Analysis of Deep Neural Networks with Random Tensors

Random tensors and related topics, IHP

Tomohiro Hayase

October 15, 2024

Based on: Ryo Karakida & TH (ICML2024, arXiv:2306.01470)

Introduction

Multi-Layer Perceptron (MLP)

MLP (multilayer-perceptron) is a composition of transforms in the form of:

$$y = \phi(Wx), x \in \mathbb{R}^m$$

where W is a parameter matrix (the transforms do not share the parameter matrices).

More precisely: Let $n_0, n_1, \ldots, n_L \in \mathbb{N}$. Parameters are

$$\theta = (W_\ell, b_\ell)_{\ell=1,\ldots,L}, W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}, b_\ell \in \mathbb{R}^{n_\ell}.$$

Forward propagation: for $x \in \mathbb{R}^{n_0}$, set $x_0 = x$ and inductively

$$h_\ell = W_\ell x_{\ell-1} + b_\ell, x_\ell = \phi(h_\ell) := \phi(h_{\ell,i})_{i \in [n_\ell]}.$$

Finally, define the output by $f_\theta(x) = h_L$.

Deep Learning

Generally, a standard formulation of supervised deep learning is as follows:

1. We are given a finite set of input/output data pairs $(x, y) \in \mathcal{D}$.
2. We are given a *deep neural network* (DNN), a composition of parameterized transformations that maps a real vector to a real vector.
3. We are given an *object function*: e.g. mean squared loss:

$$\mathcal{L}(x, y, \theta) = \frac{1}{2n_L} \sum_{j=1}^{n_L} (f_\theta(x)_j - y_j)^2.$$

   We optimize DNN w.r.t. the $\mathcal{L}$ with some regularizations.
4. Evaluate trained DNN on test data separated from $\mathcal{D}$.

Random Neural Network

Random matrices appear in the initialization of neural networks:
e.g., Gaussian (Ginibre) random matrix:

$$(W_\ell)_{i,j} \sim \mathcal{N}(0, \sigma_w^2/N), \text{ i.i.d.}$$

e.g., Haar distributed orthogonal matrix:

$$W_\ell = \sigma_w O, O \sim \text{Haar Orthogonal}$$

Furthermore, many research studies random-matrix weights in deep learning theory, such as mean-field theory, edge of chaos, signal propagation, neural network Gaussian process, neural tangent kernel, and dynamical isometry (and many topics in traditional statistical mechanics of NN).
Often, large dimensional approximations are used in them.

Random Tensor

So, what role do random tensors play in deep learning?

Random Tensor

So, what role do random tensors play in deep learning?
We introduce Tolstikhin's **MLP-Mixer** as an example of neural networks related to tensor product and random tensors. (w/ large dimensional analysis)

MLP-Mixer

MLP-Mixer: A belief structure with high performance in ImageNet-1k.

