# Flow-based models

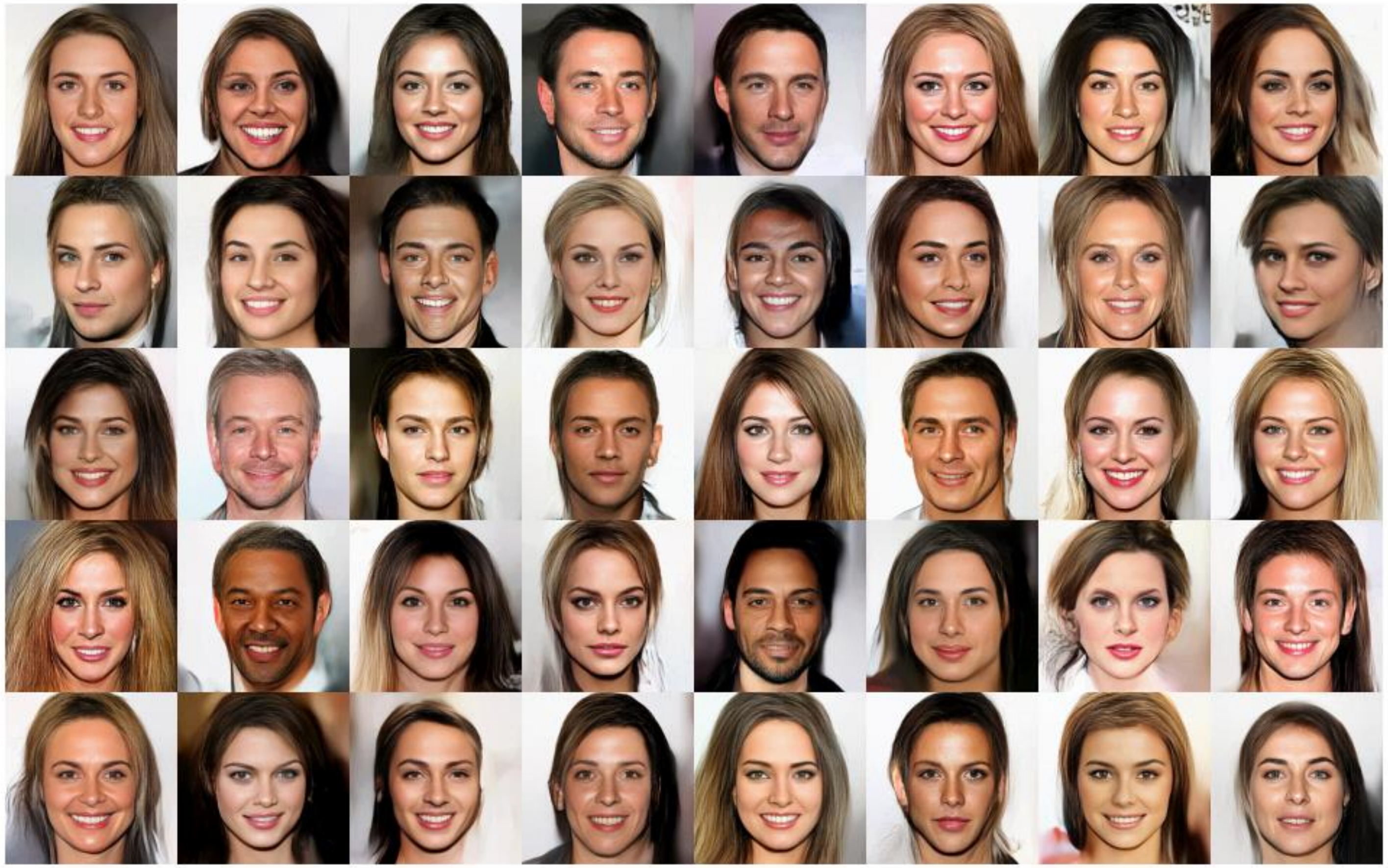


Figure 4: Random samples from the model, with temperature 0.7

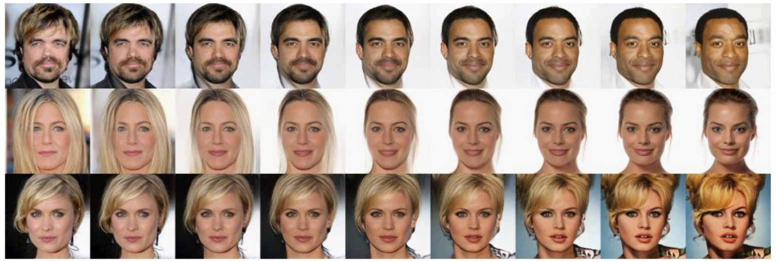
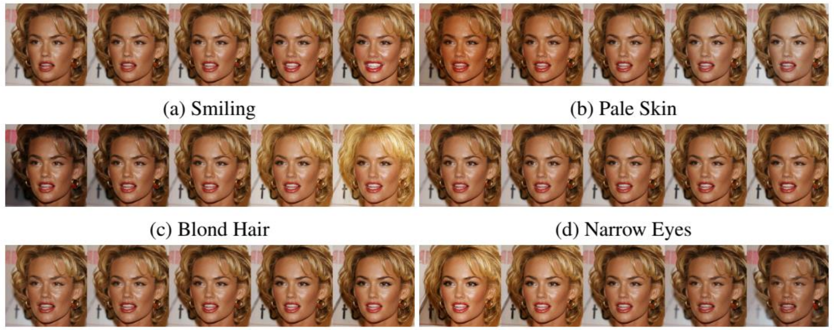


Figure 5: Linear interpolation in latent space between real images



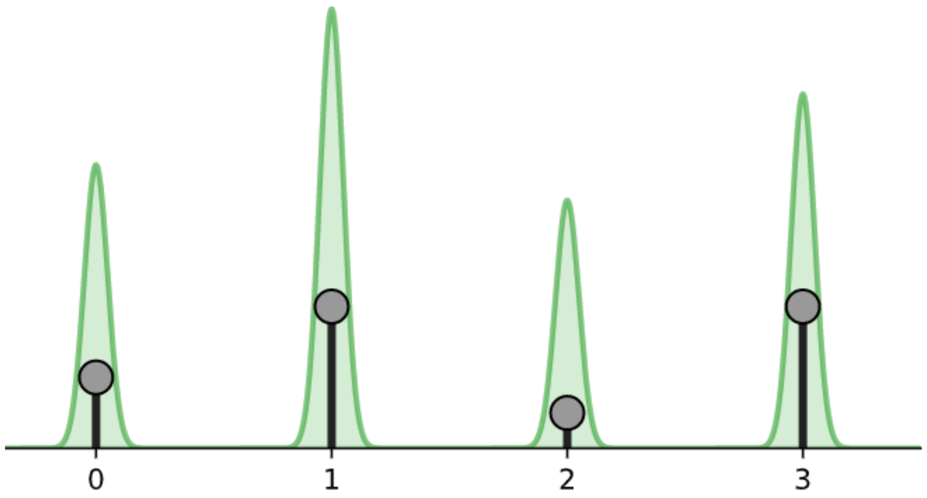
(a) Smiling (b) Pale Skin

(c) Blond Hair (d) Narrow Eyes

(e) Young (f) Male

# Normalizing flows on images

- Normalizing flows are continuous transformations
- Images contain discrete values
  → The model will assign δ-peak probabilities on integer (pixel) values only
  - These probabilities will be nonsensical, there is no smoothness




