Rethinking the Role of Spatial Mixing

George Cazenavette[1], Joel Julin[2], and Simon Lucey[3]

[1] Carnegie Mellon University
[2] University of Pittsburgh
[3] University of Adelaide

Abstract. Until quite recently, the backbone of nearly every state-of-the-art computer vision model has been the 2D convolution. At its core, a 2D convolution simultaneously mixes information across both the spatial and channel dimensions of a representation. Many recent computer vision architectures consist of sequences of isotropic blocks that disentangle the spatial and channel-mixing components. This separation of the operations allows us to more closely juxtapose the effects of spatial and channel mixing in deep learning. In this paper, we take an initial step towards garnering a deeper understanding of the roles of these mixing operations. Through our experiments and analysis, we discover that on both classical (ResNet) and cutting-edge (ConvMixer) models, we can reach nearly the same level of classification performance by *only learning channel mixing* and leaving the spatial mixers at their random initializations. Furthermore, we show that models with random, fixed spatial mixing are naturally more robust to adversarial perturbations. Lastly, we show that this phenomenon extends past the classification regime, as such models can also decode pixel-shuffled images.

Keywords: Convolution, Efficiency, Robustness

1 Introduction

For the better part of the last two decades, cascades of learned convolutions have formed the backbone of nearly every innovation in the field of computer vision and pattern recognition. From distinguishing ten digits with LeNet [16] to one thousand classes with AlexNet [15], from going very deep with VGG [20] to even deeper with InceptionNet [21], the learned 2D convolution has served as the workhorse that ushered in the new era of visual learning. Convolutional neural networks (CNNs) learned to exploit the correlations across channels and between spatially close pixels to solve a plethora of tasks.

Recently, isotropic networks (those in which the size of the representation stays fixed throughout) such as Vision Transformer [8], Image GPT [4], MLP-Mixer [22], and ConvMixer [23] have been grabbing the field's attention. These isotropic models consist of repeated blocks wherein each block consists of a spatial mixing operation (self-attention [8,4], spatial MLP [22], or depthwise convolution [23]) followed by one or more *strictly* channel-mixing layers (1×1

or "pointwise" convolutions). A strictly spatial mixing operation is defined here as any operation that is applied independently across each channel. Similarly, a strictly channel-mixing layer is applied independently across spatial pixel coordinates. The fact that all these recent state-of-the-art isotropic architectures spend a significant portion of their computation budget solely on channel-mixing parameters hints toward the possibility that channel mixing may have significantly more relative importance than previously appreciated.

It has been understood for some time that impressive performance can be obtained from networks whose weights are not all completely learned [19]. Before the success of AlexNet [15] in training deep end-to-end networks, the use of random weights in early convolutional layers played an important role in training deeper CNNs. The approach was attractive as it promised faster training times, better generalisation, and the ability to learn deeper networks. Saxe et al. [19] even offered strong theoretical motivations for why random filter weights in CNNs would offer good performance in terms of: (i) frequency selection, and (ii) translation invariance. One of the most notable works with respect to using random weights within networks can be found in the Extreme Learning Machines (ELMs) of Huang et al [13]. In its simplest form an ELM takes a proposed network architecture and randomly initializes all hidden weights, leaving only the final layer to learn. ELMs are advantageous as they allow for extremely deep networks and rapid train times (as only the final layer needs to be learned). It is widely understood, however, that these training strategies have a significant performance gap in relation to their current end-to-end learned counterparts.

In this work, we leverage the separable convolution to compare the relative effects of spatial and channel mixing on the performance of deep neural networks. By leaving either the spatial-mixing or channel-mixing parameters frozen at their random initialization and only training the others, we can isolate the contributions of the both types of parameters. Doing so reveals that learning the *channel-mixing* parameters is *far* more critical to the effectiveness of a deep model, and such models that only learn channel-mixing parameters perform nearly just as well as their fully-learned counterparts. This revelation is doubly interesting since it is the learning of the *spatial-mixing* parameters that account for much of the training cost despite their relatively low importance.

We also show that models that only learn channel-mixing parameters are naturally more resistant to adversarial attacks than fully-learned models. This robustness can be further enhanced by then directly smoothing the random, frozen spatial-mixing filters.

Lastly, we show that this phenomenon extends beyond the regime of classification problems and find that models that only learn spatial mixing are even capable of learning to invert a permutation of pixels nearly just as well as fully-learned networks.

We hope that these gained insights as to the role and effectiveness of spatial and channel mixing in deep models will be of great value to the vision community and that they will shed light on which parts of deep models are most critical to learn while helping to inspire the intelligent design of future architectures.