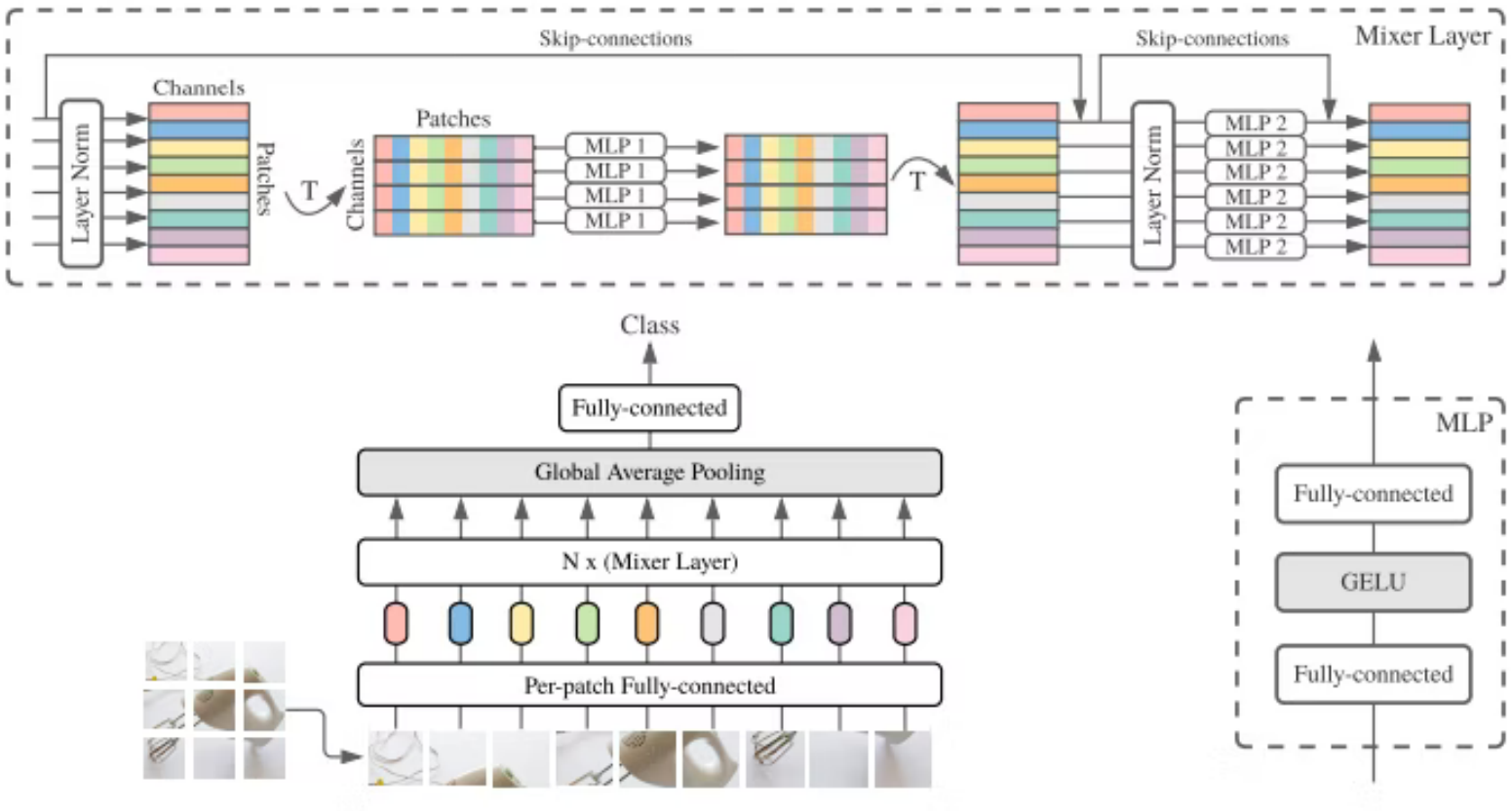
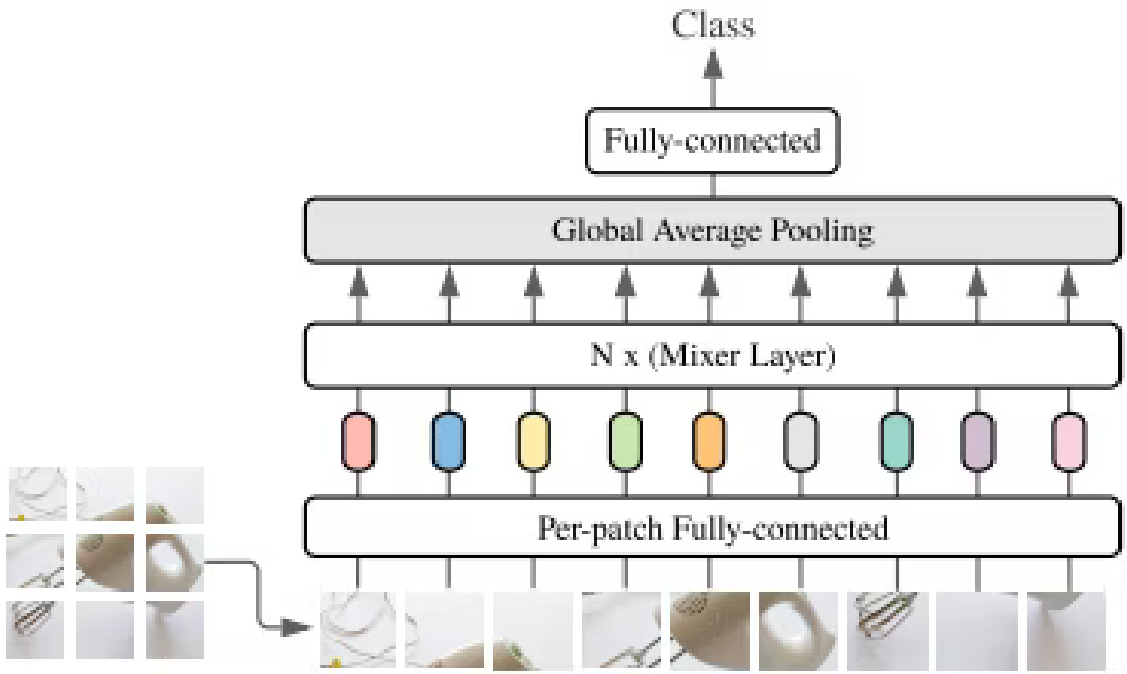


