Interspace Pruning: Using Adaptive Filter Representations to Improve Training of Sparse CNNs

Paul Wimmer[*†‡§], Jens Mehnert[*§] and Alexandru Condurache[*†]
[*]Automated Driving Research, Robert Bosch GmbH, 70469 Stuttgart, Germany
[†]Institute for Signal Processing, University of Lübeck, 23562 Lübeck, Germany
{paul.wimmer,jensericmarkus.mehnert,alexandrupaul.condurache}@de.bosch.com

Abstract

Unstructured pruning is well suited to reduce the memory footprint of convolutional neural networks (CNNs), both at training and inference time. CNNs contain parameters arranged in $K \times K$ filters. Standard unstructured pruning (SP) reduces the memory footprint of CNNs by setting filter elements to zero, thereby specifying a fixed subspace that constrains the filter. Especially if pruning is applied before or during training, this induces a strong bias. To overcome this, we introduce interspace pruning (IP), a general tool to improve existing pruning methods. It uses filters represented in a dynamic interspace by linear combinations of an underlying adaptive filter basis (FB). For IP, FB coefficients are set to zero while un-pruned coefficients and FBs are trained jointly. In this work, we provide mathematical evidence for IP's superior performance and demonstrate that IP outperforms SP on all tested state-of-the-art unstructured pruning methods. Especially in challenging situations, like pruning for ImageNet or pruning to high sparsity, IP greatly exceeds SP with equal runtime and parameter costs. Finally, we show that advances of IP are due to improved trainability and superior generalization ability.

1. Introduction

Deep neural networks (DNNs) have shown state-of-the-art (SOTA) performance in many artificial intelligence applications [52, 54, 72, 77, 80]. In order to solve these tasks, large models with up to billions of parameters are required. However, training, transferring, storing and evaluating such large models is costly [61,65]. *Pruning* [19,20,22,30,34,50] sets parts of the network's weights to zero. This reduces the model's complexity and memory requirements, speeds up inference [4] and may lead to an improved generalization ability [3, 24, 34]. In recent years, *training* sparse models became of interest, providing the benefits of reduced memory requirements and runtime not only for inference but also for training [13, 14, 35, 47, 49, 55, 66, 71, 75].

In this work, we mainly focus on methods that prune individual parameters *before* training, while the number of zeroed coefficients is kept fixed during training. With this *unstructured pruning*, a network's memory footprint can be reduced. To lower the runtime in addition, specialized soft- and hardware is needed [11, 18, 21, 51]. For training sparse networks, we distinguish between (i) *pruning at initialization* (PaI) [9, 35, 66, 71, 75] which prunes the network at initialization and fixes zeroed parameters during training, (ii) finding the sparse architecture to be finally trained by iterative train-prune-reset cycles, a so called *lottery ticket* (LT) [14,15], and (iii) *dynamic sparse training* (DST) [13,39,49] which prunes the network at initialization, but allows the pruning mask to be changed during training.

Convolutional neural networks (CNNs) are composed of layers, each having a certain number of input- and output channels. Every combination of input- and output channel is linked by a *filter* $h \in \mathbb{R}^{K\times K}$ with *kernel size* $K \times K$. A weight of h is a *spatial coefficient* $h_{i,j}$ for a spatial coordinate (i, j). Filters h can also be modeled in an *interspace*, a linear space $\{\sum_{n=1}^{K^2} \lambda_n \cdot g^{(n)} : \lambda_n \in \mathbb{R}\}$ spanned by a *filter basis* (FB) $\mathcal{F} := \{g^{(1)}, \dots, g^{(K^2)}\} \subset \mathbb{R}^{K\times K}$ [12, 69]. One possibility for a FB is the standard basis $\mathcal{B} := \{e^{(n)} : n = 1, \dots, K^2\}$ which yields the spatial representations. General interspace representations are more flexible since bases are not fixed. We represent h in an interspace in order to learn the FB $\mathcal{F}$ spanning this space along with the *FB coefficients* λ, and thereby obtain a better representation for h. Thus, setting coefficients of flexible, adaptive FBs to zero will improve results compared to prune spatial coefficients.

For deep networks, where the layers' purposes are usually unknown to the experts but learnt during training, we believe that filters should train their bases along with their coefficients. A FB $\mathcal{F}$ is *dynamic*, can be shared for any

[‡]Corresponding author. [§]Equal contribution.

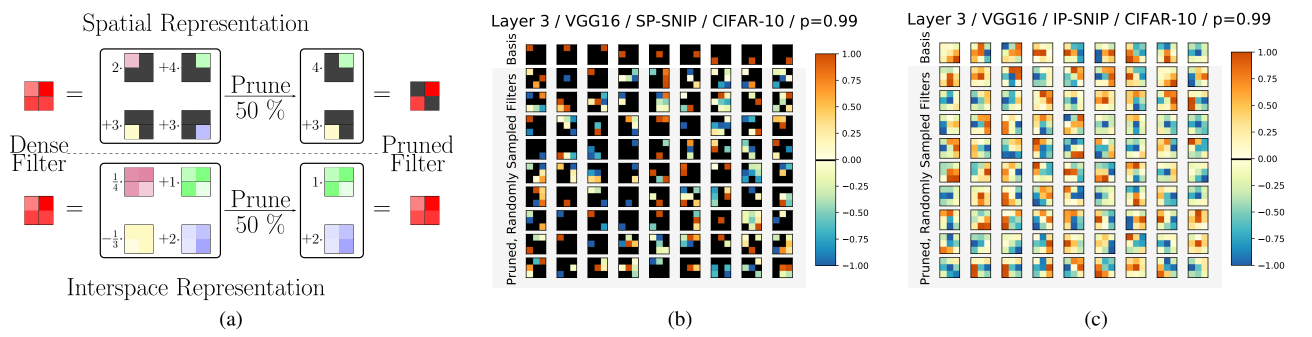

Spatial Representation
Dense Filter
Prune 50 %
Pruned Filter
Interspace Representation
(a)
Layer 3 / VGG16 / SP-SNIP / CIFAR-10 / p=0.99
Basis
Pruned, Randomly Sampled Filters
(b)
Layer 3 / VGG16 / IP-SNIP / CIFAR-10 / p=0.99
(c)

Figure 1. (a) Overview of SP and IP. Contrarily to SP (b), IP (c) produces spatially dense filters after training sparse networks. As for SP, sparsity in the interspace can be used to reduce memory requirements and, by the linearity of convolutions, also computational costs.

number of $K \times K$ filters, and is optimized jointly with its FB coefficients λ. By fitting an interspace to sparse filters during training, we overcome the lack of prior knowledge for a basis that is well suited to describe filters with few non-zero coefficients. If a filter is pruned to a single FB coefficient, $h = \lambda_n \cdot g^{(n)}$, it is not restricted since $g^{(n)}$ can change. Thus, pruning interspace coefficients of dynamic FBs keeps the CNN flexible and is called *interspace pruning* (IP). A 1-sparse filter $h = h_{i_n,j_n} \cdot e^{(n)}$ directly predefines h to stay on the fixed subspace $\mathrm{span}\{e^{(n)}\}$. Pruning spatial coefficients w.r.t. the standard basis $\mathcal{B}$ is called *standard pruning* (SP).

