CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

CSDS 600: Deep Generative Models

Normalizing Flow Models (2)

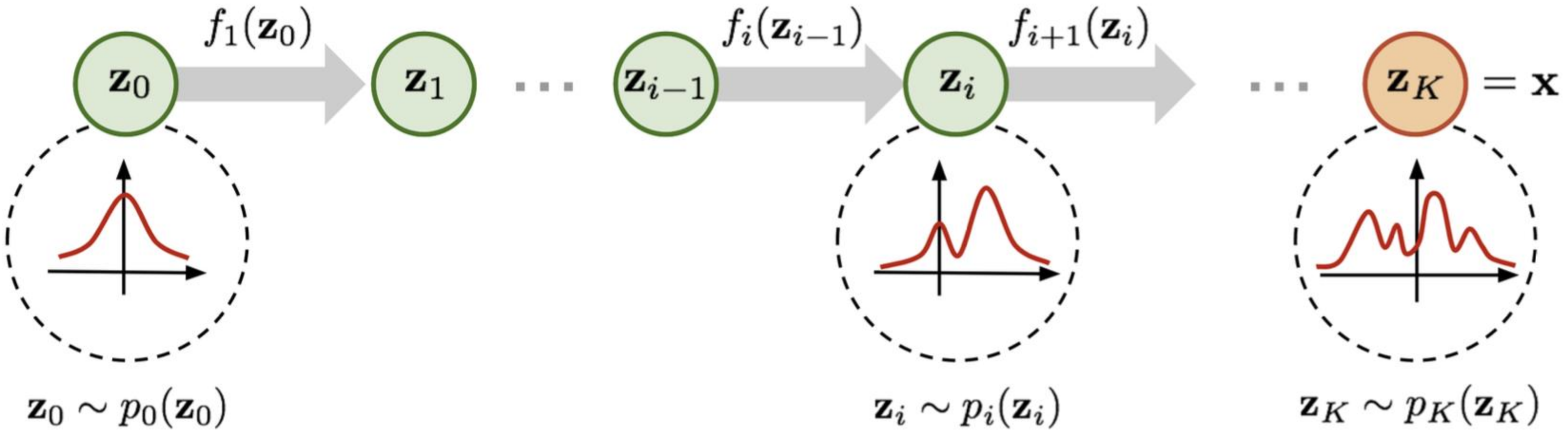
Yu Yin (yu.yin@case.edu)

Case Western Reserve University

https://yin-yu.github.io/

Recap:

- Transform simple to complex distributions via sequence of invertible transformations



- Learning via maximum likelihood over the dataset
- What we need?
  - Prior $\pi(z)$ easy to sample
  - Invertible transformations
  - Determinants of Jacobian Efficient to compute

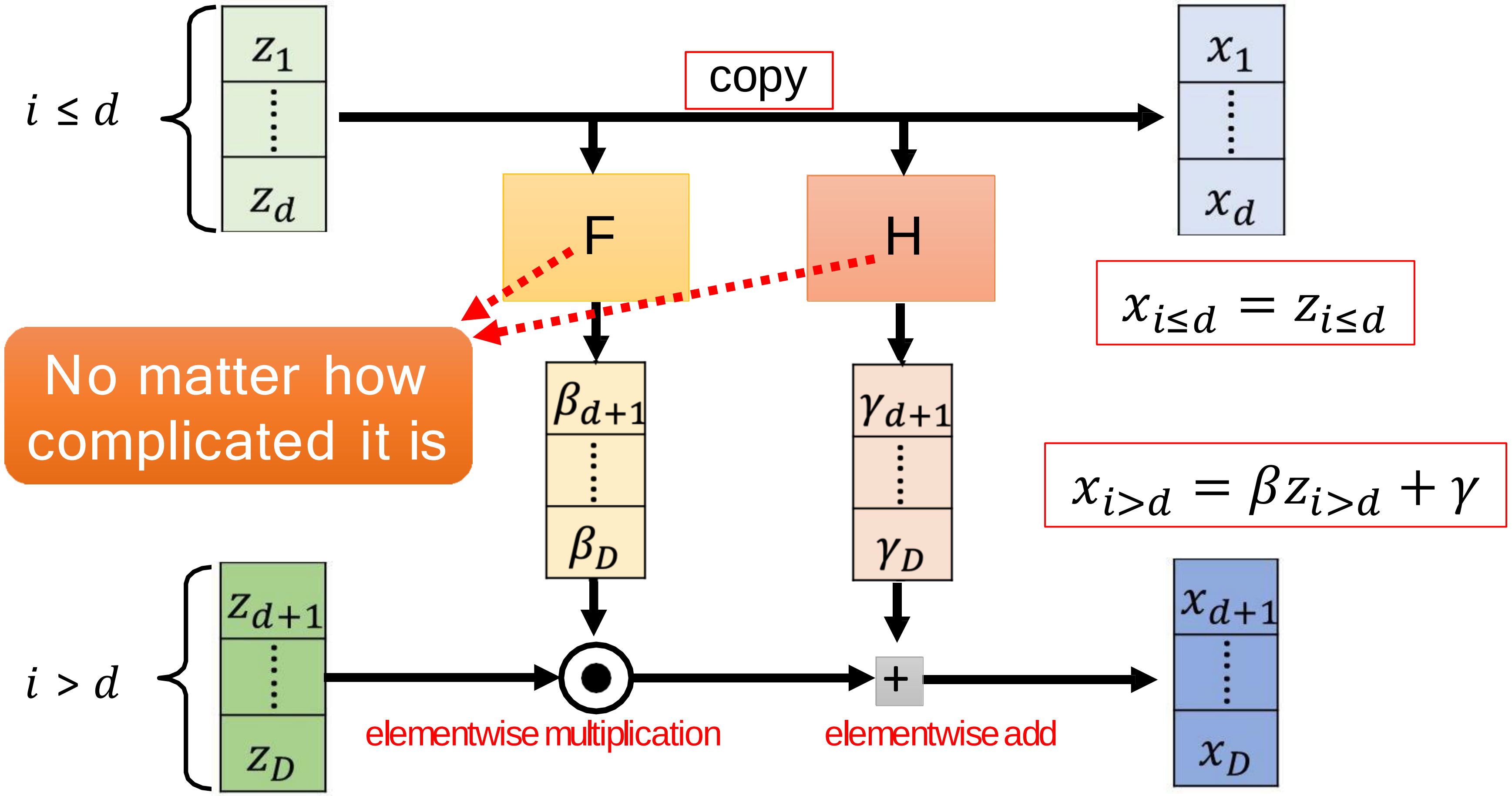
Designing invertible transformations

- A flow of transformations
  - Coupling layer
  - NICE
  - Real NVP
  - Glow
- Autoregressive models as flow models
  - MAF
  - IAF

