step ***C*** 7.2 GPU hours. Similarly for Cityscapes dataset, the training time of Step ***A*** is 45 GPU hours, Step ***B*** is 1 GPU hours, and Step ***C*** is 7.2 GPU hours. In Step ***A***, we use identical settings as M2F for the loss between the predicted segment and ground truth segment, *i.e.*, $\mathcal{L}_k$. The weight λ_{mask} is fixed at 5.0, while λ_{class} is set to 2.0 for all classes, except 0.1 for the "no object" class.

4.1 Main Results

In Table 1 and Table 2, we compare ECO-M2F with our baseline prior works on the validation set of COCO and Cityscapes dataset, respectively. In Table 1, we observe that ECO-M2F effectively reduces computational costs while upholding performance levels in comparison to M2F [5] using both SWIN-T [22] and

Table 3: Impact of β. As the value of β increases, the model places greater emphasis on reducing GFLOPs over performance both in COCO and Cityscapes datasets. Baseline is M2F [5]. (Backbone: SWIN-T)

Data	β	Performance (↑) PQ	mIoU$_p$	AP$_p$	GFLOPs (↓) Total	Tx. Enc.
COCO	Baseline	52.03	62.49	42.18	235.57	121.69
	0.0	52.24	62.95	41.61	220.61	107.18
	0.0005	52.06	62.76	41.51	202.39	88.47
	0.001	51.72	62.60	41.12	193.10	79.18
	0.02	50.79	62.25	39.71	181.64	67.71
Cityscapes	Baseline	64.00	80.77	39.26	537.85	281.13
	0.0	64.58	80.35	40.31	536.09	279.37
	0.003	64.18	80.49	39.64	507.51	250.80
	0.005	63.24	79.73	37.97	469.38	212.66
	0.01	62.09	79.58	36.04	439.67	182.95
	0.1	60.71	78.15	33.86	411.98	155.26

Table 4: Impact of K. The baseline M2F [5] refers to the complete M2F model trained with 6/5/4 layers for 50 epochs. `ECO-M2F` pertains to the training of only the gating network for 2 epochs by setting $K = 6, 5, 4$ respectively. (Dataset: COCO; Backbone: SWIN-T)

K	Model	Performance (↑) PQ	mIOU$_p$	AP$_p$	GFLOPs (↓) Total	Tx. Enc.
6	M2F [5]	52.03	62.49	42.18	235.57	121.69
	ECO-M2F	52.06	62.76	41.51	202.39	88.47
5	M2F [5]	51.61	61.93	41.55	221.95	108.07
	ECO-M2F	52.26	62.67	41.56	208.82	94.59
4	M2F [5]	51.38	62.30	41.11	208.33	94.45
	ECO-M2F	52.20	62.58	41.55	202.47	88.65

Res50 [13] backbones. Additionally, `ECO-M2F` can be seamlessly integrated into efficient encoder designs, such as Lite-M2F [5] [20], further reducing GLOPs by approximately 12.6%. With Res50 as the backbone, MF [6], YOSO [15], and RAP-SAM [40] exhibit inferior performance compared to `ECO-M2F`. Although ReMax [26] demonstrates competitive accuracy, its focus on specialized panoptic segmentation models limits its applicability. Our work, however, aims for a broader impact by creating efficient segmentation architectures that can be used for various segmentation tasks. We make similar observations on the Cityscapes dataset as presented in Table 2.

4.2 Ablation Studies

Balancing computational cost and performance. Within `ECO-M2F`, the parameter β serves as an adaptation factor that governs the trade-off between computational cost and performance. Its value, however, is contingent upon the backbone and dataset characteristics. This dependency arises due to the disparate ranges of GFLOPs and segmentation quality (represented by PQ) which are a function of the architecture components and the training data distribution. Fig. 5 illustrates `ECO-M2F` with different values of β in Step ***C*** using the exact same weights for the parent model from Step ***A*** during inference. In comparison to the M2F$_i$ model (where each is trained standalone with i layers), adjusting the value of β provides a trade-off between GFLOPs and PQ.

