Understanding MLP-Mixer as a Wide and Sparse MLP

Tomohiro Hayase [1] **Ryo Karakida** [2]

Abstract

Multi-layer perceptron (MLP) is a fundamental component of deep learning, and recent MLP-based architectures, especially the MLP-Mixer, have achieved significant empirical success. Nevertheless, our understanding of why and how the MLP-Mixer outperforms conventional MLPs remains largely unexplored. In this work, we reveal that sparseness is a key mechanism underlying the MLP-Mixers. First, the Mixers have an effective expression as a wider MLP with Kronecker-product weights, clarifying that the Mixers efficiently embody several sparseness properties explored in deep learning. In the case of linear layers, the effective expression elucidates an implicit sparse regularization caused by the model architecture and a hidden relation to Monarch matrices, which is also known as another form of sparse parameterization. Next, for general cases, we empirically demonstrate quantitative similarities between the Mixer and the unstructured sparse-weight MLPs. Following a guiding principle proposed by Golubeva, Neyshabur and Gur-Ari (2021), which fixes the number of connections and increases the width and sparsity, the Mixers can demonstrate improved performance.

1. Introduction

Multi-layer perceptron (MLP) and its variants are fundamental components of deep learning employed in various problems and for understanding the basic properties of neural networks. Despite their simplicity and long history (Rosenblatt, 1958; Schmidhuber, 2015), it has become apparent only recently that there is still significant room for improvement in the predictive performance of MLP-based architectures.

The sparseness is known as a key direction for enhancing the performance of dense MLP layers (Neyshabur et al., 2014; d'Ascoli et al., 2019; Neyshabur, 2020; Golubeva et al., 2021; Pellegrini & Biroli, 2022). For instance, Golubeva et al. (2021) reported that the prediction performance improves by increasing both the width and the sparsity of connectivity when the number of trainable parameters is fixed. The MLP-Mixer is another noteworthy direction of recent developments in MLP-based architectures (Tolstikhin et al., 2021; Touvron et al., 2022). It does not rely on convolutions or self-attention and is entirely composed of MLP layers; instead, it utilizes MLPs applied across spatial locations or feature channels. This can be regarded as an MLP that applies fully connected (FC) layers on both sides of the feature matrix. Despite its simplicity, the MLP-Mixer achieved a performance on image classification benchmarks comparable to that of more structured deep neural networks.

However, there are few studies on experimental and theoretical attempts to understand MLP-Mixer's internal mechanisms (Yu et al., 2022; Sahiner et al., 2022). This contrasts with the extensive elucidation of explicit or implicit biases in other modern architectural components (Neyshabur et al., 2014; Bjorck et al., 2018; Cordonnier et al., 2019). To further advance the MLP-based architecture, it will be crucial to unveil the underlying mechanism of the MLP-Mixer and to address questions such as: What different inductive biases does the MLP-Mixer have compared to a naive MLP? Which architectural factors significantly contribute to their superior performance?

In this study, we reveal that the sparseness, which is seemingly a distinct research concept, is the key mechanism underlying the MLP-Mixer. One can see that the MLP-Mixer is a wider MLP with sparsity inherently embedded as an inductive bias. This equivalence also provides a quantitative understanding that appropriate token and channel sizes maximizing the sparsity can improve the prediction performance. The detailed contributions are summarized as follows:

- We first identify *an effective expression of MLP-Mixer as an MLP* by vectorizing the mixing layers (in Section 3). It is composed of the permutation matrix and the Kronecker product and provides an interpretation of mixing layers as an extremely wide MLP with sparse (structured) weights.

[1]Metaverse Lab, Cluster Inc. [2]Artificial Intelligence Research Center, AIST. Correspondence to: Tomohiro Hayase <t.hayase@cluster.mu>, Ryo Karakida <karakida.ryo@aist.go.jp>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

- We consider a linear activation case of a simple Mixer and its effective expression. First, this reveals that the Kronecker-product weight possesses a bias towards an implicit L1 regularization (in Section 3.2). In deep learning, there are two aspects of sparsity: one is a large number of zero parameters (e.g., Neyshabur (2020)) and the other is the limited number of independent parameters (e.g., Dao et al. (2022)). Our evaluation of implicit regularization effectively links these two facets of sparsity. Second, regarding the limited number of independent parameters, the MLP-Mixer can be regarded as an approximation of an MLP with the Monarch matrix (in Section 3.3). Thus, the sparseness discussed in the different literature is inherently and originally incorporated into the Mixers.

- For realistic cases with non-linear activation functions, we quantitatively evaluate the high similarity between the traditional sparse-weight MLP (SW-MLP) and the MLP-Mixer. First, we confirm the similarity of hidden features by the centered kernel alignment. Second, we reveal similar performance trends when increasing sparseness (equivalent to widening) with a fixed number of connections. This means that empirical observations of Golubeva et al. (2021) on appropriate sizes of sparsity hold even in the MLP-Mixer in the sense of both improving prediction performance (in Section 4.2) and ensuring trainability (in Section 4.3). We also empirically verify that the Random-Permuted (RP) Mixer introduced in Section 5.2, which is a less structured variant of the normal Mixer, exhibits similar performance trends (Sections 5.3 & 5.4). This further solidifies our understanding that sparsity is a fundamental element underlying the Mixers.

2. Preliminaries

2.1. Related Work

MLP-based architectures. The salient property of an MLP-Mixer is that it is composed entirely of FC layers. This property is unique to the MLP-Mixer (and its concurrent work ResMLP (Touvron et al., 2022)) and different from attention-based architectures (Dosovitskiy et al., 2021). While some previous work focused on providing a relative evaluation of performance compared with the attention module (Yu et al., 2022; Sahiner et al., 2022), our purpose is to elucidate the hidden bias of MLP-Mixers as a wide and sparse MLP. Golubeva et al. (2021) investigated that the generalization performance can be improved by increasing the width in FC layers. Because they fixed the number of weights, an increase in the width caused a higher sparsity. They revealed that even for fixed sparse connectivity throughout training, a large width can improve the performance better than the dense layer.

Structured weight matrices: (i) Sparse matrix. Parameter sparsity is widely used to improve the performance and efficiency of neural networks. It is known that naive MLPs require even more data than structured deep models to improve performance due to their weak inductive bias (Bachmann et al., 2023). The sparseness is considered useful as a minimal necessary bias. One approach to make weights sparse is to determine nonzero weights dynamically, such as dense-to-sparse training (Neyshabur, 2020), pruning (Frankle & Carbin, 2019), and sparse-to-sparse training (Dettmers & Zettlemoyer, 2019; Evci et al., 2020). The other is to constrain the trainable weights from the beginning of training statically (Dao et al., 2022; Golubeva et al., 2021; Liu et al., 2022; Gadhikar et al., 2023). The current study follows the latter approach; specifically, we reveal that the mixing layers of the MLP-Mixer are implicitly related to such fixed sparse connectivity. **(ii) Kronecker product.** Constraining weight matrices to the Kronecker product and its summation has been investigated in the model-compression literature. Some works succeeded in reducing the number of trainable parameters without deteriorating the prediction performance (Zhou et al., 2015; Zhang et al., 2021) while others applied them for the compression of trained parameters (Hameed et al., 2022). In contrast, we find a Kronecker product expression hidden in the MLP-Mixer, which can be regarded as an approximation of the Monarch matrix proposed in Dao et al. (2022).

Notably, our study is completely different from merely applying sparse regularizers, Kronecker-product weights, or Monarch matrices (Dao et al., 2022) to the dense weight matrices in the mixing layers like Fu et al. (2023). Our finding is that MLP-Mixers and their generalization (i.e., the PK family) inherently possess these properties.

2.2. Notations

MLP-Mixer. An MLP-Mixer is defined as follows (Tolstikhin et al., 2021). Initially, it divides an input image into patches. Next, a per-patch FC layer is performed. After that, the blocks described as follows are repeatedly applied to them: for the feature matrix from the previous hidden layer $X \in \mathbb{R}^{S \times C}$,

$$\text{Token-MLP}(X) = W_2 \phi(W_1 X), \tag{1}$$

$$\text{Channel-MLP}(X) = \phi(X W_3) W_4, \tag{2}$$

where ϕ denotes the entry-wise activation function, $W_1 \in \mathbb{R}^{\gamma S \times S}$, $W_2 \in \mathbb{R}^{S \times \gamma S}$, $W_3 \in \mathbb{R}^{C \times \gamma C}$ and $W_4 \in \mathbb{R}^{\gamma C \times C}$. In this paper, we set the expansion factor of the hidden layers of token and channel-mixing MLPs to the same value γ for simplicity. The block of the MLP-Mixer is given by the map $X \mapsto Y$, where

$$U = X + \text{Token-MLP}(\text{LN}(X)), \tag{3}$$

$$Y = U + \text{Channel-MLP}(\text{LN}(U)). \tag{4}$$

In the end, the global average pooling and the linear classifier are applied to the last hidden layer.

S-Mixer. To facilitate theoretical insights, we introduce the S-Mixer, an idealized version of the MLP Mixer:

$$H = \phi(\phi(WX)V), \tag{5}$$

where V and W represent weight matrices. In this formulation, the shallow neural networks typically found within the mixing block of the MLP-Mixer are simplified to single layers. For the sake of simplicity in our theoretical analysis of the S-Mixer, we omit layer normalization and skip connections. It is important to note, however, that in our numerical experiments on training deep models in Section 5, these components are incorporated, demonstrating that their inclusion does not detract from the fundamental outcomes of the study. we summarized the detailed architectures used for experiments in Appendix A.

3. Properties as a sparse MLP

3.1. Vectorization

To address the similarity between MLP-Mixer and MLP, we consider vectorization of feature tensors and effective width. We represent the vectorization operation of the matrix $X \in \mathbb{R}^{S\times C}$ by $\mathrm{vec}(X)$; more precisely, $(\mathrm{vec}(X))_{(j-1)d+i} = X_{ij}, (i = 1, \dots, S, j = 1, \dots, C)$. We also define an inverse operation $\mathrm{mat}(\cdot)$ to recover the matrix representation by $\mathrm{mat}(\mathrm{vec}(X)) = X$. There exists a well-known equation for the vectorization operation and the Kronecker product denoted by $\otimes$;

$$\mathrm{vec}(WXV) = (V^\top \otimes W)\mathrm{vec}(X), \tag{6}$$

for $W \in \mathbb{R}^{S\times S}$ and $V \in \mathbb{R}^{C\times C}$. The vectorization of the feature matrix WXV is equivalent to a fully connected layer of width $m := SC$ with a weight matrix $V^\top \otimes W$. We refer to this m as the *effective width* of mixing layers.

In MLP-Mixer, when we treat each $S \times C$ feature matrix X as an SC-dimensional vector $\mathrm{vec}(X)$, the right multiplication by an $C\times C$ weight V and the left weight multiplication by a $S \times S$ weight W are represented as follows:

$$\mathrm{vec}(XV) = (V^\top \otimes I_S)\mathrm{vec}(X), \tag{7}$$

$$\mathrm{vec}(WX) = (I_C \otimes W)\mathrm{vec}(X), \tag{8}$$

where I_n denotes an $n \times n$ identity matrix. This expression clarifies that the mixing layers work as an MLP with special weight matrices with the Kronecker product. As usual, the size of S and C is approximately $10^2 \sim 10^3$, and this implies that the Mixer is equivalent to an extremely wide MLP with $m = 10^4 \sim 10^6$. Moreover, the ratio of non-zero entries in the weight matrix $I_C \otimes W$ is $1/C$ and that of $V^\top \otimes I_S$ is $1/S$. Therefore, the weight of the effective MLP is highly sparse in the sense of non-zero entries.

Here, to consider only the left-multiplication of weights, we introduce commutation matrices.

Definition 3.1. The commutation matrix J_c is an $m \times m$ matrix defined as

$$J_c\mathrm{vec}(X) = \mathrm{vec}(X^\top), \tag{9}$$

where X is an $S \times C$ matrix.

Note that for any $x \in \mathbb{R}^m$, $J_c\phi(x) = \phi(J_c x)$. In addition, we have $J_c^\top(I_S \otimes V^\top)J_c = V^\top \otimes I_S$ for any $C \times C$ matrix V. Using the commutation matrix, we find the following:

Proposition 3.2 (Effective expression of MLP-Mixer as MLP)**.** *The feature matrix of the S-Mixer (5) is a shallow MLP with width $m = SC$ as follows:*

$$vec(H) = \phi\left(J_c^\top\left(I_S \otimes V^\top\right)\phi\left(J_c\left(I_C \otimes W\right)vec\left(X\right)\right)\right). \tag{10}$$

The derivation is straightforward as described in Appendix B.1. This expression clarifies that the mixing layers work as an MLP with special weight matrices with the commutation matrix and Kronecker product.

It is easy to generalize the above expression for the S-Mixer to the MLP-Mixer, where each mixing operation is composed of shallow neural networks (2) (see Appendix B.1). This equivalence with a wide MLP with sparse weights is simple and easy to follow but has been missing in the literature.

