THESIS DEFENSE

FInC Flow: Fast and Invertible k × k Convolutions for Normalizing Flows

Aditya V Kallappa
2019702012
aditya.kallappa@research.iiit.ac.in

International Institute of Information Technology, Hyderabad, India

Outline

- Prerequisites – Generative AI
- Normalizing Flows
- Convolution – A Linear Transformation
- Inverse of convolution
- Inverse Techniques
- FInC Flow – Our Work
- GPU implementation

Generative AI

- Generative AI refers to a class of artificial intelligence algorithms that have the capability to generate new content, such as text, images, or other forms of data, based on patterns and information present in the training data

Generative AI

- Generative AI refers to a class of artificial intelligence algorithms that have the capability to generate new content, such as text, images, or other forms of data, based on patterns and information present in the training data
- Text: GPT-3, Chat-GPT, PaLM, Llama etc.

Generative AI

- Generative AI refers to a class of artificial intelligence algorithms that have the capability to generate new content, such as text, images, or other forms of data, based on patterns and information present in the training data
- Text: GPT-3, Chat-GPT, PaLM, Llama etc.
- Images: GAN, StyleGAN, DeepCream

Generative AI

- Generative AI refers to a class of artificial intelligence algorithms that have the capability to generate new content, such as text, images, or other forms of data, based on patterns and information present in the training data
- Text: GPT-3, Chat-GPT, PaLM, Llama etc.
- Images: GAN, StyleGAN, DeepCream
- Art: DALL-E, Aiva
- Audio: GANs, Normalizing Flows, VAE models

Generative AI

Involves around 4 types of generative models:

1. GAN(Generative Adversarial Networks): The generator creates synthetic data, while the discriminator evaluates whether the generated data is real or fake.
2. VAE(Variational Auto Encoders): Consists of an Encoder which compresses the data and a Decoder which tries to get the original data from a compressed encoded data
3. Normalizing Flows: Generates data by simply transforming a Normal Distribution by a series of invertible and differentiable functions
4. Diffusion Models: These models transform a simple and known distribution (e.g., Gaussian noise) into a complex and high-dimensional distribution that matches the characteristics of the target data.

Why Normalizing Flows

1. Use of bijective and differentiable allows models for easy generation and manipulation of data
2. Exact Likelihood: Unlike all the other models, we can actually compute the probability distribution of the data
3. Flexible and Explicit Density Modeling
4. Versatile data generation

Why Normalizing Flows

1. Use of bijective and differentiable allows models for easy generation and manipulation of data
2. Exact Likelihood: Unlike all the other models, we can actually compute the probability distribution of the data
3. Flexible and Explicit Density Modeling
4. Versatile data generation

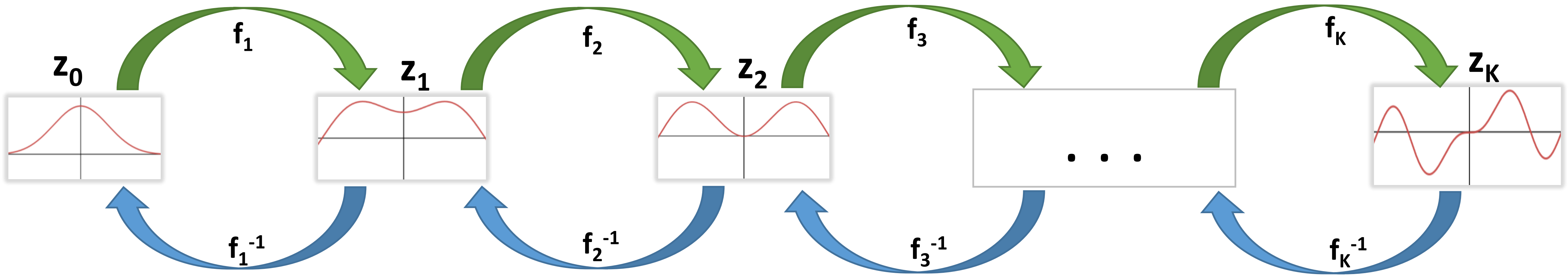
But the generation of samples can be extremely slow

Our Contributions

We develop a system/model which

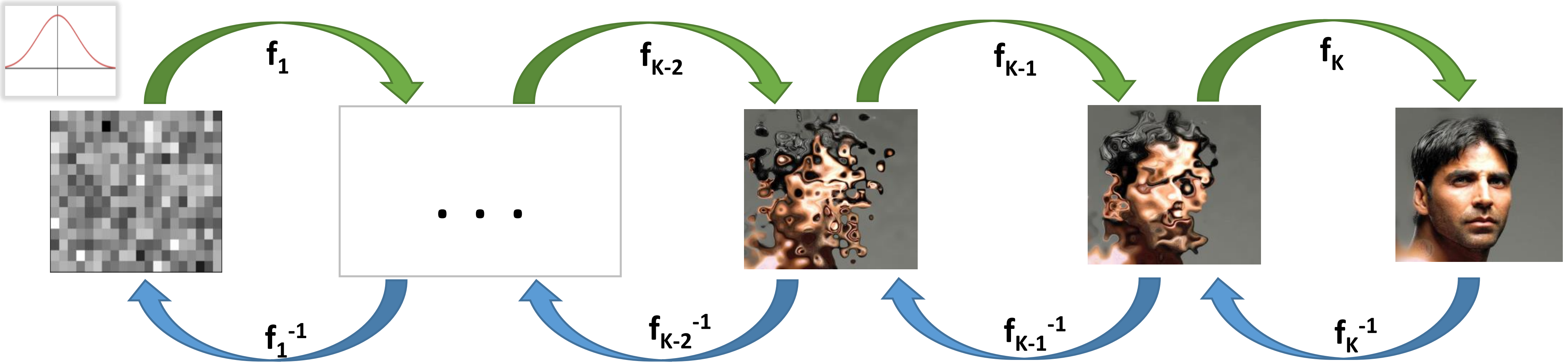
1. has a fast parallel inversion algorithm with running time $O(nk^2)$ (n is height and width of the input image and k is kernel size)
2. masks the minimal amount of learnable parameters in a layer
3. gives better sampling time comparable to other $k \times k$ convolution-based models on real-world benchmarks.

Normalizing Flows

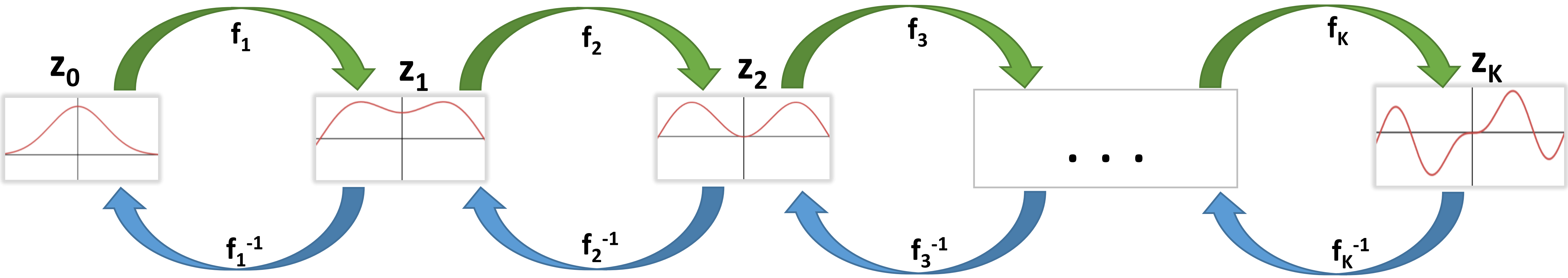


- f_i's are both differentiable and invertible

Normalizing Flows



Normalizing Flows



- f_i's are both differentiable and invertible

Normalizing Flows NLL Derivation

Let $z_0 \in \mathbb{R}^D$ be a multivariate R.V. with density $p_0(z_0)$

Normalizing Flows NLL Derivation

Let $z_0 \in \mathbb{R}^D$ be a multivariate R.V. with density $p_0(z_0)$

For $i = 1, 2, \ldots, K$, let $z_i = f_i(z_{i-1})$ be a sequence of transformations of z_0

Normalizing Flows NLL Derivation

Let $z_0 \in \mathbb{R}^D$ be a multivariate R.V. with density $p_0(z_0)$

For $i = 1, 2, \dots, K$, let $z_i = f_i(z_{i-1})$ be a sequence of transformations of z_0

Then log likelyhood of z_K is

$$\log p_K(z_K) = \log p_0(z_0) - \sum_{i=1}^{K} \log \left| \det \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right|$$

Normalizing Flows NLL Derivation

Let $z_0 \in \mathbb{R}^D$ be a multivariate R.V. with density $p_0(z_0)$

For $i = 1, 2, \dots, K$, let $z_i = f_i(z_{i-1})$ be a sequence of transformations of z_0

Then log likelyhood of z_K is

$$\log p_K(z_K) = \log p_0(z_0) - \sum_{i=1}^{K} \log \left| \det \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right|$$