– *Impact of adaptation factor β.* We analyze the impact of β on `ECO-M2F` and present our analysis in Table 3. As expected, a smaller β prioritizes segmentation quality over computations resulting in superior performance. Conversely, a larger β signifies a greater emphasis on GFLOPs. This results in a slight sacrifice in PQ leading to a significant reduction in GFLOPs.

Table 5: Stochastic Depth (SD) training. Here, all models (w/ 6 encoder layers) deterministically exit at the marked layer at inference. "USD": Unweighted; "WSD": Weighted. (Baseline: M2F w/ SWIN-T)

Model	PQ (↑) 2	3	4	5	6
Dataset: Cityscapes					
Baseline	03.21	14.37	26.66	42.50	64.00
w/ USD	59.96	61.85	63.18	62.98	63.73
w/ WSD	60.71	62.14	62.94	63.89	64.60
Dataset: COCO					
Baseline	10.16	17.02	23.43	33.63	51.71
w/ USD	49.40	50.25	50.49	50.51	50.44
w/ WSD	50.70	51.76	52.30	52.39	52.48

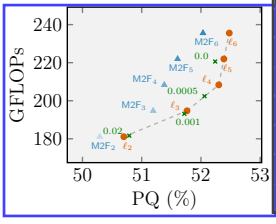

Fig. 5: M2F$_i$ is trained w/ total i layers. ℓ_i is result of same model from our Step ***A***. Values denote β. Dataset: COCO.

Table 6: Impact of backbone size. ECO-M2F shows strong performance w.r.t. baselines across all backbone sizes on both COCO and Cityscapes datasets while reducing GFLOPs in the transformer encoder. †ImageNet-21K pre-trained

Bakbone	Model	Performance (↑) PQ	mIOU$_p$	AP$_p$	GFLOPs (↓) Total	Tx. Enc.
Dataset: COCO						
SWIN-T	M2F [5]	52.03	62.49	42.18	235.57	121.69
	ECO-M2F	52.66	62.76	41.51	202.88	88.45
SWIN-S	M2F [5]	54.63	64.24	44.69	316.50	121.69
	ECO-M2F	54.76	64.46	44.48	275.86	81.85
SWIN-B†	M2F	56.40	67.09	46.29	470.98	122.56
	ECO-M2F	56.48	66.56	46.46	425.47	76.56
Dataset: Cityscapes						
SWIN-T	M2F [5]	64.00	80.77	39.26	537.85	281.13
	ECO-M2F	64.18	80.48	39.84	507.51	250.86
SWIN-S	M2F [5]	64.84	81.76	40.73	724.29	281.13
	ECO-M2F	65.12	81.64	41.17	685.86	242.82
SWIN-B†	M2F [5]	66.12	82.70	42.84	1051.19	283.14
	ECO-M2F	65.14	82.65	40.51	994.78	216.66

– *Impact of total encoder layers K in the parent architecture.* In situations where computational resources are limited, we present an approach using ECO-M2F to create a scaled-down version of the parent model. As illustrated in Tab. 4, we analyze the impact of K set to 5 and 4 (in place of the default value of 6). We set $\beta = 0.0005$ for all the models. It can be observed that ECO-M2F not only reduces the computational load but also preserves the performance of the parent model. Importantly, we *only* execute Step ***C*** for each configuration which involves training the gating network.

Impact of backbone features. We analyze the impact of backbone features on ECO-M2F in Tab. 6 and Tab. 8, since our gating network mechanism takes these features as input.

– *Size variations.* In Table 6, we evaluate ECO-M2F with backbones of different sizes. Specifically, we observe the performance with SWIN-Tiny (T), SWIN-Small (S), and SWIN-Base (B) architectures [22]. We can observe the adaptability of ECO-M2F in delivering robust performance efficiency across a range of these backbone sizes. Furthermore, this versatility of ECO-M2F extends to Lite-M2F meta-architecture as well (see supplementary material).

