Introduction to Normalizing Flows

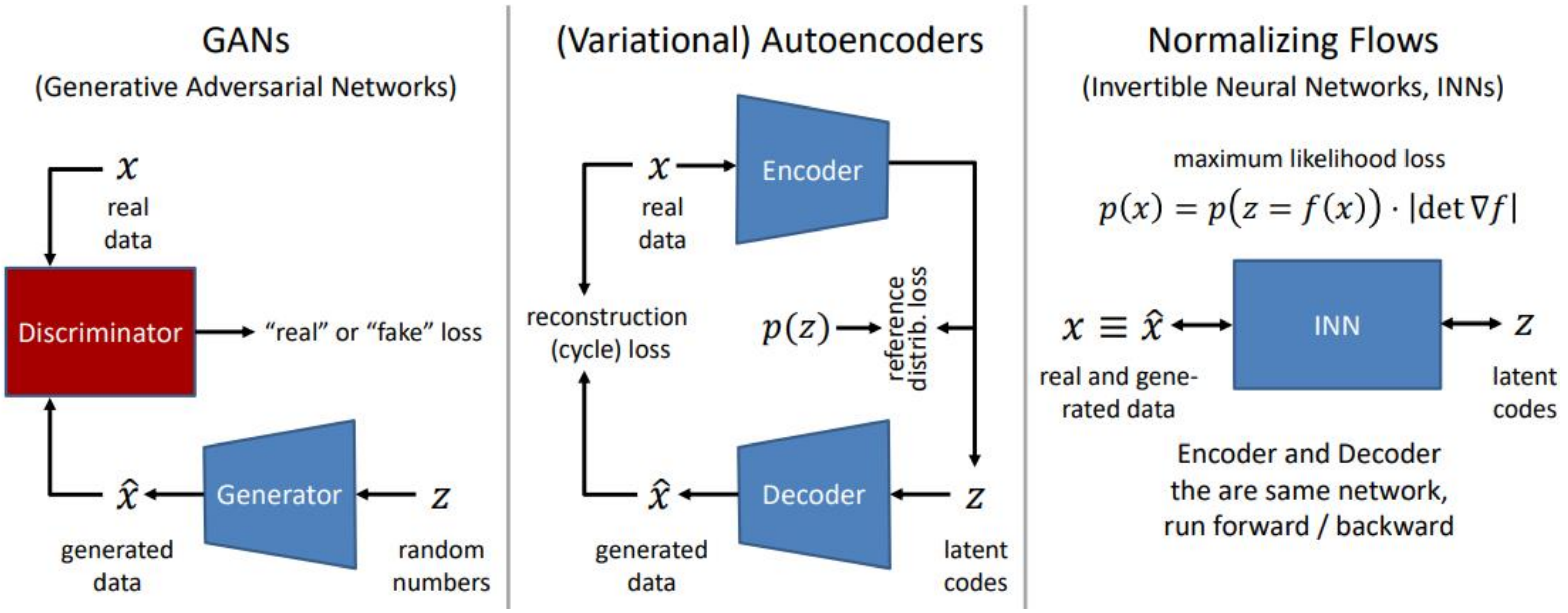
Generative Models

A **generative model** is a probability distribution over a random variable $\mathbf{X}$ which we attempt to learn from a set of observed data $\{\mathbf{x}_i\}_{i=1}^{N}$ with some probability density $p_{\mathbf{X}}(\mathbf{x})$ parameterized by θ

Given a GM we may want to generate samples, evaluate new data points, etc

Different distributions and different learning objectives and approaches lead to different GMs, e.g., GANs, VAEs, NFs etc

Generative Modelling as a Basis for Interpretable Deep Learning



What is Normalizing Flow?

Normalizing Flows are a GM built on invertible transformations

They are generally:

- Efficient to sample from $p_{\mathbf{X}}(\mathbf{x})$
- Efficient to evaluate $p_{\mathbf{X}}(\mathbf{x})$
- Highly expressive
- Useful latent representation
- Straightforward to train

What is Normalizing Flow?

A *normalizing flow* describes the transformation of a probability density through a sequence of invertible mappings. By repeatedly applying the rule for change of variables, the initial density 'flows' through the sequence of invertible mappings. At the end of this sequence we obtain a valid probability distribution and hence this type of flow is referred to as a normalizing flow.

Variational Inference with Normalizing Flows

History of Normalizing Flows

A family of non-parametric density estimation algorithms

E. G. TABAK
Courant Institute of Mathematical Sciences
AND
CRISTINA V. TURNER
FaMAF, Universidad Nacional de Córdoba

[Tabak and Turner, CPAM 2013]

NICE: NON-LINEAR INDEPENDENT COMPONENTS ESTIMATION

Laurent Dinh David Krueger Yoshua Bengio*
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

[Dinh et al, ICLR 2015]

2010 2013 2014 2015

High-Dimensional Probability Estimation with Deep Density Models

Oren Rippel*
Massachusetts Institute of Technology,
Harvard University
rippel@math.mit.edu

Ryan Prescott Adams†
Harvard University
rpa@seas.harvard.edu

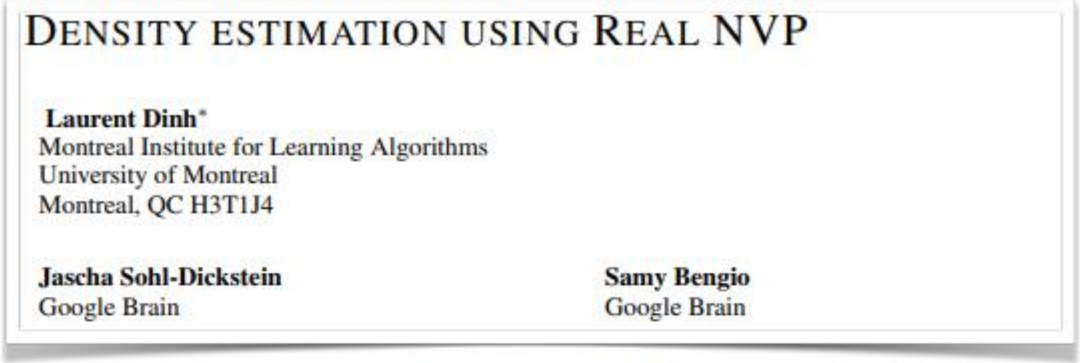
[Rippel and Adams, arXiv 2013]

Variational Inference with Normalizing Flows

Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Shakir Mohamed SHAKIR@GOOGLE.COM
Google DeepMind, London

[Rezende and Mohamed, ICML 2015]

History of Normalizing Flows



DENSITY ESTIMATION USING REAL NVP

Laurent Dinh*
Montreal Institute for Learning Algorithms
University of Montreal
Montreal, QC H3T1J4

Jascha Sohl-Dickstein
Google Brain

Samy Bengio
Google Brain

[Dinh et al, ICLR 2017]

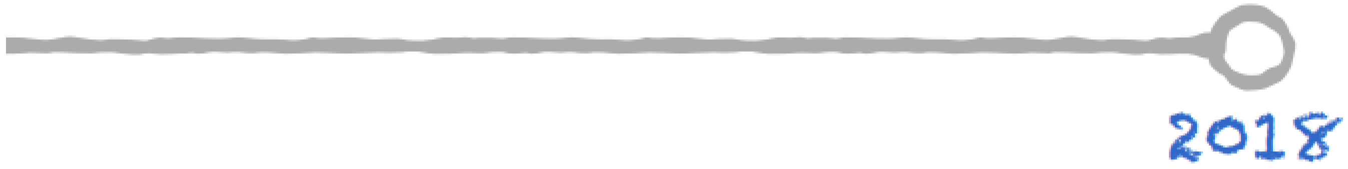
2016

History of Normalizing Flows

Glow: Generative Flow with Invertible 1×1 Convolutions

Diederik P. Kingma*, Prafulla Dhariwal*
OpenAI, San Francisco

[Kingma and Dhariwal, NeurIPS 2018]