Figure: MLP-Mixer architecture [Tolstikhin+(NeurIPS 2021, arXiv:2105.01601)]

MLP-Mixer

The input of MLP-Mixer is not input RGB images (Height $\times$ Width $\times 3$ data) itself, but $S \times C$ matrix of as follows:



Here, S is the number of patches (= number of tokens), $C = 3p^2$ is the number of entries in each patch, and p is the patch size. Such patching of images is similar to Vision Transformer (ViT).

Blocks of MLP-Mixer

MLP-Mixer is a composition of

$$\text{Token-MLP}(X) = W_2\phi(W_1X),$$
$$\text{Channel-MLP}(X) = \phi(XW_3)W_4,$$

where $W_1 \in \mathbb{R}^{\gamma S \times S}$, $W_2 \in \mathbb{R}^{S \times \gamma S}$, $W_3 \in \mathbb{R}^{C \times \gamma C}$, $W_4 \in \mathbb{R}^{\gamma C \times C}$. The Token-MLP is the difference between MLP-Mixer and ViT. MLP-Mixer replaces the self-attention block of ViT with the Token-MLP. ViT's blocks:

$$\text{Attention}(X) = \sigma(Q(X)K(X)^\top)V(X),$$
$$\text{Channel-MLP}(X) = \phi(XW_3)W_4.$$

MLP-Mixer (76.44% top-1 acc, ImageNet-1k) performs similarly to ViT (79.67%).

Effective Expression of MLP-Mixer

Similarity between MLP-Mixer and MLP via vectorization

Vectorization and effective width

We represent the vectorization operation of the matrix $S \times C$ matrix X by $\mathrm{vec}(X)$:

$$(\mathrm{vec}(X))_{(j-1)d+i} = X_{ij}, (i = 1, \ldots, S, j = 1, \ldots, C).$$

There exists a well-known equation for the vectorization operation and the tensor (or Kronecker) product denoted by $\otimes$:

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

for $W \in \mathbb{R}^{S \times S}$ and $V \in \mathbb{R}^{C \times C}$.

Effective Width of Mixing Layers

The vectorization of the feature matrix WXV is equivalent to a fully connected layer of width:

$$m = SC$$

We refer to this m as the *effective width* of mixing layers.

Vectorization of Feature Matrices

Channel-Mixing layer is converted into:

$$\mathrm{vec}(X) \mapsto (I_C \otimes W)\mathrm{vec}(X)$$

Token-Mixing layer is converted into:

$$\mathrm{vec}(X) \mapsto (V^\top \otimes I_S)\mathrm{vec}(X)$$

Sparsity in Effective MLP

Typically, $S, C \sim 10^2$ to 10^3
Mixer is equivalent to an extremely wide MLP:
$m = SC = 10^4$ to 10^6
The ratio of non-zero entries in the weight matrices:

- $I_C \otimes W$: $1/C$
- $V^\top \otimes I_S$: $1/S$

Therefore, the weight of the effective MLP is highly sparse.

Commutation Matrix

A commutation matrix J_C is defined as:

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

Note that for any entry-wise function ϕ:

$$J_c \phi(x) = \phi(J_c x), x \in \mathbb{R}^m$$

Also:

$$V^\top \otimes I_S = J_c^\top (I_S \otimes V) J_c$$

Effective Expression of MLP-Mixer

Channel-MLP Block:

$$u = \phi(J_c(I_C \otimes W_2)\phi((I_C \otimes W_1)x))$$

Token-MLP Block:

$$y = \phi(J_c^\top(I_S \otimes W_4^\top)\phi((I_S \otimes W_3^\top)u))$$

MLP with static-mask

Static Mask: Consider a matrix M of entries 0 or 1 distributed and replace W in each layer of MLP by $M \odot W$:

$$y = \phi((M \odot W)x)$$

$$M \sim \text{Bernoulli}(p)$$

The mask matrix M is fixed during the training.