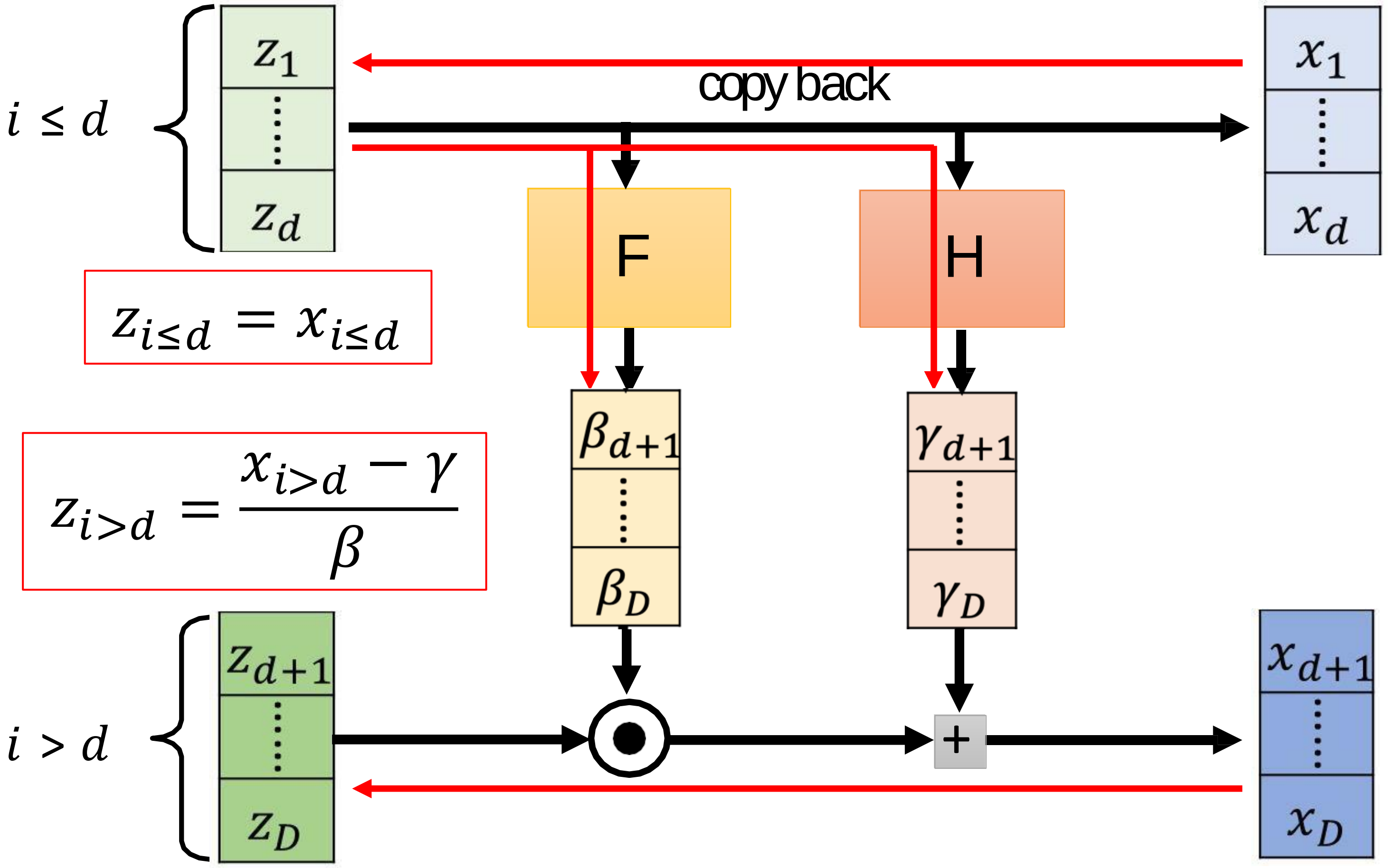
Coupling Layer

NICE

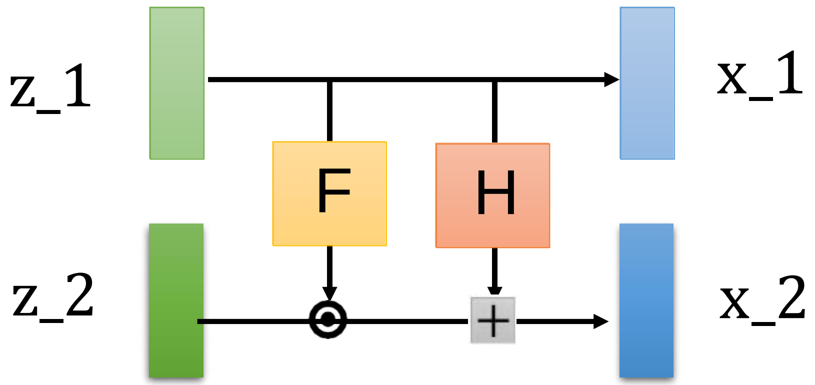
Real NVP



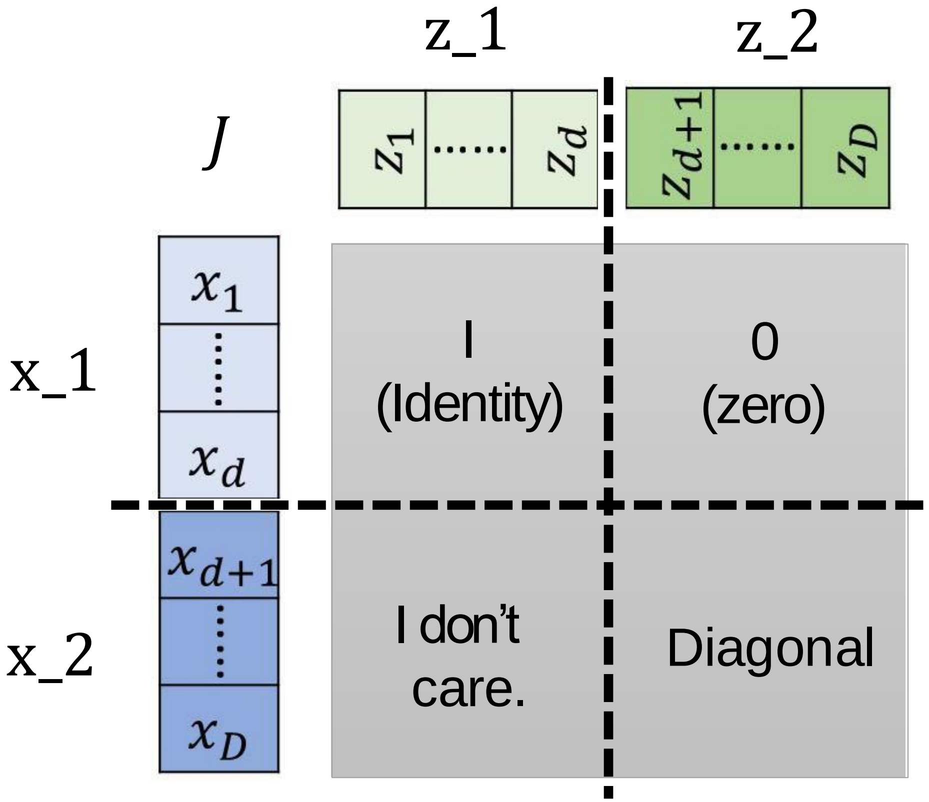
Coupling Layer



Coupling Layer



$$x_{i>d} = \beta_i z_{i>d} + \gamma_i$$

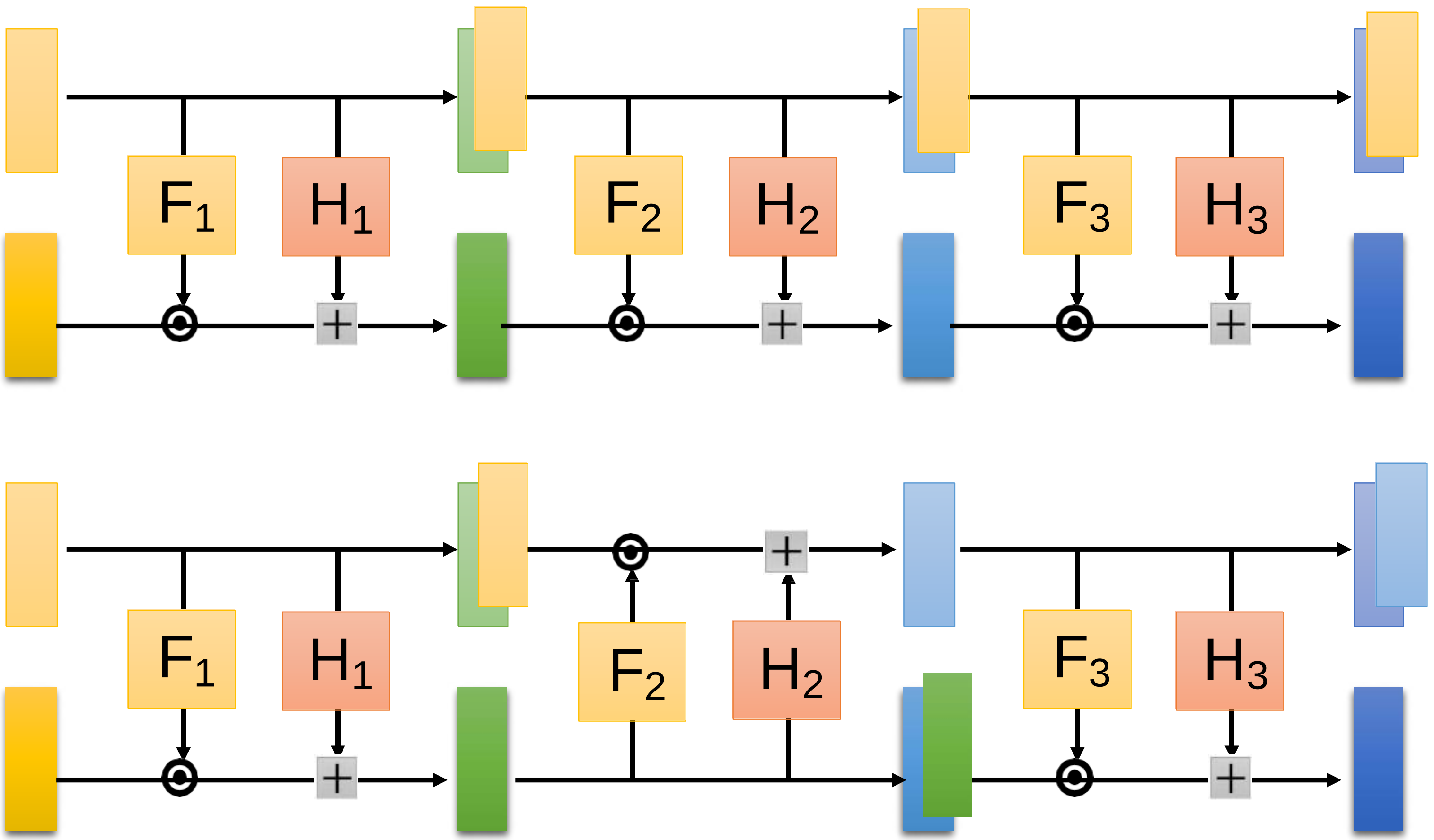


$$\det(J)$$

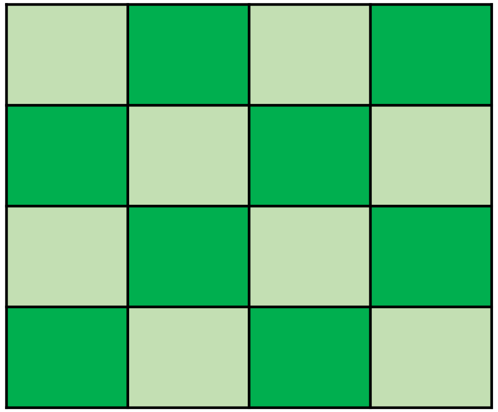
$$= \frac{\partial(x_{d+1})}{\partial(z_{d+1})}\frac{\partial(x_{d+2})}{\partial(z_{d+2})}\cdots\frac{\partial(x_{\mathrm{D}})}{\partial(z_{\mathrm{D}})}$$

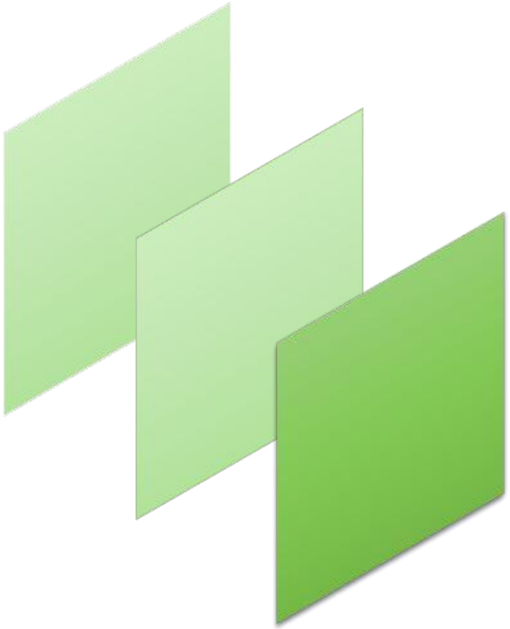
$$= \beta_{d+1}\beta_{d+2}\cdots\beta_{\mathrm{D}}$$

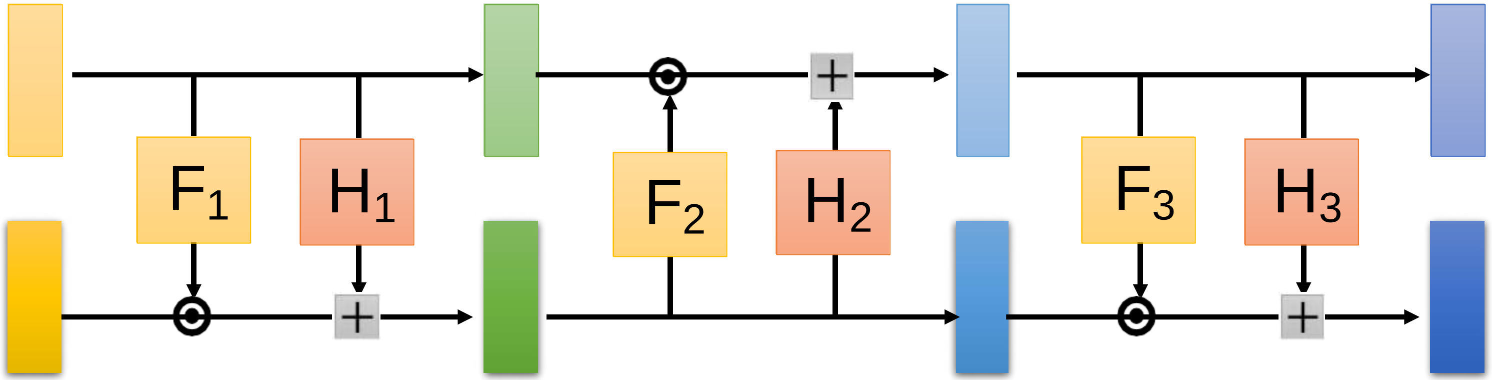
Coupling Layer - Stacking