3.2. Implicit regularization of linear mixing layers

Considering the linear activation case to gain theoretical insight is a common approach used in deep learning (Saxe et al., 2014; Arora et al., 2019). yet there has been no work on the Mixers. We find that the theoretical analysis is challenging even in the simplest case of the linear S-Mixer (6). This model is equivalent to a so-called bi-linear regression model, for which an analytical solution that minimizes the MSE loss is unknown (Hoff, 2015). This makes analyzing the linear S-Mixer more difficult compared to conventional linear networks. Despite this difficulty, we discover the following inequality that characterizes the implicit regularization of the model:

Proposition 3.3.

$$\min_{V,W} \mathcal{L}(V \otimes W) + \frac{\lambda}{2}(\|V\|_F^2 + \|W\|_F^2) \geq \min_{B\in\mathbb{R}^{SC\times SC}} \mathcal{L}(B) + \tilde{\lambda}\|B\|_1, \tag{11}$$

where $\tilde{\lambda} = \lambda/CS$, $\|\cdot\|_F$ is the Frobenius norm, and $\|\cdot\|_1$ is the L^1 norm.

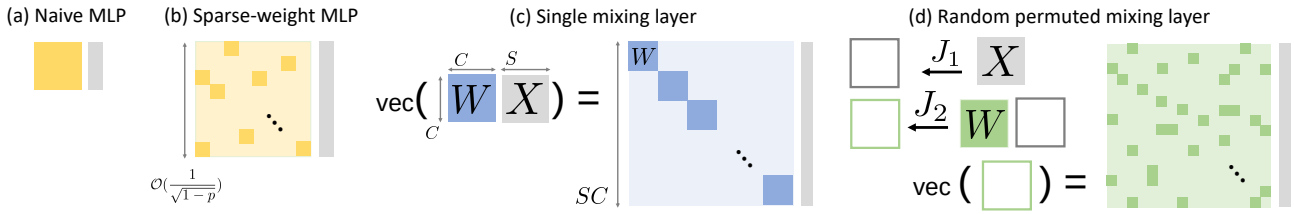

Figure 1. Schematic diagram of sparsity treated in this work. (a) A masked weight matrix $M \odot A$ in a sparse-weight MLP (SW-MLP). Its width is $O(1/\sqrt{p})$, where p is the ratio of non-zero entries in the mask M. (b) A mixing layer in an MLP-Mixer with the vectorization. The weight behaves as a block diagonal matrix. (c) A weight of a random permuted mixer (RP-Mixer), which is introduced in Section 5. The block diagonal structure is destroyed by random permutation matrices J_1, J_2 to achieve similarity to the SW-MLP.

The derivation is shown in Appendix B.4. It extends the fact that the Hadamard product parameterization has an implicit bias towards L1 regularization to the case of the Kronecker product parameterization (Hoff, 2017; Yasuda et al., 2023). Note that the number of independent parameters differs between the left and right sides of the inequality. Therefore, $\tilde{\lambda}$ might appear small, but it merely normalizes the change in parameter size (see Appendix B.4 for more details). The implicit bias towards sparsity is considered a desirable property in modern neural network architectures (Neyshabur et al., 2014; Woodworth et al., 2020; Karakida et al., 2023).

3.3. Monarch matrix hidden behind Mixers

Dao et al. (2022) proposed a *Monarch matrix* $M \in \mathbb{R}^{n \times n}$ defined by

$$M = J_c^\top L J_c R, \tag{12}$$

where L and R are the trainable block diagonal matrices, each with $\sqrt{n}$ blocks of size $\sqrt{n} \times \sqrt{n}$. The previous work claimed that the Monarch matrix is sparse in that the number of trainable parameters is much smaller than in a dense $n \times n$ matrix. Despite this sparsity, by replacing the dense matrix with a Monarch matrix, it was found that various architectures can achieve almost comparable performance while succeeding in shortening the training time. Furthermore, the product of a few Monarch matrices can represent many commonly used structured matrices such as convolutions and Fourier transformations.

Surprisingly, the MLP Mixer and the Monarch matrix, two completely different concepts, have hidden connections. By comparing (10) and (12), we find that

Corollary 3.4. *Consider a S-Mixer without an intermediate activation function, that is, its feature matrix is given by $H = \phi(WXV)$. Then, $\mathrm{vec}(H)$ is equivalent to an MLP whose weight matrix is given by a Monarch matrix with weight-sharing diagonal matrices, that is, $\phi(Mx)$ with $n = SC$, $L = I_S \otimes V^\top$ and $R = I_C \otimes W$.*

We validate the similarities between the Monarch matrix and the Mixer through experiments. Here, we consider a shallow MLP ($x \mapsto B_c B_2 \mathrm{ReLU}(B_1 x)$) where B_c is the classification layer. We compare the performance of models where weights B_i ($i = 1, 2$) are replaced with Monarch matrices M_i, and models where they are replaced with Kronecker products $V_i \otimes W_i$. In Figure 2 (left), the two models showed comparable performance. This suggests that although the Mixer incorporates a weight-sharing structure compared to Monarch, its performance is not compromised.

3.4. Comparing hidden features

Let us back to the situation in practice where the model has intermediate non-linear activation. To investigate the similarity of non-linear networks, here we consider an *unstructured sparse-weight MLP* (SW-MLP in short), which is a basic implementation in the literature of naive MLPs with sparse weights (Golubeva et al., 2021). Its weights are given by a random sparse mask, that is, $M \odot A$ where A is the original weight matrix and M is a static mask matrix whose entries are drawn from the Bernoulli distribution with a probability $p > 0$ of being one at the initialization phase. In this section, to compare the MLP-Mixer with SW-MLP sharing conditions, we consider the average $p = (S^{-1} + C^{-1})/2$ of the sparsity of MLP-Mixer for the setting of SW-MLP. Figure 1 overviews the models that we compare in the below. The random permuted one is an alternative to SW-MLP and shows the result in Section 5.

As a similarity measure, we use the centered kernel alignment (CKA) (Nguyen et al., 2021) between hidden features of MLPs with sparse weights and those of MLP-Mixers. In practice, we computed the mini-batch CKA (Nguyen et al., 2021, Section 3.1(2)) among features of trained networks. In Figure 2(a), we observed the averaged CKA achieved the maximum as an appropriate sparsity of the MLPs. By comparing Figure 2(b) and (c), we found that CKA matrix with sparse MLP was clearly higher than dense MLP. In particular, the sparse Mixer was similar to sparser MLP in hidden features. Detailed settings of all experiments are summarized in Appendix C.

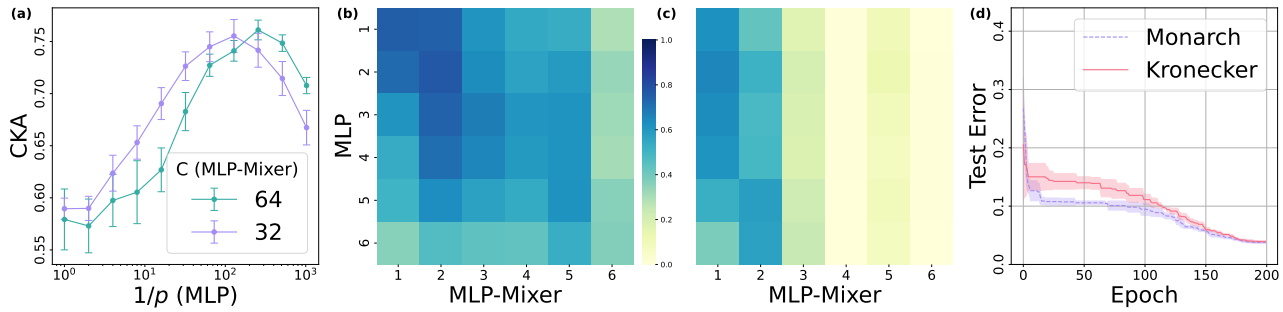

Figure 2. (a) Average of diagonal entries of CKA between trained MLP-Mixer ($S = C = 64, 32$) and MLP with different sparsity, where p is the ratio of non-zero entries in M. (b) CKA between MLP-Mixer ($S = C = 64$) and MLP with the corresponding $p = 1/64$, and (c) CKA between the Mixer and a dense MLP. (d) Test error on MNIST of shallow MLPs with Monarch matrix weights and Kronecker weights. The result is the average of five trials with different random seeds.

4. Properties as a wide MLP

In this section, we continue the discussion on whether the MLP-Mixer (or S-Mixer) exhibits tendencies similar to those of sparse-weight MLPs. In particular, we search for desirable sparsity and discuss whether there is a desirable setting of hyper-parameters on the Mixers.

4.1. Maximizing sparseness

The following hypothesis has a fundamental role:

Hypothesis 4.1 (Golubeva et al. (2021))**.** *Increasing the width up to a certain point, while keeping the number of weight parameters fixed, results in improved test accuracy.*

Intuitively, Golubeva et al. (2021) challenged the question of whether the performance improvement of large-scale deep neural networks was due to an increase in the number of parameters or an increase in width. They empirically succeeded in verifying Hypothesis 4.1; that is, the improvement was due to the increase in width in normal MLPs and ResNets (note that the width of ResNet indicates the channel size).

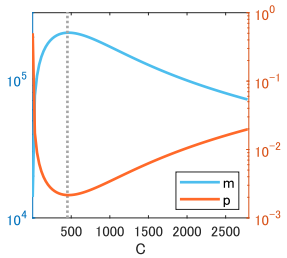

Figure 3. Theoretical line of m and p ($\Omega = 10^8, \gamma = 1$).

Let us denote by Ω the average number of connections per layer. We have

$$\Omega = p\gamma m^2, \tag{13}$$

where p is the ratio of non-zero entries in the static mask. Here, for the MLP-Mixer,

$$\Omega = \gamma(CS^2 + C^2S)/2. \tag{14}$$

The average number Ω of S-Mixer is reduced to $\gamma = 1$ in (14), which maintains the readability of the equations.

By (14), we have $S = (\sqrt{C^2 + 8\Omega/(\gamma C)} - C)/2$. For a fixed Ω and γ, the effective width is controlled by $m = SC$. Figure 3 shows m as a function of C. The width m has a single-peak form and is maximized at (C^*, S^*) as follows:

$$C^* = S^* = (\Omega/\gamma)^{1/3}, \quad \max_{S,C} m = (\Omega/\gamma)^{2/3}. \tag{15}$$

The ratio of non-zero entries $p = \Omega/\gamma m^2$ is minimized at this point, that is, the sparsity is maximized.

4.2. Comparing accuracy with effective width

To validate the similarity, we compare the test error of both networks with different sparsity. Under the fixed number Ω of connectivity per layer, the sparsity is equivalent to the wideness. Figure 4 (left) shows the test errors of MLP-Mixers and corresponding sparse weight MLPs under fixed $\Omega = 2^{19}$ and $\gamma = 2$, for several widths γm. We observed both networks' test error improved as the width increased. In this sense, MLP and MLP-Mixer have a similar tendency for performance with increasing width. However, we observed for too-wide cases around $\gamma m = 8000$ in Figure 4 (left), the test error of SW-MLP is higher than MLP-Mixer and there is little change in response to increasing width. We discuss this tendency in the next section.

4.3. Spectral analysis with increasing width

Golubeva et al. (2021) reported that if the sparsity became too high, the generalization performance of SW-MLP slightly decreased. They discussed that this decrease was caused by the deterioration of trainability, that is, it became difficult for the gradient descent to decrease the loss function. Some previous work reported that the large singular values of weight matrices at random initialization cause the deterioration of trainability in deep networks (Bjorck et al.,

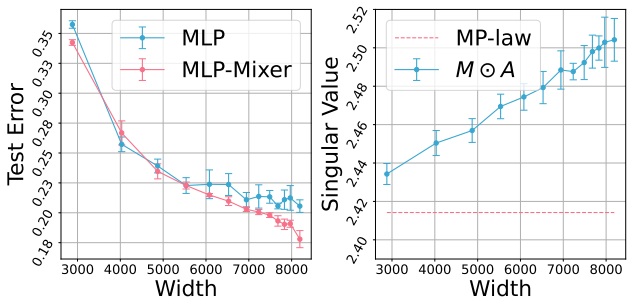

Figure 4. (left) Test error of MLPs with sparse weights and MLP-Mixers with different widths γm under the fixed Ω. We set $\Omega = 2^{19}$, $S = C = (\Omega/\gamma)^{1/3}$, and $\gamma = 2$. The x-axis represents the effective width γm. (right) The blue line indicates the averaged singular values of the weight $M \odot A$ of SW-MLP over five trials with different random seeds. The red line indicates c_γ, which is the square root of the right edge of the MP-Law.

2018). Therefore, we discuss the difference of singular values of weights between MLP-Mixer and SW-MLP.

