CNN Mixture-of-Depths

Rinor Cakaj[1,2], Jens Mehnert[1], and Bin Yang[2]

[1] Robert Bosch GmbH, Daimlerstrasse 6, 71229 Leonberg, Germany
{Rinor.Cakaj,JensEricMarkus.Mehnert}@de.bosch.com
[2] University of Stuttgart, Pfaffenwaldring 47, 70550 Stuttgart, Germany
bin.yang@iss.uni-stuttgart.de

Abstract. We introduce Mixture-of-Depths (MoD) for Convolutional Neural Networks (CNNs), a novel approach that enhances the computational efficiency of CNNs by selectively processing channels based on their relevance to the current prediction. This method optimizes computational resources by dynamically selecting key channels in feature maps for focused processing within the convolutional blocks (Conv-Blocks), while skipping less relevant channels. Unlike conditional computation methods that require dynamic computation graphs, CNN MoD uses a static computation graph with fixed tensor sizes which improve hardware efficiency. It speeds up the training and inference processes without the need for customized CUDA kernels, unique loss functions, or fine-tuning. CNN MoD either matches the performance of traditional CNNs with reduced inference times, GMACs, and parameters, or exceeds their performance while maintaining similar inference times, GMACs, and parameters. For example, on ImageNet, ResNet86-MoD exceeds the performance of the standard ResNet50 by 0.45% with a 6% speedup on CPU and 5% on GPU. Moreover, ResNet75-MoD achieves the same performance as ResNet50 with a 25% speedup on CPU and 15% on GPU.

Keywords: CNN, Mixture-of-Depths, Computational Efficiency, Inference Speed

1 Introduction

Over recent years, convolutional neural networks (CNNs) have demonstrated significant advancements in a variety of computer vision applications, such as image recognition [10,32], object detection [26,27], and image segmentation [22,41]. Despite their remarkable performance, CNNs often require substantial computational power and extensive memory usage, posing considerable challenges when deploying advanced models on devices with limited computational resources [18].

To adress these challenges, pruning techniques are widely utilized to reduce the model size and computational demands of CNNs by removing redundant weights or filters according to established criteria [11–13, 19, 23, 38, 42, 43]. However, these methods uniformly process all inputs, failing to adjust for the varying complexities of different inputs, which can lead to performance decreases [18].

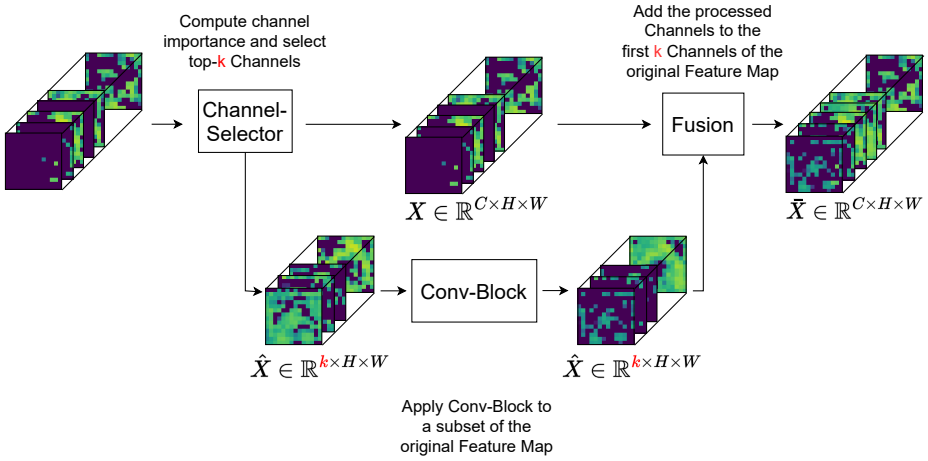


Fig. 1: Illustration of the CNN MoD mechanism, which starts with the Channel Selector module. This module computes the importance scores of each channel in the input feature map, $X \in \mathbb{R}^{C\times H\times W}$, and selects the top-$k$ channels for focussed processing in the Conv-Block. These selected channels are then processed by a Conv-Block designed to operate on a reduced dimension, $\hat{X} \in \mathbb{R}^{k\times H\times W}$, enhancing computational efficiency. The processed channels are added to the first k channels of the original feature map through a fusion operation, instead of being added back to their original positions. The resulting feature map is denoted by $\bar{X}$. This selective reintegration of refined channels with the unprocessed channels helps to preserve the dimensions of the original feature map ($X \in \mathbb{R}^{C\times H\times W}$).

An alternative approach is dynamic computing, or conditional computation, which adapts computational resources to the complexity of inputs to enhance efficiency [18,34,35]. However, the integration of these methods into hardware is challenging due to their reliance on dynamic computation graphs, which are often incompatible with systems optimized for static computation workflows [7,35]. For instance, Wu et al. [39] report increased processing times when layers are conditionally executed via a separate policy network. Despite theoretical reductions in computational demands, practical gains on hardware like GPUs or FPGAs are limited, as non-uniform tasks can interfere with the efficiencies of standard convolution operations [17,31]. Moreover, Ma et al. [24] show that floating point operations (FLOPS) are insufficient for estimating inference speed, as they often exclude element-wise operations like activation functions, summations, and pooling.

To combine the performance benefits of dynamic computing with the operational efficiency of static computation graphs, we present the CNN Mixture-of-Depths (MoD), inspired by the Mixture-of-Depths approach for Transformers [25]. The fundamental principle of CNN MoD is that only a selected subset of channels within the feature maps is essential for effective convolutional process-

ing at given layers in the network. By focusing on these essential channels, CNN MoD boosts computational efficiency without reducing network performance.

