CSCI 5922 - NEURAL NETWORKS AND DEEP LEARNING

AUTOREGRESSION AND NORMALIZING FLOWS

MIDTERM

- Midterm
  - Scoring is underway - aiming for end of this week.
  - Make-up exam is this week. Contact Amruta for logistics.
    - Exam review will be after make-up.
- Project proposal feedback is mostly done.
  - Some tolerance for risky proposals
  - Some **hard** constraints, many about baselines
  - You do not need to show the deep learning approach is better!

ASSIGNMENT 7: CONVNETS

- Task is to implement the forward and backward passes of a simple convolutional network.
- The use of the Toeplitz matrix is not allowed. The Toeplitz matrix allows convolution to be computed as matrix multiplication. See Section 4.1of A Guide to Convolution Arithmetic for Deep Learning.
- The backward pass requires the transposed convolution (also see the convolution arithmetic guide).
- The whole process can be computationally expensive. The use of an acceleration library like Numba is recommended.

- Modeling High-Dimensional Discrete Data with Multi-Layer Neural Networks, Bengio and Bengio, 1999 [link]
- Masked Autoencoder for Distribution Estimation (MADE), Germain et al, 2015 [link]
- Improved Variational Inference with Inverse Autoregressive Flow, Kingma et al, 2016 [link]

BENGIO & BENGIO, 2000

MODELING HIGH-DIMENSIONAL DISCRETE DATA WITH NEURAL NETWORKS

In a graphical model or Bayesian network, the joint probability of the random variables in the model is the product of the probability of each variable conditioned on its parents.

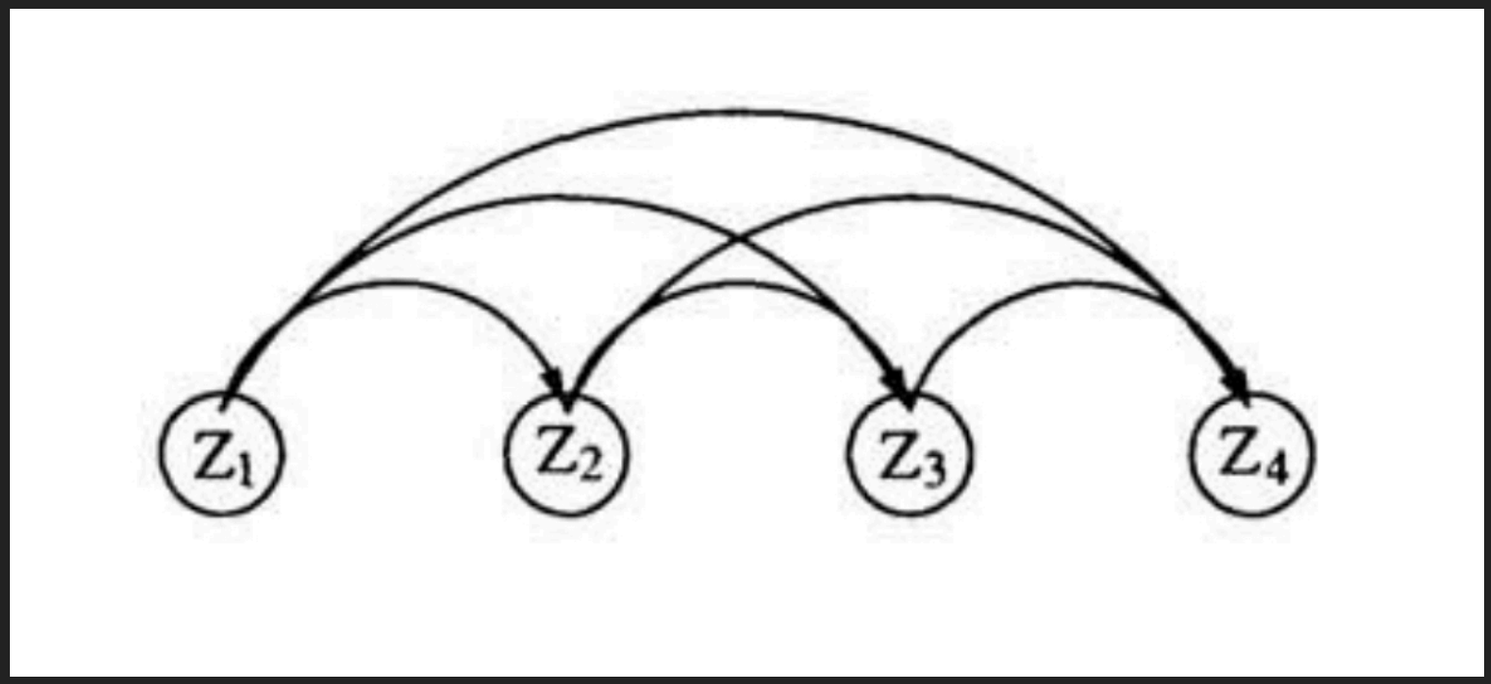
$$P(Z_1, \ldots, Z_n) = \prod_{i=1}^{n} P(Z_i \mid Parents_i)$$

