Image-Specific Adaptation of Transformer Encoders for Compute-Efficient Segmentation

Manyi Yao[†], **Abhishek Aich**[‡], **Yumin Suh**[‡], **Amit Roy-Chowdhury**[†],
Christian Shelton[†], **Manmohan Chandraker**[‡,⋆]
[‡]NEC Laboratories, America, [†]University of California, Riverside,
[⋆]University of California, San Diego

Abstract

Vision transformer-based models bring significant improvements to 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* ***Effi****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 a 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 images. 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 [13], semantic [29], and panoptic [19] segmentation tasks using one generalized architecture, owing to the transformer-based [31] modules. These universal architectures leverage 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, 38] in the literature that require careful considerations in design specifications. Hence, these segmentation architectures reduce the need for task-specific choices that favor the performance of one task over the other [6].

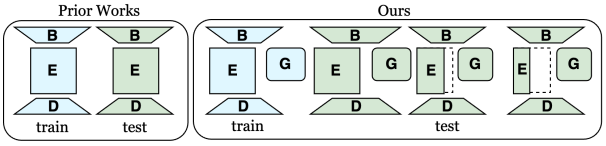
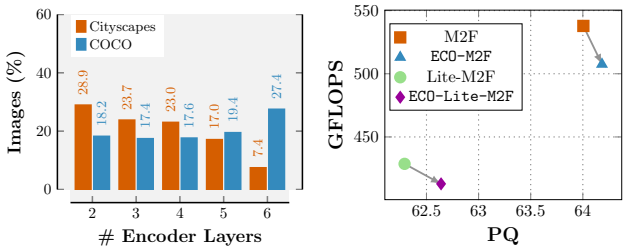


Figure 1. **Comparison to prior works.** Instead of conventional M2F-style architecture that provides a "one-size-fits-all" solution, our method `ECO-M2F` trains models to run directlys at various resource encoder depths by leveraging a gating function. **B**, **E**, **D**, and **G** denote the backbone, encoder, decoder, and (our proposed) gating network, respectively.

State-of-the-art models for universal segmentation like Mask2former (M2F) [6] are built on the key idea inspired by DETR: "mask" classification is versatile enough to address both semantic- and instance-level segmentation tasks. However, the problem of efficient M2F-style architectures has been underexplored. With backbone architectures (*e.g.*, Resnet-50 [12], 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 ne-



(a) (b)

Figure 2. (a) Histogram of images achieving best panoptic segmentation by the number of encoder layers. (b) Our method demonstrates superior performance and lower computational cost compared to the baseline models. (Dataset: Cityscapes; Backbone: SWIN-T)

glects other aspects of model optimization and leads to a "one-size-fits-all" solution (Fig. 1). This limitation leaves significant room for further efficiency improvements.

Given this growing importance of M2F-style architectures and the indispensable need for efficiency for real-world deployment, we introduce ECO-M2F or '**E**ffi**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 (a). 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, 35, 39, 42] 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 images 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 manifold:

- *Enhanced Encoder Flexibility*: It fine-tunes pre-trained M2F-style architectures for early exits from the encoder, leveraging the constant token length in hidden layers.
- *Adaptive Gating Network Training*: It introduces a Gating network to optimize encoder layer usage, allowing the architecture to determine the optimal layer count per input without performance loss or confidence thresholds.
- *Innovative Efficient Training*: The Gating network's training approach enables the architecture to adapt to varying computational budgets, with the cost of only Step **C**(about 2.5% of Step **A**on COCO [21] dataset).
- *Integration and Expansion*: ECO-M2F can incorporate recent advancements in token length scaling for transformer encoder efficiency and extend its application open-vocabulary segmentation and detection tasks.
- *Competitive Performance-Computational Trade-off*: Our method strikes a competitive balance between performance and computational efficiency, and can even excel in both areas simultaneously, as illustrated in Fig. 2 (b).

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 a 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, 36, 40, 41] for efficiency for image segmentation has mostly been on specialized architectures tailored to a single segmentation task. Unlike these preceding works, ECO-M2Fmakes 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, 35, 37, 39, 42] aim to boost inference efficiency for large transformers. Some works [23, 28, 35] used early exiting for classification tasks along with manually chosen confidence threshold in vision transformers. For example, [35] proposed an early exiting framework for classification task ViTs combining het-

erogeneous 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 [35]) 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-M2Fonly 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

3.1. Model Preliminaries

We first review the meta-architecture of M2F [6] 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" [6]), 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 $\tilde{\boldsymbol{y}}_K^{(i)}$ and corresponding label $\tilde{\ell}_K^{(i)}$) can be written as

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

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 [6] is computed as

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

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.