2 Related Work

Isotropic Architectures First popularized by the transformer [24], isotropic architectures (those in which the size of the representation remains fixed throughout) have more recently made their way into the computer vision world. Image-GPT [4] (based on the GPT language model [4]) was able to model sequences of pixels and generate new images in raster fashion. Designed for discriminative tasks, the Vision Transformer [8] used transformer blocks to achieve state-of-the-art image classification results, but only after extensive pre-training. The MLP-Mixer [22] then built off Vision Transformer's success, but used a simple spatial-mixing MLP instead of the expensive self-attention module. While the MLP-Mixer still required extensive pre-training, the newest addition to the line of isotropic architectures, the ConvMixer [23], replaces the self-attention or spatial-mixing MLP with a simple depthwise filter bank and achieves comparable results *without* any pre-training. The main thing all these models have in common is their *isotropic* structure. Here, "isotropic" refers to repeated blocks of operations in which the latent representation does not change in shape. Furthermore, in all of these models, the second half of each isotropic block is made up of one or more *strictly-channel-mixing* layers.

Separable Convolutions Instead of mixing across channel and spatial dimensions simultaneously, the separable convolution factorizes the operation into a depthwise and pointwise convolution. The depthwise convolution applies a disjoint set of depth-1 filters to each channel of the input independently while the pointwise convolution is simply a 1×1 convolution (or linear projection) on the pixels of this intermediate representation. Separable convolutions are used extensively in many state-of-the-art architectures [12,26,5] and have built-in implementations in deep learning libraries [1].

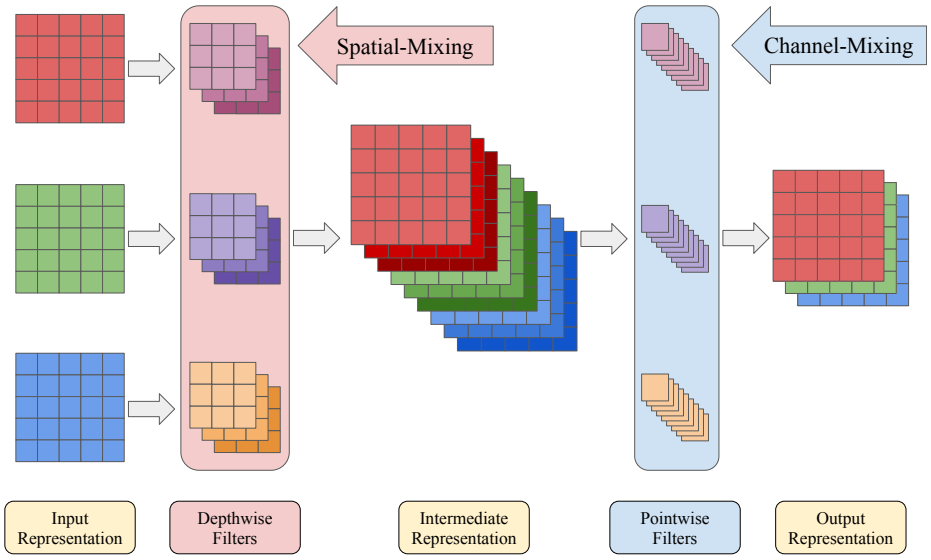
3 Background and Setup

Before our experiments, we provide some relevant exposition regarding the operations and architectures we use to illustrate the relative importance of spatial and channel mixing in deep neural networks.

3.1 The 2D Convolution

As we begin our investigation into the importance of channel mixing, we focus on the simple yet revolutionary ResNet [10] architecture. The original ResNet architecture is composed of blocks of 2D convolution operations. Given the number of in and out channels (c_{in}, c_{out}) and the size of the square kernel (k), we can represent the number of trainable parameters in the standard 2D convolution (p_{conv}) as

$$p_{conv} = c_{in} \cdot c_{out} \cdot k^2 \tag{1}$$



Illustrated Separable Convolution

Fig. 1: Separable convolutions allow us to disentangle the roles of spatial and channel mixing in deep networks by freezing either the depthwise or pointwise filters and learning the others.

With the vanilla Conv2D, it is hard to analyze the respective importance of spatial and channel mixing since the two are entangled in a single linear operation.

To disentangle the spatial and channel mixing [12,26,5], we separate the standard 2D convolution operation into a paired spatial-mixing (depthwise) and channel-mixing (1×1 or pointwise) convolution. In a separable convolution, each filter of the depthwise convolution operates on just a single channel of the input. The pointwise convolution is simply a 1×1 convolution, or a linear projection of each pixel.