Hidden features and test accuracy

We use centered kernel alignment (CKA) to validate the similarity of networks:

$$\mathrm{CKA}_{\mathrm{minibatch}}(X, Y) = \frac{k^{-1}\sum_i \mathrm{HSIC}_1(X_i X_i^\top, Y_i Y_i^\top)}{\sqrt{k^{-1}\sum_i \mathrm{HSIC}_1(X_i X_i^\top, X_i X_i^\top)}\sqrt{k^{-1}\sum_i \mathrm{HSIC}_1(Y_i Y_i^\top, Y_i Y_i^\top)}}$$

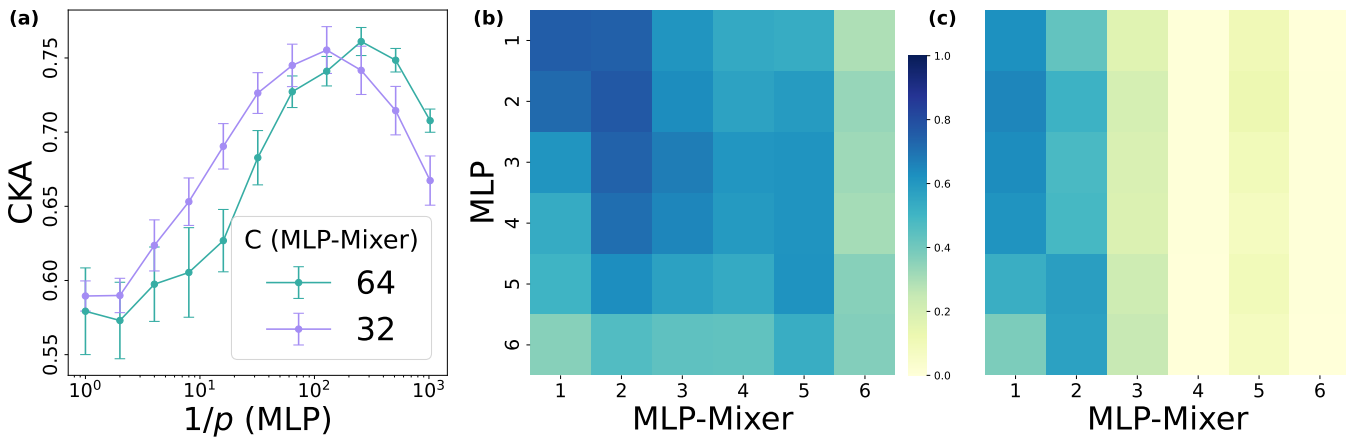


Figure: CKA results on CIFAR10

Sparseness under fixed connectoins

The following hypothesis has a fundamental role:
Hypothesis.[Goluveva+(2021)] **Increasing the width** up to a certain point while keeping the number of weight parameters fixed results in improved test accuracy.

Tendency on S and C

$$S = \frac{(\sqrt{C^2 + 8\Omega/(\gamma C)} - C)}{2}$$

$$\max_{S,C} m = (\Omega/\gamma)^{2/3}$$

The max is achieved when $C = C^*, S = S^*$ with:

$$C^* = S^* = (\Omega/\gamma)^{1/3}$$

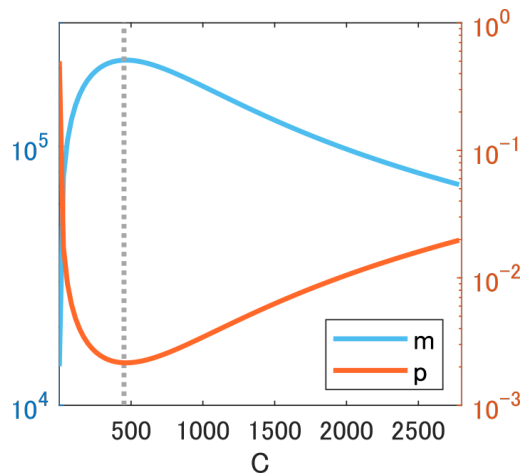


Figure: Sparseness and Width

Test Error Tendency w.r.t. Widening: Spectral Difference

Sparse Masked MLP and MLP-Mixer had a similar tendency of test error!