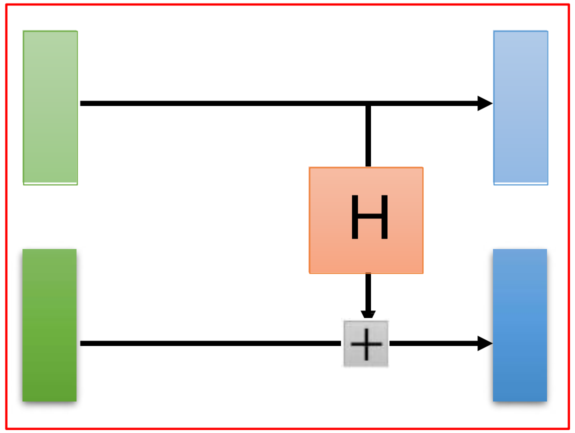
Coupling Layer







NICE: Nonlinear Independent Components Estimation



- Additive coupling layers
  - Partition the variables z into two disjoint subsets
  - $x_{1:d} = z_{1:d}$
  - $x_{d+1:n} = z_{d+1:n} + H(z_{1:d})$
  - Volume preserving transformation since determinant is 1.
- Additive coupling layers are composed together (with arbitrary partitions of variables in each layer)
- Final layer of NICE applies a rescaling transformation

Dinh et al., 2014. Nonlinear Independent Components Estimation

NICE

Rescaling layers

- Forward:

$x_i = \beta_i z_i$, where s_i > 0 is the scaling factor for the i-th dimension.

