Scaling-up Diverse Orthogonal Convolutional Networks by A Paraunitary Framework

Jiahao Su[★], Wonmin Byeon[†], Furong Huang[★]
jiahaosu@umd.edu, wbyeon@nvidia.com, furongh@umd.edu
[★]University of Maryland,
College Park, Maryland, USA
[†]Nvidia Research, Nvidia Corporation,
Santa Clara, California, USA

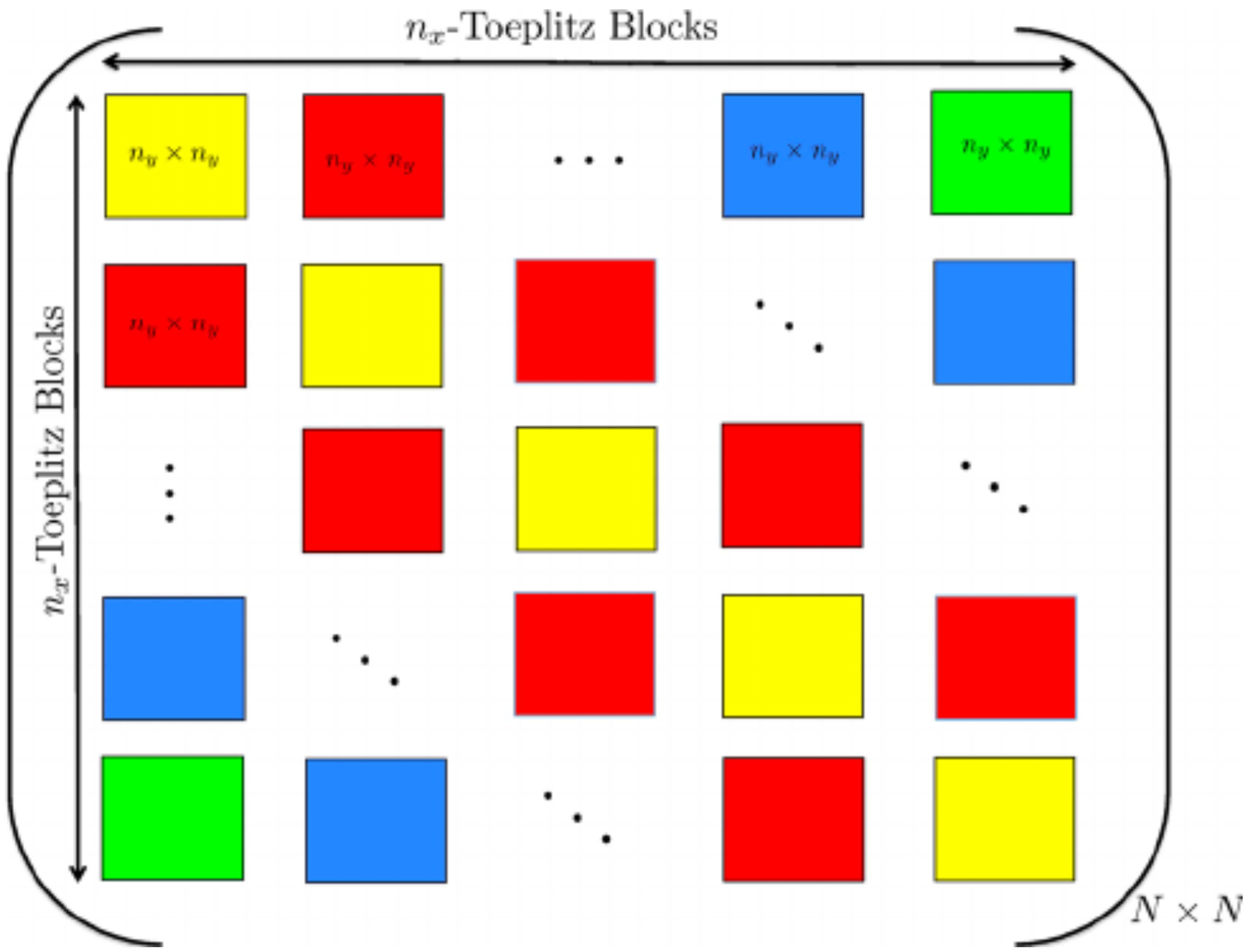
arXiv: https://arxiv.org/abs/2106.09121
Code: https://github.com/umd-huang-lab/ortho-conv

Orthogonal Convolutional Networks

Lipschitz Continuity and Gradient Stability

- Motivations:
  - **Lipschitz continuity** — adversarial robustness.
  - **Gradient stability** — well-conditioned optimization.