3.2. 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, 35, 39, 42] 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.

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.

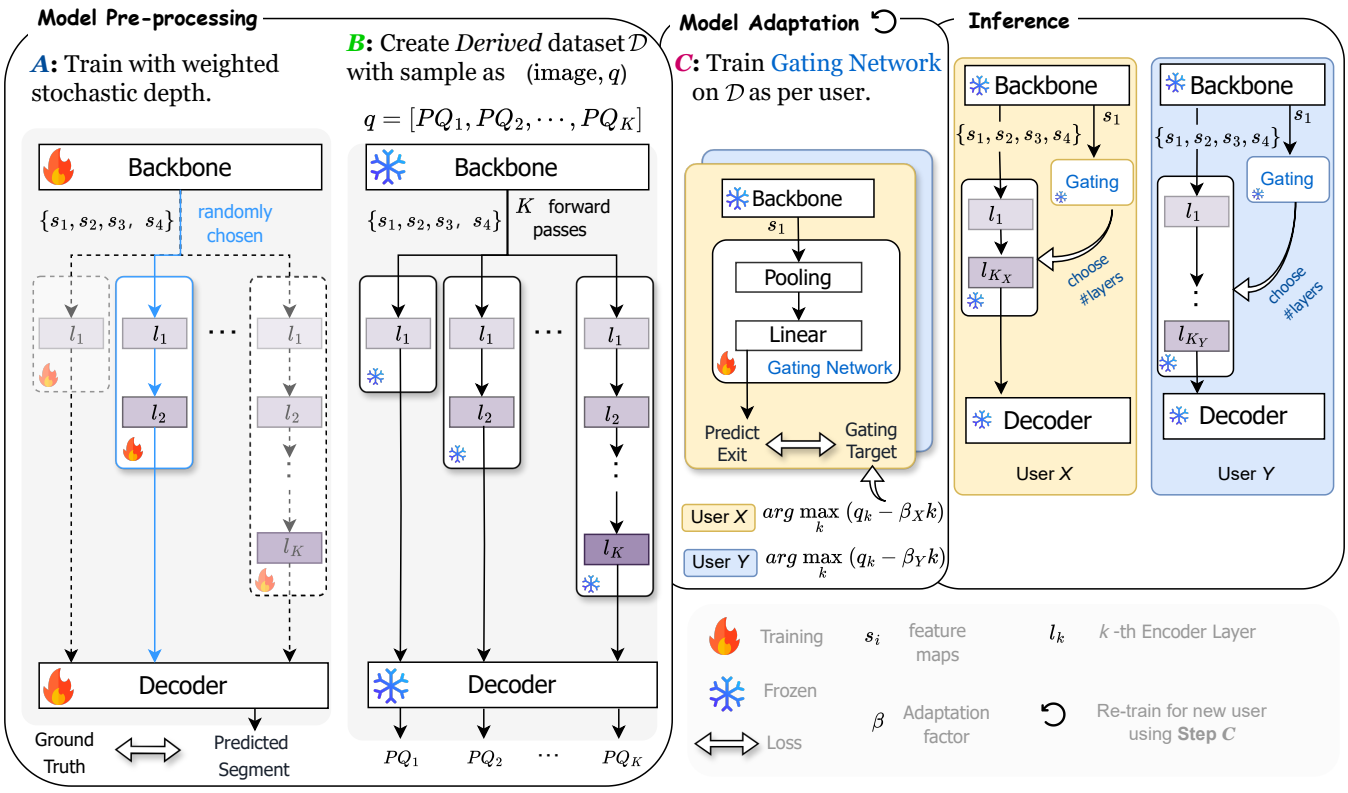


Figure 3. **ECO-M2F framework.** During the *model pre-processing* phase, we train the model to exit stochastically at K potential exits using Step **A**. Next, in Step **B**, we use this model to perform inference on the training images at each exit to create a dataset $\mathcal{D}$. In the *model adaptation* 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 exists at the layer designated by the gating network.

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

1. Step **A**: Train the parent model for early exit via the transformer encoder.
2. Step **B**: Derive a dataset (which we call the Derived dataset) from the dynamic model obtained in Step **A**.
3. 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.

3.3. Training

Step A: Training 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 the experiment results. 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).

Step B: Deriving the Gating Network Training Set 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 $(\boldsymbol{x}^{(i)}, \boldsymbol{q}^{(i)})$.

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 a 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 experiments.

