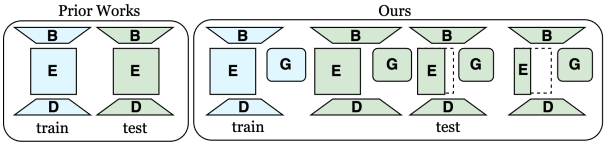
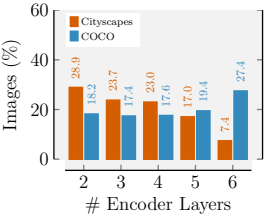
[c Ap





images achieving best

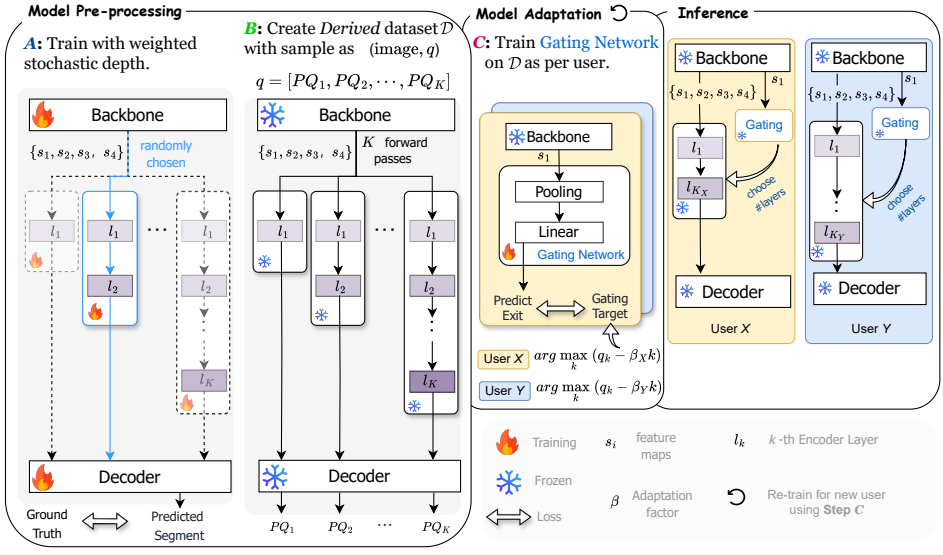
reduce computations.

icant progress in overall performance across various tasks, these models still face challenges in deployment on resource-constrained devices. Current empha-

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

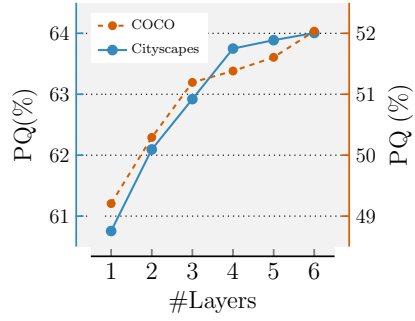
$$\{\tilde{\boldsymbol{y}} \qquad \} = h \circ f_K \circ \cdots \circ f_2 \circ f_1 \circ b(\boldsymbol{x}^{(i)}) \,. \tag{1}$$

all performance remains within acceptable bounds. Hence, we adopt an adaptive



ECO-M2F involves following three main steps:

$$\sum_{N}\sum_{K}\alpha_k\mathcal{L}_k,\ \text{where}\ \forall k<k', \alpha_k<\alpha_{k'}\ , \tag{3}$$



(4)

(5)

(6)

Saving training costs through Step C. `ECO-M2F` presents a distinct ad-

scenarios where a smaller model is desired, `ECO-M2F` necessitates training solely

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.*, $K_X < K_Y$.

that `ECO-M2F` is versatile and resource-efficient as it adapts to diverse needs and optimizes allocations.

ture M2F [5]. Further, we also integrate recently proposed transformer 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 [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
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	–
ECO-M2F	51.89	61.07	41.25	195.55	92.37

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
M2F	61.86	76.94	37.35	524.11	281.13
YOSO [15]	59.70	76.05	33.76	265.1	–
ECO-M2F	62.20	77.34	37.21	453.50	220.59

Tiny [22] pre-trained on ImageNet-1K [8], unless specified otherwise. We set the straightforward 1D adaptive average pooling operation as our pooling function.