- $W^\top W = I \Rightarrow \|y\|_\mathsf{F}^2 = \|Wx\|_\mathsf{F}^2 = x^\top W^\top W x = x^\top x = \|x\|_\mathsf{F}^2$
  - Fully-connected layer: W is a general matrix.
  - Convolutional layer: W is a block-Toeplitz matrix.
  - The layer is orthogonal iff. the matrix is orthogonal.



Block Toeplitz matrix

Orthogonal Convolutional Networks

Challenges in Learning Orthogonal Convolutions

- **Challenge 1: How to guarantee that the spectrum of block Toeplitz matrix is flat?**
  - **Incorrect:** Compute the SVD of the naively flattened convolutional kernel.
  - **Expensive:** Compute the SVD of all frequencies of the Fourier transform.

- **Challenge 2: How to maintain the orthogonal constraint during training?**
  - **Inexact:** Soft clipping through regularization.
  - **Expensive:** projected gradient descent.

- **Solution: Parameterize the convolutional layer as an orthogonal filter bank.**

Convolutional Layer as MIMO Filter Bank

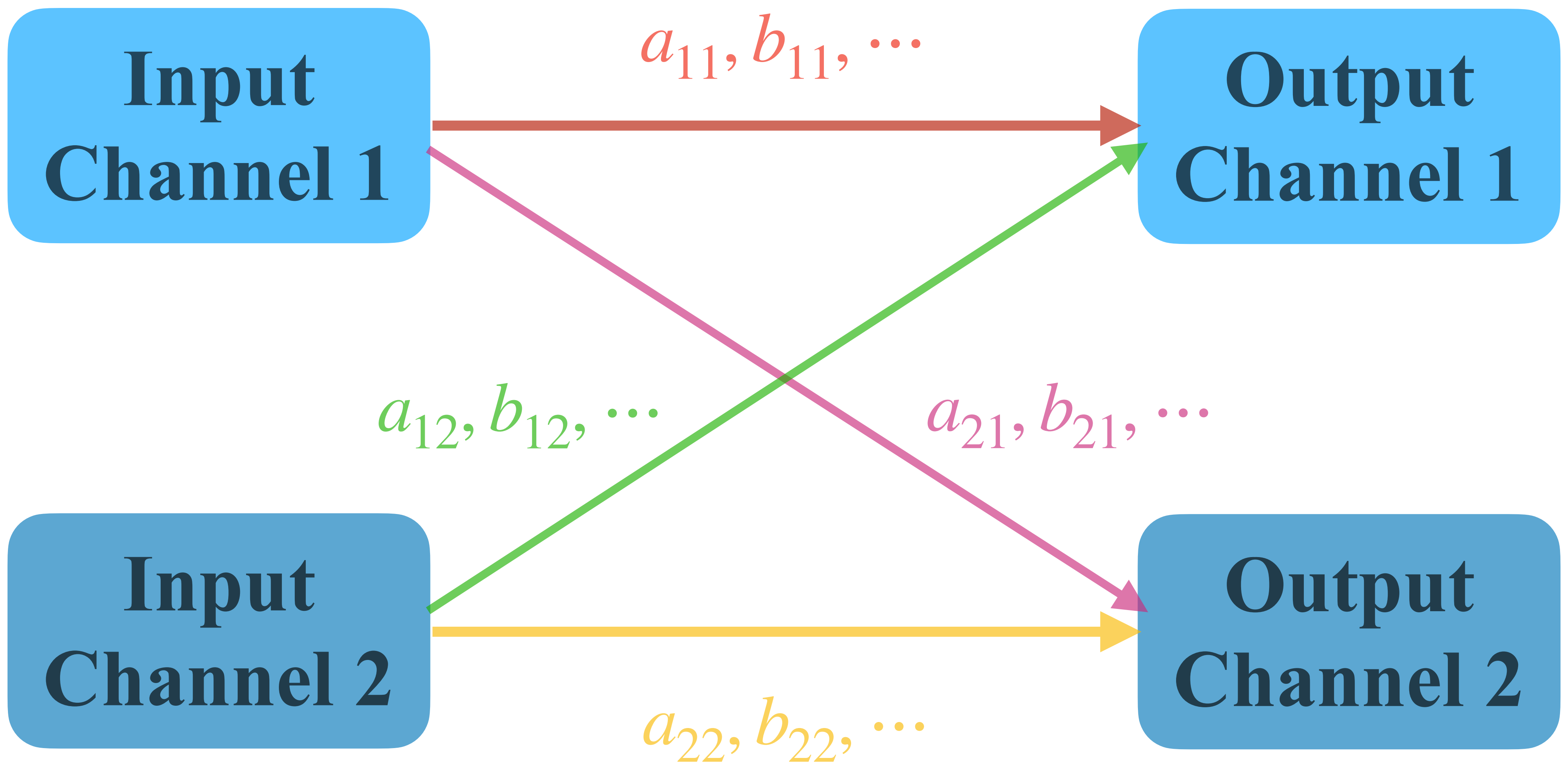
Property Characterization via Transfer Matrix