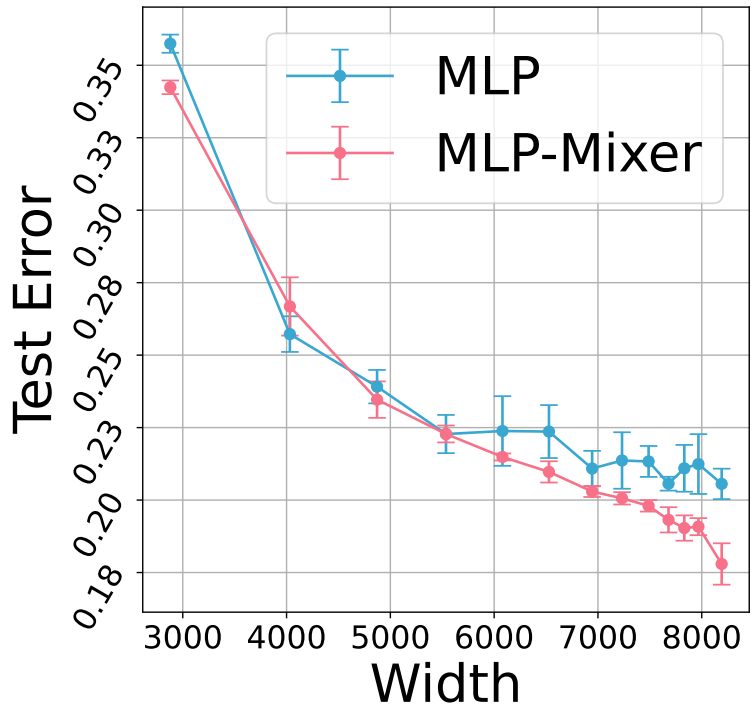


Figure: (left) Test error of MLPs with sparse weights and MLP-Mixers with different widths γm under the fixed Ω. We set $\Omega = 2^{19}$, $S = C = (\Omega/\gamma)^{1/3}$, and $\gamma = 2$. The x-axis represents the effective width γm.

Test Error Tendency w.r.t. Widening: Spectral Difference

Sparse Masked MLP and MLP-Mixer had a similar tendency of test error!

The differences in spectrum a wide limit cause the difference in training. (The s.v.d. of $J(I \otimes W)J$ converge to the MP-law.)

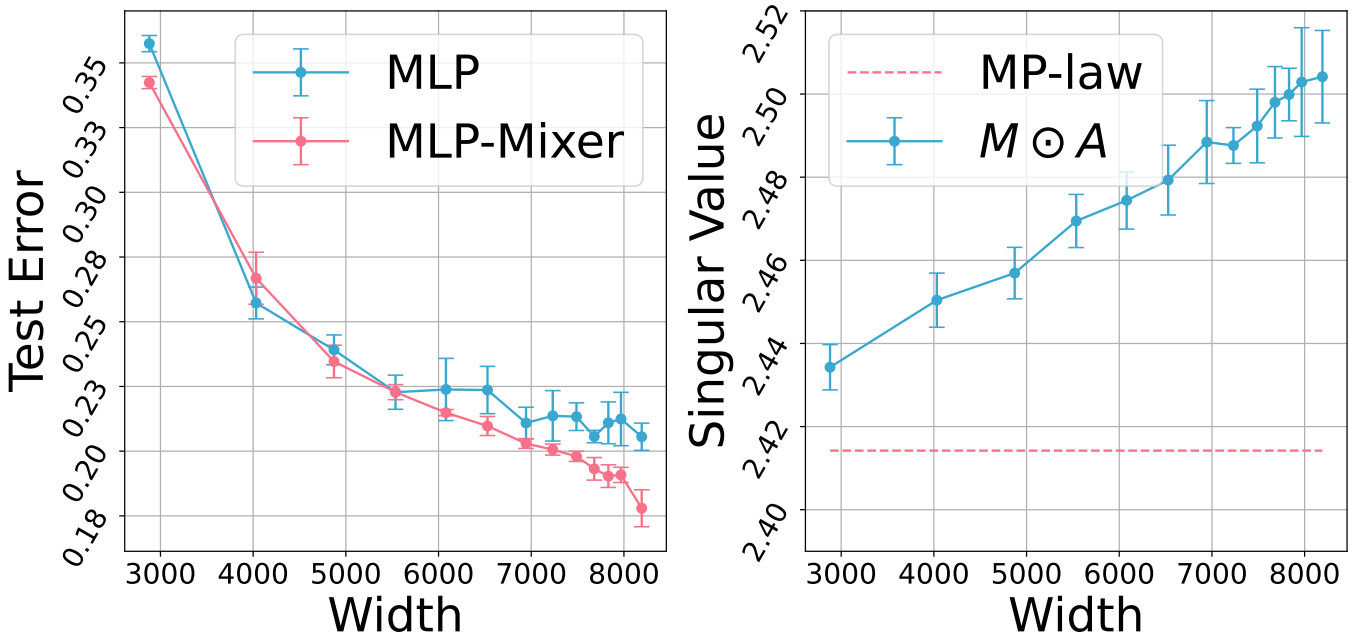


Figure: (right) The blue line indicates the averaged singular values of the weight $M \odot A$ of SW-MLP over five trials with different random seeds. The red line indicates c_γ, which is the square root of the right edge of the MP-Law.