After separating the standard 2D convolution into a depthwise and pointwise convolution, we can then introduce a *depth multiplier* to increase the number of filters per input channel. Naturally, this also increases the number of channels in the pointwise filters since the intermediate representation will now have more channels. Given the number of in and out channels (c_{in}, c_{out}), the size of the square depthwise kernel (k), and the depth multiplier d, we can represent the number of trainable parameters in the separable 2D convolution (p_{sep}) as

$$\begin{aligned} p_{sep} &= p_{depth} + p_{point} \\ &= c_{in} \cdot k^2 \cdot d + c_{in} \cdot d \cdot c_{out} \end{aligned} \tag{2}$$

By exploiting the inherent dependencies between nearby pixels, the standard 2D convolution—and its separable cousin—offer vision systems a cheap, yet effective way of extracting information from images.

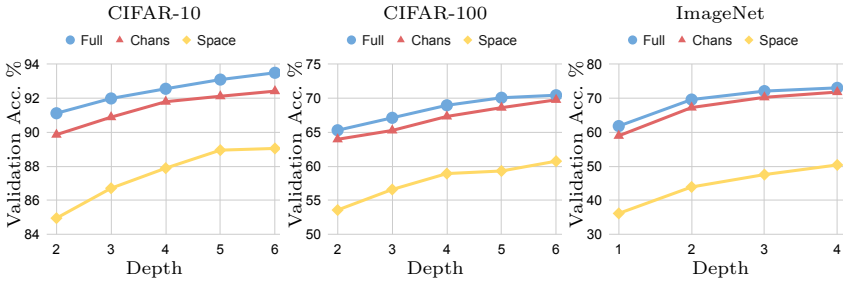


Fig. 2: While the fully learned ResNets (Full) outperform all others, we see that the models that only learn channel mixing (Chans) remain quite competitive, especially so on ImageNet (right). Conversely, the models that only learn spatial mixing (Space) lag very far behind the others.

3.2 The ConvMixer

As the latest (at the time of writing) in an ever-growing line of isotropic vision models [4,22,8], the ConvMixer [23] architecture adapts the (at this point) classical method of convolutions to the equi-sized representation (isotropic) paradigm of transformer models. Perhaps more importantly for our analysis, it serves as a state-of-the-art convolutional model in which *the convolutions are already separable*. In other words, the ConvMixer consists of strictly depthwise and pointwise convolutions. This allows us to examine the computational benefits of learning only channel mixing without the added overhead of converting the standard convolutions into separable ones.

The ConvMixer [23] architecture consists of depthwise convolutions (wrapped in a residual connection) and pointwise convolutions. Specifically, a depth-n ConvMixer consists of a one-layer patch-projection stem, n depthwise-pointwise blocks, a global average pool, and a linear classifier. All throughout, the representation maintains the size to which it is projected by the stem (hence the term *isotropic*).

3.3 Is Channel Mixing (Almost) All You Need?

By isolating the spatial and channel-mixing parts of the standard convolution into separable components, we can analyze the contribution of channel mixing alone to the success of convolutional neural networks. To do this, we now introduce a form of the separable convolution (Figure 1) wherein we leave the depthwise (spatial-mixing) filters frozen at their initialized values and only learn the pointwise (channel-mixing) filters during training. In our results, this is indicated by "Chans" whereas the fully-learned models are indicated by "Full."

Note that the number of trainable parameters does not depend on the size of the kernel (k) since only the pointwise parameters are learned. Furthermore, we can ensure the number of trainable parameters in the pointwise convolution

is equal to that of the corresponding standard Conv2D by setting $d = k^2$:

$$\begin{aligned} & d = k^2 \\ \implies & c_{in} \cdot c_{out} \cdot k^2 = c_{in} \cdot d \cdot c_{out} \\ \implies & p_{conv} = p_{point} \end{aligned} \tag{3}$$

For ResNet architectures, the standard convolution kernel is of shape 3×3, so we would set $d = 3^2 = 9$ to use the same number of parameters in our channel-learning-only convolution (and $d = 3$ to use one third of the parameters).

Here this is simply used as a heuristic to create models with similar parameter counts to the vanilla ResNet architecture. All of our separable ResNet models will thus have a depth-multiplier of 9, regardless of which parameters are being learned.

4 Experiments

After motivating with some relevant background, we now begin our study into the capabilities of networks that only learn channel mixing. Section 4.1 applies our hypothesis to classical ResNet [10] architectures. Then in Section 4.2, we move on to the naturally separable ConvMixer [23] architecture to illustrate further computational advantage. Lastly, in 4.3, we explore a practical application in the form of architectural adversarial robustness. Our code will be made public upon publication.

4.1 ResNet Experiments

For these experiments, we train all models under identical conditions: the same number of epochs, same batch size, same learning rate, same decay schedule, etc. Any and all differences (or similarities) in performance are due entirely to the intrinsic properties of the architectures themselves.