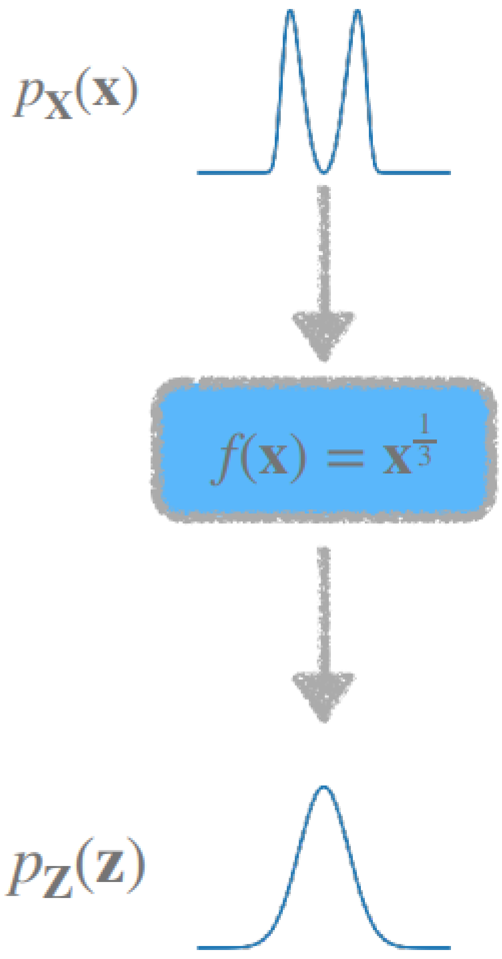


Normalizing Flows

Change of variables

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(\underbrace{f(\mathbf{x})}_{\text{Invertible Transform}}) \overbrace{\left|\det Df(\mathbf{x})\right|}^{\text{Volume Correction}}$$

where $\mathbf{Z} = f(\mathbf{X})$ is an invertible, differentiable function and $Df(\mathbf{x})$ is the Jacobian of $f(\mathbf{x})$



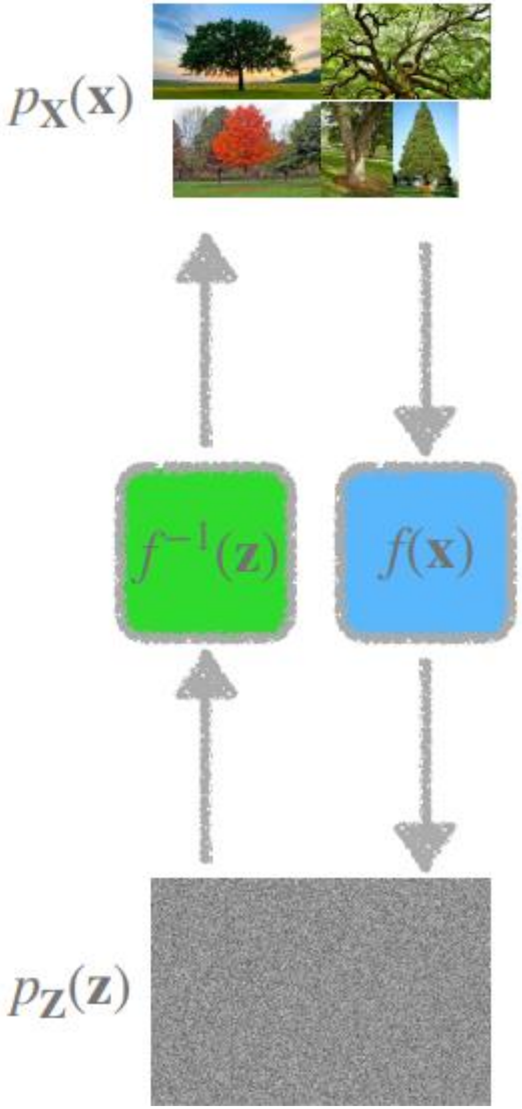
Normalizing Flows

Density evaluation:

$$p_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{Z}}(f(\mathbf{x})) \left| \det Df(\mathbf{x}) \right|$$

Sampling:

- Sample $\mathbf{z} \sim p_{\mathbf{Z}}(\,\cdot\,)$
- Compute $\mathbf{x} = f^{-1}(\mathbf{z})$



Normalizing Flows

Training can be done with maximum (log-)likelihood

$$\max_{\theta} \sum_{i=1}^{N} \log p_{\mathbf{Z}}(f(\mathbf{x}_i \mid \theta)) + \log |\det Df(\mathbf{x}_i \mid \theta)|$$

where θ are the parameters of the flow $f(\mathbf{x} \mid \theta)$

Normalizing Flows

A **flow** is a parametric function $f(\mathbf{x})$ which:

- is invertible
- is differentiable
- has an efficiently computable inverse and Jacobian determinant $|\det Df(\mathbf{x})|$

Also sometimes called a **flow layer**, **bijection**, etc.

Designing and understanding flows is the core technical challenge with NFs

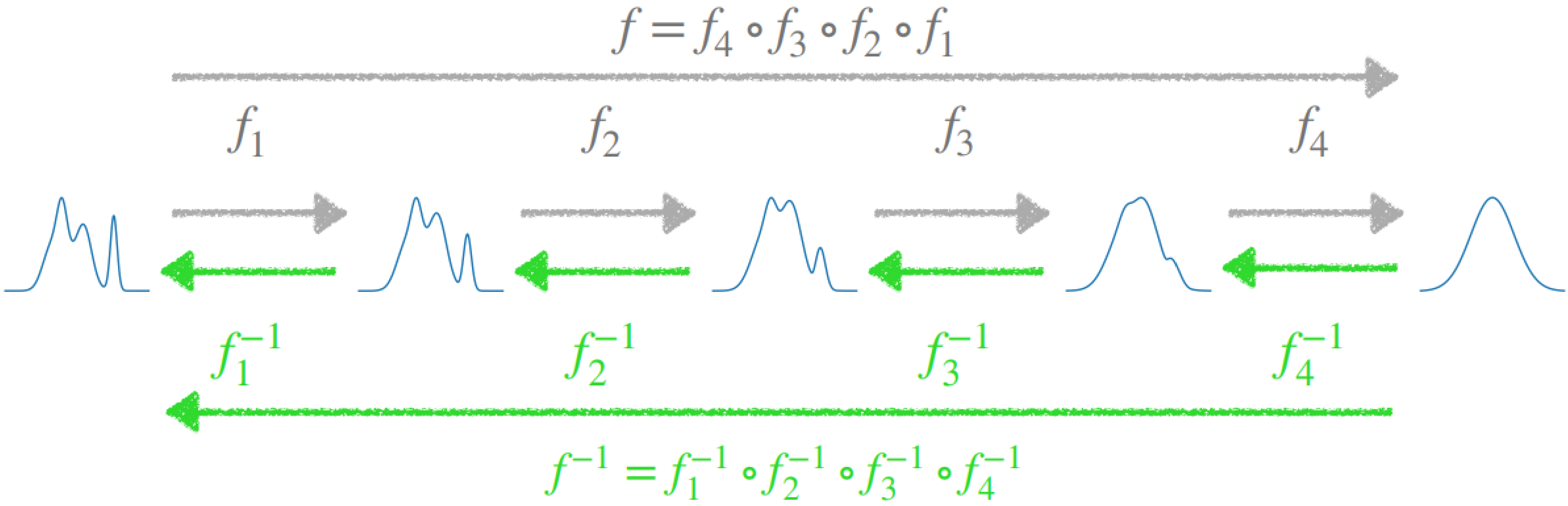
Composition of Flows

Invertible, differentiable functions are closed under composition

$$f = f_K \circ f_{K-1} \circ \cdots \circ f_2 \circ f_1$$

Build up a complex flow from composition of simpler flows

Composition of Flows



Composition of Flows

Determinant:

$$\det Df = \det \prod_{k=1}^{K} Df_k = \prod_{k=1}^{K} \det Df_k$$

Likelihood:

$$\max_{\theta} \sum_{i=1}^{N} \log p_{\mathbf{Z}}(f(\mathbf{x}_i \mid \theta)) + \sum_{k=1}^{K} \log \lvert \det Df_k(\mathbf{x}_i \mid \theta) \rvert$$

