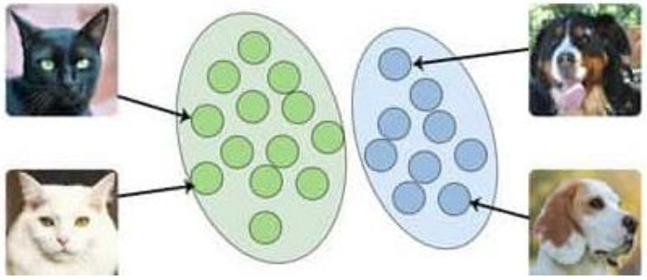
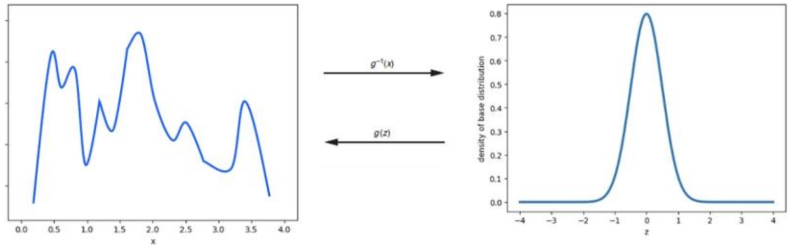
Inverse-Flow: Parallel Backpropagation for Inverse of a Convolution with Application to Normalizing Flows