UVADLC tutorial

# (Variational) dequantization

- Add (continuous) noise $u \sim q(u|x)$ to input variables $v = x + u$
- The data log-likelihood then is

$$\log p(x) = \log \int p(x+u)\, du = \log \mathbb{E}_{u\sim q(u|x)} \left[ \frac{p(x+u)}{q(u|x)} \right] \geq \mathbb{E}_{u\sim q(u|x)} \log \left[ \frac{p(x+u)}{q(u|x)} \right]$$

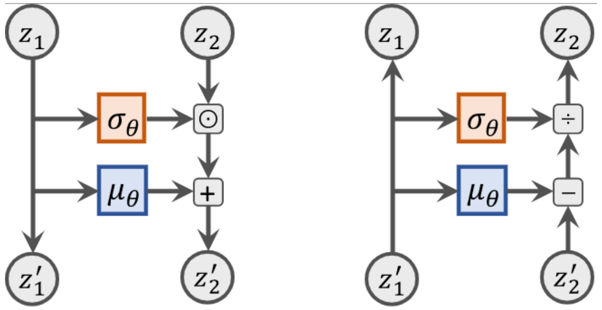
- If $q(u|x)$ is the uniform distribution the standard dequantization
  - Probability between two consecutive values is fixed
    → resemble boxy boundaries between values
- Better learn $q(u|x)$ in a variational manner
  → Variational dequantization

