SPIN: An Empirical Evaluation on Sharing Parameters of Isotropic Networks

Chien-Yu Lin[1*†], Anish Prabhu[2*], Thomas Merth[2], Sachin Mehta[2], Anurag Ranjan[2], Maxwell Horton[2], and Mohammad Rastegari[2]

[1] University of Washington, USA
[2] Apple, Inc., USA

Abstract. Recent isotropic networks, such as ConvMixer and Vision Transformers, have found significant success across visual recognition tasks, matching or outperforming non-isotropic Convolutional Neural Networks. Isotropic architectures are particularly well-suited to cross-layer weight sharing, an effective neural network compression technique. In this paper, we perform an empirical evaluation on methods for sharing parameters in isotropic networks (SPIN). We present a framework to formalize major weight sharing design decisions and perform a comprehensive empirical evaluation of this design space. Guided by our experimental results, we propose a weight sharing strategy to generate a family of models with better overall efficiency, in terms of FLOPs and parameters versus accuracy, compared to traditional scaling methods alone, for example compressing ConvMixer by $1.9\times$ while improving accuracy on ImageNet. Finally, we perform a qualitative study to further understand the behavior of weight sharing in isotropic architectures. The code is available at https://github.com/apple/ml-spin.

Keywords: Parameter Sharing, Isotropic Networks, Efficient CNNs

1 Introduction

Isotropic neural networks have the property that all of the weights and intermediate features have identical dimensionality, respectively (see Figure 1). Some notable convolutional neural networks (CNNs) with isotropic structure [15, 25] have been proposed recently in the computer vision domain, and have been applied to different visual recognition tasks, including image classification, object detection, and action recognition. These isotropic CNNs contrast with the typical "hierarchical" design paradigm, in which spatial resolution and channel depth are varied throughout the network (e.g., VGG [19] and ResNet [6]).

The Vision Transformer (ViT) [3] architecture also exhibits this isotropic property, although softmax self-attention and linear projections are used for

---

(a) Architecture of regular CNNs.

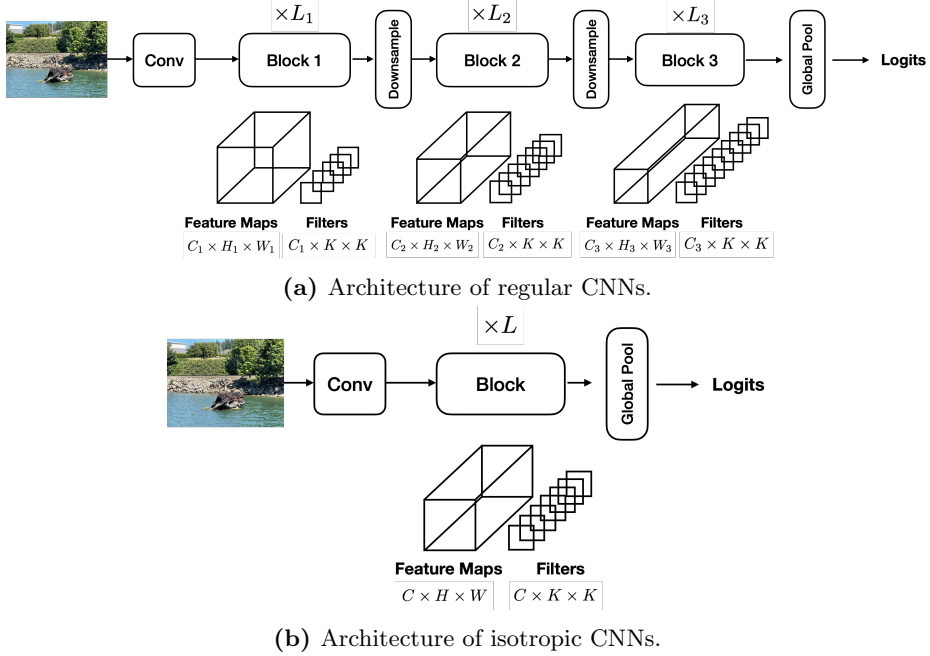**(b)** Architecture of isotropic CNNs.

Fig. 1: Basic architectures of regular and isotropic CNNs. (a) Regular CNNs vary the shape of intermediate features and weight tensors in the network while (b) isotropic CNNs fix the shape of all intermediate features and weight tensors in the network.

feature extraction instead of spatial convolutions. Follow-up works have experimented with various modifications to ViT models (e.g. replacing softmax self-attention with linear projections [24], factorized attention [27], and non-learned transformations [26]); however, the isotropic nature of the network is usually retained.

Recent isotropic models (e.g., ViT [3], ConvMixer [25], and ConvNext [15]) attain state-of-the-art performance for visual recognition tasks, but are computationally expensive to deploy in resource constrained inference scenarios. In some cases, the parameter footprint of these models can introduce memory transfer bottlenecks in hardware that is not well equipped to handle large amounts of data (e.g. microcontrollers, FPGAs, and mobile phones) [13]. Furthermore, "over-the-air" updates of these large models can become impractical for continuous deployment scenarios with limited internet bandwidth. Parameter (or weight) sharing[‡], is one approach which compresses neural networks, potentially enabling the deployment of large models in these constrained environments.