During training sparse CNNs, the problem of vanishing gradients due to spatial sparsity often occurs [66, 71, 74]. In contrast, IP pruned networks are able to learn spatially dense FBs during training, even when using sparse interspace coefficients, see Fig. 1. Therefore, IP leads to an improved information flow and better trainable models.

Although IP yields dense spatial representations, the linearity of convolutions can be used to reduce the number of computations for CNNs with sparse interspace coefficients. Compared to SP, IP only increases the number of required computations by a small, constant count. However, as IP provides superior sparse models, IP generates CNNs with faster inference speed than SP while matching the dense performance. Further, the dynamic achieved by interspace representations is cheap in terms of memory. A FB $\mathcal{F}$ has K^4 parameters as it contains K^2 filters of size $K \times K$. A single FB can be shared for all $K \times K$ filters in a CNN. Also, more than one FB can be used with just a small increase in memory requirements. For cost reasons, we do not use more FBs than the number of layers in a CNN in our experiments, resulting in *all* FBs creating an overhead of at most 0.01% of the dense network's parameters. Despite adding only few additional costs compared to using spatial weights, interspace representations significantly improve results for sparse *and* dense training.

Our core contributions are:

- Representing and training convolutional filters in the interspace, a linear space spanned by a trainable FB. The FB is optimized jointly with the FB coefficients.
- Formulating the concept of pruning for filters with interspace representation as general method to improve performance of CNNs with sparse coefficients.
- Theoretical proof of IP's improvements in Thm. 1.
- Experiments showing that IP exceeds SP for equal runtime and memory costs on SOTA sparse training methods and pruning methods which are applied during training or on pre-trained models. We demonstrate that IP's superiority is achieved by improved trainability, and at lower sparsity also due to better generalization.

2. Broader Impact

Pruning can lower costs for training, storing and evaluating DNNs. We are not aware of any negative outcome directly induced by this work. Nevertheless, as tool to improve pruning, and therefore to reduce costs for CNNs, IP could be used for any CNN based application with negative ethical or societal impact. As authors, we distance ourselves from such applications and the use of our method therein.

As we show in the paper, IP improves unstructured pruning in general and is not restricted to a special scenario. We see IP as a tool which is applied in combination with SOTA SP techniques to lower costs further. Consequently, our work is to the advantage of everyone using pruning and, by the improved generalization ability obtained by training with interspace representations, deep learning in general.

3. Related work

Related work covers *general pruning* and *pruning before training and DST*. Training a sparse model allows to learn non-zero FB coefficients and FBs jointly from scratch. Such methods naturally benefit most from interspace representations and our experiments thus place a strong focus on them.

General pruning. Pruning is divided in *structured* and *unstructured* pruning. Structured pruning removes coarse structures of the network, like channels or neurons [2, 27, 38, 67, 73, 79]. This yields lean architectures and thus reduces computation time. A more fine-grained approach is unstructured pruning where single, spatial weights are zeroed [14, 17, 20, 31, 34, 35, 49]. Unstructured pruning leads to better performance than structured pruning [36, 48] but requires soft- and hardware that supports sparse tensor computations to actually reduce runtime [21, 51]. Also, storing sparse parameters in formats such as the *compressed sparse row format* [68] creates additional overhead. This can lead to non-linear dependencies between the sparsity and actual memory/runtime costs, see also Appendix Secs. C and D.4.

Pruning can be applied at any time in training. The historically first approaches [23, 30, 31, 34, 50] use trained networks and many prune and fine-tune cycles. Criteria are often based on expensive computations of the Hessian w.r.t. the loss function. Likewise, magnitudes of trained coefficients can be used as iterative pruning criterion [17, 22, 36]. By adding sparsity forcing regularizations to the loss, pruning can be integrated dynamically into training [5, 42, 76].

Closest to our work are pruning coefficients in the frequency [41] and the Winograd domain [40]. Contrarily, we do not bind representations to a fixed basis but let the network learn its FBs self-reliantly. Moreover, IP is not a pruning method by itself, but is added on top of existing ones to boost them. Also to mention is [37], a low rank approximation of CNNs. A dense, pre-trained network is approximated by learning undercomplete dictionaries for 3D filters. We, on the contrary, represent 2D filters $h \in \mathbb{R}^{K \times K}$, prune the network instead of using low rank approximations and learn FBs jointly with the coefficients in one training.

Pruning before training and dynamic sparse training. In [14], an iterative procedure is proposed which consists of training un-pruned weights to convergence, applying magnitude pruning with a small pruning rate to the trained weights and resetting the non-zero weights to their initial value. Finally, this leads to sparse, randomly initialized networks which are well trainable – so called lottery tickets. For SOTA CNNs, resetting un-pruned weights not to the initialization but a value from an early iteration improves performance significantly [15, 59]. By applying other criteria, like *information flow* in the sparse network [53,66,71,75] or influence of non-zero weights on changing the loss [9,35,70,75], pruning can be successfully applied at initialization without pre-training the network. GraSP [71], SNIP [35] and SynFlow [66] are SOTA for PaI [16]. Dynamic sparse training [10, 13, 39, 49] adjusts pruning masks during training to ensure sparse networks while adapting the architecture to different conditions. SET [49] frequently prunes the network based on magnitudes and activates as many un-trained parameters randomly. RigL [13] improves this by recovering those weights with the biggest gradient magnitude.

4. Filter bases and interspace pruning

Inspired by *sparse dictionary learning* (SDL) (Sec. 4.1) we introduce interspace representations of convolutional filters and propose computations of resulting FB convolutions in Sec. 4.2. Further, Sec. 4.3 discusses FB sharing and the initialization of FBs and their coefficients. Finally, interspace pruning is formally defined in Sec. 4.4.