In the case of Mixers, each $\gamma C \times C$ weight V (resp. $\gamma S \times S$ weight W) is initialized by i.i.d. random variables distributed with $\mathcal{N}(0, 1/C)$ (resp. $\mathcal{N}(0, 1/S)$). Then, by the theory of Marchenko-Pastur law (Bai & Silverstein, 2010), the weight's maximal singular value is approximated by $c_\gamma := 1 + \sqrt{\gamma}$. Since the Kronecker product with identity matrix does not change the maximal singular value, each maximal singular value of $V^\top \otimes I_S$ or $I_C \otimes W$ is approximated by c_γ.

Consider the case of SW-MLP. We initialize the entries of each mask matrix M by i.i.d. Bernoulli random variables with the probability of being $1/\sqrt{p}$ is p and being 0 is $1-p$. We initialize each weight A by i.i.d. $\mathcal{N}(0, 1/m)$. Consider the maximal singular value $\lambda_{\max}$ of $M \odot A$. Set $q = \sqrt{pm}$. Then by (Hwang et al., 2019, Theorem 2.9), $\lambda_{\max}$ is approximated by $\sqrt{L_+}$ in the following sense:

$$|\lambda_{\max}^2 - L_+| \prec 1/q^4 + 1/m^{2/3}, \tag{16}$$

where $\prec$ represents stochastic domination (Hwang et al., 2019, Definition 2.3), and

$$L_+ = c_\gamma^2 + 3c_\gamma^2\sqrt{\gamma}(1-p)/q^2 + O(1/q^4). \tag{17}$$

Under the fixed Ω, by (13), the dominant term of L_+ in (17) is linear in m as follows:

$$(1-p)/q^2 = \gamma m/\Omega - 1/m = O(m) \text{ as } m \to \infty. \tag{18}$$

Therefore, the maximal singular value $\lambda_{\max}$ of $M \odot A$ increases as the width increases. In Appendix D.1, we discuss the spectrum in large Ω limit and show the same tendency on the $\lambda_{\max}$ as large m limit.

In Figure 4 (right), we observed the maximal singular value of $M \odot A$ increased by widening m with fixed Ω and γ. In particular, the value was always higher than the theoretical value of Mixer's singular values. Such large singular values deteriorate the performance of SW-MLP. Conversely, we can enlarge the width and the sparsity of the MLP-Mixer without an undesirable increase in the maximal singular value.

5. Beyond the naive MLP on the width

As seen in Section 4.2, MLP-Mixers have similar tendencies to SW-MLPs. In much wider models, to continue comparing MLP-Mixer and unstructured sparse-weight MLP, we need an alternative to static-masked MLP because of its huge computational costs, memory requirements (Appendix D.9), and ill behavior on the spectrum (Section 4.3). Thus we further discuss partially destroying MLP-Mixer's structure and propose an alternative model of SW-MLP, which is called random permuted mixer (RP-Mixer).

5.1. PK family

To introduce an alternative to sparse-weight MLPs, we propose a permuted Kronecker (PK) family as a generalization of the MLP-Mixer.

Permutation matrix: An $m \times m$ permutation matrix J is a matrix given by $(Jx)_i = x_{\sigma(i)} (i = 1, 2, \ldots, m)$ for an index permutation σ. In particular, the commutation matrix J_c is a permutation matrix (Magnus & Neudecker, 2019). For any permutation matrix J, $x \in \mathbb{R}^m$, $J\phi(x) = \phi(Jx)$.

Definition 5.1 (PK layer and PK family)**.** Let J_1, J_2 be $m \times m$ permutation matrices. For $X \in \mathbb{R}^{n_1 \times n_2}$, we define the PK layer as follows:

$$\text{PK-Layer}_W(X; J_1, J_2) := \phi[J_2(I_{n_1} \otimes W)J_1\text{vec}(X)],$$

where we set $m = n_1 n_2$, $W \in \mathbb{R}^{n_2 \times n_2}$. We refer to the set of architectures whose hidden layers are composed of PK layers as the PK family.

Since J_c is a permutation matrix, the normal S-Mixer and MLP-Mixer belong to the PK family (See Section B.3 for the details). The important point of the PK-Layer is that its width m is possibly large, but there is *no need to explicitly preserve the $m \times m$ weight matrix* in memory. We can compute the forward signal propagation by a relatively small matrix multiplication in the same manner as the MLP-Mixer: First, $J_1\text{vec}(X) =: y$ is a rearrangement of X entries. Next, we compute pre-activation by using the matrix product $(I_{n_1} \otimes W)y = W\text{mat}(y)$. Finally, we apply entry-wise activation and rearrangement by J_2. Thus, the PK layer is memory-friendly, whereas the naive dense MLP requires preserving an $m \times m$ weight matrix and is computationally demanding.

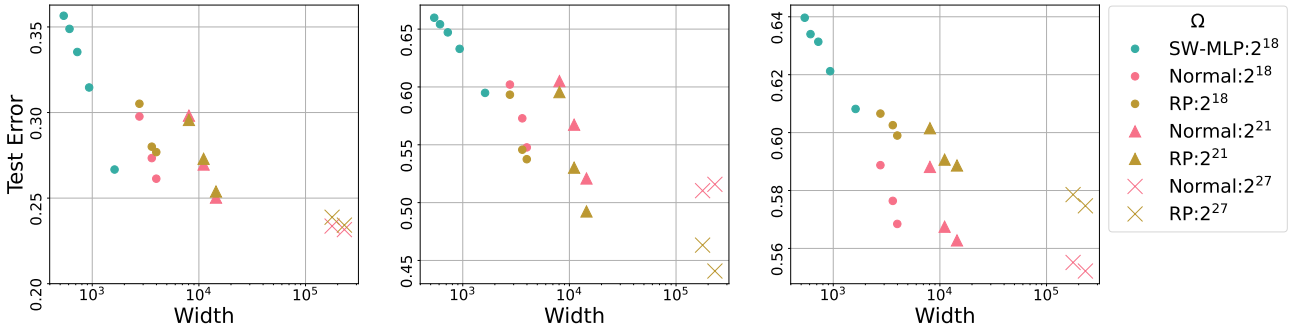

Figure 5. Test error improved as the effective width increased. This figure presents S-Mixer, RP S-Mixer, and SW-MLP models on CIFAR-10 (left), CIFAR-100 (center), and STL-10 (right). Experiments were conducted using three different random seeds, and the mean test error is depicted. The observed standard deviations were less than 0.026 for CIFAR-10, 0.056 for CIFAR-100, and 0.008 for STL-10.

Table 1. Test error on CIFAR-10/CIFAR-100/ImageNet-1k from scratch. (Upper Table) By setting S and C closer **under the same number of total connections** throughout layers, the maximal width of layers became larger in ours (Mixer-SS-W). Its test error eventually improved more than β-LASSO. (Lower Table) By setting S and C closer **under the same** Ω, the test error in ours (Mixer-B-W) improved than the original MLP-Mixer (Mixer-B/16). Each experiment is done with three random seeds.

MLP	CIFAR-10	CIFAR-100	max. width	#connections
β-LASSO (Neyshabur, 2020)	14.81	40.44	-	256M
Mixer-SS/8	15.91 ($\pm$1.55)	44.24($\pm$1.83)	6.3×10^4	256M
Mixer-SS-W	**12.07** ($\pm$0.47)	**38.13**($\pm$1.36)	$\mathbf{1.2 \times 10^5}$	255M

MLP	ImageNet-1k	max. width	Ω	S	C
Mixer-B/16 (Tolstikhin et al., 2021)	23.56	6.0×10^5	2.6×10^8	196	786
Mixer-B-W	**23.26** ($\pm$ 0.19)	$\mathbf{6.2 \times 10^5}$	2.6×10^8	256	588

5.2. Random Permuted Mixers

In normal Mixers, J_1 and J_2 are restricted to the identity or commutation. This means that the sparse weight matrices of the effective MLP are highly structured because their block matrices are diagonal. To destroy the structure, we introduce RP-Mixers. A *RP S-Mixer* has (J_1,J_2) in each PK layer, which is given by random permutation matrices as $U = \text{PK-Layer}_W(X; J_1, J_2)$ and $\text{PK-Layer}_{V^\top}(U; J_1', J_2')$. Similarly, for a *RP MLP-Mixer*, we set the PK layer corresponding to token mixing and channel mixing to the random permutation matrices. From the definition of the PK layer (5.1), this is equivalent to the effective MLP with width SC (and γSC for the MLP-Mixer) and sparse weight

$$W_{eff} = J_2(I_{n_1} \otimes W)J_1. \tag{19}$$

Because (J_1, J_2) are random permutations, the non-zero entries of W_{eff} are scattered throughout the matrix. In this sense, RP Mixers seemingly become much closer to random sparse weights than the normal Mixers. Figure 1(d) illustrates this scattered random weight configuration, while Figure S.9 presents an actual numerical example. Since the permutation matrices are orthogonal matrices, W_{eff}'s singular values remain identical to those of the normal Mixer, thereby preserving the spectrum discussed in Section 4.3.

5.3. Increasing width

Figure 5 shows that the test error improves as the effective width of the Mixers increases. We trained the normal and RP S-Mixers for various values of S and C with fixed Ω. The normal and RP S-Mixers show similar tendencies of increasing test error with respect to the effective width m. The normal and RP MLP-Mixers also show similar tendencies as is shown in Appendix D.2. Because the static-mask SW-MLP requires an $m \times m$ weight matrix, the figure cannot show an SW-MLP with a large Ω. In contrast, we can use a large effective width for the Mixers, and the error continues to increase as the width increases. This can be interpreted as the Mixers realizing a width setting where naive MLP cannot reach sufficiently. Eventually, the figure suggests that such an extremely large width is one factor contributing to the Mixer's success.

Table 1 shows a comparison of MLP-Mixer with a dynamic sparsity β-LASSO (Neyshabur, 2020). We found a wider Mixer (Mixer-SS-W) has better performance than β-LASSO. Table 1 shows a comparison of Mixer-B/16 (Tolstikhin et al., 2021) and a wider Mixer (Mixer-B-W). By setting S and C closer under fixed Ω, the maximal width of layers became larger in ours (Mixer-B-W). Its test er-

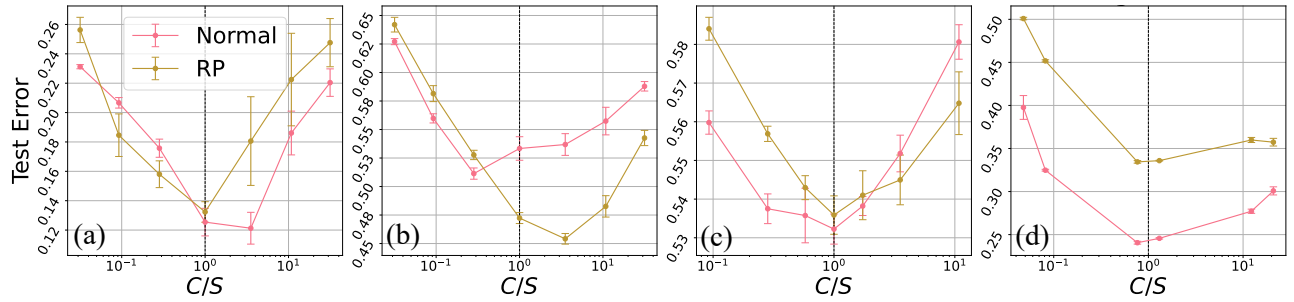

Figure 6. PK family achieves the lowest test error around $C = S$. Results of S-Mixers on (a) CIFAR-10, (b) CIFAR-100, (c) STL-10. Results of MLP-Mixers on (d) ImageNet-1k. The red lines are Normal Mixers, and the yellow ones are RP Mixers. The dashed lines are $C = S$.

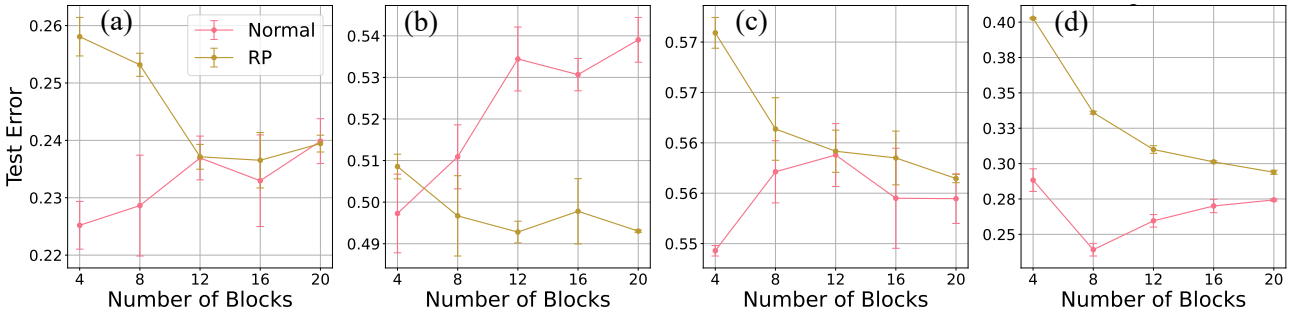

Figure 7. RP Mixers (red lines) can become comparable to or even beat normal ones (yellow lines) if the depth increases. We set $C = S = 128$. Results of S-Mixer on (a) CIFAR-10, (b) CIFAR-100, and (c) STL-10. Results of MLP-Mixer on (d) ImageNet-1k.

ror eventually improved more than the original MLP-Mixer (Mixer-B/16). In both results, the wideness improved the performance even if Ω is fixed.

