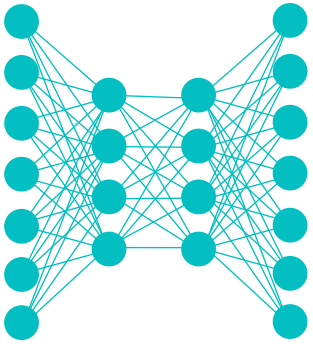
Lecture Notes for

Neural Networks and Machine Learning

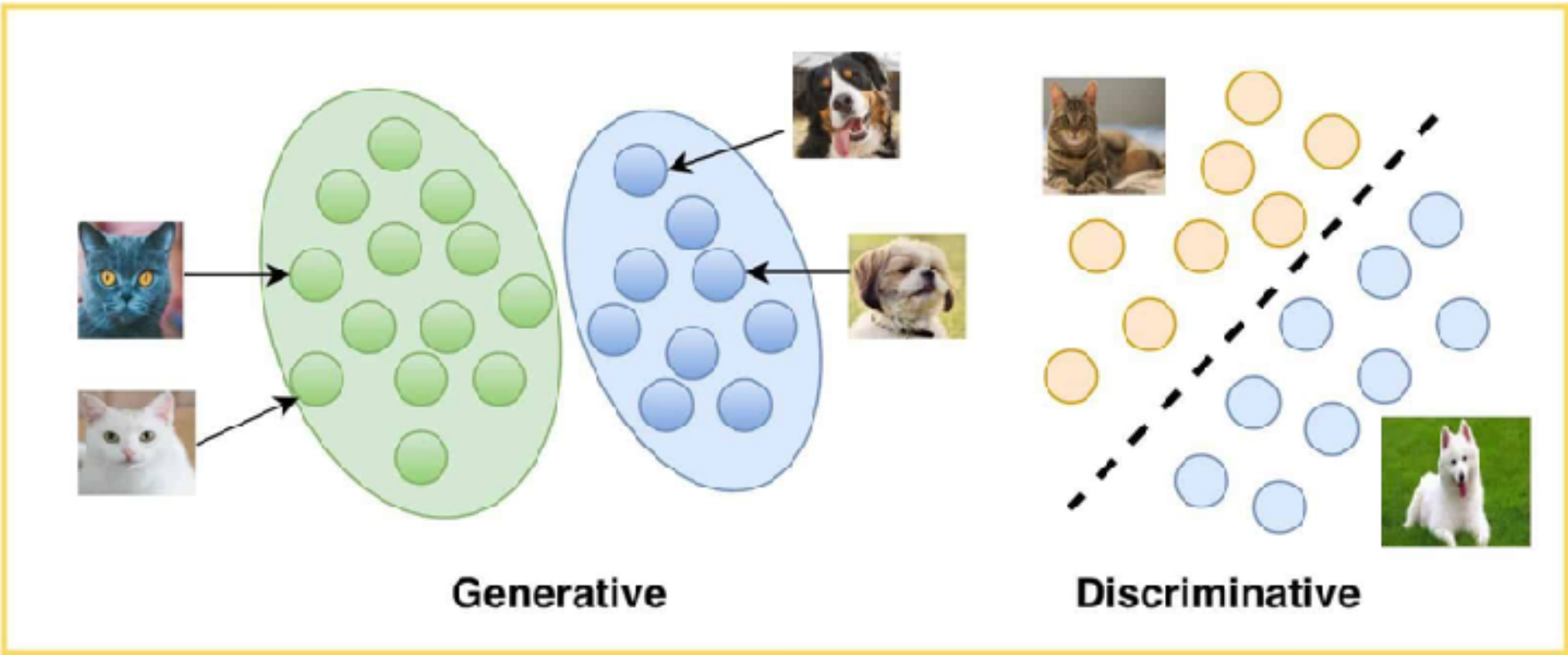
Generative Models
VAEs + Stable Diffusion



Logistics and Agenda

- Logistics
  - Grading Update
- Agenda
  - VAEs
  - Stable Diffusion Basics
  - Student Paper Presentation
  - Stable Diffusion 3
  - Final Project Town Hall