4.1. Inspiration from sparse dictionary learning

Sparse dictionary learning [1, 12, 46] optimizes a dictionary $\mathbf{F} \in \mathbb{R}^{m \times m}$ jointly with coefficients $R \in \mathbb{R}^{m \times n}$ to approximate a target $U \in \mathbb{R}^{m \times n}$ by using only s non-zero coefficients. Setting the *pruning mask* $\operatorname{supp} R := \{(i, j) : R_{i,j} \neq 0\}$, this defines a non-convex optimization problem

$$\inf_{\mathbf{F}, R} \|U - \mathbf{F} \cdot R\|_F \;\text{ s.t. } \|R\|_0 := \# \operatorname{supp} R \leq s \,. \tag{1}$$

Usually, SDL allows $\mathbf{F} \in \mathbb{R}^{m \times M}$ with arbitrary M. Since FBs are bases, we restrict $\mathbf{F}$ to be quadratic. In our context, U corresponds to all flattened filters of a convolutional layer, the dictionary $\mathbf{F}$ to the layer's flattened FB $\mathcal{F}$ and R to the FB coefficients. For a layer $h \in \mathbb{R}^{c_{out} \times c_{in} \times K \times K}$ with an associated FB $\mathcal{F} = \{g^{(1)}, \dots, g^{(K^2)}\} \subset \mathbb{R}^{K \times K}$, we have $m = K^2$ and $n = c_{out} \cdot c_{in}$. Standard magnitude pruning is a special case of SDL where $\mathbf{F}$ is fixed to form the standard basis $\mathbf{F} = \mathrm{id}_{\mathbb{R}^m}$. Accordingly,

$$\min_{\bar{R}} \|U - \bar{R}\|_F \;\text{ s.t. } \|\bar{R}\|_0 \leq s \tag{2}$$

is minimized for magnitude pruning. Since we train sparse, randomly initialized CNNs, our overall goal is not to *mimic* a given dense CNN, but to train the sparse network to *generalize* well. We consequently use Eqs. (1) and (2) only to find a decent subset of coefficients to be pruned. In contrast to SDL, deep learning methods are used to further optimize the un-pruned coefficients and additionally the FBs in the case of IP. In our experimental evaluation, we also test other methods than magnitude pruning, *i.e.* Eqs. (1) and (2). Still, Eqs. (1) and (2) measure the ability of a sparse layer to function as well as a dense layer and thus are good indicators for the general performance of IP and SP, respectively.

Most SDL algorithms [1, 12, 46] optimize $\mathbf{F}$ and R alternatingly. Whereas, SP-PaI fixes the basis as $\mathrm{id}_{\mathbb{R}^m}$ and the pruning mask $\operatorname{supp} \bar{R}$ too. This simplifies the task, but reduces the solution space. IP overcomes the small, fixed solution space by adapting the basis during training. For IP-PaI, the pruning mask $\operatorname{supp} R$ is determined heuristically and also fixed which still leads to sub-optimal architectures. As shown in this work, using expensive pre-training to find a better pruning mask via LTs or adapting $\operatorname{supp} R$ during training via DST further improves IP's performance.

Theorem 1 shows that a dynamic $\mathbf{F}$ leads to better approximations than using the standard basis. Consequently,

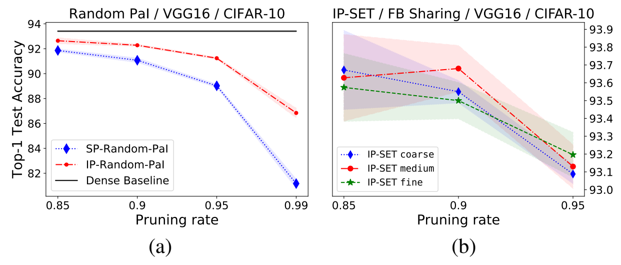


Figure 2. VGG16 on CIFAR-10: (a) Random SP-PaI and random IP-PaI. (b) `Coarse`, `medium` and `fine` FB sharing for IP-SET.

the FBs' adaptivity improves performance after pruning and Thm. 1 is a theoretical motivation for IP.

Assume a convolutional layer with c_{out} output and c_{in} input channels, kernel size $K \times K$ and $s = (1-p) \cdot c_{out} \cdot c_{in} \cdot K^2$ un-pruned coefficients. For $m = K^2 \geq 9$, the δ in Eq. (3) is numerically equal to zero if $n = c_{out} \cdot c_{in} \geq 100$ and $0 < s < c_{out} \cdot c_{in} \cdot K^2$. Thus, for each non-trivial sparsity, the adaptivity of the FB improves results. This even holds if the pruning mask for Eq. (1) is fixed to be the one of the minimizer of Eq. (2), *i.e.* starting with an arbitrary pruned network and adding an adaptive FB always improves results. The proof of Thm. 1 is shown in Appendix Sec. J. It uses the fact that Eq. (1) is smaller or equal to Eq. (2). Equality is only possible if Eq. (2) has a solution such that each $K \times K$ filter is either fully pruned or dense. This is almost impossible for big layers and a non trivial sparsity. If $\operatorname{supp} R$ is further not fixed for Eq. (1), $(\mathbf{F}, R)$ can be chosen such that Eq. (1) is always strictly smaller than Eq. (2).

Theorem 1. *Let $0 < s < m \cdot n$, $m > 1$ and $U_{i,j} \sim \mathcal{N}(0,1)$ i.i.d. Let $\varepsilon_{(1)}$ be the infimum of Eq. (1), and $\varepsilon_{(2)}$ the minimum of Eq. (2) solved by $\bar{R}^*$. Then, $\varepsilon_{(1)} < \varepsilon_{(2)}$ with $\mathbb{P} = 1$. If $\operatorname{supp} R$ is fixed for Eq. (1) to be $\operatorname{supp} \bar{R}^*$, $\varepsilon_{(1)} \leq \varepsilon_{(2)}$ is true and strict inequality holds with $\mathbb{P} \geq 1 - \delta$, where*

$$\delta = \begin{cases} \binom{n}{\frac{s}{m}} / \binom{m \cdot n}{s} & \text{if } s \equiv 0 (\operatorname{mod} m) \\ 0 & \text{else} \end{cases} . \tag{3}$$

Figure 2(a) compares SP and IP for random PaI for a VGG16 [63] trained on CIFAR-10 [32]. IP improves results tremendously compared to SP. This experimentally shows that sparse training performs better when coefficients of adaptive FBs are pruned than if spatial weights are pruned. This holds even though fixed pruning masks are used.

4.2. Interspace representation and convolutions

For a convolutional layer, let c_{out} denote its number of output channels, c_{in} its number of input channels and $K \times K$ its kernel size. To simplify formulas, we restrict the formulation to 2D convolutions with quadratic kernel, no padding, 1×1 stride and dilation. Generalizing the FB formulations to arbitrary convolutions is straightforward. A 2D convolution h describing this layer consists of $c_{out} \cdot c_{in}$ $K \times K$ filters $h^{(\alpha,\beta)} \in \mathbb{R}^{K \times K}$, *i.e.* $h = (h^{(\alpha,\beta)})_{\alpha,\beta} \in \mathbb{R}^{c_{out} \times c_{in} \times K \times K}$. Inspired by the discussion in Sec. 4.1, we now represent all $h^{(\alpha,\beta)}$ in the interspace spanned by the layer's FB $\mathcal{F} = \{g^{(1)}, \dots, g^{(K^2)}\} \subset \mathbb{R}^{K \times K}$. The FB coefficients $\lambda = (\lambda_n^{(\alpha,\beta)})_{\alpha,\beta,n} \in \mathbb{R}^{c_{out} \times c_{in} \times K^2}$ define the interspace representation of $h^{(\alpha,\beta)}$, given by

$$h^{(\alpha,\beta)} = \sum_{n=1}^{K^2} \lambda_n^{(\alpha,\beta)} \cdot g^{(n)} . \tag{4}$$

