Supplementary Material for 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

A Weight Sharing Search Space Characterization

A.1 Isotropic Network Case

Suppose we have an L layer isotropic network and a weight tensor budget of $P \in \mathbb{Z}$, where $0 < P \leq L$ (recall that $\frac{L}{P}$ is the *share rate*). When building our network, we can choose between P different weight tensors for each layer (sampling with replacement), so there are P^L choices. Thus the search space for parameter sharing can be described as $\Omega(L, P) = P^L$.

We can define the size of the search space of topologies which use *exactly* P parameter tensors as $\tilde{\Omega}(L, P) = \Omega(L, P) - \Omega(L, P-1)$. This simply reduces the size of the original search space by the number of topologies which have up to $P-1$ shared weight tensors.

A.2 "Staged" Network Case

Suppose we have a network with L_N total layers, but with S discrete stages (or, more generally, disjoint subsets of layers), where the weight tensors have identical shape only to other weight tensors in their respective stage (or subset). This is a common paradigm for many popular CNN backbone architectures, such as ResNet [1], MobileNet [2], and DenseNet [3]. Without loss of generality, we refer to all disjoint subset architectures as staged architectures.

We simplify the following analysis by assuming each stage of the network has exactly L_S layers. Then we define $\Omega_S(L_N, L_S, P)$ to be the number of non-degenerate topologies for a staged network, staying below the P weight tensor budget:

$$\Omega_S(L_N, L_S, P) = \begin{cases} 0, L_N \leq 0 \vee P \leq 0 \\ \sum_{i=1}^{\min(L_S, P)} \binom{P}{i} \tilde{\Omega}(L_S, i) \Omega_S(L_N - L_S, L_S, P - i) \end{cases} \quad (1)$$

*Equal contribution.
†Work done while interning at Apple.

It can be shown numerically that $\Omega_S(L_N, L_S, P)$ increases rapidly as a function of L_S (see Figure 1 for a graphical representation). In other words, architectures closer to isotropic architectures support more options for parameter sharing, when the overall number of layers is held constant.

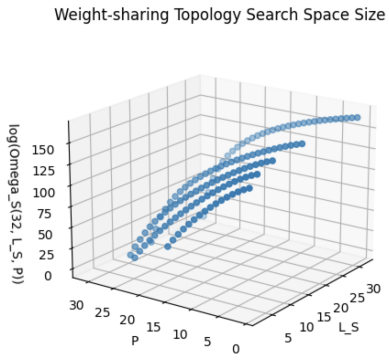


Fig. 1: Log plot of search space sizes for various L_S and P values for a depth $L_N = 32$ network. Recall that larger L_S values correspond to a "more isotropic" architecture.

B Weight Sharing in Vision Transformers (ViTs)

Table 1 shows results of parameter sharing for the DeiT architecture, which is a popular variant of ViT. We experiment with plain weight sharing and weight fusion techniques described in Section 3.2 on the DeiT-Ti and DeiT-S model, both of which have 12 layers.

Compared to the baseline, the WS-DeiT-S model is able to get similar accuracy, within 1 point Top-1 with half the parameters. At iso-parameters, the WS-DeiT models significantly outperform non-weight-sharing models. Among the weight sharing schemes, using weight fusion with pretrained weights can consistently boost accuracy. For example, using weight fusion can increase accuracy 0.55% on WS-Deit-Ti and 0.83% on WS-Deit-S compared to the plain weight sharing version.

C Training Details

We describe the training details we used to produce the experiments throughout this paper. To produce baseline accuracy of ConvMixer [6], DeiT [5] and ConvNeXt [4], we follow the default training settings described in each model's original paper and the released source code as closely as possible.

Model	Depth	Fusion Strategy	Share Rate	Params (M)	FLOPs (G)	ImgNet Acc(%)
Deit-Ti	12	-	-	5.72	1.26	72.55
	6	-	-	3.05	0.64	63.11
	4	-	-	2.16	0.44	55.61
WS-DeiT-Ti	12	-	2	3.05	1.26	68.07
			3	2.16	1.26	63.50
WS-DeiT-Ti	12	Mean	2	3.05	1.26	67.96
			3	2.16	1.26	**63.75**
WS-DeiT-Ti	12	Scalar Weighted Mean	2	3.05	1.26	**68.62**
			3	2.16	1.26	63.74
Deit-S	12	-	-	22.05	4.61	80.52
	6	-	-	11.40	2.33	74.48
	4	-	-	7.85	1.58	67.77
WS-DeiT-S	12	-	2	11.41	4.61	78.61
			3	7.87	4.61	76.67
WS-DeiT-S	12	Mean	2	11.41	4.61	**79.44**
			3	7.87	4.61	**77.11**

Table 1: DeiT Top-1 accuracy on ImageNet-1k with different weight fusion and share rates. All trained models are based off of DeiT-Ti (which has 12 transformer layers). WS-DeiT stands for the weights sharing version. All experiments use the **Sequential** sharing topology. The method with the clearly highest performance is bold-faced for each parameter regime.