Linear Flows

A linear transformation can be a flow if the matrix is invertible

$$f(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$$

Inverse: $f^{-1}(\mathbf{z}) = \mathbf{A}^{-1}(\mathbf{z} - \mathbf{b})$

Determinant: $\det Df(\mathbf{x}) = \det \mathbf{A}$

Problem:

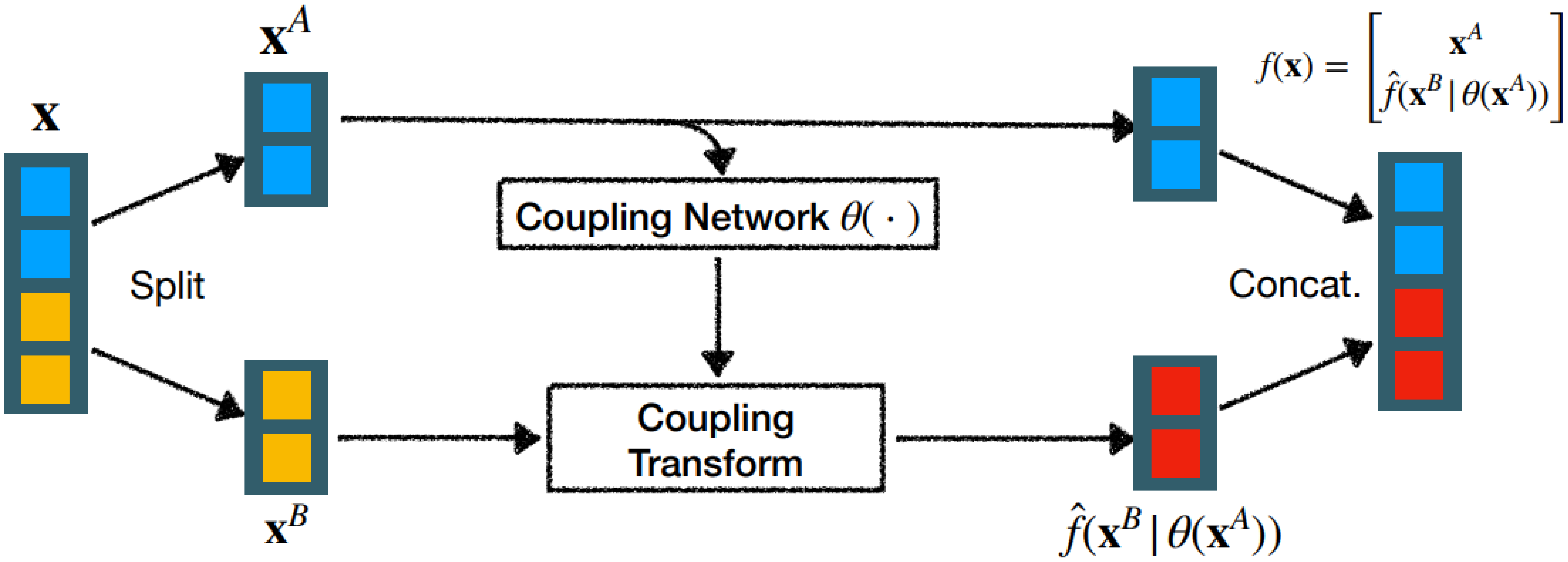
- Inexpressive (linear functions are closed under composition)
- Determinant/inverse could be $O(d^3)$

Linear Flows

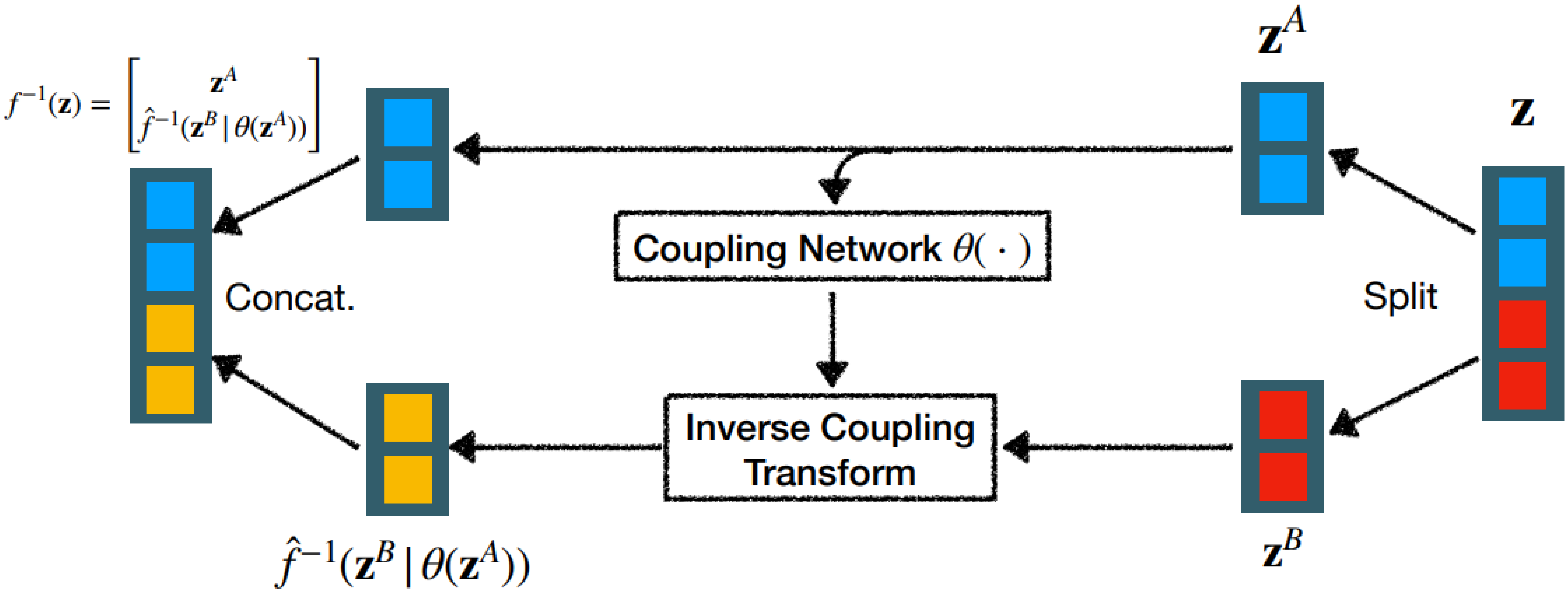
Restricting the form of the matrix can reduce the determinant/inverse costs