Sandeep Nagar, Girish Varma

MLL and C-STAR
International Institute of Information Technology Hyderabad

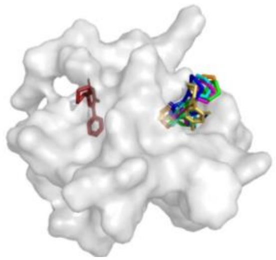
Generative Models



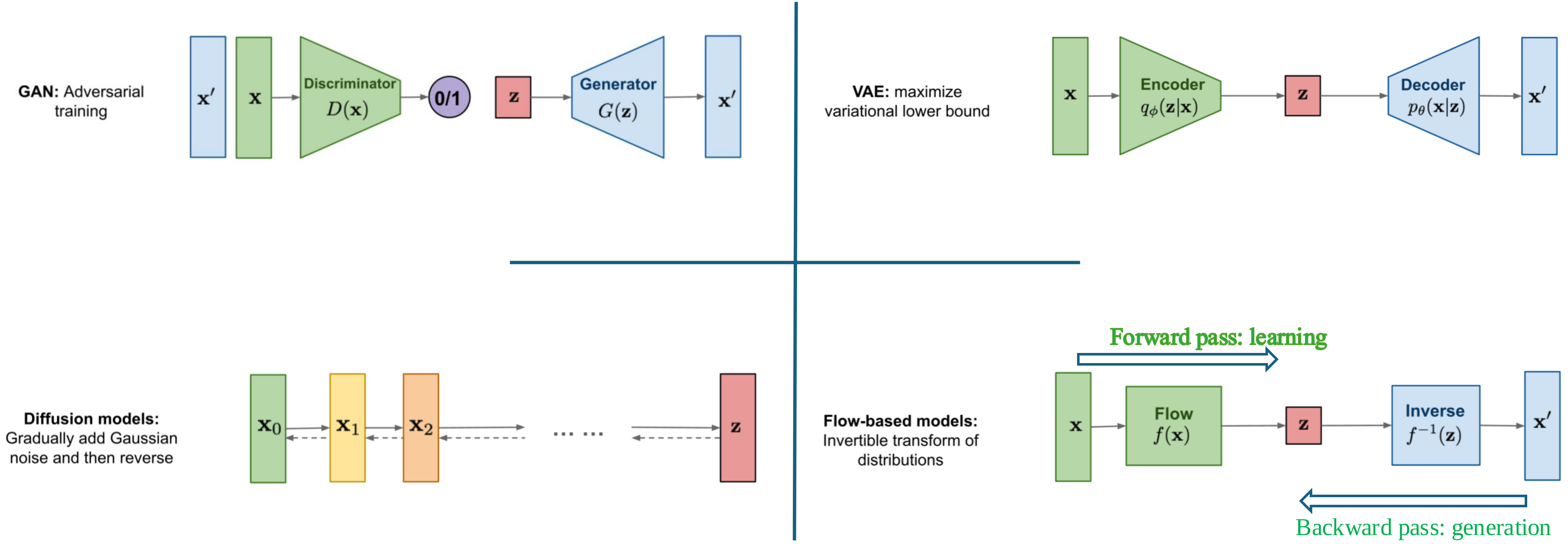


- Learn the underlying probability distribution of the dataset.
- Generate new, previously unseen samples that fit same distribution.
- Generates realistic samples.
- Applications:
  - Images,
  - Audio,
  - Text,
  - Drug design.

“a corgi playing a flame throwing trumpet”



Generative Models[1]



[1]Bond-Taylor, Sam, Adam Leach, Yang Long, and Chris G. Willcocks. "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models." *IEEE transactions on pattern analysis and machine intelligence* (2021).

Designing Fast and Invertible Normalizing Flow Models

A quote from a famous statistician, Goerge Box:

"All models are wrong, but some are useful."

Why NF?

Other Generative Models[1]:

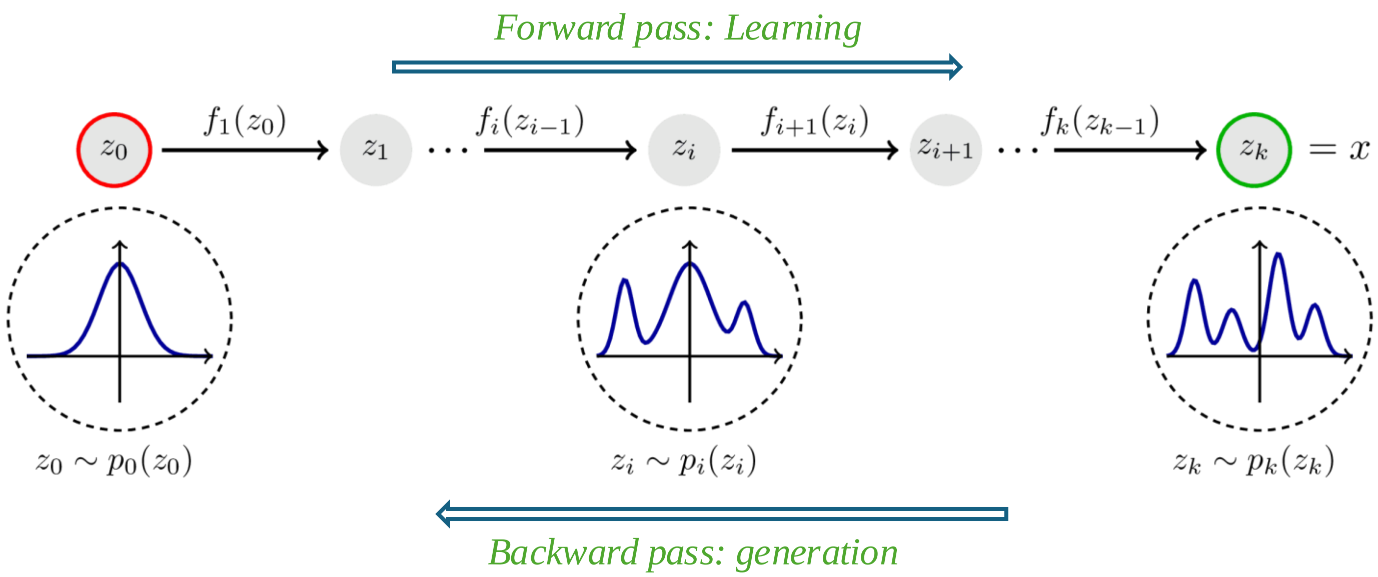
- Approximate models.
- GANs: Optimization can be challenging and unstable.
- VAEs: Blurry samples and struggle capturing complex data.
- AR: Generate samples sequentially.
- EBMs: High variance training, Slow training, and sampling.