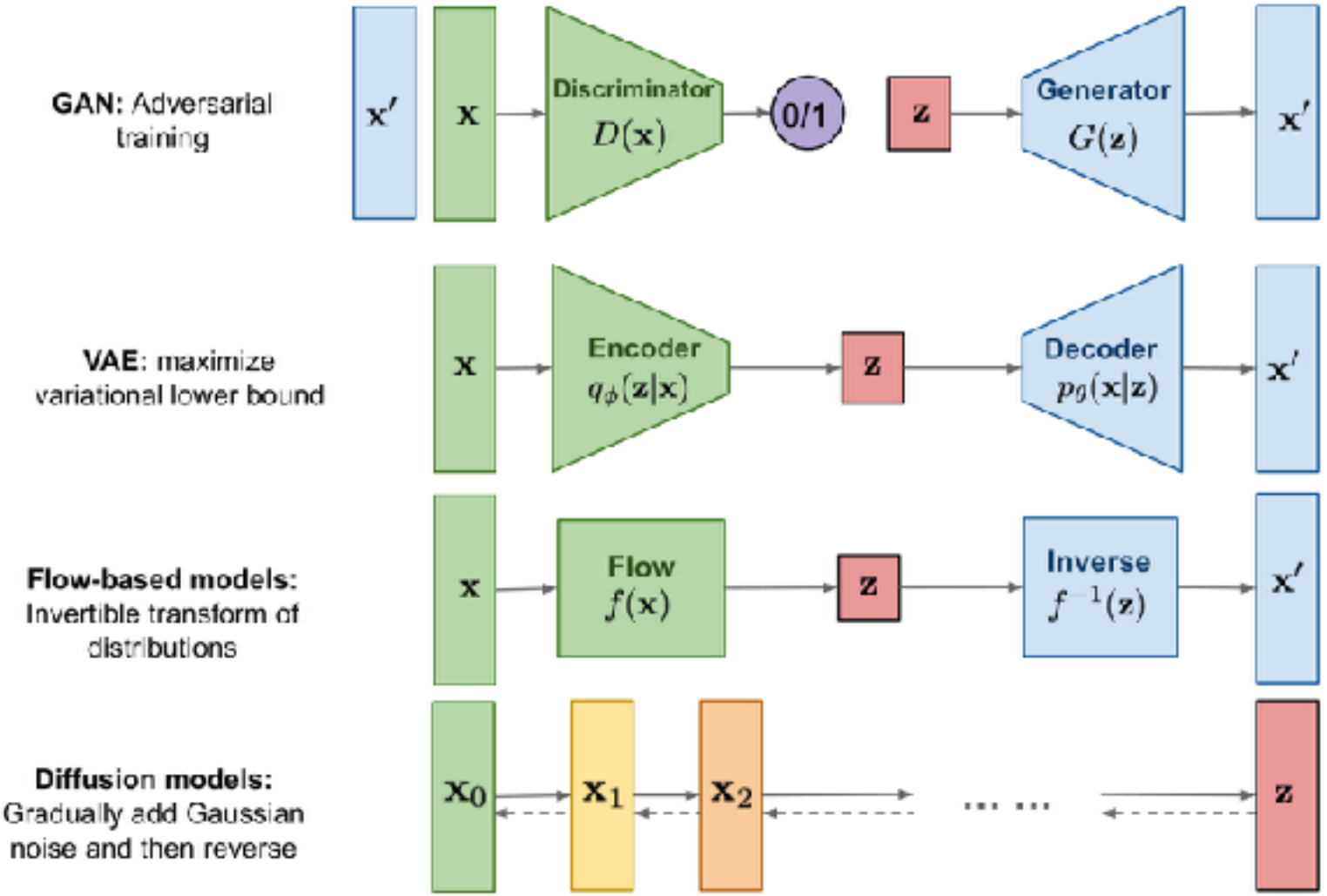
Generative Deep Learning Basics



https://learnopencv.com/generative-and-discriminative-models/

Comparative Overview, Generative Methods

https://learnopencv.com/image-generation-using-diffusion-models/



A great resource for understanding at high level: https://jalammar.github.io/illustrated-stable-diffusion/

Probability Review

some function, f

$$\mathbf{E}[f(\cdot)] = \int x \cdot f(x)\, dx \approx \sum_{\forall i} x^{(i)} \cdot f(x^{(i)})$$

$$\mathbf{E}_{z \leftarrow q(z|x)}[f(\cdot)] = \int q(z|x) \cdot f(x)\, dx \approx \sum_{\forall i} q(z|x^{(i)}) \cdot f(x^{(i)})$$

Expected value of f under conditional distribution, q

z is latent variable, $x^{(i)}$ is an observation

$$\therefore \mathbf{E}_{z \leftarrow q(z|x)}[\log f(\cdot)] = \sum_{\forall i} q(z|x^{(i)}) \cdot \log\left(f(x^{(i)})\right)$$

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

Also: Bayes Theorem

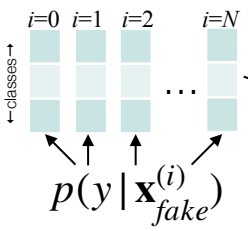
If function is a probability, this is just the negative of cross entropy of distributions:

$$H(q,p) = -\sum_{x} q(x) \cdot \log\left(p(x)\right)$$

Recall that KL divergence is a measure of difference in two distribution, and is just:

$$D_{KL}\left(p||q\right) = \sum_{x} p(x) \cdot \log\left(\frac{p(x)}{q(x)}\right) = \mathbf{E}_p\left[\log\left(\frac{p(x)}{q(x)}\right)\right]$$

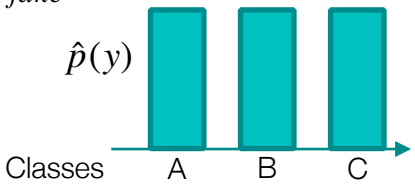
An Accepted Measure: Inception Score



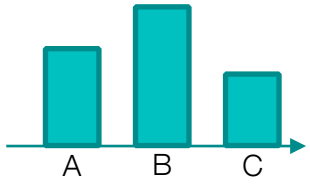
$$\hat{p}(y) = \frac{1}{N}\sum_{\forall i} p(y \mid \mathbf{x}^{(i)}_{fake})$$

marginal

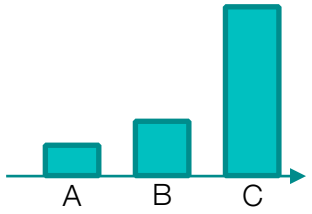
Expected class distribution through a trained CNN, like Inception should be **nearly uniform** in ideal case



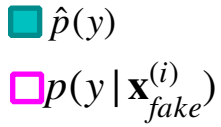
Ideal: Generator has diverse classes



Good: Generator is somewhat diverse



Bad: Generator not diverse

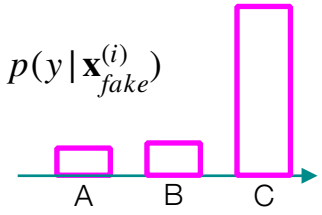


$$IS(G) \approx \exp\left(\frac{1}{N}\sum_{i} D_{KL}\left(p(y \mid \mathbf{x}^{(i)}_{fake}) \| \hat{p}(y)\right)\right)$$

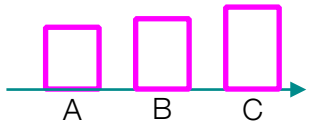
one example generated

typical generation

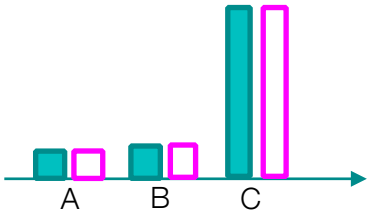
average KL Divergence of marginal of generated images with $\hat{p}$, ideally **differ dramatically**



