Fire Together Wire Together:
A Dynamic Pruning Approach with Self-Supervised Mask Prediction

Sara Elkerdawy[1]   Mostafa Elhoushi[2]   Hong Zhang[1]
Nilanjan Ray[1]
[1]University of Alberta, [2]Toronto Heterogeneous Compilers Lab, Huawei
{elkerdaw, hzhang, nray1}@ualberta.ca

Abstract

Dynamic model pruning is a recent direction that allows for the inference of a different sub-network for each input sample during deployment. However, current dynamic methods rely on learning a continuous channel gating through regularization by inducing sparsity loss. This formulation introduces complexity in balancing different losses (e.g task loss, regularization loss). In addition, regularization based methods lack transparent tradeoff hyperparameter selection to realize a computational budget. Our contribution is two-fold: 1) decoupled task and pruning losses. 2) Simple hyperparameter selection that enables FLOPs reduction estimation before training. Inspired by the Hebbian theory in Neuroscience: "neurons that fire together wire together", we propose to predict a mask to process k filters in a layer based on the activation of its previous layer. We pose the problem as a self-supervised binary classification problem. Each mask predictor module is trained to predict if the log-likelihood for each filter in the current layer belongs to the top-k activated filters. The value k is dynamically estimated for each input based on a novel criterion using the mass of heatmaps. We show experiments on several neural architectures, such as VGG, ResNet and MobileNet on CIFAR and ImageNet datasets. On CIFAR, we reach similar accuracy to SOTA methods with 15% and 24% higher FLOPs reduction. Similarly in ImageNet, we achieve lower drop in accuracy with up to 13% improvement in FLOPs reduction.

1. Introduction

Convolutional Neural Networks (CNNs) showed unprecedented growth over the past decade which represented the state-of-the-art in many fields. However, CNNs require substantially large computation and memory consumption which limits deployment on edge and embedded platforms. There are many advances in model compression research
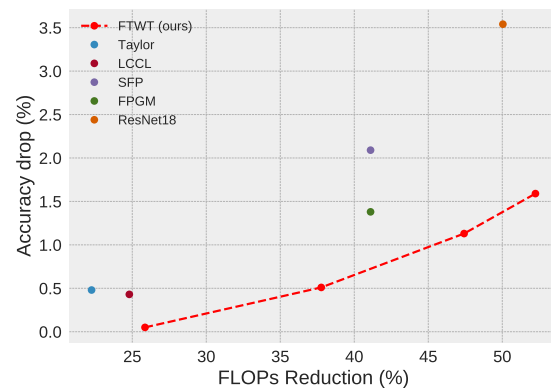

Figure 1. FLOPs reduction vs accuracy drop from baselines for various dynamic and static models on ResNet34 ImageNet.

including manually designed lightweight models [16, 17], low-bit precision [23, 47], architecture search [4, 39], and model pruning [9,21,29,34]. Most of the compression techniques are agnostic to the input data and optimize for a *static* efficient model. Recent efforts in pruning literature propose to keep the backbone for the baseline model as a whole and do inference using different sub-networks conditioned on the input. This is known as dynamic pruning, where different routes are activated based on the input which allows higher degree of freedom and more flexibility in comparison to static pruning.

Current dynamic pruning approaches typically introduce a regularization term to induce sparsity over a continuous parameter for channel gating/masking [6, 10, 40]. Others adopt policy gradient introduced in reinforcement learning [42] to learn different routes. These methods require careful tuning in training to tackle issues such as training stability with schedule annealing [40], biased training handling [18], or predefined pruning ratio per layer [10, 25]. Also, as noted in [6], additional sparsity loss degrades task loss as it is difficult to balance the task loss and the pruning

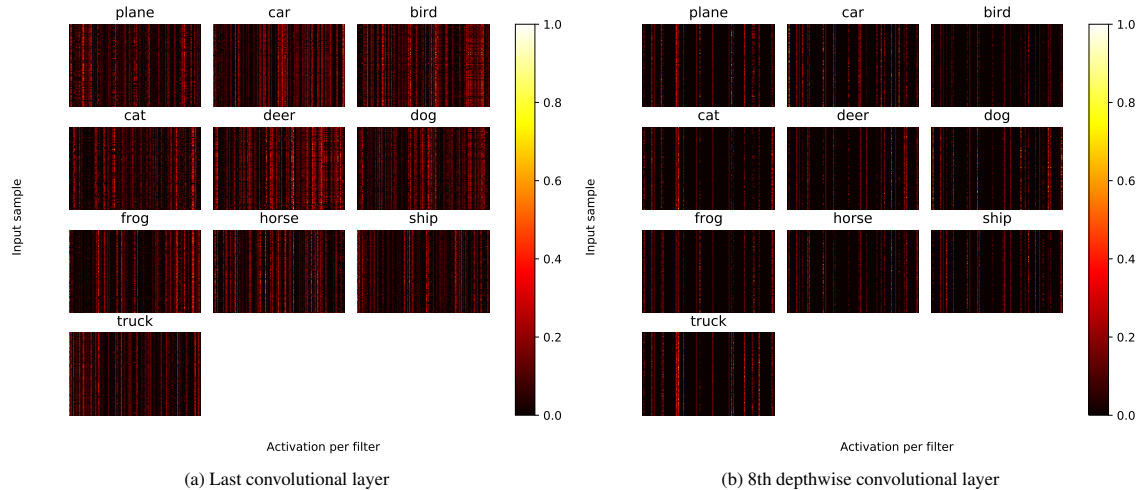| | (a) Last convolutional layer | (b) 8th depthwise convolutional layer |