- Standard convolutional layer:
  - $\mathcal{Y}_{:,t} = \sum_s \mathcal{W}_{:,t,s} * \mathcal{X}_{:,s}$

    ↓ **Convolution theorem**
  - $Y_t(z) = \sum_s W_{ts}(z) X_s(z)$

    ↓ **Matrix multiplication**
  - $\mathbf{Y}(z) = \mathbf{W}(z)\mathbf{X}(z)$

- The **transfer matrix** $\mathbf{W}(z)$ characterizes the properties of a convolutional layer.



A two-input-two-output convolutional layer

$$\mathbf{W}(z) = \underbrace{\begin{bmatrix} a_{11} a_{12} \\ a_{21} a_{22} \end{bmatrix}}_{\mathbf{A}} + \underbrace{\begin{bmatrix} b_{11} b_{12} \\ b_{21} b_{22} \end{bmatrix}}_{\mathbf{B}} z^{-1} + \cdots$$

$$= \mathbf{A} + \mathbf{B} z^{-1}$$

Orthogonal Convolutional Layer as Paraunitary System

From Paraunitary System to Orthogonal Matrices

- **Challenge 1: How to guarantee that the block-Toeplitz matrix is orthogonal?**

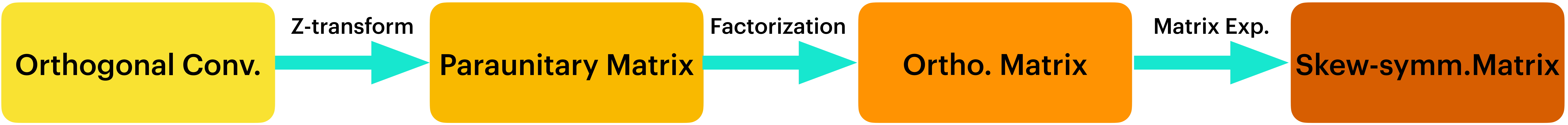
- **Solution: Constrain the transfer matrix to be <u>paraunitary</u>.**
  - A filter bank is orthogonal iff. $\mathbf{W}(z)$ is paraunitary: $\mathbf{W}^{\dagger}(e^{j\omega})\mathbf{W}(e^{j\omega}) = \mathbf{I}, \forall \omega$.
  - A paraunitary matrix is factorized as $\mathbf{W}(z) = \mathbf{U}[\mathbf{P}_1 + (\mathbf{I} - \mathbf{P}_1)z^{-1}]\cdots[\mathbf{P}_N + (\mathbf{I} - \mathbf{P}_N)z^{-1}]$, $\mathbf{U}$ is orthogonal and $\mathbf{P}_n$ is a projection matrix ($\mathbf{P}_n = \mathbf{V}_n\mathbf{V}_n^{\top}$ and $\mathbf{V}_n$ is column-orthogonal).

Constrained Optimization over Matrix Manifolds

From Orthogonal Matrices to Unconstrained Parameters

- **Challenge 2: How to maintain the orthogonal constraint during training?**

- **Solution: Parametrize the orthogonal matrices using matrix exponential.**
  - $\mathbf{U} = \exp(\mathbf{S})$, where $\exp(\mathbf{S}) = \sum_{k=0}^{\infty} \mathbf{S}^k / k!$ and S is a skew-symmetric matrix.
  - The skew-symmetric matrix is characterized up its upper-triangle entries.