The MoD mechanism, shown in Figure 1, enhances CNN feature map processing by dynamically selecting the most important channels for focused computation within Conv-Blocks. MoD begins with the Channel Selector, illustrated in Figure 2, which is designed similarly to the Squeeze-and-Remember block [14]. It evaluates the importance of each channel within the input feature map $X \in \mathbb{R}^{C \times H \times W}$. Adaptive average pooling first reduces each channel to $\tilde{X} \in \mathbb{R}^{C \times 1 \times 1}$, which is then processed by a two-layer fully connected neural network with a bottleneck design. A sigmoid activation function generates scores indicating the importance of each channel.

Following this, the Channel Selector uses a top-k selection mechanism to identify the k most crucial channels based on the computed importance scores. These channels are sent to the Conv-Block, designed to operate on the reduced dimension $\hat{X} \in \mathbb{R}^{k \times H \times W}$, thus improving computational efficiency. To ensure that the gradients from the processed output $\hat{X}$ effectively optimize the channel selection process, the processed channels are scaled by their respective importance scores. This allows the gradients to flow back to the Channel Selector, enabling the learning of channel importance throughout training.

The final step involves a fusion operation, in which the processed channels are added to the first k channels of the original feature map. This fusion not only preserves the original dimensions of the feature map but also enhances feature representation by combining processed and unprocessed channels.

The reduction in the number of processed channels within Conv-Blocks is controlled by a hyperparameter $c \geqslant 1$, where $k = \lfloor \frac{C}{c} \rfloor$ defines the number of channels to process. Here, C is the total number of input channels in the Conv-Block. For example, in a typical ResNet [10] architecture, a bottleneck block that usually processes 1024 channels will only process 16 channels ($k = 16$) when $c = 64$. This selective processing significantly reduces the computational load by focusing on a smaller subset of channels. Additionally, the sizes of the kernels within the Conv-Blocks are adjusted to match the reduced number of input channels, as detailed in Section 3.2.

Empirical evaluations indicate that the optimal integration of MoD within the CNN architecture involves alternating them with standard Conv-Blocks. In architectures like ResNets [10] or MobileNetV2 [29], Conv-Blocks are organized into modules containing multiple Conv-Blocks of the same type (i.e., with the same number of output channels). Each module begins with a standard block, and is then followed by a MoD Block. This alternating arrangement is based on the principles of the Mixture-of-Depths for Transformers [25]. This does not imply that additional MoD Blocks are added to the existing sequence, but rather every second Conv-Block in the original architecture is replaced by a MoD Block, ensuring the overall depth of the architecture remains unchanged.

CNN MoD achieves performance comparable to traditional CNNs but with reduced inference times, GMACs, and parameters, or it surpasses them while maintaining similar inference times, GMACs, and parameters.

2 Related Work

This section reviews two strategies for enhancing CNN computational efficiency: static pruning and dynamic computing.

2.1 Static Pruning

Static pruning techniques aim to reduce the computational burden of CNNs by eliminating redundant model parameters. Early work in this area primarily targeted weight pruning, which selectively removes individual weights that have little effect on the model's output [8,9,37]. However, these methods tend to create irregular sparsity patterns that do not align well with hardware optimizations [18]. More recently, structured pruning approaches like channel pruning have gained prominence. These methods offer a more hardware-compatible form of sparsity by discarding entire channels based on their assessed importance [12,20]. For example, FPGM, or Filter Pruning via Geometric Median, identifies and removes filters that are closest to the geometric median within a layer, targeting those considered less crucial [12]. Similarly, HRank evaluates filters based on the rank of their generated feature maps, pruning those that contribute the least to the output's information content [20].

Despite their benefits, these static pruning methods permanently remove computations, potentially reducing the model's capacity to represent complex features. This permanent reduction can be particularly limiting for complex images that may require more detailed processing, suggesting that a one-size-fits-all approach to pruning might be suboptimal for diverse real-world applications. Additionally, these methods often require fine-tuning or specialized loss functions, further complicating their implementation.

2.2 Dynamic Computing

Dynamic computing, or conditional computation, dynamically adjusts computational resources according to the complexity of the input, potentially maintaining high model accuracy while reducing computation. Architectures like BranchyNet [33] and MSDNet [15] implement early exits for simpler inputs to decrease average computational load. Furthermore, policy-driven methods such as BlockDrop [39] and GaterNet [3] dynamically decide which network blocks to execute using a policy network, which adapts in real-time to the input.

Further innovations in dynamic computing utilize gating functions to enable selective processing of Conv-Blocks. SkipNet [36] and ConvNet-AIG [34] dynamically skip the processing of whole Conv-Blocks based on the observation that individual blocks can be removed without interfering with other blocks in the ResNet [10]. Some methods [5, 35, 40] exploit spatial sparsity to reduce the computations, e.g. Verelst et al. [35] learn a pixel-wise execution mask for each block and only calculate on those specified locations. Dynamic Dual Gating [18] introduces another layer of complexity by identifying informative features along

spatial and channel dimensions, allowing the model to skip unimportant regions and channels dynamically during inference.

Many works on conditional execution primarily highlight reductions in theoretical complexity. However, in practical settings, the application of these dynamic methods often confronts hardware limitations due to their dependency on dynamic computation graphs. This reliance can lead to inefficiencies and extended execution times on hardware systems that are optimized for static computations [7, 17, 31, 39].

Recently, Raposo et al. [25] introduced the Mixture-of-Depths for Transformers, a method that selectively processes tokens within Transformer blocks to effectively reduce computational load. This innovative approach inspired us for our CNN MoD approach.

2.3 CNN MoD: Combining Static and Dynamic Advantages

The CNN MoD approach combines the benefits of static pruning and dynamic computing within a unified framework. Below are the key advantages of our approach:

- **Static Computation Graph:** CNN MoD retains a static computation graph, which enhances both training and inference time.
- **No Custom Requirements:** Does not require customized CUDA kernels, additional loss functions, or fine-tuning.
- **Dynamic Resource Allocation:** Dynamically allocates computational resources to channels based on their importance, effectively optimizing training and inference speeds.