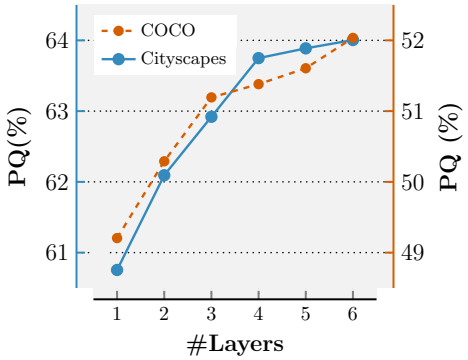


Figure 4. **Intuition for Eq. (4).** This figure shows that prioritizing PQ requires more encoder layers, while fewer layers lead to poorer PQ. (Backbone: SWIN-T; training set).

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.

3.5. Experiments Settings

Datasets Our study illustrates the adaptability of ECO-M2F in dynamically managing the trade-off between computation and performance based on M2F [6] 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 [6] 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 [6]. 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 [38], and ReMax [26].

Architecture Details We focus on standard backbones Res50 [12] 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 [6]. 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 [6], 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, 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.

Model	Performance (↑)			GFLOPs (↓)	
	PQ	mIoU_p	AP_p	Total	Encoder
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 [6]	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 [6]	51.73	61.94	41.72	229.10	135.00
MF [5]	46.50	57.80	33.00	181.00	–
PEM [3]	46.38	55.95	34.25	110.90	–
YOSO† [15]	48.40	58.74	36.87	114.50	–
RAP-SAM†[38]	46.90	–	–	123.00	–
ReMax† [26]	53.50	–	–	–	–
ECO-M2F	51.89	61.07	41.25	195.55	92.37

Table 1. **Evaluation on COCO dataset.**. Our method achieves competitive performance with notable reductions in GFLOPs and can be combined with other efficient encoder designs to further enhance overall efficiency. †Task-specific architectures

3.6. Main Results

As shown in Fig. 2 (b), ECO-M2F accuracy-efficiency trade-off can be better than the baseline models. Detailed comparisons are provided in Tab. 1 and Sec. 3.6, where we benchmark ECO-M2F against baseline prior works on the validation set of COCO and Cityscapes datasets, respectively. In Tab. 1, we observe that ECO-M2F effectively reduces computational costs while upholding performance levels in comparison to M2F [6] using both SWIN-T [22] and Res50 [12] backbones. Additionally, ECO-M2F can be seamlessly integrated into efficient encoder designs, such as Lite-M2F [6] [20], further reducing GLOPs by ap-

Model	Performance (↑)			GFLOPs (↓)	
	PQ	mIoU_p	AP_p	Total	Encoder
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 [6]	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 [6]	61.86	76.94	37.35	524.11	281.13
PEM [3]	61.07	77.62	34.11	236.60	–
YOSO† [15]	59.70	76.05	–	265.10	–
ReMax†[26]	65.40	–	–	294.70	–
ECO-M2F	62.20	77.34	37.21	453.50	220.59

Table 2. **Evaluation on Cityscapes dataset**. Our method balances strong performance with lower GFLOPs, making it more efficient than other models that excel in only one aspect. †Task-specific architectures

Method	β	PQ	mIoU_p	AP_p	GFLOPs
M2F	-	39.71	46.09	26.49	139.81
ECO-M2F	0.0005	38.79	44.71	24.81	127.21
ECO-M2F	0.0010	38.80	44.22	24.78	118.23
ECO-M2F	0.0020	38.63	44.11	24.59	116.95

Table 3. **Evaluation on ADE20K dataset**. The performance of ECO-M2F on the ADE20K dataset highlights its capability to adaptively balance efficiency and effectiveness. (backbone: Res50)

proximately 12.6%. With Res50 as the backbone, MF [5], YOSO [15], and RAP-SAM [38] underperform compared to ECO-M2F. While ReMax [26] shows competitive accuracy, its specialization in panoptic segmentation limits its general applicability. Our work, however, aims for a broader impact by creating efficient segmentation architectures that can be used for various segmentation tasks and enhance the parent architecture without the size constraints of ReMax. We observe similar results on the Cityscapes dataset (see Sec. 3.6) and on the ADE20K dataset [43] (as shown in Sec. 3.6).

3.7. Ablation Studies

Impact of Adaptation Factor β We analyze the impact of β on ECO-M2F and present our analysis in Tab. 4. 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.