Derivation:

$$z_1 = f_1(z_0) \; ; \; z_0 = f_1^{-1}(z_1)$$

$$p_1(z_1) = p_0(z_0) \left| \det \frac{\partial f_1^{-1}(z_1)}{\partial z_1} \right|$$

$$p_1(z_1) = p_0(z_0) \left| \det \left(\frac{\partial f_1(z_0)}{\partial z_0} \right)^{-1} \right|$$

By the identity, $\det(A^{-1}) = (\det A)^{-1}$

$$p_1(z_1) = p_0(z_0) \left| \det \frac{\partial f_1(z_0)}{\partial z_0} \right|^{-1}$$

Normalizing Flows NLL Derivation

By the identity, $\det(A^{-1}) = (\det A)^{-1}$

$$p_1(z_1) = p_0(z_0) \left| \det \frac{\partial f_1(z_0)}{\partial z_0} \right|^{-1}$$

$$\log p_1(z_1) = \log p_0(z_0) - \log \left| \det \frac{\partial f_1(z_0)}{\partial z_0} \right|$$

The above equation applies to any z_i, z_{i-1}

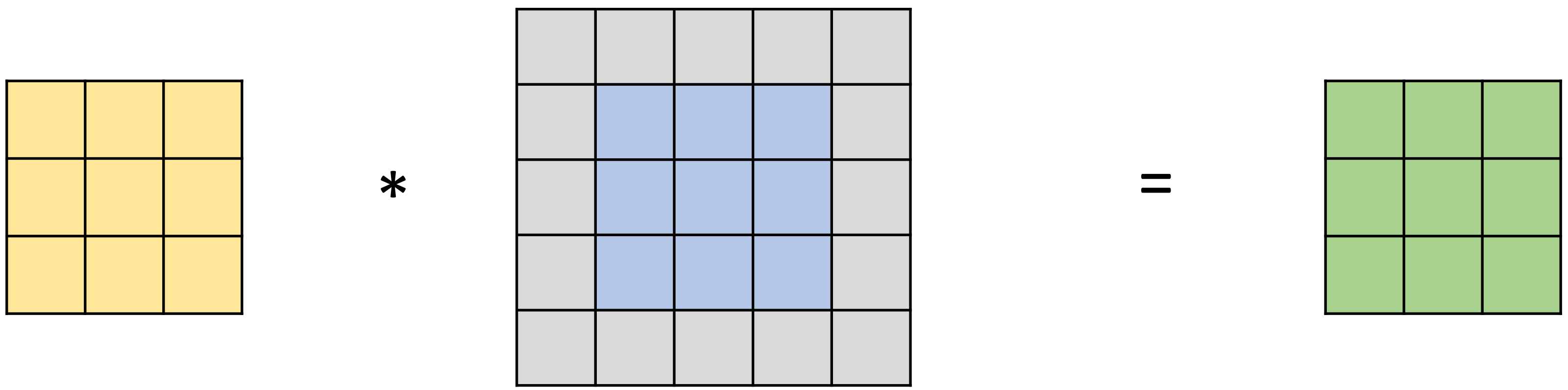
$$\log p_K(z_K) = \log p_{K-1}(z_{K-1}) - \log \left| \det \frac{\partial f_K(z_{K-1})}{\partial z_{K-1}} \right|$$

But $\log p_{K-1}(z_{K-1}) = \log p_{K-2}(z_{K-2}) - \log \left| \det \frac{\partial f_{K-1}(z_{K-2})}{\partial z_{K-2}} \right|$

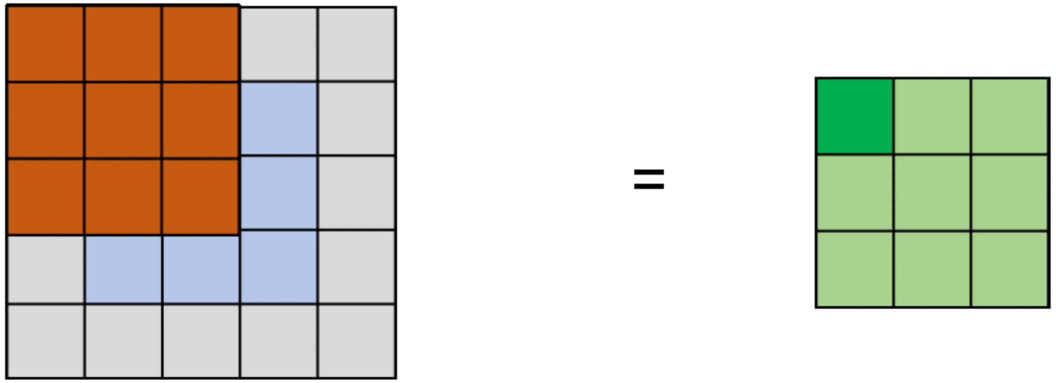
$$\vdots$$

$$\log p_K(z_K) = \log p_0(z_0) - \sum_{i=1}^{K} \log \left| \det \frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right|$$

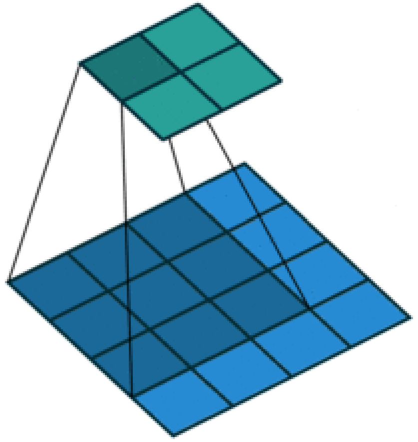
Convolution



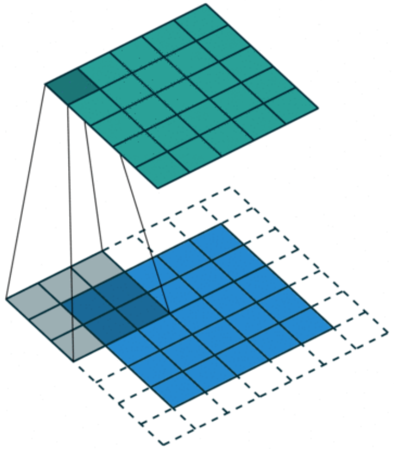
Convolution



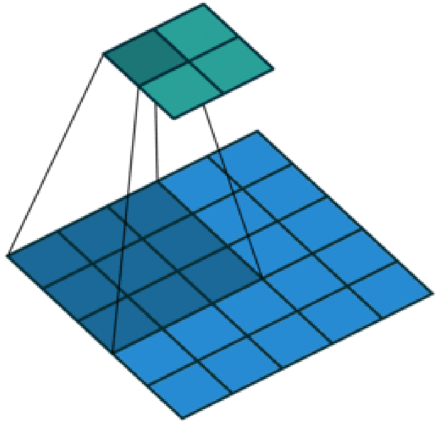
Convolution – Padding, Stride, Step



Padding = 0, Stride = 1



Padding = "same", Stride = 1



Padding = 0, Stride = 2

Image Credits: https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html

Notations

n – Height and Width of an image

H – Height of an image

W – Width of an image

k – Height and Width of a convolution filter (kernel)

k x k – Size of a kernel

C – Number of channels

* - Convolution operator

Convolution – A Linear Transformation

- At each step of the convolution, k x k multiplications occur
- In other words, every value in the output can be thought of as a linear function of kernel weights and subset of input (that is, input values in the corresponding window)
- In fact, we can say input is being linearly transformed by a matrix (M) referred to as Convolution Matrix

Convolution – A Linear Transformation

$$
\begin{array}{|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 \\ \hline
0 & x_{11} & x_{12} & x_{13} & 0 \\ \hline
0 & x_{21} & x_{22} & x_{23} & 0 \\ \hline
0 & x_{31} & x_{32} & x_{33} & 0 \\ \hline
0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}
*
\begin{array}{|c|c|c|}
\hline
w_{11} & w_{12} & w_{13} \\ \hline
w_{21} & w_{22} & w_{23} \\ \hline
w_{31} & w_{32} & w_{33} \\ \hline
\end{array}
=
\begin{bmatrix}
w_{22} & w_{23} & 0 & w_{32} & w_{33} & 0 & 0 & 0 & 0 \\
w_{21} & w_{22} & w_{23} & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 \\
0 & w_{21} & w_{22} & 0 & w_{31} & w_{32} & 0 & 0 & 0 \\
w_{12} & w_{13} & 0 & w_{22} & w_{23} & 0 & w_{32} & w_{33} & 0 \\
w_{11} & w_{12} & w_{13} & w_{21} & w_{22} & w_{23} & w_{31} & w_{32} & w_{33} \\
0 & w_{12} & w_{13} & 0 & w_{22} & w_{23} & 0 & w_{32} & w_{33} \\
0 & 0 & 0 & w_{12} & w_{13} & 0 & w_{22} & w_{23} & 0 \\
0 & 0 & 0 & w_{11} & w_{12} & w_{13} & w_{21} & w_{22} & w_{23} \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22}
\end{bmatrix}
\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix}
=
\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}
$$