# Coupling layers

- Given input $\boldsymbol{z}$ the output of the transformation is

$$\boldsymbol{z}' = \begin{bmatrix} \boldsymbol{z}'_{1:j} \\ \boldsymbol{z}'_{j+1:d} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}_{1:j} \\ \mu_\theta(\boldsymbol{z}_{1:j}) + \sigma_\theta(\boldsymbol{z}_{1:j}) \odot \boldsymbol{z}_{j+1:d} \end{bmatrix}$$

  - $\mu_\theta, \sigma_\theta$ are neural networks with shared parameters

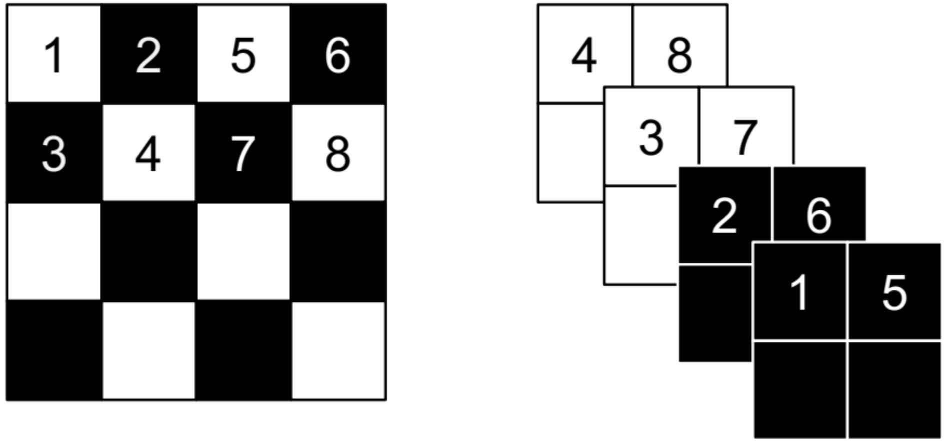



- Easy inverse: $\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_{1:j} \\ \dfrac{\left(\boldsymbol{z}'_{j+1:d} - \mu_\theta(\boldsymbol{z}_{1:j})\right)}{\sigma_\theta(\boldsymbol{z}_{1:j})} \end{bmatrix}$

- Easy triangular Jacobian $\dfrac{\partial \boldsymbol{z}'}{\partial \boldsymbol{z}} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \dfrac{\partial \boldsymbol{z}'_{j+1:d}}{\partial \boldsymbol{z}_{1:j}} & \mathrm{diag}\left(\sigma_\theta(\boldsymbol{z}_{1:j})\right) \end{bmatrix}$
  - The log determinant is $\sum_j \log \sigma_\theta(\boldsymbol{z}_j)$

