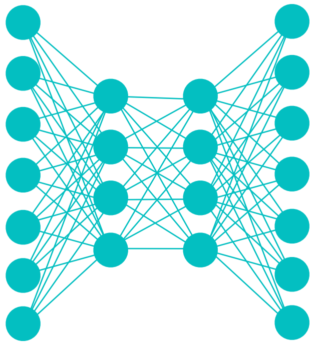
Lecture Notes for

Neural Networks and Machine Learning

Generative Networks and Auto-Encoding Generators

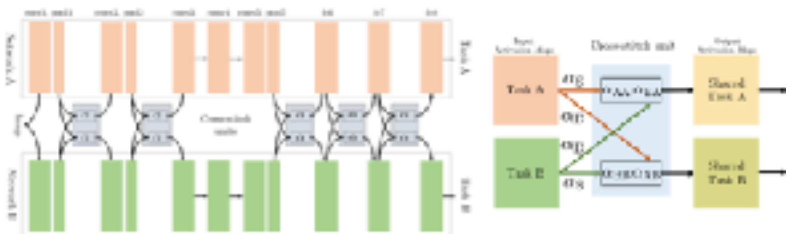


Logistics and Agenda

- Logistics
  - Lab three due date pushed back (see schedule)
- Agenda
  - A historical perspective of generative Neural Networks
  - Variational Auto-Encoding
  - VAE in Keras Demo (if time)
  - Adversarial Auto-Encoders

Last Time

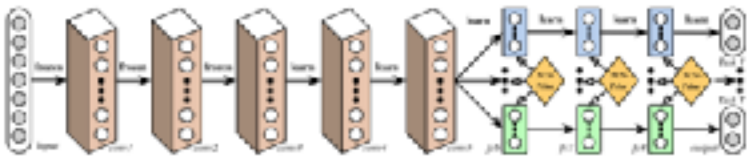
Multi-task: Cross Stitch Networks



- Only works for simultaneous multi-label problems
  - like semantic segmentation and surface normal segmentation (clustering similarly facing objects)
- Take a learned weighted sum of the activations
- Works a little better than single task, but no worse

https://arxiv.org/pdf/1604.03539.pdf

Multi-task: Deep Relationship Networks



- Start training traditionally
- Minimize Kronecker Product between fully connected task specific layers
  - that is, make Covariance between layers close to identity
  - encourages feature maps in each task to be **less correlated** to feature maps of another task

https://arxiv.org/pdf/1506.02117.pdf

Multi-Task Learning in Keras with Multi-Label Data

Fashion week, colors and dresses

"finish demo"

Follow Along: https://www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/

Multi-Task Learning

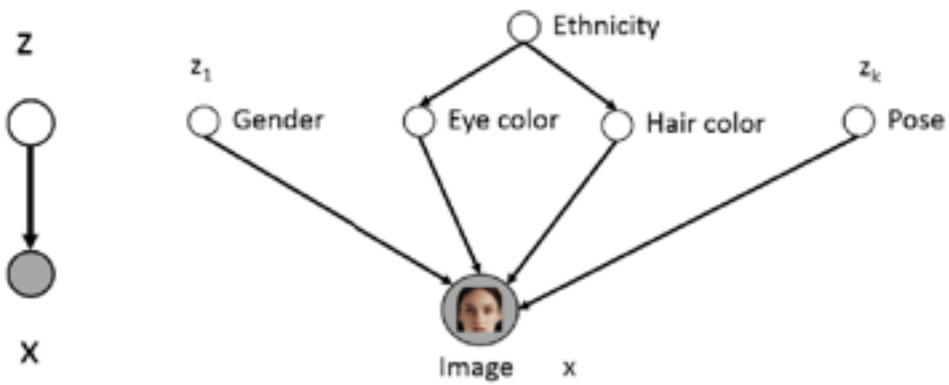
School Data, Computer Surveys

"finish demo"


