S1 Implementation Details and Further Results on ImageNet

S1.1 Implementation Details on ImageNet

We performed our MoD experiments on ImageNet for 100 epochs, using PyTorch 1.10.1 [26] across four Nvidia GeForce 1080Ti GPUs, following standard training protocols.[⋆⋆⋆] Our configuration included a batch size of 256, an initial learning rate of 0.1 decreasing by a factor of 0.1 every 30 epochs, momentum set at 0.9, and weight decay of 1×10^{-4}. This setup aligns with the practices of comparable studies, such as DGNet [19].

S1.2 Architecture Details of ResNet MoD Models

Table 5 shows the layer configurations of the ResNet MoD models. The models incorporates alternating block patterns in each module.

Table 5: Layer Configurations of ResNet MoD Models

Model	Layer Configuration	Block Type
ResNet18-MoD	[2, 2, 2, 2]	Basic Block
ResNet26-MoD	[2, 2, 3, 4]	Basic Block
ResNet34-MoD	[3, 4, 6, 3]	Basic Block
ResNet42-MoD	[3, 3, 6, 6]	Basic Block
ResNet50-MoD	[3, 4, 6, 3]	Bottleneck Block
ResNet75-MoD	[3, 4, 14, 3]	Bottleneck Block
ResNet86-MoD	[3, 4, 18, 3]	Bottleneck Block
ResNet101-MoD	[3, 4, 23, 3]	Bottleneck Block
ResNet152-MoD	[3, 8, 36, 3]	Bottleneck Block

S1.3 Architecture Details of MobileNetV2 MoD Models

The MobileNetV2-MoD-L represents a deeper configuration of the standard MobileNetV2. The standard MobileNetV2 [31] architecture utilizes a sequence of inverted residual blocks with a specific configuration pattern defined by the parameters t (expansion factor), c (number of channels), n (number of times the block is repeated), and s (stride). The MobileNetV2-MoD-L model modifies these parameters as shown in Table 6.

S1.4 Comparative Performance of ResNet Models on ImageNet

Table 7 shows further results for MoD-enhanced ResNets, detailing improvements in computational efficiency, model compactness, and inference speeds.

[⋆⋆⋆] Refer to PyTorch's official training recipes at PyTorch repository.

Table 6: Comparison of Standard MobileNetV2 and MobileNetV2-MoD-L Configurations

Layer	**Standard Configuration**	**MoD-L Configuration**
1	[1, 16, 1, 1]	[1, 16, 1, 1]
2	[6, 24, 2, 2]	[6, 32, 2, 2]
3	[6, 32, 3, 2]	[6, 64, 3, 2]
4	[6, 64, 4, 2]	[6, 96, 4, 2]
5	[6, 96, 3, 1]	[6, 128, 3, 1]
6	[6, 160, 3, 2]	[6, 160, 3, 2]
7	[6, 320, 1, 1]	[6, 320, 1, 1]

Table 7: Comparative performance of standard and ResNet-MoD models on the ImageNet dataset. The table evaluates top-1 accuracy, computational complexity (GMAC), model size (Params, in millions), and inference speed improvements on CPU and GPU.

Method	**Top-1**	**GMAC**	**Params**	**Inference (ms)**		**Speed-up**	
	Acc (%)		**(M)**	**CPU**	**GPU**	**CPU**	**GPU**
ResNet34	73.92	3.68	21.29	98.83	1.27	—	—
ResNet42-MoD	72.03	2.29	17.72	64.01	0.88	1.54	1.45
ResNet34-MoD	71.44	2.06	12.93	60.79	0.83	1.63	1.53
ResNet18	70.37	1.82	11.18	53.86	0.75	—	—
ResNet26-MoD	69.53	1.36	11.4	42.16	0.58	1.28	1.28
ResNet18-MoD	64.05	0.89	5.46	33.63	0.47	1.60	1.58

S1.5 Impact of Channel Parameter c and Integration Strategy on Accuracy and Inference Times

We evaluate the influence of the channel parameter c on the top-1 validation accuracy and inference times of the ResNet50-MoD model on the ImageNet dataset. The parameter c determines the number of channels processed by the MoD approach. Additionally, we compare two strategies for reintegrating processed channels: adding them to the first k channels (S) versus adding them back to their original positions (OP).