BENGIO & BENGIO, 2000

MODELING HIGH-DIMENSIONAL DISCRETE DATA WITH NEURAL NETWORKS

$$P(Z_1, \ldots, Z_n) = \prod_{i=1}^{n} P(Z_i \mid Z_1, \ldots, Z_{i-1})$$

Fully-connected "left-to-right" graphical model

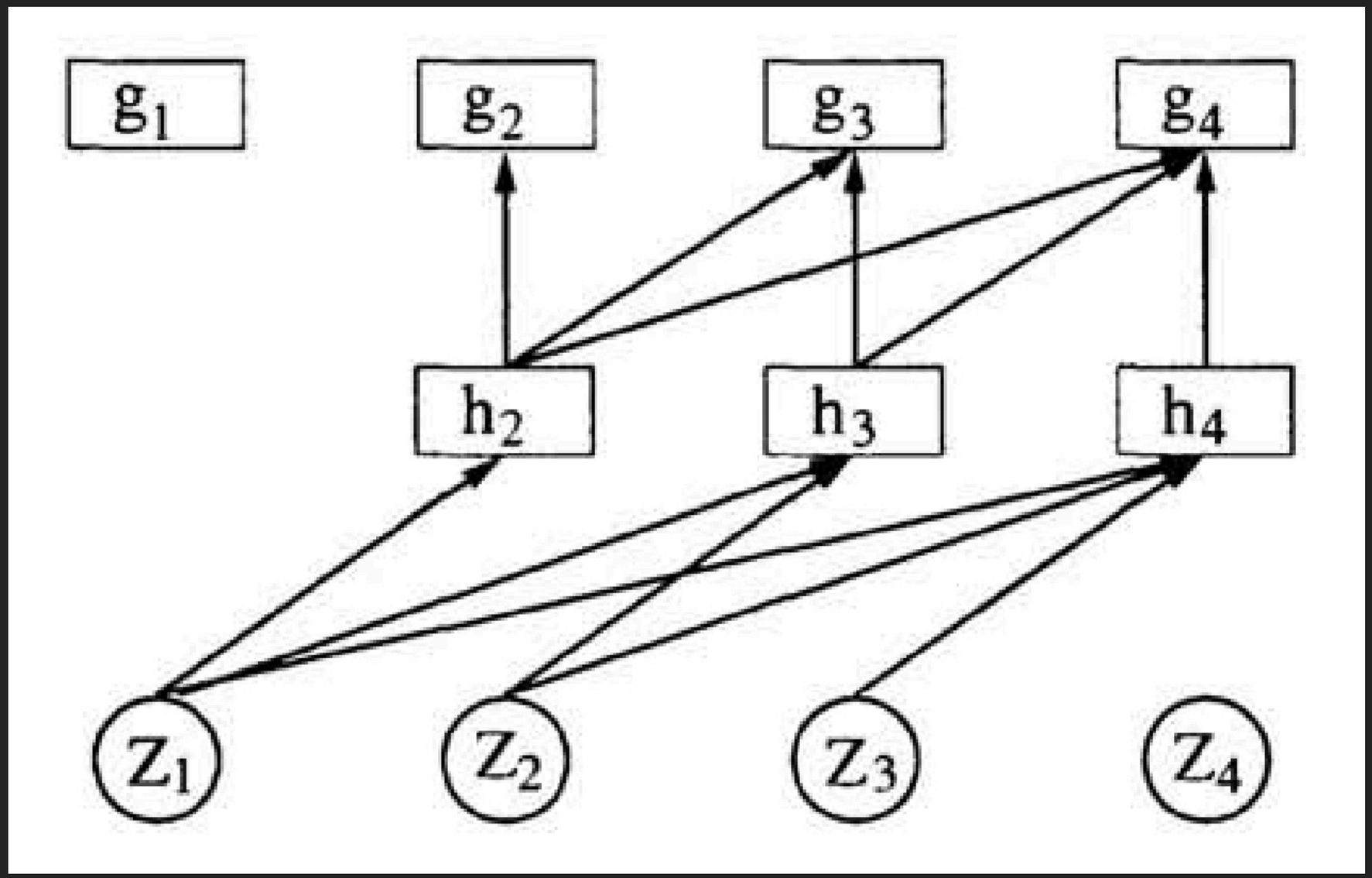


BENGIO & BENGIO, 2000

MODELING HIGH-DIMENSIONAL DISCRETE DATA WITH NEURAL NETWORKS