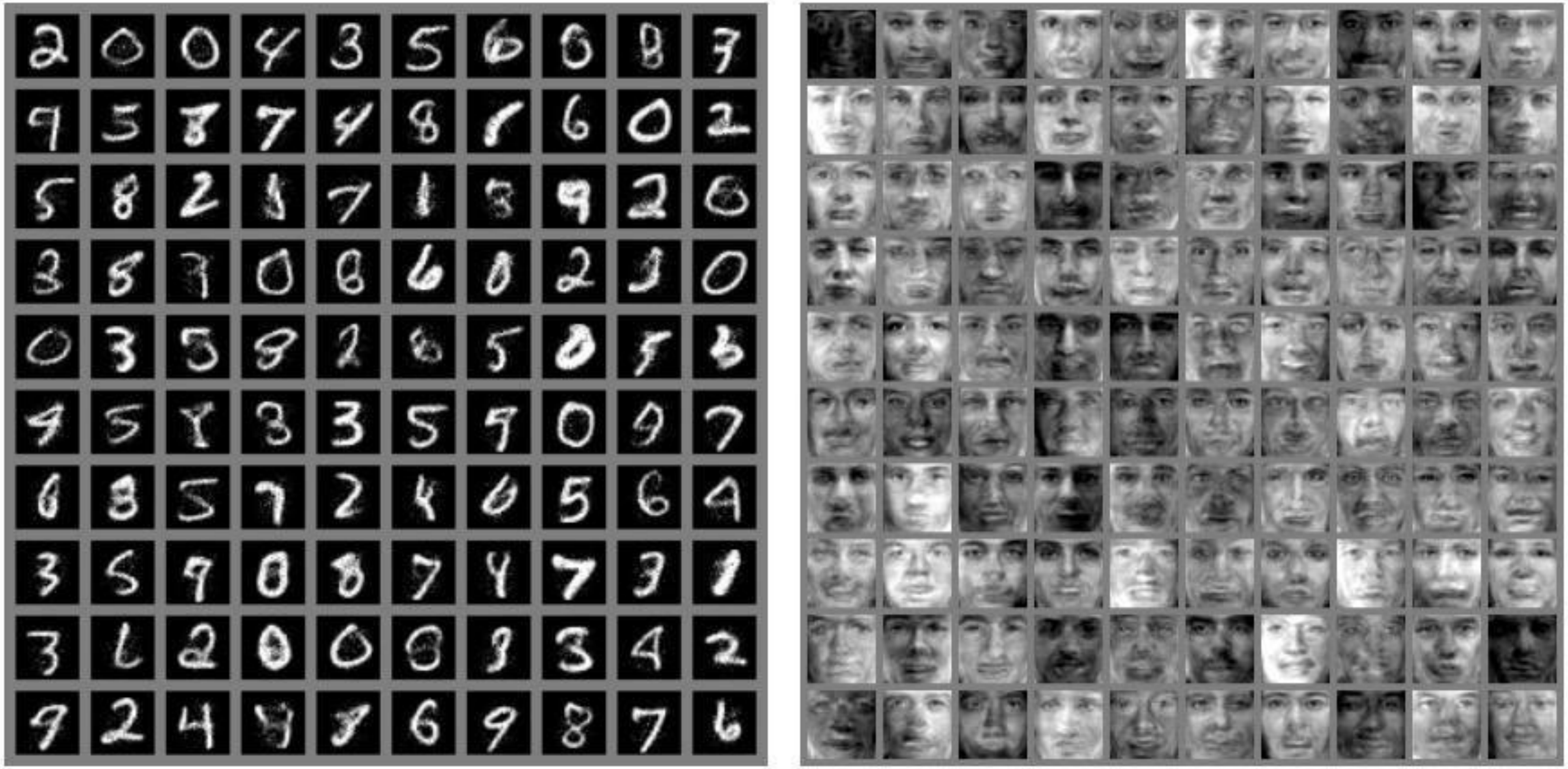
- Inverse:

$$z_i = \frac{x_i}{\beta_i}$$

- Jacobian:

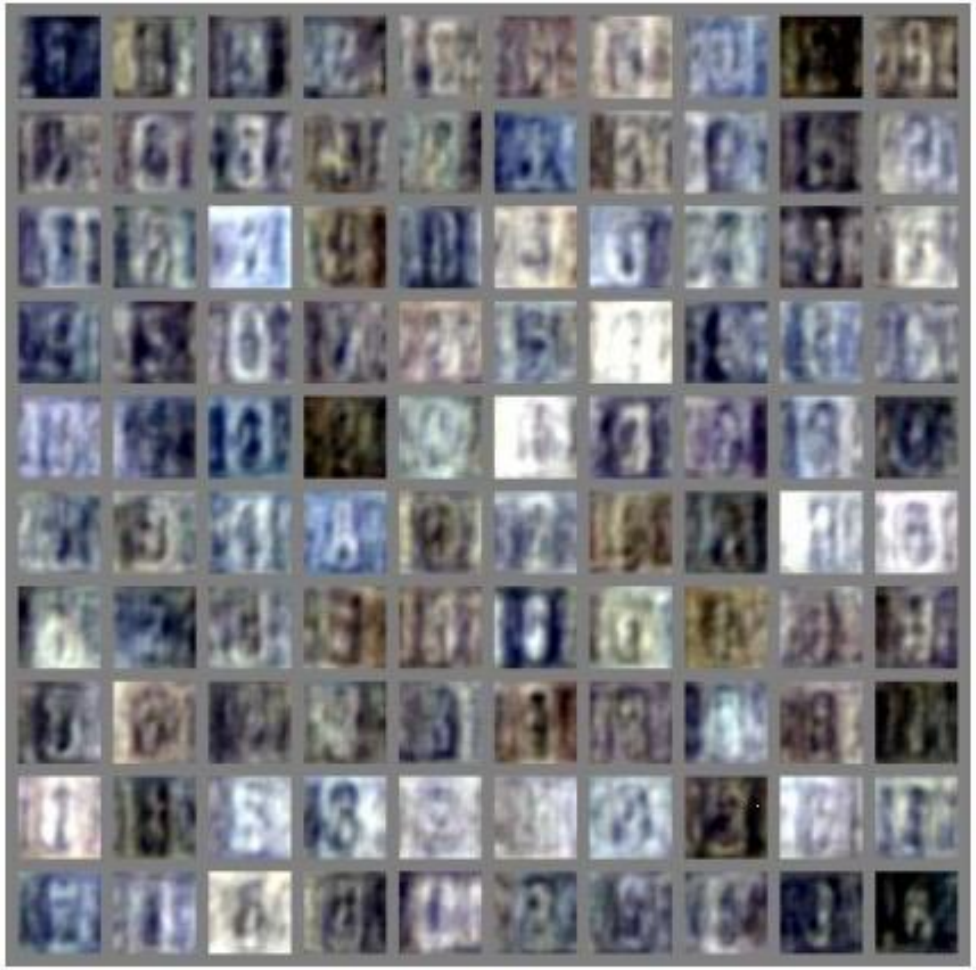
$$J = diag(\beta)$$

Samples generated via NICE



(a) Model trained on MNIST (b) Model trained on TFD

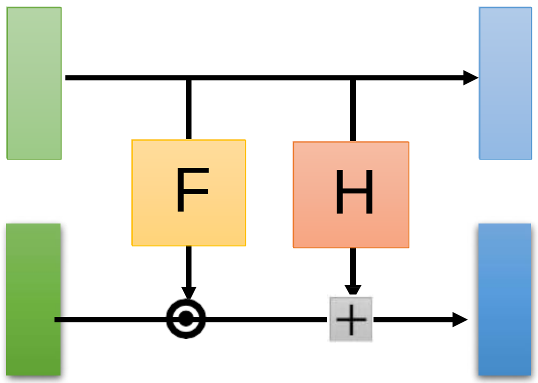
Samples generated via NICE



(c) Model trained on SVHN

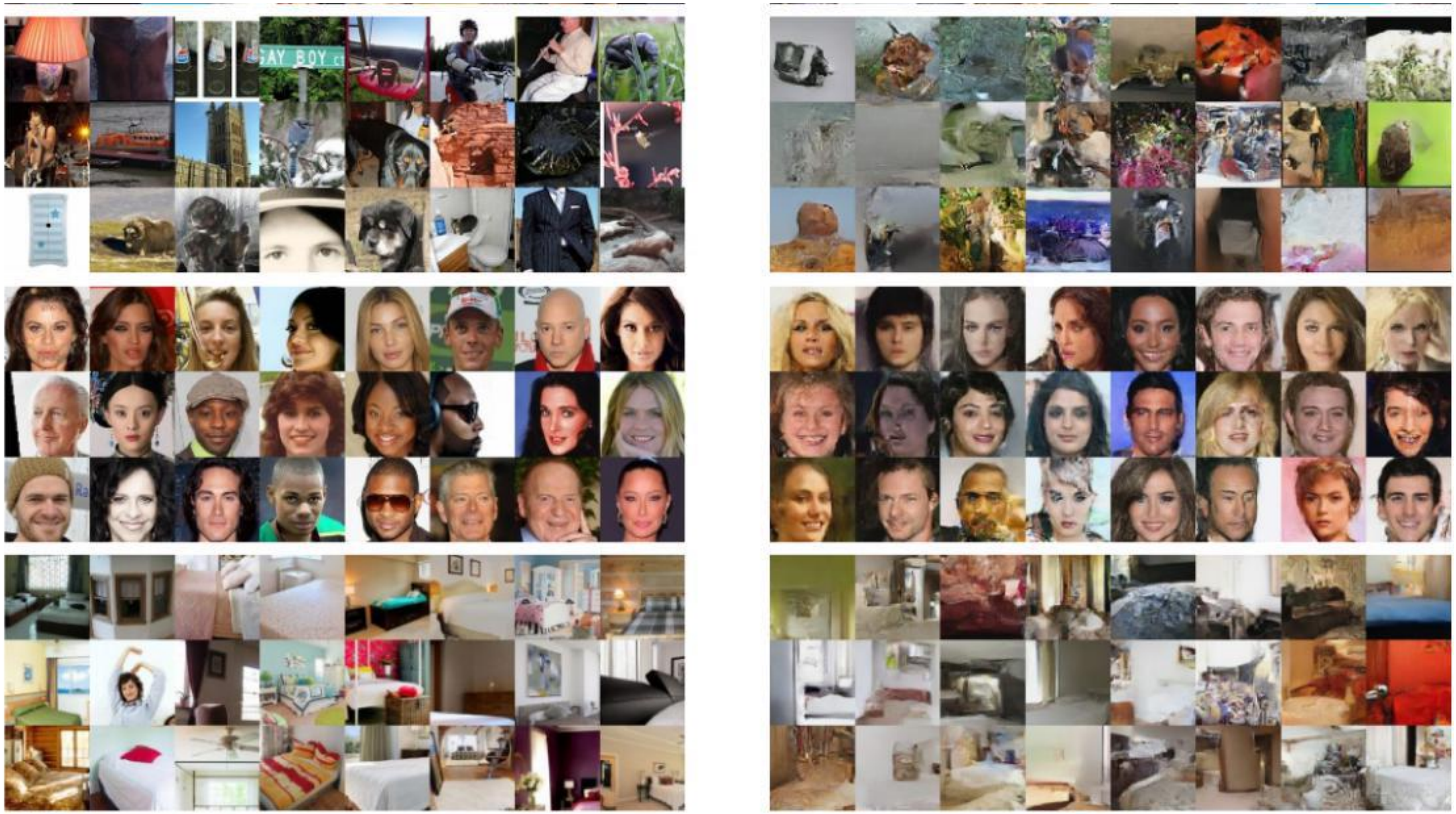
(d) Model trained on CIFAR-10

Real NVP: Real Non-Volume Preserving

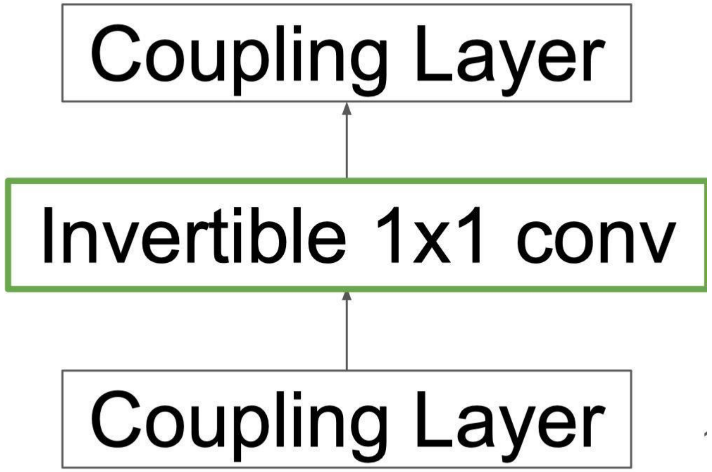


- Coupling layers
  - Partition the variables z into two disjoint subsets
  - $x_{1:d} = z_{1:d}$
  - $x_{d+1:n} = z_{d+1:n} \odot F(z_{1:d}) + H(z_{1:d})$
  - **Non-volume preserving transformation** in general since determinant can be less than or greater than 1
- Coupling layers are composed together (with arbitrary partitions of variables in each layer)

Samples generated via Real NVP



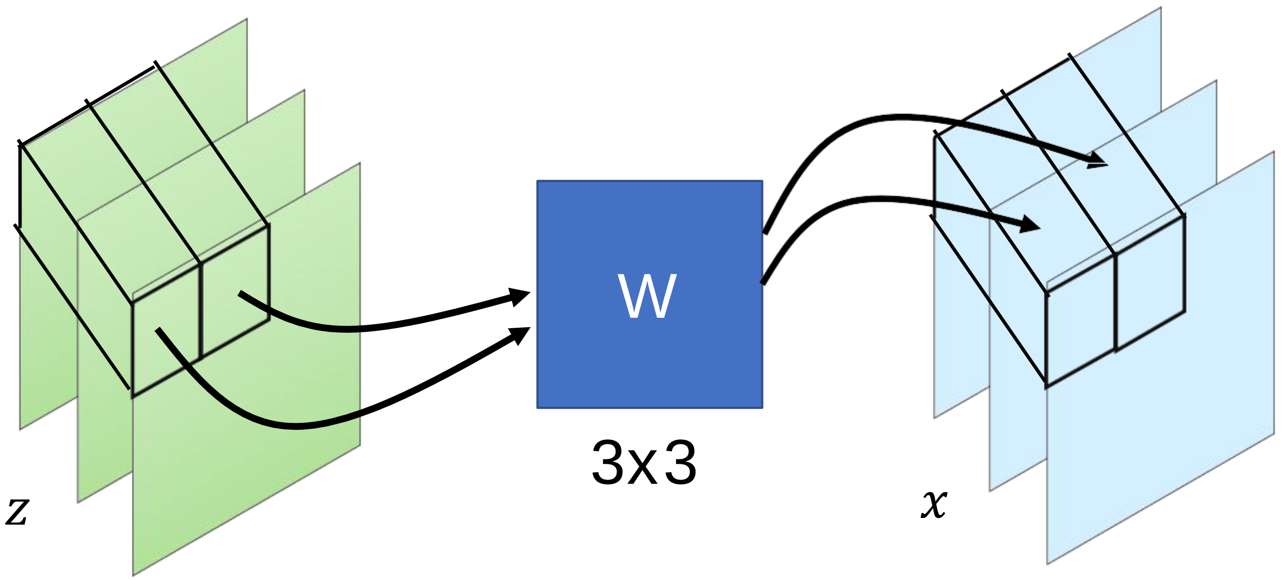
Glow: Generative Flow with Invertible 1×1 Convolutions