Input **Kernel** **M** **Vectorized Input** **Vectorized output**

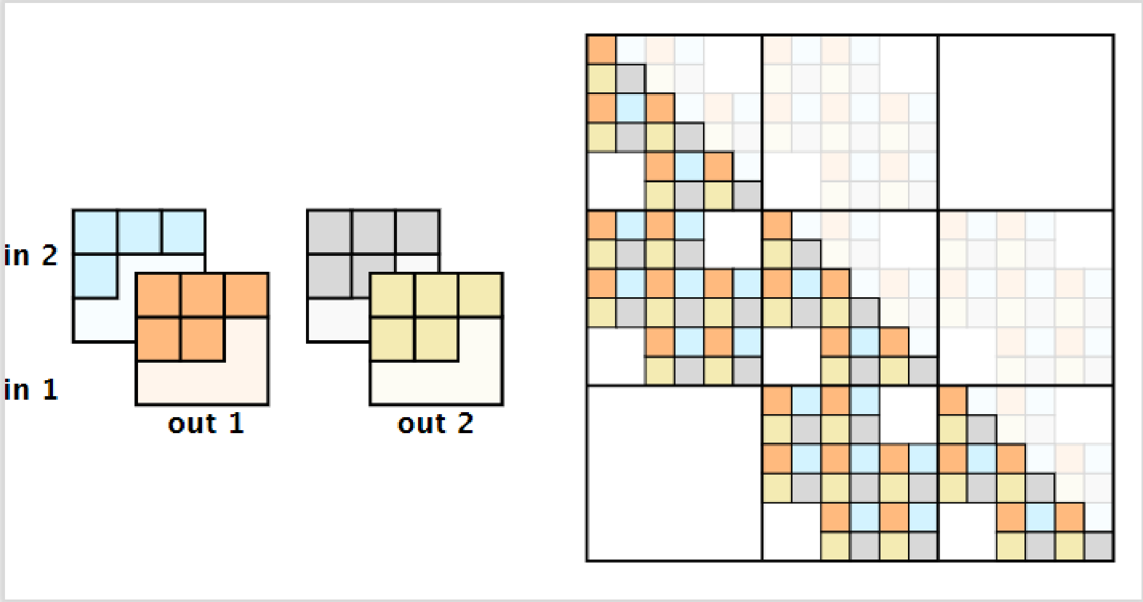
Fig. General Convolution – A linear transformation. **M**, the Convolution Matrix

Inverse Techniques

Emerging Flows

- Leverages the fact that convolution is associative
- Two Auto regressive Convolutions are chained
- Each of the Auto regressive Convolution is chosen so that **M** is triangular
- Inverse Time is $O(n^2k^2)$

$$K_2 * (K_1 * X) = (K_2 * K_1) * X$$

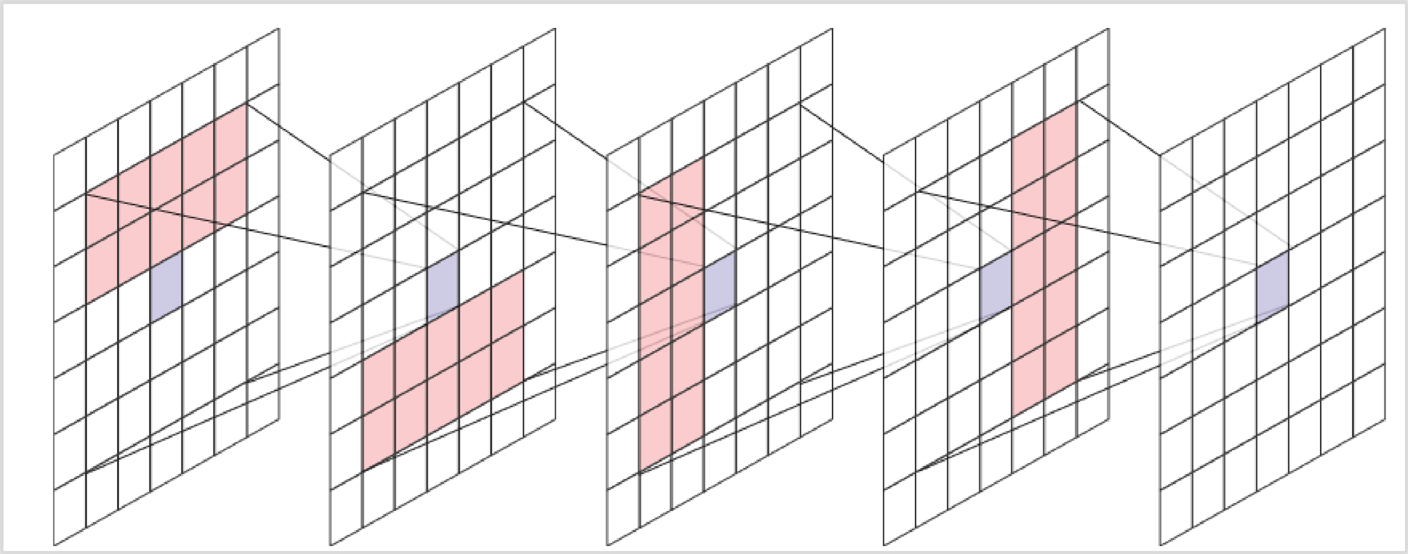


Credits: Hoogeboom, E., Van Den Berg, R., and Welling, M. (2019). Emerging convolutions for generative normalizing flows. In International Conference on Machine Learning, pages 2771–2780. PMLR.

Convolution Techniques

MACOW: Masked Convolutional Flow

- Leverages the fact that convolution is associative
- Four Masked Convolutions are chained
- Inverse Time is $O(nk^2)$

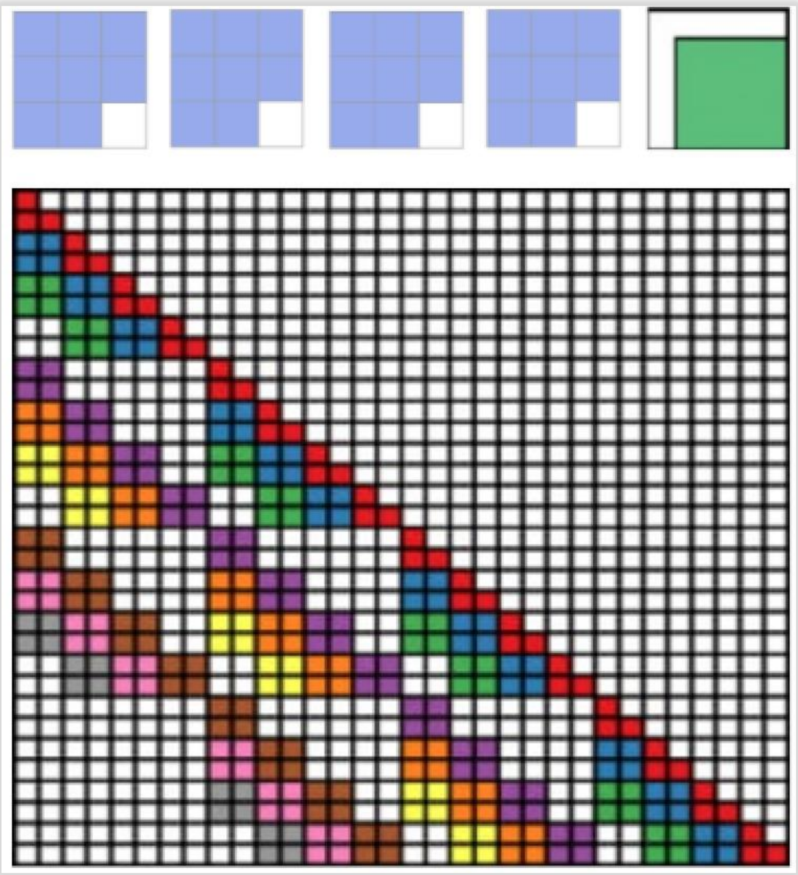


Credits: Ma, X., Kong, X., Zhang, S., and Hovy, E. (2019). Macow: Masked convolutional generative flow. Advances in Neural Information Processing Systems, 32.

Inverse Techniques

CInC Flow

- Padding the input is done so that the resulting **M** is triangular
- Very minimal amount of masking required
- Requires only 1 filter per convolution
- Inverse Time is $O(n^2k^2)$



Credits: Nagar, S., Dufraisse, M., and Varma, G. (2021). CInc flow: Characterizable invertible 3 x 3 convolution. In The 4th Workshop on Tractable Probabilistic Modeling.