3 Method: CNN Mixture-of-Depths

The term "Mixture-of-Depths" in our approach refers to the selective processing strategy where not every channel is processed in every convolutional block within the same module, resulting in varied processing depths. This strategy allows for some channels to be processed more frequently than others within the same module, thereby optimizing computational resources and enhancing feature extraction efficiency. This section presents the CNN Mixture-of-Depths approach. It consists of three main components:

1. **Channel Selector:** This component selects the top-k most important channels from the input feature map based on their relevance to the current prediction. This selection helps to focus computational resources effectively by processing only those channels that are crucial for the current task.
2. **Convolutional Block:** The selected channels are then processed in a ConvBlock. This block can be adapted from existing architectures such as ResNets [10] or ConvNext [21] and is designed to enhance the features of the selected channels. To ensure that the Channel Selector is optimized during training, the processed feature maps are multiplied by their importance scores at the end of this block.

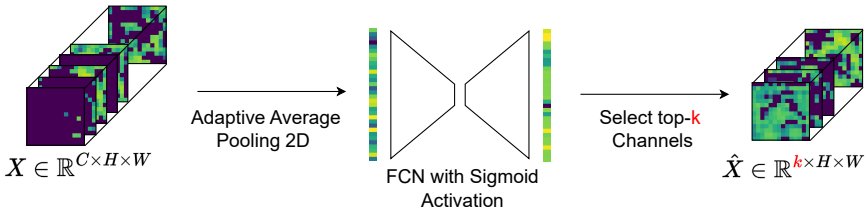


Fig. 2: Illustration of the Channel Selection Process in MoD. The process begins with an input tensor $X \in \mathbb{R}^{C \times H \times W}$, which undergoes adaptive average pooling to reduce spatial dimensions to 1×1, preserving channel information. The pooled output is processed through a two-layer fully connected network with a reduction factor ($r = 16$), followed by a sigmoid activation to generate channel-wise scores. These scores are used to select the top-k channels. This forms a subset of the original tensor with reduced channel dimension but original spatial dimensions.

3. **Fusion Operation:** After processing in the Conv-Block, the enhanced channels are integrated back into the original feature map. This is done by adding the processed channels to the first k channels of the feature map, which is then passed on to the subsequent layer without additional adjustments. This step ensures that the processed features are preserved and that the feature map maintains its original dimensions for further processing.

3.1 Channel Selector

Figure 2 shows how the Channel Selector dynamically selects the most significant channels for processing in the Conv-Block. It operates in two main stages:

1. **Adaptive Channel Importance Computation:** Initially, the Channel Selector compresses the input feature map $X \in \mathbb{R}^{C \times H \times W}$ using adaptive average pooling, reducing its dimension to $\tilde{X} \in \mathbb{R}^{C \times 1 \times 1}$. This compressed feature map is processed through a two-layer fully connected network with a bottleneck design, set to a ratio $r = 16$, and concludes with a sigmoid activation to create a score vector $\mathbf{s} \in \mathbb{R}^{C}$. Each element of $\mathbf{s}$ quantifies the importance of the corresponding channel.
2. **Top-k Channel Selection and Routing:** Utilizing the importance scores $\mathbf{s}$, the Channel Selector selects the top-k channels. These selected channels are routed to the Conv-Block for processing. The original feature map X is routed around the Conv-Block to the fusion process described in Section 3.3.

This selection process allows the Channel Selector to efficiently manage computational resources while maintaining a fixed computational graph, making it possible to dynamically choose which channels to process.

3.2 Channel Processing Dynamics

After the Channel Selector selects the top-k channels, they are processed within the Conv-Block. This block is adaptable from architectures such as ResNets [10] or ConvNext [21], and is designed to process a reduced number of channels.

The number of channels k processed in each Conv-Block is determined by the formula $k = \lfloor \frac{C}{c} \rfloor$, where C represents the total input channels of the block, and c is a hyperparameter determining the extent of channel reduction. For instance, in a standard ResNet [10] bottleneck block that typically processes 1024 channels, setting $c = 64$ reduces the processing to only 16 channels ($k = 16$). Unlike the standard bottleneck flow where channels transition from 1024 to 256 and back to 1024, in a MoD bottleneck block, the channel dimensions are significantly reduced from 16 to 4 and then back to 16, focusing computational efforts on the most essential features and enhancing efficiency. In our empirical investigations, we found that the hyperparameter c should be set to the maximal number of input channels in the first Conv-Block and remain the same in every MoD Block throughout the CNN. For example, $c = 64$ for ResNet [10] (see Supplementary Material) and $c = 16$ for MobileNetV2 [29].

The final step in the Conv-Block involves multiplying the processed channels with the importance scores obtained from the Adaptive Channel Importance Computation. This step ensures that gradients are effectively propagated back to the Channel Selector during training, which is needed for optimizing the selection mechanism.

3.3 Fusion Mechanism

The Fusion Mechanism combines processed and unprocessed channels to maintain the dimensionality required for subsequent convolutional operations. This approach ensures that the remaining non-selected channels, which contain useful features for later stages, are preserved, thereby maintaining a comprehensive feature set necessary for the model's performance.

Consider the original input feature map X with dimensions $\mathbb{R}^{C \times H \times W}$, where C represents the total number of channels, and H and W denote the height and width of the feature map, respectively. The processed feature map $\hat{X}$, resulting from the adapted Conv-Block applied to the selected top-k channels, has dimensions $\mathbb{R}^{k \times H \times W}$. To integrate $\hat{X}$ with X, the fusion operation is performed as follows:

$$\bar{X}[:, : k, :] = \hat{X} + X[:, : k, :], \tag{1}$$

$$\bar{X}[:, k :, :] = X[:, k :, :]. \tag{2}$$

where (1) adds processed features to the first k channels of X, and (2) maintains the remaining unprocessed channels. This formula confirms that $\bar{X}$, the feature map after fusion, has the same number of channels C as the original input X, preserving the dimensions needed for subsequent layers.