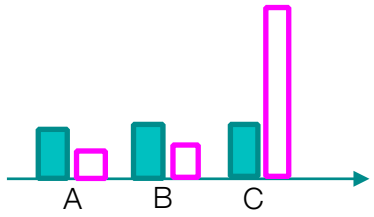
Ideal: Single Example is distinct class



Bad: Single Example is not really distinct



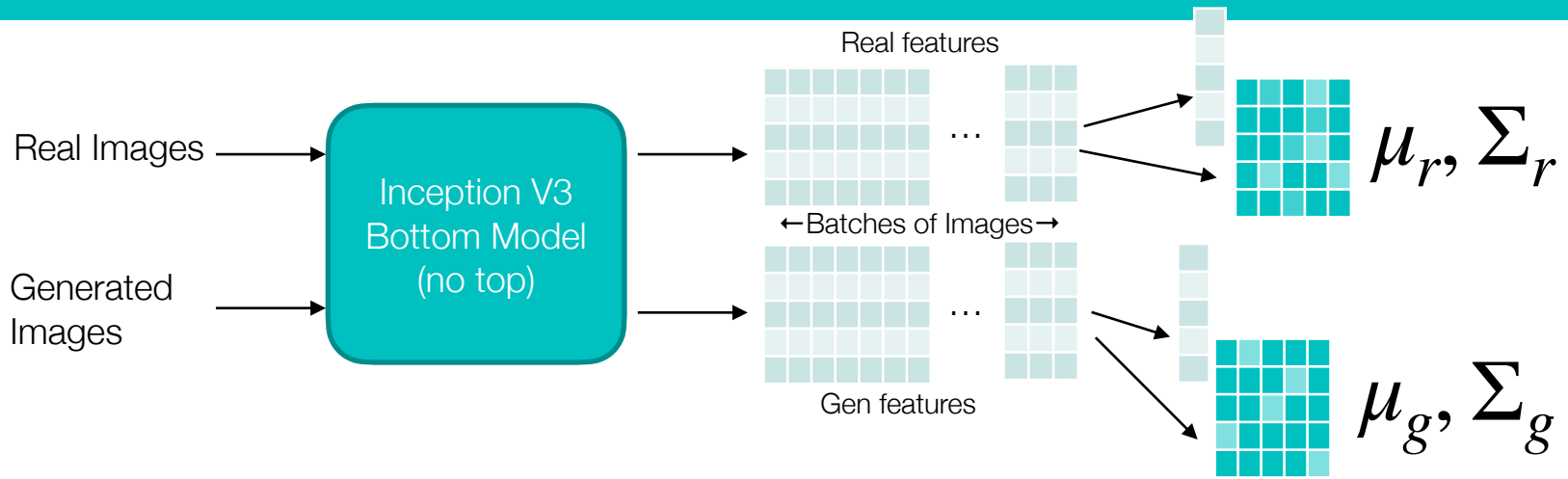
Bad: Distinct because not diverse



Ideal: Diverse and Distinct

Other Explanation: https://medium.com/octavian-ai/a-simple-explanation-of-the-inception-score-372dff6a8c7a

Another Measure: Fréchet Inception Distance

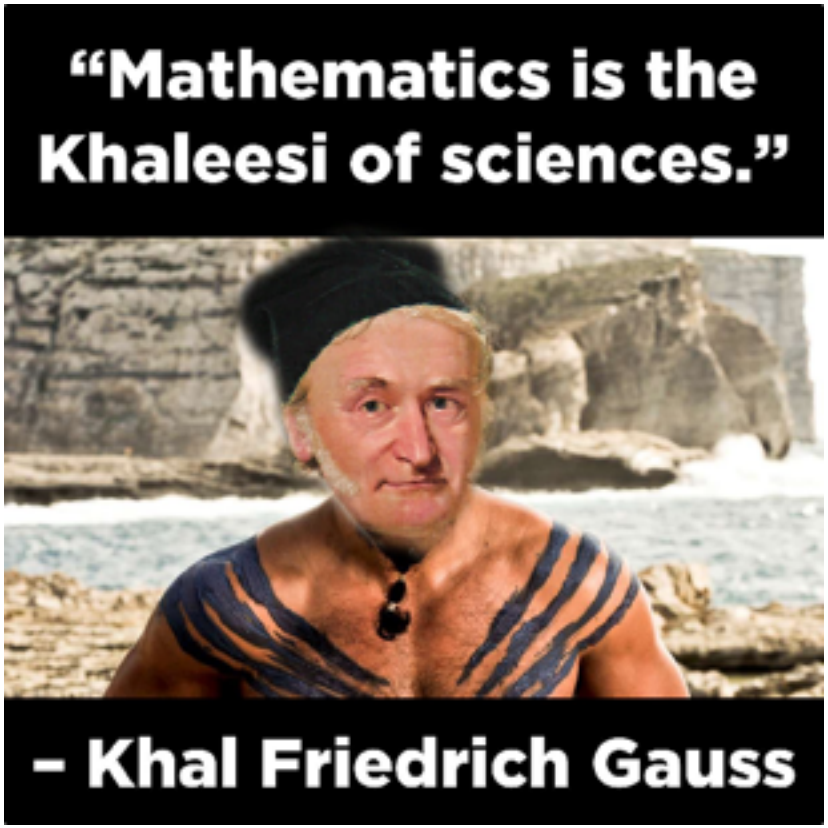


- Compare the distribution of data from CNN model (like inception)
- If distributed Gaussian, then we can use Fréchet Distance as follows:

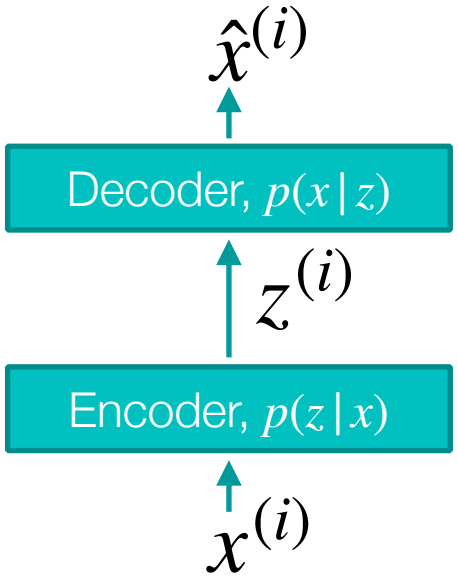
$$d_F\left(\mathcal{N}(\mu_r, \Sigma_r), \mathcal{N}(\mu_g, \Sigma_g)\right)^2 = \underbrace{\|\mu_r - \mu_g\|_2^2}_{\text{diff in feature wise means}} + \text{tr}\left(\underbrace{\Sigma_r + \Sigma_g - 2\left(\Sigma_r\Sigma_g\right)^{\frac{1}{2}}}_{\text{diff in feature wise covariances}}\right)$$

"The FID compares the mean and standard deviation of the deepest layer in Inception v3. These layers are closer to output nodes that correspond to real-world objects such as a specific breed of dog or an airplane, and further from the shallow layers near the input image." Wikipedia

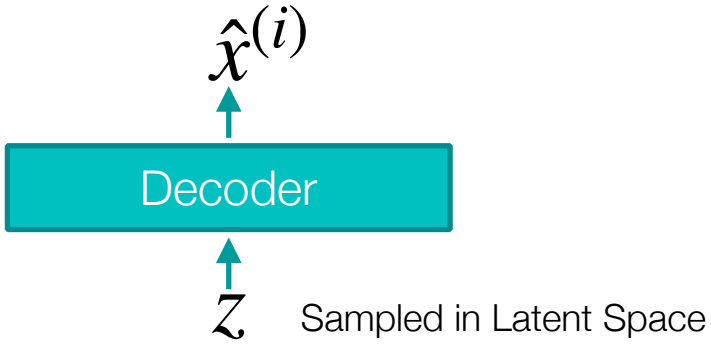
Variational Auto Encoding



Can Auto Encoding Generate Samples?



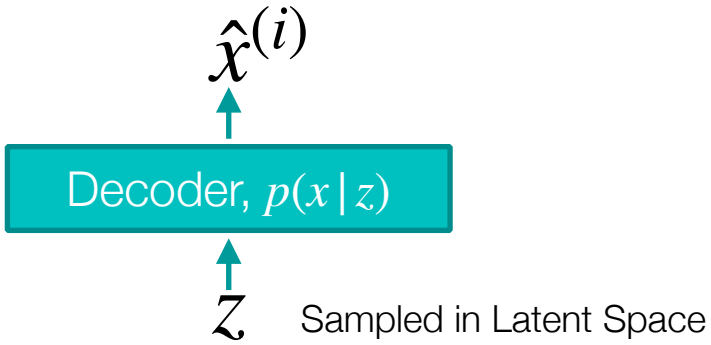
Once trained, is it possible to generate data?



- Does this work for simple auto encoding?
  - Yes, but not satisfactory results (latents are too "wild")
- Learned space is **not continuous** (need to avoid parts of space)
- Features could be highly **correlated**, related in **complex** ways
  - Creates **difficulty in sampling** from the latent space
- Need to **define constraints** on latent space…

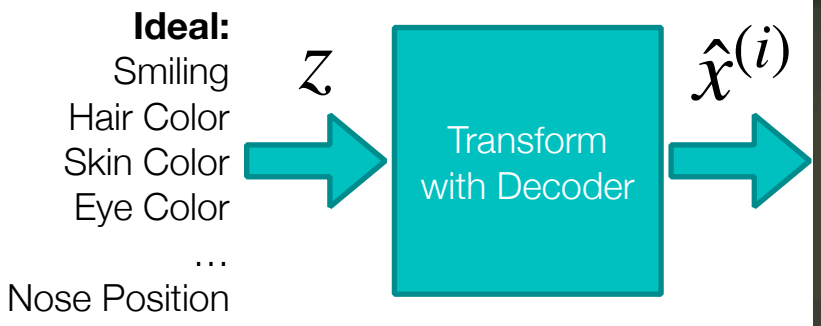
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

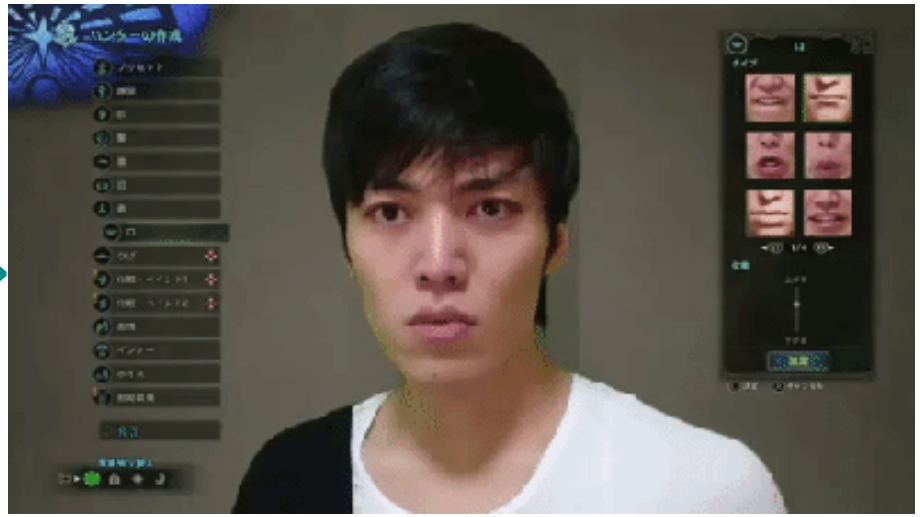
Reasonable constraints for latent space?