[‡]We interchangeably use the terms parameter and weight sharing throughout this paper.

Isotropic DNNs, as shown Figure 1, are constructed such that a layer's weight tensor has identical dimensionality to that of other layers. Thus, cross-layer parameter sharing becomes a straightforward technique to apply, as shown in ALBERT [13]. On the other hand, weight tensors within non-isotropic networks cannot be shared in this straightforward fashion without intermediate weight transformations (to coerce the weights to the appropriate dimensionality). In Appendix A, we show that the search space of possible topologies for straightforward cross-layer parameter sharing is significantly larger for isotropic networks, compared to "multi-staged" networks (an abstraction of traditional, non-isotropic networks). This rich search space requires a comprehensive exploration. Therefore, in this paper, we focus on isotropic networks, with the goal of finding practical parameter sharing techniques that enable high-performing, low-parameter neural networks for visual understanding tasks. To extensively explore the weight sharing design space for isotropic networks, we experiment with different orthogonal design choices (Section 3.2). Specifically, we explore (1) different sharing topologies, (2) dynamic transformations, and (3) weight fusion initialization strategies from pretrained non-sharing networks. Our results show that parameter sharing is a simple and effective method for compressing large neural networks versus standard architectural scaling approaches (e.g. reduction of input image size, channel size, and model depth). Using a weight sharing strategy discovered from our design space exploration, we achieve nearly identical accuracy (to non-parameter sharing, iso-FLOP baselines) with significantly reduced parameter counts. Beyond the empirical accuracy versus efficiency experiments, we also investigate network representation analysis (Section 5) and model generalization (Appendix F) for parameter sharing isotropic models.

2   Related Works

Cross-layer Parameter Sharing. Cross-layer parameter sharing has been explored for both CNN- and Transformer-based models [2, 9, 11–13, 20, 21]. For instance, Kim et al. [9] applies cross-layer parameter sharing across an entire heterogeneous CNN. However, they share weights at the granularity of filters, whereas we share weights at the granularity of layers. In terms of our framework, Kubilius et al. [11] experiments with Uniform-Strided, proposing a heterogeneous network based off of the human visual cortex. With isotropic networks, we can decouple parameter sharing methods from the constraints imposed by heterogeneous networks. Thus, we expand the scope of weight sharing structures from their work to isotropic networks.

Cross-layer parameter sharing is explored for isotropic Transformer models for the task of neural language modeling [2, 13] and vision [21]. Lan et al. [13] experiments with Uniform-Sequential, and Dehghain et al. [2] experiments with universal sharing (i.e. all layers are shared). Takase et al. [21] experiments with 3 strategies, namely Uniform-Sequential, Uniform-Strided, and Cycle. In this paper, we extend these works by decomposing the sharing topology into com-
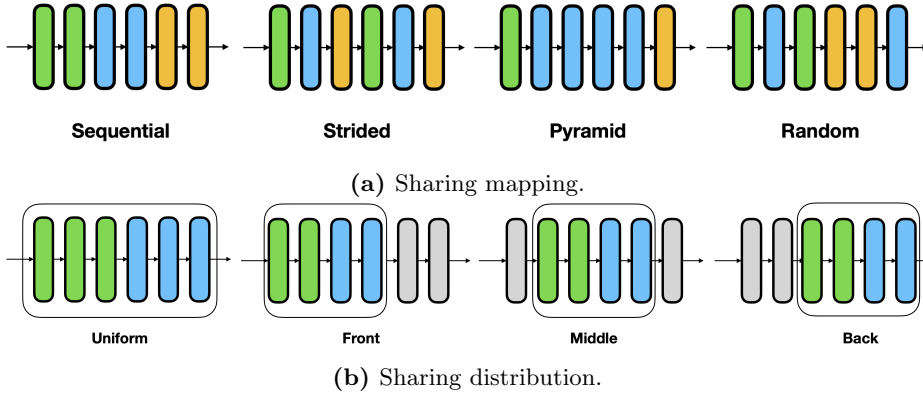
(a) Sharing mapping.

(b) Sharing distribution.

Fig. 2: Sharing topologies. In (a), sharing mapping determines which layers share the same weights while in (b), sharing distribution determines how the weight sharing layers are distributed in the network. Layers with the same color share weights. Layers outside of the sharing section do not share weights. Best viewed in color.

binations of different sharing mappings (Figure 2a) and sharing distributions (Figure 2b).