This is a basis transformation of the spatial representation

$$h^{(\alpha,\beta)} = \sum_{n=1}^{K^2} h_{i_n,j_n}^{(\alpha,\beta)} \cdot e^{(n)} , \; e_{i,j}^{(n)} = \delta_{i,i_n} \cdot \delta_{j,j_n} . \tag{5}$$

Normally, $h^{(\alpha,\beta)}$ is defined in spatial representation. Thus, spatial coefficients are stored in $h^{(\alpha,\beta)} \in \mathbb{R}^{K \times K}$. Whereas, FB coefficients are specified by vectors $\lambda^{(\alpha,\beta)} \in \mathbb{R}^{K^2}$. By linearity, a 2D FB convolution $(Y^{(\alpha)})_\alpha = Y = h \star X$ with input feature map $X = (X^{(\beta)})_\beta \in \mathbb{R}^{c_{in} \times h \times w}$ can be computed for each output channel $\alpha \in \{1, \dots, c_{out}\}$ as

$$Y^{(\alpha)} = \sum_{\beta=1}^{c_{in}} h^{(\alpha,\beta)} \star X^{(\beta)} \overset{(4)}{=} \sum_{\beta=1}^{c_{in}} \sum_{n=1}^{K^2} \lambda_n^{(\alpha,\beta)} (g^{(n)} \star X^{(\beta)}) . \tag{6}$$

Gradients of the loss $\mathcal{L}$ are needed to train the FB coefficients λ and the FB $\mathcal{F}$. Backpropagation formulas for them are derived in Appendix Sec. D.2. It holds for all n, α, β

$$\frac{\partial \mathcal{L}}{\partial \lambda_n^{(\alpha,\beta)}} = \left\langle g^{(n)}, \frac{\partial \mathcal{L}}{\partial h^{(\alpha,\beta)}} \right\rangle , \; \frac{\partial \mathcal{L}}{\partial g^{(n)}} = \sum_{\alpha,\beta} \lambda_n^{(\alpha,\beta)} \frac{\partial \mathcal{L}}{h^{(\alpha,\beta)}} . \tag{7}$$

4.3. Filter basis sharing and initialization

For kernel size 1×1, the FB formulation is, up to a rescaling, equivalent to the spatial representation. Thus, we assume a CNN with L_c convolutional layers with $K > 1$ to be given and do not apply the FB formulation to 1×1 convolutions. In this work, we test three versions of FB sharing. Our FB sharing schemes differ in their *granularity*. The `coarse` scheme shares one global FB $\mathcal{F}$ for all layers $l = 1, \dots, L_c$. Whereas, the `fine` scheme shares a FB $\mathcal{F}^{(l)}$ for each layer l, thus it uses L_c FBs. In between lies the `medium` scheme with 5 FBs in total. For ResNets [26], one FB is shared for each of the 5 convolutional blocks. For VGG16 [63], convolutional layers $\{1, 2\}$, $\{3, 4\}$, $\{5, 6, 7\}$, $\{8, 9, 10\}$ and $\{11, 12, 13\}$ share one FB each. The number of FBs increases from `fine` to `coarse`. The total number of FBs in the network, J, satisfies $J \leq L_c$. Consequently, the number of parameters in *all* FBs in the network is bounded from above by $L_c \cdot K^4$. Note for the CNNs used in this work, $L_c \cdot K^4$ is at most 0.01% of all parameters in the model. Thus, the additional parameter costs for IP with our proposed sharing schemes are neglectable.

The dimension of the space spanned by each layer does not change for different FB sharing schemes and is equal to using spatial representations. However, `coarse` sharing correlates all layers in the network by using and updating the same interspace. For `fine` sharing, each layer has its own interspace which is adapted more fine-grained. For spatial representations, the basis $\mathcal{B}$ is fixed, not updated and does not induce correlations between weights. We found different sharing schemes to work best for varying training/model/dataset combinations. Figure 2(b) shows our FB sharing schemes for different pruning rates. `Coarse` sharing works best for higher numbers of trained parameters. By correlating all layers through a global FB, we assume it to have a regularizing effect on training, see also Sec. 5.4. `Fine` sharing makes the network more flexible. Thus, results are the best ones for high pruning rates where the network is not able to overfit on the training data anymore. In between, `medium` sharing reaches the best results by combining the best of both worlds.

In this work, we use a simple initialization for FBs and FB coefficients. We initialize each FB as $\mathcal{B}$ and the FB coefficients with a `kaiming normal` initialization [25]. This scheme is equivalent to the `kaiming normal` initialization for standard CNNs – which is also used for dense baselines and SP experiments. In Appendix Sec. G, we propose further initialization schemes for the interspace.

4.4. Interspace pruning and cost comparison

SP is modeled by superimposing *pruning masks* $\bar{\mu}^{(\alpha,\beta)} \in \{0,1\}^{K\times K}$ over filters $h^{(\alpha,\beta)} \in \mathbb{R}^{K\times K}$. This results in sparse filters $h^{(\alpha,\beta)} \odot \bar{\mu}^{(\alpha,\beta)}$, with the Hadamard product $\odot$. Filters represented in the interspace have coefficients $\lambda^{(\alpha,\beta)} \in \mathbb{R}^{K^2}$ w.r.t. a FB $\mathcal{F}$. Thus, interspace pruning is defined by masking FB coefficients with pruning masks $\mu^{(\alpha,\beta)} \in \{0,1\}^{K^2}$ via $\lambda^{(\alpha,\beta)} \odot \mu^{(\alpha,\beta)}$. Combined with Eq. (6), IP yields sparse computations of convolutions:

$$Y^{(\alpha)} = \sum_{\beta=1}^{c_{in}} \sum_{n \in \operatorname{supp} \mu^{(\alpha,\beta)}} \lambda_n^{(\alpha,\beta)} \cdot \left(g^{(n)} \star X^{(\beta)}\right) . \tag{8}$$

The *pruning rate* p for SP (p_{SP}) and IP (p_{IP}) is defined as

$$p_{SP} = 1 - \frac{\|\Lambda\|_0}{D} \,,\; p_{IP} = 1 - \frac{\|\Lambda\|_0 + \sum_{j=1}^{J} \|\mathcal{F}^{(j)}\|_0}{D} \,. \tag{9}$$

For SP, $\Lambda \in \mathbb{R}^D$ denotes the network's parameters, whereas $\Lambda \in \mathbb{R}^D$ contains all parameters except the FBs themselves in the IP setting. Thus, Λ has exactly the same number of elements for IP and SP. The pruning rates Eq. (9) are the fractions of parameters being equal to zero. To have a fair comparison between IP and SP, we normalize the number of non-zero parameters with the total count of coefficients in the standard dense network, *i.e.* the dense network without FBs. The number of bias and batch normalization parameters is tiny compared to convolutional and fully connected layers. Also, all parameters of FBs together are at most 0.01% of D in our experiments. Consequently, we only prune weights of fully connected layers as well as spatial- and FB coefficients of convolutional layers. FBs, bias and batch normalization parameters are all trained.