	Inverse	Determinant
Full	$O(d^3)$	$O(d^3)$
Diagonal	$O(d)$	$O(d)$
Triangular	$O(d^2)$	$O(d)$
Block Diagonal	$O(c^3 d)$	$O(c^3 d)$
LU Factorized *[Kingma and Dhariwal 2018]*	$O(d^2)$	$O(d)$
Spatial Convolution *[Hoogeboom et al 2019; Karami et al., 2019]*	$O(d \log d)$	$O(d)$
1x1 Convolution *[Kingma and Dhariwal 2018]*	$O(c^3 + c^2 d)$	$O(c^3)$

Coupling Flows



Coupling Flows



Coupling Flows

Jacobian:

$$Df(\mathbf{x}) = \begin{bmatrix} \mathbf{I} & 0 \\ \frac{\partial}{\partial \mathbf{x}^A} \hat{f}(\mathbf{x}^B \,|\, \theta(\mathbf{x}^A)) & D\hat{f}(\mathbf{x}^B \,|\, \theta(\mathbf{x}^A)) \end{bmatrix}$$

Determinant:

$$\det Df(\mathbf{x}) = \det D\hat{f}(\mathbf{x}^B \,|\, \theta(\mathbf{x}^A))$$

Coupling Flows

Coupling Transforms

- Additive [NICE, Dinh et al 2014]

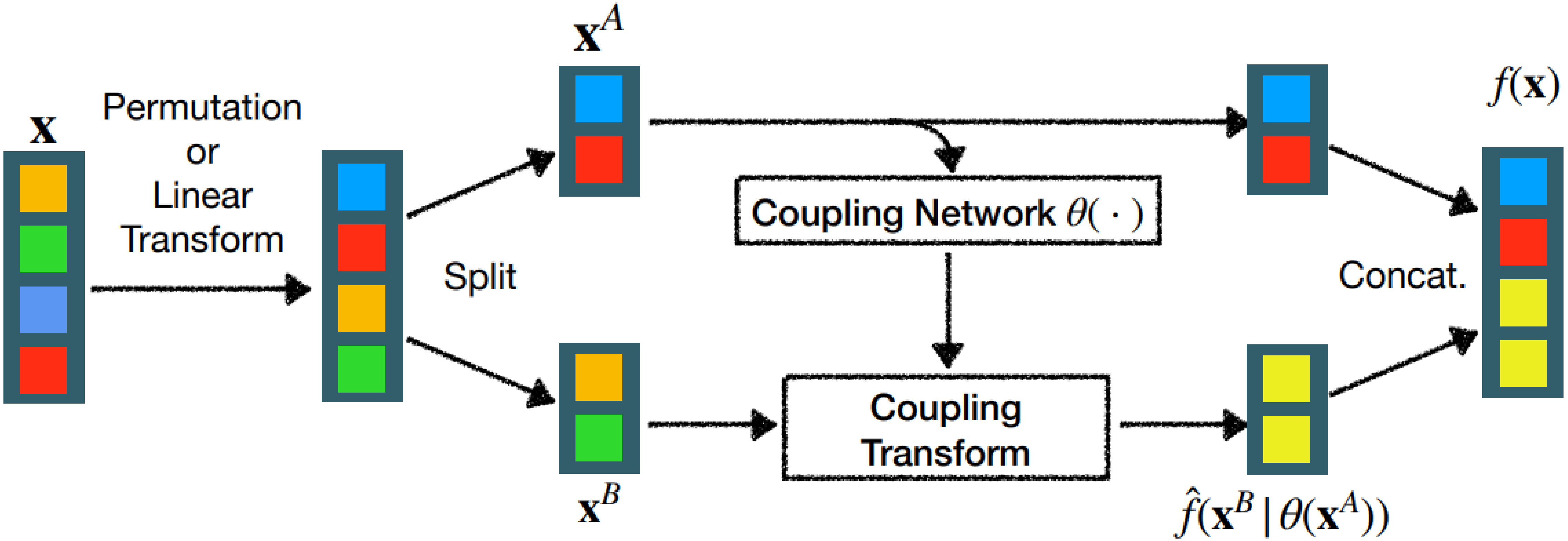
$$\hat{f}(\mathbf{x} \mid \mathbf{t}) = \mathbf{x} + \mathbf{t}$$

- Affine [RealNVP, Dinh et al 2016]

$$\hat{f}(\mathbf{x} \mid \mathbf{s}, \mathbf{t}) = \mathbf{s} \odot \mathbf{x} + \mathbf{t}$$

- MLPs [NAF, Huang et al, 2018], MixLogCDF [Flow++, Ho et al, 2019], Splines [Spline Flow, Durkan et al, 2019], etc…