Dynamic Recurrence for Sharing Parameters. Several works [1, 4, 14, 18] explore parameter sharing through the lens of dynamically repeating layers. However, each technique is applied to a different model architecture, and evaluated in different ways. Thus, without a common framework, it's difficult to get a comprehensive understanding of how these techniques compare. While this work focuses only on static weight sharing, we outline a framework that may encompass even these dynamic sharing schemes. In general, we view this work as complementary to explorations on dynamic parameter sharing, since our analysis and results could be used to help design new dynamic sharing schemes.

3   Sharing Parameters in Isotropic Networks

In this section, we first motivate why we focus on isotropic networks for weight sharing (Section 3.1), followed by a comprehensive design space exploration of methods for weight sharing, including empirical results (Section 3.2).

3.1   Why Isotropic Networks?

Isotropic networks, shown in Figure 1b, are simple by design, easy to analyze, and enable flexible weight sharing, as compared to heterogeneous networks.

Simplicity of Design. Standard CNN architectural design, whether manual [6, 17]) or automated through methods like neural architecture search [7, 22]), require searching a complex search space, including what blocks to use, where and when to downsample the input, and how the number of channels should vary throughout the architecture. On the other hand, isotropic architectures form a much simpler design space, where just a single block (e.g., attention block in Vision Transformers or convolutional block in ConvMixer) along with network's depth and width must be chosen. The simplicity of implementation for these architectures enables us to more easily design generic weight sharing methods across various isotropic architectures. The architecture search space of these networks is also relatively smaller than non-isotropic networks, which makes them a convenient choice for large scale empirical studies.

Increased Weight Sharing Flexibility. Isotropic architectures provide significantly more flexibility for designing a weight sharing strategy than traditional networks.

We define the *sharing topology* to be the underlying structure of how weight tensors are shared throughout the network. Suppose we have an isotropic network with $L \geq 1$ layers and a weight tensor "budget" of $1 \leq P \leq L$. The problem of determining the optimal sharing topology can be seen as a variant of the set cover problem; we seek a set cover with no more than B disjoint subsets, which maximizes the accuracy of the resulting network. More formally, a possible sharing topology is an ordered collection of disjoint subsets $\mathcal{T} = (\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_P)$, where $\cup_{i=1}^{B} \mathcal{S}_i = \{1, 2, ..., L\}$ for some $1 \leq P \leq L$. We define $\frac{L}{P}$ to be the *share rate*.

We characterize the search space in Appendix A, showing that isotropic networks support significantly more weight sharing topologies than heterogeneous networks (when sharing at the granularity of weight tensors). This substantially increased search space may yield more effective weight sharing strategies in isotropic networks than non-isotropic DNNs, a reason why we are particularly interested in isotropic networks

Cross-layer Representation Analysis. To better understand if the weights of isotropic architectures are amenable to compression through weight sharing, we study the representation of these networks across layers. We hypothesize that layers with similar output representations will be more compressible via weight sharing. To build intuition, we use Centered Kernel Alignment (CKA) [10], a method that allows us to effectively measure similarity across layers.

Figure 3 shows the pairwise analysis of CKA across layers within the ConvMixer network. We find significant representational similarity for nearby layers. This is not unexpected, given the analysis of prior works on iterative refinement in residual networks [8]. Interestingly, we find that CKA generally peaks in the middle of the network for different configurations of ConvMixer. Overall, these findings suggest that isotropic architectures may be amenable to weight sharing, and we use this analysis to guide our experiments exploring various sharing topologies in Section 3.2.
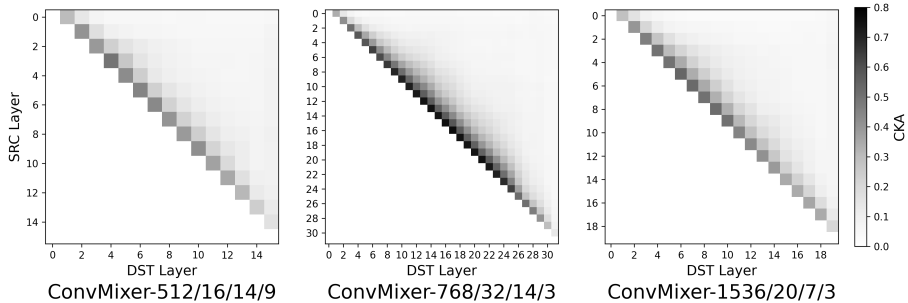
Fig. 3: CKA similarity analysis on ConvMixer's intermediate feature maps shows that the output feature maps of neighboring layers and especially the middle layers have the highest similarity. Here, we compute the CKA similarity of each layer's output feature maps. The diagonal line and the lower triangle part are masked out for clarity. The CKA for the diagonal line is 1 since they are identical. The CKA for the lower triangle is the mirror of the upper triangle. Best viewed on screen.

3.2 Weight Sharing Design Space Exploration

When considering approaches to sharing weights within a neural network, there is an expansive design space to consider. This section provides insights as well as empirical evaluation to help navigate this design space. We first consider the weight sharing topology. Then, we introduce lightweight dynamic transformations on the weights to increase the representational power of the weight-shared networks. Finally, we explore how to use the trained weights of an uncompressed network to further improve accuracy in weight-sharing isotropic networks. All experiments done in this section are based on a ConvMixer model with 768 channels, depth of 32, patch extraction kernel size of 14, and convolutional kernel size of 3.