As shown in Table 8, reintegrating processed channels into the first k channels consistently outperforms the original position strategy, yielding better accuracy across all values of c. Furthermore, using $\mathbf{c = 64}$ has the optimal balance between accuracy and inference time. It should be noted that $c > 64$ is not feasible since the number of channels in the first Conv-Block of ResNets is limited to 64.

To further analyze the integration strategy, Table 9 presents a comparison of the performance of standard and ResNet-MoD models where the processed channels are added to the last k channels of the original feature map. This experimental variation aims to assess the impact of consistent channel positioning on network performance. The results indicate that MoD-lk models perform comparably to their counterparts where processed channels are added to the first k

Table 8: Top-1 accuracy and inference times in ms on ImageNet for different values of c, comparing results when processed channels are added to the first k channels (S) versus their original positions (OP) for ResNet50-MoD.

c	Top-1 (S)	Top-1 (OP)	GPU (S)	CPU (S)
2	72.51	55.36	2.23	132.64
4	73.74	60.78	1.99	118.85
8	74.43	63.05	1.90	114.85
16	74.74	65.35	1.97	114.06
32	74.68	70.80	1.83	111.30
64	74.79	70.31	1.75	108.74

channels, suggesting that maintaining a consistent position for processed information within the feature maps is beneficial for optimizing model performance.

Table 9: This table presents a comparison between standard MoD models and the MoD-lk models, where processed channels are integrated into the last k channels. The evaluation covers top-1 accuracy, computational complexity (GMAC), model size (in millions of parameters), and inference speed improvements on both CPU and GPU. The aim is to analyze the impact of channel positioning on the performance of ResNet models on the ImageNet dataset.

Method	Top-1	GMAC	Params	Inference (ms)		Speed-up	
	Acc (%)		(M)	CPU	GPU	CPU	GPU
ResNet86-MoD	76.72	3.92	25.60	150.96	2.40	1.06	1.05
ResNet86-MoD-lk	76.64	3.92	25.60	150.96	2.40	1.06	1.05
ResNet75-MoD	76.27	3.48	23.10	128.90	2.19	1.25	1.15
ResNet75-MoD-lk	76.22	3.48	23.10	128.90	2.19	1.25	1.15
ResNet50-MoD	74.79	2.60	18.11	108.74	1.75	1.48	1.44
ResNet50-MoD-lk	74.57	2.60	18.11	108.74	1.75	1.48	1.44

S1.6 Impact of MoD on Performance Variance

We provide standard deviations for the ResNet experiments on ImageNet in Table 10. Other state-of-the-art pruning and dynamic computation methods, such as DGNet [19], Batch-Shaping [1], ConvNet-AIG [36], HRANK [21], FPGM [13], and DynConv [37], do not report standard deviations, preventing direct comparison. Nevertheless, the consistent results across different MoD configurations demonstrate that the MoD approach does not introduce additional variance, as evidenced by the low standard deviations.

Table 10: Top-1 Accuracy and standard deviation comparison on ImageNet across various ResNet and ResNet-MoD models. This comparison shows that the MoD approach does not increase variance and maintains performance stability similar to standard ResNet models.