Algorithm 1 FB 2D Convolution with IP

```
1:  instance variables                                         ▷ of IP_FB_2DConv
2:      filter_basis: $\{g^{(1)},\dots,g^{(K^2)}\} \subset \mathbb{R}^{K\times K}$
3:      fb_coefficients: $(\lambda_n^{(\alpha,\beta)})_{\alpha,\beta,n} \in \mathbb{R}^{c_{out}\times c_{in}\times K^2}$
4:      pruning_mask: $(\mu_n^{(\alpha,\beta)})_{\alpha,\beta,n} \in \{0,1\}^{c_{out}\times c_{in}\times K^2}$
5:      conv_args      ▷ e.g. stride, padding, groups, ...
6:  def FORWARD_PASS($X$)                    ▷ input $X \in \mathbb{R}^{c_{in}\times h\times w}$
7:      for all $\beta \in \{1,\dots,c_{in}\}, n \in \{1,\dots,K^2\}$ do
8:          $Z_n^{(\beta)} = \text{Conv2D}(g^{(n)}, X^{(\beta)}, \text{conv\_args})$
9:      for all $\alpha \in \{1,\dots,c_{out}\}$ do
10:         $Y^{(\alpha)} = \sum_{\{(\beta,n):\mu_n^{(\alpha,\beta)}=1\}} \lambda_n^{(\alpha,\beta)} \cdot Z_n^{(\beta)}$
11:     return $Y = (Y^{(\alpha)})_{\alpha=1}^{c_{out}}$
```

Computational cost comparison. As discussed, parameter costs for IP with our FB sharing schemes are only negligibly bigger than for SP. By the linearity of convolutions, the sparsity of filters in the interspace can be used to reduce computational costs, see Eq. (8). In Appendix Sec. D, computational costs are calculated and compared for IP and SP. Costs are measured by the number of theoretically required *floating point operations* (FLOPs) for a convolutional layer and are independent of the used FB sharing scheme. IP's overhead is composed of additional costs in the forward and backward pass. For inference, only the additional cost of the forward pass counts. Both, SP and IP, need specialized soft- and hardware that supports sparse computations to actually reduce runtime.

Assume a layer with kernel size $K \times K$, c_{in} input and c_{out} output channels. In the forward pass, SP has $1-p$ times the FLOPs cost of the dense layer. Due to l. 7-8 in Alg. 1, IP has a constant overhead ${K^2}/{c_{out}}$. In total, IP has $1 - p + {K^2}/{c_{out}}$ times the FLOPs cost of the dense layer.

In the backward pass, the number of FLOPs for IP is in $\mathcal{O}\left(cost\left(\frac{\partial \mathcal{L}}{\partial h}\right)\right)$, *i.e.* comparable to the cost of computing the dense gradient of layer h in spatial representation.

As discussed, IP needs more computations for inference than SP for equal sparsity. However, since IP finds superior sparse models, IP actually achieves *a higher speed up* in real time measurements than SP while reaching similar or even better performance, as will be shown Fig. 5(a).

Pruning methods. Algorithm 1 describes sparse FB 2D convolutions with IP in pseudo code. Since automatic differentiation is standard in modern deep learning frameworks, backpropagation formulas for FB convolutions are computed automatically and are not included in Alg. 1. The FB in Alg. 1 might be shared over several layers, see Sec. 4.3. Our experiments in Sec. 5 compare SP and IP on various sparse training and other pruning methods, namely:

DST randomly prunes the model at initialization. During training, unimportant coefficients are pruned based on their magnitude. In each layer, the same number of parameters is *regrown* by activating their gradients. SET regrows coefficients randomly whereas RigL regrows those with high gradient magnitude. The pruning mask is updated each $1,500$ iterations for SET and $4,000$ for RigL. A cosine schedule is used to reduce the number of pruned/regrown coefficients.

LT pre-trains the network for $t_0 = 500$ steps. Then, the network is trained to convergence. Now, 20% of the non-zero coefficients are pruned based on their magnitude. The un-pruned part of the CNN is reset to its value at t_0. The whole procedure is applied k times in total until the desired pruning rate $p = 1 - 0.8^k$ is reached. Ultimately, the final sparse network is trained, starting at t_0.

PaI prunes the model at initialization without pre-training or changing the pruning mask during training. Random PaI prunes weights i.i.d. with probability p. SNIP trains coefficients which have high influence on changing the loss $\mathcal{L}$ when training starts. GraSP finds coefficients which improve the gradient flow at the beginning of training most. SynFlow keeps coefficients with high information throughput which is measured by their influence on the total *path norm* of the sparse network.

Gradual Magnitude Pruning (GMP) [17] starts training with dense coefficients. During training, the CNN is gradually sparsified based on the coefficients' magnitudes. Pruned parameters are fixed at zero, thus never regrow.

Fine-Tuning (FT) [59] uses a pre-trained network. The $p \cdot D$ coefficients with smallest magnitude are pruned. The pre-trained coefficients of the sparse CNN are fine-tuned with the learning rate schedule of the dense training.

All these methods were developed for SP. Yet, in our experiments they are applied unchanged to the interspace setting. For more details see Appendix Secs. F and G.

5. Experiments and discussion

Section 5.1 covers the experimental setup. Next, Sec. 5.2 compares the three SOTA PaI methods [35, 66, 71] for IP and SP. In Sec. 5.3, we discuss IP and SP for more sophisticated sparse training methods, namely LTs [15] and the DST methods SET [49] and RigL [13]. Furthermore, we show that IP also improves SP on classical pruning methods applied during training, GMP [17], and on pre-trained models, FT [59]. Improved trainability and generalization ability of IP compared to SP is shown and discussed in Sec. 5.4.

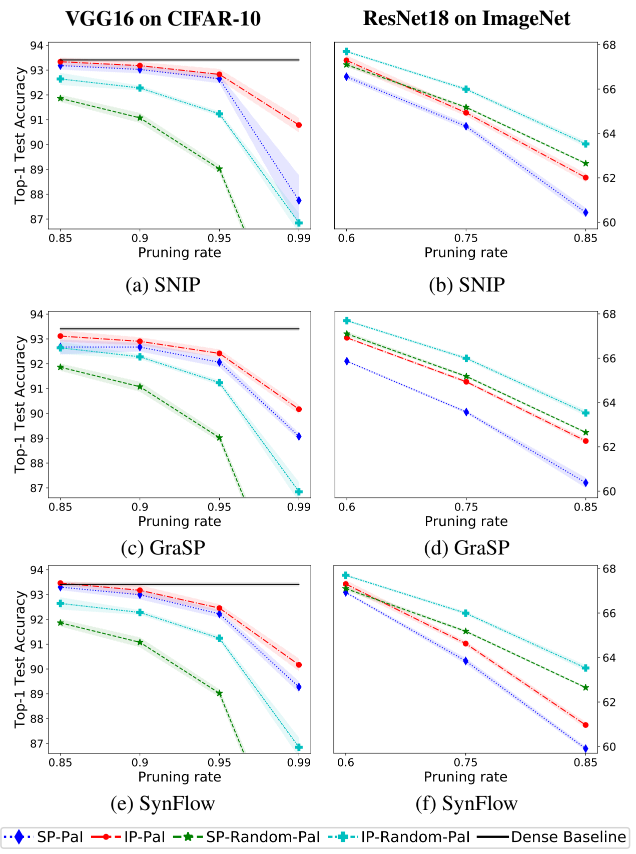
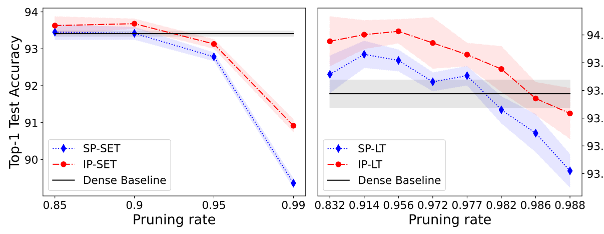