5.4. Performance at the optimal width and sparsity

Figure 6 confirms that the maximum width (15) derived from Hypothesis 4.1 adequately explains the empirical performance of the Mixers. Models were trained using supervised classifications for each dataset. For CIFAR-10, CIFAR-100 and STL-10, we trained normal and RP S-Mixers. We fixed the dimension of the per-patch FC layer and changed S and C while maintaining a fixed number of Ω. It can be observed that the test error was minimized around $C = S$, as expected from Hypothesis 4.1 and (15). This tendency was common to the normal and RP Mixers. For ImageNet-1k, we trained normal and RP MLP-Mixers. Similarly, its performance is maximized around $C = S$. We observed a similar tendency also in different settings of hyperparameters in Appendix D.3 and Appendix D.5.

Remark on depth: As is shown in Figure 7, we observed that RP Mixers tended to underperform at limited depths but could achieve comparable or better results than normal Mixers with increased depths. This seems rational due to deep RP-Mixer's resistance to overfitting or shallow RP-Mixer's small receptive fields (see Appendix D.4 for the details). We also confirmed that this dependence on depth did not change the fundamental similarity regarding the width and sparsity.

6. Conclusion and future directions

This work provides novel insight into how the MLP-Mixer effectively behaves as a wide MLP with sparse weights. The analysis in the linear activation case elucidates the implicit sparse regularization through the Kronecker-product expression and reveals a connection to Monarch matrices. The SW-MLP, normal and RP Mixers exhibit a quantitative similarity in performance trends, verifying that the sparsity is the key mechanics underlying the MLP-Mixer. Maximizing the effective width and sparsity leads to improved performance. We expect that the current work will serve as a foundation for exploring further sophisticated designs of MLP-based architectures and the efficient implementation of neural networks with desirable implicit biases.

Exploring the potential of MLPs with structured weights further will be interesting. Theoretically evaluating the optimal width and sparsity could be an exciting research topic (Edelman et al., 2023). As we noted, the solvability of global minima and dynamics in mixing layers, even with

linear activation, remains uncertain, and the theory has yet to fully address or circumvent this issue. It would be also interesting to clarify whether other potential candidates for memory-friendly architectures with desirable inductive biases. In particular, weight sharing is known to perform well for image domains, occasionally outperforming networks without weight sharing (Ott et al., 2020). Given that the Mixers can be seen as approximations of MLPs with weight-shared Monarch matrices, it will be an interesting theme to evaluate the validity of such approximations.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgment

RK acknowledges the funding support from JST FOREST (Grant Number: JPMJFR226Q) and JSPS KAEKENHI (Grant Number: 22H05116, 23K16965).

References

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

Bachmann, G., Anagnostidis, S., and Hofmann, T. Scaling mlps: A tale of inductive bias. In *Advances in Neural Information Processing Systems*, 2023.

Bai, Z. and Silverstein, J. W. *Spectral analysis of large dimensional random matrices*, volume 20. Springer, 2010.

Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. Understanding batch normalization. *Advances in Neural Information Processing Systems*, 2018.

Cordonnier, J.-B., Loukas, A., and Jaggi, M. On the relationship between self-attention and convolutional layers. In *International Conference on Learning Representations*, 2019.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123, 2019.

Dao, T., Chen, B., Sohoni, N. S., Desai, A., Poli, M., Grogan, J., Liu, A., Rao, A., Rudra, A., and Ré, C. Monarch: Expressive structured matrices for efficient and accurate training. In *International Conference on Machine Learning*, pp. 4690–4721. PMLR, 2022.

d'Ascoli, S., Sagun, L., Biroli, G., and Bruna, J. Finding the needle in the haystack with convolutions: on the benefits of architectural bias. *Advances in Neural Information Processing Systems*, 2019.

Dettmers, T. and Zettlemoyer, L. Sparse networks from scratch: Faster training without losing performance. *arXiv preprint arXiv:1907.04840*, 2019.

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*, 2021.

Edelman, B. L., Goel, S., Kakade, S., Malach, E., and Zhang, C. Pareto frontiers in neural feature learning: Data, compute, width, and luck. In *Advances in Neural Information Processing Systems*, 2023.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Folland, G. B. *Real Analysis: Modern Techniques and Their Applications*. John Wiley & Sons, 2013.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Fu, D. Y., Arora, S., Grogan, J., Johnson, I., Eyuboglu, S., Thomas, A. W., Spector, B. F., Poli, M., Rudra, A., and Re, C. Monarch mixer: A simple sub-quadratic gemm-based architecture. In *Advances in Neural Information Processing Systems*, 2023.

Gadhikar, A. H., Mukherjee, S., and Burkholz, R. Why random pruning is all we need to start sparse. In *International Conference on Machine Learning*, pp. 10542–10570. PMLR, 2023.

Golubeva, A., Neyshabur, B., and Gur-Ari, G. Are wider nets better given the same number of parameters? In *International Conference on Learning Representations*, 2021.

Hameed, M. G. A., Tahaei, M. S., Mosleh, A., and Nia, V. P. Convolutional neural network compression through generalized kronecker product decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 36:771–779, 2022.

Hoff, P. D. Multilinear tensor regression for longitudinal relational data. *The annals of applied statistics*, 9(3): 1169, 2015.

Hoff, P. D. Lasso, fractional norm and structured sparse estimation using a hadamard product parametrization. *Computational Statistics & Data Analysis*, 115:186–198, 2017.

Hwang, J. Y., Lee, J. O., and Schnelli, K. Local law and Tracy–Widom limit for sparse sample covariance matrices. *The Annals of Applied Probability*, 29(5):3006 – 3036, 2019.

Karakida, R. and Osawa, K. Understanding approximate fisher information for fast convergence of natural gradient descent in wide neural networks. *Advances in neural information processing systems*, 2020.

Karakida, R., Takase, T., Hayase, T., and Osawa, K. Understanding gradient regularization in deep learning: Efficient finite-difference computation and implicit bias. In *International Conference on Machine Learning*, pp. 15809–15827. PMLR, 2023.

Liu, S., Chen, T., Chen, X., Shen, L., Mocanu, D. C., Wang, Z., and Pechenizkiy, M. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *International Conference on Learning Representations*, 2022.

Magnus, J. R. and Neudecker, H. *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons, 2019.

Martens, J. and Grosse, R. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pp. 2408–2417. PMLR, 2015.

Neyshabur, B. Towards learning convolutions from scratch. In *Advances in Neural Information Processing Systems*, 2020.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

Nguyen, T., Raghu, M., and Kornblith, S. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021.

Ott, J., Linstead, E., LaHaye, N., and Baldi, P. Learning in the machine: To share or not to share? *Neural Networks*, 126:235–249, 2020.

Pellegrini, F. and Biroli, G. Neural network pruning denoises the features and makes local connectivity emerge in visual tasks. In *International Conference on Machine Learning*, pp. 17601–17626. PMLR, 2022.

Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Sahiner, A., Ergen, T., Ozturkler, B., Pauly, J., Mardani, M., and Pilanci, M. Unraveling attention via convex duality: Analysis and interpretations of Vision Transformers. In *International Conference on Machine Learning*, pp. 19050–19088. PMLR, 2022.

Saxe, A., McClelland, J., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proceedings of the International Conference on Learning Representations*, 2014.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. MLP-Mixer: An all-MLP architecture for vision. In *Advances in Neural Information Processing Systems*, 2021.

Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., et al. ResMLP: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Wightman, R. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., and Srebro, N. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pp. 3635–3673. PMLR, 2020.

Yasuda, T., Bateni, M., Chen, L., Fahrbach, M., Fu, G., and Mirrokni, V. Sequential attention for feature selection. In *International Conference on Learning Representations*, 2023.

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. MetaFormer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 10819–10829, 2022.

Zhang, A., Tay, Y., Zhang, S., Chan, A., Luu, A. T., Hui, S., and Fu, J. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters. In *International Conference on Learning Representations*, 2021.

Zhou, S., Wu, J.-N., Wu, Y., and Zhou, X. Exploiting local structures with the Kronecker layer in convolutional networks. *arXiv:1512.09194*, 2015.

A. Details of Architectures

Here, we overview more technical details of all models: MLP-Mixer, Simple Mixer (S-Mixer), and MLP with sparse weights (SW-MLP). In Section A.1, we introduce the transformation from the input image to the first hidden layer. In Section A.2, we overview some detailed formulation of the models including skip connection and layer normalization.

Table S.1 . Summary table of models. The notation hidden in the block means whether the model has a hidden layer in each MLP block. The symbol (*) indicates the setting depends on the kind of experiment. Here, GAP is the global average pooling.

Model	Per-patch FC	Skip-conn.	Layer-norm	Hidden in the block	GAP
SW-MLP	✓	✓	✓	(*)	✓
(RP) MLP-Mixer	✓	✓	✓	✓	✓
(RP) S-Mixer	✓	✓	✓		✓
Shallow Kronecker					
Shallow Monarch					

A.1. Per-patch FC Layer

The first layer of the MLP-Mixer is given by the so-called per-patch FC layer, which is a single-layer channel mixing. In all experiments, for a patch size P, the input image is decomposed into HW/P^2 non-overlapping image patches with size $P \times P$; we rearrange the $H \times W$ input images with 3 channels into a matrix whose size is given by $(HW/P^2) \times 3P^2 = S_0 \times C_0$. For the rearranged image $X \in \mathbb{R}^{S_0 \times C_0}$, the per-patch fully connected (FC) layer is given by

$$Y = XW^\top, \tag{S.1}$$

where W is a $C \times C_0$ weight matrix. We use the per-patch FC layer not only for Mixers but also for SW-MLP.

Remark on per-patch FC layer: The original study set the size of the mixing layers to $S = S_0$. In contrast, to investigate the contribution of each input image size and hidden layer size independently, it is rational to change (S, C) independent of (S_0, C_0). Therefore, we make the per-patch FC transform the input size C_0 to the output size C and the first token mixing layer transform S_0 to S.

A.2. MLP-Mixer and S-Mixer

Let us denote a block of the MLP-Mixer by

$$f_{W_1,W_2}(X) = \phi(XW_1^\top)W_2^\top, \tag{S.2}$$

and that of the S-Mixer by

$$f_{W_1}(X) = \phi(XW_1^\top). \tag{S.3}$$

We set $\phi = \mathrm{GELU}$.

A.2.1. MLP-Mixer

We set the layer normalization (LN) by

$$\mathrm{LN}(X) = \frac{X - m(X)}{\sqrt{v(X) + \epsilon}} \odot \gamma + \beta, \quad X \in \mathbb{R}^{S \times C}, \tag{S.4}$$

where $\odot$ denotes the Hadamard product , $m(X)$ (resp. $v(X)$) is the empirical mean (resp. the empirical variance) of X with respect to the channel axis, and γ, β are trainable parameters. We set $\epsilon = 10^{-5}$ in all experiments.

In the implementation of fully connected layers, we use only the right matrix multiplications in the same way as the original MLP-Mixer (Tolstikhin et al., 2021). A token-mixing block $X \mapsto U$ of MLP-Mixer is given by

$$U = X + f_{W_1,W_2}(\mathrm{LN}(X)^\top)^\top, \tag{S.5}$$

where W_1 is $S \times \gamma S$ and W_2 is $\gamma S \times S$. Similarly, we set a channel-mixing block $U \mapsto Y$ as

$$Y = U + f_{W_3,W_4}(\mathrm{LN}(U)), \tag{S.6}$$

where W_3, W_4 are weight matrices.

We refer to the composed function $X \mapsto Y$ of the token-mixing block and the channel-mixing one as *a base block* of the MLP-Mixer. The MLP-Mixer with L-blocks is composed in the order of the per-patch FC layer, the L base blocks, and the global average pooling with the layer normalization, and the last fully connected classification layer.

A.2.2. S-MIXER

The S-Mixer without random permutations is implemented by replacing the MLP-block f_{W_1,W_2} and f_{W_3,W_4} in the MLP-Mixer with FC blocks. That is, token-mixing and channel-mixing blocks are given by

$$U = X + f_W(\mathrm{LN}(X)^\top)^\top, \tag{S.7}$$

$$Y = U + f_V(\mathrm{LN}(U)), \tag{S.8}$$

where W and V are weight matrices. The transpose of the input matrix in the token-mixing block is implemented by rearrangement of entries. We decided to apply both skip-connection and layer normalization even in the S-Mixer. This is a rather technical requirement for ensuring the decrease of training loss in deep architectures.