Coupling Flows



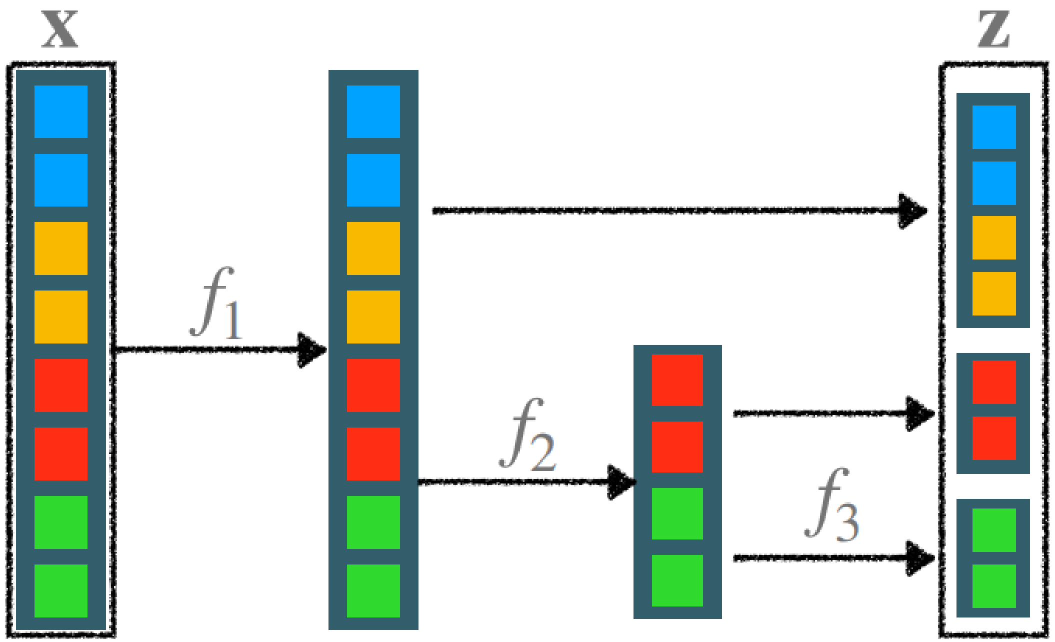
Multi-Scale Flows

A flow preserves dimensionality, but this is expensive in high dimensions

Just stop using subsets of dimensions

Practically, acts like dropping dimensions

Multi-Scale Flows

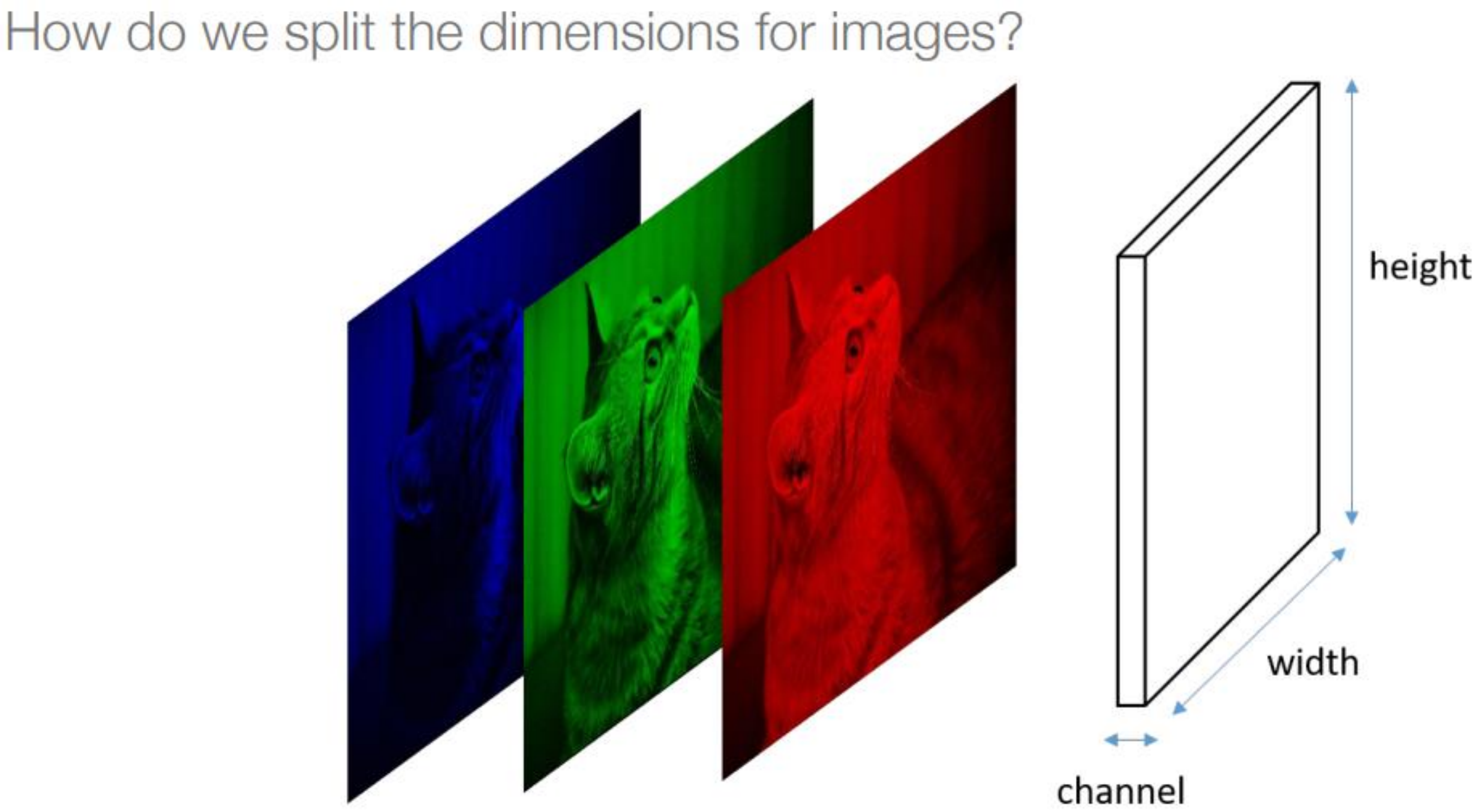


Multi-scale flows are just a special coupling flow

$$f(\mathbf{x}) = (\mathbf{x}^A, \hat{f}(\mathbf{x}^B \,|\, \theta))$$

- Important: must track "dropped" dimensions to preserve invertibility

Multi-Scale Flows



Multi-Scale Flows

"Squeeze" the spatial arrangement to get more channels

Quantization

Normalizing Flows are a model of continuous data

Pixel intensities are typically discrete or **quantized**

Quantization

ML learning of continuous models w/ discrete data can cause singularities

Really want to optimize

Probability Density of
Continuous Values

$$P_{\mathbf{Y}}(\mathbf{y}) = \int_{[0,1]^D} p_{\mathbf{X}}(\mathbf{y} + \mathbf{u}) p_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}$$

Probability of
Discrete Values

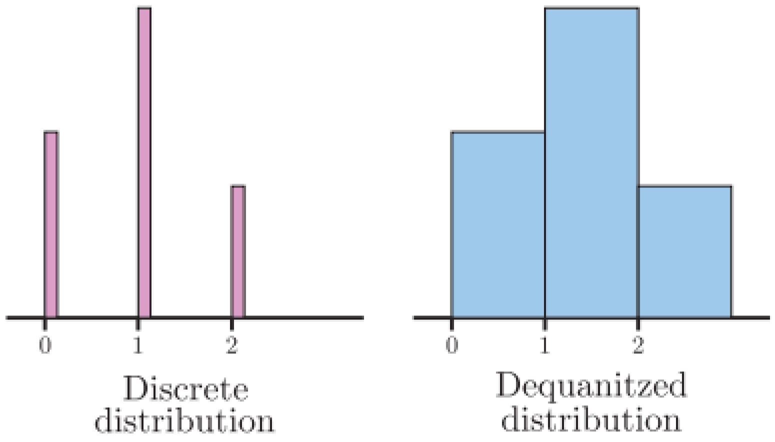
Probability Density of
Quantization Noise

Quantization

During training, **dequantize** the data (i.e., add noise)

$$P_{\mathbf{Y}}(\mathbf{y}) = \int_{[0,1]^D} p_{\mathbf{X}}(\mathbf{y}+\mathbf{u}) p_{\mathbf{U}}(\mathbf{u}) d\mathbf{u}$$

$$\approx \frac{1}{K} \sum_{k=1}^{K} p_{\mathbf{X}}(\mathbf{y}+\mathbf{u}_k)$$



Simplest choice of $p_{\mathbf{U}}$ is uniform

Common Flow Architectures for Images

	Transformations	Dequantization	Multi-Scale
NICE [Dinh et al, 2014]	Additive Coupling + Diagonal Linear	Uniform	No
RealNVP [Dinh et al, 2016]	Affine Coupling + Channelwise Permutation	Uniform	Yes
Glow [Kingma and Dhariwal, 2018]	Affine Coupling + Channelwise Linear	Uniform	Yes
Flow++ [Ho et al, 2019]	MixLogCDF Coupling + Channelwise Linear	Variational	Yes

Multiple Possibilities for Normalizing Flows

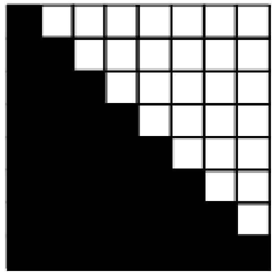
Autoregressive Models

Chain rule decomposition:

$$p(x_1, \dots, x_D) = \prod_i p_i(x_i \mid x_{<i})$$

triangular reparameterization:

$\forall i: \quad x_i = f_i(z_i, x_{<i})$ monoton.