$$P(Z_i \mid Z_1, \ldots, Z_{i-1}) \sim g_i$$

Bengio and Bengio create an architecture for autoregressive modeling that exploits the universal approximation capability of neural networks.

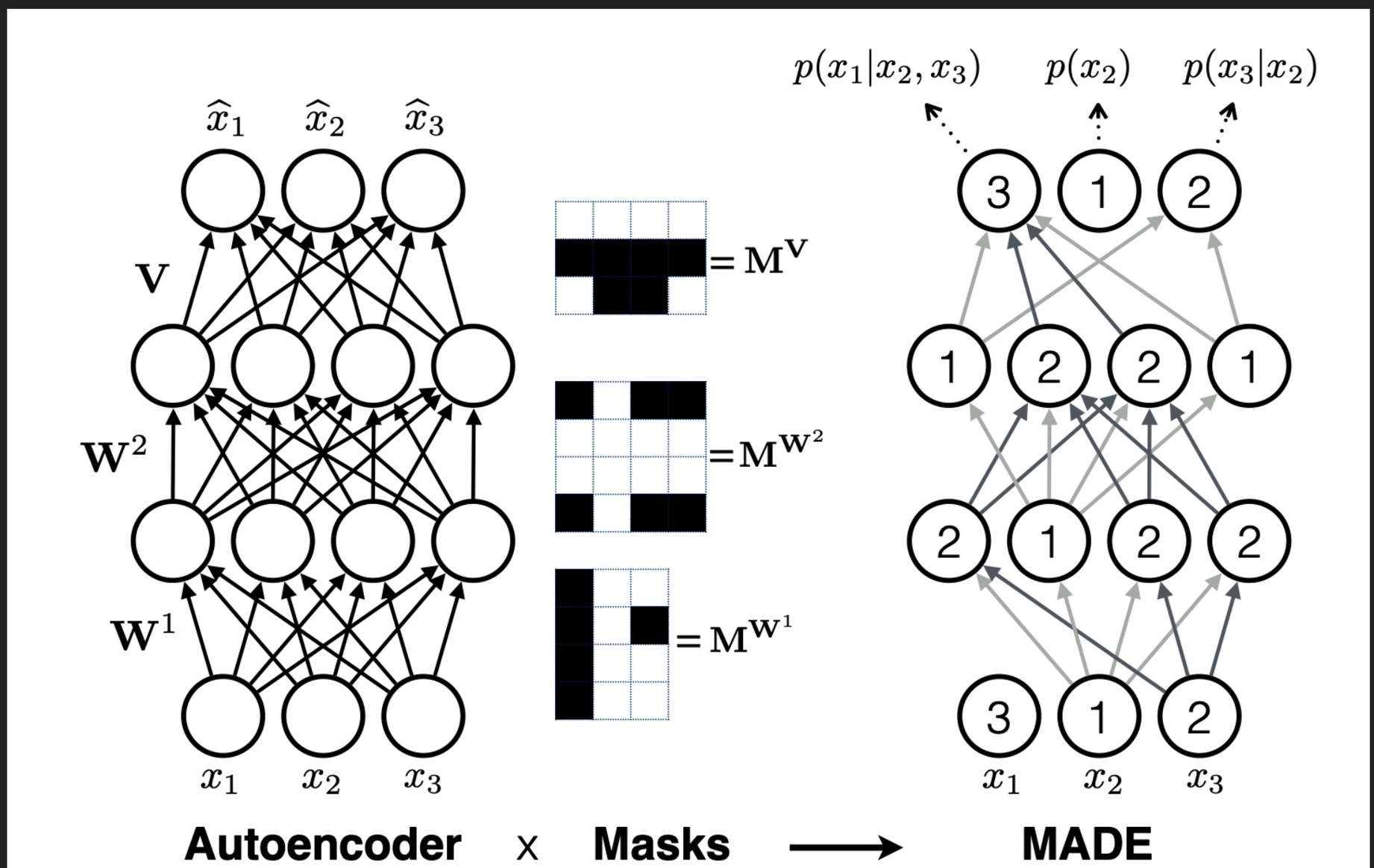


GERMAIN ET AL, 2015

MASKED AUTOENCODER FOR DISTRIBUTION ESTIMATION (MADE)