A.2.3. MIXERS WITH PERMUTED KRONECKER LAYERS

Here we implement generalized MLP-Mixer and S-Mixer with permutation matrices and PK-layers. Recall that for any matrix X,

$$X^\top = \mathrm{Mat}(J_c \mathrm{vec}(X)), \tag{S.9}$$

where J_c is the $m \times m$ commutation matrix. Therefore, the token-mixing block of the S-Mixer is

$$U = X + \mathrm{Mat} \circ J_c^\top \circ \mathrm{vec} \circ f_W \circ \mathrm{Mat} \circ J_c \circ \mathrm{vec} \circ \mathrm{LN}(X) \tag{S.10}$$

$$= X + \mathrm{Mat} \circ \text{PK-Layer}_W(\mathrm{LN}(X); J_c, J_c^\top). \tag{S.11}$$

Similarly, the channel-mixing block of the S-Mixer is equal to

$$Y = U + \mathrm{Mat} \circ \text{PK-Layer}_V(\mathrm{LN}(U); I, I), \tag{S.12}$$

where I is the identity matrix. Note that skip-connections gather J_c and $J_c^\top$ in the same mixing block for compatibility in shapes of hidden units.

To get examples of PK family and to generalize Mixers, we implement the random permuted (RP) S-Mixer with skip-connections by replacing J_c and $J_c^\top$ with i.i.d. random permutation matrices J_1 and J_2:

$$U = X + \mathrm{Mat} \circ \text{PK-Layer}_W(\mathrm{LN}(X); J_1, J_2). \tag{S.13}$$

We implement the random permutations by random shuffling of output $m = SC$ indexes of vectors. We freeze it during the training step. Note that we avoid using an $m \times m$ matrix representation of J_x for memory efficiency. We implement the random permuted (RP) MLP-Mixer by the same way as the RP-S-Mixer.

A.2.4. THE SKIP-CONNECTION IN THE FIRST BLOCK

The first token-mixing block has the input shape (S_0, C) and the output shape (S, C). However, we need to change S with fixing S_0 in some experiments. To control the difference of S_0 and S, we set the first token-mixing block as follows:

$$U = \text{SkipLayer}(X) + \text{PK-Layer}_W(\mathrm{LN}(X); J_1, J_2), \tag{S.14}$$

where the skip layer is given by

$$\text{SkipLayer}(X) = \mathrm{LN}(\tilde{W} X), \tag{S.15}$$

where $\tilde{W}$ is a $S \times S_0$ weight matrix. For a fair comparison, we use the skip layer even if $S = S_0$ in the experiments that we sweep S. We use the same setting for the MLP-Mixer as for the S-Mixer.

A.2.5. SPARSE-WEIGHT (SW) MLP

Let $0 < p \leq 1$ and $m \in \mathbb{N}$. We implement each matrix of a static sparse weight FC block with the freezing rate p as follows:

$$x \mapsto x + \phi((M \odot W)\,\mathrm{LN}(x)), \quad x \in \mathbb{R}^m, \tag{S.16}$$

where M is the mask matrix whose $m^2 p$ entries are randomly chosen and set to be one with a probability p and the others are set to be zero with a probability $1 - p$. The mask matrix M is initialized before training and it is frozen during training.

We also consider the SW-MLP consists of sparse weight MLP-blocks as follows:

$$x \mapsto x + \phi\left((M_2 \odot W_2)\phi((M_1 \odot W_2)\,\mathrm{LN}(x))\right), \quad x \in \mathbb{R}^m. \tag{S.17}$$

W_1 and W_2 are weight matrices with hidden features γm, where γ is an expansion factor. M_1, M_2 are mask matrices whose $\gamma m^2 p$ entries are randomly chosen and set to be one with a probability p and the others are set to be zero with a probability $1 - p$.

SW-MLP with L-blocks is composed in the order of the per-patch FC layer, vectorization, L static sparse weight FC blocks (or MLP blocks), and the last classification FC layer.

B. Analysis

B.1. Derivation of Proposition 4.1

For $H = \phi(\phi(WX)V)$, by using $\mathrm{vec}(WXV) = (V^\top \otimes W)\mathrm{vec}(X)$, we have

$$\mathrm{vec}(H) = \phi((V^\top \otimes I_S)\mathrm{vec}(\phi(WX)) \tag{S.18}$$

$$= \phi((V^\top \otimes I_S)\phi((I_C \otimes W)x)). \tag{S.19}$$

Because $J_c^\top (A \otimes B) J_c = (B \otimes A)$ (Magnus & Neudecker, 2019) and any permutation matrix J is commutative with the entry-wise activation function: $J\phi(x) = \phi(Jx)$, we obtain

$$\mathrm{vec}(H) = \phi(J_c^\top (I_S \otimes V^\top)\phi(J_c(I_C \otimes W)x)). \tag{S.20}$$

The skip-less MLP-Mixer (4) is expressed as follows: $Y = \text{Channel-MLP}(\text{Token-MLP}(X)))$, and then

$$u = \phi(J_c(I_C \otimes W_2)\phi((I_C \otimes W_1)x)),$$
$$y = \phi(J_c^\top (I_S \otimes W_4^\top)\phi((I_S \otimes W_3^\top)u)).$$

where $u = \mathrm{vec}(U)$ and $y = \mathrm{vec}(Y)$.

It may be informative that a similar transformation between the matrix and vector is used in a completely different context of deep learning, that is, the Kronecker-factored Approximate Curvature (K-FAC) computation for natural gradient descent (Martens & Grosse, 2015). K-FAC assumes layer-wise preconditioner given by the Kronecker product, that is, $(B \otimes A)^{-1}\mathrm{vec}(\nabla_W Loss(W))$ where A and B correspond to the Gram matrices of the forward and backward signals. This K-FAC gradient can be computed efficiently because it is reduced to a matrix computation of $A^{-1}\nabla_W Loss(W)(B^\top)^{-1}$. Therefore, the trick of formulating a large matrix-vector product for the product among relatively small matrices is common between K-FAC and the aforementioned effective expression. It is noteworthy that an approximation using the Kronecker product, which may seem rough at first glance, can sometimes be sufficiently effective. For instance, K-FAC has the ability to provide exact gradient values in over-parameterized linear models (Karakida & Osawa, 2020).

B.2. Product of Random Permutations

The uniformly distributed random $m \times m$ permutation matrix is given by $J = J_g$ where g is the uniformly distributed random variable on the permutation group S_m of m elements. Then the uniform distribution over S_m is the Haar probability measure, which is translation invariant (see (Folland, 2013) for the detail), that is, $J = J_\sigma J_g$ is also uniformly distributed if σ is S_m-valued uniformly distributed random variables and $g \in S_m$ is constant. Therefore, $J = J_\sigma J_\rho$ is a uniformly distributed random permutation matrix for independent and uniformly distributed random variables σ and ρ on S_m.

B.3. Representation as a PK-family

By (10), one can see that the block of the S-Mixer is

$$U = \text{PK-Layer}_W(X; I, J_c),\ \text{PK-Layer}_{V^\top}(U; I, J_c^\top) \tag{S.21}$$

For MLP-Mixer, the Token-MLP is

$$U = \text{PK-Layer}_{W_2}(\text{PK-Layer}_{W_1}(X; I, J_1); J_1^\top, J_c) \tag{S.22}$$

and Channel-MLP is

$$\text{PK-Layer}_{W_4^\top}(\text{PK-Layer}_{W_3^\top}(U; I, J_2); J_2^\top, J_c^\top) \tag{S.23}$$

for the arbitrary permutation matrices J_1 and J_2.

B.4. Proof of implicit regularization

Here we prove Proposition 3.3 and show a connection between that and (Hoff, 2017). Since each weight V, W are C^2, S^2 dimensional vectors, we only need to show the following Lemma B.1.

Lemma B.1. *Let $m, n \in \mathbb{N}$. We write $||x||_p$ for the L_p norm of any real vector x. Let $f : \mathbb{R}^{mn} \to \mathbb{R}$. Set $h : \mathbb{R}^{mn} \to \mathbb{R}$, $g : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ as follows:*

$$\phi(\beta) = f(\beta) + \lambda||\beta||_2, \tag{S.24}$$

$$h(\beta) = f(\beta) + \frac{\lambda}{\sqrt{mn}}||\beta||_1, \tag{S.25}$$

$$g(u, v) = f(u \otimes v) + \lambda(||u||_2^2 + ||v||_2^2)/2. \tag{S.26}$$

Set

$$\Theta_p := \{u \otimes v \mid u \in \mathbb{R}^m, v \in \mathbb{R}^n\}. \tag{S.27}$$

Then it holds that

$$\inf_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} g(u, v) = \inf_{\beta\in\Theta_p} \phi(\beta) \geq \inf_{\beta\in\mathbb{R}^{mn}} h(\beta). \tag{S.28}$$

Proof. Since $||u \otimes v||_2 = ||u||_2||v||_2$, we have

$$\inf_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} g(u, v) = \inf_{\beta\in\Theta_p} \left[f(\beta) + \lambda/2 \inf_{v\in\mathbb{R}^n} (||\beta||_2^2/||v||_2^2 + ||v||_2^2) \right]. \tag{S.29}$$

The inner $\inf$ achieves the maximum when $||v||_2^2 = ||\beta||_2$, and the maximum is $||\beta||_2$. (Note that it is **not squared** L2-norm.) Therefore,

$$\inf_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} g(u, v) = \inf_{\beta\in\Theta_p} (f(\beta) + \lambda||\beta||_2). \tag{S.30}$$

Lastly, by the Hölder's inequality, it hold that $||\beta||_1 \leq ||\beta||_2\sqrt{mn}$, which complete the proof. □

To describe the connection of the implicit L1 regularization in the case of Hadamard product (Hoff, 2017), consider matrix $\mathbf{1}\mathbf{1}^\top$, where $\mathbf{1}$ is the vector with all entries are one. For simplicity, consider the case $S = C$. It is easy to recover the general case by changing the proportional coefficients. Then we have

$$\text{Tr}(W^\top W \otimes \mathbf{1}\mathbf{1}^\top) + \text{Tr}(\mathbf{1}\mathbf{1}^\top \otimes V^\top V) = C^2||W||_F^2 + S^2||V||_F^2 = C^2(||W||_F^2 + ||V||_F^2) \tag{S.31}$$

Put

$$g(W, V) := \mathcal{L}(W \otimes V) + \lambda/2(\|W\|_F^2 + \|V\|_F^2). \tag{S.32}$$

We have

$$g(W, V) = \mathcal{L}(W \otimes V) + \frac{\lambda}{2C^2}(\mathrm{Tr}(W^\top W \otimes \mathbf{1}\mathbf{1}^\top) + \mathrm{Tr}(\mathbf{1}\mathbf{1}^\top \otimes V^\top V)) \tag{S.33}$$

Here consider the vectorization: $u(W) := \mathrm{vec}(W \otimes \mathbf{1}\mathbf{1}^\top)$, $v(V) := \mathrm{vec}(\mathbf{1}\mathbf{1}^\top \otimes V)$. Then

$$u(W) \odot v(V) = \mathrm{vec}(W \otimes V). \tag{S.34}$$

Thus we have

$$\inf_{W,V} g(u, v) = \inf_{W,V} \mathcal{L}(u \circ v) + \frac{\lambda}{2C^2}(\|u\|_2^2 + \|v\|_2^2), \tag{S.35}$$

Because of the domain of u and v, we have

$$\inf_{W,V} g(u, v) \geq \inf_{u,v \in \mathbb{R}^{S^2C^2}} \mathcal{L}(u \circ v) + \frac{\lambda}{2C^2}(\|u\|_2^2 + \|v\|_2^2). \tag{S.36}$$

By Hoff (2017), the right-hand side is equal to

$$\inf_{\beta \in \mathbb{R}^{S^2C^2}} \mathcal{L}(\beta) + \frac{\lambda}{2C^2}\|\beta\|_1. \tag{S.37}$$

Thus the inequality for the Kronecker product has a relationship with (Hoff, 2017) from the vectorization and Kronecker product with $\mathbf{1}\mathbf{1}^\top$.

Let us focus on the normalization factor $1/mn$ and $1/C^2$. In general, there is a trivial relationship between L1 and L2 regularization as follows: For $\theta \in \mathbb{R}^M$,

$$\inf_\theta \mathcal{L}(\theta) + \sqrt{M}\lambda\|\theta\|_2 \geq \inf_\theta \mathcal{L}(\theta) + \lambda\|\theta\|_1, \tag{S.38}$$

this directly follows from the Hölder's inequality. However, in our situation, the parameter space of $W \otimes V$ is a subspace of $\mathbb{R}^{S^2C^2}$, so it is not clear whether the same constant regularization for both L1 and L2 regularization is suitable. At the initialization, we set entries of W, V independently distributed with $\mathcal{N}(0, 1/C)$, thus by the law of large numbers,

$$\|W\|_F^2, \|V\|_F^2 = O(C). \tag{S.39}$$

In the case of dense MLP, we initialize β as $\mathcal{N}(0, 1/SC)$, thus

$$\|\beta\|_1 = O(S^2C^2/\sqrt{SC}) = O(C^3). \tag{S.40}$$

Thus

$$\frac{\|\beta\|_1}{\|W\|_F^2}, \frac{\|\beta\|_1}{\|V\|_F^2} = O(C^2). \tag{S.41}$$

Therefore, the normalizing factor $1/C^2$ in (S.37) matches the scale transformation according to the dimension of parameter spaces. Without the normalizing factor, the L1 regularization term will be C^2 times larger than the loss $\mathcal{L}$.