Figure 2. Maximum activations in all features at the last convolutional layer and a middle layer in mobilenetv1 CIFAR-10. Each row in a subplot represents an input sample. Samples that belong to the same class activate same group of filters. Better visualized in color.

loss especially under high pruning ratio as shown in Figure 1. Moreover, the FLOPs reduction of these dynamic methods is dependent on the target sparsity preset hyperparameter. This hyperparameter selection lacks transparent relation between sparsity hyperparameter and the reached FLOPs; thus, hinders practical efficient training with many iterations of trial and error to achieve a target FLOPs reduction.

In this paper, we tackle these issues by formulating the problem as a self-supervised binary classification task. We generate the binary mask of the current layer (wiring) based on the activation (firing) of the previous layer. We draw inspiration from the Hebbian theory [31] in Neuroscience with a twist that we enforce this wiring-firing relation instead of a study of causation as in the theory. Figure 2 plots the maximum response for each filter (x-axis) of the last convolutional layer and a middle layer of MobileNet-V1 for random input samples (y-axis) grouped by their class. The plot shows that samples that belong to the same class tend to activate the same combination of filters and thus we only need to process a handful of the filters. It is worth noting that the number of clusters vary per layer. Similar to other dynamic pruning methods, we learn a decision head for channel gating. However, we learn the gating using binary cross entropy loss per channel. Each layer predicts the filters which are most likely to be highly activated given the layer's input activations. We generate ground truth binary masks per layer based on the mass of the heatmap per sample. This formulation provides advantages in two aspects. First, the channel gating loss implicitly complies and adapts to the backbone's status which stabilizes training in comparison to the case with sparsity regularization or RL-based training. Second, reduction in FLOPs can be esti-

mated before training, as the target mask is controlled by the generated ground truth mask which gives an estimate on the reduction. This simplifies the hyperparameter selection that controls pruning ratio. The main contributions are summarized as follows:

- A novel loss formulation with self-supervised ground truth mask generation that is stochastic gradient descent (SGD) friendly with no gradient weighting tricks.

- We propose a novel dynamic signature based on the heatmap mass without a pre-defined pruning ratio per layer.

- Simple hyperparameter selection that enables FLOPs reduction estimation before training. This simplifies realizing a prior budget target with bounded hyperparameter search space.