In our experiments, we tested various strategies for integrating the processed channels back into the feature map X. These strategies included adding the

processed channels back to their original positions (see Supplementary Material). However, empirical results did not show any improvements. It seems beneficial for the network's performance to consistently use the same positions within the feature maps for processed information. This observation is confirmed by experiments where the processed channels were added to the last k channels of the feature map X, yielding results comparable to those achieved when added to the first k channels (see Supplementary Material).

3.4 Integration in CNN Architecture

MoD can be integrated into various CNN architectures, such as ResNets [10], ConvNext [21], VGG [30], and MobileNetV2 [29]. These architectures are organized into modules containing multiple Conv-Blocks of the same type (i.e., with the same number of output channels). Through our experiments, we found out that alternating MoD Blocks with standard Conv-Blocks in each module is the most effective integration method. We also explored using MoD Blocks in every block and selectively within specific modules, however, the alternating strategy proved to be the most effective approach. It is important to note that MoD Blocks replace every second Conv-Block, maintaining the original architecture's depth (e.g., 50 layers in a ResNet50 [10]). Each module starts with a standard block, such as a BasicBlock [10], followed by an MoD Block. This alternating pattern indicates that the network can handle substantial capacity reductions, provided that there are regular intervals of full-capacity convolutions. Furthermore, this method ensures that MoD Blocks do not interfere with the spatial dimension-reducing convolutions that typically occur in the first block of each module.

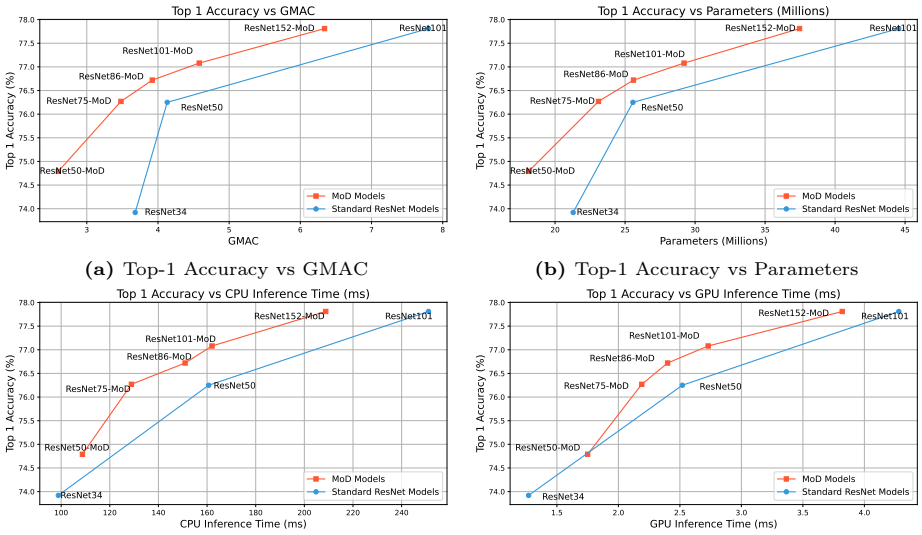
4 Experiments

This section evaluates the CNN Mixture-of-Depths on various tasks, including supervised image recognition on CIFAR-10/100 [16] and ImageNet [28], semantic segmentation on Cityscapes [4], and object detection on Pascal VOC [6]. Results for CIFAR-10/100 are detailed in the Supplementary Material.

Performance benchmarks were carried out on both GPU and CPU platforms to evaluate the running times. These measurements were taken using the Timer and Compare functions from the `torch.utils.benchmark` package, with tests performed on an Intel(R) Core(TM) i7-10850H CPU @ 2.70GHz and a NVIDIA Quadro RTX 3000 GPU.

4.1 Image Recognition on ImageNet

We evaluated the CNN MoD on the ImageNet dataset [28], containing 1.2 million training images, 50,000 validation images, and 150,000 test images across 1,000 categories. The experiments were performed using three different random



(a) Top-1 Accuracy vs GMAC

(b) Top-1 Accuracy vs Parameters

(c) Top-1 Accuracy vs CPU Inference Time

(d) Top-1 Accuracy vs GPU Inference Time

Fig. 3: ResNet MoD models outperform standard ResNets under similar computational constraints, as shown across four panels. Panel (a) shows the higher accuracy per GMAC, highlighting better computational efficiency, while panel (b) illustrates the improved parameter efficiency. Panels (c) and (d) demonstrate ResNet MoD's superior top-1 accuracy with comparable or faster inference times on CPU and GPU.

seeds to ensure the robustness and reproducibility of the results. Implementation details and further results are provided in the Supplementary Material.

Table 1 presents the comparative results for various network configurations, highlighting our approach against advanced pruning and dynamic computing strategies. This includes comparisons with established methods such as DGNet [18], Batch-Shaping [1], ConvNet-AIG [34], HRANK [20], FPGM [12], and DynConv [35]. We report for all models the top-1 classification accuracy on the validation set, along with attributes such as no requirement for fine-tuning (No FT), simple training mechanisms (Simple Train.), the availability of a GPU implementation (GPU Impl.), a static computational graph (Static Graph), and enhanced training speed (Faster Train.). Here, "Simple Training" indicates that no customized loss functions or complex hyperparameter tuning is required, and "Faster Training" is determined by measuring the relative training time compared to standard models.

The MoD Blocks reduce the computational load in every second block, allowing the construction of deeper architectures with the computational costs comparable to less deep standard models. Specifically, in the ResNet86-MoD, the number of Conv-Blocks in the third layer is 18, compared to just 6 blocks in the standard ResNet50, while maintaining nearly the same GMACs, parameters, and inference times. This expansion is enabled by the MoD Blocks that

process fewer channels. Similarly, the ResNet75-MoD increases the number of Conv-Blocks in the third layer to 14. For further details, see Supplementary Material.

As described in Section 3.2, the value of c in all MoD Blocks is set to the maximum number of input channels in the first Conv-Block. For instance, $c = 64$ for ResNet [10] and $c = 16$ for MobileNetV2 [29].