Normalizing flow Models[1]:

- Probabilistic models,
- High-dimensional spaces,
- Combinatorial spaces,
- Tractable
- One-to-one mapping.
- Latent space exploration (z).
- Need Fast Invertible Function, y = f(x)
- Intuitive Maximum Likelihood loss function

[1]Bond-Taylor, Sam, Adam Leach, Yang Long, and Chris G. Willcocks. "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models." *IEEE transactions on pattern analysis and machine intelligence* (2021).

Normalizing Flows Models



- Coupling layers:
- Autoregressive Flows:

$$\boldsymbol{x}_K = f_K \circ \cdots \circ f_2 \circ f_1(\boldsymbol{x}_0),$$

$$\ln p_K(\boldsymbol{x}_K) = \ln p_0(\boldsymbol{x}_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial \boldsymbol{x}_{k-1}} \right|.$$

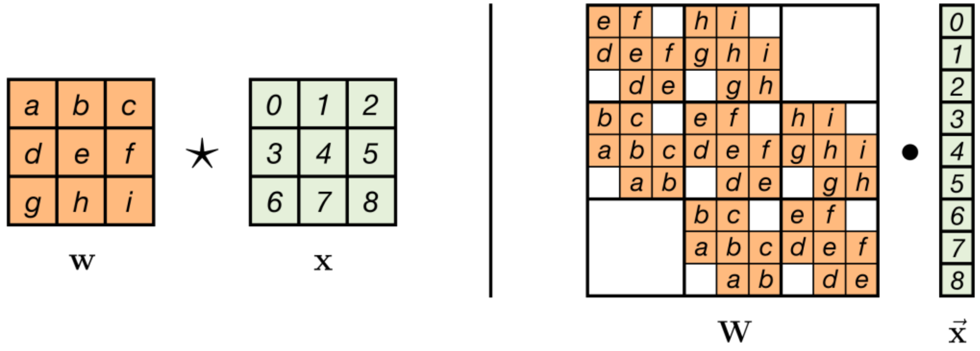
Problem flow models:

- restricted triangular Jacobian,
- meaning that all inputs cannot interact with each other.

[1]Bond-Taylor, Sam, Adam Leach, Yang Long, and Chris G. Willcocks. "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models." *IEEE transactions on pattern analysis and machine intelligence* (2021).

Complexity of finding the Inverse of Convolution

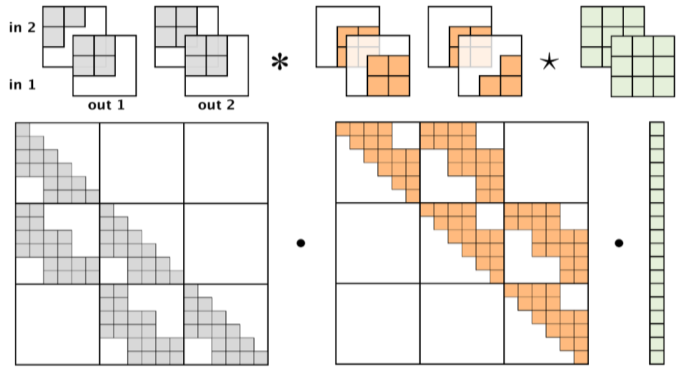
Std. Convolution:



Gaussian Elimination $O(n^4)$

n: input size
k: inv kernel size

Emerging Convolution[1]:



QR decomposition $O(n^3)$

E. Hoogeboom et. al. ICML'19

CInC Convolution[2]:

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

Back Substitution $O(n^2)$

S. Nagar et. al. TPM- W, UAI'21

FInC Convolution[4]:

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

Back Substitution and Parallel $O(k^2 n)$

k: input size

A. Kallappa et. al. VISAPP'23

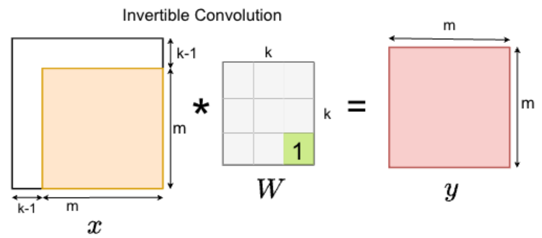
Backpropagation algorithm for the Inverse of Convolution Layer