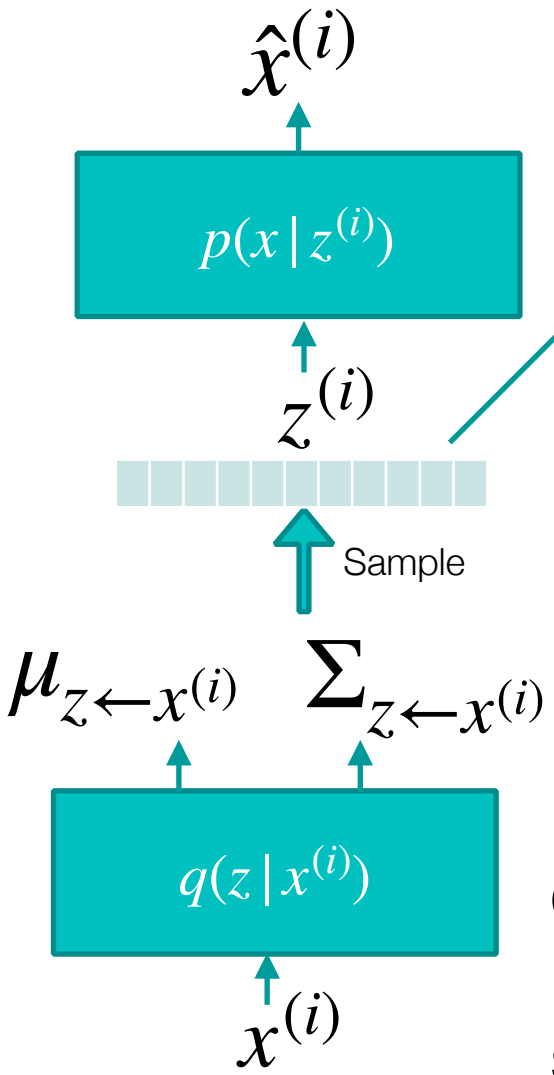
What is the ideal $p(z)$?

- $p(z)$ should be simple, easy to sample from: **Normally Distributed**
- Each component should be independent and identically distributed (i.i.d).: **Diag. Covariance**
  - Encourages features that may be semantic, like expert might select
- **Disentangled** features





Variational Encoding Motivation



Step Three: Make conditional p and q Similar

$$D_{KL}\left[q(z \mid x^{(i)}) \| p(z \mid x)\right] = \mathbf{E}_{q(z|x)}\left[\log\left(\frac{q(z \mid x^{(i)})}{p(z \mid x)}\right)\right]$$

Step Four: Use Variational Inference

Assume that a family of distributions can maximize likelihood of observing $x^{(i)}$:

$$\log p(x)_{\forall i} \approx \mathbf{E}_{\mathbf{z} \leftarrow q(z|x)}\left[\log p(x^{(i)})\right]$$

Max Log Lik:: maximize probability of observed $x^{(i)}$ given family of distributions q hope this is a good approximation

Output of network, q, are the mean and covariance for sampling a variable z

Need a new formulation

$$\log p(x)_{\forall i} \approx \mathbf{E}_{\mathbf{z} \leftarrow q(z|x)} \left[\log p(x^{(i)}) \right]$$ Maximize!

$$= \mathbf{E}_q \left[\log \frac{p(x^{(i)} \mid z) p(z)}{p(z \mid x^{(i)})} \frac{q(z \mid x^{(i)})}{q(z \mid x^{(i)})} \right]$$ Variational Bayes + multiply by one

$p(z \mid x^{(i)})$ this is still a problem

$$= \mathbf{E}_q \left[\log p(x^{(i)} \mid z) \right] + \mathbf{E}_q \left[\log \frac{p(z)}{q(z \mid x^{(i)})} \right] + \mathbf{E}_q \left[\log \frac{q(z \mid x^{(i)})}{p(z \mid x^{(i)})} \right]$$

$$= \mathbf{E}_q \left[\log p(x^{(i)} \mid z) \right] - \mathbf{E}_q \left[\log \frac{q(z \mid x^{(i)})}{p(z)} \right] + \mathbf{E}_q \left[\log \frac{q(z \mid x^{(i)})}{p(z \mid x^{(i)})} \right]$$

$$= \mathbf{E}_q \left[\log p(x^{(i)} \mid z) \right] - D_{KL} \left[q(z \mid x^{(i)}) \| p(z) \right] + D_{KL} \left[q(z \mid x^{(i)}) \| p(z \mid x^{(i)}) \right]$$

always non-negative

$$\log p(x)_{\forall i} \geq \mathbf{E}_q \left[\log p(x^{(i)} \mid z) \right] - D_{KL} \left[q(z \mid x^{(i)}) \| p(z) \right]$$ Will Maximize Lower Bound

Can we motivate this in a different way?

The Loss Function

Maximize through
Error of **Reconstruction**
Same as minimizing cross entropy

want $p(z)$ to be $\mathcal{N}(\mu = 0, \Sigma = I)$
latent space **constraint**, **normality**

$$q(z \mid x^{(i)}) \rightarrow (\mu_{z|x}, \Sigma_{z|x}) \quad p(z) \rightarrow \mathcal{N}(0,1)$$

$$D_{KL}\left((\mu, \Sigma) \| \mathcal{N}(0,1)\right) = \frac{1}{2}\left(\text{tr}(\Sigma) + \mu \cdot \mu^T - \underbrace{k}_{|z|} - \log\left(\det(\Sigma)\right)\right)$$