2. Related Work

Static Pruning. Static pruning removes weights in the offline training stage and apply the same compressed model to all samples. Unstructured pruning [9, 11, 30, 37] targets removing individual weights with minimal contribution. The limitation of the unstructured weight pruning is that dedicated hardware and libraries [37] are needed to achieve speedup from the compression. Structured pruning is becoming a more practical solution where filters or blocks are ranked and pruned based on a criterion [7, 13, 15, 27, 32, 34]. Earlier filter-pruning methods [22, 32] require calculation of layer-wise sensitivity analysis to generate the model signature (i.e number of filters per layer) before pruning. Sensitivity analysis is computationally expensive, especially as
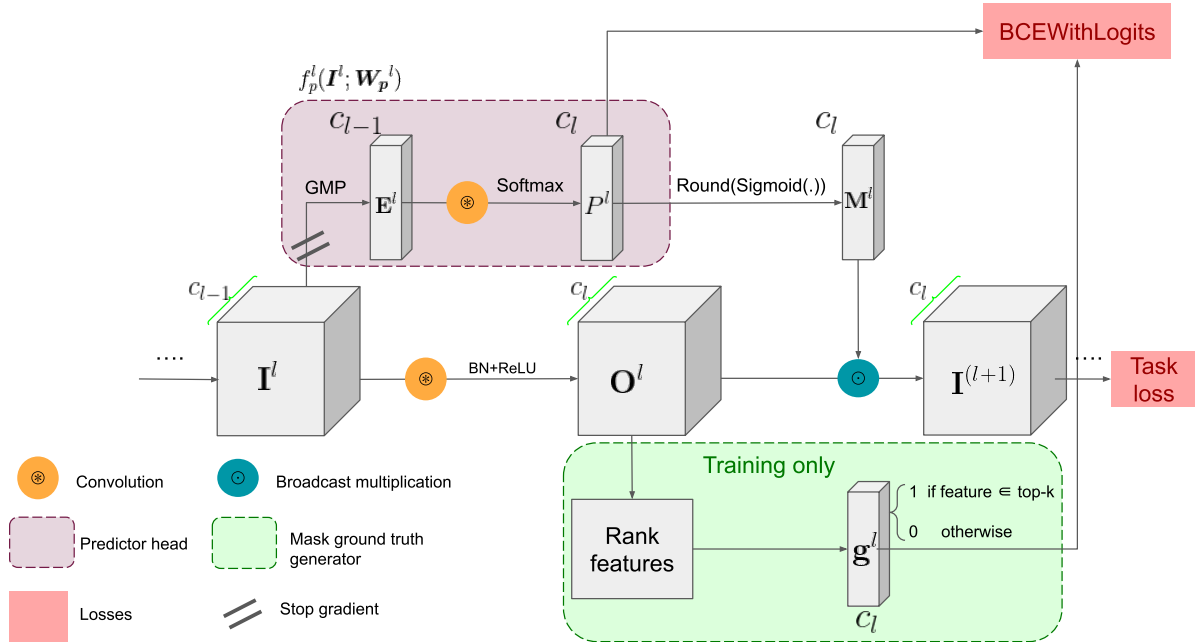
Figure 3. Proposed pipeline for training dynamic routing for one layer. For a layer l, prediction head $f_p^l(\boldsymbol{I}^l; \boldsymbol{W_p}^l)$ takes an input $\boldsymbol{I}^l$, applies global max pooling (GMP), normalizes with Softmax, then feeds to 1x1 convolution to generate logits $\boldsymbol{P^l}$ for the binary mask $\boldsymbol{M^l}$. Binary Cross Entropy (BCEWithLogits) loss penalizes the mask prediction based on the top-k obtained from the unpruned feature maps $\boldsymbol{O}^l$.

models grow deeper. Recent methods [28, 34, 41] learn a global importance measure. Molchanov *et al.* [34] propose a Taylor approximation on network's weights where the filter's gradients and norm are used to approximate its global importance score. Liu *et al.* [28] and Wen *et al.* [41] introduce a sparsity loss in addition to the task loss as a regularization then prune filters whose criterion are less than a threshold.

Dynamic Pruning. In contrast to one-model-fits-all deployment as in static pruning, dynamic pruning processes different routes per input sample. Similar to static pruning, methods can adopt different granularity to prune. Channel gating network (CGNet) [18] is a fine-grained method that skips zero locations in feature maps. A subset of input channels are processed by each kernel. The decision gating is learnt through regularization with a complex approximating non-differentiable function. They adopt group convolution and shuffle operation to balance filters frequency updates from different group of features. Closest to our work, we focus on dynamic filter pruning methods. In Runtime Neural Pruning (RNP) [25], a decision unit is modeled as a global recurrent layer, which generates discrete actions corresponding to four preset channel selection groups. The group selection is trained with reinforcement learning. Similarly, in BlockDrop [43], a policy network is trained to skip blocks in residual networks instead of only channels. D²NN [26] defines a variant set of conditional branches in

DNN, and uses Q-learning to train the branching policies. These methods train their policy functions by reinforcement learning which can be a non-trivial costly optimization task along with the CNN backbone. Feature Boosting and Suppression (FBS) [10] method generates continuous channel saliency and uses a predefined pruning ratio per layer to obtain a discrete binary gating. LCS [40] propose to obtain a discrete action from N learned group of channels sampled from Gumbel distribution. They adopt annealing temperature to stabilize training and introduce diversity in the learned routes by gradient normalization trick. These existing regularization-based methods require additional careful tuning to stabilize training that is either caused by policy gradients or to enforce learning diverse routes as to not converge to a static pruning. Hence, we propose to formulate the channel selection as a self-supervised binary classification in which interpretable routes can be studied and simply trained with common SGD.

3. Methodology

In this section we first explain the mechanism for dynamic gating. Then, we discuss how we design the decision heads and the supervised loss

3.1. Channel Gating

Let $\boldsymbol{I^l}$, $\boldsymbol{W^l}$ be the input features, and weights of a convolution layer l, respectively, where $\boldsymbol{I^l} \in \mathbb{R}^{c_{l-1} \times w_l \times h_l}$,

$W^l \in \mathbb{R}^{c_l \times c_{l-1} \times k_l \times k_l}$, and c_l is the number of filters in layer l. A typical CNN block consists of a convolution operation ($*$), batch normalization (BN), and an activation function (f) such as the commonly used ReLU. Without loss of generality, we ignore the bias term because of BN inclusion, thus, the output feature map O^l can be written as $O^l = f(\text{BN}(I^l * W^l))$. We predict a binary mask $M^l \in \mathbb{R}^{c_l}$ denoting the highly activated output feature maps O^l from the input activation map I^l by applying a decision head f_p^l with learnable parameters W_p^l. Masked output I^{l+1} is then represented as $I^{l+1} = O^l \odot Binarize(f_p^l(I^l; W_p))$. $Binarize(.)$ function is $round(Sigmoid(.))$ to convert logits to a binary mask. The prediction of the highly activated output feature maps allows for processing filters f where $M_f^l = 1$ in the inference time and skipping the rest. Our decision head has $c_{l-1} \times c_l$ FLOPs cost per layer l which is negligible.