Weight Sharing Topologies. Isotropic networks provide a vast design space for sharing topologies. We perform an empirical study of various sharing topologies for the ConvMixer architecture, evaluated on the ImageNet dataset. We characterize these topologies by the (1) *sharing mapping* (shown in Figure 2a), which describes the structure of shared layers, and (2) the *sharing distribution* (shown in Figure 2b), which describes which subset of layers sharing is applied to. We study the following sharing mappings:

1. **Sequential**: Neighboring layers are shared in this topology. There is motivated by our cross layer similarity analysis in Section 3.1 and Figure 3, which suggest that local structures of recurrence may be promising.
2. **Strided**: This topology defines the recurrence on the network level rather than locally. If we consider having P blocks with unique weights, we first run all of the layers sequentially, then we repeat this whole structure L/P times.

Network	Sharing Distribution	Sharing Mapping	Params (M)	FLOPs (G)	Top-1 Acc (%)
ConvMixer	-	-	20.46	5.03	75.71
WS-ConvMixer	Uniform	Sequential Strided Random	11.02	5.03	**73.29** 72.80 Diverged
WS-ConvMixer	Middle	Sequential Pyramid	11.02	5.03	73.14 **73.22**
WS-ConvMixer	Front Back	Sequential	11.02	5.03	**73.31** 72.35

Table 1: Effect of different sharing distributions and mappings on the performance of weight-shared (WS) ConvMixer with a share rate of 2. In order to maintain the fixed share rate 2 for non-uniform sharing distributions (i.e., Middle, Front and Back), we apply sharing to 8 layers with share rate $3\times$ and have 16 independent layers. For Middle-Pyramid, the network is defined as $[4 \times 1, 1 \times 2, 2 \times 3, 2 \times 4, 2 \times 3, 1 \times 2, 4 \times 1]$, where for each element $N \times S$, N stands for the number of sharing layers and S the share rate for the layer. All experiments were done with a ConvMixer with 768 channels, depth of 32, patch extraction kernel size of 14, and convolutional kernel size of 3.

3. **Pyramid**: This topology is an extension of Sequential, which has increasingly more shared sequential layers as you approach the center of the network. This is inspired by (1) empirical results in Figure 3 that show a similar structure in the layer-wise similarity and (2) neural network compression methods (e.g. quantization and sparsity methods), which leave the beginning and end of the network uncompressed [5, 16].

4. **Random**: We randomly select which layers are shared within the network, allowing us to understand how much the choice of topology actually matters.

For the sharing distribution, we consider applying (1) **Uniform**, where sharing mapping is applied to all layers, (2) **Front**, where sharing mapping is applied to the front of the network, (3) **Middle**, where sharing mapping is applied to the middle of the network, and (4) **Back**, where sharing mapping is applied to the back of the network. Note that front, middle and back sharing distributions results in a non-uniform distribution of share rates across layers.

Figure 2 visualizes different sharing topologies while Table 1 shows the results of these sharing methods on the ImageNet dataset. When share rate is 2, ConvMixer with uniform-sequential, middle-pyramid, and front-sequential sharing topology result in similar accuracy (2.5% less than the non-shared model) while other combinations result in lower accuracy. These results are consistent with the layer-wise similarity study in Section 3.1, and suggests that layer-wise similarity may be a reasonable metric for determining which layers to share.

Because of the simplicity and flexibility of *uniform-sequential* sharing topology, we use it in the following experiments unless otherwise stated explicitly.

Lightweight Dynamic Transformations on Shared Weights. To improve the performance of a weight shared network, we introduce lightweight dynamic transformations on top of the shared weights for each individual layer. With this, we potentially improve the representational power of the weight sharing network without increasing the parameter count significantly.

To introduce the lightweight dynamic transformation used in this study, we consider a set of N layers to be shared, with a shared weight tensor W_s. In the absence of dynamic transforms, the weight tensor W_s would simply be shared among all N layers. We consider $W_i \in \mathbb{R}^{C \times C \times K \times K}$ to be the weights of the i-th layer, where C is the channel size and K is the kernel size. With a dynamic weight transformation function f_i, the weights W_i at the i-th layer becomes

$$W_i = f_i(W_s) \tag{1}$$

The choose f_i to be a learnable lightweight affine transformation that allows us to transform the weights without introducing heavy computation and parameter overhead. Specifically, $f_i(W) = \mathbf{a} * W + \mathbf{b}$ applies a grouped point-wise convolution with weights $\mathbf{a} \in \mathbb{R}^{C \times G}$ and bias $\mathbf{b} \in \mathbb{R}^C$ to W, where G is the number of groups. The number of groups, $G \in [1, C]$, can be varied to modulate the amount of inter-channel mixing.