Model Architectures For our first set of experiments, we employ the ResNet [10] architecture with the identity mapping modifications introduced in [11]. A depth-n CIFAR [14] ResNet contains a 1-layer stem, $2n$ layers for each of the 3 blocks, and a linear classifier, for a total of $6n + 2$ layers. Similarly an ImageNet [6] ResNet contains a 1-layer stem, $2n$ layers for each of the 4 blocks, and a linear classifier, for a total of $8n + 2$ layers. The "depth" in our plots represents this n. All separable ResNets have a depth multiplier of 9.

Classification Gap For our ResNets that learn only channel mixing (Chans), any spatial mixing is done using depthwise filters fixed at their random initializations. Conversely, for those that learn only spatial mixing (Space), any channel mixing is done through the pointwise 1x1 convolutions frozen at their initial states.

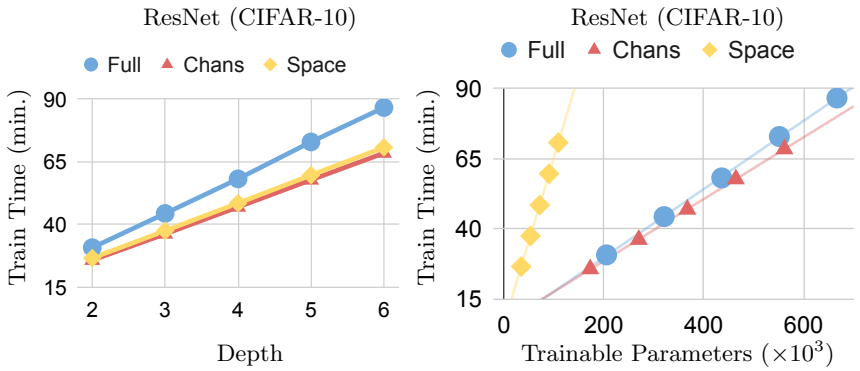


Fig. 3: **Left:** We see a large increase in speed when learning only channel-mixing parameters as opposed to both spatial and channel-mixing. **Right:** Spatial-mixing parameters seem to have a much higher computational cost than channel-mixing.

When we look at the performance of fully-learned ResNets versus those learning *only* channel mixing (Chans), we make a startling discovery: the gap in classification accuracy is barely there at all (Figure 2). One might question the generalization of this observation, noting that CIFAR [14] is a simple dataset, and the performance may just be saturated. Yet, the trend also holds for ImageNet [6], a *much* larger and more complex dataset. Furthermore, we also see that networks only learning spatial mixing (Space) consistently perform *significantly* worse than those learning only channel mixing.

One might note that the number of spatial mixing parameters in a separable convolution, learned or not, is inherently smaller than that of channel mixing parameters. This is true; however, the same can also be said of the standard convolution. In a typical network, the channel mixing dimensions that contribute to the weight size ($c_{in} \cdot c_{out}$) are typically much larger than the spatial mixing dimensions ($k \cdot k$).

Ultimately, the shown fact that networks learning only channel mixing achieve competitive results to those that are fully learned comes as quite the surprise and calls into question the significance of learning the spatial and channel mixing operations in general.

Computational Considerations A forward pass and back-propagation through an unlearned layer should be significantly faster than learnable one. We see this clearly in Figure 3: the networks with fully-learned separable convolutions (Full) take significantly longer to train than those with frozen spatial or channel-mixing layers (Space, Chans).

In Figure 3, we also see that the networks only learning spatial mixing (Space) have a much higher ratio of compute time per trainiable parameter than the fully learned separable networks (Chans). At first, this might just seem due to the spatial-only networks simply having fewer trainable parameters than the fully learned ones. However, we also see that the networks only learning channel mix-

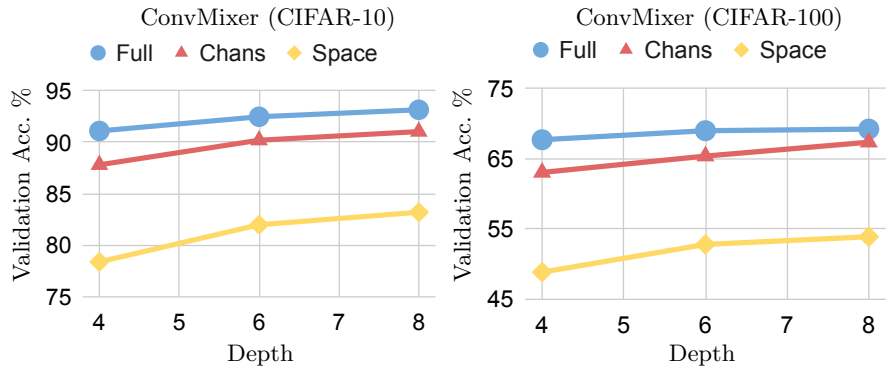


Fig. 4: With the ConvMixer architecture, we can better analyze the direct contributions of spatial and channel mixing without altering the original model. We again see networks that only learn channel mixing (Chans) remain competitive with their fully-learned counterparts (Full) while completely out-classing those that only learn spatial mixing (Space).