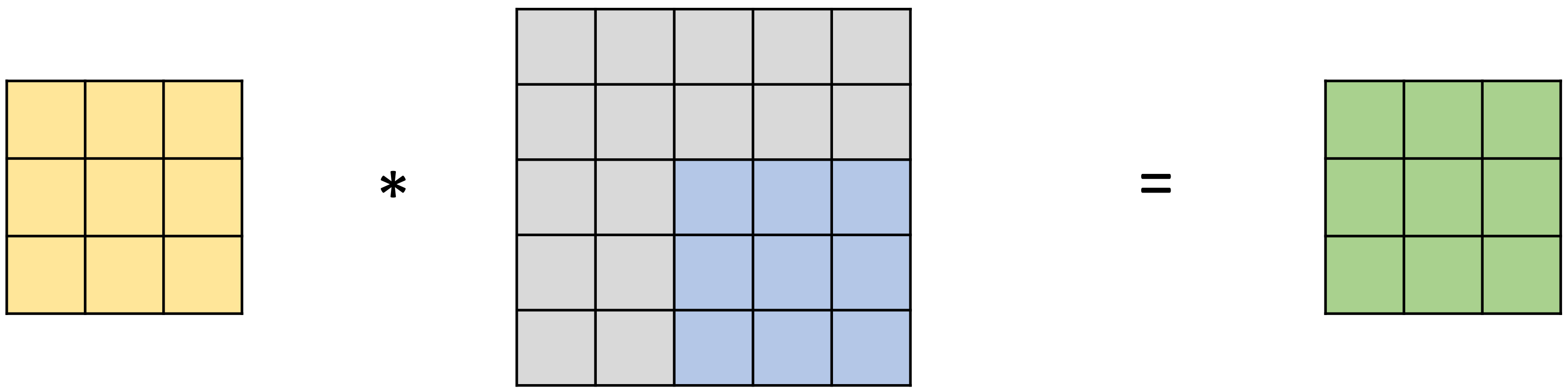
Inverse Techniques

- We observe that techniques mask much of the kernel values and thus resulting in needing more kernels
- We observe that the inverse time is $O(n^2k^2)$
- Can we do better?

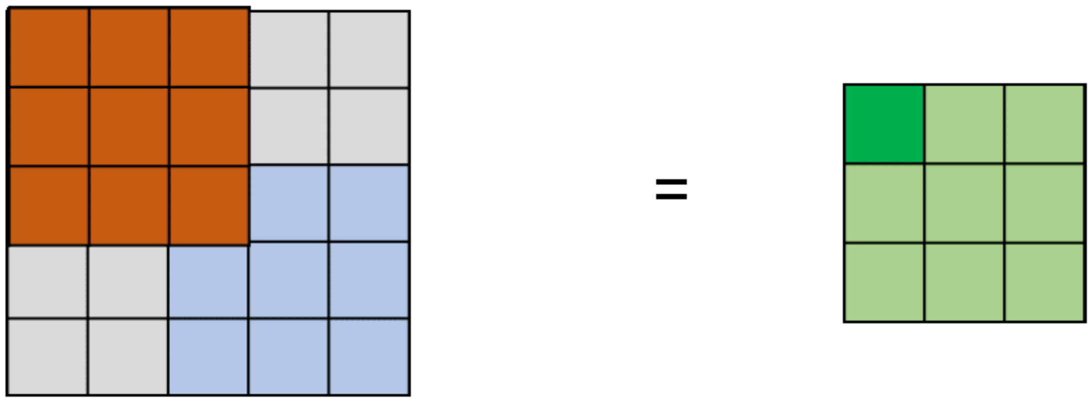
Method	# of ops	# params / CNN layer	Complexity of Jacobian	Inverse
FInC Flow	$(2n-1)k^2$	k^2-1	1	exact
Woodbury	cn^2	k^2	$O(d^2(c+n)+d^3)$	exact
MaCow	$4nk^2$	$k(\lceil \frac{k}{2} \rceil - 1)$	$O(n^3)$	exact
Emerging	$2n^2k^2$	$k(\lceil \frac{k}{2} \rceil - 1)$	$O(n)$	exact
CInC Flow	n^2k^2	k^2-1	1	exact
MintNet	$3n$	$\frac{k^2}{3}$	$O(n)$	approx
SNF	k^2	k^2	approx	approx

n x n – Size of the image; *c* – Number of Channels; *k x k* – Filter Size; *d* – Intermediate Latent Dimension Space

Top Left Convolutions



Top Left Convolutions



Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{12} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

Top Left Convolutions

$$
\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}
$$

$$y_{11} = w_{22} x_{11}$$

Top Left Convolutions

$$\begin{bmatrix} \boxed{x_{11}} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} \boxed{x_{11}} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{12} = w_{21} x_{11} + w_{22} x_{12}$$

Top Left Convolutions

$$\begin{bmatrix} \boxed{x_{11} \quad x_{12}} & x_{13} \\ x_{21} \quad x_{22} & x_{23} \\ x_{31} \quad x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{13} = w_{21} x_{12} + w_{22} x_{13}$$

Top Left Convolutions

$$\begin{bmatrix} \boxed{x_{11} \quad x_{12} \quad x_{13}} \\ x_{21} \quad x_{22} \quad x_{23} \\ x_{31} \quad x_{32} \quad x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{21} = w_{12} x_{11} + w_{22} x_{21}$$

Top Left Convolutions

$$
\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}
*
\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}
=
\begin{bmatrix}
w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\
w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\
0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\
0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\
0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22}
\end{bmatrix}
\times
\begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix}
=
\begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}
$$

$$y_{22} = w_{11}x_{11} + w_{12}x_{12} + w_{21}x_{21} + w_{22}x_{22}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{23} = w_{11} x_{12} + w_{12} x_{13} + w_{21} x_{22} + w_{22} x_{23}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{31} = w_{12} x_{21} + w_{22} x_{31}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{32} = w_{11} x_{21} + w_{12} x_{22} + w_{21} x_{31} + w_{22} x_{32}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ \boxed{0} & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{33} = w_{11} x_{22} + w_{12} x_{23} + w_{21} x_{32} + w_{22} x_{33}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} \boxed{w_{22} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0} \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{11} = w_{22} x_{11}$$

Top Left Convolutions

$$
\begin{bmatrix} \boxed{x_{11}} & \boxed{x_{12}} & x_{13} \\ \boxed{x_{21}} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix}
w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\
w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\
0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\
0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\
0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22}
\end{bmatrix} \times \begin{bmatrix} \boxed{x_{11}} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}
$$

$$y_{12} = w_{21} x_{11} + w_{22} x_{12}$$

$$y_{21} = w_{12} x_{11} + w_{22} x_{21}$$

Top Left Convolutions

$$\begin{bmatrix} \boxed{x_{11}} & \boxed{x_{12}} & \boxed{x_{13}} \\ \boxed{x_{21}} & \boxed{x_{22}} & x_{23} \\ \boxed{x_{31}} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} \boxed{x_{11}} \\ \boxed{x_{12}} \\ x_{13} \\ \boxed{x_{21}} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{13} = w_{21} x_{12} + w_{22} x_{13}$$

$$y_{22} = w_{11} x_{11} + w_{12} x_{12} + w_{21} x_{21} + w_{22} x_{22}$$

$$y_{31} = w_{12} x_{21} + w_{22} x_{31}$$

Top Left Convolutions

$$
\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}
$$

$$y_{23} = w_{11} x_{12} + w_{12} x_{13} + w_{21} x_{22} + w_{22} x_{23}$$

$$y_{32} = w_{11} x_{21} + w_{12} x_{22} + w_{21} x_{31} + w_{22} x_{32}$$

Top Left Convolutions

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & \boxed{x_{33}} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} = \begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\ w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\ w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\ 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\ 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\ 0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} \end{bmatrix} \times \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ y_{21} \\ y_{22} \\ y_{23} \\ y_{31} \\ y_{32} \\ y_{33} \end{bmatrix}$$