We train all models on ImageNet-1K dataset without additional data. For ConvMixer, we use a learning rate of 0.01 and batch size of 64 for each GPU. The data augmentations we use includes RandAugment, MixUp, CutMix and random erasing. We use the AdamW optimizer with weight decay 2e-5 and a cosine learning rate schedule with a single cycle. For ConvMixer-1536/20/7/3, we train 150 epochs. For ConvMixer-768/32/14/3 and 512/16/14/9, we train for 300 epochs. For the Weight Sharing ConvMixer models, we use exact the same training setting as each corresponding baseline model to train. It is worth noting that we are able to show that the weight sharing ConvMixer can perform well without any parameter tuning, but this architecture may have a different optimal setting for these hyper parameters, for example due to having less parameters, and proper tuning may further boost performance of our method.

For DeiT and ConvNeXt, we follow the same settings proposed in the respective papers. All DeiT variants were trained with an effective batch size of 256 on 4 GPUs (note that the learning rate is scaled appropriately according to their scaling rule).

Network	Share BN	Share Bias	Share Dwise	Share Pwise	Params (M)	FLOPs (G)	ImgNet Acc(%)
ConvMixer	×	×	×	×	20.5	5.03	74.93
WS-ConvMixer	✓	✓	✓	✓	10.84	5.03	Diverged
	×	✓	✓	✓	10.89		73.17
	×	×	✓	✓	10.92		73.19
	×	×	×	✓	11.02		73.29

Table 2: Top-1 Accuracy on ImageNet ablating which operation within a ConvMixer Block are shared. We consider sharing the BatchNorm, Bias of convolutional layer, Depthwise Convolution, and Pointwise Convolution. The model size used in this comparison is 768/32/14/3 for channel/depth/patch size/kernel size. The sharing rate is 2. We apply no transformation or weight fusion in this study.

D Ablation on Sharing Different Components in ConvMixer Block

For the ConvMixer architecture, there are many components in each block we can choose whether to share. These components include Pointwise and Depthwise Convolution layer, bias for each Convolution layer, and the BatchNorm layer. In Table 2, we provide a full ablation study on sharing all the components, and gradually turn-off sharing on BatchNorm, Bias, and Depthwise Convolution, in order of the number of parameters each component contains.

As Table 2 shows, Pointwise Convolution layer contains the majority of the parameters of each block and only sharing Pointwise Convolution layer results in the best accuracy. Therefore, in our main study, we only share weights for Pointwise Convolution layers for ConvMixer models.

E Ablation on Weight Fusion Networks without Utilizing Pretrained Weights.

In Section 3.2, we described weight fusion methods to fuse a pretrained network's weights as initialization for a weight sharing model and showed accuracy improvement. Such weight fusion methods can also be applied without using a pretrained network's weights. To further understand the effect of the proposed weight fusion techniques, we perform an ablation study on applying the same weight fusion strategies with networks that are regularly initialized.

As results of the ablation study in Table 3 show, applying weight fusion to a randomly initialized model does not improve accuracy. It is worth noting that, after fusion, the number of effective weights will be the same as a vanilla weight sharing model, so there is no increase in representational power. This ablation study empirically shows that using the fused weights from a pretrained network

Network	Weight Init.	Weight Fusion	Params (M)	FLOPs (G)	Top-1 Acc (%)
ConvMixer	Regular	×	20.5	5.03	75.71
WS-ConvMixer	Regular	×	10.84	5.03	74.29
	Regular	Choose First			74.25
	Regular	Mean			74.25
WS-ConvMixer	Pretrained	Choose First	10.84	5.03	74.88
	Pretrained	Mean			75.28

Table 3: Ablations on whether using a pretrained network to initialize a weight sharing network when using weight fusion. 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. We used slightly different hyperparameters for this ablation study and have slightly higher accuracy for WS-ConvMixers.

to initialize a weight sharing model (see Section 3.2) is what leads to improved accuracy, rather than the weight sharing fusion alone.