Table 2 shows the effect of different number of groups in the dynamic weight transformation on the performance and efficiency (in terms of parameters and FLOPs) of ConvMixer on the ImageNet dataset. As Table 2 shows, using $G = 64$, the dynamic weight transformation slightly improves accuracy by 0.07% (from 73.29% to 73.36%) with 7% more parameters (from 11.02M to 11.8M) and 11.9% more FLOPs (from 5.03G to 5.63G). Despite having stronger expressive power, dynamic weight transformation does not provide significant accuracy improvement with under 10% of overhead on number of weights and FLOPs and sometimes even degrading accuracy.

Initializing Weights from Pretrained Non-sharing Networks. Here we consider how we can use the weights of a pretrained, uncompressed network to improve the parameter shared version of an isotropic network. To this end, we introduce transformations on the original weights to generate the weights of the shared network for a given sharing topology. We define $V_j \in \mathbb{R}^{C \times C \times K \times K}$ to be the j-th pretrained weight in the original network, and $u_j \in \mathbb{R}^C$ to be the corresponding pretrained bias. The chosen sharing topology defines a disjoint set cover of the original network's layers, where each disjoint subset maps a group of layers from the original network to a single shared weight layer. Concretely, if the weight W_i is shared among S_i layers $\{i_1, i_2, ..., i_{S_i}\}$ in the compressed network, then we define $W_i = F_i(V_{i_1}, V_{i_2}, ..., V_{i_{S_i}})$, where we can design each F_i. We refer to F as the *fusion strategy*. In all experiments we propagate the gradient back to the original, underlying V_j weights. Importantly, F does not incur a cost at inference-time, since we can constant-fold this function once we finish training.

Network	Weight Transformation?	Group Rate	Params (M)	FLOPs (G)	Top-1 Acc (%)
ConvMixer	-	-	20.46	5.03	75.71
WS-ConvMixer	✗	-	11.02	5.03	73.29
	✓	1	11.05	5.04	72.87
	✓	16	11.20	5.17	73.20
	✓	32	11.40	5.31	73.14
	✓	64	11.80	5.63	**73.36**

Table 2: Effect of affine transformations on the performance of Weight Shared ConvMixer model with a sharing rate of 2. All experiments were done with a ConvMixer with 768 channels, depth of 32, patch extraction kernel size of 14, and convolutional kernel size of 3.

Network	Fusion Strategy	Params (M)	FLOPs (G)	Top-1 Acc (%)
ConvMixer	-	20.5	5.03	75.71
WS-ConvMixer	-	10.84	5.03	73.23
	Choose First			74.81
	Mean			74.91
	Scalar Weighted Mean			**75.15**
	Channel Weighted Mean			**75.15**
	Pointwise Convoulution			Diverged

Table 3: Effect of different fusion strategies (Section 3.2) on the performance of ConvMixer. All experiments were done with a ConvMixer with 768 channels, depth of 32, patch extraction kernel size of 14, and convolutional kernel size of 3. All weight sharing ConvMixer models share groups of 2 sequential layers.

One simple fusion strategy would be to randomly initialize a single weight tensor for this layer. Note that this is the approach we have used in all previous experiments. We empirically explore the following fusion strategies:

- **Choose First**: In this setup we take the first of the set of weights within the set: $W_i = F_i(V_{i_1}, V_{i_2}, ..., V_{i_S}) = V_{i_1}$. The choice of the first weight (V_{i_1}), rather than any other weight, is arbitrary. Training this method from scratch is equivalent to our vanilla weight sharing strategy.
- **Mean**: We take the average of all the weight tensors within the set, $W_i = \frac{1}{S_i} \sum_{k=1}^{S_i} V_{i_k}$ and $b_i = \frac{1}{S_i} \sum_{k=1}^{S_i} u_{i_k}$.
- **Scalar Weighted Mean**: Same as the average, except each weight tensor gets a learned scalar weighting, $W_i = \frac{1}{S_i} \sum_{k=1}^{S_i} \alpha_{i_k} V_{i_k}$, $\alpha_i \in \mathbb{R}$. We take a simple mean of the bias, just as in the Mean strategy. The idea here is to

provide the ability to learn more complex fusions, of which Choose First
strategy, and Mean are special cases.

- **Channel Weighted Mean**: Rather than a scalar per layer, each weight
  tensor has a learned scalar for every filter, $W_i = \frac{1}{S_i} \sum_{k=1}^{S_i} \vec{\alpha_i} V_{i_k}$, $\vec{\alpha_i} \in \mathbb{R}^C$.
  Again, we take a simple mean of the bias, just as the Mean strategy. This
  strategy should allow the model to choose filters from specific weight tensors,
  or learn linear combinations.
- **Pointwise Convolution**: In this transformation, a pointwise convolution
  is applied to each layers weights, that maps to the same size filter, $W_i = \frac{1}{S_i} \sum_{k=1}^{S_i} A_i * V_{i_k}$, $A_i \in \mathbb{R}^{C \times C}$. This should allow arbitrary mixing and per-
  mutations of the kernels of each layer.