Luke Wood

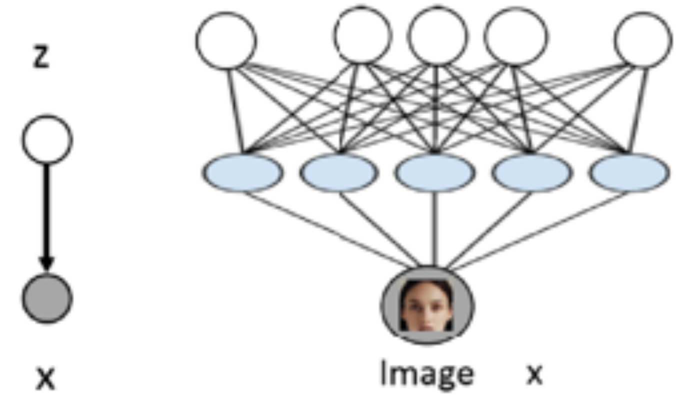
Follow Along: LectureNotesMaster/05 LectureMultiTask.ipynb

Motivations: Generative Latent Variables



$$p(\mathbf{x} \,|\, \mathbf{z})$$

Hard: **z** is expertly chosen



$$p(\mathbf{x} \,|\, \mathbf{z})$$

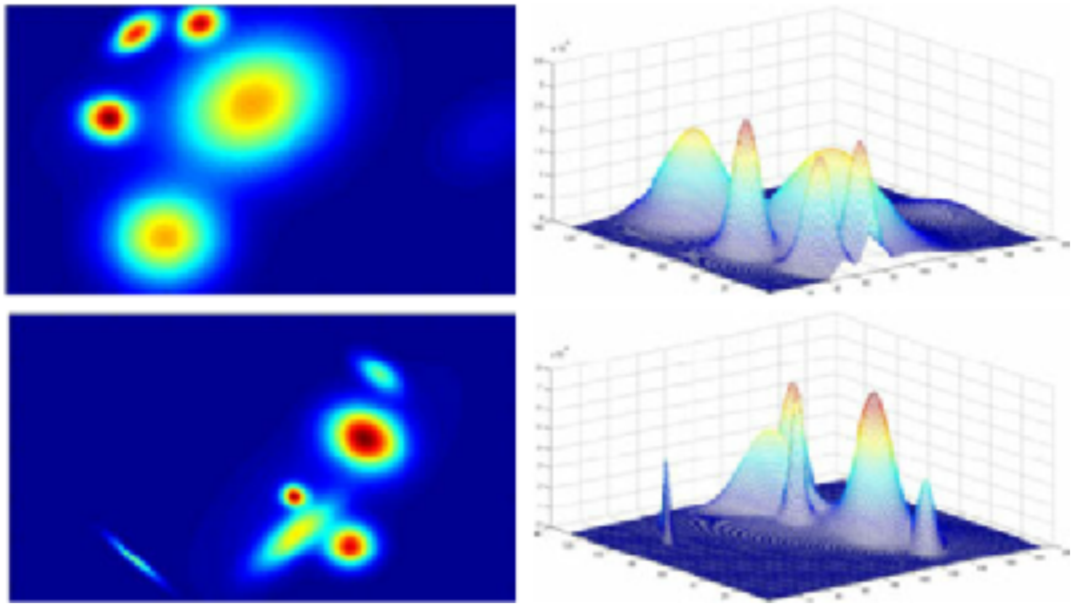
Not as Hard: **z** is trained,
latent variables are uncontrolled

Want: $$p(\mathbf{x}) \approx \sum_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x} \,|\, \mathbf{z})$$

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

Motivation: Mixtures for Simplicity

Want: $$p(\mathbf{x}) \approx \sum_{\mathbf{z}} p(\mathbf{z}) p_\theta(\mathbf{x} \,|\, \mathbf{z})$$



- Each latent variable is mostly independent of other latent variables
- The sum of various mixtures can approximate most any distribution
- Good choice for conditional is Normal Distribution
- Can parameterize $p(x|z)$ to be a Neural Network

$$p_\theta(\mathbf{x} \,|\, \mathbf{z} = k) = \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