F Further Model Analysis

In this section, we provide analysis to further understand why weight sharing is effective for isotropic architectures. All analysis below is performed on the ConvMixer architecture. We first analyze the robustness of these models, followed by further representation analysis with Centered Kernel Alignment (CKA).

F.1 Robustness to Label Noise

We analyze the robustness of our weight sharing model in the presence of label noise. Our analysis follows [7]. We first choose a noise level $l \in [0, 1]$. This corresponds to the fraction of training image labels to adjust. We then randomly choose a fraction l of images from the training set and set their label to a random category label. We then proceed with training as usual. Note that the labels are unchanged after their initial alteration at the beginning of training. We do not modify the evaluation set in any way. In Figure 2 we show that our weight sharing ConvMixer using the weight fusion method described in Section 3.2 is significantly more robust to noise than the baseline ConvMixer. We are able to exceed the baseline model in absolute Top-1 at all noise levels above zero, while halving the parameters of the model and iso-FLOPs. These results suggest that our weight sharing models provide increased robustness, and part of our empirical improvements in accuracy may be attributable to this property.

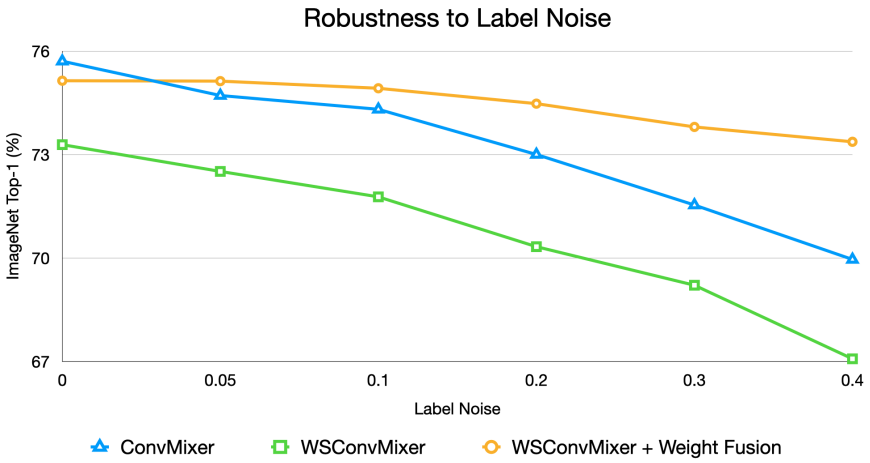


Fig. 2: Label noise analysis of ConvMixers. The model size used in this comparison is 768/32/14/3 for channel/depth/patch size/kernel size. At any label noise level above zero, the WSConvMixer + WeightFusion model outperforms the baseline ConvMixer, with half the parameters and iso FLOP. This suggests that our weight sharing method generates models that are more robust to noise, and this may be part of the reason we find empirically compelling results.

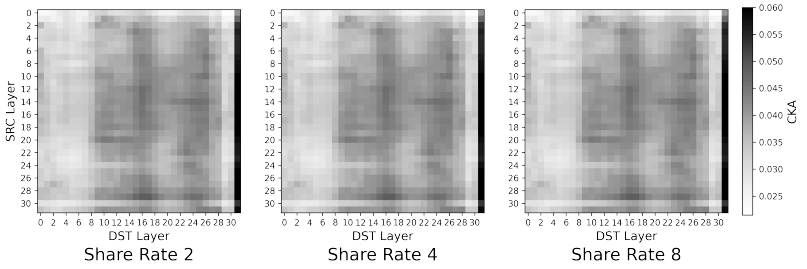
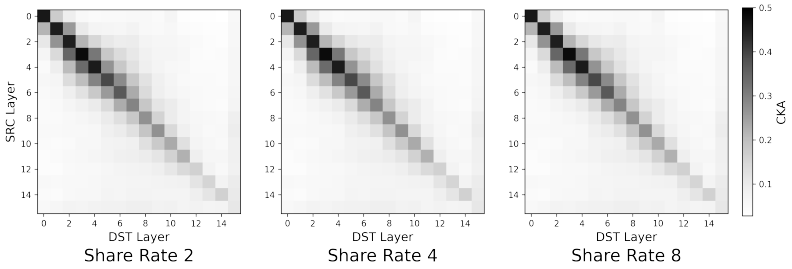