The MoD models achieve performances comparable to traditional CNNs but with reduced inference times or they surpass them while maintaining similar inference times. For instance, our ResNet75-MoD matches the performance of the standard ResNet50 and achieves an 15% speed-up on GPU and 25% on CPU. Similarly, the ResNet50-MoD model significantly enhances processing speeds, achieving a 27% improvement on CPU and 48% on GPU, with a slight impact on accuracy. Furthermore, the ResNet86-MoD model improves accuracy by 0.45% with a speed-up of 6% on CPU and 5% on GPU. A test variant, MoD50 rand. using randomized channel selection, shows a performance decline of about 0.8%. This highlights the significance of strategic channel selection in our MoD approach. Additionally, as shown in the Supplementary Material, the MoD approach does not increase performance variance, maintaining consistency comparable to standard ResNet models. Figure 3 presents a comparative performance analysis of ResNet MoD models against standard ResNets for different computational metrics.

Dynamic approaches like DGNet [18] and ConvNet-AIG [34], which lack data on real-world inference speed-up, show simulated GMAC reductions only, indicating potential discrepancies in practical applications.

The results presented in Table 2 demonstrate the effectiveness of applying the MoD approach to MobileNetV2 architectures. The MobileNetV2-MoD-L model, with a deeper configuration (see Supplementary Material), maintains nearly the same top-1 accuracy as the standard MobileNetV2, while achieving a 11% speed-up on CPU and 10% on GPU. The standard MobileNetV2-MoD, despite a slight decrease in accuracy, offers a significant speed-up of 43% on CPU and 39% on GPU, illustrating the benefits of the MoD approach in balancing performance with computational efficiency.

4.2 Semantic Segmentation on Cityscapes

For semantic segmentation, we utilized the Cityscapes dataset [4], which contains high-quality, finely annotated images from 50 European cities, divided into 19 semantic classes (2,975 training, 500 validation, and 1,525 test images). The experiments were performed using three different random seeds. Implementation details are provided in the Supplementary Material.

Table 3 presents the performance of Fully Convolutional Networks (FCNs) with ResNet-based MoD backbones for semantic segmentation on the Cityscapes validation dataset. The FCN with a ResNet86-MoD backbone matches the inference times of the standard FCN-R50 model while enhancing the mean Intersection over Union (mIoU) by 0.95%, demonstrating MoD's ability to boost segmentation accuracy without additional computational costs. The FCN-R75-MoD

Table 1: This table presents the MoD models against other dynamic pruning and computational efficiency strategies. The table evaluates each method based on top-1 accuracy (Acc %), computational complexity (GMAC), model size (Params, in millions - M) and speed-up for CPU and GPU (Speed-up CPU, Speed-up GPU). Models abbreviated as "R50" refer to ResNet50 architectures. Notably, only MoD models incorporate all the attributes such as No Fine-Tuning (No FT), Simple Training (Simple Train.), GPU Implementation (GPU Impl.), a Static Computational Graph (Static Graph), and Enhanced Training Speed (Faster Train.).

Model	No FT	Simple Train.	GPU Impl.	Static Graph	Faster Train.	Top-1 Acc (%)	GMAC	Params (M)	Inference (ms) CPU	Inference (ms) GPU	Speed-up CPU	Speed-up GPU
ResNet152-MoD	✓	✓	✓	✓	✓	77.81	6.34	37.46	208.80	3.82	1.20	1.12
ResNet101						77.81	7.80	44.55	251.05	4.28	1.00	1.00
ResNet101-MoD	✓	✓	✓	✓	✓	77.08	4.58	29.21	162.03	2.73	1.55	1.57
ResNet86-MoD	✓	✓	✓	✓	✓	76.72	3.92	25.60	150.96	2.40	1.06	1.05
R50-DGNet [18]	✓	✗	✗	✗	✗	76.41	1.65*	29.34	—[1]	—[1]	—	—
ResNet75-MoD	✓	✓	✓	✓	✓	76.27	3.48	23.10	128.90	2.19	1.25	1.15
ResNet50						76.25	4.13	25.56	160.66	2.52	1.00	1.00
R50-Batch-Shaping [1]	✓	✗	✗	✗	✗	75.70	2.07*	15.31	—	—	1.31[2]	—
R50-ConvNet-AIG [34]	✓	✗	✗	✗	✗	75.45	2.59*	26.56	—	—	1.29[2]	—
R50-HRANK [20]	✗	✗	✓	✓	✗	74.98	2.30	16.15	—[3]	—[3]	—	—
ResNet50-MoD	✓	✓	✓	✓	✓	74.79	2.60	18.11	108.74	1.75	1.48	1.44
R50-FPGM [12]	✗	✗	✓	✓	✗	74.83	1.91	—	—	—	—	1.61[4]
R50-DynConv [35]	✓	✗	✓	✗	✗	74.40	2.25	25.56	—	—	—	—[5]
R50-FPGM [12]	✓	✗	✓	✓	✗	74.13	1.91	—	—	—	—	1.61[4]
MoD50 rand.	✓	✓	✓	✓	✓	74.02	2.59	17.11	100.06	1.43	1.77	1.43

[1] No CPU/GPU speed-up given for ResNet50.
[2] CPU speed-ups for Batch-Shaping, ConvNet-AIG are from Batch-Shaping paper.
[3] HRANK lacks implementation to compute CPU/GPU inference.
[4] Only GPU speed-up reported for FPGM.
[5] GPU speed-up for DynConv depends on a custom CUDA kernel.
* GMAC values are simulated estimates.

provides similar accuracy as standard models but with computational efficiency improvements, achieving a 9% speed-up on CPU and 7% on GPU.

4.3 Object Detection on Pascal VOC

For our object detection experiments, we utilized the PASCAL VOC dataset, employing the mmobjectdetection library [2] to configure and train our models. The experiments were performed using three different random seeds. Detailed implementation notes are provided in the Supplementary Material.