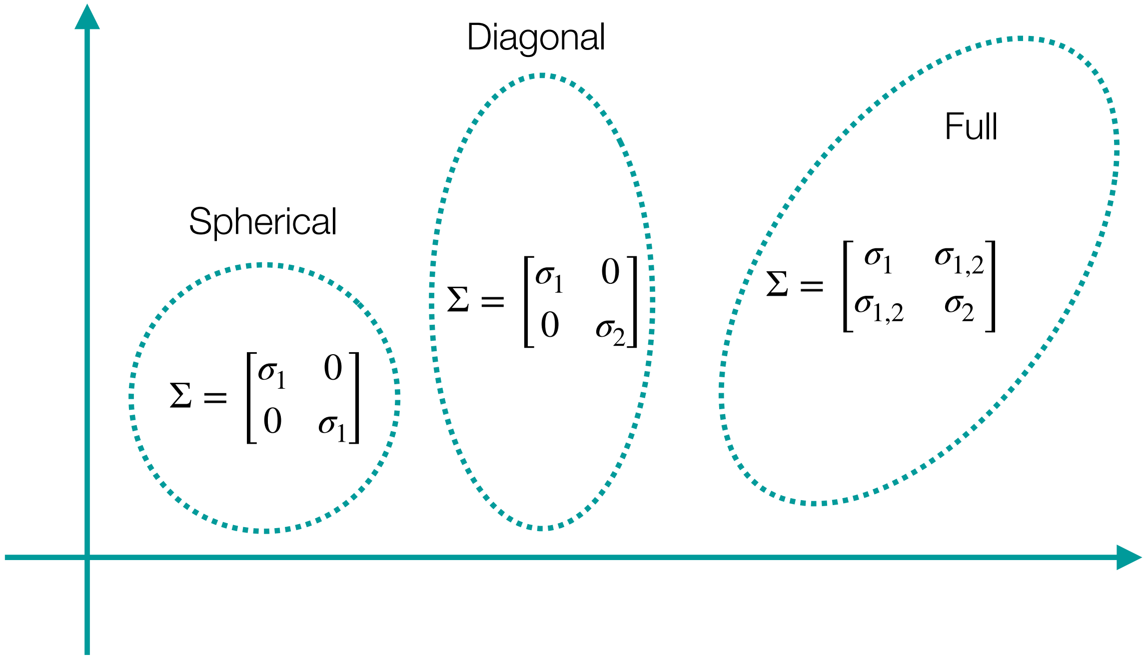
mean and covariance learned

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

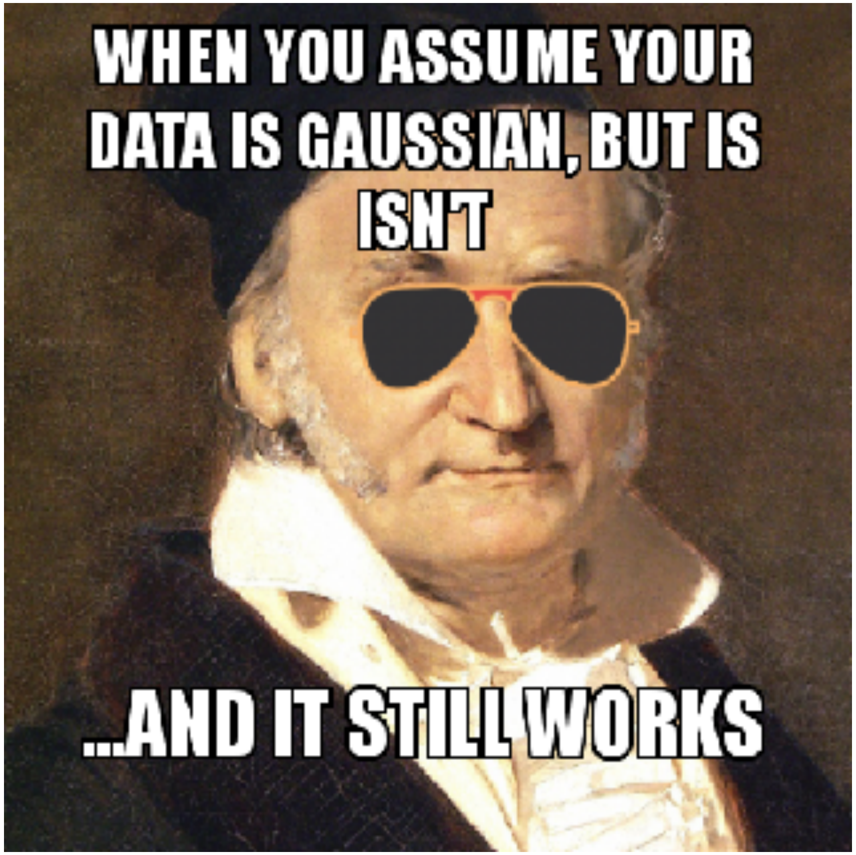
Motivation: Mixtures for Simplicity

$$= \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

mean and covariance learned



A History of Generative Networks



Aside: Notation

some function