Table 3 shows that the Channel Weighted Mean fusion strategy allows us to
compress the model by $2\times$ while maintaining the performance of original net-
work. Furthermore, in Section 5, we show that weight fusion strategies allow us
to learn representations similar to the original network.

4   Effect of Parameter Sharing on Different Isotropic Networks on the ImageNet dataset

We evaluate the performance of the parameter sharing methods introduced in
Section 3.2 on a variety of isotropic architectures. For more information on the
training set-up and details, see Appendix C.

4.1   Parameter Sharing for ConvMixer

Typically, when considering model scaling, practitioners often vary parameters
including the network depth, width, and image resolution, which scale the per-
formance characteristics of the model [23]. In Table 4, we show that weight
sharing models can significantly outperform baselines with the same FLOPs and
parameters generated through traditional scaling alone, for example improving
accuracy by roughly 10% Top-1 in some cases. We also show a full family of
weight sharing ConvMixer models across multiple architectures in Table 5, and
find that weight sharing can reduce parameters by over $2\times$ in many architectures
while maintaining similar accuracy. These results show that weight sharing, in
addition to typical scaling methods, is an effective axis for model scaling.

4.2   Parameter Sharing for Other Isotropic Networks

Although our evaluations have focused on ConvMixer, the methods discussed
in Section 3 are generic and can be applied to any isotropic model. Here, we
show results of applying parameter sharing to ConvNeXt [15] and the Vision
Transformer (ViT) architecture.

Network (C/D/P/K)	Resolution	Weight Sharing?	Share Rate	Params (M)	FLOPs (G)	Top-1 Acc(%)
768/32/14/3	224	✗	-	20.5	5.03	75.71
576/32/14/3	322	✗	-	11.8	5.92	70.326
768/16/14/3	322	✗	-	10.84	5.32	74.20
768/32/14/3	224	✓	2	11.02	5.03	**75.14**
384/32/14/3	448	✗	-	5.5	5.23	58.83
768/8/14/3	448	✗	-	6.04	5.38	68.31
768/32/14/3	224	✓	4	6.3	5.03	**71.91**
288/32/14/3	644	✗	-	3.25	6.23	40.46
768/4/14/3	644	✗	-	3.63	6.04	57.75
768/32/14/3	224	✓	8	3.95	5.03	**67.19**

Table 4: Weight sharing vs. model scaling for the ConvMixer model on ImageNet. For a fair comparison, we generate models with similar FLOPs and network parameters to our family of weight sharing models using traditional model scaling methods. Weight sharing methods achieve significantly better performance than traditional model scaling. See Table 5 for more details on the weight sharing model.

ConvNeXt. Table 6 shows the results of parameter sharing on the ConvNeXt isotropic architecture. With parameter sharing, we are able to compress the model by 2× while maintaining similar accuracy on the ImageNet dataset.

Vision Transformer (ViT). We also apply our weight sharing method to a Vision Transformer, a self-attention based isotropic network. Due to space limit, we report accuracy numbers in Appendix B. Furthermore, we discuss the differences between applying weight sharing methods to CNNs versus transformers.

4.3 Comparison with State-of-the-art Weight Sharing Methods.

Table 7 compares the performance of weight sharing methods discussed in Section 3.2 with existing methods [4, 9, 14] on ImageNet. Compared to existing methods, our weight sharing schemes are effective; achieving higher compression rate while maintaining accuracy. For example, ConvMixer-768/32, ConvMixer-156/20, and ConvNeXt-18 with weight sharing and weight fusion achieve 1.86x, 1.91x and 1.92 share rate while having a similar accuracy. Existing weight sharing techniques [4,9] can only achieve at most 1.58x and 1.45x share rate at while maintaining accuracy. Although [14] can achieve 12x share rate, it results in a 8.8% accuracy drop.

These results show that isotropic networks can achieve a high share rate while maintaining accuracy with simple weight sharing methods. The traditional pyramid style networks, while using complicated sharing schemes [4, 9, 14], the

Network (C/D/P/K)	Weight Sharing?	Share Rate	Weight Fusion?	Params (M)	FLOPs (G)	Top-1 Acc(%)
1536/20/7/3	✗	-	-	49.4		78.03
	✓	2	✓	25.8	48.96	**78.47**
	✓	4	✓	14		75.76
	✓	10	✗	6.9		72.27
768/32/14/3	✗	-	-	20.5		75.71
	✓	2	✓	11.02	5.03	**75.14**
	✓	4	✓	6.3		71.91
	✓	8	✓	3.95		67.19
512/16/14/9	✗	-	-	5.7		67.48
	✓	2	✓	3.63	1.33	**65.04**
	✓	4	✓	2.58		59.34
	✓	8	✗	2.05		54.25

Table 5: Weight sharing family of ConvMixer model on ImageNet. Significant compression rates can be achieved without loss in accuracy across multiple isotropic ConvMixer models. We also generate a full family of weight sharing models by varying the *share rate*, which is the reduction factor in number of unique layers for the weight shared model compared to the original. C/D/P/K represents the dimension of channel, depth, patch and kernel of the model. If *weight fusion* is specified, the channel weighted mean strategy described in Section 3.2 is used.