Method	**Top-1 Acc. (%) $\pm$ Std. Dev.**
R152-MoD	77.81 ± 0.05
R101	77.81 ± 0.07
R101-MoD	77.08 ± 0.08
R86-MoD	76.72 ± 0.04
R75-MoD	76.27 ± 0.07
R50	76.25 ± 0.19
R50-MoD	74.79 ± 0.08

S2 Implementation Details Semantic Segmentation

In the Cityscapes experiments, PyTorch 1.10.1 [26] and four Nvidia GeForce 1080Ti GPUs were used. Using the MMSegmentation Framework [4], we utilized FCN [23] on the dataset [5], which consists of 2,975 training, 500 validation, and 1,525 testing images across 19 semantic classes. Training involved resizing, random cropping, flipping, photometric distortion, normalization, and padding. Testing employed multi-scale flip augmentation and normalization.

The FCN model, with a ResNet50 backbone, used an Encoder-Decoder architecture with FCN-Head as the decode and auxiliary heads. The model used SyncBN and a dropout ratio of 0.1, with the auxiliary head contributing 40% to the total loss.

Optimization was via SGD (learning rate 0.01, momentum 0.9, weight decay 0.0005). A polynomial decay learning rate policy was applied (power 0.9, minimum learning rate 1e-4), over 80,000 iterations with checkpoints and evaluations (focusing on mIoU) every 8,000 iterations.

S3 Implementation Details Object Detection

Model Configuration: We configure our Faster R-CNN [29] with a ResNet-50 backbone and a Feature Pyramid Network (FPN) neck for multi-scale feature extraction.

Data Preprocessing and Augmentation: Our preprocessing pipeline employs a sequence of transformations to prepare input images for object detection tasks. Initially, images are loaded and their corresponding annotations are retrieved. Subsequently, we resize the images to a resolution of $(1000, 600)$, ensuring the preservation of their original aspect ratio. To augment the dataset and introduce variability, we apply random horizontal flips with a 50% probability. This augmentation strategy is applied uniformly across the training dataset, aiming to enhance model robustness and generalization capability. For validation, images undergo a similar resizing process without the application of random flips, maintaining consistency in evaluation conditions.

Training Configuration: The model is trained on the combined train sets of VOC2007 and VOC2012 and evaluated on the VOC2007 val set. Training uses a batch size of 2. We adopt SGD with momentum and weight decay, adjusting the learning rate as per a predefined schedule. The mean Average Precision (mAP) metric, calculated using the "11points" interpolation method, serves as the evaluation metric.

Evaluation: The evaluation on the VOC2007 val set employs the standard VOC mAP metric, adhering to the "11points" method. This setup mirrors the training configuration but without data augmentation, ensuring deterministic inference.

S4 Experiments on CIFAR

Our evaluation of the MoD approach was performed on the CIFAR-10/100 datasets, comprising 50,000 training and 10,000 test color images of 32x32 pixels. We utilized a range of CNN architectures for our experiments, including ResNet18/34/50 [11] and VGG16/19-BN [32]. Table 11 presents the results of applying the MoD approach to the CNNs trained on CIFAR-10/100.

To ensure robustness and reproducibility, each model was trained and evaluated five times using different random seeds, impacting network initialization, data ordering, and augmentation processes. We present the mean test accuracy and its standard deviation for these trials. The data split comprised 90% for training and 10% for validation, with the best-performing model on the validation set chosen for the final evaluation.

S5 Comparative Analysis of MoD in CNNs and Transformers

In the main body of this paper, we detailed the application of the MoD approach to CNNs. This section aims to outline how this approach differs from the Mixture-of-Depths application in Transformers [27].

Token vs. Channel Processing:

- **Transformers:** In Transformers, MoD operates at the token level. Tokens represent the units of data processed throughout the model's architecture, typically as subwords or whole words. They are processed throughout the entire Transformer architecture.
- **CNNs:** Conversely, MoD in CNNs treats channels within feature maps as "tokens". This novel approach differs from Transformers because channels in traditional CNNs are not treated as tokens, and their significance and composition vary from one convolutional layer to the next.