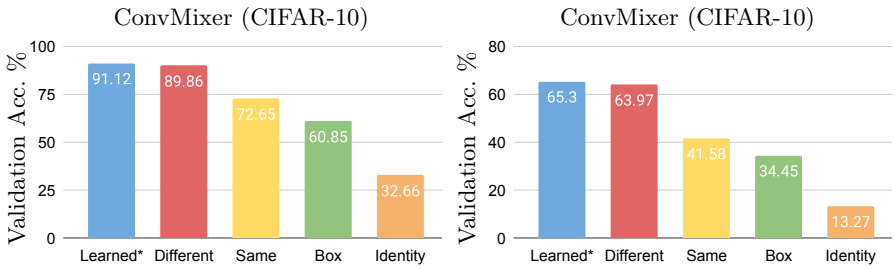


Fig. 5: While *randomly* initialized filters can provide competitive results, the same is not true for *any* arbitrary, fixed filter. Random filters work best when they are uncorrelated from eachother, allowing them to extract different information.

ing (Chans), which *also* have fewer trainable parameters than the fully trainable ones, have a *lower* train time to trainable parameter ratio than the fully learned separable convolutions (Full). With these two observations in mind, we can infer that learnable spatial-mixing parameters have a higher associated computational cost than learnable channel-mixing parameters.

Static Filter Structure While we have shown that learning spatial mixing is not necessary to achieve competitive performance without changes to the model's architecture, the static filters cannot be *completely* arbitrary; they must still hold some basic properties to adequately transform the input signal.

Looking at Figure 5, we see the performance of the fully-learned separable convolutions as the leftmost (blue) column. The next column (red) represents the separable convolution that only learns channel mixing. We again note that the randomly initialized spatial mixing performs only slightly worse than the fully-learned version. We label this column "Different" to contrast with the next (yellow) column, "Same." For this experiment, we apply the *same* set of 9 ran-

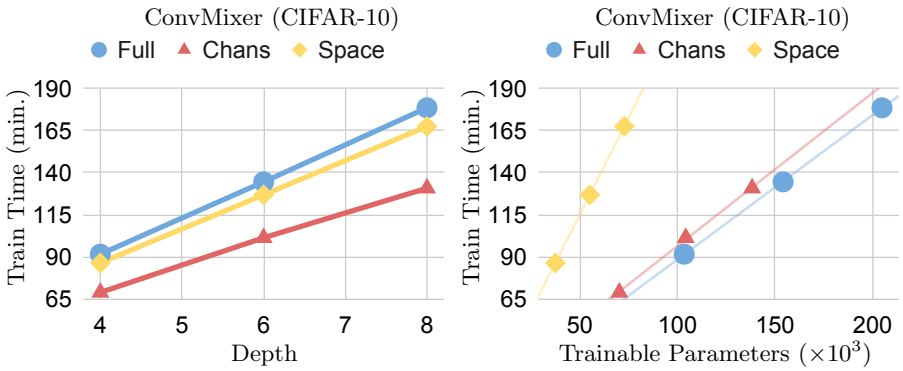


Fig. 6: We only see a minimal time save by not learning the channel mixing, but there is significant speedup when not learning the spatial mixing. A much larger portion of the total train time of the fully-learnable version seems to be taken up by the spatial mixing parameters than the channel mixing counterparts.

domly initialized kernels to *every input channel*. The large performance drop here shows us that highly-correlated filters do not make good feature extractors since they cover the same spectrum of signals. The next drop-off to the box filter (green) occurs because box filters are *perfectly* translationally correlated with each other. Lastly, the catastrophic failure of the identity filter (orange) tells us that although we might not need to *learn* it, spatial mixing is still an integral part of discriminative networks.

4.2 ConvMixer Experiments

We use the code provided along with the ConvMixer paper and a model with 128 channels, a kernel size of 8, and a patch-size of 1 across various depths.

Naturally Separable Performance Now that we are dealing with a naturally separable architecture, all comparisons can be made *directly* with the original model. In Figure 4, we again see that if we learn only channel mixing, we still achieve results competitive with the fully-learned architecture.

As we saw with ResNet (Figure 3), the spatial-learning ConvMixers have a much higher ratio of train time to trainable parameter count than the channel-learning models, as seen in Figure 6. This observation is again validated by the fact that the channel-learning ConvMixers have a significantly shorter train time than the spatial-learning ConvMixers despite having higher trainable parameter counts.