– *Pre-trained weight variations.* We explore how different pre-training strategies for the backbones affect the performance of ECO-M2F in Table 8. We initialize the backbone weights using both *supervised learning* (SL) and *self-supervised learning* (SSL) techniques on the ImageNet-1K dataset [8]. For SSL pre-training on the SWIN-T [22] backbone, we utilize MoBY [35]. For SSL pre-training on the Res50 [13] backbone, we use DINO [3]. With MoBY weights on the SWIN-T backbone, ECO-M2F maintains 99.7% of the PQ of M2F while

Table 7: Impact on DETR. We extend our approach to DETR [2] for object detection tasks as `ECO-DETR`. (Dataset: COCO; Backbone: Res50)

Model	Performance (↑) AP	AP_{50}	AP_S	AP_M	AP_L	GFLOPs (↓) Total	Tx. Enc.
DETR [2]	42.0	62.4	20.5	45.8	61.1	83.59	9.92
ECO-DETR($\beta = 0.0001$)	41.9	62.2	20.8	45.8	60.4	81.87	6.41
ECO-DETR($\beta = 0.001$)	40.9	61.5	20.2	44.5	59.1	79.92	4.52
ECO-DETR($\beta = 0.01$)	40.2	61.0	18.7	43.6	58.8	79.33	3.94

Table 8: Impact of backbone weights. `ECO-M2F` applies to backbone weights obtained through self-supervised pre-training as well. (Dataset: COCO)

Bakbone	Model	Performance (↑) PQ	$mIOU_p$	AP_p	GFLOPs (↓) Total	Tx. Enc.
SWIN-T(MoBY [36])	M2F [5]	51.64	62.42	41.93	235.57	121.69
	ECO-M2F	51.43	62.43	41.27	230.56	89.04
Res50(DINO [3])	M2F [5]	51.57	61.65	41.24	229.41	126.05
	ECO-M2F	51.80	61.97	41.93	161.07	88.44

reducing GFLOPs in the transformer encoder by 26.34%. With DINO weights on the Res50 backbone, we observe a reduction of 29.79% in GFLOPs in the transformer encoder, alongside a slight improvement in PQ.

Performance in object detection. To demonstrate the applicability of our method beyond segmentation, we extend our proposed three step recipe for object detection task using DETR [2] for object detection task and name it `ECO-DETR`. In particular, we vary the values of the adaptation factor β to analyze performance-computation trade-offs of `ECO-DETR` in Table 7. Clearly, the resultant architecture maintains the performance but helps in reducing the computations of the encoder (*e.g.*, it achieves a 35.38% reduction in GFLOPs in the transformer encoder without significantly impacting performance).

4.3 Qualitative Comparisons

We present a few examples of predicted segmentation maps in Fig. 6 with SWIN-T [22] backbone. Compared to the parent architecture, `ECO-M2F` consistently shows strong performance while self-selecting the encoder layers based on the input examples, both in everyday scenes (on COCO dataset) as well as intricate traffic scenes (on Cityscapes dataset).

5 Conclusions

In this paper, we propose an efficient transformer encoder design `ECO-M2F` for the Mask2Former-style frameworks. `ECO-M2F` provides a three-step training recipe that can be used to customize the transformer encoder on the fly given the input image. The first step involves training the parent model to *be dynamic* by allowing stochastic depths at the transformer encoder. The second step involves creating a derived dataset from the training dataset which contains a pair of image and layer number that provides the highest segmentation quality. Finally, the third step involves training a gating network, whose function is to decide the number of layers to be used given the input image. Extensive experiments demonstrate that `ECO-M2F` achieves significantly reduced computational complexity compared to established methods while maintaining competitive performance in universal segmentation. Our results highlight `ECO-M2F`'s ability to

Fig. 6: Qualitative visualizations. We illustrate few examples of predicted segmentation maps from M2F [5] (*middle* column) and `ECO-M2F` (*last* column). *Top* two rows are from the COCO dataset, whereas *bottom* two rows are from the Cityscapes dataset. (Backbone: Swin-T) *Zoom-in for best view.*

dynamically trade-off between performance and efficiency as per requirements, showcasing its adaptability across diverse architectural configurations, and can be applied to models for object detection tasks.