$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{b} + \mathbf{W} \cdot \mathbf{M}^{\mathbf{W}})\mathbf{x})$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{c} + (\mathbf{W} \cdot \mathbf{M}^{V})\mathbf{h}(\mathbf{x})$$



VAN DEN OORD ET AL, 2016

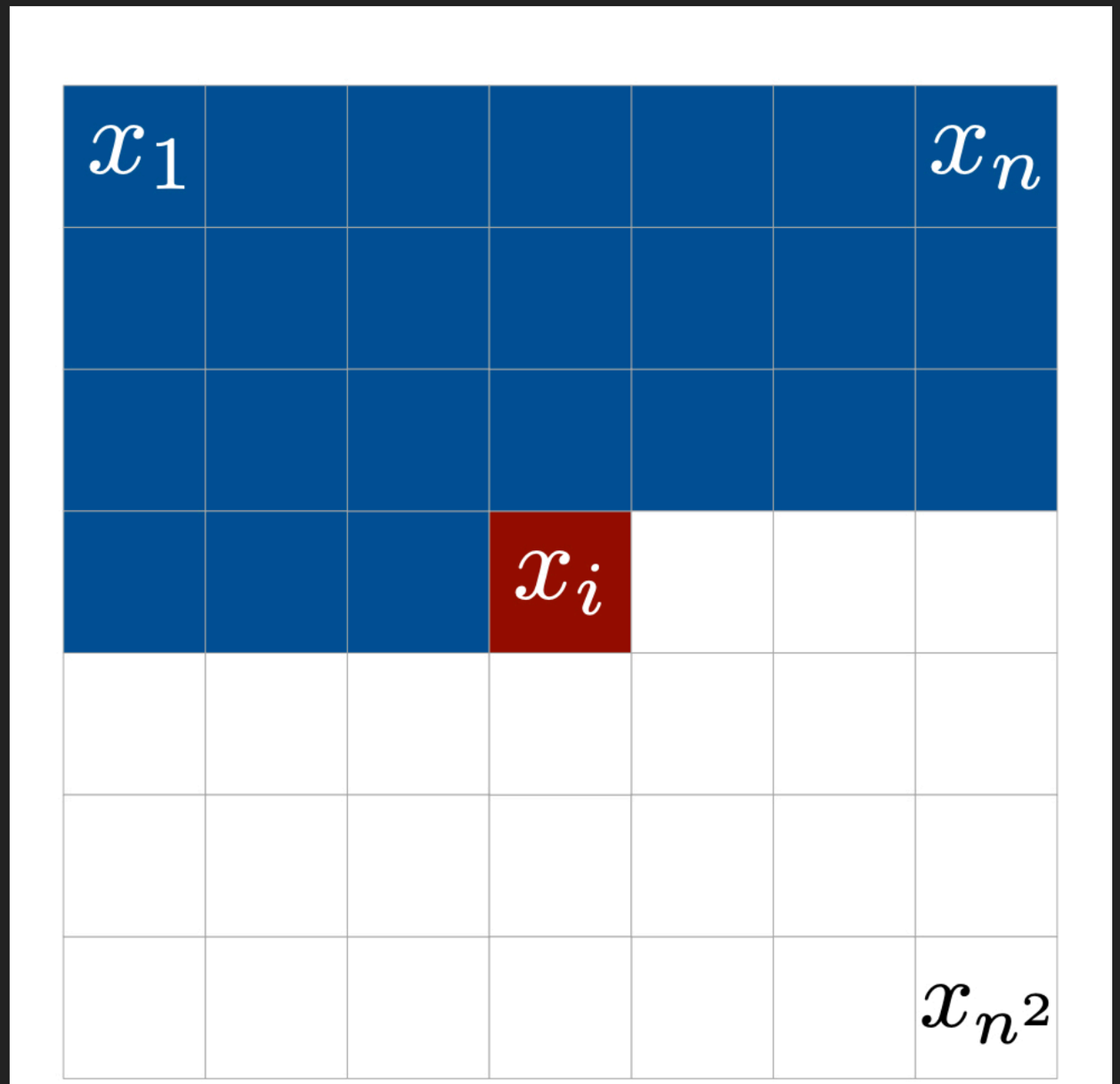
PIXEL RECURRENT NEURAL NETWORKS

The probability of e.g. an entire image is the product of the conditional probabilities of each pixel given the preceding pixels.