Filter Observations After years of networks using mostly 3×3 filters, the ConvMixer [23] takes a step back to larger filters that are more interpretable to the human eye. In Figure 7, we see the learned filters of a depth-4 ConvMixer.

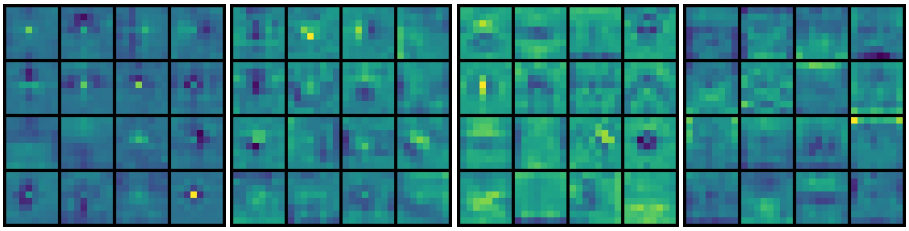


Fig. 7: Learned depthwise filters from a ConvMixer. Learned filters seem to follow different distributions depending on layer.

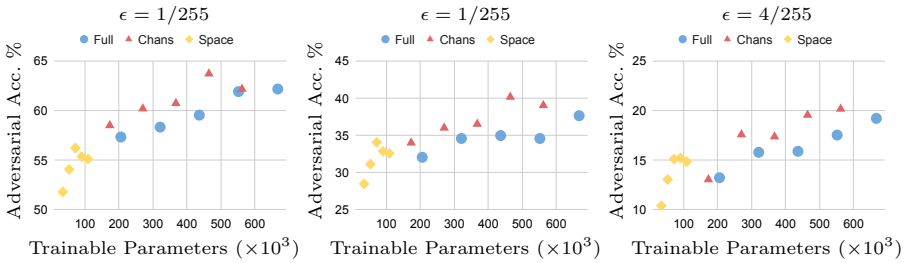


Fig. 8: Adversarial accuracy in separable ResNets (CIFAR-10). The networks with static spatial mixing (Chans) are clearly more robust to adversarial perturbations than their learned counterparts (Full) and those that learn only spatial mixing (Space). (ϵ is the magnitude of the adversarial perturbations.)

The learned filters are clearly more structured than their random counterparts, and the filters of each layer seem to follow a different distribution. Despite the structure clearly present in these learned filters, the random filters (as seen in Figure 9) still yield comparable performance.

4.3 Adversarial Robustness

Another practical application of our discovery lies in the adversarial setting. Adversarial examples are those specifically curated to fool a neural network into making a mistake at inference time by adding targeted noise to the sample [9]. Typically, the adversarial perturbations are small enough in magnitude that the semantic meaning of the example does not change and that a human observer cannot even notice them.

The Fast Gradient Sign Method First, we will quickly introduce one of the first and most simple adversarial attacks developed: the Fast Gradient Sign Method [9]. This method calculates the adversarial perturbations by taking the gradient of the network's loss with respect to the target image and mapping them to the ℓ_∞ ball.

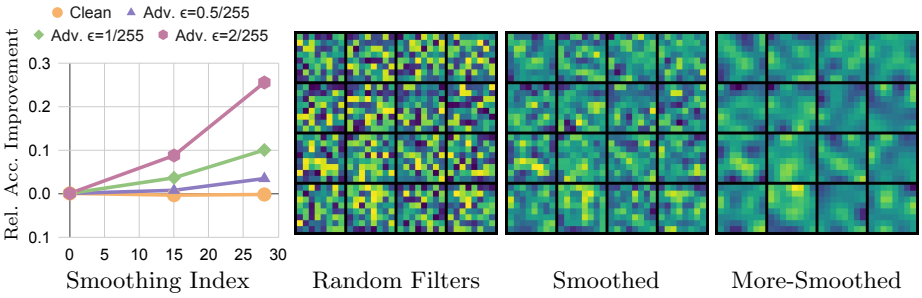


Fig. 9: Smoothing the random filters causes our trained models become significantly more robust to adversarial perturbations.

Given a neural network ($\mathcal{F}$), target sample ($\mathbf{x}$), and label (y), the adversarial example ($\tilde{\mathbf{x}}$) is calculated as

$$\tilde{\mathbf{x}} + \epsilon \cdot \text{sign}\left(\frac{\partial \mathcal{F}(\mathbf{x}, y)}{\partial \mathbf{x}}\right) \tag{4}$$

where ϵ is the radius of the ℓ_∞ ball.