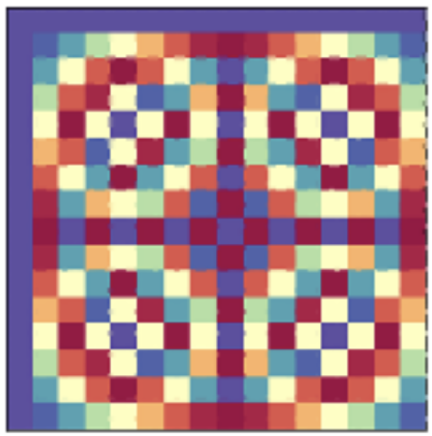
What role of J and $\otimes$ in other Models?

Q. We can present Mixer using J an $\otimes$. Are there other DNN which relies on such structured weights?
A. Monarch.

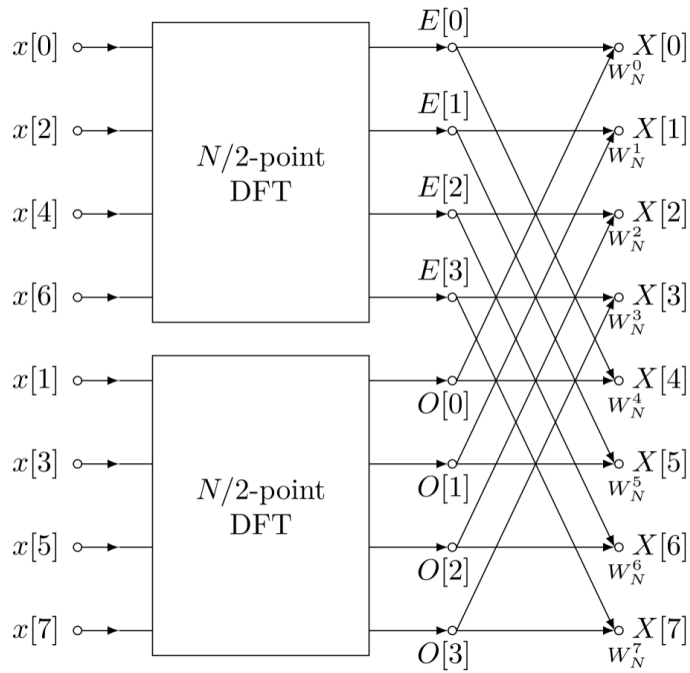
FFT

MLP-Mixer's similarity to Monarchs

Monarch Mixer: NeuRIPS2023, arXiv:310.12109 Can we design a structured weight representing a fast Fourier transform (FFT)?



Fourier Transform



Admits *O(N log N)* Algorithm

Figure: Visualization of FFT (from Dan.Fu, hazyresearch.stanford.edu/)

Butterflies

MLP-Mixer's similarity to Monarchs

FFT is decomposed into a product of butterfly matrices.