Baseline is M2F [5]. (Backbone: SWIN-T)

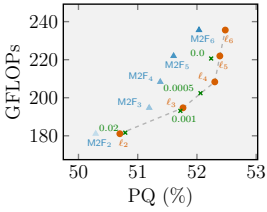
Baseline	52.03	62.49	42.18	235.57	121.69
0.0	52.24	62.95	41.61	220.61	107.18
0.0005	52.06	62.76	41.51	202.39	88.47
0.001	51.72	62.60	41.12	193.10	79.18
0.02	50.79	62.25	39.71	181.64	67.71
Baseline	64.00	80.77	39.26	537.85	281.13
0.0	64.58	80.35	40.31	536.09	279.37
0.003	64.18	80.49	39.64	507.51	250.80
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

$K = 6, 5, 4$ respectively. (Dataset: COCO; Backbone: SWIN-T)

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

Table 5: Stochastic Depth (SD) training. Here, all models (w/ 6 encoder layers) trained w/ total computational resources are limited, we present an approach using ECO-M2F Lite-M2F meta-architecture as well (see supplementary material).

Model	PQ (↑)				
	2	3	4	5	6
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
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



Values denote β. Dataset: COCO.

ECO-M2F shows strong performance w.r.t. Weighted. (Baseline: M2F w/ SWIN-T)

coder. †ImageNet-21K pre-trained

SWIN-T	M2F [5]	52.03	62.49	42.18	235.57	121.69
	ECO-M2F	52.06	62.76	41.51	202.39	88.47
SWIN-S	M2F [5]	54.63	64.24	44.69	316.50	121.69
	ECO-M2F	54.76	64.46	44.48	275.96	81.05
SWIN-B†	M2F	56.40	67.09	46.29	470.98	122.56
	ECO-M2F	56.49	66.56	46.40	425.17	76.50
SWIN-T	M2F [5]	64.00	80.77	39.26	537.85	281.13
	ECO-M2F	64.18	80.49	39.64	507.51	250.80
SWIN-S	M2F [5]	64.84	81.76	40.73	724.29	281.13
	ECO-M2F	65.12	81.64	41.17	665.38	222.22
SWIN-B†	M2F [5]	66.12	82.70	42.84	1051.19	283.14
	ECO-M2F	65.44	82.05	40.51	984.73	216.69