C. Experimental Setting

C.1. Figure 2

We set the number of blocks of the MLP-Mixer, denoted by L, to 3. Both the Token-mixing block and the Channel-Mixing block are present L times each, resulting in a total of 6 blocks when counted separately. We also set $\gamma = 2$.

For the comparison, the sparse-weight MLP replaces the components within the MLP-Mixer, namely $(I_C \otimes W_1), (I_C \otimes W_2)$ and $(W_3^\top \otimes I_S), (W_4^\top \otimes I_S)$, with the form $M \odot A$. In this context, there's no distinction between the token-mixing and the channel-mixing blocks, leading to a total of 6 blocks.

In Figure 2 (a), we compare the MLP-Mixer with $S = C = 64, 32$ and the SW-MLP. The sparsity of the SW-MLP is taken as 2^{-n} where n ranges from 0 to 10. We set the patch size as 4. Each network is trained on CIFAR10 with a batch size of 128, for 600 epochs, a learning rate of 0.01, using auto-augmentation, AdamW optimizer, momentum set to 0.9, and cosine annealing. We utilize three different random seeds for each training.

Figure 2(b) shows the CKA (for a specific seed) of the MLP-Mixer with $S = C = 64$ and the MLP with a sparsity of 1/64, based on the results from Figure 2(a). However, the features targeted for CKA are taken from the layer just before the skip-connection in each block, and they are displayed on the axis in the order of proximity to the input, labeled as 1 through 6. Figure 2(c) similarly compares the dense MLP (i.e. sparsity$= 1$).

Figure 2(d) shows the test error of the shallow MLP with monarch matrix weight and Kronecker weight matrix. The training uses MNIST, with optimizer adamw with 200 epochs, and the cosine annealing of leaning rate with an initial value of 0.01. Table S.1 summarizes the difference of the shallow models between other treated models.

C.2. Figure 4

In Figure 4 (left), we compare the MLP-Mixer and the MLP under the same training settings as in Figure 2(a). Now, here $\Omega = 2^{19}$ and $\gamma = 2$, with C values being 4,8,12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 64. The S values are determined corresponding to C while keeping Ω fixed, resulting in values: 360, 252, 203, 173, 152, 136, 124, 113, 104, 96, 89, 83, 64. We utilized four different random seeds for each training.

In Figure 4 (right), we used the same values on Ω, γ and C, S as the Figure 4 (left). We run 10 trials with different random seeds to plot the singular values of the sparse weight matrix.

C.3. Table 1

CIFAR10, CIFAR100 We used single Tesla V100 GPU to compute the runtime. The training is by AdamW with the mini-batch size 128. We used a 32-bit float. The runtime is averaged over 600 epochs. The averaged runtime is on four different random seeds.

For our experiments, we utilized Tesla V100 GPUs, accumulating approximately 300 GPU hours. The networks were trained on datasets, either CIFAR-10 or CIFAR-100. We set $L = 2$ and $\gamma = 4$ for the Mixer-SS/8 and Mixer-SS-W. For the Mixer-SS/8 configuration, the parameters were set as $p = 8$, $S = 16$, and $C = 986$. In our wider setting (Mixer-SS-W), the parameters were defined as $p = 4$, $S = 64$, and $C = 487$. The training was conducted over 4000 epochs with the cosine-annealing. We employed the AdamW optimizer and incorporated auto-augmentation techniques. The chosen mini-batch size was 1024, and experiments were run with three distinct random seeds. To optimize results, we conducted a hyper-parameter search for the best initial learning rate from the set $\{0.04, 0.05, 0.06\}$. The rate that provided the highest accuracy, when averaged over the random seeds, was subsequently chosen for experiments. Specifically, the learning rate 0.06 was selected for Mixer-SS/8 on CIFAR-100, while 0.04 was the preferred rate for all other scenarios.

ImageNet-1k For the Mixer-B-W, we set $L = 8, p = 14, C = 588, S = 256$, and $\gamma = 4.57$. The other settings are the same as the other experiments on ImgeNet-1k (Appendix C.5).

C.4. Figure 5

We utilized Tesla V100 GPUs and approximately 400 GPU hours for this experiment. We trained three types of MLPs; S-Mixer, RP S-Mixer, and SW-MLP architectures. All MLPs incorporated a per-patch FC layer as the first block, with a patch size of $P = 4$. The input token size was fixed at $S_0 = (32/P)^2 = 64$. We trained the models on the CIFAR-10, CIFAR-100, and STL-10 datasets, along with data augmentations such as random cropping and random horizontal flipping. The input images were resized to a size of $32 \times 32 \times 3$. We employed Nesterov SGD with a mini-batch size 128 and a momentum of 0.9 for training, running for 200 epochs. The initial learning rate was set to 0.02, and we used cosine annealing for learning rate scheduling. To ensure robustness, we conducted three trials for each configuration and reported the mean and standard deviation of the results. Unless otherwise specified, these settings were used throughout the whole study on CIFAR-10, CIFAR-100, and STL-10.

(i) S-Mixer and RP S-Mixer We conducted training experiments on the S-Mixer architecture with eight blocks. In order to explore various cases of integer pairs (S, C) that approximately satisfy the equation

$$\Omega = \frac{CS^2 + SC^2}{2}. \tag{S.42}$$

The number of connections, denoted as Ω, was fixed at $\Omega = 2^{18}, 2^{21}$ and 2^{27}. For each value of Ω, the pairs (C, S) were chosen in a symmetric manner. It should be noted that if $(C, S) = (a, b)$ is a solution, then $(C, S) = (b, a)$ is also a solution. The selected pairs for each value of Ω are as follows:

- $\Omega = 2^{18}$: $(C, S) = (16, 173), (32, 113), (48, 83), (83, 48), (113, 32), (173, 16)$.
- $\Omega = 2^{21}$: $(C, S) = (16, 504), (32, 346), (64, 226), (226, 64), (346, 32), (504, 16)$.
- $\Omega = 2^{27}$: $(C, S) = (128, 1386), (256, 904), (904, 256), (1386, 128)$.

(ii) SW-MLP. For a fair comparison between the Mixers and SW-MLPs, we set the first layer of both models to the same per-patch FC structure. We trained SW-MLPs with eight blocks, where the hidden layers of these MLPs share a common $\Omega = 2^{18}$ connectivity pattern. Following the per-patch fully connected layer, the feature matrix is vectorized and processed as a standard MLP with masked sparse connectivity.

For each freezing rate $1 - p$, we determined the width m of the hidden units using the equation:

$$\Omega = m^2 p, \quad m = \sqrt{\frac{\Omega}{p}}. \tag{S.43}$$

We set $1 - p = 0.1, 0.3, 0.5, 0.7, 0.9$, which correspond to $m = 540, 612, 724, 935, 1619$, respectively.

C.5. Figure 6

We set the number of blocks L and the other hyper-parameters as in Table S.2 .

Table S.2 . Hyper-parameters of models for normal/RP Mixers in Figure 6.

dataset	L	Ω	γ	optimizer	init. lr	mini-batch	epoch	hard aug.
CIFAR10	12	2^{21}	None	AdamW	0.02	1024	2000	✓
CIFAR100	12	2^{21}	None	SGD	0.1	128	600	
STL10	12	2^{18}	None	SGD	0.1	128	2000	
ImageNet-1k	8	290217984	4	AdamW	0.001	4096	300	✓

CIFAR-10, 100, STL-10. We utilized Tesla V100 GPUs and approximately 200 GPU hours for our experiments. We used a single GPU per each run. We set $\Omega = 2^{18}$ and used the same pairs (S, C) satisfying equation S.42 as Sec. C.4.

We set the initial learning rate to 0.1. On CIFAR-10, we trained models 200 epochs, 600-epochs on CIFAR-100, and we did five trials with random seeds. We trained models 2000-epochs on STL-10 with three random seeds.

ImageNet-1k. We utilized Tesla V100 GPUs and approximately 4000 GPU hours for our experiments. for training MLP-Mixer and RP MLP-Mixer on ImageNet-1k; we used a GPU cluster of 32 nodes of 4 GPUs per node for each run.

We set the expansion factor $\gamma = 4$ for both token-mixing MLP and channel-mixing MLP. We set $\Omega = 290217984 = (768^2 \cdot 196 + 768 \cdot 196^2)\gamma/2$ on a baseline $P = 16, (S, C) = (196, 768)$. We sweep $P = 7, 8, 14, 16, 28, 32$ and set $C = 3P^2$ and set S so that it approximately satisfies the equation

$$\Omega = \frac{\gamma(CS^2 + SC^2)}{2}. \tag{S.44}$$

For each setting, we did three trials with random seeds.

Training on the ImageNet-1k is based on the timm library (Wightman, 2019). We used AdamW with an initial learning rate of 10^{-3} and 300 epochs. We set the mini-batch size to 4096 and used data-parallel training with a batch size of 32 in each GPU. We use the warm-up of with the warm-up learning rate 10^{-6} and the warm-up epoch 5. We used the cosine annealing of the learning rate with a minimum learning rate 10^{-5}. We used the weight-decay 0.05. We applied the random erasing in images with a ratio of 0.25. We also applied the random auto-augmentation with a policy rand-m9-mstd0.5-inc1. We used the mix-up with $\alpha = 0.8$ and the cut-mix with $\alpha = 1.0$ by switching them in probability 0.5. We used the label smoothing with $\varepsilon = 0.1$.

C.6. Figure 7

For our experiments in Figure 7, we utilized Tesla V100 GPUs, with approximately 70 GPU hours utilized. We trained both S-Mixer and RP S-Mixer models on CIFAR-10, CIFAR-100, and STL-10 datasets. We considered different numbers of blocks, specifically $L = 4, 8, 12, 16, 20$. The values of S and C were fixed at 128. Each configuration was evaluated using three trials with different random seeds.

D. Supplementary Experiments

D.1. Trainability of highly sparse weights

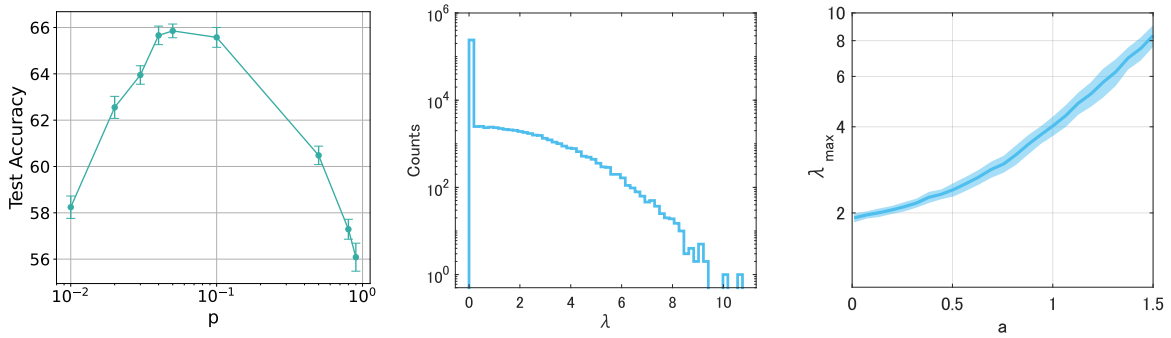

Figure S.1 . On the trainability of SW-MLP. (Left) Trainability decreases as the sparsity $1 - p$ becomes too high. We set γ according to (Golubeva et al., 2021) under fixed $\Omega = 2^{16}$. We performed five trials for each p with random seeds. (Center) The singular value distribution of the sparse weight at random initialization. We set $a = 1.5$, $\Omega = 10^3$ and performed 50 trials. (Right) The largest eigenvalue monotonically increases as the sparsity increases.

Golubeva et al. (2021) found that as the sparsity (width) increased to some extent, the generalization performance improved. They also reported that if the sparsity became too high, the generalization performance slightly decreased. They discussed that this decrease was caused by the deterioration of trainability, that is, it became difficult for the gradient descent to decrease the loss function. In fact, we confirmed their decrease of the performance in SW-MLP as is shown in Figure S.1 (left). In contrast, we hardly observed such a decrease in performance for the Mixers. This seems rational because we can take an arbitrary small sparsity $1 - p$ for the SW-MLP while it is lower-bounded for the Mixers as is described in Section 4.