Limitations. While `ECO-M2F` offers dynamic trade-offs between performance and efficiency according to specific needs, the adaptation factor β is a hyperparameter that needs separate tuning for each use case. This is because it relies on the model configuration and dataset characteristics.

Efficient Transformer Encoders for Mask2Former-style models (Supplementary Material)

A Additional Experiments

Impact of backbone size on Lite-M2F. We apply `ECO-M2F` on Lite-M2F using various backbone sizes, including SWIN-Tiny (T), SWIN-Small (S), and SWIN-Base (B) architectures [22]. Lite-M2F is a specific variant based on Lite-DETR [20]. We used the configuration named "Lite-DETR H3L1-(6+1)×1" given its strong performance in detection relative to the computations required. However, we adjust this configuration to (5+1) when applying our approach to Lite-M2F. Further, we use their without the key-aware deformable attention [20] proposed in their paper. This adjustment is necessary because Lite-M2F actually has 6 encoder layers, and the original configuration might introduce an additional layer that isn't present in the model. Following this, we identify layers 2 to 5 as potential exits, followed by the last layer, layer 6 in the transformer encoder. We retain layer 6 and do not consider it as a feasible exit point as it leverages features from all scales provided by the backbone, making it essential to the model's functionality. As shown in Tab. T1, we observe that `ECO-M2F` effectively reduces computational cost while maintaining performance across Lite-M2F variants, which underscores the versatility and robustness of `ECO-M2F` across different model architectures and sizes.

Impact of target and loss settings for gating network training. We investigate various target and loss settings during the training of the gating network. Specifically, we compare the approach detailed in the main paper, using one-hot target and cross-entropy loss (referred to as "hard-CE" in Tab. T2), with three alternative methods that do not involve setting a specific target exit for each image.

First, we consider using cross-entropy loss between the output of the utility function $u(\cdot)$ and the predicted logit passed through a softmax function (referred to as "u-CE"), *i.e.*,

$$\mathcal{L}_{\text{gating}} = \sum_{i}^{N} \sum_{k}^{K} u^{(i)}(k) \ln[\text{softmax}(g_k^{(i)})] .$$

Second, we apply a softmax function to the utility function $u(k)$ and use cross-entropy as the loss function (referred to as "soft-CE"), *i.e.*,

$$\mathcal{L}_{\text{gating}} = \sum_{i}^{N} \sum_{k}^{K} \text{softmax}(u^{(i)}(k)) \ln[\text{softmax}(g_k^{(i)})] .$$

Third, we apply a softmax function to the utility function, but use mean squared error (MSE) loss instead (referred to as "soft-MSE"), *i.e.*,

$$\mathcal{L}_{\text{gating}} = \sum_{b}^{N} \sum_{k}^{K} \left[\text{softmax}(u^{(i)}(k)) - \text{softmax}(q_k^{(i)}) \right]^2 .$$

The analysis in Tab. T2 is conducted using the SWIN-T [22] backbone on the COCO dataset. We observe that "hard-CE" yields the most favorable results. As a result, we use this approach consistently in the main paper.

Table T1: Impact of backbone size on Lite-M2F. Our Lite-ECO-M2F maintains the performance of Lite-M2F while reducing GFLOPs for different datasets and for different backbones.

Bakbone	Model		Performance (↑) PQ	mIOU$_p$	AP$_p$	GFLOPs (↓) Total	Tx.	Enc.
Dataset: Cityscapes								
SWIN-T	Lite-M2F	[20]	62.29	79.43	36.57	428.71		172.00
	Lite-ECO-M2F		62.34	79.39	36.52	412.89		156.17
SWIN-S	Lite-M2F	[20]	63.54	79.74	39.12	615.15		171.99
	Lite-ECO-M2F		63.62	80.21	37.91	599.32		145.99
SWIN-B	Lite-M2F	[20]	64.48	82.34	39.21	942.05		174.01
	Lite-ECO-M2F		64.56	81.43	39.52	921.15		153.11
Dataset: COCO								
SWIN-T	Lite-M2F	[20]	52.70	63.08	41.10	193.79		79.78
	Lite-ECO-M2F		52.84	63.23	42.18	178.43		64.42
SWIN-S	Lite-M2F	[20]	54.30	64.81	43.94	269.26		74.45
	Lite-ECO-M2F		54.47	64.74	43.55	253.86		59.06