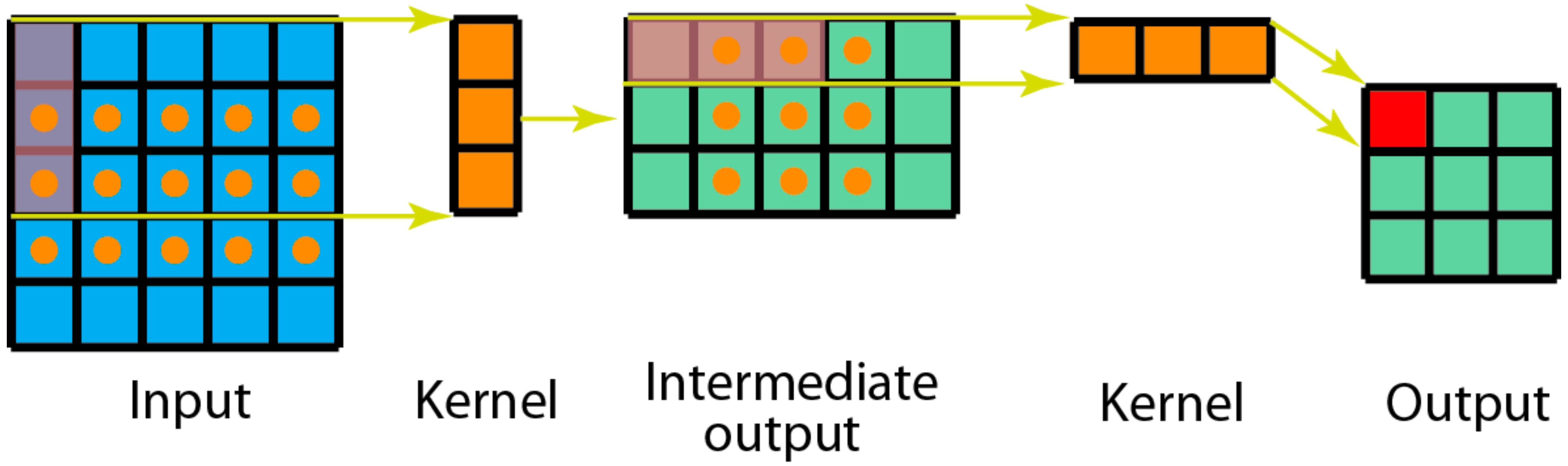
Diverse Orthogonal Convolutions

Convolution Variants and Multi-dimensional Extensions

- Convolution variants:
  - The design of these variants reduce to paraunitary systems.
  - We can use the same parameterization (factorization and matrix exponential) to construct all these variants.
- Multi-dimensional extensions:
  - Construct a (separable) MD orthogonal convolution by a number of 1D ones.
  - 2D case: $\mathbf{H}(z_1, z_2) = \mathbf{H}_1(z_1)\mathbf{H}_2(z_2)$
  - Also support convolution variants.

Table 1: **Variants of convolutions.** We present the modified Z-transforms, $\underline{\boldsymbol{Y}}(z)$, $\underline{\boldsymbol{H}}(z)$, and $\underline{\boldsymbol{X}}(z)$ for each convolution such that $\underline{\boldsymbol{Y}}(z) = \underline{\boldsymbol{H}}(z)\underline{\boldsymbol{X}}(z)$ holds. In the table, $\boldsymbol{X}^{[R]}(z) \triangleq [\boldsymbol{X}^{0|R}(z)^\top, \dots, \boldsymbol{X}^{R-1|R}(z)^\top]^\top$ and $\widetilde{\boldsymbol{X}}^{[R]}(z) = [\boldsymbol{X}^{-0|R}(z), \dots, \boldsymbol{X}^{-(R-1)|R}(z)]$. For group convolution, $\boldsymbol{h}^g$ is the filter for the g^{th} group with $\boldsymbol{H}^g(z)$ being its Z-transform, and blkdiag $(\cdot)$ stacks multiple matrices into a block-diagonal matrix.