Can get this by manipulating the KL for normal distribution

Determinant of diagonal matrix is simple. Motivates diagonal covariance…

$$= \frac{1}{2}\left(\sum_k \Sigma_{k,k} + \sum_k \mu_k^2 - \sum_k 1 - \log\left(\prod_k \Sigma_{k,k}\right)\right)$$

$$= \frac{1}{2}\left(\sum_k \Sigma_{k,k} + \sum_k \mu_k^2 - \sum_k 1 - \sum_k \log \Sigma_{k,k}\right)$$

$$\geq \mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)} \mid z_k)\right] - D_{KL}^k\left[q(z \mid x_k^{(i)}) \| p(z_k)\right]$$

$$= \frac{1}{2}\sum_k \left(\Sigma_{k,k} + \mu_k^2 - 1 - \log \Sigma_{k,k}\right)$$

The Covariance Output

$$\geq \mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)}|z)\right] - D_{KL}\left[q(z|x^{(i)})\|p(z)\right]$$

$$q(z|x^{(i)}) \rightarrow (\mu_{z|x}, \widehat{\Sigma}_{z|x})$$

now $q(z|x^{(i)})$ outputs the log covariance

$$\log \Sigma_{k,k} = \widehat{\Sigma}_{k,k}$$

$$= \frac{1}{2}\sum_{k}\left(\Sigma_{k,k} + \mu_k^2 - 1 - \log \Sigma_{k,k}\right)$$

raw covariance is not numerically stable because of underflow

$$= \frac{1}{2}\sum_{k}\left(\exp\left(\widehat{\Sigma}_{k,k}\right) + \mu_k^2 - 1 - \widehat{\Sigma}_{k,k}\right)$$

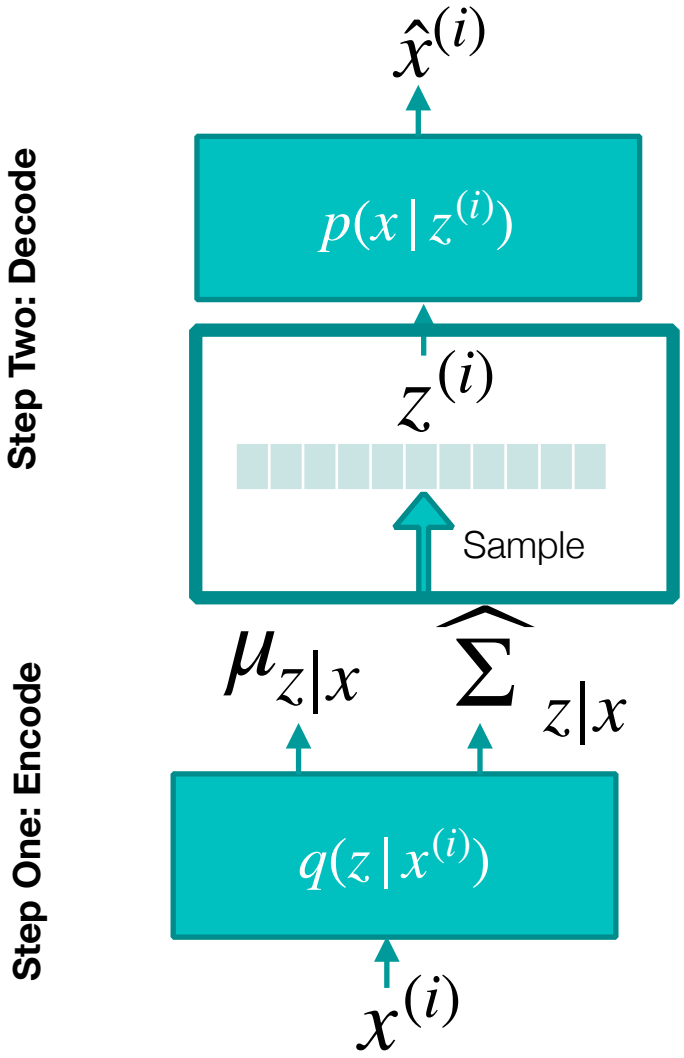
so we will have the neural network output log variance

Also, remember we assume **diagonal covariance**, so z's are not correlated

This means covariance is only a vector of variances (the diagonal of Σ)

Back Propagating

$$\geq \mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)}\,|\,z)\right] - D_{KL}\left[q(z\,|\,x^{(i)})\|p(z)\right]$$



This is partially differentiable by chain rule…

$$\mathcal{N}(\mu_{z|x}, \exp(\widehat{\Sigma_{z|x}})) = z$$

$$= \mu(x^{(i)}) + \exp(\widehat{\Sigma(x^{(i)})}) \cdot \mathcal{N}(0,1)$$

To update q,
we need to back propagate
through sampling layer. How?

The Loss Function Implementation

```
# Encode the input into a mean and variance parameter
z_mean, z_log_variance = encoder(input_img)
# Draw a latent point using a small random epsilon
z = z_mean + exp(z_log_variance) * epsilon

# Then decode z back to an image
reconstructed_img = decoder(z)
# Instantiate a model
model = Model(input_img, reconstructed_img)
```

$\mu(x^{(i)})$ $\widehat{\Sigma(x^{(i)})}$ $q(z|x^{(i)})$

$$z = \mu(x^{(i)}) + \exp(\widehat{\Sigma(x^{(i)})}) \cdot \mathcal{N}(0,1)$$

$\hat{x}^{(i)} = p(x^{(i)}|z)$

```
def vae_loss(self, x, z_decoded):
    x = K.flatten(x)
    z_decoded = K.flatten(z_decoded)
    xent_loss = keras.metrics.binary_crossentropy(x, z_decoded)
    kl_loss = -5e-4 * K.mean(
        1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
    return K.mean(xent_loss + kl_loss)
```

$$-\mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)}|z)\right]$$

$$-\lambda \sum_k 1 + \widehat{\Sigma(x^{(i)})} - \mu(x^{(i)})^2 - \exp(\widehat{\Sigma(x^{(i)})})$$