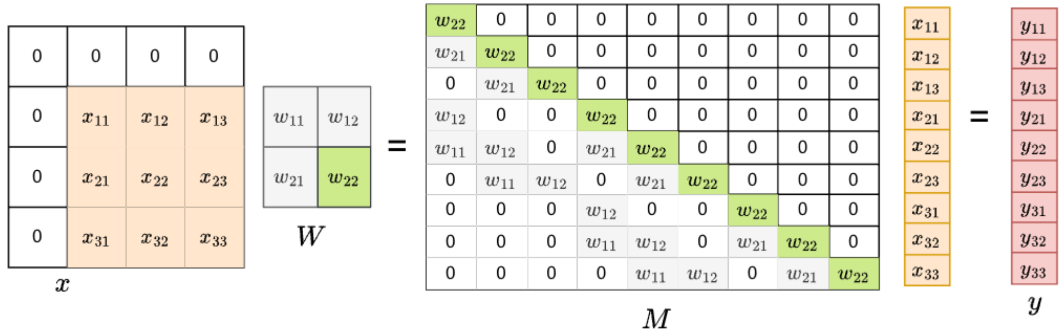
For training: $x = y * W'$ Inverse of Std Conv.

For sampling: $y = x * W$ Std. Convolution

where w' = inv(W).

Advantages:
- Independent of the size (m)
- Fast sampling
- Backprop algorithm for the Inverse of Convolution





$$y_p = x_p + \sum_{q \in \Delta(p)} W_{(k,k)-p+q} \cdot x_q \qquad (1)$$

Using chain rule of differentiation, we get that

$$\frac{\partial L}{\partial y_p} = \sum_q \frac{\partial L}{\partial x_q} \times \frac{\partial x_q}{\partial y_p}. \qquad (2)$$

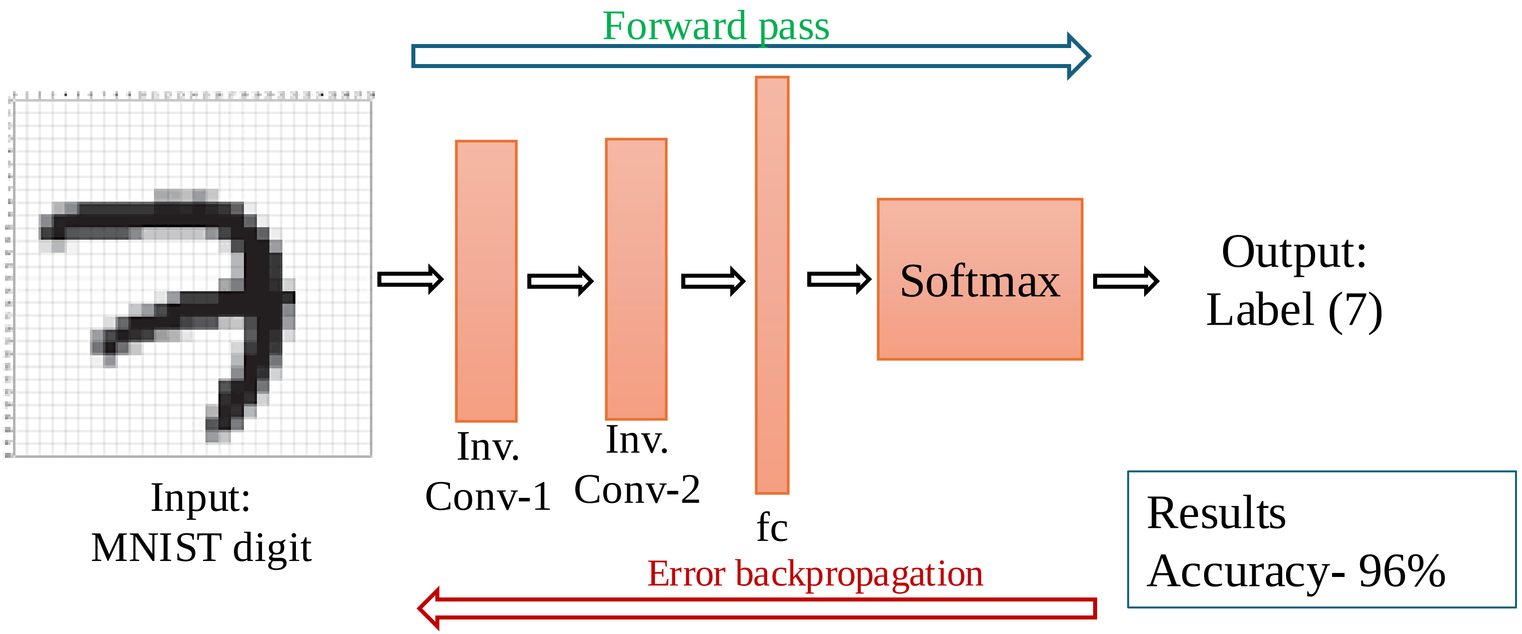
Theorem 1.