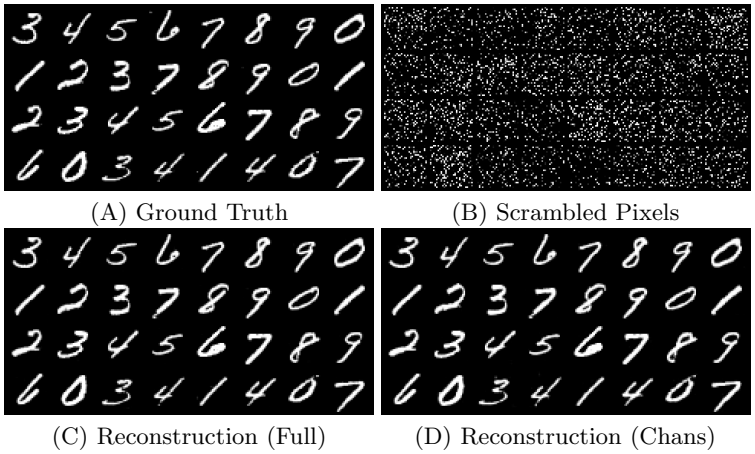
This is a simple attack that can be defended against by a variety of methods [2,9,18,17,3], but it is still a useful tool for analyzing the natural adversarial robustness of an architecture without taking any preventative measures.

Natural Robustness in Separable ResNets As we see in Figure 8, ResNets with separable convolutions that only learn channel mixing (Sep-Res) are significantly more robust to the adversarial perturbations than the fully-learned separable ResNets. Without any further modification, the models that only learn channel mixing seem to be less susceptible to targeted noise.

Induced Robustness in ConvMixers Looking at the learned filters (Figure 7) , one could mistakenly link the *smoothness* of learned filters to their poor adversarial robustness, relative to that of random filters (Figure 8). Our next experiment shows this to be false.

We can artificially increase the smoothness of our *random* filters by applying a low-pass smoothing operation directly to the filters themselves. After doing so, the high-frequency components of our random filters have been removed, and the filters now focus more on the mid to low frequencies, just as the learned filters do.

In Figure 9, we see that instead of hurting performance, smoothing the random filters actually significantly *increases* their robustness to adversarial attacks while not hurting performance on clean data (orange line) at all. In fact, we see that smoothing results in a relative performance increase of over 25% for the highest magnitude of attack. We then conclude that there must be a different reason (statistical shortcuts, etc.) for the learned filters' poor robustness.



(A) Ground Truth (B) Scrambled Pixels

(C) Reconstruction (Full) (D) Reconstruction (Chans)

Fig. 10: Models that only learn channel mixing are capable of learning near perfect reconstruction of MNIST digits from their shuffled pixels.

4.4 Pixel Un-Shuffling

Thus far, we have only shown the power of models that only learn channel mixing as it applies to classification tasks.

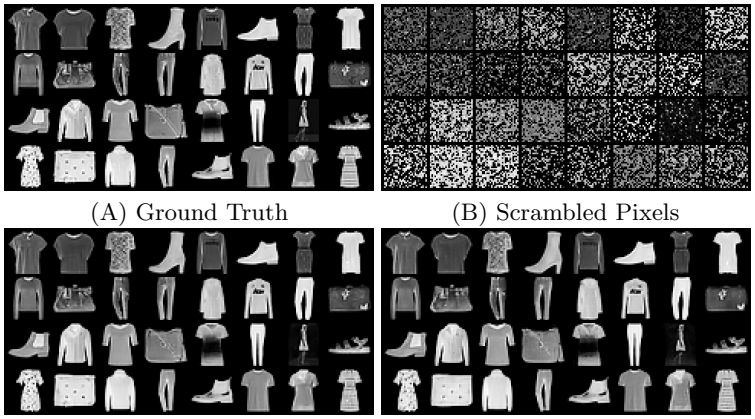
As evidence that such models can also prove effective for other tasks, we now apply them to the *pixel un-shuffling* problem. We define this problem as applying a deterministic random permutation of the spatial coordinates of all the pixels of the input image and tasking the model with reconstructing the un-corrupted signal. (To be clear, the same permutation pattern is applied to every input image.)

For images $\mathbf{x}_i$, pixel permutation σ, and network $\mathcal{F}$ with parameters θ, our formal training objective becomes

$$\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathcal{F}(\sigma(\mathbf{x}_i); \theta)\|^2$$

For this task, we use a patch-size 1 ConvMixer with a slight modification: instead of a global average pool and classification head, we simply apply a linear projection back up to the appropriate number of channels (1 or 3). We chose the ConvMixer because it's isotropic structure makes it ideal for image-to-image translation, and its natively separable design allows for easy analysis of the effects of spatial and channel mixing.

Intuitively, one would think the learning of spatial mixing is crucial for the task of "moving" pixels back to their correct location. However, our experiments show that, yet again, models that only learn channel mixing perform nearly just as well as their fully-learned counterparts. Randomly selected examples from the test sets can be seen in Figures 10, 11, and 13.