Convolution Type	Spatial Representation	Spectral Representation $\underline{\boldsymbol{Y}}(z)$	Spectral Representation $\underline{\boldsymbol{H}}(z)$	Spectral Representation $\underline{\boldsymbol{X}}(z)$
Standard	$\boldsymbol{y}[i] = \sum_{n \in \mathbb{Z}} \boldsymbol{h}[n]\boldsymbol{x}[i-n]$	$\boldsymbol{Y}(z)$	$\boldsymbol{H}(z)$	$\boldsymbol{X}(z)$
R-Dilated	$\boldsymbol{y}[i] = \sum_{n \in \mathbb{Z}} \boldsymbol{h}^{\uparrow R}[n]\boldsymbol{x}[i-n]$	$\boldsymbol{Y}(z)$	$\boldsymbol{H}(z^R)$	$\boldsymbol{X}(z)$
$\downarrow R$-Strided	$\boldsymbol{y}[i] = \sum_{n \in \mathbb{Z}} \boldsymbol{h}[n]\boldsymbol{x}[Ri-n]$	$\boldsymbol{Y}(z)$	$\widetilde{\boldsymbol{H}}^{[R]}(z)$	$\boldsymbol{X}^{[R]}(z)$
$\uparrow R$-Strided	$\boldsymbol{y}[i] = \sum_{n \in \mathbb{Z}} \boldsymbol{h}[n]\boldsymbol{x}^{\uparrow R}[i-n]$	$\boldsymbol{Y}^{[R]}(z)$	$\boldsymbol{H}^{[R]}(z)$	$\boldsymbol{X}(z)$
G-Group	$\boldsymbol{y}[i] = \sum_{n \in \mathbb{Z}} \text{blkdiag}(\{\boldsymbol{h}^g[n]\})\boldsymbol{x}[i-n]$	$\boldsymbol{Y}(z)$	$\text{blkdiag}(\{\boldsymbol{H}^g(z)\})$	$\boldsymbol{X}(z)$



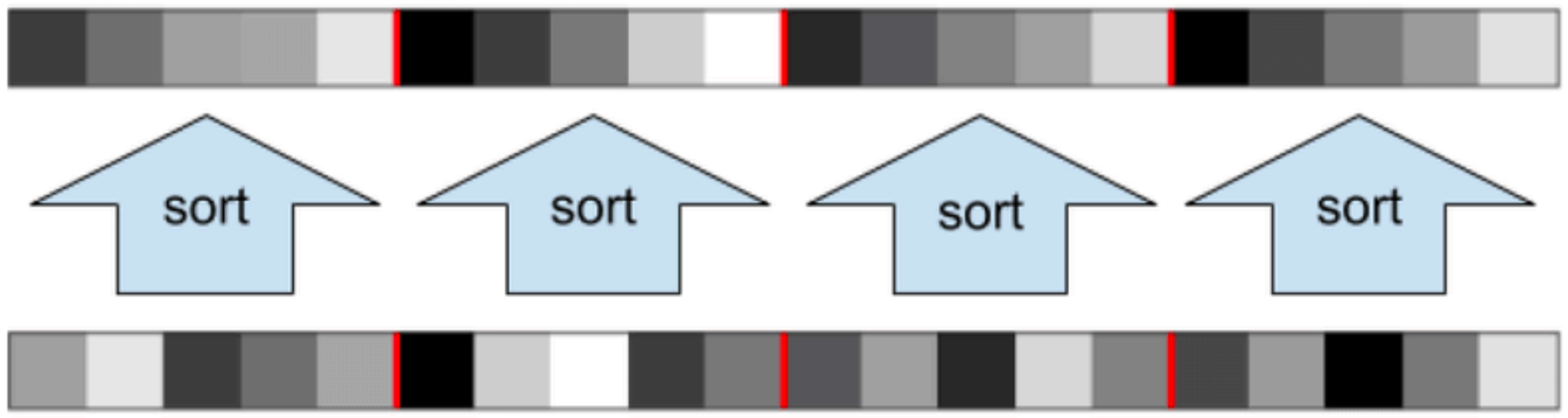
A separable 2D convolution as two 1D convolutions

Lipschitz Networks

Robustness Against Adversarial Attacks

- Lipschitz networks for robustness.
  - **Linear layer:** Orthogonal convolution
  - **Activation layer:** GroupSort activation
  - **(Optional) Shortcut:** Scaled addition