$$y_{33} = w_{11} x_{22} + w_{12} x_{23} + w_{21} x_{32} + w_{22} x_{33}$$

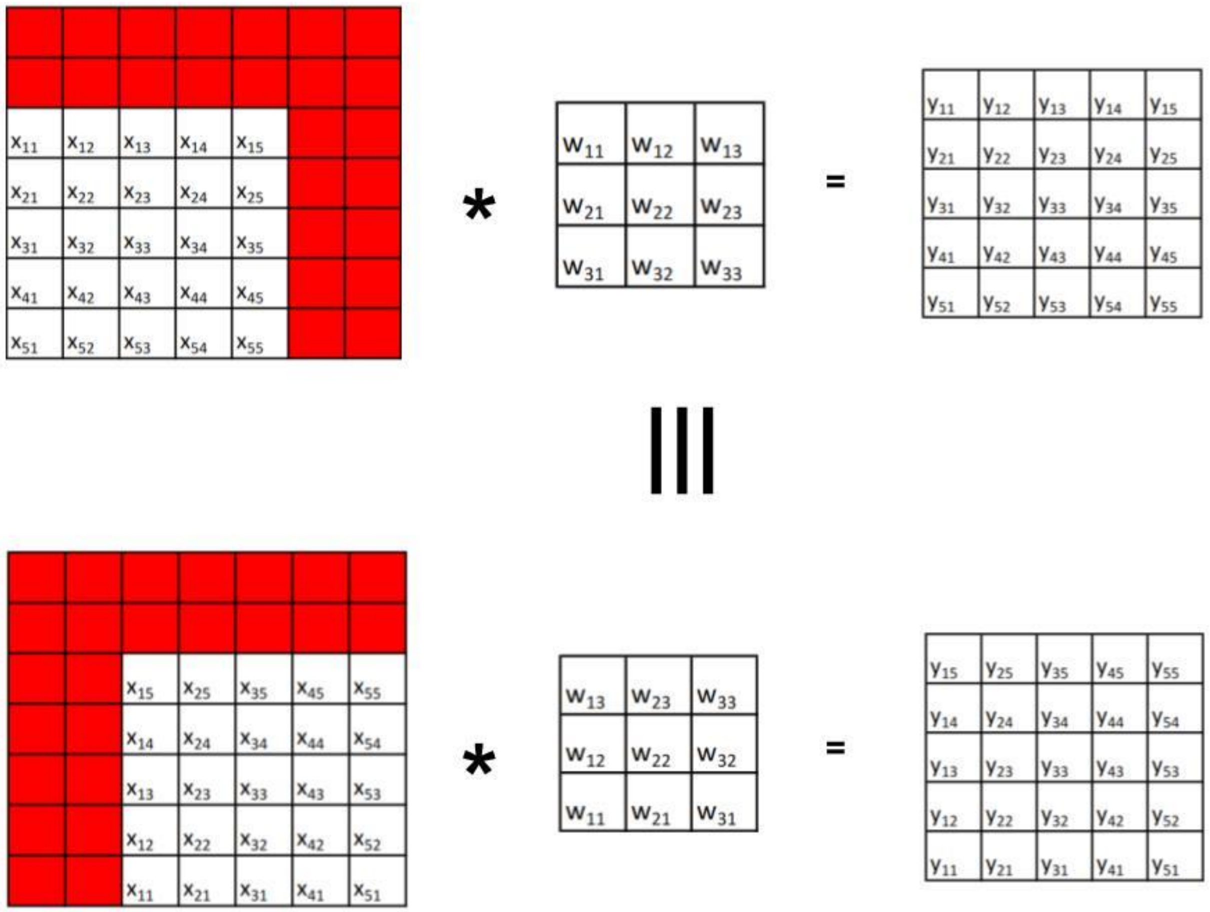
Top Left Convolutions

$$
\begin{array}{|c|c|c|c|}
\hline
0 & 0 & 0 & 0 \\
\hline
0 & x_{11} & x_{12} & x_{13} \\
\hline
0 & x_{21} & x_{22} & x_{23} \\
\hline
0 & x_{31} & x_{32} & x_{33} \\
\hline
\end{array}
*
\begin{array}{|c|c|}
\hline
w_{11} & w_{12} \\
\hline
w_{21} & w_{22} \\
\hline
\end{array}
=
\begin{bmatrix}
w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 & 0 & 0 \\
w_{12} & 0 & 0 & w_{22} & 0 & 0 & 0 & 0 & 0 \\
w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 & 0 \\
0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 & 0 & 0 \\
0 & 0 & 0 & w_{12} & 0 & 0 & w_{22} & 0 & 0 \\
0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22} & 0 \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & w_{21} & w_{22}
\end{bmatrix}
\begin{bmatrix}
x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33}
\end{bmatrix}
$$

Padded Input ($X^{(TL)}$) Kernel

Convolution Matrix (**M**) Vectorized Input(x)

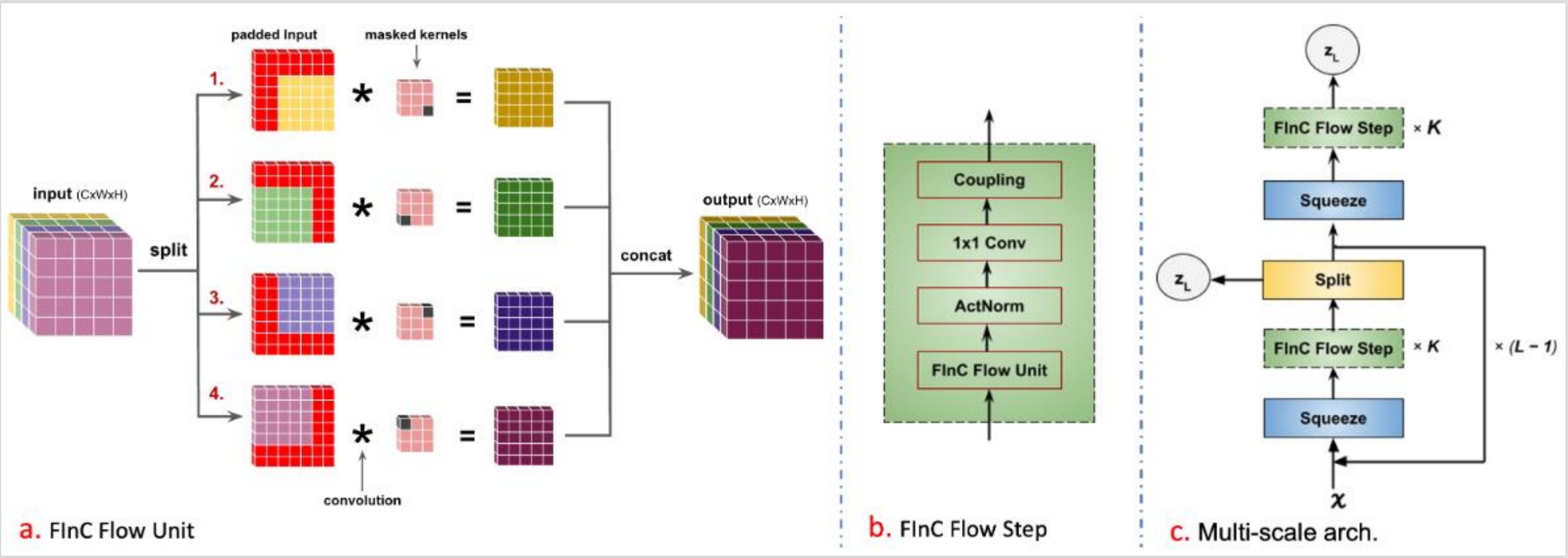
FInC Flow – Our Approach



FInC Flow – Our Approach

FInC Flow

- Padding the input is done so that the resulting **M** is triangular
- Very minimal amount of masking required
- Inverse Time is O(nk^2)



FINC Flow – Architecture

ActNorm

- Performs an affine transformation of the activations using a scale and bias parameter per channel, similar to batch normalization.
- Batch Normalization is not generally used because for large images, we need to train with batch_size = 1 and it creates a problem

$$y_{i,j} = s \odot x_{i,j} + b$$

$$x_{i,j} = \frac{y_{i,j} - b}{s}$$

FINC Flow – Architecture

Affine Coupling Layer

- A powerful reversible transformation where the forward function, the reverse function and the log-determinant are computationally efficient

$$
\begin{aligned}
x_a, x_b &= \text{split}(x) \\
(\log s, t) &= \text{NN}(x_b) \\
s &= \exp(\log s) \\
y_a &= s \odot x_a + t \\
y_b &= x_b \\
y &= \text{concat}(y_a, y_b)
\end{aligned}
$$

$$
\begin{aligned}
y_a, y_b &= \text{split}(y) \\
(\log s, t) &= \text{NN}(y_b) \\
s &= \exp(\log s) \\
x_a &= \frac{y_a - t}{s} \\
x_b &= y_b \\
x &= \text{concat}(x_a, x_b)
\end{aligned}
$$

FINC Flow – Architecture

Invertible 1x1 Convolution

- To incorporate a permutation along the channel dimension, we include a trainable invertible 1×1 convolution layer to generalize the permutation operation as:

$$y_{i,j} = W x_{i,j}$$

$$x_{i,j} = W^{-1} y_{i,j}$$

Bijective Functions

Split

- Input is split into two halves across the channel dimension. We retain the first half and a function parameterized by first half transform the second half.
- The transformed second half is modeled as Gaussian samples, are the latent vectors.

Squeeze

- This function/layer reduces the feature dimension by total four, two across the height dimension and two across the width dimension resulting in increases the channel dimension by four

FInC Flow – Our Approach

Theorem 1: The inverse of the pixels on the diagonals of a TL padded convolution can be computed independently and parallelly

FInC Flow – Our Approach

Theorem 1: The inverse of the pixels on the diagonals of a TL padded convolution can be computed independently and parallelly

Defn: Diagonal Elements: Let $x_{i,j}$ & $x_{i',j'}$ be any two elements of a matrix. Then $x_{i,j}$ & $x_{i',j'}$ are said to be on the same diagonal if $i+j = i'+j'$

FInC Flow – Our Approach

Theorem 1: The inverse of the pixels on the diagonals of a TL padded convolution can be computed independently and parallelly

Defn: Diagonal Elements: Let $x_{i,j}$ & $x_{i',j'}$ be any two elements of a matrix. Then $x_{i,j}$ & $x_{i',j'}$ are said to be on the same diagonal if $i+j = i'+j'$

Proof:

Because this is a TL padded convolution, $y_{i,j}$ depends only on the $k \times k$ window of $x_{\leq i, \leq j}$

$$y_{i,i} = f(x_{\leq i, \leq j})$$

FInC Flow – Our Approach

Theorem 1: The inverse of the pixels on the diagonals of a TL padded convolution can be computed independently and parallelly

Defn: Diagonal Elements: Let $x_{i,j}$ & $x_{i',j'}$ be any two elements of a matrix. Then $x_{i,j}$ & $x_{i',j'}$ are said to be on the same diagonal if $i+j = i'+j'$

Proof:

Because this is a TL padded convolution, $y_{i,j}$ depends only on the $k \times k$ window of $x_{\leq i, \leq j}$

$$y_{i,i} = f(x_{\leq i, \leq j})$$

$$x_{i,j} = \frac{y_{i,j} - f_1(x_{<i,<j})}{w_{k,k}}$$

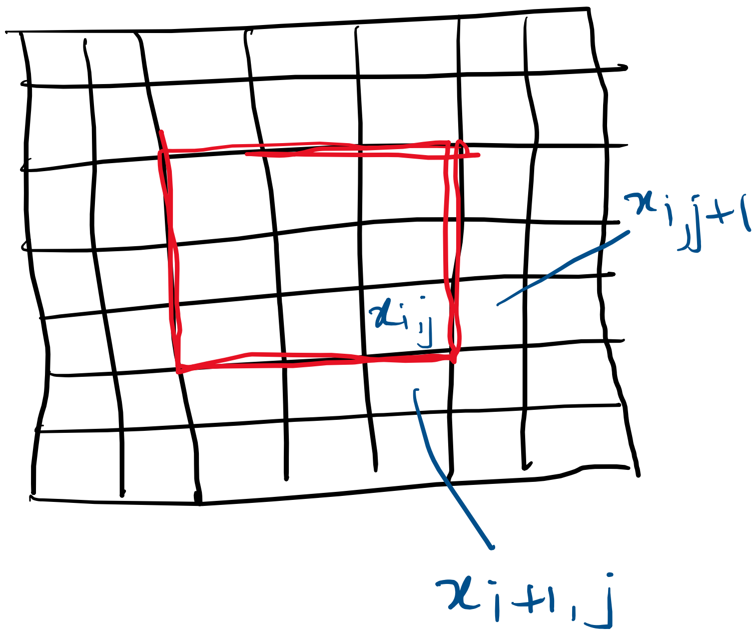
$$x_{i,j} = y_{i,j} - f_1(x_{<i,<j}) \quad \ldots\ldots\ldots \quad (1)$$

FInC Flow – Our Approach

$$x_{i+1,j} = y_{i+1,j} - f_2(x_{<i+1,j})$$

$$= y_{i+1,j} - \alpha\, x_{i,j} - f_3(x_{i+1,j-1}, x_{i+1,j-2}, \ldots)$$

$$- f_4(x_{<i,<j})$$



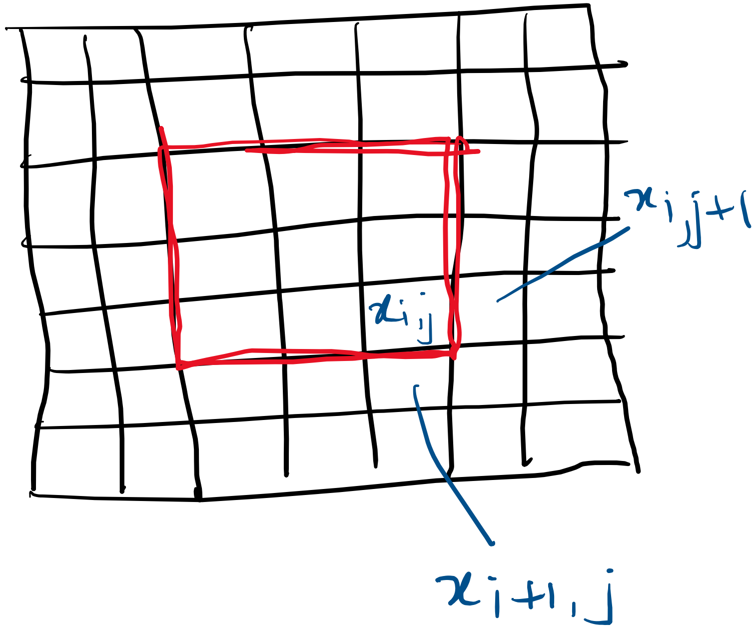
FInC Flow – Our Approach

$$x_{i+1,j} = y_{i+1,j} - f_2(x_{<i+1,j})$$

$$= y_{i+1,j} - \alpha\, x_{i,j} - f_3(x_{i+1,j-1}, x_{i+1,j-2}, \ldots) - f_4(x_{<i,<j})$$

But $x_{i,j}$ & $x_{i+1,j-1}$ can be computed independently

As is the case for $x_{i+1,j-2}$ & $x_{i,j-1}$ & so on.



FInC Flow – Our Approach

$$x_{i+1,j} = y_{i+1,j} - f_2(x_{<i+1,j})$$

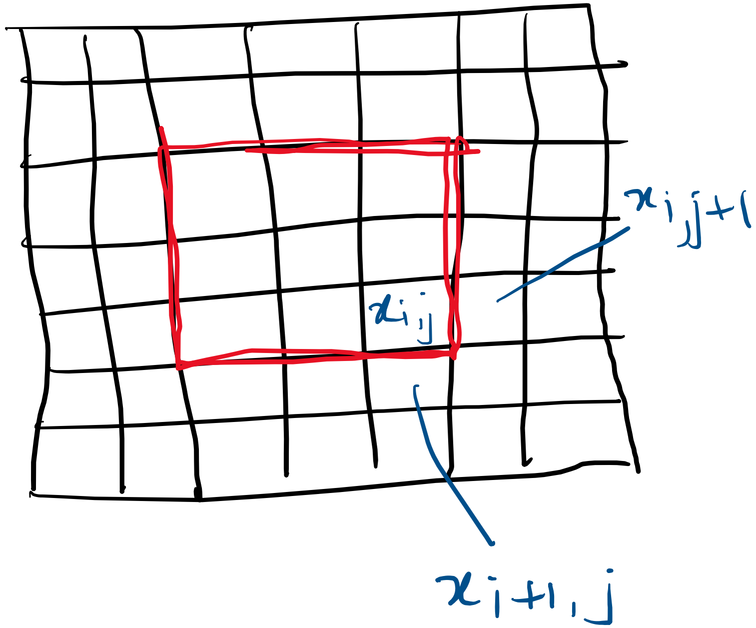
$$= y_{i+1,j} - \alpha\, x_{i,j} - f_3(x_{i+1,j-1}, x_{i+1,j-2}, \ldots) - f_4(x_{<i,<j})$$

But $x_{i,j}$ & $x_{i+1,j-1}$ can be computed independently

As is the case for $x_{i+1,j-2}$ & $x_{i,j-1}$ & so on.

So, $x_{i+1,j} = y_{i+1,j} - \alpha\, x_{i,j} - F_1(\text{remaining pixels})$

||'ly, $x_{i,j+1} = y_{i,j+1} - \beta\, x_{i,j} - F_2(\text{remaining pixels})$



FInC Flow – Our Approach

Algorithm 1: Fast Parallel Inverse Algorithm for a TL Padded Convolution Block

```
Input: K: Kernel of shape (C, C, k_H, k_W)
Y: output of the conv of shape (C, H, W)
Result: X: inverse of the conv. with shape (C, H, W).
Initialization: X ← Y ;
for d ← 0, H + W − 1 do
    for c ← 0, C − 1 do
        /* The below lines of code executes parallelly on different
           threads on GPU for every index (c, h, w) of X on the dth
           diagonal.                                                  */
        for k_h ← 0, k_H − 1 do
            for k_w ← 0, k_W − 1 do
                for k_c ← 0, C − 1 do
                    if pixel (k_c, h − k_h, w − k_w) not out of bounds then
                        X[c, h, w] ←
                          X[c, h, w]−X[k_c, h−k_h, w−k_w]∗K[c, k_c, k_H−k_h−1, k_W−k_w−1];
                    end
                end
            end
        end
        /* synchronize all threads                                    */
    end
end
```

FInC Flow – Our Approach

Theorem *2*: Algorithm *1* uses only $(H + W - 1)k^2$ sequential operations.

Proof:

We have proved in Theorem *1* that the inverse of pixels on a single diagonal can be computed parallelly in one iteration of Algorithm *1*. Since there are $H + W - 1$ number of diagonals in a matrix and there are at maximum k^2 entries in a row of the convolutional matrix, the number of sequential operations needed will be $(H + W - 1)k^2$.