The outcomes of our experiments with Faster R-CNN models on the PASCAL VOC dataset are presented in Table 4. The Faster-RCNN with a ResNet86-MoD backbone demonstrates the MoD approach's effectiveness by achieving a 0.37% improvement in mean Average Precision (mAP) and a 0.4% increase in Average Precision at 50% IoU threshold (AP50) compared to the standard Faster-RCNN-R50 model. Although GPU inference speed slightly decreases to 0.96x of the

Table 2: Evaluation of standard and modified MobileNetV2 models on ImageNet. This table details top-1 accuracy, computational complexity (MMAC), model size (Params, in millions), and inference performance on CPU and GPU.

Method	Top-1	MMAC	Params	Inference (ms)		Speed-up	
	Acc (%)		(M)	CPU	GPU	CPU	GPU
MobileNetV2	71.63	320.36	3.5	47.76	0.78	—	—
MobileNetV2-MoD-L	71.55	344.76	3.34	42.94	0.71	1.11	1.10
MobileNetV2-MoD	69.34	220.56	2.94	33.43	0.56	1.43	1.39

Table 3: Performance comparison of FCN models on the Cityscapes validation dataset, contrasting standard ResNet50 backbones against MoD versions. The table demonstrates that MoD models deliver performance comparable to or better than the standard FCN, with either reduced or similar inference times. Metrics include FLOPS (T), parameters (M), and inference durations (ms).

Model	Perf. Metrics (%)			FLOPS	Params	Inference (ms)		Speed-up	
	aAcc	mIoU	mAcc	(T)	(M)	CPU	GPU	CPU	GPU
FCN-R86-MoD	95.75	73.75	81.75	0.377	47.15	8272.40	202.65	1.01	1.02
FCN-R50	95.68	72.80	80.25	0.393	47.11	8389.35	206.30	—	—
FCN-R75-MoD	95.72	72.72	80.56	0.359	44.66	7687.20	192.50	1.09	1.07
FCN-R50-MoD	95.37	71.58	79.54	0.322	39.66	6757.20	166.50	1.24	1.24

baseline, CPU processing speed is enhanced, showing a 1.10x improvement over the standard model. Similarly, the Faster-RCNN-R75-MoD model maintains accuracy close to the baseline while boosting inference efficiency, achieving an 11% speed-up on CPU and 5% on GPU.

5 Channel Selection and Regularization Analysis

5.1 Channel Selector Mechanism

In this section, we investigate the practical operation of the Channel Selector within our CNN MoD approach. This analysis focuses on the Channel Selector within a MoD Block in the third module of a ResNet75-MoD. To provide a clearer understanding of the Channel Selector's behavior, we analyze how frequently channels are chosen in the top-k routing mechanism. We use five different classes from the ImageNet validation set: plane, truck, church, cliff, and pug. For each class, fifty samples are drawn from the validation set, and we count the number of times each channel is selected. The percentage of occurrences for each channel is plotted in Figure 4. For comparison, we also plot the percentages across all 1000 classes in Figure 4a. Our analysis shows that the Channel Selector selects

Table 4: Performance comparison of F-RCNN models on the Pascal VOC validation dataset, contrasting standard and ResNet-MoD backbones. The table shows that MoD variants match or exceed the standard F-RCNN's efficiency, achieving similar or better object detection performance with faster or equivalent inference times. Metrics include FLOPS (T), parameters (M), and inference durations (ms).

Model	Perf. Metrics (%)		FLOPS	Params	Inference (ms)		Speed-up	
	mAP	AP50	(T)	(M)	CPU	GPU	CPU	GPU
Faster-RCNN-R86-MoD	77.67	77.67	0.108	41.486	2813.9	83.5	1.10	0.96
Faster-RCNN-R50	77.30	77.27	0.111	41.446	3101.5	80.5	—	—
Faster-RCNN-R75-MoD	77.04	77.03	0.104	38.989	2784.5	76.4	1.11	1.05
Faster-RCNN-R50-MoD	75.10	75.10	0.949	33.994	2453.3	69.3	1.26	1.16

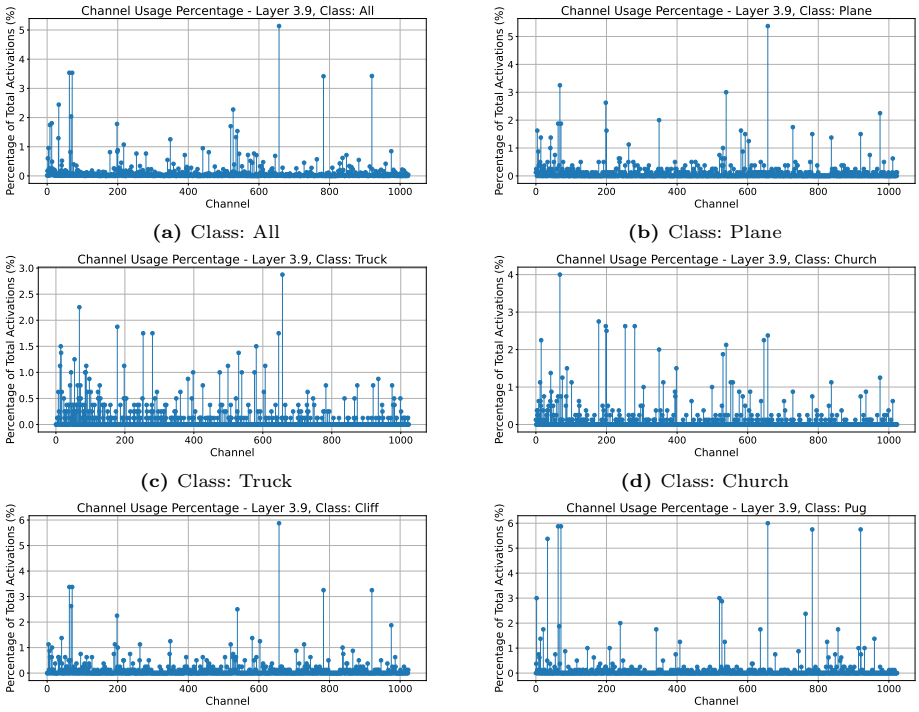
different channels for different classes, indicating that it adapts to enhance the detection of class-specific features. Additionally, some channels are selected more frequently across various classes, suggesting a potential for further computational load reduction through channel pruning techniques.