$$y = \alpha x + (1 - \alpha)g(x),\ 0 < \alpha < 1$$



GroupSort Activation

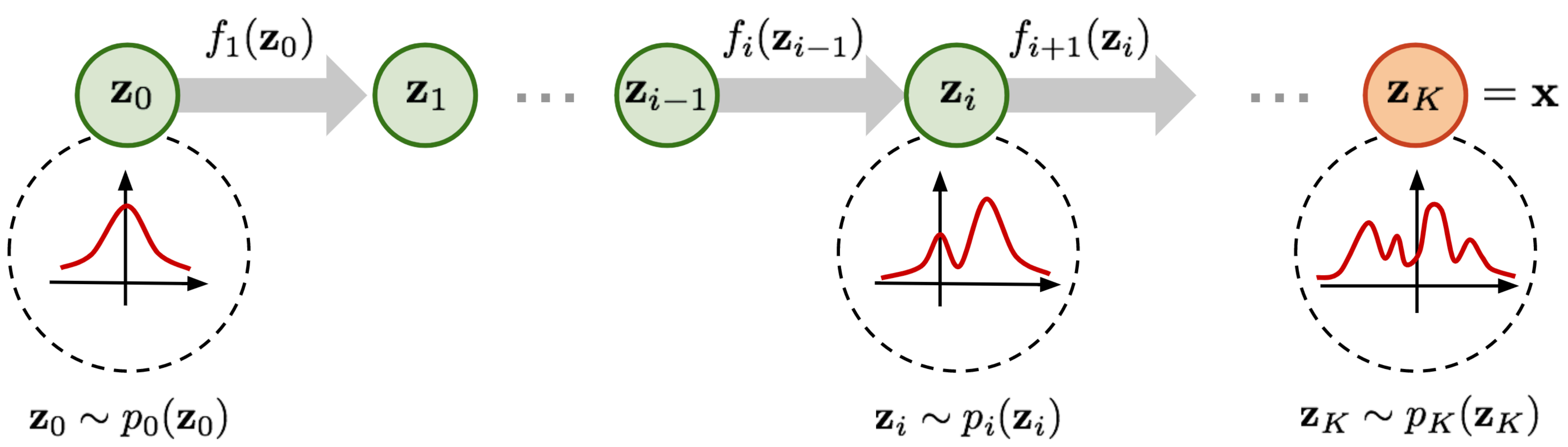
	22 layers									
Width	1	3	6	8	10	1	3	6	8	10
	Clean (%)					PGD with $\epsilon = 36/255$ (%)				
Ours	79.90	82.22	87.21	**88.10**	87.82	67.95	70.88	74.30	75.12	**76.46**
Cayley	79.11	84.82	85.85	-	-	69.79	65.61	74.81	-	-
RKO	82.71	84.19	84.33	84.55	-	72.40	74.36	75.66	76.41	-

	34 layers									
Width	1	3	6	8	10	1	3	6	8	10
	Clean (%)					PGD with $\epsilon = 36/255$ (%)				
Ours	81.24	88.17	**88.92**	-	-	69.21	71.85	**75.09**	-	-
Cayley	82.46	84.29	-	-	-	71.27	74.73	-	-	-
RKO	81.51	83.24	83.92	-	-	71.38	73.84	75.03	-	-

Flow-based Generative Models

Image Generation by Residual Flows

- Flow-based generative model:
  - Training: From data x to latent z
    $x \xrightarrow{f_n^{-1}} h_{n-1} \xrightarrow{f_{n-1}^{-1}} h_{n-2} \xrightarrow{f_{n-2}^{-1}} \cdots \xrightarrow{f_2^{-1}} h_1 \xrightarrow{f_1^{-1}} z$
  - Generation: From Gaussian z to data x
    $z \xrightarrow{f_1} h_1 \xrightarrow{f_2} h_2 \xrightarrow{f_3} \cdots \xrightarrow{f_{n-1}} h_{n-1} \xrightarrow{f_n} x$
- Invertible residual network (i-ResNet)
  - $y = f(x) = x + g(x)$
  - f is invertible if g is Lipschitz.
  - We construct g using Lipschitz network.



Flow-based Generative Model
https://lilianweng.github.io/posts/2018-10-13-flow-models/

Model	MNIST
Glow (Kingma & Dhariwal, 2018)	1.05
FFJORD (Grathwohl et al., 2018)	0.99
i-ResNet (Behrmann et al., 2019)	1.05
Residual Flow (Chen et al., 2019)	0.97
SC-Fac Residual Flow (Ours)	0.896

Bits per dimension (bpd) for MNIST dataset.

Thanks for watching this video!

- arXiv: https://arxiv.org/abs/2106.09121
- Code link: https://github.com/umd-huang-lab/ortho-conv

- Citation:

```
@article{DBLP:journals/corr/abs-2106-09121,
  author  = {Jiahao Su, Wonmin Byeon, and Furong Huang},
  title   = {Scaling-up Diverse Orthogonal Convolutional Networks
             with a Paraunitary Framework},
  journal = {CoRR},
  volume = {abs/2106.09121},
  year    = {2021}
}
```