FInC Flow – Our Approach

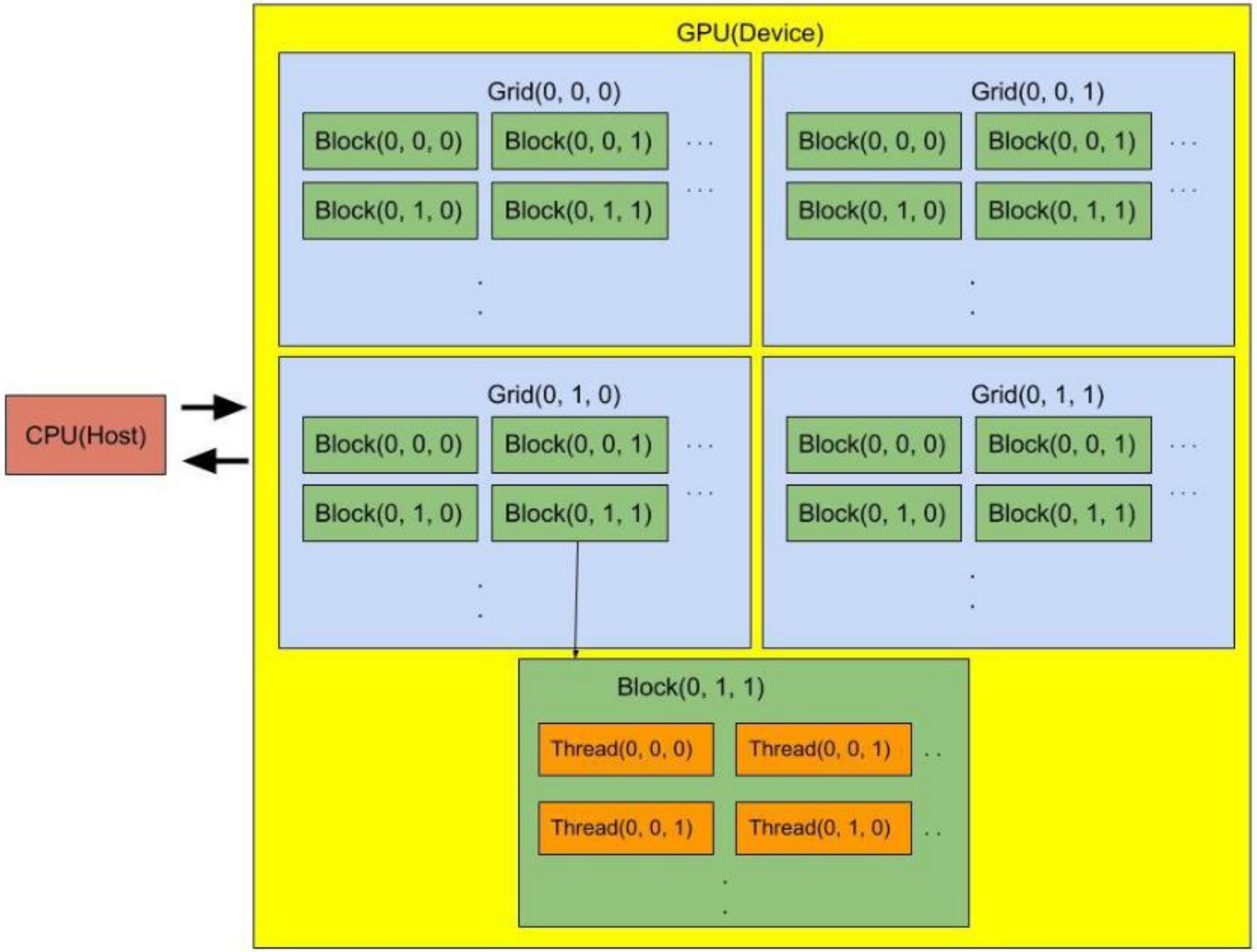
Algorithm 2: Fast Inverse Algorithm for FInC Flow Unit

Input: K_1, K_2, K_3, K_4 - Convolution Kernels of different PCB, Y - Output of the FInC Flow *Unit*

Result: X - Input to the FInC Flow Unit / Inverse of the FInC Flow Unit

1. $Y_1, Y_2, Y_3, Y_4 \leftarrow split(Y)$
2. Flip $Y_2, Y_3, Y_4, K_2, K_3, K_4$ (inplace) appropriately to match TL padding
3. $X \leftarrow concat(Y_1, Y_2, Y_3, Y_4)$
4. $K \leftarrow concat(K_1, K_2, K_3, K_4)$
5. Apply Algorithm 1 with input K, Y to get X
6. $X_1, X_2, X_3, X_4 \leftarrow split(X)$
7. Flip X_2, X_3, X_4 appropriately to get the correct output
8. $X \leftarrow concat(X_1, X_2, X_3, X_4)$

GPU (CUDA) - Architecture



CUDA Architecture

CUDA - Programming

A typical execution of a CUDA C/C++ code involves several steps like

1. Allocate memory on the host for input and output
2. Allocate memory on the device(GPU)
3. Copy data from host to device
4. Launch the kernel - CUDA function and execute it
5. Copy the results back to host
6. Free the memory on both the host and the device

CUDA - Programming

```
int tid = threadIdx.x + blockDim.x * (threadIdx.y + blockDim.y * threadIdx.z);
```

ThreadID Calculation Code

```
int bid = blockIdx.x + gridDim.x * (blockIdx.y + gridDim.y * blockIdx.z);
```

BlockID Calculation Code

Algorithm 1 Code: Link
Algorithm 2 Code: Link

CUDA – Main Function

```
for (int d = 1; d <= H + W - 1; d++) { // Iterating over diagonal index
    for (int c = 0; c < C; c++) {
        // all elements of the dth diagonal computed in parallel
        int relevant_threads = d;
        if (d > H) {
            if (d <= W) {
                relevant_threads = H;
            } else {
                relevant_threads = H + W - d;
            }
        }

        dim3 threads(B, 1, 1);
        dim3 blocks(relevant_threads, 1, 1);

        AT_DISPATCH_FLOATING_TYPES(input.type(), "finc_inverse_cuda", ([&] {
            cinc_cuda_inverse_kernel<scalar_t><<<blocks, threads>>>(
                input.packed_accessor<scalar_t, 4, torch::RestrictPtrTraits,size_t>(),
                kernel.packed_accessor<scalar_t, 4, torch::RestrictPtrTraits,size_t>(),
                output.packed_accessor<scalar_t, 4, torch::RestrictPtrTraits,size_t>(),
                c,
                d,
                relevant_threads
            );
        }));

        cudaDeviceSynchronize();

    }
}
```

CUDA – Kernel

```
const auto tid = blockIdx.x;
const auto b = threadIdx.x;
int h, w, n = H;
// h = tid % m; // batch index encoded in lower indices modulo batchsize
// tid = (tid - h) / m; // remaining part of tid has the index along the diagonal
if (d <= n) {
    h = d - 1 - tid;
    w = tid;
} else {
    w = (d - n)  + tid;
    h = n - 1 - tid;
}

// compute entry of the output in the diagonal d assigned to this thread
output[b][c][h][w] = input[b][c][h][w];
for (int k_h = 0; k_h < K_H; k_h++) {
    if (h - k_h < 0) break;
    for (int k_w = 0; k_w < K_W; k_w++) {
        if (w - k_w < 0) break;
        for (int k_c = 0; k_c < C; k_c++) {
            if (k_h == 0 && k_w == 0) {
                if (k_c == c) continue;
            }
            output[b][c][h][w] -= output[b][k_c][h - k_h][w - k_w] * kernel[c][k_c][K_H - k_h - 1][K_W - k_w - 1];
        }
    }
}
```

Results

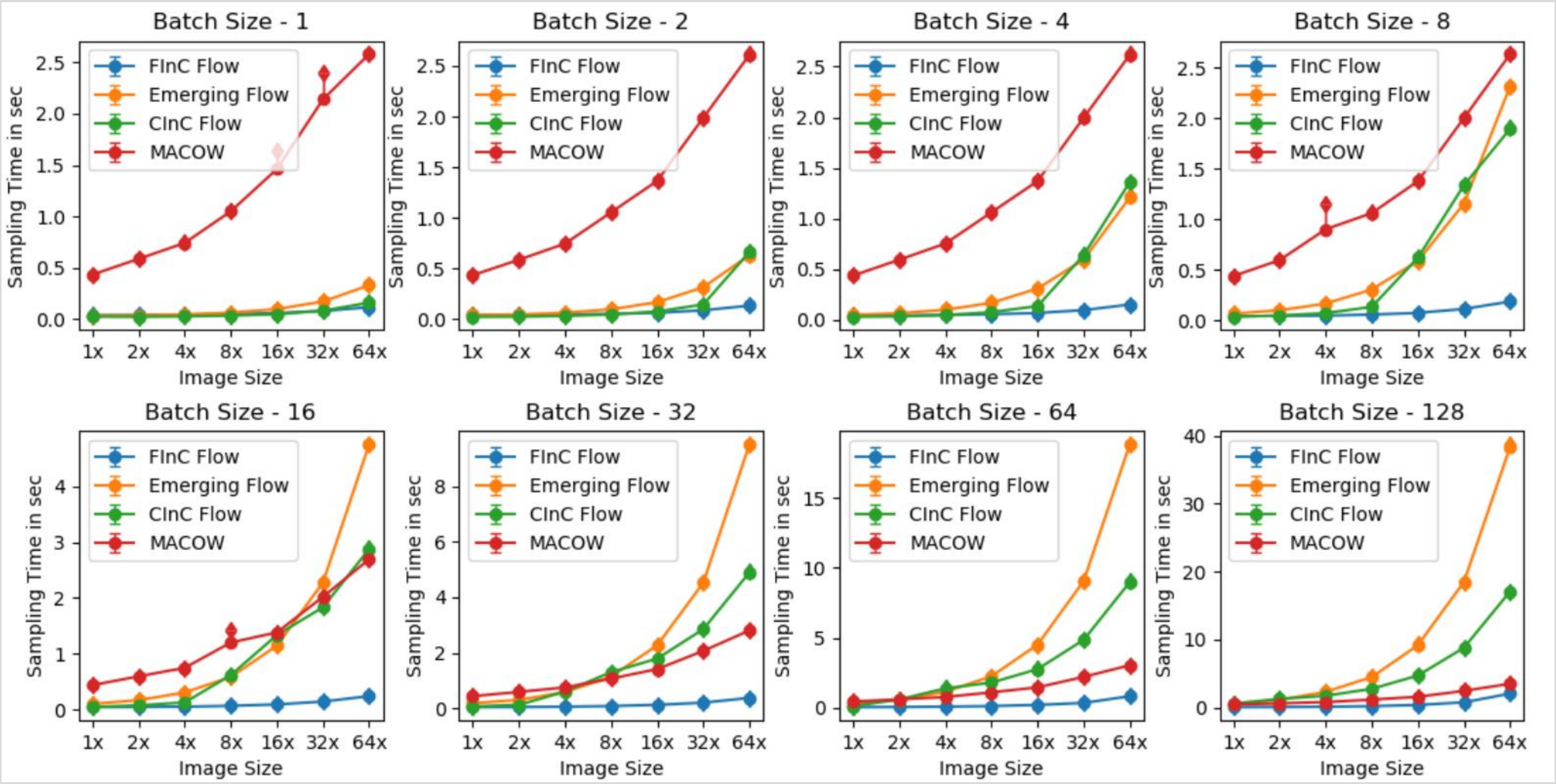
Comparison of BPD, FT, ST with other Convolution Based Models