$$P(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i \mid x_1, \ldots, x_{i-1})$$

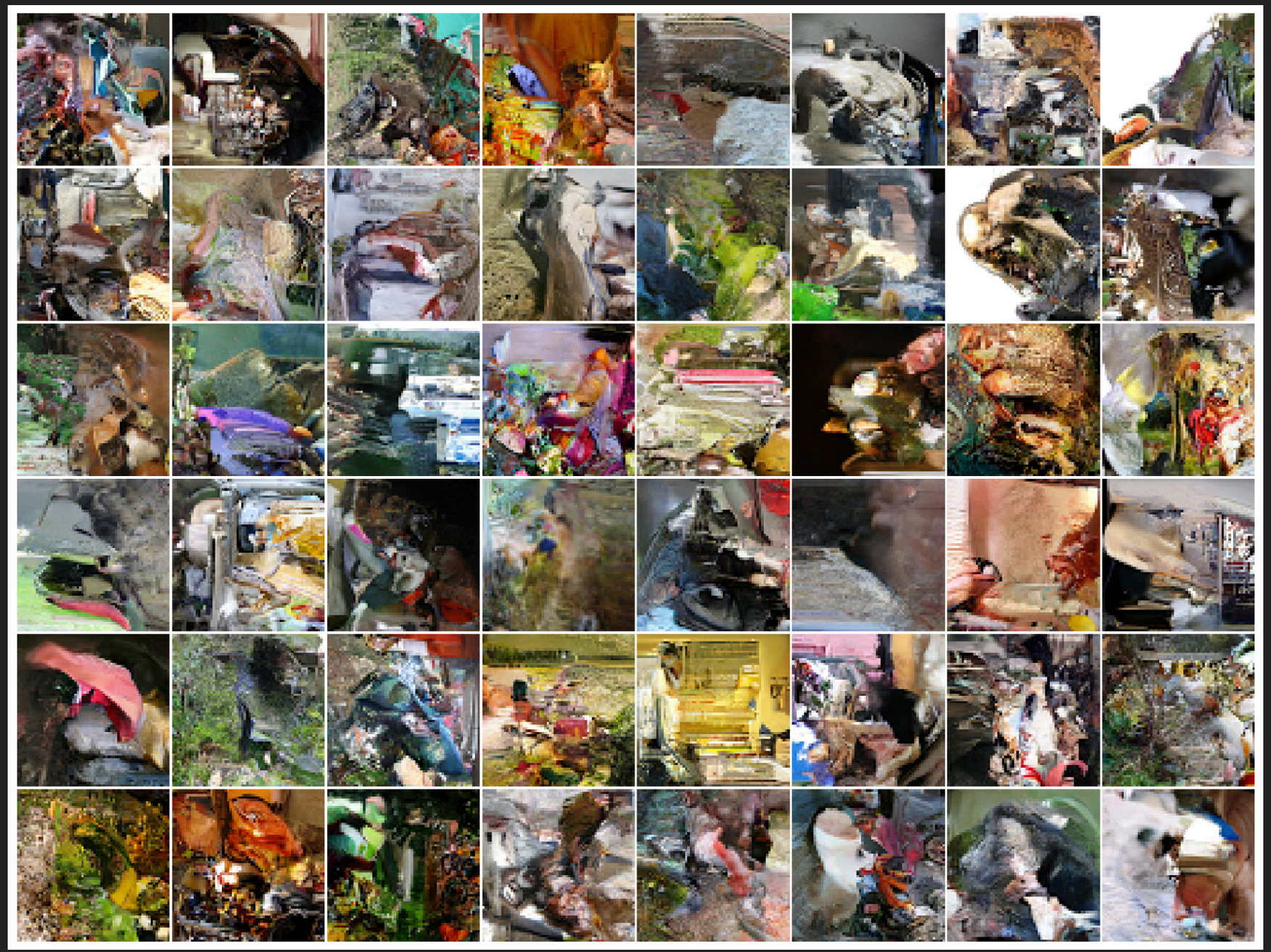
The target of each pixel is categorical, not real, and the output over each pixel is a softmax over 8-bit unsigned integers.