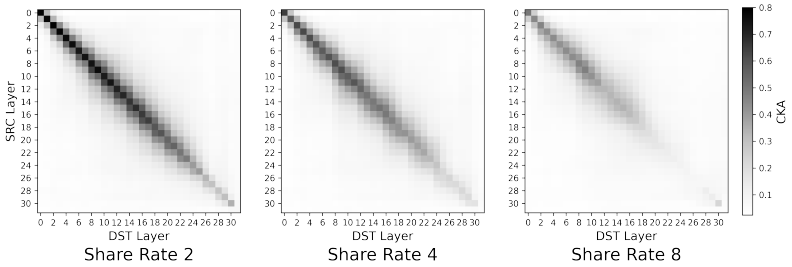
Fig. 3: CKA analysis on vanilla WS-ConvMixer-768/32/14/3 model. We compute pairwise analysis of WS-ConvMixer layer feature maps to the non-sharing ConvMixer model using CKA. For this WS-ConvMixer model, it has 73.29%, 70.11%, 66.31% accuracy on ImageNet for share rates 2, 4, and 8 respectively

F.2 Further CKA Analysis on WS-ConvMixer

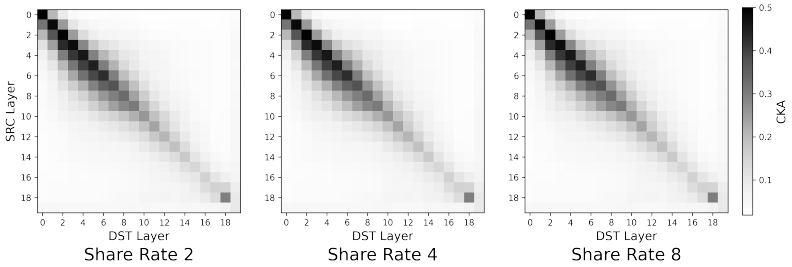
In this section, we provide more representational analysis using CKA on the Weight Sharing (WS) ConvMixer. In Figure 3, we show that for various share rates, the vanilla WS-ConvMixer does not have any clear pattern of layer-wise representational similarity with the original ConvMixer model (768/32/14/3 ar-



(a) WSConvMixer-512/16 with Weight Fusion. The original ConvMixer has 67.48% accuracy on ImageNet. For WS-ConvMixer with Weight Fusion, it has 65.04%, 59.34%, and 52.95% accuracy on ImageNet.



(b) WSConvMixer-768/32 with Weight Fusion. The original ConvMixer has 75.71% accuracy on ImageNet. For WS-ConvMixer with Weight Fusion, it has 75.14%, 67.15%, and 59.69% accuracy on ImageNet.



(c) WSConvMixer-1536/20 with Weight Fusion. The original ConvMixer has 78.03% accuracy on ImageNet. For WS-ConvMixer with Weight Fusion, it has 78.47%, 75.76%, and 71.4% accuracy on ImageNet.

Fig. 4: CKA analysis on WS-ConvMixer model with Weight Fusion. We compute pairwise analysis of WS-ConvMixer layer feature maps to the non-sharing ConvMxier model using CKA

chitecture setting). It's also worth noting the absolute value of similarity is quite low for these models. On the other hand, in Figure 4, we show that in the WS-ConvMixer models with weight fusion we see a clear relationship in the representations learned by the weight sharing model compared to the original, as well as significantly higher absolute similarity. This trend holds across multiple architecture settings (2/16/14/9, 768/32/14/3 and 1536/20/14/3) and share rates (2, 4, 8). This finding suggests that weight sharing models have the ability to generate similar representations to the original models, even with significantly less parameters, but need advanced training methods such as our weight fusion strategy to achieve this in practice.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015)
2. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv **abs/1704.04861** (2017)
3. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks (2018)
4. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. arXiv preprint arXiv:2201.03545 (2022)
5. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers distillation through attention (2021)
6. Trockman, A., Kolter, J.Z.: Patches are all you need? CoRR **abs/2201.09792** (2022), https://arxiv.org/abs/2201.09792
7. Wortsman, M., Horton, M., Guestrin, C., Farhadi, A., Rastegari, M.: Learning neural network subspaces. ArXiv **abs/2102.10472** (2021)