- Datasets Used:
  - MNIST
  - CIFAR-10
  - Imagenet 32x32
  - Imagenet 64x64

Model	MNIST			CIFAR-10			Imagenet-32x32			Imagenet-64x64		
	BPD	FT	ST	BPD	FT	ST	BPD	FT	ST	BPD	FT	ST
Emerging	–	0.16	0.62	3.34	0.49	17.19	4.09	0.73	25.79	3.81	1.71	137.04
MaCow	–	–	–	3.16	1.49	3.23	–	–	–	3.69	2.91	8.05
CInC Flow	–	–	–	3.35	0.42	7.91	4.03	0.62	11.97	3.85	1.57	55.71
MintNet	0.98	0.16	17.29	3.32	2.09	230.17	4.06	2.08	230.44	–	–	–
FInC Flow (our)	1.05	0.14	0.09	3.39	0.37	0.41	4.13	0.48	0.52	3.88	1.43	2.11

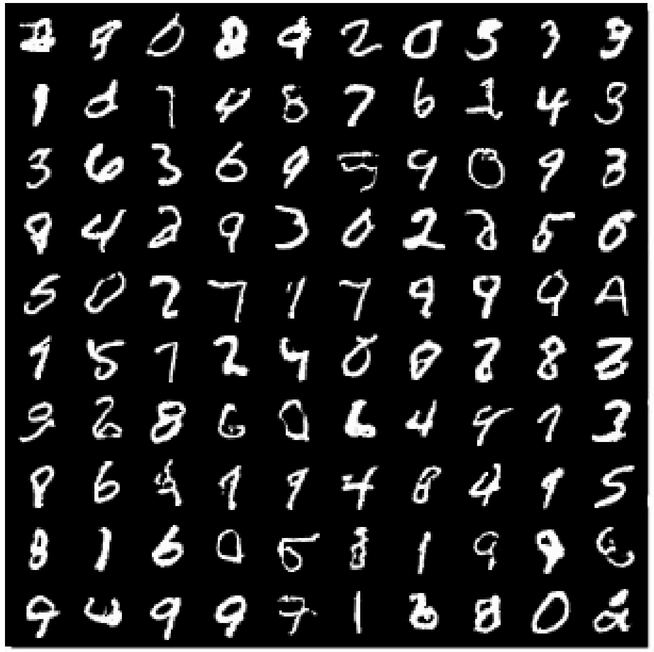
Results

Comparison of Sample Times with other models

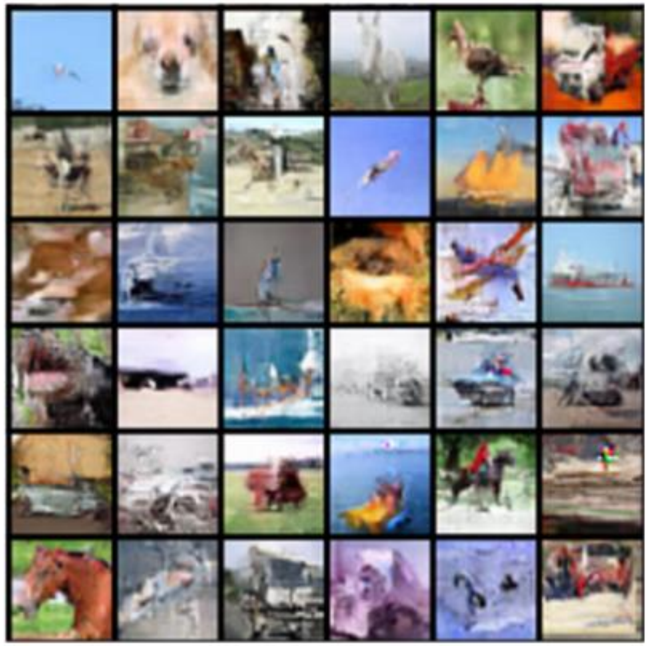


Results

Generate Samples



(a) MNIST



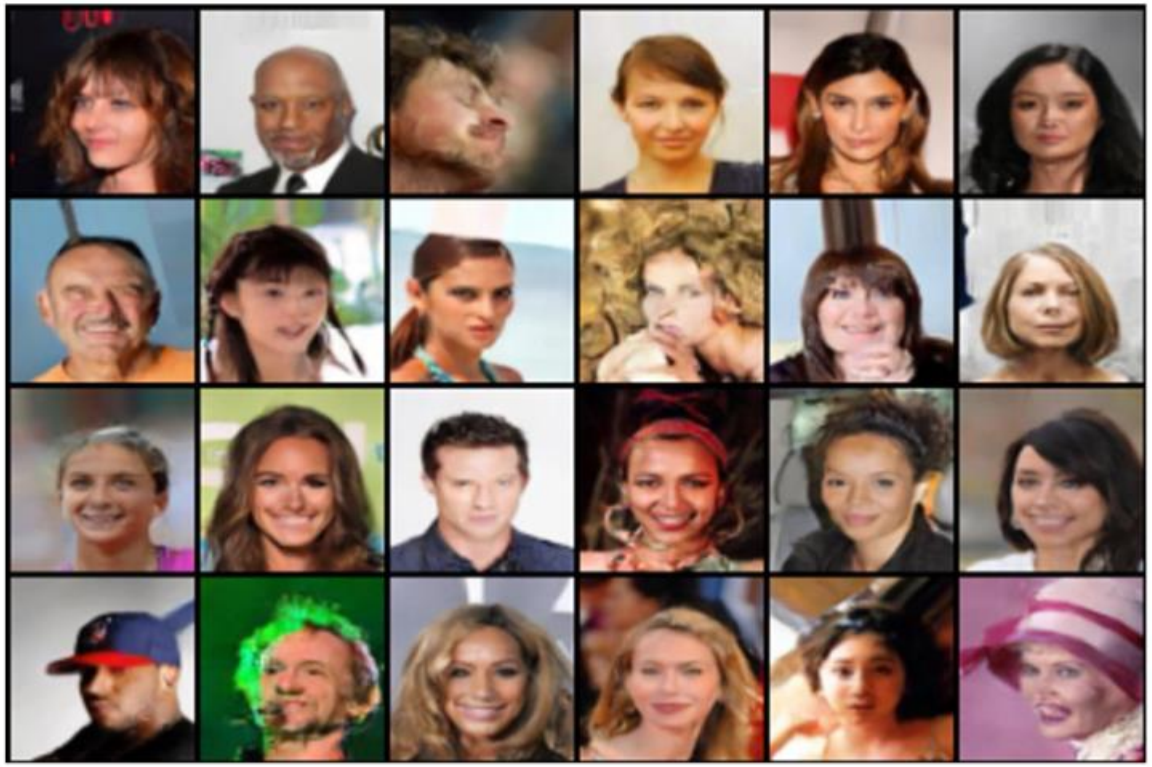
(b) CIFAR-10

(c) ImageNet-64x64

Results

Image Reconstruction

Input Images



Reconstructed Images

Results

Dataset	(L, K)	#Params	#Samples	Algorithm 1 ST	Algorithm 2 ST
MNIST	(2, 16)	10M	1	0.09	0.07
MNIST	(2, 16)	10M	100	0.12	0.09
CIFAR10	(3, 32)	45M	1	0.73	0.30
CIFAR10	(3, 32)	45M	100	0.92	0.47
Imagenet32	(4, 48)	67M	1	1.01	0.44
Imagenet32	(4, 48)	67M	100	1.38	0.73
Imagenet64	(3, 32)	45M	1	1.42	0.50
Imagenet64	(3, 32)	45M	100	2.31	1.42

Comparison of Algorithm 1 and Algorithm 2

Conclusion

With a parallel inversion approach, we present a $k \times k$ invertible convolution for Normalizing flow models. We utilize it to develop a model with highly efficient sampling pass, normalizing flow architecture. We implement our parallel algorithm on GPU and presented benchmarking results, which show a significant enhancement in forward and sampling speeds when compared to alternative methods for $k \times k$ invertible convolution

Any Questions?

https://github.com/aditya-v-kallappa/FInCFlow