3.2. Self-Supervised Binary Gating

Our proposed method as shown in Figure 3 learns this dynamic routing in a self-supervised way by inserting a predictor head after each convolutional block to predict the highly k activated filters of the next layer. The k value is automatically calculated per input based on the mass of heatmap.

Loss function   The ground truth binary mask of the highly activated features is attainable by sorting the norm of the features. The overall training objective is:

$$\min_{\{W, W_p\}} L_{total} = L_{ent}(f_n(\boldsymbol{x}; W), \boldsymbol{y}_k) \\ + L_{pred}(\{f_p^l(I^l; W_p{}^l), g^l\}_L) \quad (1)$$

where f_n is the backbone of the baseline model, L_{ent} is the cross-entropy task loss, L_{pred} is the total predictor loss for all layers $l \in 1...L$. In details, we define L_{pred} as follows:

$$L_{pred}(\{P^l, g^l\}_L) = \\ \sum_l^L \sum_f^{F_l} BCEWithLogits(P_f^l, g_f^l) \quad (2)$$

where P^l is the output of the decision head $f_p^l(I^l; W_p{}^l)$, g^l is the generated ground truth mask based on the top-k highly activated output O^l, $BCEWithLogits$ is a $Sigmoid$ followed by the binary cross entropy loss $BCE(p,g) = -(g\log(p) + (1-g)\log(1-p))$.

Number of activated k   We automatically calculate k by keeping a constant percentage r of the mass of heatmap. For each channel $i = 1,...,c_l$, we keep the maximum

response by applying a global maximum pooling (GMP) $(\text{GMP}(|O_i^l|))$. For each input example, k is the number of filters kept such that the cumulative mass of the sorted normalize activation reaches $r\%$. Ground truth generation algorithm is shown in Algorithm 1. We use the same r for all layers, however, each sample will have different pruning ratio per layer based on its activation. As the target binary groundtruth is generated from the activations of the unmasked filters, FLOPs reduction can be loosely estimated prior to training to adjust r accordingly. This advantage adds to the practicality of our method which does not rely on indirect hyperparameter tuning to reach a budget target in FLOPs reduction. It is also worth mentioning that $r = 1$ is a special case that indicates the decision head will predict the completely deactivated features (e.g maximum response is zero). This enables maintaining accuracy of baseline with ideally trained decision head for highly sparse backbones.

Algorithm 1 Binary mask ground truth generation

Input: $I^1 \dots I^L, r$
Output: g binary ground truth with 0 as to prune
 1: $gt \leftarrow ones(L, c_l)$
 2: **for** $l \leftarrow 1$ to L **do**
 3:     $acts \leftarrow \text{GMP}((|O^l|))$
 4:     $normalized \leftarrow acts / \sum acts$
 5:     $sorted, idx \leftarrow \text{SORT}(normalized, \text{“}descend\text{”})$
 6:     $cumulative \leftarrow \text{CUMSUM}(sorted)$
 7:     $prune\_idx \leftarrow \text{WHERE}(cumulative > r)$
 8:     $gt[l][prune\_idx] \leftarrow \boldsymbol{0}$
 9: **end for**

3.3. Prediction Head Design

Prediction head design should be modeled in a simple way to reduce overhead over baseline network. In forward pass, we apply GMP that reduces feature map $\mathbf{I}^1$ per layer to $E^l \in \mathbb{R}^{c^{l-1} \times 1 \times 1}$. Next, we apply 1x1 convolution on the flattened embedding E^l to produce the mask's logits. We experiment with two training modes: 1) decoupled, and 2) joint. In both modes, we train backbone weights and those of decision heads in parallel. The distinction is whether we do a fully differentiable training (joint) or stop gradients from heads to backpropagate to the backbone and vise versa (decoupled). In joint training, the decision head is fully differentiable except at the binarization part. Similar to previous works [8, 33, 46], we utilize straight-through estimator (STE) to bypass the non-differentiable function. An issue to consider is losses interference with multiple losses at different depth in the network as pointed out in [19]. Losses interference highlights that feature maps can be biased towards achieving high accuracy to local task more than the overall architecture. Unlike other methods that rely on careful training tuning to manage gradients from different losses,