As a side note, we give here quantitative insight into the trainability from the perspective of the singular value of the weight matrix. Some previous work reported that the large singular values of weight matrices at random initialization cause the deterioration of trainability in deep neural networks (Bjorck et al., 2018). Following this line of study, let us consider a random weight of SW-MLP. Set the width by $m = \Omega^{(1+a)/2}$ and the sparsity by $p = 1/\Omega^a$ with a constant $a > 0$ and take a large Ω limit. We use this scaling because our interest is in the case where the expected number of weights, i.e., $m^2 p = \Omega$, is independent of the scaling of p. We generate $Z = M \odot W$ where $W_{ij} \sim \mathcal{N}(0, 1)$ and M is a static mask matrix whose entries are given by the Bernoulli distribution with probability p. The singular value of the weight matrix Z is equivalent to the square root of the eigenvalue of $Q = ZZ^\top$. Because we are uninterested in a trivial scale factor, we scaled the matrix Q as Q/c where c denotes the average over the diagonal entries of Q. This makes the trace of Q, that is, the summation (or average) of eigenvalues, a constant independent of a. We computed the eigenvalues of Q and obtained the singular values of Z denoted by λ.

As is shown in Figure S.1 (center), the spectrum of the singular values peaked around zero but widely spread up to its edge. Figure S.1 (right) demonstrates that the largest singular value becomes monotonically large for the increase of a. Because the larger singular value implies the lower trainability (Bjorck et al., 2018), this is consistent with the empirical observation of (Golubeva et al., 2021) and our Figure S.1 (left).

In contrast, the Mixers are unlikely to suffer from the large singular values as follows. Suppose S-Mixer with $S = C \gg 1$ for simplicity. Then, each layer of the effective MLP has $p = 1/C$ which corresponds to the scaling index $a = 1/3$ in SW-MLP. Actually, its singular value becomes further better than $a = 1/3$, because the weight matrices of the normal and RP Mixers are structured: Consider the singular values of $Z = J_2(I_C \otimes W)J_1$ with a $C \times C$ random Gaussian matrix W and permutation matrices (J_1, J_2). Then, the singular values of Z are equivalent to those of W, excluding duplication. Therefore, the singular values of the Mixers are determined only by the dense weight matrix W. Define $Q = WW^\top$. Because the normalized matrix Q/c obeys the Marchenko-Pastur law in the random matrix theory and its largest eigenvalue is given by 4 in the infinite C limit (Bai & Silverstein, 2010). This means that the largest singular value of the normalized W is 2 and corresponds to $a = 0$ of SW-MLP (i.e., dense MLP) with the infinite Ω limit in Figure S.1 (right). Thus, we can say that from the perspective of random weights, the trainability of the Mixers is expected to be better than that of SW-MLP.

We can also extend our analysis to the models incorporating the expansion factor γ. For SW-MLP with MLP blocks, the expected number of weights is given by $\Omega = \gamma p m^2$. We just need to replace p in the S-Mixer case to γp and the monotonic increase of the largest singular value appears as well. For the MLP-Mixer, its normalized W is a $\gamma C \times C$ matrix. According to the Marchenko-Pastur law for rectangular random matrices, as $C \to \infty$, the largest singular value approaches a constant value of $1 + \sqrt{\gamma}$. This corresponds to the singular value of $a = 0$ in the corresponding SW-MLP, and the result is similar as in the S-Mixer.

Note that the effective width of the mixing layers is sufficiently large but still has an upper bound (15). It satisfies

$$(\sqrt{1 + 8\Omega/\gamma} - 1)/2 \leq m \leq (\Omega/\gamma)^{2/3}, \tag{S.45}$$

where the equality of the lower bound holds for $S = 1$ or $C = 1$. In contrast, for SW-MLP, we have no upper bound and only the lower bound $\sqrt{\Omega} \leq m$, where this equality holds for a dense layer. We can consider an arbitrarily small p and a large m for a fixed Ω if we neglect the issue of memory (Golubeva et al., 2021). Golubeva et al. (2021) reported that extremely small p can cause a decrease in test accuracy owing to the deterioration of trainability. We observed a similar deterioration for the SW-MLP, as shown in this section, but not for the Mixers. This is expected because m is upper-bounded in the Mixers and the trainability is less damaged than that of the SW-MLP with high sparsity.

D.2. MLP-Mixer with increasing width

Figure S.2 shows the test accuracy improves as the width increases on the models SW-MLP and normal and RP MLP-Mixer even if the expansion factor γ is incorporated in the models. We set the expansion factor $\gamma = 4$. We set the initial learning rate to be 0.1. For normal and RP MLP-Mixer, we set $C = 16, 32, 48, 64, 83, 113, 173$ and determined S by combinations of C and $\Omega = 2^{18}, 2^{20}$. For SW-MLP, we set $p = 0.1, 0.3, 0.5, 0.7, 0.9$ and set the width by $m = \sqrt{\Omega/\gamma p}$.

D.3. Increasing expanding factor

The effective MLP expression of the MLP-Mixer (S.21) has two widths: $m = SC$ and γSC. As both are proportional to SC, we have focused on changing SC and fixed γ so far. Here, we consider a complementary setting, that is, changing γ with fixed $m = SC$. By substituting $S = m/C$ into (14), we obtain $\gamma = 2\Omega/(m(C + m/C))$. Similar to m of C shown in Fig. 3, this γ is a single-peak function of C and takes its maximum as

$$C^* = S^* = \sqrt{m}, \quad \max_{S,C} \gamma = \Omega/(m\sqrt{m}). \tag{S.46}$$

Fig. S.3 (left) confirms that increasing the width (γ) leads to performance improvement as is expected from Hypothesis4.1. We trained normal and RP MLP-Mixers with various γ in a realistic range. We plotted some cases of fixed Ω under the same m. Fig. S.3 (right) shows the test accuracy maximized around $C = S$ as is expected from (S.46).

D.4. Dependence on depth

As shown in Figs.5-S.3 , both the normal and RP Mixers exhibited similar tendencies for a fixed depth. Fig. 7 confirms that by increasing the depth, i.e., the number of blocks, RP S-Mixers can even become comparable to the normal ones or

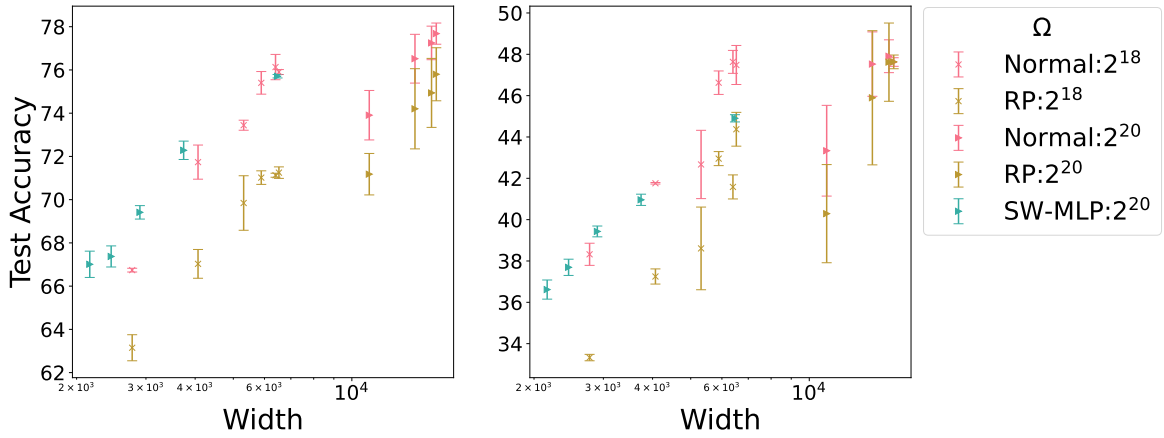

Figure S.2 . Test accuracy improves as the effective width increases. MLP-Mixer, RP-MLP-Mixer, and SW-MLP with $\gamma = 4$ on (Left) CIFAR-10, (Right) CIFAR-100.

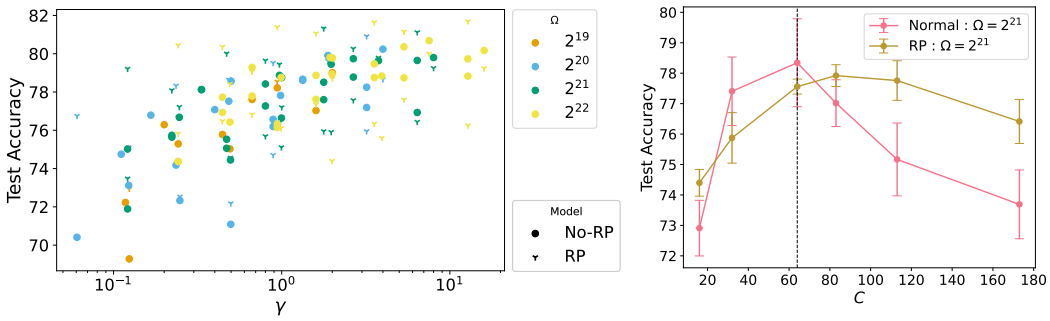

Figure S.3 . (Left) Increasing expansion factor γ improved the test error in normal and RP MLP-Mixers. (Right) The lowest error is achieved around $C = S$ with fixed $m = 4096$.

better than them in some cases. First, we observed that, when the depth was limited, the RP Mixers were inferior to the normal Mixers in most cases. As we increased the depth, we observed that in some cases, overfitting occurred for the normal Mixer, but not for the RP one (see also the training and test losses shown in Section D.6). In such cases, the results of the RP Mixers were comparable (in Figs. 7(left, right)) or better (in Fig. 7(center)). Although RP Mixers are not necessarily better than normal ones, it is intriguing that even RP Mixers defined by a random structure can compete with normal Mixers.

The random permutation in RP-Mixer initially selects and fixes *the size of receptive field* (i.e., the number of input pixels flowing into a neuron in each upper layer through weights) at a rate of $1/\sqrt{m}$. Therefore, the training becomes difficult in shallow networks. However, in the RP-Mixer, since an independent random permutation is chosen for each layer, the receptive field expands as the number of layers increases. From this, the performance of the RP-Mixer improves as layers are added. The normal MLP-Mixer performs well even with fewer layers because the receptive field and diagonal-block structure are aligned.

D.5. Replacement of J_c in specific layers

Figure S.4 provides more detailed insight into the case where the depth is limited and RP Mixers perform worse than normal Mixers. We investigated special S-Mixers whose l-th block was replaced with its RP counterpart while the other layers remained the same. Interestingly, when the accuracy deterioration apparently appears (e.g., cases of CIFAR-10 and STL-10 in Figure 7), this deterioration is attributed to the first block. This seems rational because the neighboring image patches are likely to be correlated, which makes the input units to the first token mixing correlated. Although the usual mixing weights can reflect such neighboring structures, RP Mixers randomly choose tokens and may lose the neighboring structure specific to the images. However, as the depth increases, the token mixing layers can merge all the tokens, which is consistent with the increase in the accuracy of the RP Mixers, as confirmed in Figure 7. Thus, we conclude that the RP and normal mixing

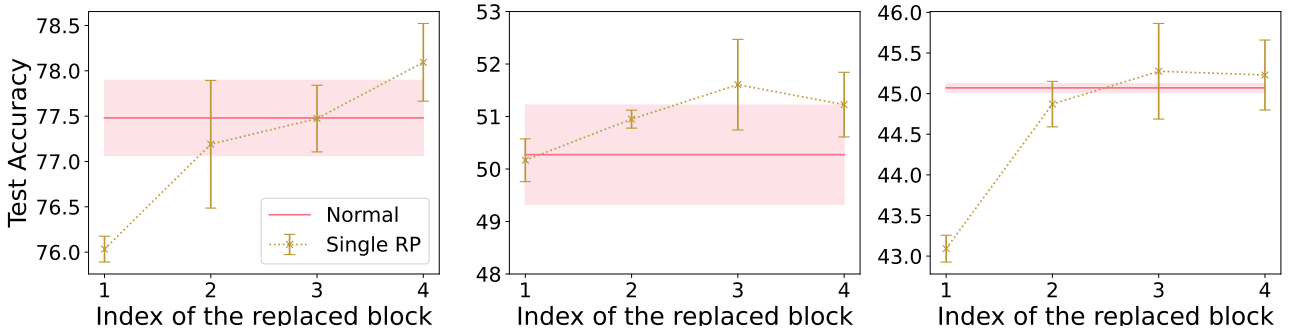

Figure S.4 . Replacing a single block of the normal Mixer with a corresponding RP block clarifies that the upstream layers are functionally commutative with the RP block. (Left) CIFAR-10, (Center) CIFAR-100, (Right) STL-10. We set $C = S = 128$.