5.2 Regularization Effect of Selective Processing

An interesting byproduct of the CNN MoD approach is its regularization effect, caused by the selective processing of channels in the modified Conv-Blocks. Traditional CNN architectures typically process all channels within each Conv-Block, which can lead to learning overly specific features that may not generalize well to new data. In contrast, the MoD framework selectively reduces the number of channels processed, focusing on those most important ones for the current task. This selective processing naturally encourages the network to prioritize and refine features that are more likely to be generalizable across various images. As a result, MoD acts as a form of natural regularization, pushing the network to extract more useful information from a limited set of inputs channels and thereby promoting the learning of robust, broadly applicable features. This is particularly advantageous for preventing overfitting, a common challenge in deep learning models, without additional explicit regularization strategies.

6 Conclusion

In this work, we presented CNN MoD, an approach inspired by the Mixture-of-Depths method developed for Transformers [25]. This technique combines the advantages of static pruning and dynamic computing within a single framework. It optimizes computational resources by dynamically selecting key channels in feature maps for focused processing within the Conv-Blocks, while skipping less relevant channels. It maintains a static computation graph, which optimizes the training and inference speed without the need for customized CUDA kernels, additional loss functions, or fine-tuning. These attributes separate MoD from



(a) Class: All **(b)** Class: Plane

(c) Class: Truck **(d)** Class: Church

(e) Class: Cliff **(f)** Class: Pug

Fig. 4: Channel selection frequencies within the third module of a ResNet75-MoD for five diverse ImageNet classes: plane, truck, church, cliff, and pug. The analysis shows the percentage of times channels are selected by the Channel Selector out of the total selections in the layer, based on fifty samples per class. These percentages are compared to a baseline derived from all 1000 classes (4a), indicating that the Channel Selector selects different channels for different classes.

other dynamic computing methods, offering a significant simplification of both training and inference processes. CNN MoD achieves performance comparable to traditional CNNs but with reduced inference times, GMACs, and parameters, or it surpasses them while maintaining similar inference times, GMACs, and parameters in image recognition, semantic segmentation, and object detection.

While the fusion operation currently achieves significant efficiency gains, especially with high-dimensional datasets like ImageNet and real-world tasks such as Cityscapes and Pascal VOC, further optimization of the slicing operation could unlock even greater improvements. Future work will explore refining this component, potentially through a customized CUDA kernel. Further research into the optimal number of processed channels within layers is also promising for optimizing performance.

References

1. Bejnordi, B.E., Blankevoort, T., Welling, M.: Batch-shaping for learning conditional channel gated networks. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net (2020), https://openreview.net/forum?id=Bke89JBtvB
2. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open MMLab Detection Toolbox and Benchmark. arXiv preprint arXiv:1906.07155 (2019)
3. Chen, Z., Li, Y., Bengio, S., Si, S.: You Look Twice: GaterNet for Dynamic Filter Selection in CNNs. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 9172–9180. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.00939, http://openaccess.thecvf.com/content_CVPR_2019/html/Chen_You_Look_Twice_GaterNet_for_Dynamic_Filter_Selection_in_CNNs_CVPR_2019_paper.html
4. 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 (CVPR) (June 2016)
5. Dong, X., Huang, J., Yang, Y., Yan, S.: More is Less: A More Complicated Network with Less Inference Complexity. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017. pp. 1895–1903. IEEE Computer Society (2017). https://doi.org/10.1109/CVPR.2017.205, https://doi.org/10.1109/CVPR.2017.205
6. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision **88**(2), 303–338 (Jun 2010)
7. Graves, A.: Adaptive Computation Time for Recurrent Neural Networks. CoRR **abs/1603.08983** (2016), http://arxiv.org/abs/1603.08983
8. Guo, Y., Yao, A., Chen, Y.: Dynamic Network Surgery for Efficient DNNs. In: Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 1379–1387 (2016), https://proceedings.neurips.cc/paper/2016/hash/2823f4797102ce1a1aec05359cc16dd9-Abstract.html
9. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both Weights and Connections for Efficient Neural Network. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. pp. 1135–1143 (2015), https://proceedings.neurips.cc/paper/2015/hash/ae0eb3eed39d2bcef4622b2499a05fe6-Abstract.html
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90, https://doi.org/10.1109/CVPR.2016.90

11. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 2234–2240. ijcai.org (2018). https://doi.org/10.24963/IJCAI.2018/309, https://doi.org/10.24963/ijcai.2018/309
12. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019. pp. 4340–4349. Computer Vision Foundation / IEEE (2019). https://doi.org/10.1109/CVPR.2019.00447, http://openaccess.thecvf.com/content_CVPR_2019/html/He_Filter_Pruning_via_Geometric_Median_for_Deep_Convolutional_Neural_Networks_CVPR_2019_paper.html
13. He, Y., Zhang, X., Sun, J.: Channel Pruning for Accelerating Very Deep Neural Networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 1398–1406. IEEE Computer Society (2017). https://doi.org/10.1109/ICCV.2017.155, https://doi.org/10.1109/ICCV.2017.155
14. Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 7132–7141. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00745, http://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html
15. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-Scale Dense Networks for Resource Efficient Image Classification. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net (2018), https://openreview.net/forum?id=Hk2aImxAb
16. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
17. Lavin, A., Gray, S.: Fast Algorithms for Convolutional Neural Networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 4013–4021. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.435, https://doi.org/10.1109/CVPR.2016.435
18. Li, F., Li, G., He, X., Cheng, J.: Dynamic Dual Gating Neural Networks. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 5310–5319 (2021), https://api.semanticscholar.org/CorpusID:244477302
19. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning Filters for Efficient ConvNets. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=rJqFGTslg
20. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: HRank: Filter Pruning Using High-Rank Feature Map. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 1526–1535. Computer Vision Foundation / IEEE (2020). https://doi.org/10.1109/CVPR42600.2020.00160, https://openaccess.thecvf.com/content_CVPR_2020/html/Lin_HRank_Filter_Pruning_Using_High-Rank_Feature_Map_CVPR_2020_paper.html

21. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A ConvNet for the 2020s. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11966–11976 (2022). https://doi.org/10.1109/CVPR52688.2022.01167
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 3431–3440. IEEE Computer Society (2015). https://doi.org/10.1109/CVPR.2015.7298965, https://doi.org/10.1109/CVPR.2015.7298965
23. Luo, J., Wu, J., Lin, W.: ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 5068–5076. IEEE Computer Society (2017). https://doi.org/10.1109/ICCV.2017.541, https://doi.org/10.1109/ICCV.2017.541
24. Ma, N., Zhang, X., Zheng, H., Sun, J.: ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV. Lecture Notes in Computer Science, vol. 11218, pp. 122–138. Springer (2018). https://doi.org/10.1007/978-3-030-01264-9_8, https://doi.org/10.1007/978-3-030-01264-9_8
25. Raposo, D., Ritter, S., Richards, B.A., Lillicrap, T.P., Humphreys, P.C., Santoro, A.: Mixture-of-Depths: Dynamically allocating compute in transformer-based language models. CoRR **abs/2404.02258** (2024). https://doi.org/10.48550/ARXIV.2404.02258, https://doi.org/10.48550/arXiv.2404.02258
26. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 779–788. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.91, https://doi.org/10.1109/CVPR.2016.91
27. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada. pp. 91–99 (2015), https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y, https://doi.org/10.1007/s11263-015-0816-y
29. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00474, http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html
30. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference

on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556
31. Sze, V., Chen, Y., Yang, T., Emer, J.S.: Efficient Processing of Deep Neural Networks: A Tutorial and Survey. Proc. IEEE **105**(12), 2295–2329 (2017). https://doi.org/10.1109/JPROC.2017.2761740, https://doi.org/10.1109/JPROC.2017.2761740
32. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. pp. 1–9. IEEE Computer Society (2015). https://doi.org/10.1109/CVPR.2015.7298594, https://doi.org/10.1109/CVPR.2015.7298594
33. Teerapittayanon, S., McDanel, B., Kung, H.T.: BranchyNet: Fast inference via early exiting from deep neural networks. In: 23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016. pp. 2464–2469. IEEE (2016). https://doi.org/10.1109/ICPR.2016.7900006, https://doi.org/10.1109/ICPR.2016.7900006
34. Veit, A., Belongie, S.J.: Convolutional Networks with Adaptive Inference Graphs. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11205, pp. 3–18. Springer (2018). https://doi.org/10.1007/978-3-030-01246-5_1, https://doi.org/10.1007/978-3-030-01246-5_1
35. Verelst, T., Tuytelaars, T.: Dynamic Convolutions: Exploiting Spatial Sparsity for Faster Inference. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020. pp. 2317–2326. Computer Vision Foundation / IEEE (2020). https://doi.org/10.1109/CVPR42600.2020.00239, https://openaccess.thecvf.com/content_CVPR_2020/html/Verelst_Dynamic_Convolutions_Exploiting_Spatial_Sparsity_for_Faster_Inference_CVPR_2020_paper.html
36. Wang, X., Yu, F., Dou, Z., Darrell, T., Gonzalez, J.E.: SkipNet: Learning Dynamic Routing in Convolutional Networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII. Lecture Notes in Computer Science, vol. 11217, pp. 420–436. Springer (2018). https://doi.org/10.1007/978-3-030-01261-8_25, https://doi.org/10.1007/978-3-030-01261-8_25
37. Wimmer, P., Mehnert, J., Condurache, A.: COPS: Controlled Pruning Before Training Starts. In: International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021. pp. 1–8. IEEE (2021). https://doi.org/10.1109/IJCNN52387.2021.9533582, https://doi.org/10.1109/IJCNN52387.2021.9533582
38. Wimmer, P., Mehnert, J., Condurache, A.: Interspace Pruning: Using Adaptive Filter Representations to Improve Training of Sparse CNNs. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 12517–12527. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.01220, https://doi.org/10.1109/CVPR52688.2022.01220
39. Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R.S.: BlockDrop: Dynamic Inference Paths in Residual Networks. In: 2018 IEEE

Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 8817–8826. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00919, http://openaccess.thecvf.com/content_cvpr_2018/html/Wu_BlockDrop_Dynamic_Inference_CVPR_2018_paper.html

40. Xie, Z., Zhang, Z., Zhu, X., Huang, G., Lin, S.: Spatially Adaptive Inference with Stochastic Feature Sampling and Interpolation. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J. (eds.) Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12346, pp. 531–548. Springer (2020). https://doi.org/10.1007/978-3-030-58452-8_31, https://doi.org/10.1007/978-3-030-58452-8_31
41. Yu, F., Koltun, V.: Multi-Scale Context Aggregation by Dilated Convolutions. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.07122
42. Yu, R., Li, A., Chen, C., Lai, J., Morariu, V.I., Han, X., Gao, M., Lin, C., Davis, L.S.: NISP: Pruning Networks Using Neuron Importance Score Propagation. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 9194–9203. Computer Vision Foundation / IEEE Computer Society (2018). https://doi.org/10.1109/CVPR.2018.00958, http://openaccess.thecvf.com/content_cvpr_2018/html/Yu_NISP_Pruning_Networks_CVPR_2018_paper.html
43. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware Channel Pruning for Deep Neural Networks. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada. pp. 883–894 (2018), https://proceedings.neurips.cc/paper/2018/hash/55a7cf9c71f1c9c495413f934dd1a158-Abstract.html