	Model	Dynamic?	Top-1 Acc. (%)	FLOPs red. (%)
VGG16-BN	Baseline	–	93.82	—
	L1-norm [21]	N	93.00	34
	ThiNet [32]	N	93.36	50
	CP [15]	N	93.18	50
	Taylor-50 [34]	N	92.00	51
	RNP [25]	Y	92.65	50
	FBS [10]	Y	93.03	50
	LCS [40]	Y	93.45	50
	FTWT$_J$ ($r = 0.92$)	Y	93.55	**65**
	FTWT$_D$ ($r = 0.92$)	Y	**93.73**	56
	Taylor-59 [34]	N	91.50	59
	FTWT$_D$ ($r = 0.85$)	Y	**93.19**	73
	FTWT$_J$ ($r = 0.88$)	Y	92.65	**74**
ResNet56	Baseline	–	93.66	–
	Uniform from [40]	N	74.39	50
	ThiNet [32]	N	91.98	50
	SFP [13]	N	92.56	48
	LCS [40]	Y	92.57	52
	FTWT$_D$ ($r = 0.80$)	Y	**92.63**	**66**
	FTWT$_J$ ($r = 0.88$)	Y	92.28	54
MobileNetV1	Baseline	–	90.89	–
	MobileNet_75 [16]	N	89.79	42
	MobileNet_50 [16]	N	87.58	73
	FTWT$_D$ ($r = 1.0$)	Y	91.06	**78**
	FTWT$_J$ ($r = 1.0$)	Y	**91.21**	**78**

Table 1. Results on CIFAR-10. FLOPs red. indicates reduction in FLOPs in percentage. r in our method states the hyperparameter ratio in Algorithm 1. **x** in FTWT$_x$ indicates joint (J) or decoupled (D) training.

we train the heads along with backbone in parallel yet collaboratively as the masks are generated from the current status of the model. The groundtruth binary masks are explicitly adjusted by the updated backbone weights, thus, implicitly complying to the backbone learning speed.

4. Experiments and Analysis

We evaluate our method on CIFAR [20] and ImageNet [5] datasets on a variety of architectures such as VGG [38], ResNet [12] and MobileNet [16]. In all architectures, ground truth masks are generated after each conv-BN-ReLU block. For CIFAR baseline models, we train for 200 epochs using a batch-size of 128 with SGD optimizer. The initial learning rate (lr) 0.1 is divided by 10 at epochs 80, 120, and 150. We use a momentum of 0.9 with weight decay of 5^{-4}. For ImageNet, we use the pre-trained models in PyTorch [36] as baselines. Weights of decision heads are trained with 0.1 as initial lr and same lr schedule as backbone. We use a 4 V100-GPU machine in our experiments.

4.1. Experiments on CIFAR

We follow similar training settings used in baseline for dynamic training, but we train all models with initial learning rate of $1e^{-2}$. We report the average accuracy over three repeated experiments and FLOPs reduction on CIFAR-10 on multiple architectures in Table 1. Our method (FTWT) achieves higher FLOPs reduction on similar top1-accuracy than static and dynamic pruning methods. We achieve up to 66% FLOPs reduction on VGG-16 and ResNet-56, that is higher than dynamic filter pruning methods RNP [25], FBS [10], LCS [40] by up to 15%. Joint training performs equally well as decoupled training on high r thresholds. However, the accuracy drops in comparison to decoupled training on lower thresholds. That is due to conflict increase between losses as can be seen on $\approx 73\%$ FLOPs reduction on VGG. We further achieve 73% FLOPs reduction on VGG with only 0.63% accuracy drop. Moreover, FTWT outperforms smaller variants of MobileNet in accuracy by 3.42% with higher FLOPs reduction.

We visualize the number of unique combination of filters (clusters) that are activated over the whole dataset D and the pruning ratio per layer in Figure 4. Meaning that, each
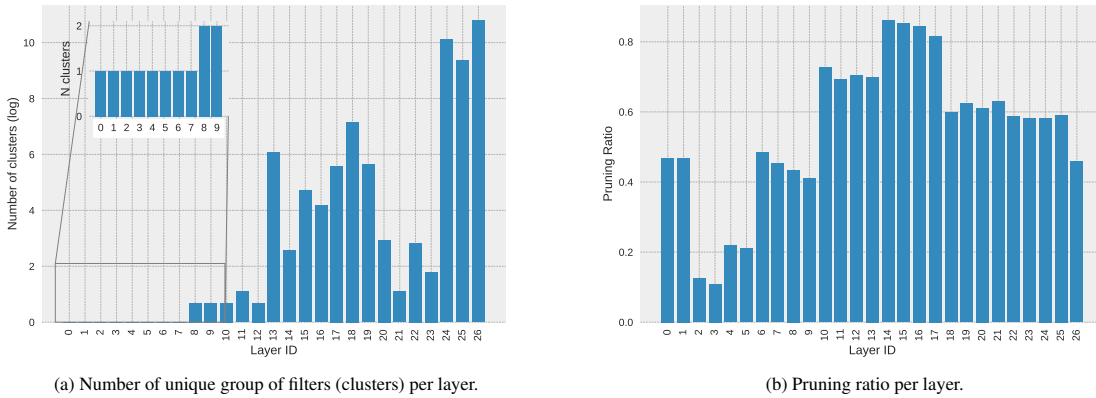
(a) Number of unique group of filters (clusters) per layer.