– *Pre-trained weight variations.* We explore how different pre-training strate-

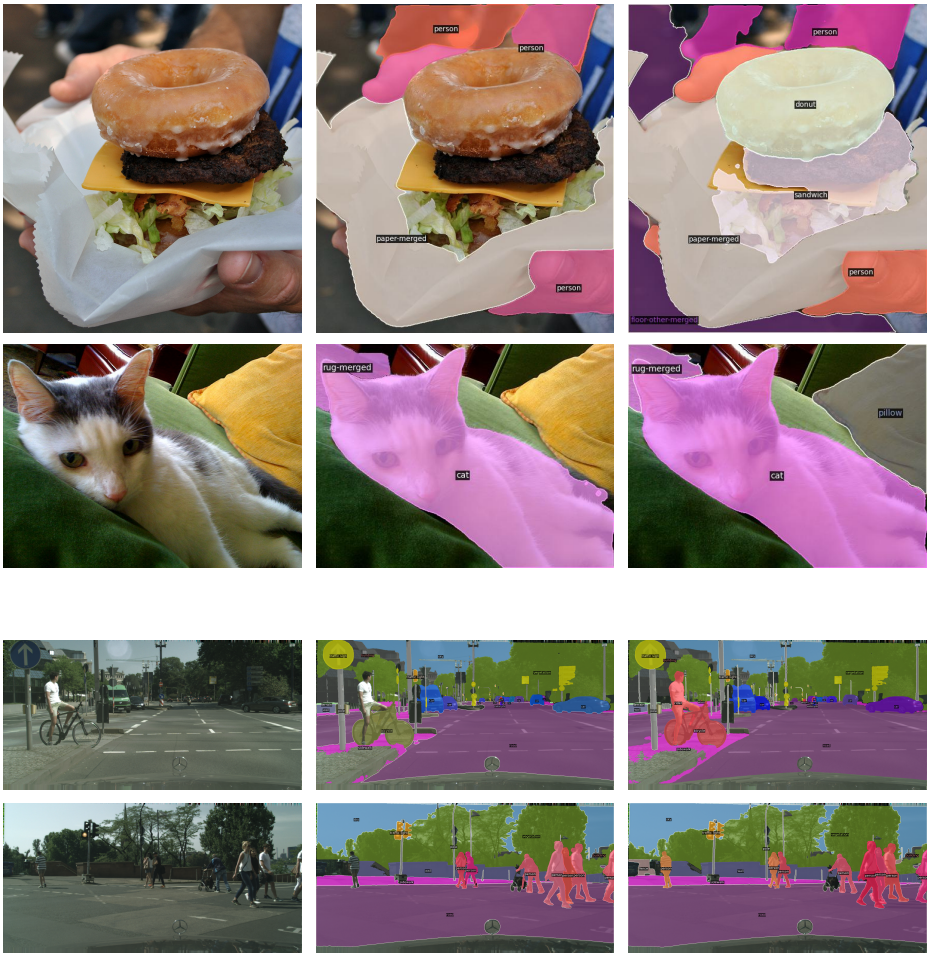
Backbone: Res50)

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

SWIN-T(MoBY [36])	M2F [5]	51.64	62.42	41.93	235.57	121.69
	ECO-M2F	51.48	62.49	41.27	203.56	89.64
Res50(DINO [3])	M2F [5]	51.57	61.65	41.24	229.41	126.05

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-

transformer encoder without significantly impacting performance).



tation maps from M2F [5] (*middle* column) and `ECO-M2F` (*last* column). *Top* two rows

rameter that needs separate tuning for each use case. This is because it relies on

(Supplementary Material)

using various backbone sizes, including SWIN-Tiny (T), SWIN-Small (S), and

encoder. We retain layer 6 and do not consider it as a feasible exit point as

one-hot target and cross-entropy loss (referred to as "hard-CE" in Tab. T2), with

$$\sum^{N}\sum^{K} u^{(i)}(k)\ln[\mathrm{softmax}(g^{\prime\prime}$$

$$\sum^{N}\sum^{K} \mathrm{softmax}(u^{(i)}(k))\ln[\mathrm{softmax}(g^{\prime\prime}$$