The native implementation of this is several LSTMs stacked atop one another. As you can imagine, it is quite slow (~50 sec/image).



VAN DEN OORD ET AL, 2016

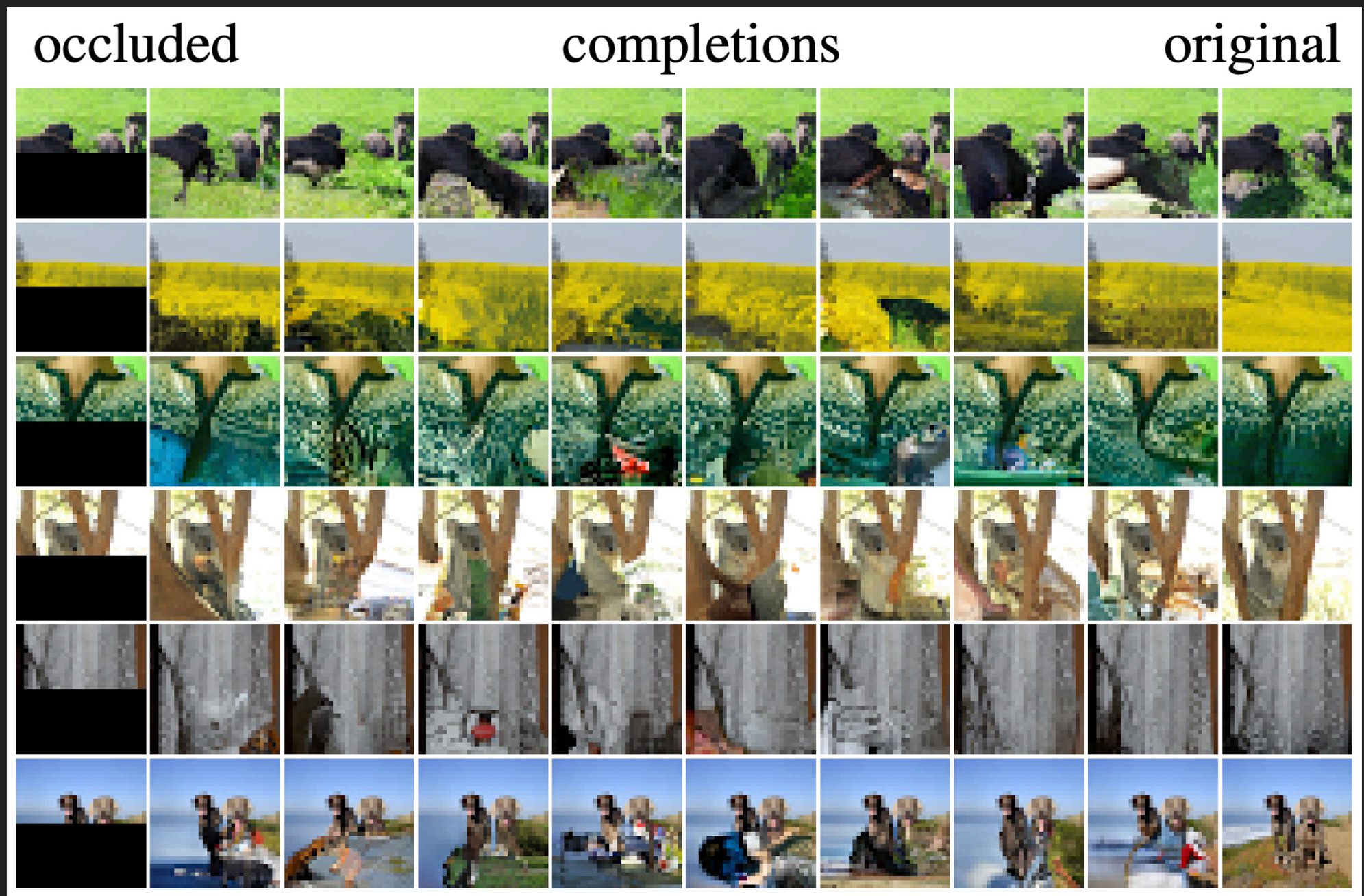
PIXEL RECURRENT NEURAL NETWORKS



VAN DEN OORD ET AL, 2016

PIXEL RECURRENT NEURAL NETWORKS

Why does this generate better images than those generated from scratch?