Figure 3. Comparing SP and IP for PaI methods, SNIP, GraSP and SynFlow together with random PaI for CIFAR-10 and ImageNet.

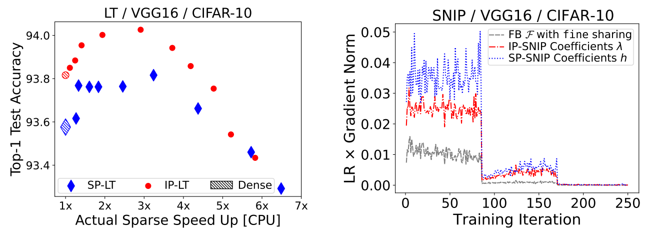
5.1. Experimental setup

We compare IP and SP for a VGG16 [63] on CIFAR-10 [32] and ResNets 18 and 50 [26] on ImageNet ILSVRC2012 [60]. Models are trained with cross entropy loss. We report mean and std of five runs for CIFAR-10 and three for ImageNet. Weight decay is applied on coefficients but not on FBs. Coefficients of 3×3 filters and their FBs are trained jointly, whereas fixed FBs $\mathcal{F} = \mathcal{B}$ are used for 1×1 filters. For ResNet18 we fix the FB $\mathcal{F} = \mathcal{B}$ for the 7×7 convolution whereas the 7×7 FB is trained for ResNet50. We use `medium` FB sharing for CIFAR-10 experiments, `fine` for ResNet50 and `coarse` sharing for all 3×3 convolutions for the ResNet18 on ImageNet. For SP and dense baselines, standard CNNs are used. As common in the literature, we report ImageNet results on the validation set. Note, we use training schedules intended for the corresponding SP method for both, SP and IP. In particular, FBs are trained without optimized hyperparameters. Thus, they use the same learning rate as all parameters. More details on hyperparameters, evaluation and used CNN archi-



(a) SET for VGG16 on CIFAR-10 (b) LT for VGG16 on CIFAR-10

Figure 4. Comparison between SP and IP for (a) the DST method SET and (b) LT on a VGG16 trained on CIFAR-10.



(a) (b)

Figure 5. VGG16 on CIFAR-10: (a) Top-1 test accuracy over real time acceleration for IP- and SP-LT. (b) Gradient L_2 norm $\times$ learning rate (LR) for SP- and IP-SNIP for layer 1 and $p = 0.85$.

tectures are given in Secs. H and I in the Appendix.

5.2. Pruning at initialization methods

Figure 3 compares SP and IP for PaI methods SNIP [35], GraSP [71] and SynFlow [66] together with random PaI for a VGG16 on CIFAR-10 and a ResNet18 on ImageNet.

The experiments show that pruning FB coefficients instead of spatial parameters leads to significant improvements in top-1 test accuracy while having the same memory costs. This holds true for all PaI methods, pruning rates and for high p in particular. In comparison to CIFAR-10, IP improves results on ImageNet even more. However, the three methods SNIP, GraSP and SynFlow are all outperformed by random PaI for ResNet18 on ImageNet. This demonstrates that these methods perform well for smaller datasets but show inferior results for small networks on big scale datasets like ImageNet. Still, as discussed earlier, the use of IP significantly improves all PaI methods, including random PaI. Section 5.3 shows that IP benefits from a stronger underlying pruning method to improve results further.

Despite optimizing FBs in addition to FB coefficients, IP does not induce instability compared to SP, see Fig. 5(b) and standard deviations in Fig. 3. In Appendix Sec. D.3, we show that the upper bounds for the gradient norms of FBs $\frac{\partial \mathcal{L}}{\partial \mathcal{F}}$ and FB coefficients $\frac{\partial \mathcal{L}}{\partial \lambda}$ are both determined by $\|\frac{\partial \mathcal{L}}{\partial h}\|$. This boundedness of the gradients leads to stable convergence for both, $\mathcal{F}$ and λ, while the convergence behavior of λ and the standard coefficients h is similar, see Fig. 5(b).

5.3. DST, LTs and classical pruning methods

For SP, more expensive or sophisticated methods like LT and DST improve sparse training results compared to PaI. We want to analyze whether this also applies to the IP setting. Furthermore, we want to check if IP boosts the SOTA methods LT and RigL as well. Finally, we benchmark IP and SP on various SOTA unstructured pruning methods for a ResNet50 on ImageNet.

DST and LT on CIFAR-10. IP improves DST and LTs significantly, see Figs. 4(a) and (b). For all p, IP-LT surpasses SP-LT. IP needs to train 3.7 times less parameters ($p = 0.977$) than SP to reach SP's best result for $p = 0.914$.

Top-1 Accuracy for ResNet50 on ImageNet			
Method	$p = 0.0$	$p = 0.8$	$p = 0.9$
SP-FT	77.15 ± 0.04	77.02 ± 0.03	75.67 ± 0.09
IP-FT	$\mathbf{77.30 \pm 0.04}$	$\mathbf{77.18 \pm 0.01}$	$\mathbf{75.89 \pm 0.09}$
SP-GMP	76.64 ± 0.06	75.37 ± 0.01	73.57 ± 0.06
IP-GMP	$\underline{77.16 \pm 0.04}$	$\underline{75.71 \pm 0.05}$	$\underline{74.20 \pm 0.07}$
SP-RigL	77.15 ± 0.04	75.75 ± 0.10	73.88 ± 0.06
IP-RigL	$\mathbf{77.30 \pm 0.04}$	$\underline{76.03 \pm 0.08}$	$\underline{74.32 \pm 0.11}$

Table 1. ResNet50 trained on ImageNet for 100 epochs.

IP-LT matches the dense baseline while training only 1.4% of its parameters and outperforms it for all $p \leq 0.98$. Comparable results hold for SET. IP-SET improves the dense baseline for $p \leq 0.9$, whereas SP-SET only matches it. Similar to PaI, IP-SET greatly exceeds SP-SET for high p. Comparing Figs. 3 and 4 shows that spending more effort in finding the sparse architecture (LT) or adapting it during training (SET) improves performance compared to PaI for both, SP and IP.