could be neural networks

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

Expected value of f under conditional distribution, q

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

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

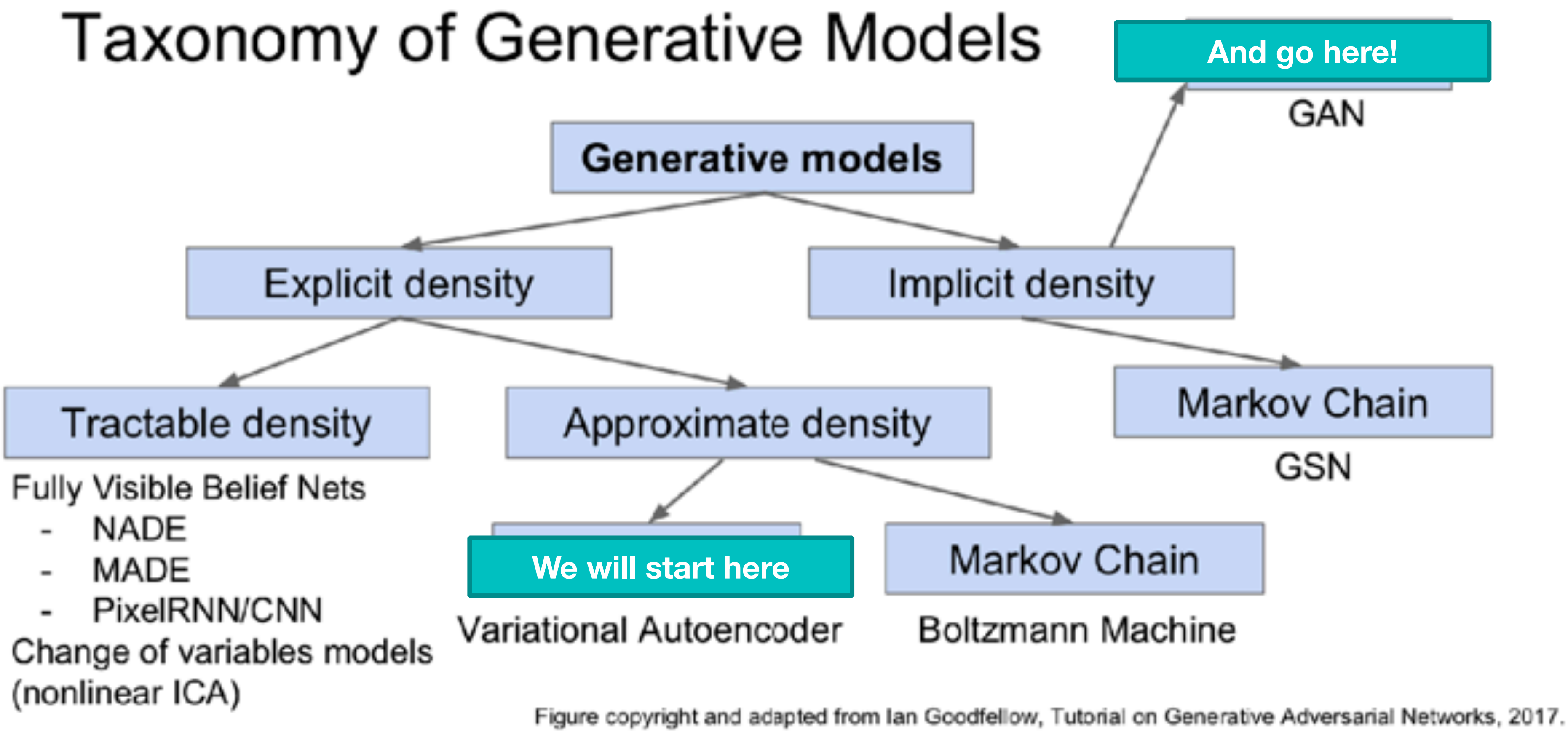
If function is a probability, this is just the negative of cross entropy of distributions:

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

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

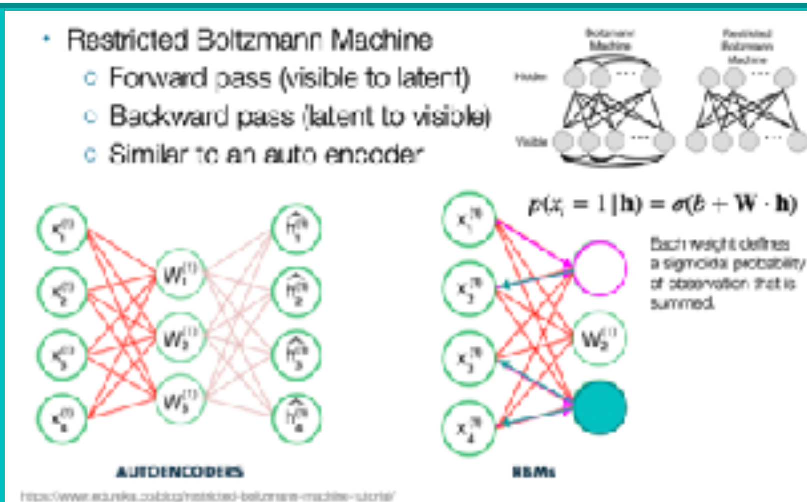
$$D(p||q) = \sum_{x} p(x) \cdot \log\left(\frac{p(x)}{q(x)}\right) = \mathbf{E}_p\left[\log\left(\frac{p(x)}{q(x)}\right)\right]$$