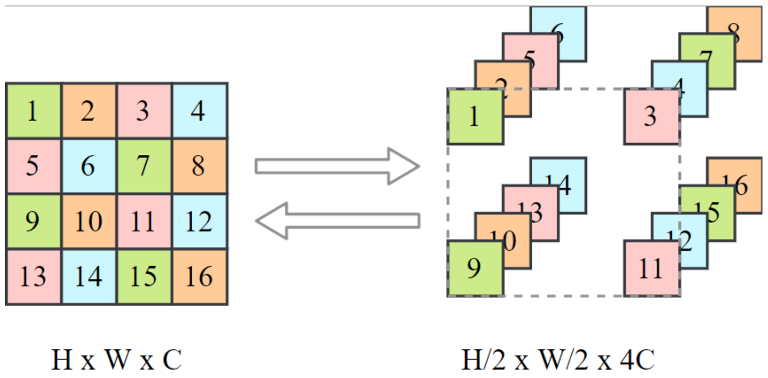
# Splitting dimensions in images

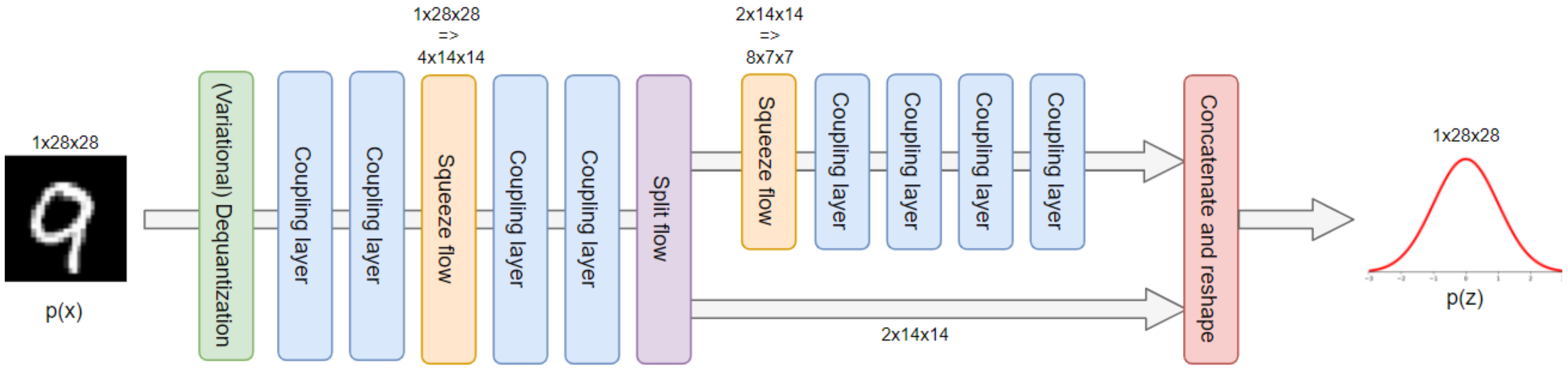
- Use masking
  - Checkers pattern
  - Splitting across channels
- Alternate dimensions between consecutive layers
  $\rightarrow$ not always the same $1{:}d$ dimensions remain untouched



# Multi-scale architecture

- Invertibility → number of dimensions before and after $f$ is the same
  - High computational complexity for large images
- Apply new transformations to half the input only
  - For the other half use the prior (previous) trasnfromations
- Use squeeze to turn spatial to channel dimensions
  - And split for halving the input




[UVADLC tutorial]