$$\frac{\partial x_q}{\partial y_p} = \begin{cases} 1 - \sum_{q \in \Delta(p)} W_{(k,k)-p+q} \cdot \frac{\partial x_q}{\partial y_p} & \text{if } p = q \\ 0 & \text{if } q \not\leq p \\ -\sum_{r \in \Delta(p)} W_{(k,k)-r} \frac{\partial x_{p-r'}}{\partial y_p} & \text{otherwise.} \end{cases}$$

Theorem 2.

$$\frac{\partial x_q}{\partial W_a} = \begin{cases} 0 & \text{if } q \leq a \\ -\sum_{q' \in \Delta q(a)} W_{q'-a} \cdot \frac{\partial x_{q-q'}}{\partial W_a} - x_{q-a} & \text{if } q > a \end{cases}$$

Inverse-Flow: Inverse of Convolution and its Backpropagation Algorithm



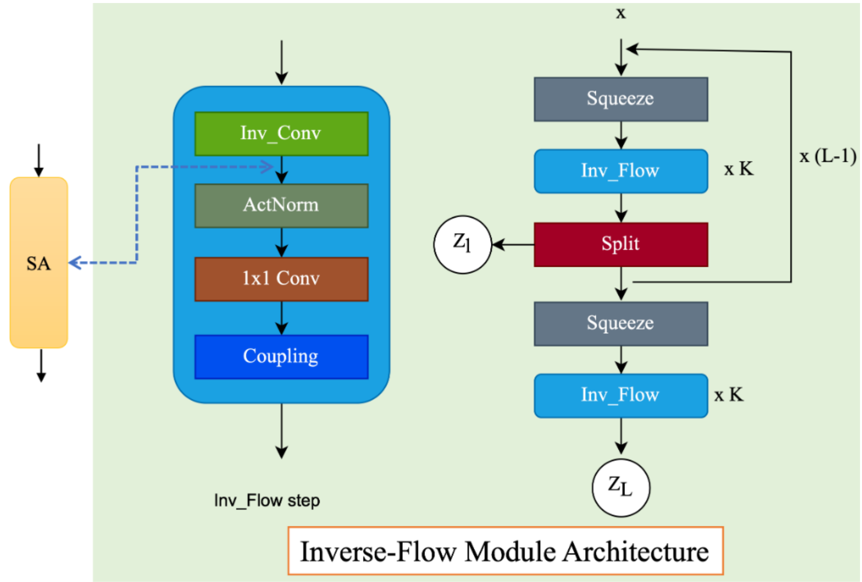
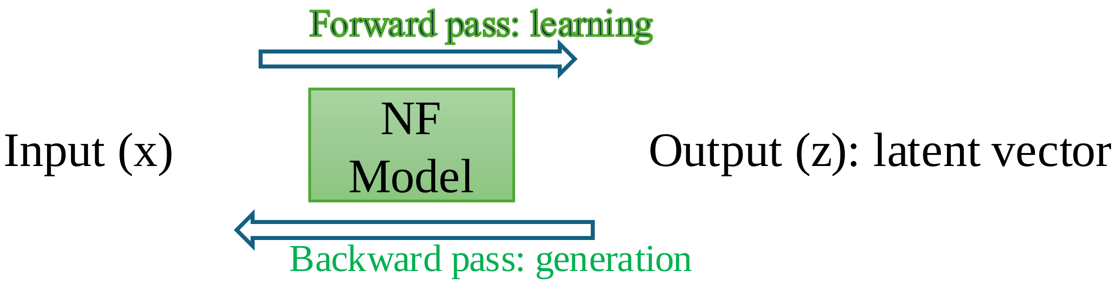