ResNet50 on ImageNet. Table 1 compares IP and SP on the SOTA pruning methods RigL [13], GMP [17] and FT [59]. As shown, IP outperforms all underlying SP methods for a ResNet50 on ImageNet. Results are significantly improved with interspace representations even though more than 50% of the coefficients of a ResNet50 are 1×1 convolutions which are equivalent for IP and SP. For example, IP-FT has similar performance as a standard dense model while training only 20% of its parameters. Note, using FBs does not only boost training sparse CNNs but dense training too, which will be discussed in more detail in Sec. 5.4.

Computational costs. Up to now, IP and SP were compared for equal memory costs. As analyzed in Sec. 4.4, IP has a small computational overhead compared to SP for equal sparsity. In applications, the actual runtime is more important than the theoretically required FLOPs. Thus, we compare the performance of IP and SP w.r.t. the actual acceleration on a CPU achieved by using sparse representations. Details on the implementation are provided in the Ap-

Method	VGG16 on CIFAR-10			
	$p = 0.85$		$p = 0.99$	
	Train	Test	Train	Test
SP-SET	99.85	93.45	94.20	89.36
IP-SET	99.89	**93.63**	**96.89**	**90.92**
SP-SNIP	99.94	93.18	93.96	87.75
IP-SNIP	99.96	**93.34**	**98.38**	**90.79**
	ResNet50 on ImageNet			
	$p = 0.8$		$p = 0.9$	
SP-RigL	74.64	75.75	71.30	73.88
IP-RigL	**75.39**	**76.03**	**72.08**	**74.32**

Table 2. Generalization gaps for various pruning methods.

	Pruning rate p			
Method	0.0	0.35	0.6	0.85
	SNIP			
SP	93.4 ± 0.1	93.4 ± 0.1	93.3 ± 0.2	93.2 ± 0.2
IP-`coarse`	$\mathbf{93.9 \pm 0.2}$	$\mathbf{93.7 \pm 0.2}$	$\mathbf{93.8 \pm 0.1}$	$\mathbf{93.5 \pm 0.0}$
IP-`medium`	$\mathbf{93.9 \pm 0.2}$	93.6 ± 0.2	93.7 ± 0.2	93.3 ± 0.2
IP-`fine`	93.7 ± 0.1	93.3 ± 0.2	93.3 ± 0.2	93.2 ± 0.1
	SET			
SP	93.4 ± 0.1	93.5 ± 0.2	93.3 ± 0.2	93.5 ± 0.2
IP-`coarse`	$\mathbf{93.9 \pm 0.1}$	$\mathbf{93.9 \pm 0.2}$	$\mathbf{93.8 \pm 0.1}$	$\mathbf{93.7 \pm 0.2}$
IP-`medium`	$\mathbf{93.9 \pm 0.1}$	93.7 ± 0.2	$\mathbf{93.8 \pm 0.1}$	93.6 ± 0.2
IP-`fine`	93.7 ± 0.1	93.6 ± 0.2	93.6 ± 0.2	93.6 ± 0.2

Table 3. Varying FB sharing schemes for lower pruning rates p.

pendix Sec. D.4. IP indeed has a longer runtime for equal sparsity due to the mentioned extra computations. However, by boosting performance of sparse models, IP reaches similar results than dense training with 5.2 times speed up and better results than SP for equal runtime, see Fig. 5(a).

5.4. Generalization and trainability

We consider generalization as the ability to correctly classify unseen data [44]. In this context a major aspect is the relationship between performance on the train and test set. Ideally, the performance on the train set should be optimal and a strong indicator for the performance on the test set. The *generalization gap* is the difference between train and test accuracy. Generalization can be improved by regularizations [6, 28, 33, 64, 78], enabling the model to use geometrical prior knowledge about the scene [7, 8, 29, 56, 57], shifting the model back to an area where it generalizes well [43, 45, 58, 62] but also by pruning the network [3, 24, 34].

Table 2 shows training and test accuracy for the IP- and SP versions of SET and SNIP for a VGG16 on CIFAR-10 as well as RigL for a ResNet50 on ImageNet. IP pruned networks train better than SP pruned ones for all p. Note, the used ImageNet training is highly regularized. Thus, the test accuracy is *higher* than the train accuracy. For ImageNet and $p = 0.99$ on CIFAR-10, IP has a bigger generalization gap than SP. This is due to a much better training accuracy for IP, which in the end leads to an improved test accuracy. However, IP has a smaller generalization gap than SP for $p = 0.85$ on CIFAR-10 where the model overfits.

Table 3 further shows that IP can generally improve results for pruning rates where training overfits. Note, $p = 0$ is dense training and SP for $p = 0$ is standard dense training. Improved performance in the dense setting can not be explained by IP's superior expressiveness (Thm. 1) since IP and SP can represent the same if all parameters are unpruned. We hypothesize that correlating filters in a CNN via FB sharing regularizes training, thereby improving generalization. One indicator of this is the fact that correlating *all* filters via `coarse` sharing shows the best results while `fine` sharing has comparable results to SP. Consequently, interspace representations can also be used to regularize dense training even for ResNet50 on ImageNet, see Tab. 1. After training, dense interspace representations can be converted to standard ones to reduce computational costs for inference. By optimizing weight decay and initialization schemes, IP's performance can be increased even further, as shown in Appendix Sec. B.

6. Conclusions and directions for future work

IP significantly improves results compared to pruning spatial coefficients. We demonstrate this by achieving SOTA results with the application of IP to SOTA standard PaI, LT, DST as well as classical pruning methods.

Theorem 1 proofs that IP leads to better sparse approximations than SP. Especially, IP generates models with *higher sparsity and equal performance* than SP. Also, FB representations combined with FB sharing *improve generalization* of overfitting CNNs, even for dense training. This comes with the prize of a small computational overhead for inference and additional gradient computations during training. Nevertheless, we show that sparse interspace representations accelerate dense baselines more than SP while keeping or even improving the baseline's performance.

We believe that IP can be enhanced by adapting more advanced strategies of SDL to the joint training of $\mathcal{F}$ and λ. Adapting IP to structured pruning is an option to maintain the network's accuracy while reducing inference time for arbitrary soft- and hardware. Combining IP with low rank tensor approximations lowers computational costs as well and is discussed in Appendix Secs. B and D. The interspace representation is an adaptive basis transformation of a finite dimensional vector space. Therefore, FBs $\mathcal{F}$ are not limited to represent convolutional filters but can express arbitrary vectors, like columns or small blocks of a matrix. This makes the concept of IP available for MLPs or self-attention modules.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. 3

[2] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, 13(3):1–18, 2017. 2

[3] Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. In *Advances in Neural Information Processing Systems 33*, 2020. 1, 8

[4] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag. What is the state of neural network pruning? In *Proceedings of Machine Learning and Systems 2*, 2020. 1

[5] Miguel A. Carreira-Perpinan and Yerlan Idelbayev. "Learning-compression" algorithms for neural net pruning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 3