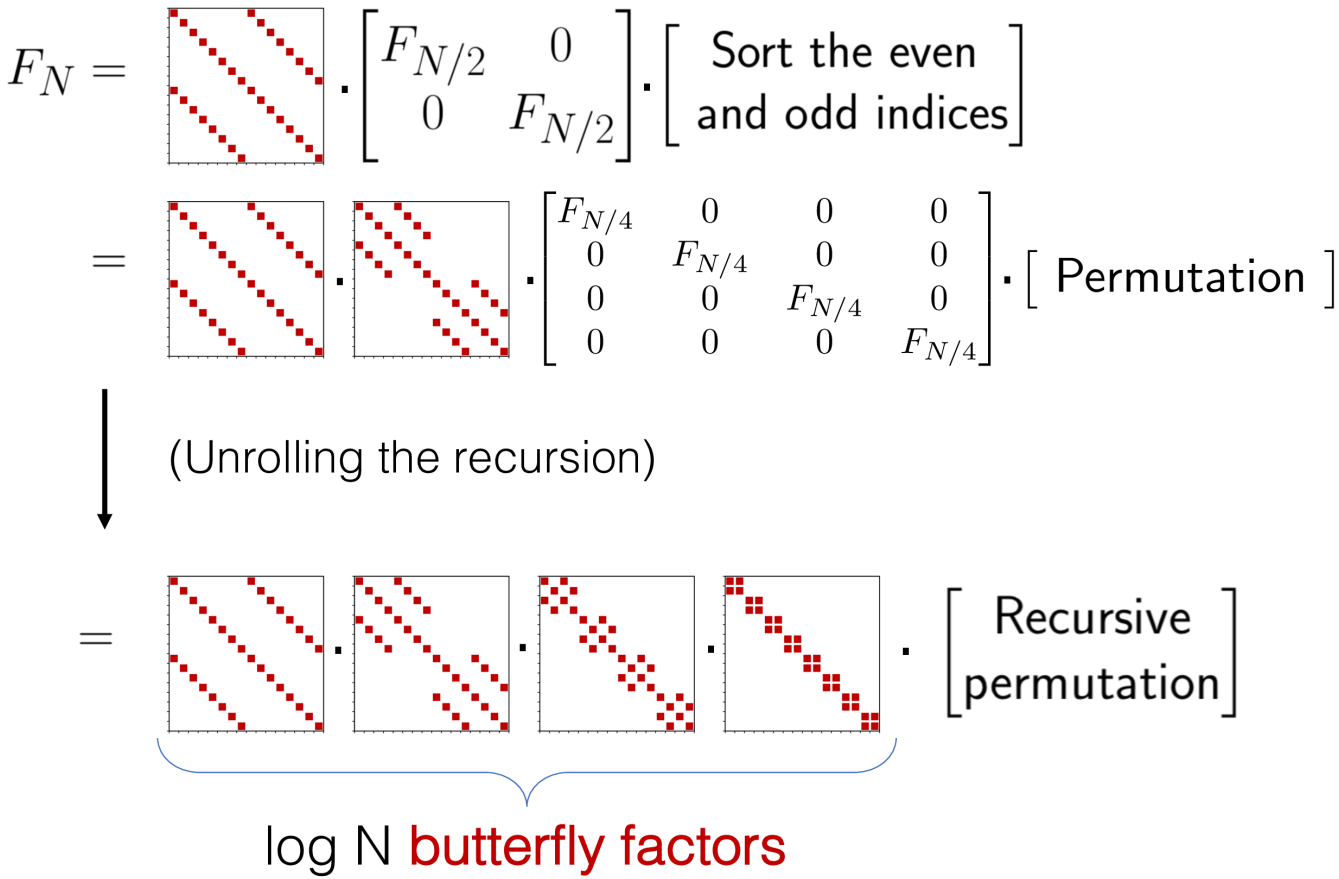


Figure: from Dan.Fu, hazyresearch.stanford.edu/

The Monarch matrix

MLP-Mixer's similarity to Monarchs

Tri. Dao+ (arXiv:2204.00595) proposed a monarch matrix:

$$M = J_c^\top L J_c R$$

where L and R are trainable block diagonal matrices, each with $\sqrt{n}$ blocks of size $\sqrt{n} \times \sqrt{n}$:

$$L = \mathrm{diag}(L_1, \ldots, L_{\sqrt{n}}),$$
$$R = \mathrm{diag}(R_1, \ldots, D_{\sqrt{n}})$$

Butterfly can be written as a Monarch matrix.
In general, Monarch matrices

- be sparse in trainable parameters
- achieves comparable performance to dense matrices
- can represent many commonly used structured matrices

Tensor Product v.s. Monarch

MLP-Mixer's similarity to Monarchs

Without activation, the Mixer's weight (Tensor/Kronecker Product) is a Monarch matrix that shares block-diagonal weights.
Mixer:

$$y = J_c(I_C \otimes W)J_c(I_S \otimes V)x.$$

In our experiments, Monarch and Kroneker (Tensor) perform similarly.

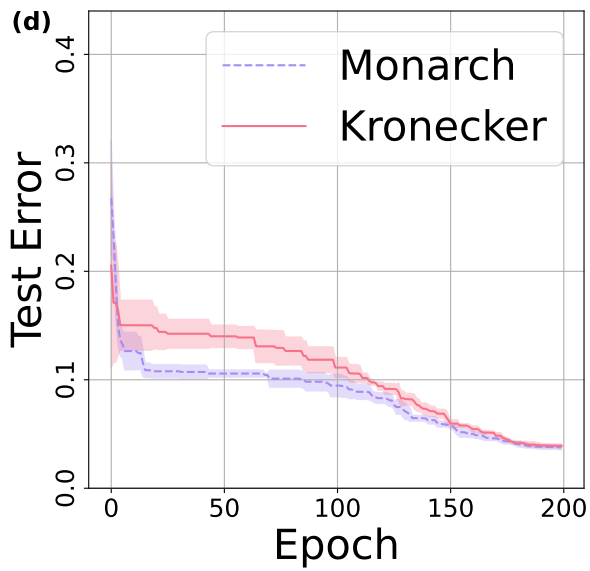


Figure: CIFAR10, 10 seeds. (from RK & TH, arXiv:23)

Large Scale Experiments: ImageNet-1k

Random Permuted Mixer

Alternative to MLP with static masks

We introduce Random-Permuted (RP) Mixer by replacing $J_c^\top (I \otimes V) J_c$ with **random permutation matrices**:

$$J_1 (I \otimes V) J_2$$

where J_1 and J_2 are independent uniformly distributed permutation matrices.