ECO-M2F *vs.* M2F w/ Weighted Stochastic Depth (WSD) We present a comparison between M2F trained with WSD and ECO-M2F in Fig. 5. The blue dots denote the results when M2F w/ WSD exits always at ith layer ($i = 2, \cdots, 6$). The red "x"s indicate the performance of our approach with the introduction of the Gating module, which varies with changes in β. The results clearly demonstrate that ECO-M2F effectively self-selects encoder layers for compute reduction while preserving performance, illustrating its efficiency in optimizing the trade-off between computation and accuracy.

Data	β	Performance (↑)			GFLOPs (↓)	
		PQ	mIoU_p	AP_p	Total	Encoder
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 the adaptation factor β**. As the value of β increases, the model places greater emphasis on reducing GFLOPs over performance both in COCO and Cityscapes datasets. Baseline here is M2F [6]. (Backbone: SWIN-T)

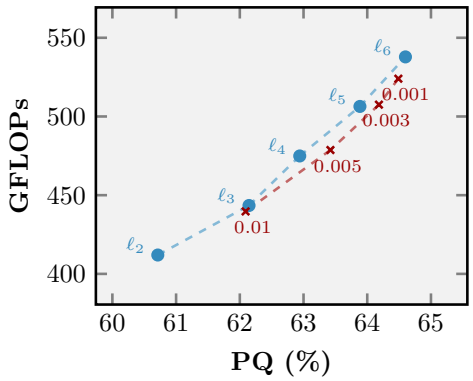


Figure 5. ECO-M2F outperforms M2F with WSD by effectively reducing computation without compromising performance, as shown by the varying results with the Gating module across different β values. (Dataset: Cityscapes)

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 Sec. 3.3, using one-hot target and cross-entropy loss (referred to as "hard-CE" in Tab. 5), with three alternative methods that do not involve

Method	Performance (↑)			GFLOPs (↓)	
	PQ	mIOU_p	AP_p	Total	Encoder
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

Table 5. **Impact of target and loss in gating network training**. We use "hard-CE" loss for training our gating network in all of the other results. (Backbone: SWIN-T; Dataset: COCO)

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.*,

$$\mathcal{L}_{\text{gating}} = \sum_{i}^{N} \sum_{k}^{K} \Big[\text{softmax}(u^{(i)}(k)) - \text{softmax}(g_k^{(i)})\Big]^2 \,.$$

The analysis in Tab. 5 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, all of the other results use this approach.

Batch Inference and FPS Comparison ECO-M2F maintains efficient batch processing by utilizing a shared decoder across all encoder exit points, ensuring that throughput is not compromised. As shown in Fig. 6, our method significantly improves the frames per second (FPS) across various batch sizes, demonstrating a clear advantage in processing speed.

4. Conclusions

In this paper, we propose an efficient transformer encoder design ECO-M2F for the Mask2Former-style frameworks. ECO-M2Fprovides 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 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.

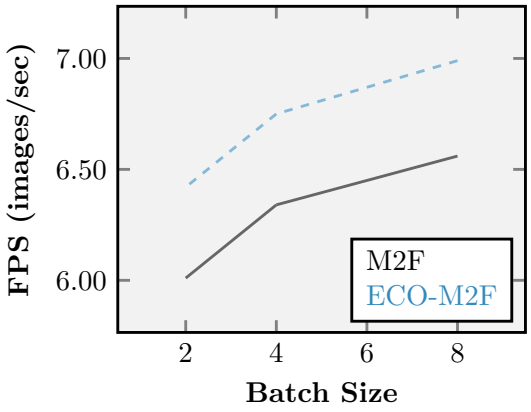


Figure 6. **FPS vs. batch size.** ECO-M2F consistently achieves higher FPS than M2F [6] across all batch sizes. (Dataset: Cityscapes; Backbone: Res50)

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.

References

[1] Abdallah Ammar, Mahmoud I. Khalil, and Cherif Salama. Rt-yoso: Revisiting yoso for real-time panoptic segmentation. In *2023 5th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 306–311, 2023. 2

[2] 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*, pages 213–229. Springer, 2020. 1, 2

[3] Niccolò Cavagnero, Gabriele Rosi, Claudia Cuttano, Francesca Pistilli, Marco Ciccone, Giuseppe Averta, and Fabio Cermelli. Pem: Prototype-based efficient maskformer for image segmentation. *CVPR*, 2024. 6, 7

[4] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. 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*, pages 12475–12485, 2020. 2

[5] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021. 1, 2, 6, 7

[6] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 1, 2, 3, 6, 7, 8

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2, 6

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[9] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136, 2015. 6

[10] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking bisenet for real-time semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9716–9725, 2021. 2