Kingma & Dhariwal, 2018

Glow

1x1 Convolution



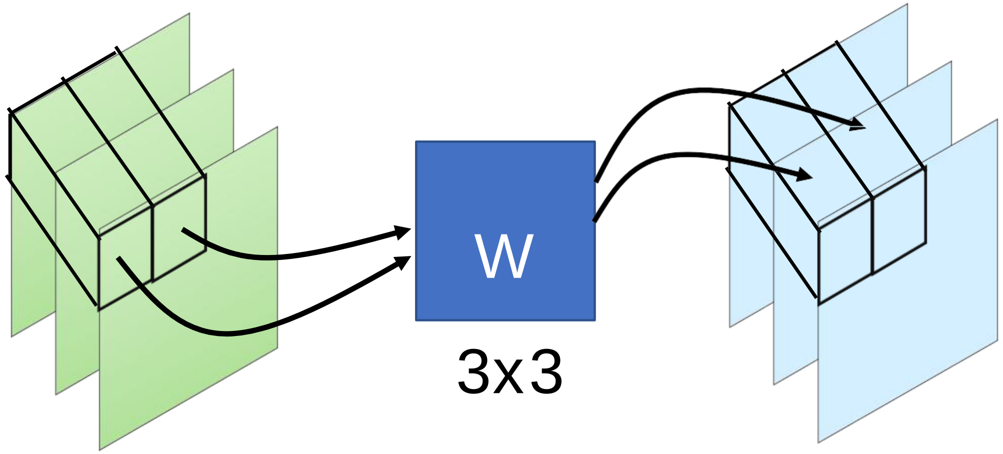
W can shuffle the channels.

If *W* is invertible, it will be easy to compute W^{-1}

$$\begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Glow

1x1 Convolution



$$J = \begin{bmatrix} \partial x_1/\partial z_1 & \partial x_1/\partial z_2 & \partial x_1/\partial z_3 \\ \partial x_2/\partial z_1 & \partial x_2/\partial z_2 & \partial x_2/\partial z_3 \\ \partial x_3/\partial z_1 & \partial x_3/\partial z_2 & \partial x_3/\partial z_3 \end{bmatrix}$$

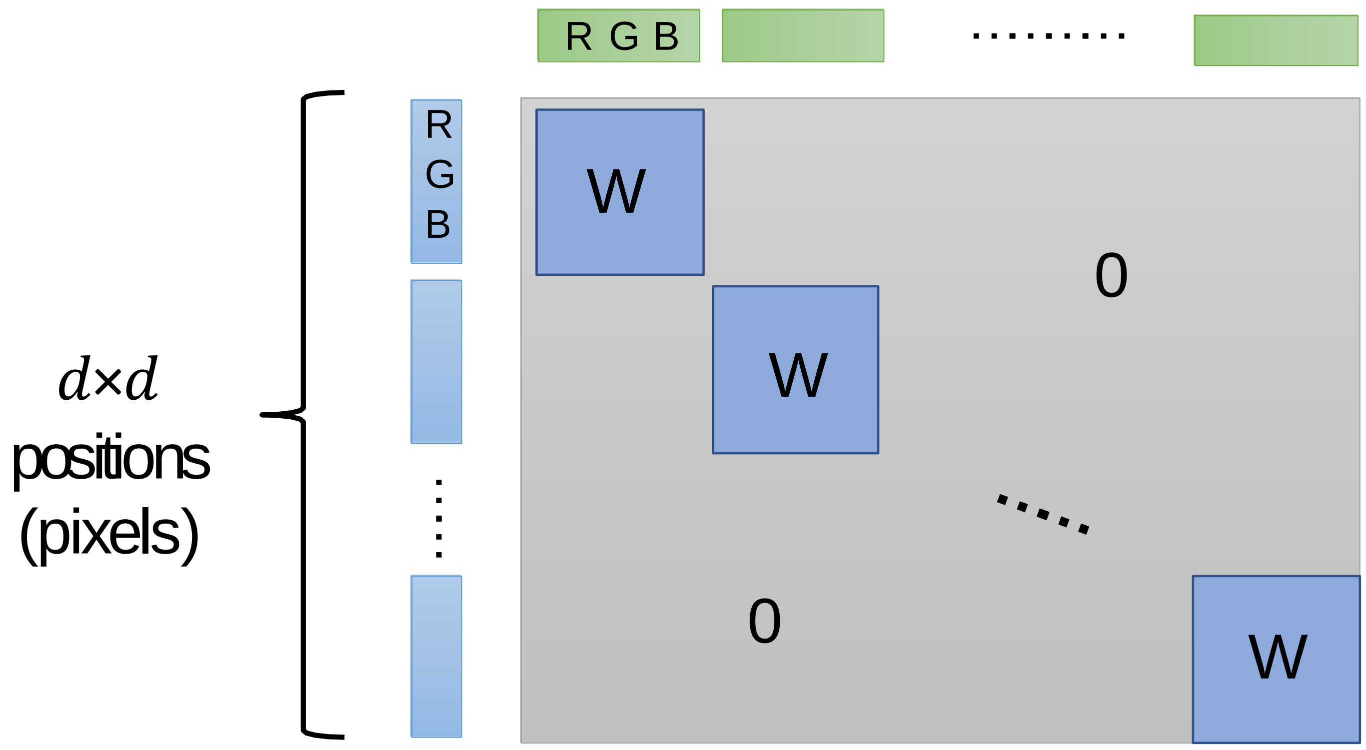
$$= \begin{bmatrix} w_{11} & w_{12} & w_{12} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = W$$

$$x = \boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{W}z$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{12} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}$$

Glow

- $(det(W))^{d*d}$
- If W *is 3*3, computing* $det(W)$ is easy.