Network	Depth	Share Rate	Params (M)	FLOPs (G)	Top-1 Acc(%)
ConvNeXt	18	-	22.3	4.3	78.7
	9	-	11.5	2.2	75.3
WS-ConvNeXt	18	2	11.5		**78.07**
		4	6.7	4.3	76.11
		6	4.3		72.07
		9	3.1		68.75

Table 6: Effect of weight sharing on the ConvNeXt model on ImageNet. WS-ConvNeXxt has 2x less number of parameters but still achieves similar accuracy to the original ConvNeXt model.

share rate is usually limited. Note that although our sharing schemes can achieve higher share rates, existing methods like [9,14] are able to directly reduce FLOPs, which our method does not address.

Network	Share Rate	Params (M)	FLOPs (G)	Top-1 Acc(%)
ConvMixer-768/32 [25] (baseline)	-	20.5	5.03	75.71
WS-ConvMixer-768/32-S2 (ours)	**1.86**	**11.02**	5.03	**75.14**
ConvMixer-1536/20 [25] (baseline)	-	49.4	48.96	78.03
WS-ConvMixer-1536/20-S2 (ours)	**1.91**	**25.8**	48.96	**78.47**
ConvNeXt-18 [15] (baseline)	-	22.3	4.3	78.7
WS-ConvNeXt-18-S2 (ours)	**1.92**	**11.5**	4.3	**78.07**
WS-ConvNeXt-18-S4 (ours)	3.33	6.7	4.3	76.11
ResNet-152 [6] (baseline)	-	60	11.5	78.3
IamNN [14]	12	5	2.5-9	69.5
ResNet-101 [6] (baseline)	-	44.54	7.6	77.95
DR-ResNet-65 [4]	1.58	28.12	5.49	78.12
DR-ResNet-44 [4]	2.2	20.21	4.25	77.27
ResNet-50 [6] (baseline)	-	25.56	3.8	76.45
DR-ResNet-35 [4]	1.45	17.61	3.12	76.48
ResNet50-OrthoReg [9]	1.25	20.51	4.11	76.36
ResNet50-OrthoReg-SharedAll [9]	1.6	16.02	4.11	75.65

Table 7: Share rate and ImageNet accuracy comparison with existing weight sharing methods.

5 Representation Analysis

In this sections, we perform qualitative analysis of our weight sharing models to better understand why they lead to improved performance and how they change model behavior. To do this, we first analyze the representations learned by the original network, compared to one trained with weight sharing. We follow a similar set-up to Section 3.1. We use CKA as a metric for representational similarity and compute pairwise similarity across all layers in both the networks we aim to compare. In Figure 4(a) we first compare the representations learned by a vanilla weight sharing method to the representations of the original network. We find that there is no clear relationship between the representations learned. Once we introduce the weight fusion initialization strategy (Section 3.2), we find significant similarity in representations learned, as shown in Figure 4(b). This suggests that our weight fusion initialization can guide the weight shared models to learn similar features to the original network. In Appendix F, we further analyze the weight shared models and characterize their robustness compared to standard networks.
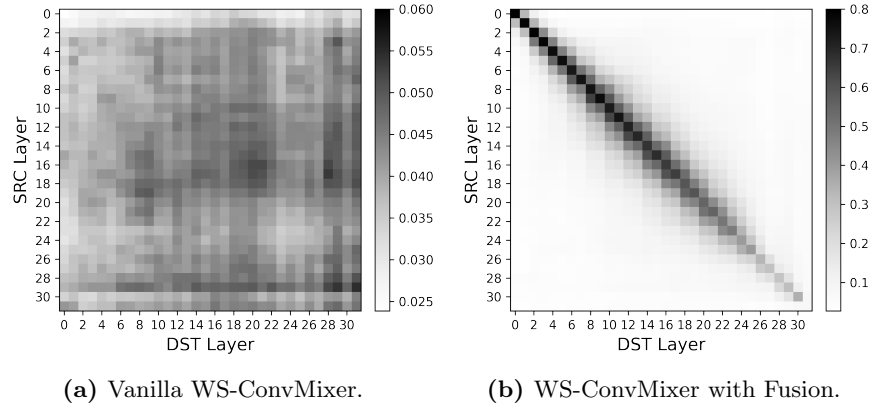
(a) Vanilla WS-ConvMixer.   **(b)** WS-ConvMixer with Fusion.

Fig. 4: (a) The CKA similarity analysis of a standard ConvMixer's intermediate feature maps compared to a vanilla weight shared ConvMixer, with share rate of 2. (b) The same analysis but compare to a weight shared ConvMixer initialized with weight fusion. The channel weighted mean fusion strategy is used (see Section 3.2).