$x \equiv \hat{x} \longleftrightarrow$  $\longleftrightarrow z$

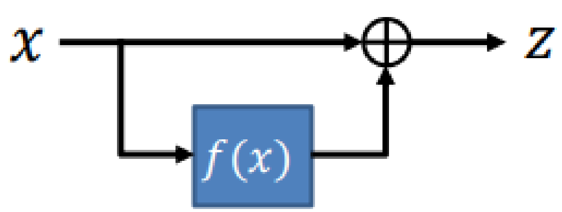
inverse direction inefficient

⇨ use two complementary nets

example: parallel WaveNet

iResNets
(invertible residual networks)

Residual block:



$$z = x + f(x)$$

is invertible when

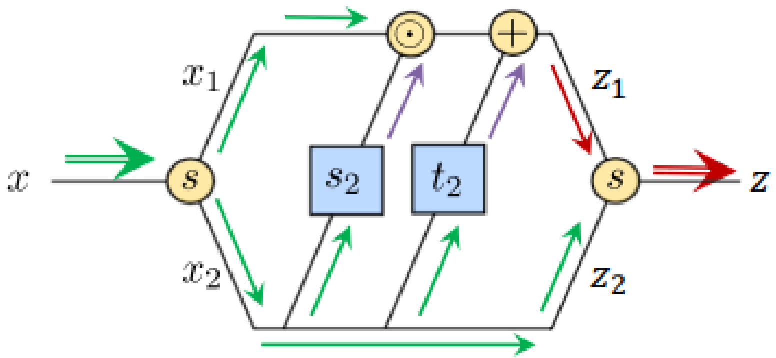
$$\|f(x)\|_{\text{Lipshitz}} < 1$$

inverse direction is reasonably efficient (fixpoint or Newton iterations)

example: Residual Flow Net

RealNVP

Affine coupling layer:



$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} x_1 \cdot s_2(x_2) + t_2(x_2) \\ x_2 \end{bmatrix}$$

inverse is equally efficient:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} (z_1 - t_2(z_2))/s(z_2) \\ z_2 \end{bmatrix}$$

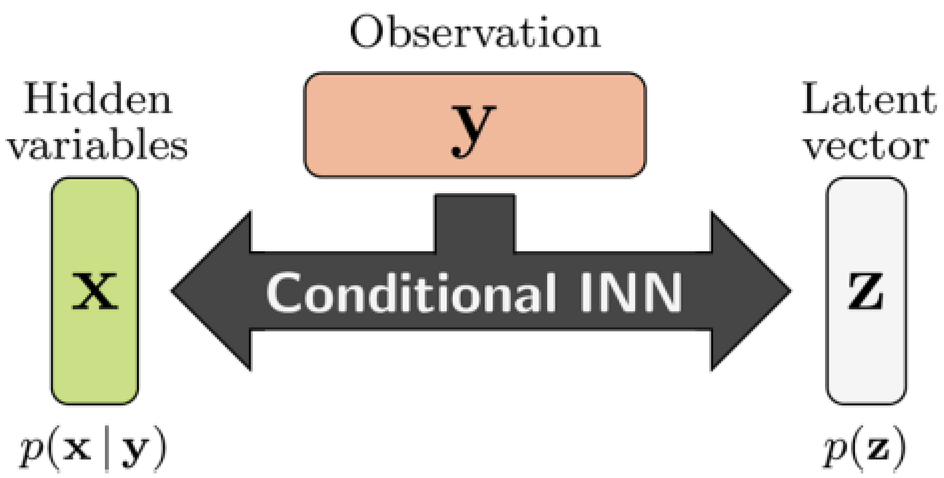
example: GLOW

Conditional INN

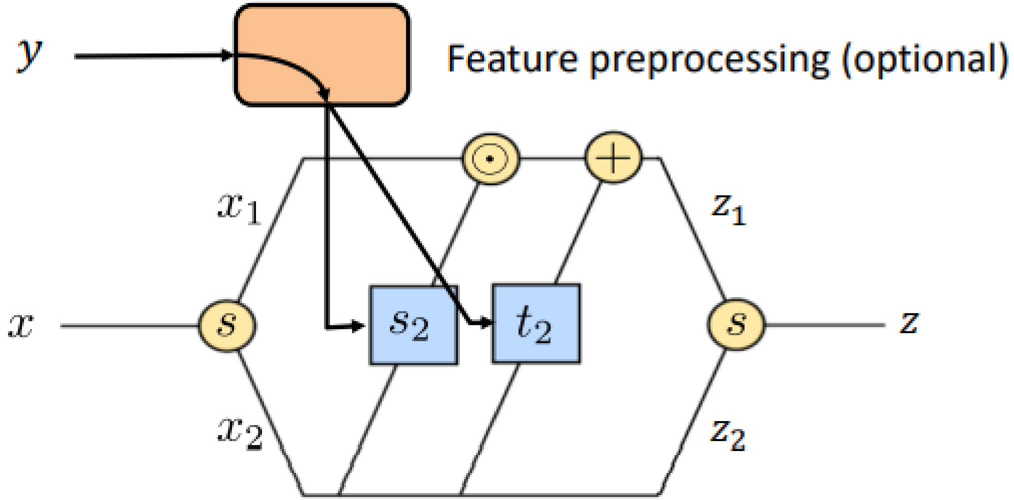
Conditional INN (cINN) adapts vanilla INN for conditional probabilities

- Reparametrize $x \sim p(x \mid y)$ as $x = g_\theta(z; y)$ with $z \sim p_Z(z)$ and forward process $z = f_\theta(x; y) = g_\theta^{-1}(x; y)$
- Minimum log-likelihood loss becomes

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{N} \left(\frac{1}{2} \| f_\theta(x^{(i)}; y^{(i)}) \|_2^2 - \sum_l \text{sum}\left(\log s_{\theta,l}\left(x_{l2}^{(i)};\ y^{(i)} \right) \right) \right)$$

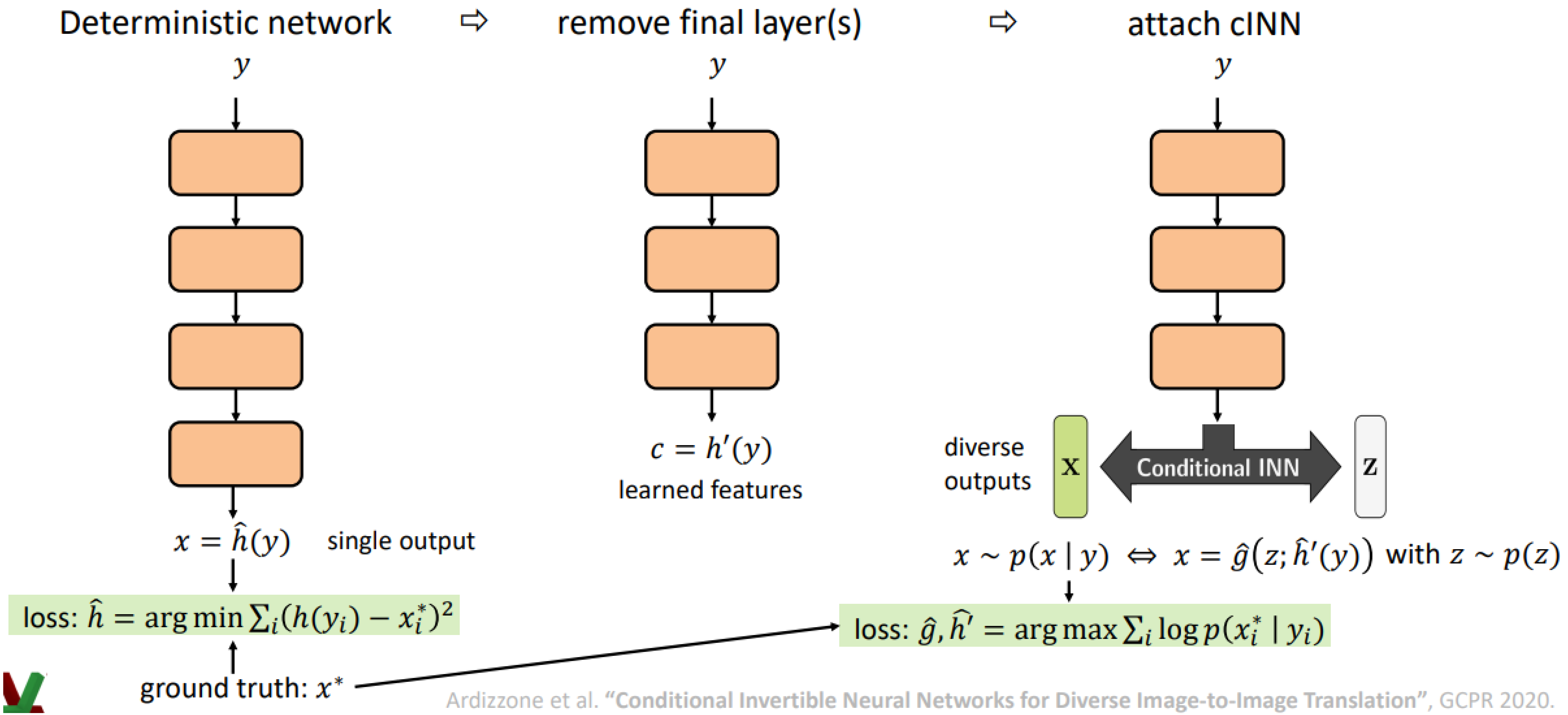


simple change of coupling layer architecture:
feed y as additional input to subnets s, t