Table T2: Impact of target and loss in gating network training. We use "hard-CE" loss for training our gating network in the main paper. (Backbone: SWIN-T; Dataset: COCO)

Method	Performance (↑) PQ	mIOU$_p$	AP$_p$	GFLOPs (↓) Total	Tx.	Enc.
hard-CE	52.06	62.76	41.51	202.39		88.47
u-CE	52.16	62.58	41.57	207.49		94.06
soft-CE	51.64	62.75	40.88	202.08		87.85
soft-MSE	51.54	62.73	40.91	198.46		84.53

B Additional Qualitative Results

We provide additional examples of predicted segmentation maps in Fig. F1.

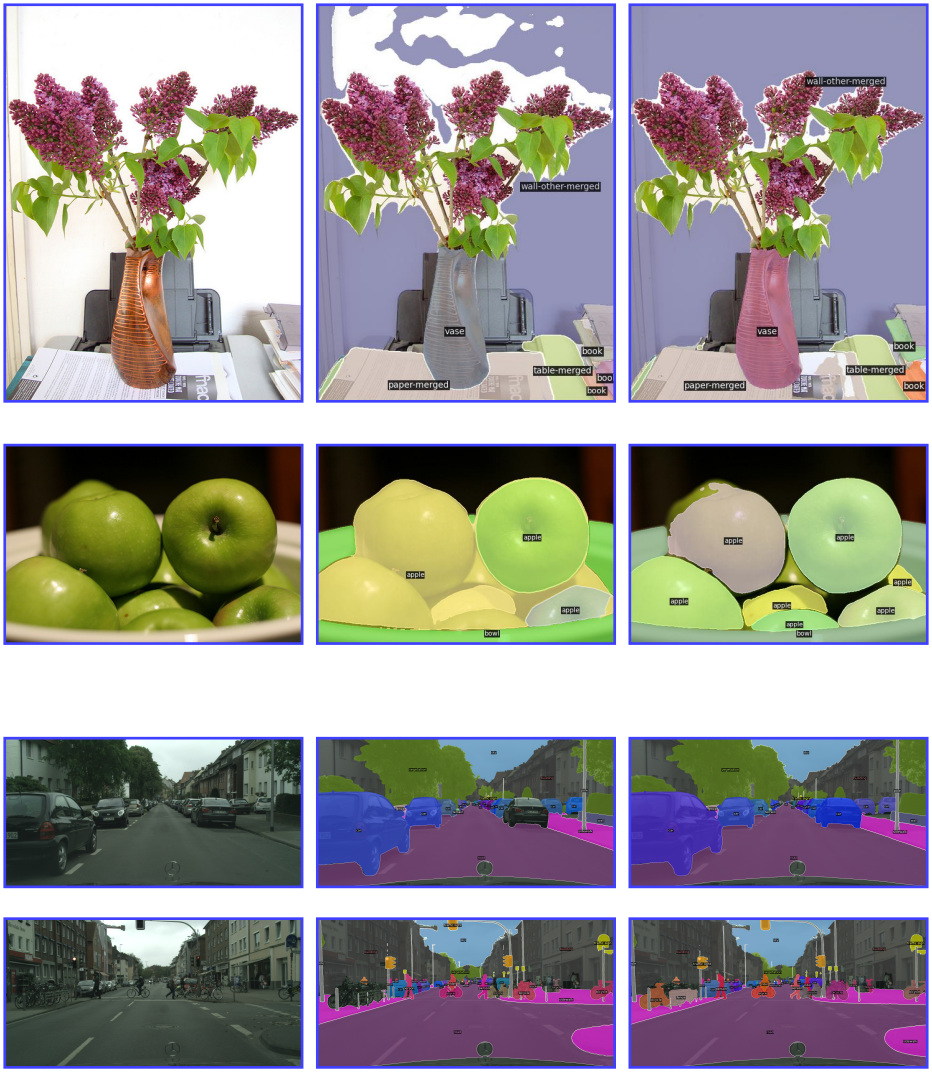

Fig. F1: Qualitative visualizations. We provide additional examples of predicted segmentation maps from M2F [5] (*middle* column) and `ECO-M2F` (*last* column). *Top* two rows are from the COCO dataset, whereas *bottom* two rows are from the Cityscapes dataset. Please zoom in for a clearer view of the details. (Backbone: SWIN-T)

References

1. Ammar, A., Khalil, M.I., Salama, C.: Rt-yoso: Revisiting yoso for real-time panoptic segmentation. In: 2023 5th Novel Intelligent and Leading Emerging

Sciences Conference (NILES). pp. 306–311 (2023). https://doi.org/10.1109/NILES59815.2023.10296714 4
2. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020) 2, 3, 13
3. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the International Conference on Computer Vision (ICCV) (2021) 12, 13
4. Cheng, B., Collins, M.D., Zhu, Y., Liu, T., Huang, T.S., Adam, H., Chen, L.C.: Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12475–12485 (2020) 4
5. Cheng, B., Misra, I., Schwing, A.G., Kirillov, A., Girdhar, R.: Masked-attention mask transformer for universal image segmentation. In: CVPR (2022) 1, 2, 3, 4, 5, 9, 10, 11, 12, 13, 14, 17
6. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. Advances in Neural Information Processing Systems **34**, 17864–17875 (2021) 1, 3, 10, 11
7. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016) 2, 9
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 10, 12
9. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International journal of computer vision **111**, 98–136 (2015) 9
10. Fan, M., Lai, S., Huang, J., Wei, X., Chai, Z., Luo, J., Wei, X.: Rethinking bisenet for real-time semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9716–9725 (2021) 4
11. Gu, X., Cui, Y., Huang, J., Rashwan, A., Yang, X., Zhou, X., Ghiasi, G., Kuo, W., Chen, H., Chen, L.C., et al.: Dataseg: Taming a universal multi-dataset multi-task segmentation model. Advances in Neural Information Processing Systems **36** (2024) 1, 3
12. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 1
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 2, 10, 11, 12
14. Hou, R., Li, J., Bhargava, A., Raventos, A., Guizilini, V., Fang, C., Lynch, J., Gaidon, A.: Real-time panoptic segmentation from dense detections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8523–8532 (2020) 4
15. Hu, J., Huang, L., Ren, T., Zhang, S., Ji, R., Cao, L.: You only segment once: Towards real-time panoptic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17819–17829 (2023) 2, 4, 9, 10, 11
16. Jain, J., Li, J., Chiu, M.T., Hassani, A., Orlov, N., Shi, H.: Oneformer: One transformer to rule universal image segmentation. In: Proceedings of the IEEE/CVF