Image Generation using the above-proposed Inverse Convolution Layers and Backpropagation Algorithm



(a) Original (b) Reconstructed

Inverse-Flow: Inverse of Convolution and Backpropagation Algorithm

Results:

Table 3: Performance comparison for MNIST with block size ($K = 16$) and number of blocks ($L = 2$).

Method	ST	NLL	BPD	Param	Inverse
SNF	99 ±2.1	699	1.28	10.1	approx
FIncFlow	90 ±2.2	655	1.15	10.2	exact
MintNet	320±2.8	630	0.98	125.9	approx
Emerging	814±6.2	640	1.09	11.4	exact
Inverse-Flow	52 ±1.3	710	1.31	1.6	exact

Table 6: Performance comparison for CIFAR dataset with block size ($K = 16$) and number of blocks ($L = 2$). SNF uses approx for inverse, and MintNet uses autoregressive functions. *time in seconds.

Method	BPD	ST	FT	Param
SNF	3.52	16.8 ±2.7	609 ±5.4	1.682
MintNet	3.51	25.0* ±1.5	2458 ±6.2	12.466
Woodbury	3.48	7654.4 ±13.5	119 ±2.5	12.49
MaCow	3.40	790.8 ±4.3	1080 ±6.6	2.68
CInC Flow	3.46	1710.0 ±9.5	615 ±5.0	2.62
Butterfly Flow	3.39	311.8 ±4.0	1325 ±7.5	12.58
FInc Flow	3.59	194.8 ±2.5	548 ±6.2	2.72
Inverse-Flow	3.57	91.6 ±6.5	722 ±7.0	1.76

Ablation Study:

input size	Sampling Time (ST)	Forward Time (FT)	GPU memory (GB)
256x256	16.54 ±0.21	427 ±20	8.025
128x128	16.22 ±0.23	342 ±15	3.822
64x64	15.94 ±0.14	264 ±11	1.316
32x32	15.53 ±0.29	224 ±11	0.375
16x16	15.55 ±0.12	211 ±10	0.131

Table 7: Inverse-Flow: K = 2, L = 32, Sample size = 100, batch size = 100. All times in milliseconds

input size	ST	FT	memory (GB)
256x256	23.08 ±0.17	285 ±09	8.016
128x128	20.74 ±0.15	265 ±15	3.822
64x64	17.99 ±0.40	243 ±11	1.314
32x32	16.71 ±0.12	238 ±11	0.359
16x16	17.21 ±0.22	255 ±10	0.129

Table 8: FInC Flow: K = 2, L=32. Sample size =100, batch size = 100. All times in milliseconds

kernel size	Sampling Time (ST)	Forward Time (FT)	Params (M)
11x11	17.87 ±0.42	2428 ±28	6.068
9x9	17.36 ±0.22	1762 ±30	5.146
7x7	19.20 ±0.10	1461 ±23	4.407
5x5	20.79 ±0.15	934 ±17	3.856
3x3	18.11 ±0.20	402 ±08	3.487
2x2	17.90 ±0.25	364 ±07	3.372

Table 10: Inverse Flow model size and sampling time for different kernel sizes: Model Arch, K = 2, L=32. Sample size = 100, batch size = 100. All times in milliseconds

Questions?