Ardizzone et al. **"Conditional Invertible Neural Networks for Diverse Image-to-Image Translation"**, GCPR 2020.

Conditional INN



Ardizzone et al. **"Conditional Invertible Neural Networks for Diverse Image-to-Image Translation"**, GCPR 2020.

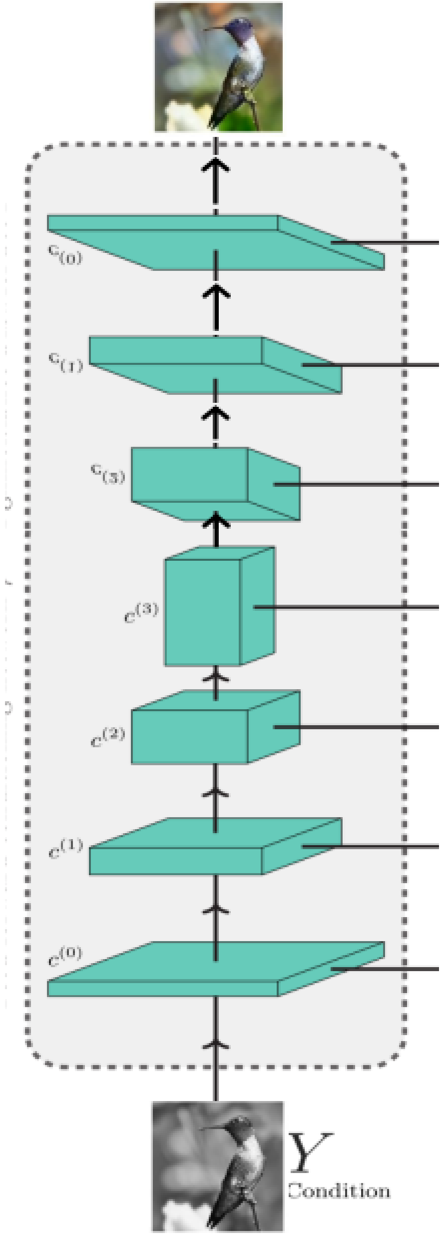
Conditional INN

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

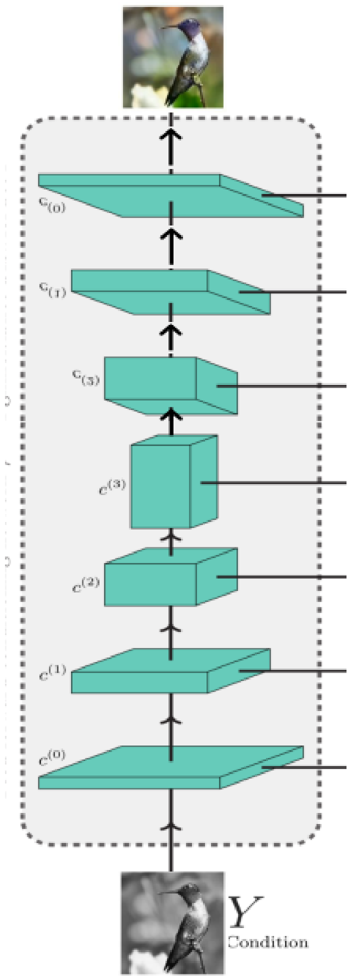
$[y, \hat{x}]$

  - deterministic network: single result

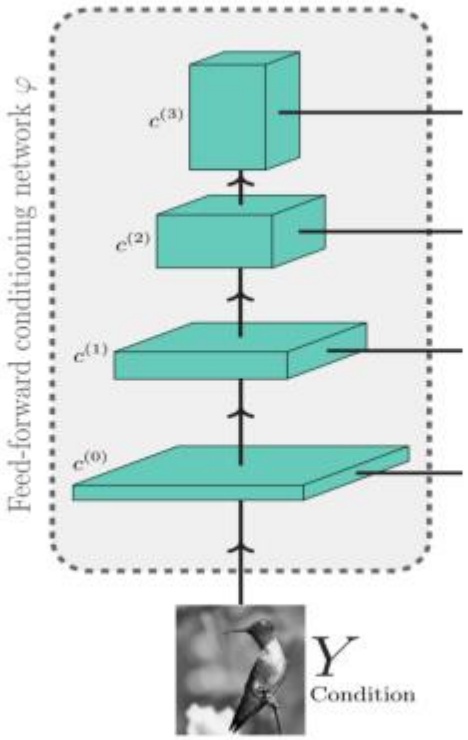


deterministic
U-net

Conditional INN



deterministic
U-net

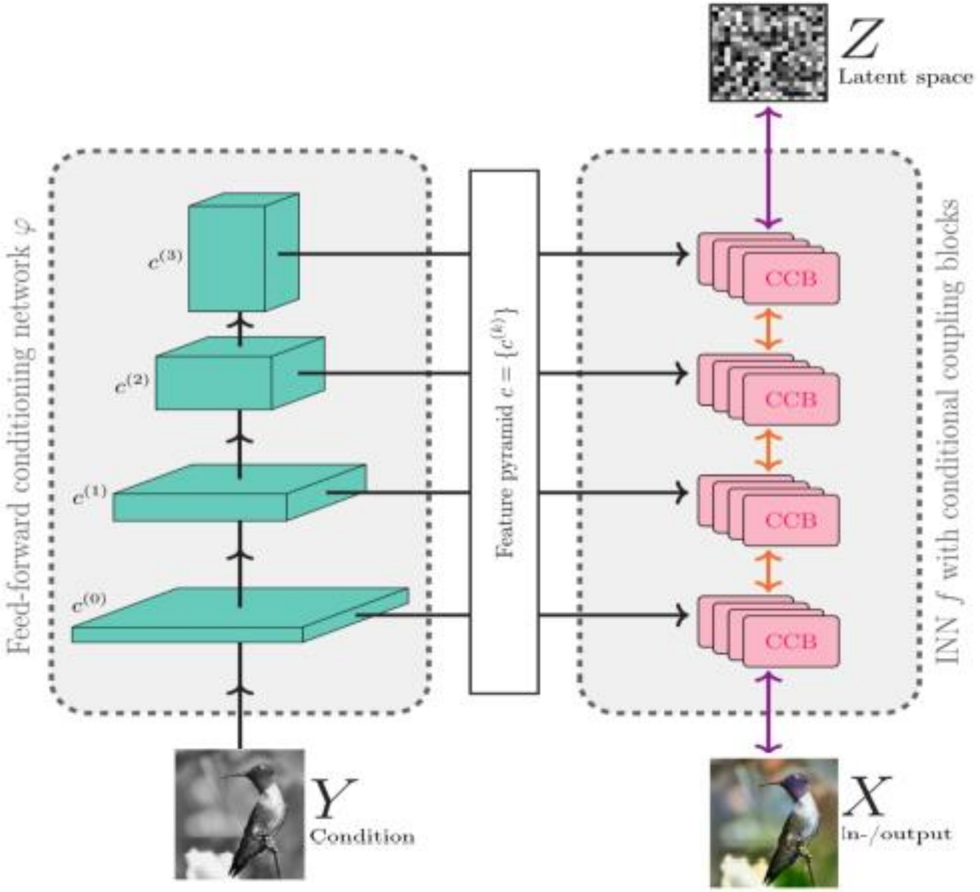


Conditional INN

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

$p(x|y)$



  - cINN: diverse results

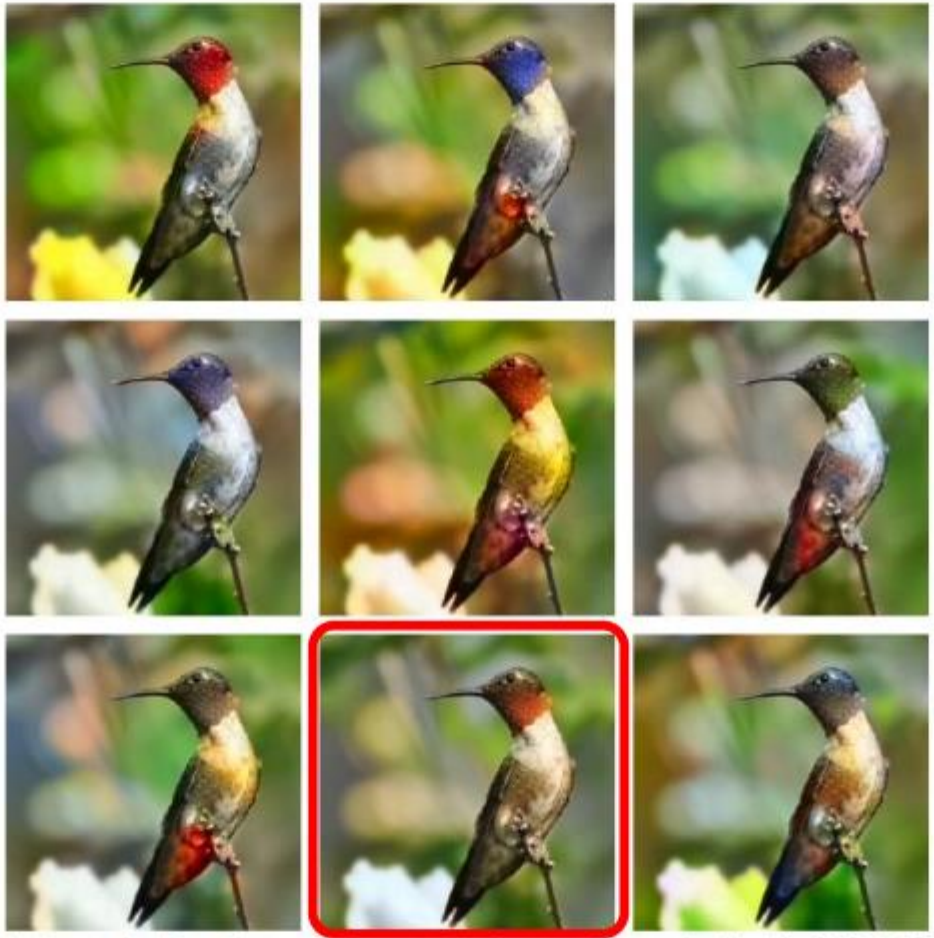
Conditional INN

- Colorization as an inverse problem:
  - forward process: turn color image to grayscale by taking the L-channel in Lab color space