Samples generated via Glow

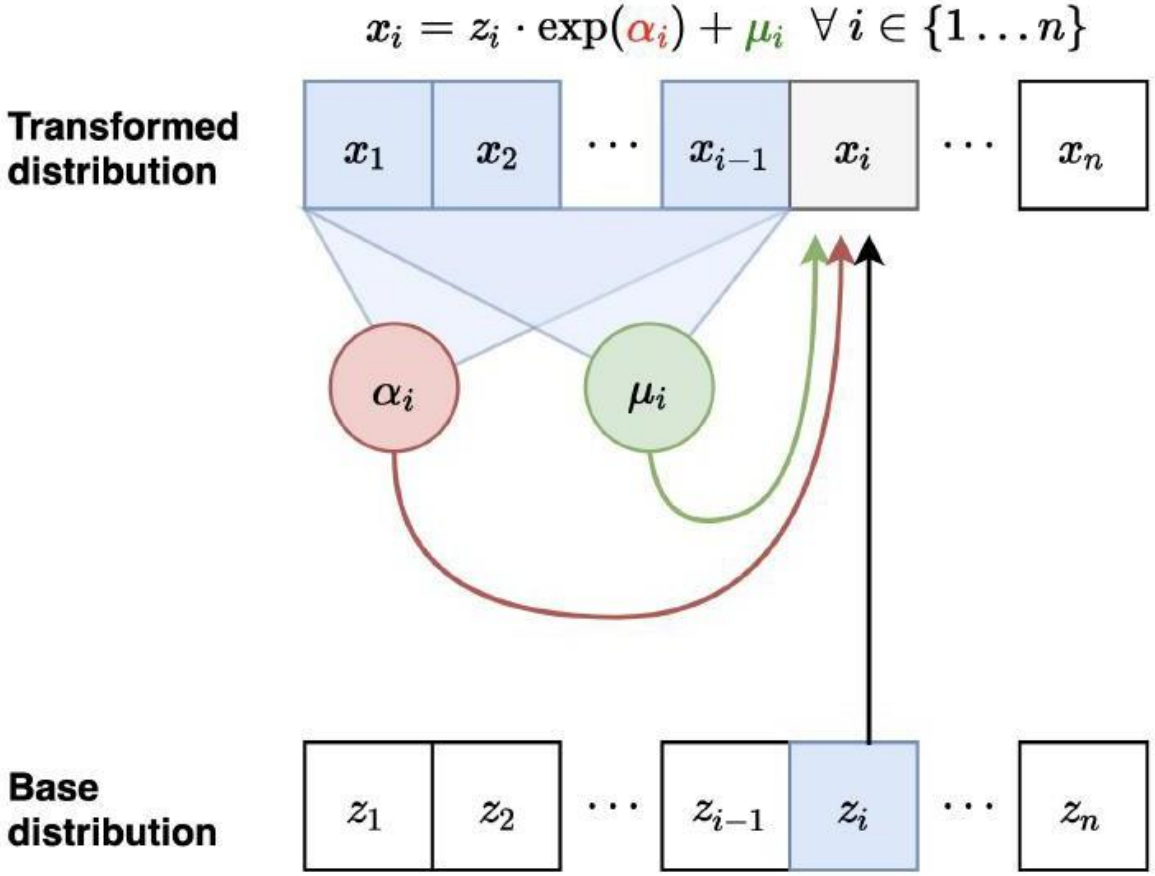
Samples generated via Glow

Figure 5: Linear interpolation in latent space between real images

Autoregressive models as flow models

- Consider a Gaussian autoregressive model:
  - $p(x) = \prod_{i=1}^{D} p(x_i | x_{<i})$
  - Such that $p(x_i | x_{<i}) = N(\mu_i(x_1, \ldots, x_{i-1}), \exp\left(\alpha_i(x_1, \ldots, x_{i-1})\right)^2)$, μ_i, α_i are neural networks.
- Sampler for this model:
  - Sample $z_i \sim N(0,1)$
  - Let $x_i = \exp(\alpha_i)\, z_i + \mu_i$ <-- look like coupling layer
- **Flow interpretation**: transform **z** to **x** via invertible transformation (parameterized by μ_i, α_i)

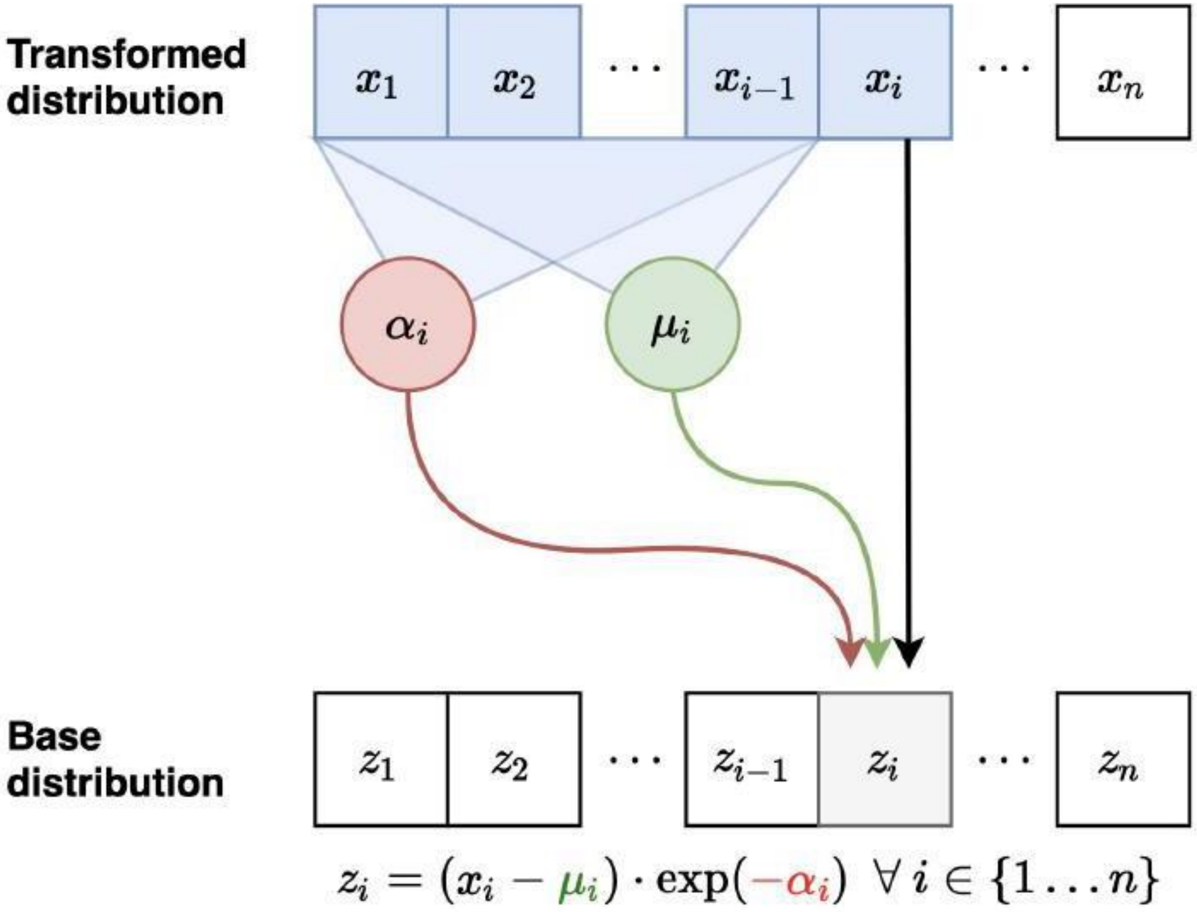
Masked Autoregressive Flow (MAF)



- Forward: (**z** to **x**)
  - $x_i = z_i \exp(\alpha_i) + \mu_i$
  - Compute α_{i+1}, μ_{i+1}
- Sampling is sequential and slow (like autoregressive)