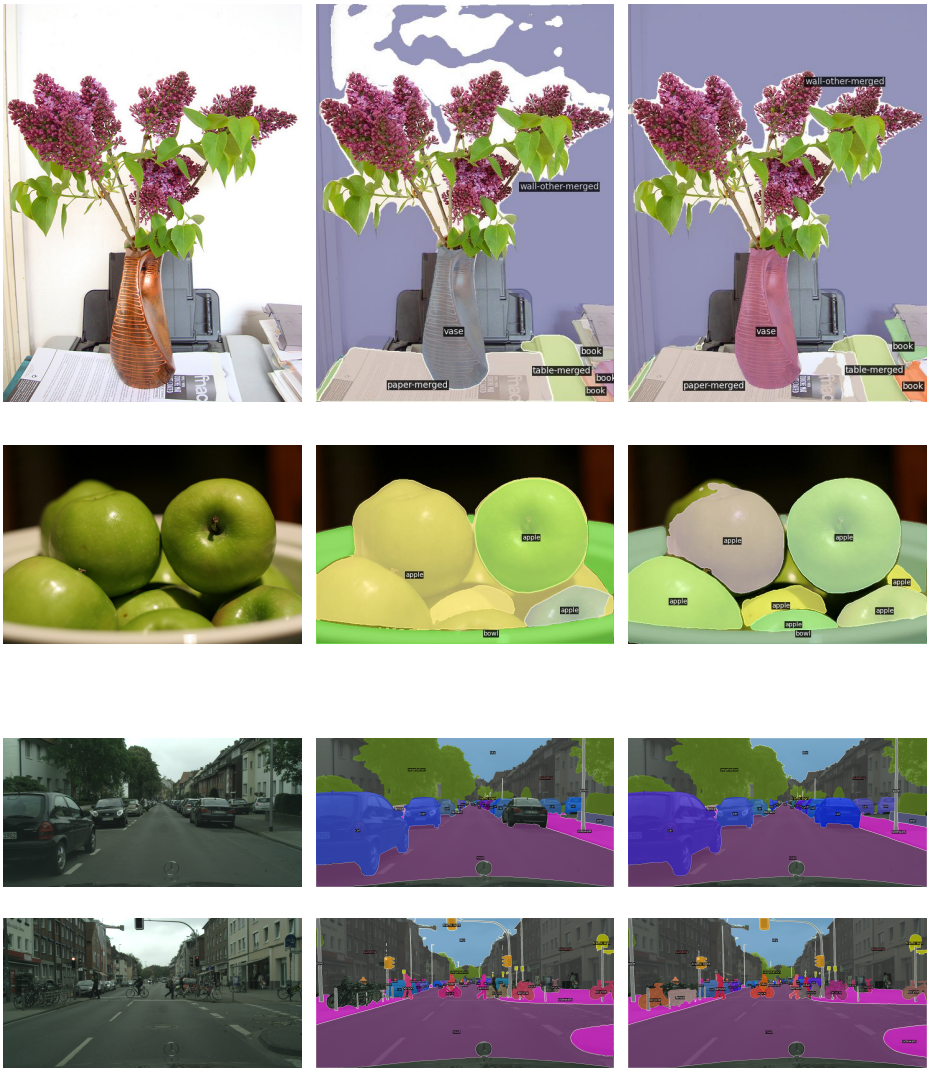
$$\sum^{N}\sum^{K}\Big[\text{softmax}(u^{(i)}(k)) - \text{softmax}(g$$

CE" loss for training our gating network

Dataset: COCO)

SWIN-T	Lite-M2F [20]	62.29	79.43	36.57	428.71	172.00
	Lite-ECO-M2F	62.64	79.99	36.52	412.88	156.17
SWIN-S	Lite-M2F [20]	63.54	79.74	39.12	615.15	171.99
	Lite-ECO-M2F	63.32	80.21	37.91	588.82	145.66
SWIN-B	Lite-M2F [20]	64.48	82.34	39.21	942.05	174.01
	Lite-ECO-M2F	64.66	81.40	39.52	921.15	153.11
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	64.14	43.55	258.00	63.96

Method	**Performance** (↑)			**GFLOPs** (↓)	
	PQ	**mIOU**$_p$	**AP**$_p$	**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



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

NILES59815.2023.10296714 4

5, 9, 10, 11, 12, 13, 14, 17

17864–17875 (2021) 1, 3, 10, 11

(2024) 1, 3

4, 9, 10, 11

1, 3

(2019) 1, 2, 5, 7, 9

2, 3, 9, 10, 11, 15, 16

(2021) 2, 10, 12, 13, 15, 16

(2023) 2, 9, 10

(2023) 2, 9, 10, 11

(CVPR). pp. 13504–13513 (2023) 2, 4, 5

github.com/facebookresearch/detectron2 (2019) 10