# GLOW, FLOW, FLOW++

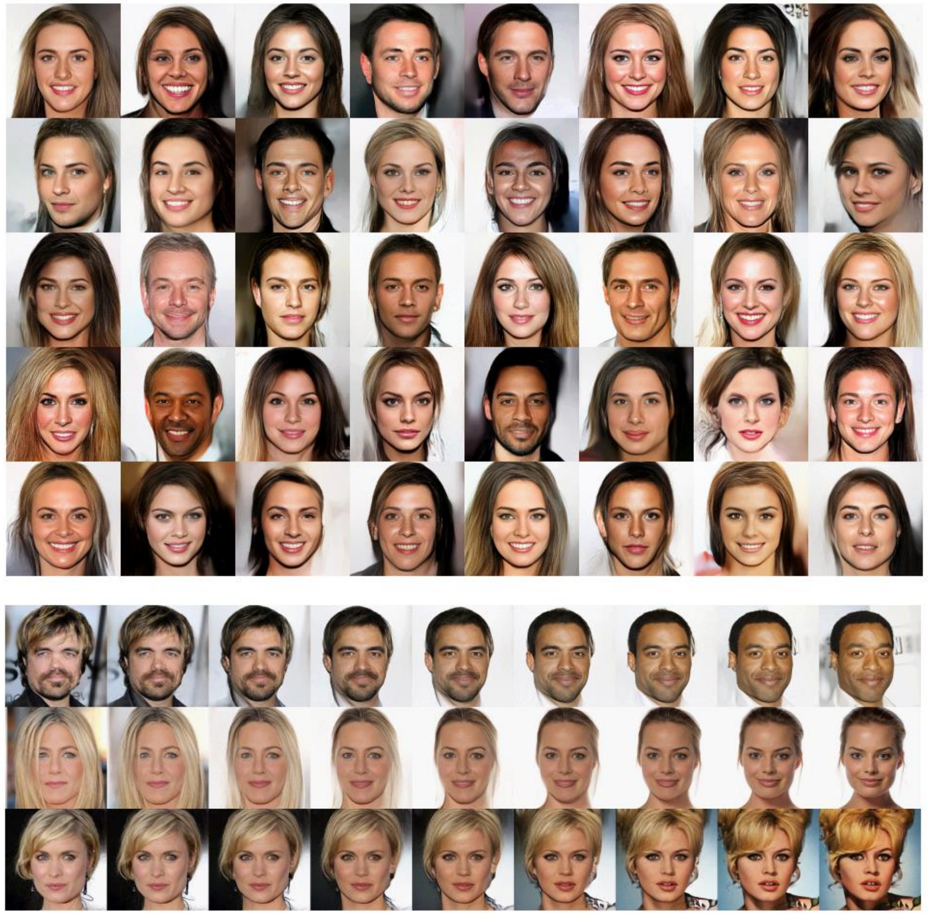


Figure 5: Linear interpolation in latent space between real images

*Kingma, Dhariwal, Glow: Generative Flow with Invertible 1x1 Convolutions*

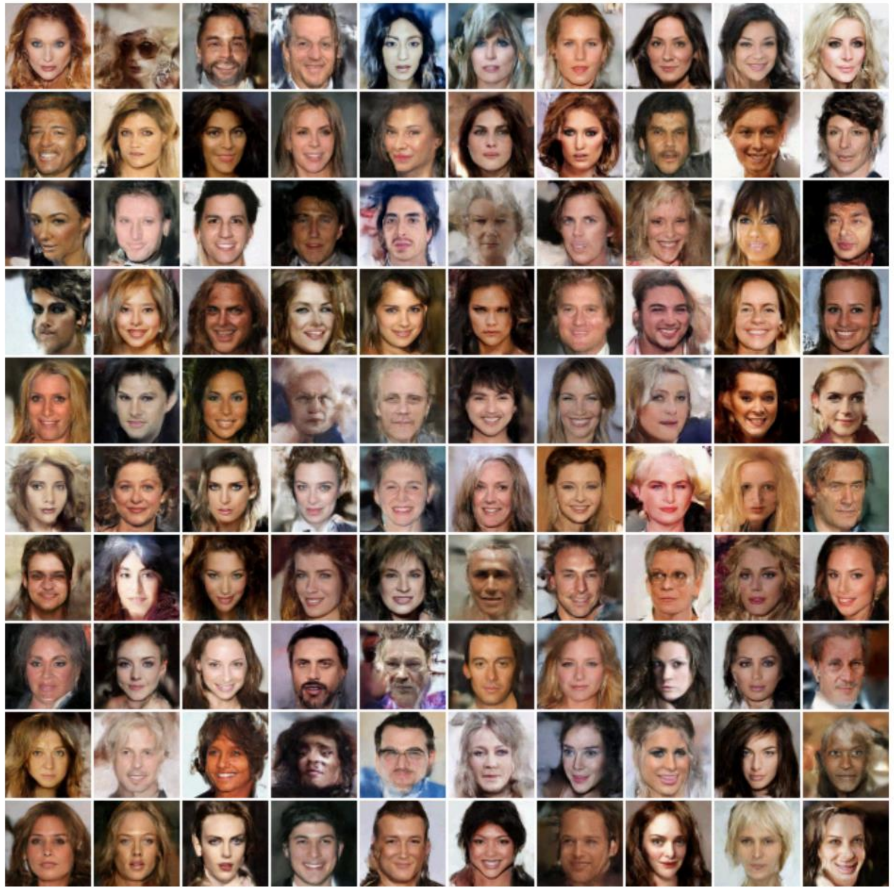


*Figure 4.* Samples from Flow++ trained on 5-bit 64x64 CelebA, without low-temperature sampling.