(A) Ground Truth (B) Scrambled Pixels

(C) Reconstruction (Full) (D) Reconstruction (Chans)

Fig. 11: Moving to Fashion-MNIST, a significantly more complex dataset than MNIST (Digits), the models that learn only channel mixing still achieve near perfect unshuffling with results almost indistinguishable from those of the fully-learned models.

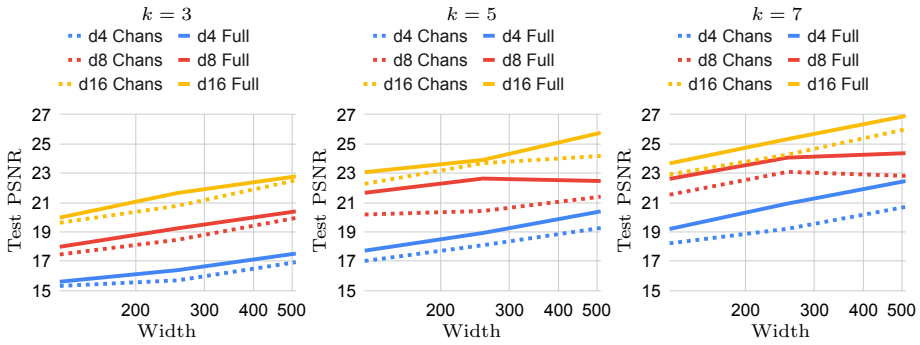


Fig. 12: **CIFAR-10**: PSNR for Pixel Un-Shuffling.

On all of MNIST [7], Fashion-MNIST [25] and CIFAR-10 [14], the test-set results for both model types are shockingly close. Unlike the classification problem, these results give us a clear, human-interpretable notion of the fact that networks can seemingly learn effective point-wise linear transforms to leverage the natural spatial mixing provided by the randomly-initialized depth-wise convolutions. In Figure 12, we include plots showing PNSR trends for various model architectures on CIFAR-10. We see that the performance increases with depth, width, and kernel size. We include PNSR plots for MNIST and Fashion-MNIST as well as results for models that only learn spatial mixing in the supplementary material.

(A) Ground Truth (B) Permuted Pixels

(C) Reconstruction (Full) (D) Reconstruction (Chans)

Fig. 13: **CIFAR-10**: Randomly selected test set samples and reconstructions from best performing model ($d = 16$, $w = 512$, $k = 7$)

5 Conclusion

While previous works have shown the merits of fully frozen networks as feature extractors [19,13], none (to the best of our knowledge) have explored the question of which parameters are the most important or beneficial to learn.

In this work, we have done just that; we have shown empirically that networks that only learn channel mixing can reach nearly the same performance as fully-learned networks without any further alterations. We also showed that networks with random spatial-mixing weights are naturally more robust to adversarial attacks, and we also offer a method for *further* increasing the adversarial robustness of such networks. Lastly, we showed that this phenomenon extends past the classification regime and that such restricted models are also capable of learning other tasks, such as pixel un-shuffling.

We hope our work will be of use to the vision community, spurring further questions as to the inner-workings of neural networks and leading to more efficient and robust models.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org 3
2. Cazenavette, G., Murdock, C., Lucey, S.: Architectural adversarial robustness: The case for deep pursuit. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7150–7158 (2021) 11
3. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: Adversarial attacks and defences: A survey (2018) 11
4. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International Conference on Machine Learning. pp. 1691–1703. PMLR (2020) 1, 3, 5
5. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017) 3, 4
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009) 6, 7
7. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012) 13
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020) 1, 3, 5
9. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014) 10, 11
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), http://arxiv.org/abs/1512.03385 3, 6
11. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European conference on computer vision. pp. 630–645. Springer (2016) 6
12. 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 (2017) 3, 4
13. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1-3), 489–501 (2006) 2, 14
14. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 6, 7, 13
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012) 1, 2
16. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998) 1
17. Li, Y., Wang, Y.: Defense against adversarial attacks in deep learning. Applied Sciences **9**(1) (2019). https://doi.org/10.3390/app9010076, https://www.mdpi.com/2076-3417/9/1/76 11

18. Romano, Y., Aberdam, A., Sulam, J., Elad, M.: Adversarial noise attacks of deep learning architectures: Stability analysis via sparse-modeled signals. Journal of Mathematical Imaging and Vision **62**(3), 313–327 (2020) 11
19. Saxe, A.M., Koh, P.W., Chen, Z., Bhand, M., Suresh, B., Ng, A.Y.: On random weights and unsupervised feature learning. In: Icml (2011) 2, 14
20. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 1
21. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015) 1
22. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., 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 1, 3, 5
23. Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022) 1, 3, 5, 6, 9
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017) 3
25. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017) 13
26. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018) 3, 4