(b) Pruning ratio per layer.

Figure 4. MobileNetV1 CIFAR10 distributions

	Method	Dynamic?	Top-1 Acc. (%)			FLOPs red. (%)
			Baseline	Pruned	Delta	
ResNet34	Taylor [34]	N	73.31	72.83	0.48	22.25
	LCCL [6]	Y	73.42	72.99	0.43	24.80
	FTWT ($r = 0.97$)	Y	73.30	73.25	**0.05**	25.86
	FTWT ($r = 0.95$)	Y	73.30	72.79	0.51	37.77
	SFP [13]	N	73.92	71.83	2.09	41.10
	FPGM [14]	N	73.92	72.54	1.38	41.10
	FTWT ($r = 0.93$)	Y	73.30	72.17	**1.13**	47.42
	ResNet18 [12]	N	73.30	69.76	3.54	50.04
	FTWT ($r = 0.92$)	Y	73.30	71.71	**1.59**	52.24
ResNet18	PFP-B [24]	N	69.74	65.65	4.09	43.12
	SFP [13]	N	70.28	67.10	3.18	41.80
	LCCL [6]	Y	69.98	66.33	3.65	34.60
	FBS [10]	Y	70.70	68.20	2.50	49.49
	FTWT ($r = 0.91$)	Y	69.76	67.49	**2.27**	51.56
MobileNetV1	MobileNetV1-75 [16]	N	69.76	67.00	2.76	42.85
	FTWT ($r = 1$)	Y	69.57	69.66	**-0.09**	41.07

Table 2. Results on ImageNet. Baseline accuracy for each method is reported along with the pruned model's accuracy and accuracy change from baseline. FLOPs red. represents reduction in FLOPs in percentage. Negative delta indicates increase in accuracy from the baseline. r in our method states the hyperparameter ratio in Algorithm 1

sample i that produces a binary mask $m_{i,j}$ per layer j, the unique clusters per layer is $set(m_{0,j}, ..., m_{i,j}, ...m_{|D|,j})$. In LCS and RNP, a fixed number of clusters is preset as a hyperparameter for all layers, we show in Figure 4a that layers differ in the number of diverse clusters. Our method adjusts different number of clusters per layer automatically due to self-supervised mask generation mechanism. For easier visualization, y-axis is shown in log scale. Early layers have small diversity in the group of filters activated, thus, act similar to static pruning. This is sensible as early layers detect low-level features and have less dependency on the input. On the other hand, the number of clusters increases as we go deeper in the network. It is worth mentioning that these

different clusters are fine-grained, which means clusters can differ in one filter only. We also calculated the percentage of core filters that are shared among all clusters per layer. We found that the range of the percentage of core filters with respect to total filters vary from 0.4 to 1.0. This gives insight on why static pruning methods result in large drop in accuracy with large pruning. As the attainable pruning ratio is limited by the number of core filters and further pruning will limit the model's capacity. An interesting future research question would be if we can determine the compressibility of a model based on the core filters ratio notion. Finally, Figure 4b shows the pruning ratio per layer, as to be expected, the later layers are more heavily pruned than early

layers as layers get wider and more compressible. We notice heavy pruning reaching 85% in the middle layers with a sequence of layers with 512 filters.

4.2. Experiments on ImageNet

For ImageNet, we train for 90 epochs with initial learning rate of 10^{-2} that decays each 30 epochs by 0.1. Experiments on ImageNet is done with the decoupled training mode. Table 2 shows drop in accuracy from baseline for each method to account for training differences due to augmentations. Results show that our method achieve smaller drop in accuracy with higher FLOPs reduction in comparison to other SOTA methods. We achieve similar accuracy reduction as LCCL with 13% higher FLOPs reduction on ResNet34. On the other hand, on similar FLOPs reduction (≈ 25), we have minimal drop in accuracy ($\approx 0.05\%$). Although we achieve similar accuracy on ResNet18 with FBS, the latter requires a predefined number of filters to keep per layer. On the other hand, our method dynamically assigns pruning ratio per layer which shows the effectiveness of our heatmap mass as a criteria. We also compare with architecture's smaller variants such as ResNet18 and MobileNet-75. We outperform ResNet18 and MobileNet-75 by $\approx 2\%$ in accuracy on similar computation budget.