[6] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13*, 2000. 8

[7] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016. 8

[8] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. Spherenet: Learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision*, 2018. 8

[9] Pau de Jorge, Amartya Sanyal, Harkirat Behl, Philip Torr, Grégory Rogez, and Puneet K. Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. In *International Conference on Learning Representations*, 2021. 1, 3

[10] Tim Dettmers and Luke Zettlemoyer. Sparse networks from scratch: Faster training without losing performance. *CoRR*, abs/1907.04840, 2019. 3

[11] Erich Elsen, Marat Dukhan, Trevor Gale, and Karen Simonyan. Fast sparse convnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[12] K. Engan, S. O. Aase, and J. H. Husøy. Method of optimal directions for frame design. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 5*, 1999. 1, 3

[13] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1, 3, 6, 7

[14] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2018. 1, 3

[15] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1, 3, 6

[16] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Pruning neural networks at initialization: Why are we missing the mark? In *International Conference on Learning Representations*, 2021. 3

[17] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019. 3, 6, 7

[18] Trevor Gale, Matei Zaharia, Cliff Young, and Erich Elsen. Sparse gpu kernels for deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020. 1

[19] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1

[20] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances in Neural Information Processing Systems 29*. 2016. 1, 3

[21] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016. 1, 3

[22] 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 28*. 2015. 1, 3

[23] Stephen Jose Hanson and Lorien Y. Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems 1*. 1989. 3

[24] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, 1992. 1, 8

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, 2015. 5

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 4, 6

[27] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. *Proceedings of the European conference on computer vision*, 2018. 2

[28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 8

[29] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, 2015. 8

[30] Steven A. Janowsky. Pruning versus clipping in neural networks. *Physical Review A*, 39:6600–6603, 1989. 1, 3

[31] Ehud D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990. 3

[32] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2012. `http://www.cs.toronto.edu/~kriz/cifar.html`. 4, 6

[33] Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems 4*. 1992. 8

[34] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*. 1990. 1, 3, 8

[35] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H.S. Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations*, 2019. 1, 3, 6, 7

[36] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 2017. 3

[37] Yawei Li, Shuhang Gu, Luc Van Gool, and Radu Timofte. Learning filter basis for convolutional neural network compression. In *IEEE International Conference on Computer Vision*, 2019. 3

[38] Zhengang Li, Geng Yuan, Wei Niu, Pu Zhao, Yanyu Li, Yuxuan Cai, Xuan Shen, Zheng Zhan, Zhenglun Kong, Qing Jin, Zhiyu Chen, Sijia Liu, Kaiyuan Yang, Bin Ren, Yanzhi Wang, and Xue Lin. Npas: A compiler-aware framework of unified network pruning and architecture search for beyond real-time mobile acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[39] Shiwei Liu, Lu Yin, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Do we actually need dense over-parameterization? In-time over-parameterization in sparse training. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. 1, 3

[40] Xingyu Liu, Jeff Pool, Song Han, and William J. Dally. Efficient sparse-winograd convolutional neural networks. In *International Conference on Learning Representations*, 2018. 3

[41] Zhenhua Liu, Jizheng Xu, Xiulian Peng, and Ruiqin Xiong. Frequency-domain dynamic pruning for convolutional neural networks. In *Advances in Neural Information Processing Systems 31*, 2018. 3

[42] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l0 regularization. In *International Conference on Learning Representations*, 2018. 3

[43] Julia Lust and Alexandru Paul Condurache. Gran: An efficient gradient-norm based detector for adversarial and misclassified examples. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020. 8

[44] Julia Lust and Alexandru Paul Condurache. A survey on assessing the generalization envelope of deep neural networks at inference time for image classification. *CoRR*, abs/2008.09381, 2020. 8

[45] Julia Lust and Alexandru Paul Condurache. Efficient detection of adversarial, out-of-distribution and other misclassified samples. *Neurocomputing*, 470:335–343, 2022. 8

[46] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010. 3

[47] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *Proceedings of the 37th International Conference on Machine Learning*, 2020. 1

[48] Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, and William J. Dally. Exploring the granularity of sparsity in convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017. 3

[49] Decebal Mocanu, Elena Mocanu, Peter Stone, Phuong Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9, 2018. 1, 3, 6

[50] Michael C. Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in Neural Information Processing Systems 1*. 1989. 1, 3

[51] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W. Keckler, and William J. Dally. Scnn. *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017. 1, 3

[52] Daniel S. Park, Yu Zhang, Chung-Cheng Chiu, Youzheng Chen, Bo Li, William Chan, Quoc V. Le, and Yonghui Wu. Specaugment on large scale datasets. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020. 1

[53] Shreyas Malakarjun Patil and Constantine Dovrolis. PHEW: Constructing sparse networks that learn fast and generalize well without training data. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. 3

[54] Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V. Le. Meta pseudo labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021. 1

[55] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[56] Matthias Rath and Alexandru Paul Condurache. Invariant integration in deep convolutional feature space. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020. 8

[57] Matthias Rath and Alexandru Paul Condurache. Improving the sample-complexity of deep classification networks with invariant integration. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2022. 8

[58] Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, 2019. 8

[59] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. 3, 6, 7

[60] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 6

[61] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green AI. *Communications of the ACM*, 63(12):54–63, 2020. 1

[62] Joan Serrà, David lvarez, Vicen Gmez, Olga Slizovskaia, Jos F. Nez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2020. 8

[63] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 4, 6

[64] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 8

[65] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 1

[66] Hidenori Tanaka, Daniel Kunin, Daniel L Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *Advances in Neural Information Processing Systems 33*, 2020. 1, 2, 3, 6, 7

[67] Yehui Tang, Yunhe Wang, Yixing Xu, Yiping Deng, Chao Xu, Dacheng Tao, and Chang Xu. Manifold regularized dynamic network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[68] W.F. Tinney and J.W. Walker. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proceedings of the IEEE*, 55(11):1801–1809, 1967. 3

[69] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations*, 2017. 1

[70] Stijn Verdenius, Maarten Stol, and Patrick Forré. Pruning via iterative ranking of sensitivity statistics. *CoRR*, abs/2006.00896, 2020. 3

[71] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *International Conference on Learning Representations*, 2020. 1, 2, 3, 6, 7

[72] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 1

[73] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[74] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. FreezeNet: Full performance by reduced storage costs. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2

[75] Paul Wimmer, Jens Mehnert, and Alexandru Condurache. COPS: Controlled pruning before training starts. In *International Joint Conference on Neural Networks*, 2021. 1, 3

[76] Huanrui Yang, Wei Wen, and Hai Li. DeepHoyer: Learning sparser neural network with differentiable scale-invariant sparsity measures. In *International Conference on Learning Representations*, 2020. 3

[77] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Proceedings of the European conference on computer vision*, 2020. 1

[78] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations*, 2017. 8

[79] Tao Zhuang, Zhixuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *Advances in Neural Information Processing Systems 33*, 2020. 2

[80] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. In *Advances in Neural Information Processing Systems 33*, 2020. 1