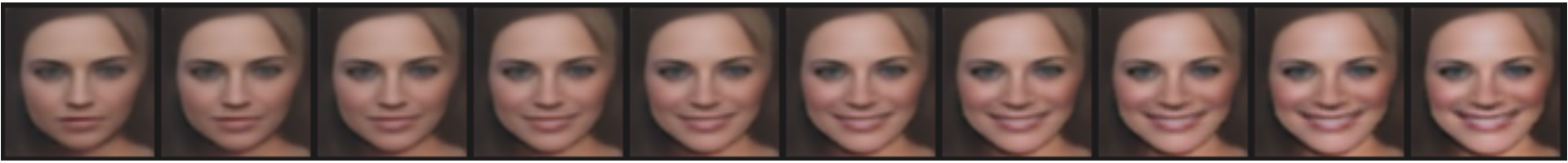
Note: Flipped from maximization to minimization and added lambda for tradeoff in reconstruction, normal latent space

Total Loss Function: Reconstruction error and KL Divergence of the latent space to normal distribution

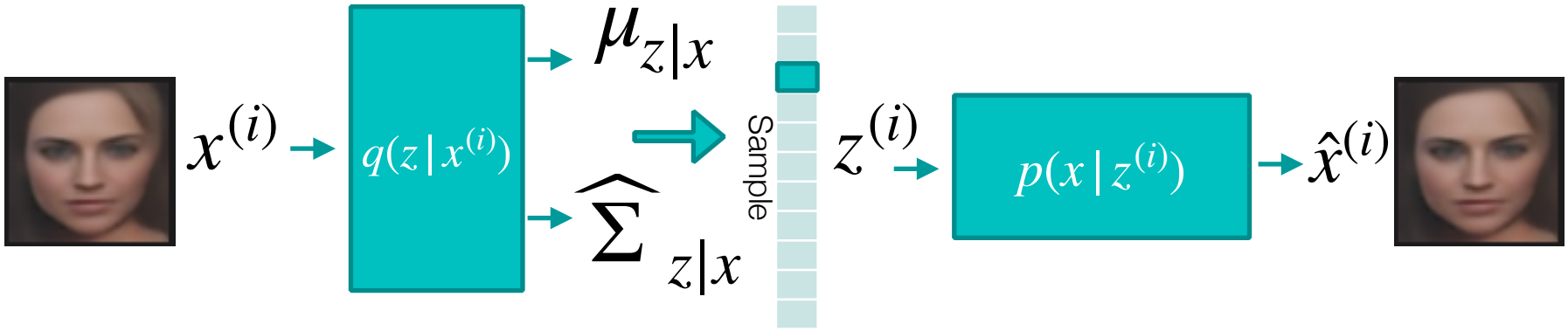
$$= -\mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)}|z)\right] - \lambda \sum_k 1 + \widehat{\Sigma(x^{(i)})} - \mu(x^{(i)})^2 - \exp(\widehat{\Sigma(x^{(i)})})$$

Now that its trained, so what?

Encoding faces, then adjust the “z” that relates to smiling.



Investigate what happens by moving around each z_i

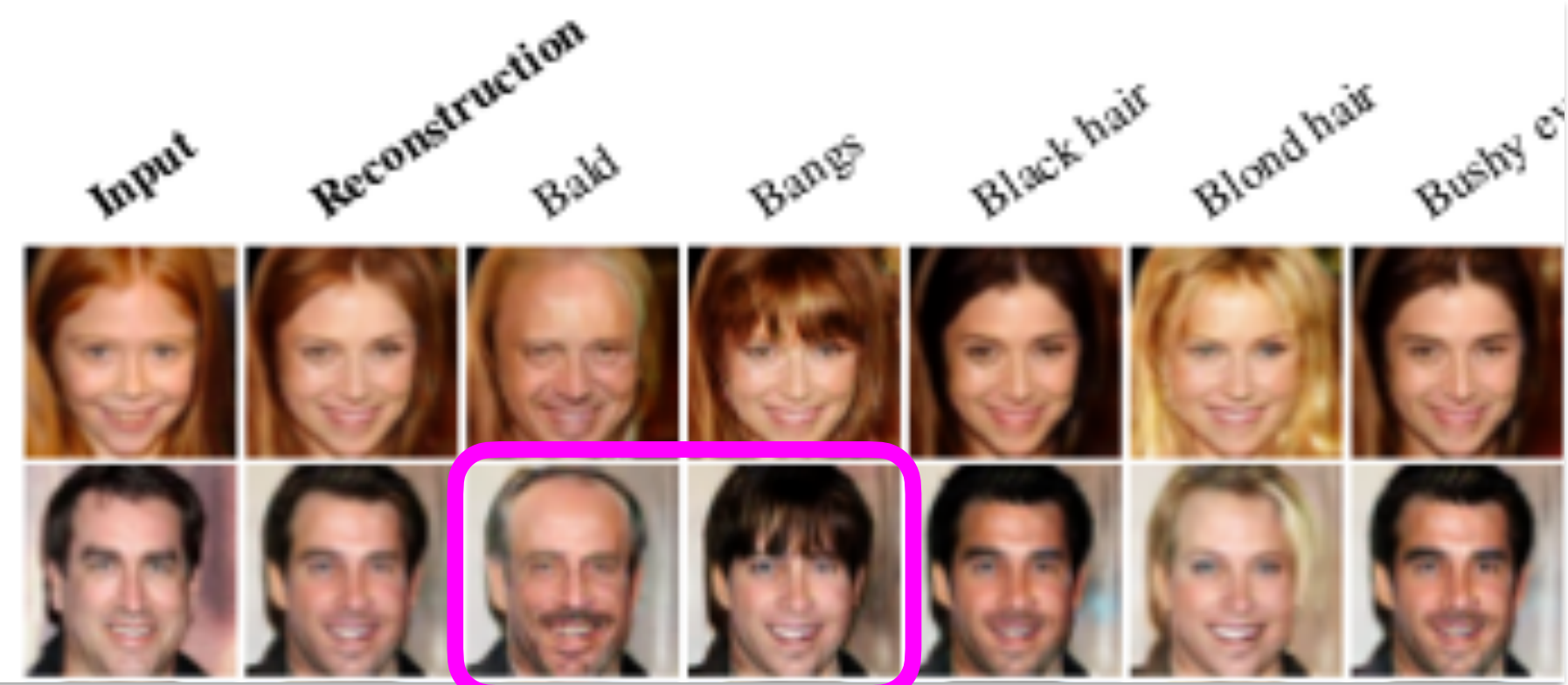


VAE Examples

Encoding faces, then adjust the “z” that relates to smiling.