*Kingma, Dhariwal, Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design*

# Categorical normalizing flows [not in exams]

- Normalizing flows with variational inference to learn representations of categorical data on continuous space
  - Learnable, smooth, support for higher dimensions

- Learning must ensure no loss of information
  → the volumes that represent categorical data must not-overlap
  - Otherwise, to which category does the representation correspond to?

$$p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z}\sim q(\cdot|\boldsymbol{x})}\left[\frac{\prod_i p(x_i|\boldsymbol{z}_i)}{q(\boldsymbol{z}|\boldsymbol{x})} p(\boldsymbol{z})\right]$$

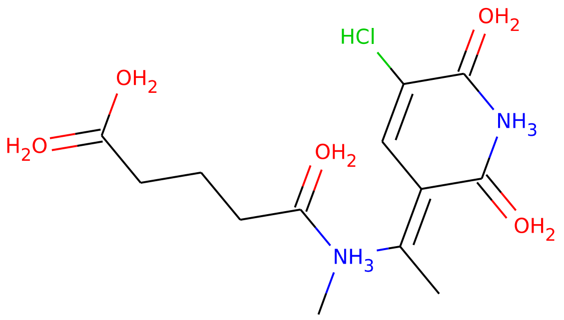
  - Factorized posterior $\prod_i p(x_i|\boldsymbol{z}_i)$ to encourage learning non-overlapping $\boldsymbol{z}_i$

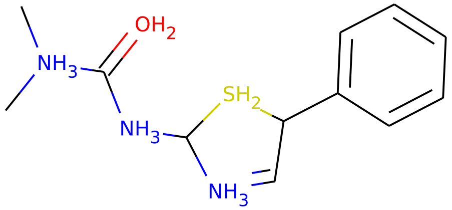
*Lippe and Gavves, Categorical Normalizing Flows via Continuous Transformations, in submission to ICLR 2021*

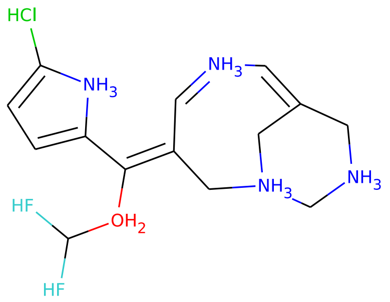
# Graph generation with categorical normalizing flows

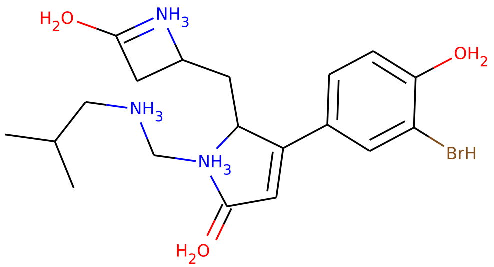
Results on the Zinc250k dataset (224k examples)

| **Method** | **Validity** | **Uniqueness** | **Novelty** | **Reconstruction** | **Parallel** | **General** |
|---|---|---|---|---|---|---|
| JT-VAE | 100% | 100% | 100% | 71% | ✗ | ✗ |
| GraphAF | 68% | 99.10% | 100% | 100% | ✗ | ✓ |
| R-VAE | 34.9% | 100% | – | 54.7% | ✓ | ✓ |
| GraphNVP | 42.60% | 94.80% | 100% | 100% | ✓ | ✓ |
| GraphCNF | 83.41% (±2.88) | 99.99% (±0.01) | 100% (±0.00) | 100% (±0.00) | ✓ | ✓ |
| + Sub-graphs | 96.35% (±2.21) | 99.98% (±0.01) | 99.98% (±0.02) | 100% (±0.00) | ✓ | ✓ |



# Normalizing flows: pros and cons

- Starting from a simple density like a unit Gaussian we can obtain any complex density that match our data without even knowing its analytic form
- Tractable density estimation
- Efficient parallel sampling and learning
- Often very many transformations required → Very large networks needed
- Constrained to invertible transformations with tractable determinant
- Tied encoder and decoder weights
- Transformations cannot easily introduce bottlenecks

UVADLC tutorial

# A summary of properties

| | Training | Likelihood | Sampling | Compression |
|---|---|---|---|---|
| **Autoregressive models (e.g., PixelCNN)** | Stable | Yes | Slow | No |
| **Flow-based models (e.g., RealNVP)** | Stable | Yes | Fast/Slow | No |
| **Implicit models (e.g., GANs)** | Unstable | No | Fast | No |
| **Prescribed models (e.g., VAEs)** | Stable | Approximate | Fast | Yes |

17

*J. Tomczak's lecture from April, 2019*

# Summary

- Early autoregressive models
- Modern autoregressive models
- Normalizing flows
- Flow-based models

All mentioned papers as reading material