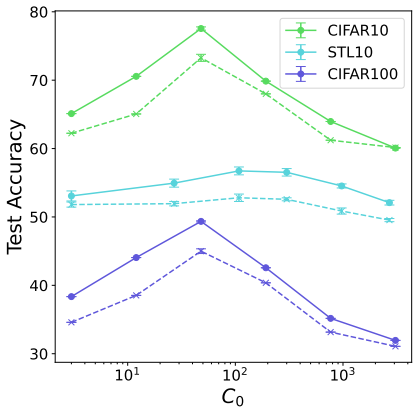

Figure S.5 . Dependence on C_0.

layers have almost the same inductive bias, especially in the upstream layers.

We utilized Tesla V100 GPUs and approximately 10 GPU hours for our experiments. Consider the S-Mixer architecture consisting of four blocks and $S = C = 64$. In this study, we trained a modified version of the S-Mixer architecture by replacing one of the four blocks with a block that incorporates random permutation. The training was conducted on CIFAR-10, CIFAR-100, and STL-10 datasets. The optimizer and training settings used in this experiment were consistent with those described in Section C.6.

D.6. Remark on input patch size

In this study, we focused on changing the size of the mixing layers and fixed the input token and channel size (S_0, C_0). In other words, the patch size P, satisfying $C_0 = 3P^2$, is fixed. While we observe that our experimental results hold regardless of the patch size, one naive question is whether there is any optimal patch size for achieving the highest accuracy. Although this is beyond the scope of this study, we show the performance depending on C_0 in Figure S.5 as a side note. The number of mixing layers is fixed at $C = S = 64$. We observed that the optimal C_0 depended on data; $C_0 = 48$ $(S_0 = 64)$ for CIFAR-10 and 100, and $C_0 = 108$ $(S_0 = 225)$ for STL-10. Note that the dimension of an input image is $S_0C_0 = 3,072$ for CIFAR datasets and $24,300$ for STL-10. It would be rational that the optimal patch size depends on the detailed information of data.

We utilized Tesla V100 GPUs and approximately 30 GPU hours for our experiments. We conducted training experiments on CIFAR-10, CIFAR-100, and STL-10 datasets using both S-Mixer and RP S-Mixer architectures with $S = C = 64$, along with four blocks. For optimization, we set the initial learning rate to be 0.1.

For CIFAR-10 and CIFAR-100, we trained the models for 200 epochs and evaluated them with different patch sizes $(P = 1, 2, 4, 8, 16, 32)$. We performed three trials with different random seeds for each setting. On the STL-10 dataset, we

resized the images to $90 \times 90 \times 3$ and trained the models for 400 epochs. We varied the patch size ($P = 1, 3, 6, 10, 18, 30$) and performed five trials with different random seeds for each setting.

D.7. RP can prevent the overfitting

In Figure S.6 , we explored several values of C fixed Ω, the normal model shows overfitting of more than twice the magnitude compared to the RP model, especially C is the largest one in the exploring range. In this case, S takes the smallest value among the explored values. This suggests that RP has a regularization effect beyond the token-mixing side and affects the channel-mixing side, particularly when C is large.

To mitigate overfitting, additional augmentation (auto-augmentation, based on (Cubuk et al., 2019)) was applied to the dataset, and the model was switched from S-Mixer to MLP-Mixer ($\gamma = 4$) due to the observed slower decrease in training loss for S-Mixer in Figure S.7 . RP S-Mixer outperformed S-Mixer in terms of test loss for $C = 173$, indicating that RP still provides overfitting prevention even with relatively strong data augmentations.

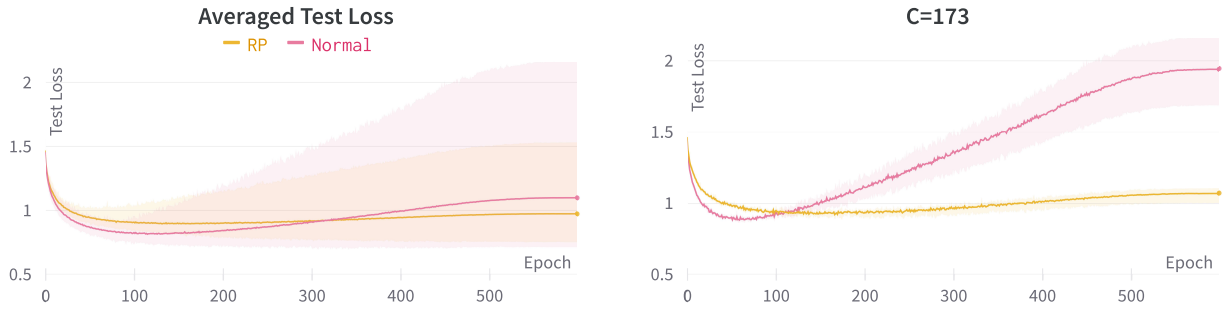

Figure S.6 . (Left) The average test loss curves are shown for $C = 16, 32, 64, 114, 173$ and five trials with different random seeds. The models used in this experiment were Normal and RP S-Mixers trained on CIFAR-10 with $L = 8$ for 600 epochs. The initial learning rate was set to 0.1. (Right) The test loss curve for $C = 173$ represents the worst case of overfitting. The shaded area in both figures represents the range between the maximum and minimum values.

Figure S.7 (right) illustrates that RP did not reach the relatively small training loss as the normal model. To address this, SGD was replaced with AdamW as the optimizer, with a reduced initial learning rate (lr $= 0.01$) due to the instability observed with lr $= 0.1$ in Figure S.8 . This resulted in reduced overfitting in the $C > S$ region, and RP performed exceptionally well compared to the normal model for $C = S = 64$. In Figure S.8 , neither the normal nor RP models exhibited a significant increase in test loss for $C = S$. However, while the normal model's test loss plateaued, the RP model continued to decrease its test loss, eventually surpassing the normal model in terms of test accuracy. This highlights the potential of RP to outperform the normal model with a choice of optimization such as AdamW.

D.8. Visualization of how random permutation breaks the structure

Observing the differences between RP-Mixer, MLP-Mixer, and SW-MLP from the perspective of weights. Figure S.9 shows three types of models where components can only be 0 or 1. In RP-Mixer, it is evident that the block structure of MLP-Mixer has been disrupted.

D.9. Computational Costs

Table S.3 shows the computational resource of SW-MLPs and RP-Mixers. In the setting for ImageNet, SW-MLP requires huge memory and runtime, whereas RP-Mixer needs 10^3 to 10^6 times less. Note that the spacial complexity of the SW-MLP (resp. the RP-Mixer) is $O(m^2) = O(S^2C^2)$ (resp. $O(C^2 + S^2)$) as $S, C \to \infty$. Therefore, the RP-Mixer is more memory-efficient and computationally undemanding than the SW-MLP.

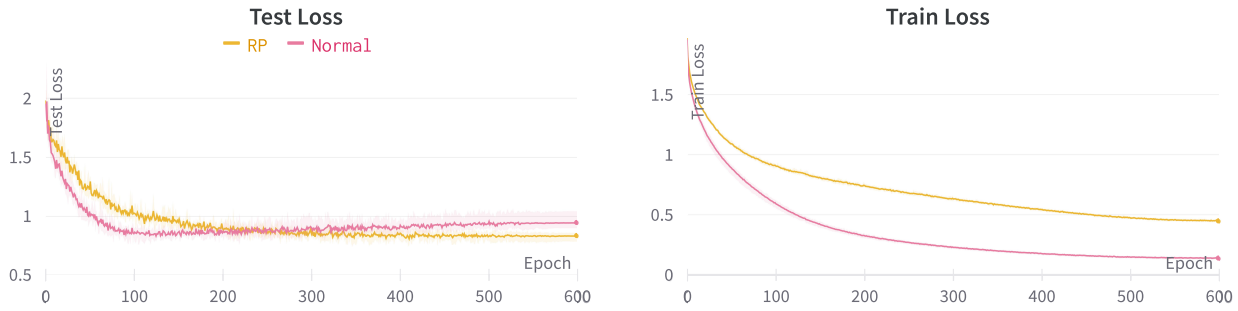

Figure S.7 . (Left) Average of test loss curves. (Right) Average train loss with $C = 173$. In both figures, the area shows max and min values. We trained normal and RP MLP-Mixer with $\gamma = 4$ with an initial learning rate of 0.1 and a mini-batch size of 256 for 600 epochs. The results are average of five trials. The shaded area in the figure represents the range of values between the maximum and minimum values.

Table S.3 . Comparison on memory requirements, FLOPs(floating point operations), and averaged runtime. For the three models, we set $S = 256, C = 588, L = 8, \gamma = 4$ for ImageNet and $S = 64, C = 48, L = 3, \gamma = 2$ for CIFAR.

Model	Memory(CIFAR/ImageNet)	FLOPs(CIFAR/ImageNet)	Runtime(CIFAR)
SW-MLP	3.22GB / 23.2TB	805M / 5.80T	$28.2(\pm 0.00)$ s/epoch
RP-Mixer	3.94MB / 26.3MB	12.5M / 4.06G	$6.41(\pm 0.01)$ s/epoch
MLP-Mixer	3.93MB / 26.3MB	12.5M / 4.06G	$6.08(\pm 0.26)$ s/epoch

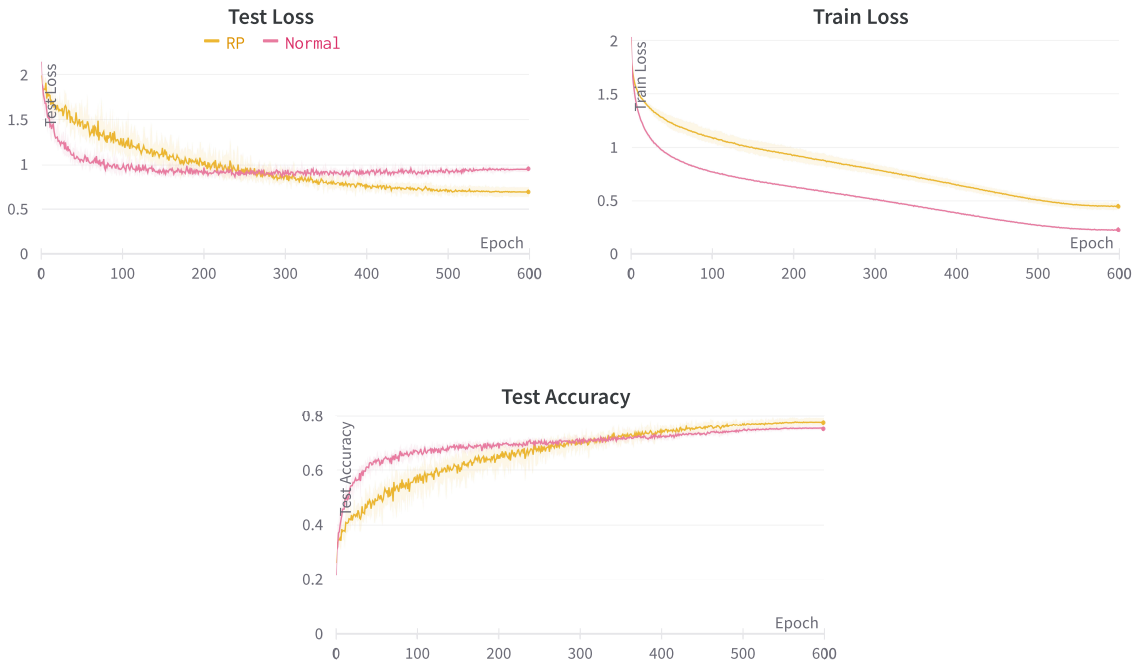

Figure S.8 . Results of training with AdamW and auto-augmentation with $S = C = 64$. The RP MLP-Mixer exceeded the results of the normal one in test loss and test accuracy. (Left) Average of test loss curves. (Right) Average train loss. (Lower) Test Accuracy curves. We set the initial learning rate to be 0.01 with a mini-batch size of 256 and 600 epochs. In all figures, the results are average of five trials and the area shows the range between the maximum and minimum values.

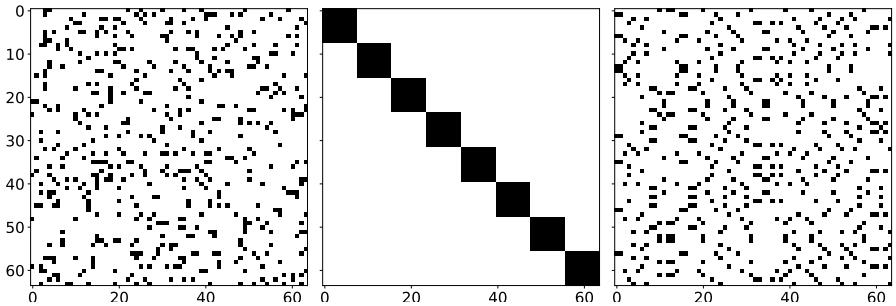

Figure S.9 . Samples of weight matrices at initialization. (Left) $M \odot A$, (center)$I_C \otimes W$, (right)$J_1(I_C \otimes W)J_2$, $C = S = 8, m = CS, p = 1/S$, and J_1 and J_2 are independent random permutation matrices. Each sample is a realization of one trial.