VAE Examples

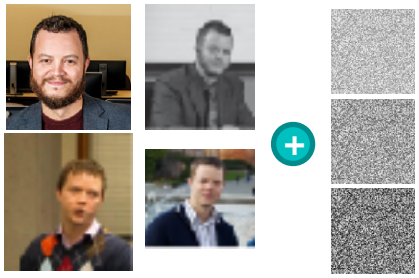
Different, automatically found z, latent variables



Stable Diffusion

The Diffusion Process, Simplified

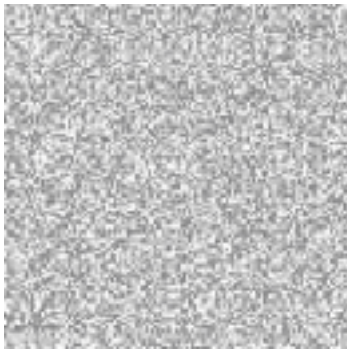
- ***Guiding*** Example: Predict noise sample in an image



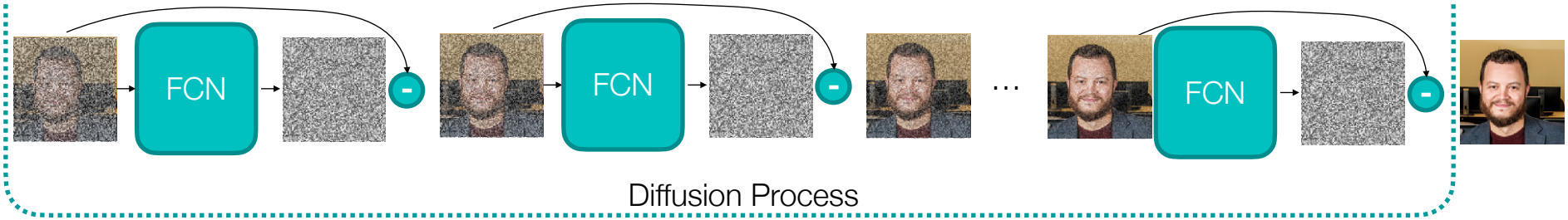
Create Dataset

Add in "t" steps of noise

Full Convolutional Network, Noise Predictor



Train Network on Synthetic Examples, various values of "*t*"



Diffusion Process

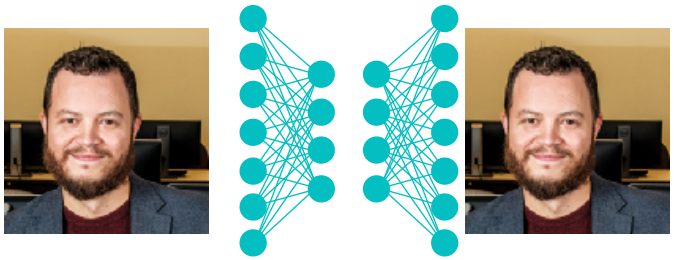
- Now we could generate great looking images from noise!!

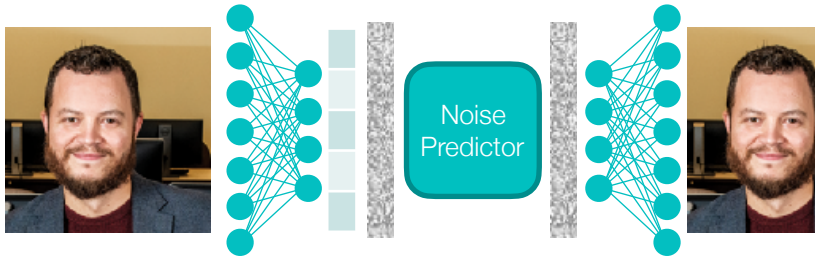
Departure to Latent Space

Departure to Latent Space Our approach starts with the analysis of already trained diffusion models in pixel space: Fig. 2 shows the rate-distortion trade-off of a trained

Rombach et al., 2022, https://arxiv.org/pdf/2112.10752.pdf

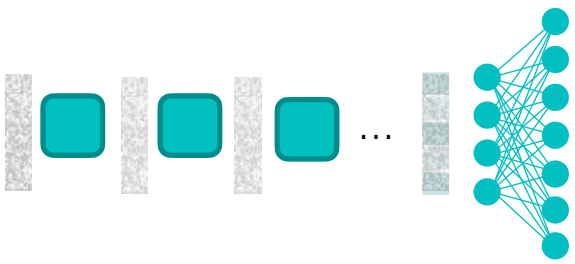
- Start with a nice VAE
- Train noise prediction in latent space
- Perform diffusion in latent Space
- Generate from VAE decoder



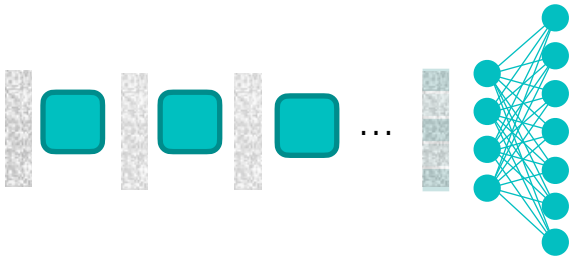


Random Noise

$z \sim \mathcal{N}(0,I)$



$z \sim \mathcal{N}(0,I)$



Eric Larson, Brown University

Eric Larson, Disney Animator

Lecture Notes for

Neural Networks and Machine Learning

Stable Diffusion

Next Time:
Reinforcement Learning

Reading: None