6   Conclusion

Isotropic networks have the unique property in which all layers in the model have the same structure, which naturally enables parameter sharing. In this paper, we perform a comprehensive design space exploration of shared parameters in isotropic networks (SPIN), including the weight sharing topology, dynamic transformations and weight fusion strategies. Our experiments show that, when applying these techniques, we can compress state-of-the-art isotropic networks by up to 2 times without losing any accuracy across many isotropic architectures. Finally, we analyze the representations learned by weight shared networks and qualitatively show that the techniques we introduced, specifically fusion strategies, guide the weight shared model to learn similar representations to the original network. These results suggest that parameters sharing is an effective axis to consider when designing efficient isotropic neural networks.

References

1. Battash, B., Wolf, L.: Adaptive and iteratively improving recurrent lateral connections. CoRR **abs/1910.11105** (2019), http://arxiv.org/abs/1910.11105
2. Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., Kaiser, L.: Universal transformers. ArXiv **abs/1807.03819** (2019)
3. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

4. Guo, Q., Yu, Z., Wu, Y., Liang, D., Qin, H., Yan, J.: Dynamic recursive neural network. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5142–5151 (2019)
5. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. ArXiv **abs/1506.02626** (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
7. Howard, A.G., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H.: Searching for mobilenetv3. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 1314–1324 (2019)
8. Jastrzebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., Bengio, Y.: Residual connections encourage iterative inference. CoRR **abs/1710.04773** (2017), http://arxiv.org/abs/1710.04773
9. Kim, D., Kang, W.: Learning shared filter bases for efficient convnets. CoRR **abs/2006.05066** (2020), https://arxiv.org/abs/2006.05066
10. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.E.: Similarity of neural network representations revisited. ArXiv **abs/1905.00414** (2019)
11. Kubilius, J., Schrimpf, M., Hong, H., Majaj, N.J., Rajalingham, R., Issa, E.B., Kar, K., Bashivan, P., Prescott-Roy, J., Schmidt, K., Nayebi, A., Bear, D., Yamins, D.L.K., DiCarlo, J.J.: Aligning artificial neural networks to the brain yields shallow recurrent architectures (2018)
12. Kubilius, J., Schrimpf, M., Hong, H., Majaj, N.J., Rajalingham, R., Issa, E.B., Kar, K., Bashivan, P., Prescott-Roy, J., Schmidt, K., Nayebi, A., Bear, D., Yamins, D.L.K., DiCarlo, J.J.: Brain-like object recognition with high-performing shallow recurrent anns. CoRR **abs/1909.06161** (2019), http://arxiv.org/abs/1909.06161
13. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. In: ICLR. OpenReview.net (2020), http://dblp.uni-trier.de/db/conf/iclr/iclr2020.htmlLanCGGSS20
14. Leroux, S., Molchanov, P., Simoens, P., Dhoedt, B., Breuel, T.M., Kautz, J.: Iamnn: Iterative and adaptive mobile neural network for efficient image classification. CoRR **abs/1804.10123** (2018), http://arxiv.org/abs/1804.10123
15. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. arXiv preprint arXiv:2201.03545 (2022)
16. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: ECCV (2016)
17. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 4510–4520 (2018)
18. Shen, Z., Liu, Z., Xing, E.P.: Sliced recursive transformer. CoRR **abs/2111.05297** (2021), https://arxiv.org/abs/2111.05297
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2015)
20. Spoerer, C.J., Kietzmann, T.C., Mehrer, J., Charest, I., Kriegeskorte, N.: Recurrent networks can recycle neural resources to flexibly trade speed for accuracy in visual recognition. bioRxiv (2020). https://doi.org/10.1101/677237, https://www.biorxiv.org/content/early/2020/03/26/677237
21. Takase, S., Kiyono, S.: Lessons on parameter sharing across layers in transformers. CoRR **abs/2104.06022** (2021), https://arxiv.org/abs/2104.06022

22. Tan, M., Chen, B., Pang, R., Vasudevan, V., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2815–2823 (2019)
23. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)
24. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., Dosovitskiy, A.: Mlp-mixer: An all-mlp architecture for vision. CoRR **abs/2105.01601** (2021), https://arxiv.org/abs/2105.01601
25. Trockman, A., Kolter, J.Z.: Patches are all you need? CoRR **abs/2201.09792** (2022), https://arxiv.org/abs/2201.09792
26. Wang, G., Zhao, Y., Tang, C., Luo, C., Zeng, W.: When shift operation meets vision transformer: An extremely simple alternative to attention mechanism. CoRR **abs/2201.10801** (2022), https://arxiv.org/abs/2201.10801
27. Zhai, S., Talbott, W., Srivastava, N., Huang, C., Goh, H., Zhang, R., Susskind, J.M.: An attention free transformer. CoRR **abs/2105.14103** (2021), https://arxiv.org/abs/2105.14103