Notes:

- RP-Mixer is less structured than MLP-Mixer: RP-Mixer does not share tokens.
- RP-Mixer is more algebraically structured than MLP with random static masks: $M \odot W$

Similarity of MLP-Mixer and RP-Mixer

Under a fixed number of connections, MLP-Mixer and RP-Mixer shared the similar tendency on C/S: they had minimal test error around $S/C = 1$, that is, in maximal width.

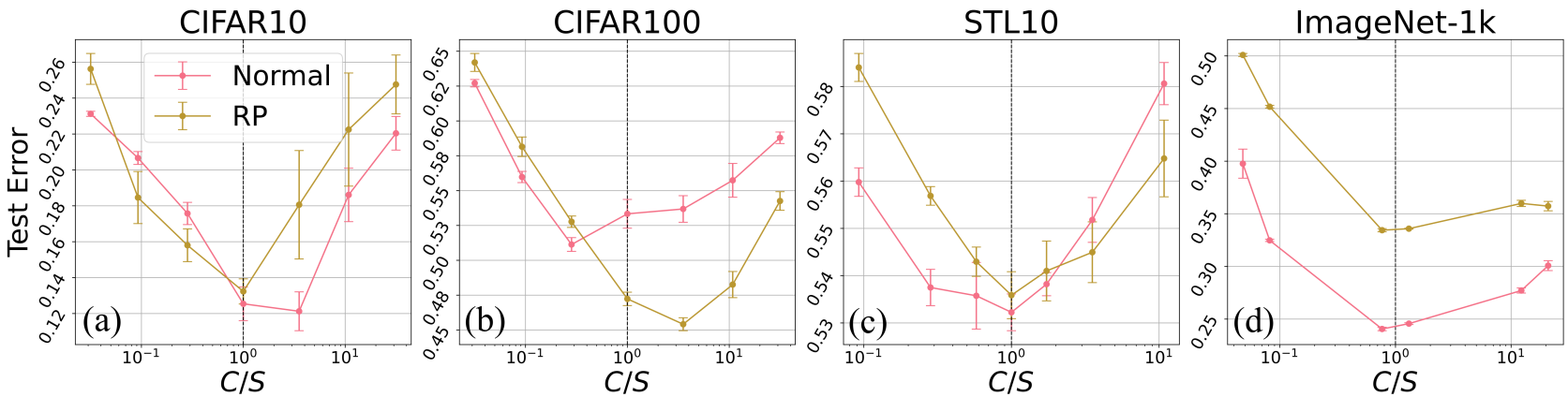


Figure: Comparison of MLP-Mixer and RP-Mixer

Application to HPS: Increase in width with a fixed number of connections

Hypothesis (Golubeva et al. 2021): Increasing the width while maintaining a fixed number of weight parameters improves test accuracy.

The average number of non-zero entries per layer:

$$\Omega = \frac{\gamma(CS^2 + SC^2)}{2}$$

MLP	CIFAR-10	CIFAR-100	max. width	#connections
β-LASSO	85.19	59.56	-	256M
Mixer-SS/8	84.09 ($\pm$1.55)	55.76($\pm$1.83)	6.3×10^4	256M
Ours	**87.93** ($\pm$0.47)	**61.87**($\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]	76.44	6.0×10^5	2.6×10^8	196	786
Ours	**76.74** ($\pm$ 0.19)	$\mathbf{6.2 \times 10^5}$	2.6×10^8	256	588

Figure: Test accuracy improvement by widening layers

Summary

Summary and Future Work

- Widening gives us a theoretical understanding of DNN.
- Widening when fixing the number of connections gives us practical knowledge of MLP-Mixer: MLP-Mixer behaves as an extremely wide and sparse MLP.
- J and $\otimes$ also appear in other DNN (Monarch)

Future Work:

- When does freeness have a role in practical deep neural networks?
- Other random but hierarchical structured features/network out of Monarch or Random Permutation $+$ $\otimes$?
- (WIP) Replacing $\otimes$ with $*$ (**free product**) has a meaningful effect in ML?