REZENDE AND MOHAMED, 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

> The Gaussian posterior distribution used in e.g. a variational auto encoder is, effectively, a high-bias model.

We would like more flexible distributions that are tractable.

Normalizing flows provide the ability to transform (simple, tractable) distributions into more complex distributions while being both *invertible* – requires log determinant of the Jacobian to be defined – and *fast* – because the determinant of the Jacobian is $O(d^3)$

REZENDE AND MOHAMED, 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

For a given transformation f of a latent variable z:

$$q(\mathbf{z}) = q(\mathbf{z}) \left| det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$

For a series of transformations $\mathbf{z}_K = f_K \circ \ldots \circ f_2 \circ f_1(\mathbf{z}_0)$

$$ln\ q_K(\mathbf{z}_K) = ln\ q_0(\mathbf{z}_0) - \sum_{k=1}^{K} ln \left| det \frac{\partial f_k}{\partial \mathbf{z}_{k-1}} \right|$$

REZENDE AND MOHAMED, 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

Decoder Network

$\mu_1 + \sigma_1 \cdot \epsilon_1$
$\mu_2 + \sigma_2 \cdot \epsilon_2$
$\cdots$
$\mu_K + \sigma_K \cdot \epsilon_K$

$KL(N(0,I), N(\mu, \sigma))$ $\mu + \sigma \cdot \epsilon$ $KL(N(0,I), N(\mu, \sigma))$

Encoder Network

Encoder Network

TYPES OF FLOWS

VOLUME-PRESERVING FLOWS

If the transformation preserves volume, the Jacobian determinant is 0. The challenge of designing a volume-preserving flow is making the transformation flexible.

Examples: NICE, RealNVP, Householder Flow

GENERAL NORMALIZING FLOWS

The Jacobian determinant is not guaranteed to be 0. The transformations are less constrained and can be more flexible.

Examples: Planar flows, Inverse Autoregressive Flows

INFINITESSIMAL FLOWS

Recall the notion of increasing the depth of the sequence of transformations to infinity such that each transformation can be seen as continuous and modeled as a partial differential equation. This will show up later when Teo Price-Broncucia presents ODE Nets.

REZENDE AND MOHAMED, 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

A *planar* flow is a transformation of the form $f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$

with elementwise non-linearity *h*, free parameters $\lambda = \mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}$

and log determinant

$$\psi(\mathbf{z}) = h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}$$

$$\left| det \frac{\partial f}{\partial \mathbf{z}} \right| = |det(\mathbf{I} + \mathbf{u}\psi(\mathbf{z})^T)| = |1 + \mathbf{u}^T\psi(\mathbf{z})|$$

How is the planar flow limited?

REZENDE AND MOHAMED, 2015

VARIATIONAL INFERENCE WITH NORMALIZING FLOWS

A *planar* flow is a transformation of the form $f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$

with element-wise non-linearity *h*, free parameters $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$

and log determinant

$$\psi(\mathbf{z}) = h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}$$

$$\left| det \frac{\partial f}{\partial \mathbf{z}} \right| = |det(\mathbf{I} + \mathbf{u}\psi(\mathbf{z})^T)| = |1 + \mathbf{u}^T\psi(\mathbf{z})|$$

How is the planar flow limited?

[It] can be interpreted as a MLP with a bottleneck hidden layer with a single unit. Since information goes through the single bottleneck, a long chain of transformations is required to capture high-dimensional dependencies. - Kingma et al, 2016, p. 3

INVERSE AUTOREGRESSIVE FLOWS

KINGMA ET AL, 2016

IMPROVED VARIATIONAL INFERENCE WITH INVERSE AUTOREGRESSIVE FLOW [CODE]

The transformation

$$y_0 = \mu_0 + \sigma_0 \epsilon_0$$
$$y_i = \mu_i(\mathbf{y}_{1:i-1}) + \sigma_i(\mathbf{y}_{1:i-1} \cdot \epsilon_i)$$

with inverse

$$\epsilon = (\mathbf{y} - \mu(\mathbf{y}))/\sigma(\mathbf{y})$$

has three virtues.

It addresses the single-unit bottleneck of planar flows.

The inverse can be parallelized, because the elements of ε do not depend on one another.