[11] Xiuye Gu, Yin Cui, Jonathan Huang, Abdullah Rashwan, Xuan Yang, Xingyi Zhou, Golnaz Ghiasi, Weicheng Kuo, Huizhong Chen, Liang-Chieh Chen, et al. Dataseg: Taming a universal multi-dataset multi-task segmentation model. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 6

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[14] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8523–8532, 2020. 2

[15] Jie Hu, Linyan Huang, Tianhe Ren, Shengchuan Zhang, Rongrong Ji, and Liujuan Cao. You only segment once: Towards real-time panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17819–17829, 2023. 1, 2, 6, 7

[16] Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. Oneformer: One transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2989–2998, 2023. 1, 2

[17] Zihang Jiang, Zhiqiang Gong, Yifan Xu, and Jianming Wang. Multi-exit vision transformer with custom fine-tuning for fine-grained image recognition. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 5233–5237, 2023. 2, 3

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 6

[19] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation, 2019. 1, 2, 3, 5, 6

[20] Feng Li, Ailing Zeng, Shilong Liu, Hao Zhang, Hongyang Li, Lei Zhang, and Lionel M Ni. Lite detr: An interleaved multi-scale encoder for efficient detr. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18558–18567, 2023. 1, 6, 7

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 2, 6

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1, 6, 8

[23] Zehui Liu, Yutong Sun, Yiming Li, Zhiqiang Zhou, Jinshan Hu, and Fei Li. Multi-exit vision transformer for dynamic inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5214–5223, 2021. 2, 3

[24] Wenyu Lv, Shangliang Xu, Yian Zhao, Guanzhong Wang, Jinman Wei, Cheng Cui, Yuning Du, Qingqing Dang, and Yi Liu. Detrs beat yolos on real-time object detection. *arXiv preprint arXiv:2304.08069*, 2023. 1, 6, 7

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[26] Shuyang Sun, Weijun Wang, Qihang Yu, Andrew Howard, Philip Torr, and Liang-Chieh Chen. Remax: Relaxing for better training on efficient panoptic segmentation. *arXiv preprint arXiv:2306.17319*, 2023. 1, 6, 7

[27] Jiacheng Tang, Zehui Liu, Yiming Li, Yutong Sun, Zhiqiang Zhou, Jinshan Hu, and Fei Li. 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), pages 13504–13513, 2023. 2, 3

[28] Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. 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*, pages 10781–10791, 2023. 2, 3

[29] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008. 1

[30] Florian Valade, Mohamed Hebiri, and Paul Gay. Eero: Early exit with reject option for efficient classification with limited budget, 2024. 2, 3

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 2

[32] Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 1, 2023. 2

[33] Xin Wang, Wenhai Zhou, Xiaohong He, Xiangyu Peng, Feng Wei, and Yuhang Guo. Single-layer vision transformers for more accurate early exits with less overhead. *Pattern Recognition*, 136:102243, 2022. 2, 3

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[35] Falcon Xu, Xiang Zhang, Zhendong Ma, Jianbo Wang, Jing Hu, and Jian Sun. Lgvit: Dynamic early exiting for accelerating vision transformer. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1958–1966, 2023. 2, 3

[36] Jiacong Xu, Zixiang Xiong, and Shankar P Bhattacharyya. Pidnet: A real-time semantic segmentation network inspired by pid controllers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19529–19539, 2023. 2

[37] Mengwei Xu, Wangsong Yin, Dongqi Cai, Rongjie Yi, Daliang Xu, Qipeng Wang, Bingyang Wu, Yihao Zhao, Chen Yang, Shihe Wang, et al. A survey of resource-efficient llm and multimodal foundation models. *arXiv preprint arXiv:2401.08092*, 2024. 2

[38] Shilin Xu, Haobo Yuan, Qingyu Shi, Lu Qi, Jingbo Wang, Yibo Yang, Yining Li, Kai Chen, Yunhai Tong, Bernard Ghanem, et al. Rap-sam: Towards real-time all-purpose segment anything. *arXiv preprint arXiv:2401.10228*, 2024. 1, 6, 7

[39] Junyi Yang, Xiaoling Zhang, Xin Zhang, Jilin Tang, and Xiwei Li. Exploiting face recognizability with early exit vision transformers. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 6341–6345, 2023. 2, 3

[40] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 2

[41] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision*, 129: 3051–3068, 2021. 2

[42] Tianyu Zhang, Xinyu He, Zhongli Qin, and Jian Sun. Adaptive deep neural network inference optimization with eenet. In *Proceedings of the 37th International Conference on Machine Learning*, pages 15983–15993, 2023. 2, 3

[43] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5122–5130, 2017. 7