4.3. Ablation Study

4.3.1 Uncertainty under Dataset Shift

In this section, we measure the sensitivity of our routing to dataset shift. Metrics under dataset shift are rarely inspected in model pruning literature. We believe in its importance as inference complexity increases and thus would like to initiate reporting such comparisons. We conduct experiments for VGG16-bn CIFAR-10 with a high pruning ratio 73% for all pruned models. Inspired by [35], we report Brier score [3] under different type of noise such as Gaussian blur Table 3a and additive noise Table 3b with baseline dense model as a reference. As can be seen, our method is more resilient than static Taylor pruning with a lower brier scores. We also compare with static uniform pruning, we achieve a similar (sometimes slightly lower) Brier score. This shows the resilience of our model to data shift even when compared with static pruning decision that is not data-dependent. Finally, as to be expected, the dense model is the most resilient to noise. However, our method still shows a fair quality matching overall. We attribute this distribution stability to the softmax in the head. The softmax acts as a normalizer which reduces sensitivity to distribution shift. We compare our method with and without softmax normalization in the decision head to verify this hypothesis. Table 3c shows Brier scores with additive and blurring noise for this comparison. As can be seen, indeed, the normalizer stabilizes the decision masks output especially in the case of blurring.

4.3.2 Dynamic Signature and Dynamic Routing

We investigate decoupling the effect from the dynamic signature (i.e. pruning ratio per layer) per sample from the dynamic routing (i.e group of filters to be activated). We explore the effectiveness of dynamic routing with a pre-defined signature for all inputs. In these experiments, signature is pre-defined using Taylor criteria proposed in [34] as a case study. As in previous setup, we select the highly activated k features where k is defined by the signature while samples differ in which k filters are selected. Table 6 shows results of dynamic routing under different pruning ratios. As can be seen, dynamic routing performs better than static inference especially on high pruning ratio by up to 4%. Training setup is the same for CIFAR as explained in paper, however, we train ImageNet models for 30 epochs using finetune setup instead of training setup with 90 epochs as differentiated in [29] to accelerate the experiment.

4.3.3 Hyperparameter r selection

The hyperparameter r (i.e mass ratio) is selected based on a simple evaluation before training. W estimate the FLOPs before training by applying the groundtruth masks on the pretrained frozen dense model over the training set (one-shot pass). Subsequently, we get an estimate before training is initiated of the expected FLOPs reduction under different r values. This simplifies hyperparameter selection to achieve a target FLOPs reduction. On the other hand, sparse regularization hyperparameter is usually fine-tuned with a cross-validation process and requires a trial and error of multiple full training to achieve a target budget. There is no direct relation between the regularization weight and the final achieved FLOPs reduction knowingly before training. Our method simplifies the selection and makes it a more practical option when a target budget is given as a prior. Table 4 shows the estimated FLOPs before training using different thresholds, r, and the actual reached FLOPs reduction after training. The difference in reduction is due to the inaccuracy of the decision heads. Nonetheless, the estimated FLOPs gives a good approximation to the final reached FLOPs and thus reduce the hyperparameter search.

4.4. Theoretical vs Practical Speedup

For all compression methods, including static and dynamic pruning, there is often a wide gap between FLOPs reduction and realistic speedup due to other factors such as I/O delays and BLAS libraries. Speedup is hardware and backend dependent as shown in prior works [1, 7, 45]. We test the realistic speed on PyTorch [36] using MKL backend on AMD CPU using a single thread as shown on Table 5.

Limitations. Our realistic speedup is less than FLOPs reduction and that is attributed to two factors: 1) Data transfer overhead from slicing the dense weight matrix based

σ	Dense model	FTWT (ours)	Taylor Pruning	Uniform Pruning
0.5	0.11	**0.12**	0.20	**0.12**
0.7	0.16	**0.18**	0.39	0.19
0.9	0.38	**0.39**	0.57	0.42
1.09	0.69	**0.58**	0.66	0.61
1.27	0.74	**0.68**	0.69	0.71
1.45	0.76	**0.73**	0.74	0.75

(a) Gaussian blur noise.

σ	Dense model	FTWT (ours)	Taylor Pruning	Uniform Pruning
0.00	0.11	**0.12**	0.16	**0.12**
0.02	0.11	**0.12**	0.16	**0.12**
0.05	0.11	**0.12**	0.17	0.13
0.11	0.13	**0.14**	0.20	0.15
0.14	0.19	**0.21**	0.30	0.22
0.20	0.39	0.43	0.51	**0.42**

(b) Additive Gaussian noise.

Gaussian Blur		Additive Noise	
FTWT with normalization	FTWT without normalization	FTWT with normalization	FTWT without normalization
0.12	0.19	**0.12**	0.12
0.18	0.48	**0.12**	0.13
0.39	0.73	**0.14**	0.15
0.58	0.85	**0.21**	0.24
0.68	0.90	**0.43**	0.47

(c) Our method with and without softmax normalization in decision heads.

Table 3. Dataset shift experiments: Numbers represent Brier score on CIFAR-10 VGG16

Model	Est. FLOPs (%)	Final FLOPs (%)
MobileNet$_{(1.0)}$	42.3	41.07
Resnet34$_{(0.97)}$	23.32	25.86
Resnet34$_{(0.95)}$	31.77	37.77

Table 4. Estimated FLOPs *before* training under different thresholds (indicated in parentheses) vs achieved FLOPs after training.