Papamakarios et al., 2017

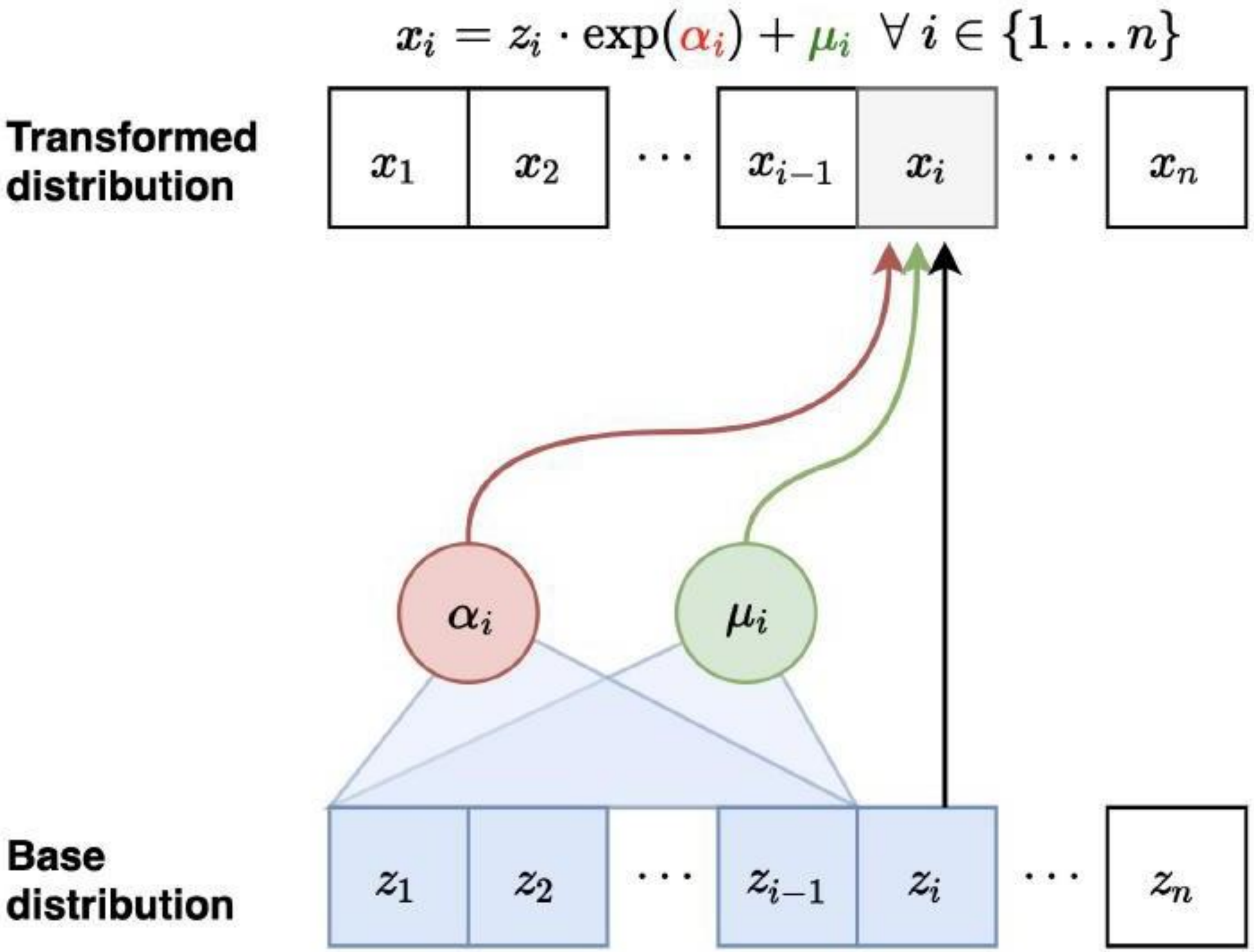
MAF



- Inverse (**x** to **z**)
  - $z_i = (x_i - \mu_i) \exp(-\alpha_i)$
  - can be done in parallel.

- Jacobian is lower diagonal; hence determinant can be computed efficiently

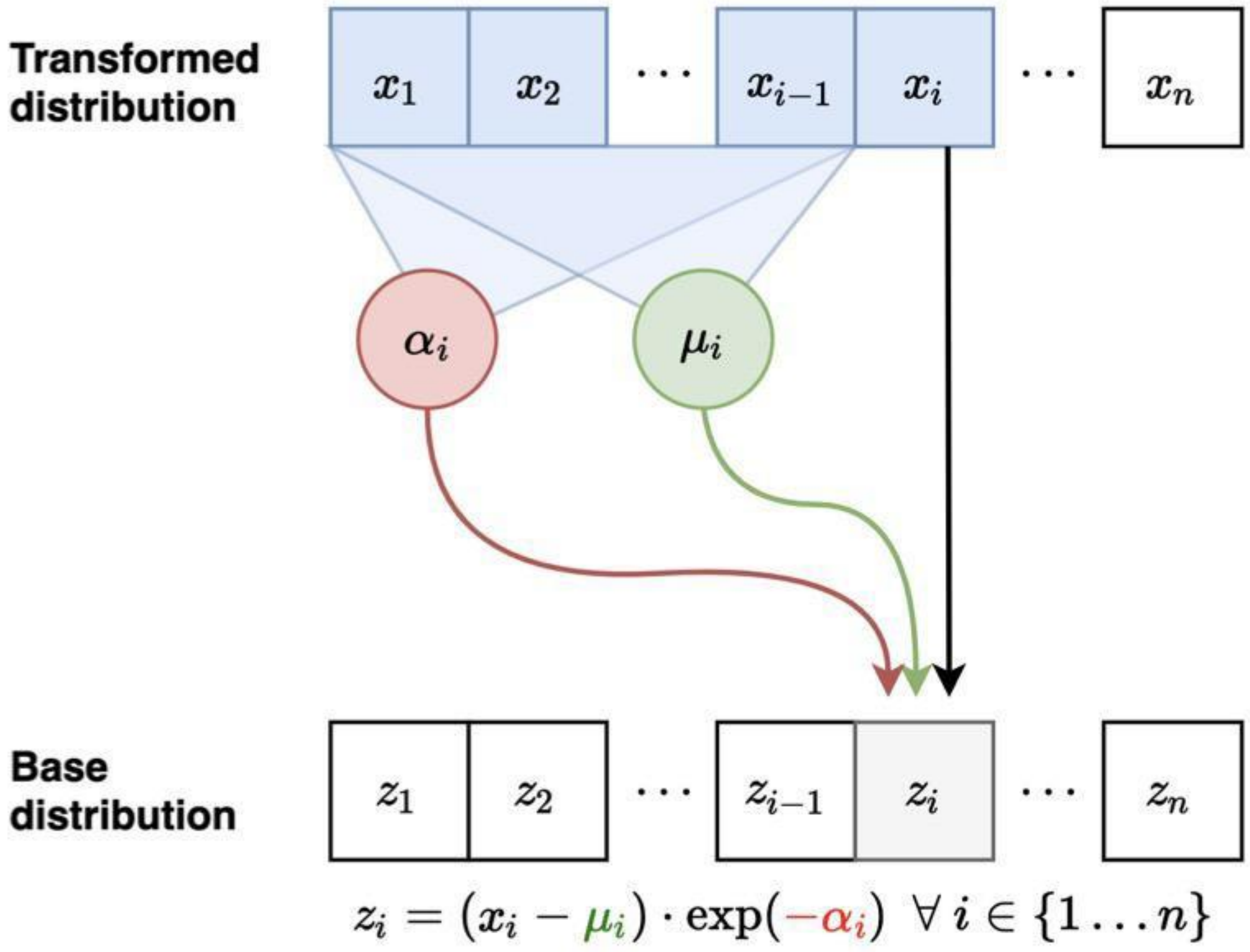
Inverse Autoregressive Flow (IAF)



- **Forward**: (**z** to **x**)
  - $x_i = z_i \exp(\alpha_i) + \mu_i$
  - parallel

Kingma et al., 2016

IAF is inverse of MAF



- Forward: (**z** to **x**)
  - $x_i = z_i \exp(\alpha_i) + \mu_i$
  - parallel

- **Inverse** (**x** to **z**)
  - $z_i = (x_i - \mu_i) \exp(-\alpha_i)$
  - compute α_i, μ_i
  - sequential

IAF vs. MAF

- Computational tradeoffs
  - MAF: Fast likelihood evaluation, slow sampling
  - IAF: Fast sampling, slow likelihood evaluation
- MAF more suited for training based on MLE, density estimation
- IAF more suited for real-time generation

Thank You

- Questions?

- Email: yu.yin@case.edu

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Reference slides

- https://lilianweng.github.io/posts/2018-10-13-flow-models/
- Hao Dong. Deep Generative Models
- Hung-yi Li. Flow-based Generative Model