Token vs. Channel Selection:

- **Transformers:** Selection is based on a linear projection that assigns a scalar value to each token.
- **CNNs:** CNNs use a mini neural network (incorporating Adaptive Average Pooling 2D, a two-layer fully connected network with Sigmoid activation) inspired by Squeeze-and-Excitation blocks [15], specifically tailored for image-based tasks.

Architecture Modification:

- **Transformers:** The architecture of Transformer blocks remains unchanged with the use of MoD; only the quantity of processed tokens varies. Transformer models are designed to handle a variable number of tokens.
- **CNNs:** In CNNs, varying the number of channels in convolutional layers is impractical. MoD thus requires adjustments to the convolutional layers themselves, including a reduction in the number of channels in the convolution kernels to match the reduced number of input feature map channels.

Reintegration of Processed Information:

- **Transformers:** Processed tokens are added back to their original counterparts, a method made effective through the use of positional encoding.
- **CNNs:** Unlike in Transformers, neither replacing nor adding back processed channels to their original positions has proven effective in CNNs. More effective is the addition of processed channels to a fixed set of channels, such as the first k channels, to maintain consistency in locating processed information within the network.

Table 11: Performance metrics comparison on the CIFAR-10 and CIFAR-100 datasets using standard models and their MoD variants. This table illustrates the trade-off between efficiency and inference time, demonstrating that the MoD models can achieve comparable performance to the standard models at faster inference times or improved performance at comparable inference times. FLOPS are in millions of multiply-accumulate operations (MMAC), parameters in millions (M), and inference times in milliseconds (ms).

Model	**Set**	**Test Acc.**	**FLOPS**	**Params**	**Inference (ms)**		**Speed-up**	
		(%)	**(MMAC)**	**(M)**	**CPU**	**GPU**	**CPU**	**GPU**
ResNet18	C10	94.04 ± 0.08	557	11.17	15.67	0.24	-	-
ResNet18-MoD	C10	92.37 ± 0.18	255	4.95	8.32	0.14	1.88	1.73
ResNet34	C10	93.69 ± 0.27	1016	21.28	30.52	0.42	-	-
ResNet34-MoD	C10	93.83 ± 0.20	633	12.42	18.10	0.27	1.69	1.54
ResNet50	C10	93.31 ± 0.33	1310	23.52	48.47	0.83	-	-
ResNet50-MoD	C10	93.24 ± 0.24	808	16.07	33.76	0.58	1.44	1.41
ResNet18	C100	76.47 ± 0.18	557	11.22	14.99	0.23	-	-
ResNet18-MoD	C100	72.73 ± 0.21	255	4.99	8.89	0.13	1.69	1.75
ResNet34	C100	77.07 ± 0.41	1160	21.33	30.67	0.42	-	-
ResNet34-MoD	C100	76.86 ± 0.23	633	12.47	18.66	0.27	1.64	1.55
ResNet50	C100	76.17 ± 0.63	1310	23.71	46.64	0.81	-	-
ResNet50-MoD	C100	76.76 ± 0.61	808	16.26	34.81	0.58	1.34	1.41
VGG16-BN	C10	93.27 ± 0.11	315	15.25	8.80	0.14	-	-
VGG16-BN-MoD	C10	91.79 ± 0.14	155	9.83	5.47	0.16	1.61	0.87
VGG19-BN	C10	93.21 ± 0.07	400	20.57	11.55	0.18	-	-
VGG19-BN-MoD	C10	91.82 ± 0.18	155	9.91	5.81	0.22	1.99	0.81
VGG16-BN	C100	72.48 ± 0.32	315	15.30	9.27	0.14	-	-
VGG16-BN-MoD	C100	69.23 ± 0.25	155	9.88	6.41	0.15	1.45	0.95
VGG19-BN	C100	71.34 ± 0.12	400	20.61	11.38	0.18	-	-
VGG19-BN-MoD	C100	69.19 ± 0.11	155	9.96	5.75	0.23	1.98	0.80