Model	FLOPs reduction (%)	Latency reduction (%)
ResNet34	52.18	27.17
	37.77	19.78
	25.86	11.08

Table 5. Realistic vs theoretical speedup on ImageNet on AMD Ryzen Threadripper 2970WX CPU with batch size of 1.

Dataset	Model	FLOPs (%)	Top-1 acc. (%) Static	Top-1 acc. (%) Dynamic
CIFAR-10	VGG16-BN	50	92.00	**93.80**
		85	91.12	**92.75**
	ResNet56	70	91.61	**92.09**
CIFAR-100	VGG16-BN	30	72.65	**73.67**
		65	68.17	**72.18**
		93	58.74	**60.05**
ImageNet	ResNet18	45	64.89	**65.11**

Table 6. Accuracy comparison of dynamic routing with a pre-defined signature and its counterpart with static inference.

on the mask prediction, which can be mitigated by backends with efficient in-place sparse inference. 2) Speed up is dependent on the model's signature and hardware's specs. Pruning from later layers that process smaller input resolution might not achieve as much speedup as pruning from early layers. Constraint aware optimization using Alternating Direction Method of Multipliers (ADMM) [2] such as proposed in [44] can be further integrated with our method to optimize over latency instead of FLOPs.

5. Conclusion

In this paper, we propose a novel formulation for dynamic model pruning. Similar to other dynamic pruning methods, we equip a cheap decision head to the original convolutional layer. However, we propose to train the decision heads in a self-supervised paradigm. This head predicts the most likely to be highly activated filters given the layer's input activation. The masks are trained using a binary cross entropy loss decoupled from the task loss to remove losses interference. We generate the mask ground truth based on a novel criteria using the heatmap mass per input sample. In our experiments, we showed results on various architectures on CIFAR and ImageNet datasets, and our approach outperforms other dynamic and static pruning methods under similar FLOPs reduction.

Acknowledgment

References

[1] Simone Bianco, Remi Cadene, Luigi Celona, and Paolo Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018. 7

[2] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. 8

[3] Glenn W Brier et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950. 7

[4] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 1

[5] 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. 5

[6] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2017. 1, 6

[7] Sara Elkerdawy, Mostafa Elhoushi, Abhineet Singh, Hong Zhang, and Nilanjan Ray. To filter prune, or to layer prune, that is the question. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2, 7

[8] Sara Elkerdawy, Hong Zhang, and Nilanjan Ray. Lightweight monocular depth estimation model by joint end-to-end filter pruning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 4290–4294. IEEE, 2019. 4

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 1, 2

[10] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018. 1, 3, 5, 6

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015. 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. 5, 6

[13] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 2, 5, 6

[14] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 6

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 2, 5

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 5, 6

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1

[18] Weizhe Hua, Yuan Zhou, Christopher De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *arXiv preprint arXiv:1805.12549*, 2018. 1, 3

[19] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017. 4

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[21] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1, 5

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ICLR*, 2017. 2

[23] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019. 1

[24] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. 6

[25] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2178–2188, 2017. 1, 3, 5

[26] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 3

[27] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 2

[28] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE ICCV*, pages 2736–2744, 2017. 3

[29] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 1, 7

[30] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017. 2

[31] Siegrid Lowel and Wolf Singer. Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. *Science*, 255(5041):209–212, 1992. 2

[32] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 2, 5

[33] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018. 4

[34] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 1, 2, 3, 5, 6, 7

[35] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530*, 2019. 7

[36] 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. *arXiv preprint arXiv:1912.01703*, 2019. 5, 7

[37] Sayeh Sharify, Alberto Delmas Lascorz, Mostafa Mahmoud, Milos Nikolic, Kevin Siu, Dylan Malone Stuart, Zissis Poulos, and Andreas Moshovos. Laconic deep learning inference acceleration. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pages 304–317. IEEE, 2019. 2

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5

[39] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. 1

[40] Yulong Wang, Xiaolu Zhang, Xiaolin Hu, Bo Zhang, and Hang Su. Dynamic network pruning with interpretable layerwise channel selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6299–6306, 2020. 1, 3, 5

[41] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016. 3

[42] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 1

[43] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018. 3

[44] Haichuan Yang, Yuhao Zhu, and Ji Liu. Ecc: Platform-independent energy-constrained deep neural network compression via a bilinear regression model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11206–11215, 2019. 8

[45] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017. 7

[46] Xucheng Ye, Pengcheng Dai, Junyu Luo, Xin Guo, Yingjie Qi, Jianlei Yang, and Yiran Chen. Accelerating cnn training by pruning activation gradients. In *European Conference on Computer Vision*, pages 322–338. Springer, 2020. 4

[47] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 1