Appendix

We provide more details of our experiments, including the configuration of all models reported in the main text (Appendix A) and the training setup (Appendix B). We also provide the speed test results of Dyn-Perceiver implemented on ResNet (Appendix C).

A. Model Configuration

In Fig. 8 and Fig. 9, we reported the results of different-sized Dyn-Perceiver. Here we list the detailed configuration of Dyn-Perceiver implemented on ResNet (Tab. 5), RegNet-Y (Tab. 6) and MobileNet-v3 (Tab. 7). SA is the abbreviation of self-attention.

B. Training Settings

Image classification. The training settings of Dyn-Perceiver are demonstrated in Tab. 8. For simplicity, we use the same settings to train MobileNet-based models and the ResNet/RegNet-based models, except for the training epochs. For each experiment, we select batch size from {1024, 2048} based on model sizes and the GPU memory.
Action recognition. We use the TSM [34] code base[2] and add temporal shift to our latent code $\mathbf{Z}$ in self-attention blocks. We follow the settings in the official implementation and sum up the classification loss from different exits with the same weights as in ImageNet training.
Object detection. We finetune the ImageNet-pretrained checkpoints on COCO for 12 epochs following the official settings of RetinaNet [36] in the MMDetection code base[3] Since the feature maps from different stages are required by the detection head, we do not perform early exiting here, and the experiment is only to demonstrate the capability of Dyn-Perceiver to serve as a detection backbone.

C. Speed Test for ResNet-based Model

We also test the practical efficiency of our smallest ResNet-based Dyn-Perceiver (model 1 in Tab. 5) on the desktop i5 CPU and the A100 GPU. The latency-accuracy curves on CPU and GPU are presented in Fig. 13a and Fig. 13b, respectively. It could be observed that Dyn-Perceiver achieves satisfying speedup on the two devices when achieving the same accuracy with the baseline.

[2] https://github.com/mit-han-lab/temporal-shift-module.
[3] https://www.github.com/open-mmlab/mmdetection.

	ResNet50-based Model				
Model index	1	2	3	4	5
ResNet width factor	0.375×	0.5×	0.5×	0.625×	0.75×
token number L	128	128	256	192	128
# SA blocks	[3,3,9,3]	[3,3,9,9]	[3,3,9,3]	[3,3,9,3]	[3,3,9,3]
SA widening factor	4	4	4	4	2

Table 5: ResNet-50 configurations. The 5 configurations correspond to the 5 curves in Fig. 8 (a) of the paper.

	RegNet-based Model					
Model index	1	2	3	4	5	6
RegNet size	400M	400M	800M	800M	1.6G	3.2G
token number L	128	256	128	256	256	256
# SA blocks	[6,6,9,9]	[3,3,9,9]	[3,3,9,6]	[3,3,9,6]	[6,6,9,6]	[6,6,9,9]
SA widening factor	4	4	4	4	2	4

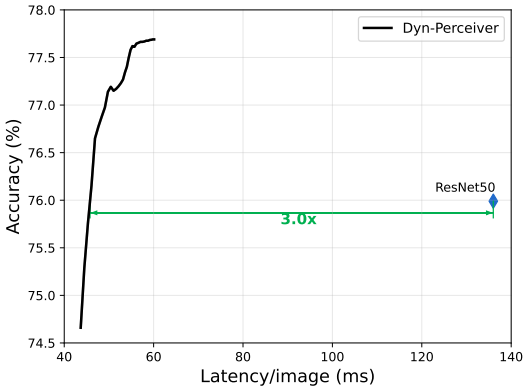
Table 6: RegNet-Y configurations. The 6 configurations correspond to the 6 curves in Fig. 8 (b) of the paper.

	MobileNet-based Model				
Model index	1	2	3	4	5
MobileNet width factor	0.75×	1.00×	1.00×	1.25×	1.5×
token number L	128	128	128	128	256
# SA blocks	[3,3,9,9]	[3,3,9,9]	[6,6,9,9]	[6,6,9,9]	[3,3,9,9]
SA widening factor	4	4	4	4	4

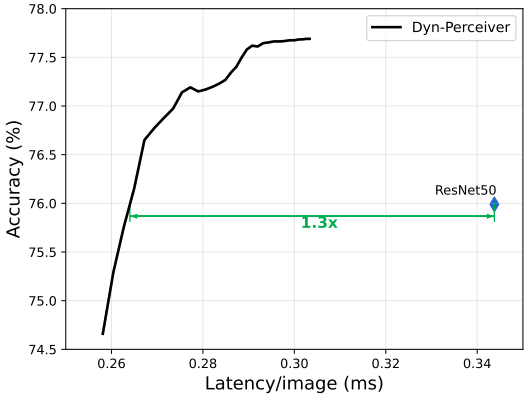
Table 7: MobileNet-v3 configurations. The 5 configurations correspond to the 5 curves in Fig. 9 of the paper.

Training Config	Dyn-Perceiver MobileNet-based	Dyn-Perceiver ResNet/RegNet-based
optimizer	adamW	adamW
base learning rate	{1e-3, 2e-3}	{1e-3, 2e-3}
weight decay	0.05	0.05
optimizer momentum	β_1, β_2=0.9, 0.999	β_1, β_2=0.9, 0.999
batch size	{1024, 2048}	{1024, 2048}
training epochs	600	300
learning rate schedule	cosine decay	cosine decay
warmup epochs	20	20
warmup schedule	linear	linear
layer-wise lr decay [1]	None	None
randaugment [9]	(9, 0.5)	(9, 0.5)
label smoothing [52]	0.1	0.1
mixup [72]	0.8	0.8
cutmix [70]	1.0	1.0
stochastic depth [26]	None	None
layer scale [55]	None	None
gradient clip	None	None
exp. mov. agv.(EMA) [48]	0.9999	0.9999

Table 8: ImageNet-1K training settings.



(a) CPU.



(b) GPU.

Figure 13: Speed test results of Dyn-Perceiver built on ResNet (0.375×, model 1 in Tab. 5).