Conference on Computer Vision and Pattern Recognition. pp. 2989–2998 (2023) 1, 3
17. Jiang, Z., Gong, Z., Xu, Y., Wang, J.: Multi-exit vision transformer with custom fine-tuning for fine-grained image recognition. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 5233–5237 (2023) 2, 4, 5
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017) 10
19. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation (2019) 1, 2, 5, 7, 9
20. Li, F., Zeng, A., Liu, S., Zhang, H., Li, H., Zhang, L., Ni, L.M.: Lite detr: An interleaved multi-scale encoder for efficient detr. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18558–18567 (2023) 2, 3, 9, 10, 11, 15, 16
21. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. pp. 740–755. Springer (2014) 2, 3, 9
22. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021) 2, 10, 12, 13, 15, 16
23. Liu, Z., Sun, Y., Li, Y., Zhou, Z., Hu, J., Li, F.: Multi-exit vision transformer for dynamic inference. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5214–5223 (2021) 2, 4, 5
24. Lv, W., Xu, S., Zhao, Y., Wang, G., Wei, J., Cui, C., Du, Y., Dang, Q., Liu, Y.: Detrs beat yolos on real-time object detection. arXiv preprint arXiv:2304.08069 (2023) 2, 9, 10
25. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019) 10
26. Sun, S., Wang, W., Yu, Q., Howard, A., Torr, P., Chen, L.C.: Remax: Relaxing for better training on efficient panoptic segmentation. arXiv preprint arXiv:2306.17319 (2023) 2, 9, 10, 11
27. Tang, J., Liu, Z., Li, Y., Sun, Y., Zhou, Z., Hu, J., Li, F.: You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13504–13513 (2023) 2, 4, 5
28. Tang, S., Wang, Y., Kong, Z., Zhang, T., Li, Y., Ding, C., Wang, Y., Liang, Y., Xu, D.: You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10781–10791 (2023) 2, 4, 5
29. Tu, Z.: Auto-context and its application to high-level vision tasks. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008) 1
30. Valade, F., Hebiri, M., Gay, P.: Eero: Early exit with reject option for efficient classification with limited budget (2024) 2, 4, 5
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 1, 3

32. Wan, Z., Wang, X., Liu, C., Alam, S., Zheng, Y., Qu, Z., Yan, S., Zhu, Y., Zhang, Q., Chowdhury, M., et al.: Efficient large language models: A survey. arXiv preprint arXiv:2312.03863 **1** (2023) 4
33. Wang, X., Zhou, W., He, X., Peng, X., Wei, F., Guo, Y.: Single-layer vision transformers for more accurate early exits with less overhead. Pattern Recognition **136**, 102243 (2022) 2, 4, 5
34. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 10
35. Xie, Z., Lin, Y., Yao, Z., Zhang, Z., Dai, Q., Cao, Y., Hu, H.: Self-supervised learning with swin transformers. arXiv preprint arXiv:2105.04553 (2021) 12
36. Xie, Z., Lin, Y., Yao, Z., Zhang, Z., Dai, Q., Cao, Y., Hu, H.: Self-supervised learning with swin transformers. arXiv preprint arXiv:2105.04553 (2021) 13
37. Xu, F., Zhang, X., Ma, Z., Wang, J., Hu, J., Sun, J.: Lgvit: Dynamic early exiting for accelerating vision transformer. In: Proceedings of the 32nd ACM International Conference on Multimedia. pp. 1958–1966 (2023) 2, 4, 5
38. Xu, J., Xiong, Z., Bhattacharyya, S.P.: Pidnet: A real-time semantic segmentation network inspired by pid controllers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19529–19539 (2023) 4
39. Xu, M., Yin, W., Cai, D., Yi, R., Xu, D., Wang, Q., Wu, B., Zhao, Y., Yang, C., Wang, S., et al.: A survey of resource-efficient llm and multimodal foundation models. arXiv preprint arXiv:2401.08092 (2024) 4
40. Xu, S., Yuan, H., Shi, Q., Qi, L., Wang, J., Yang, Y., Li, Y., Chen, K., Tong, Y., Ghanem, B., et al.: Rap-sam: Towards real-time all-purpose segment anything. arXiv preprint arXiv:2401.10228 (2024) 2, 9, 10, 11
41. Yang, J., Zhang, X., Zhang, X., Tang, J., Li, X.: Exploiting face recognizability with early exit vision transformers. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 6341–6345 (2023) 2, 4, 5
42. Yu, C., Gao, C., Wang, J., Yu, G., Shen, C., Sang, N.: Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. International Journal of Computer Vision **129**, 3051–3068 (2021) 4
43. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 325–341 (2018) 4
44. Zhang, T., He, X., Qin, Z., Sun, J.: Adaptive deep neural network inference optimization with eenet. In: Proceedings of the 37th International Conference on Machine Learning. pp. 15983–15993 (2023) 2, 4, 5