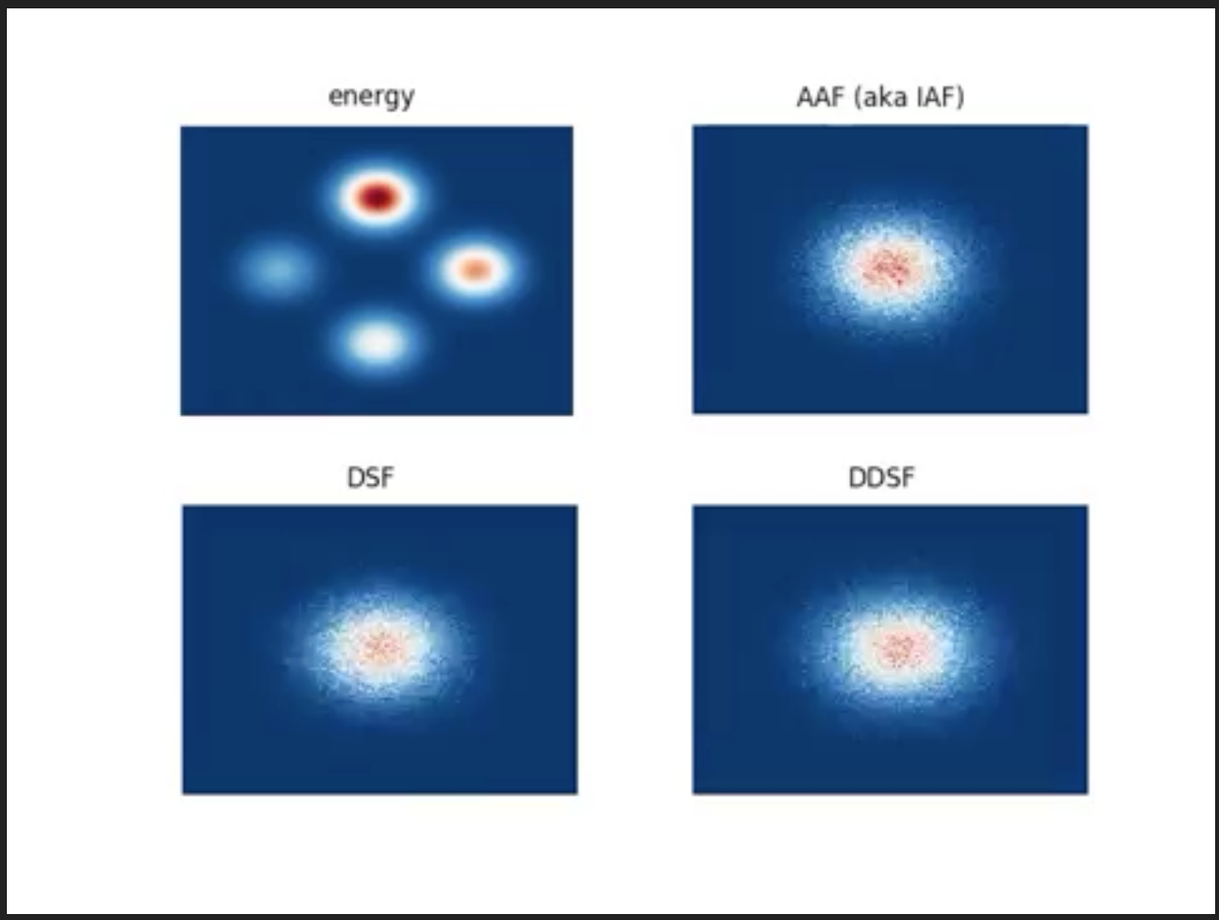
And the log determinant, due to the Jacobian being lower triangular with a simple diagonal, is simple:

$$log\ det \left| \frac{d\epsilon}{d\mathbf{y}} \right| = \sum_{i=1}^{D} - log\sigma_i(\mathbf{y})$$

NEURAL AUTOREGRESSIVE FLOWS

HUANG ET AL, 2019

NEURAL AUTOREGRESSIVE FLOWS [CODE]



https://www.youtube.com/watch?v=r_lpF6n9Fi8

- Glow: Generative Flow With Invertible 1x1 Convolutions [link]
- VideoFlow: A Flow-Based Generative Model for Video, Kumar et al, 2019 [link]
- Masked Autoregressive Flow for Density Estimation [link]
- Density Estimation using Real NVP, Dinh et al, 2017 [link]
- Sylvester Normalizing Flows for Variational Inference [link]