  - inverse problem: reconstruct **realistic** color channels
    $y = L \quad \Rightarrow \quad \hat{x} = [a, b]$

y

$z \sim p_z(z)$

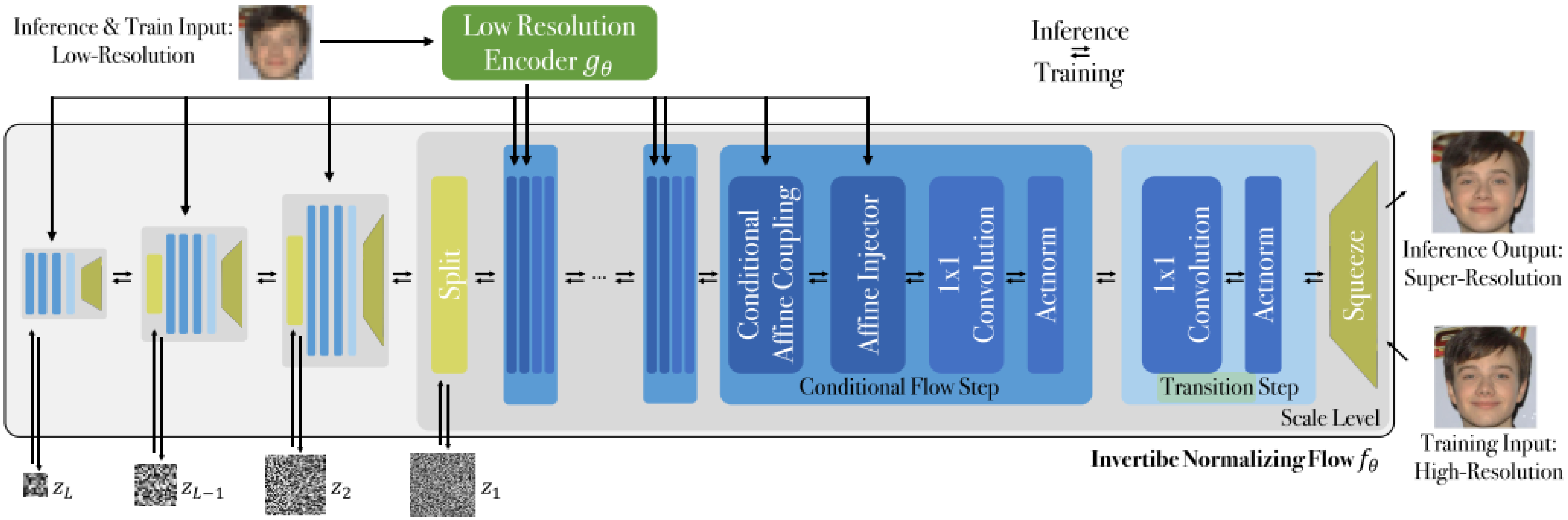
$x \sim \hat{p}(x \mid y)$



  - cINN: diverse results
  - Quiz: Which color image is the ground-truth?

Ardizzone et al. **"Conditional Invertible Neural Networks for Diverse Image-to-Image Translation"**, GCPR 2020.

SRFlow



Inference & Train Input:
Low-Resolution
Low Resolution
Encoder g_θ
Inference
Training
Split
Conditional
Affine Coupling
Affine Injector
1x1
Convolution
Actnorm
Conditional Flow Step
1x1
Convolution
Actnorm
Transition Step
Squeeze
Scale Level
Inference Output:
Super-Resolution
Training Input:
High-Resolution
Invertibe Normalizing Flow f_θ
z_L
z_{L-1}
z_2
z_1

Reference

- Introduction - CVPR2021 (mbrubake.github.io)
- Deep Learning and Artificial Intelligence in Biomedical Research (mbrubake.github.io)