Taxonomy of Generative Models

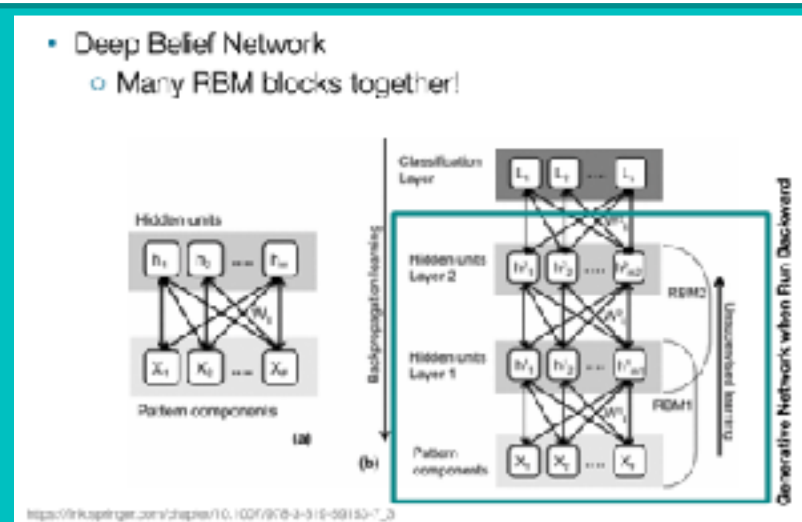


http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

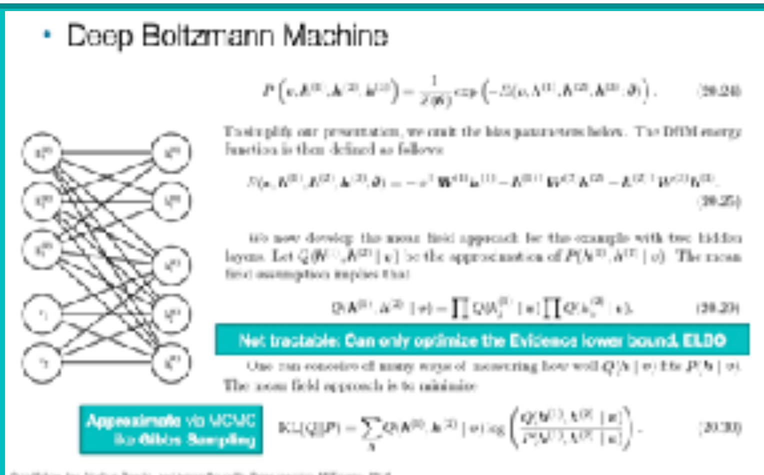
Abridged History of Generative Networks



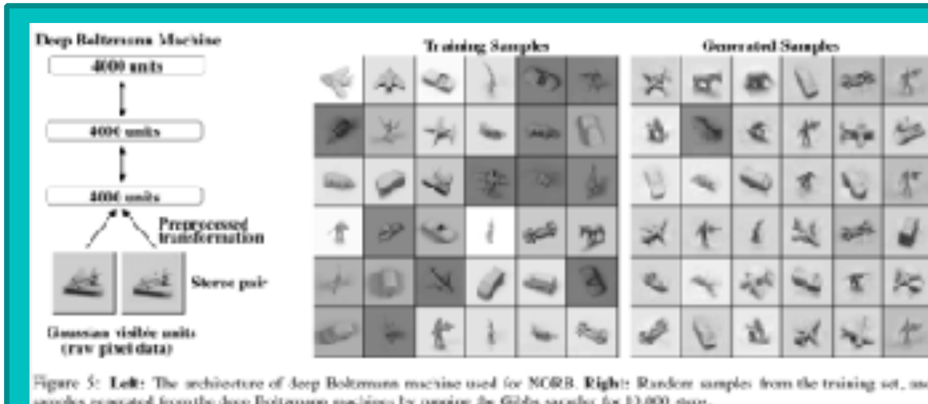
2006 Restricted Boltzmann Machine



2007, Deep Belief Networks
RBMs with many layers



2009 Deep Boltzmann Machine
Goodfellow, Bengio, Courville

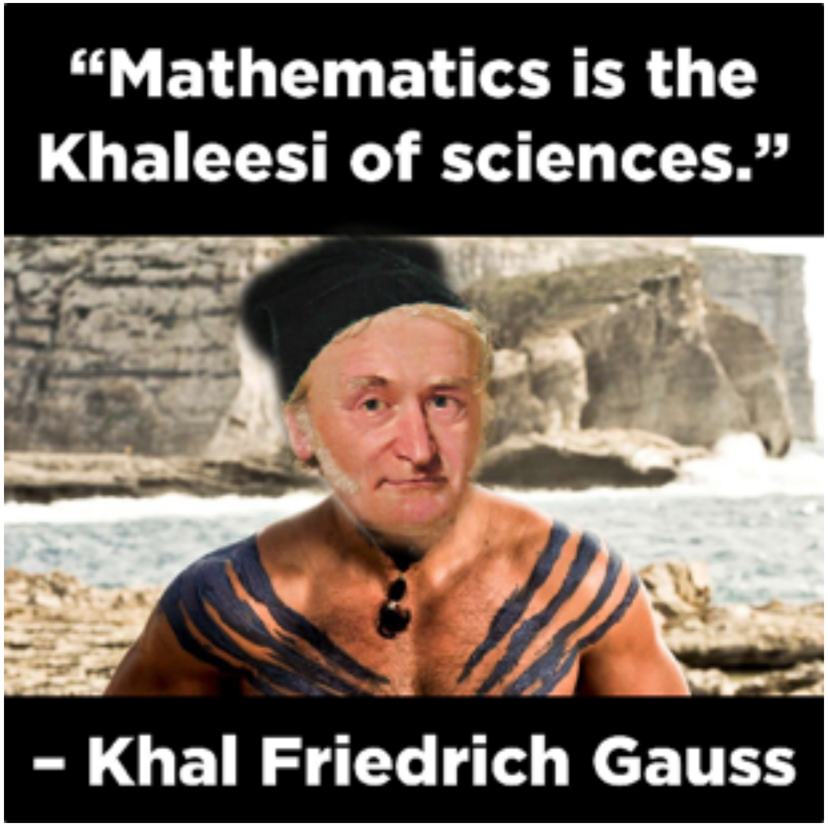


2009, Practical Examples
Salakhutdinov and Hinton

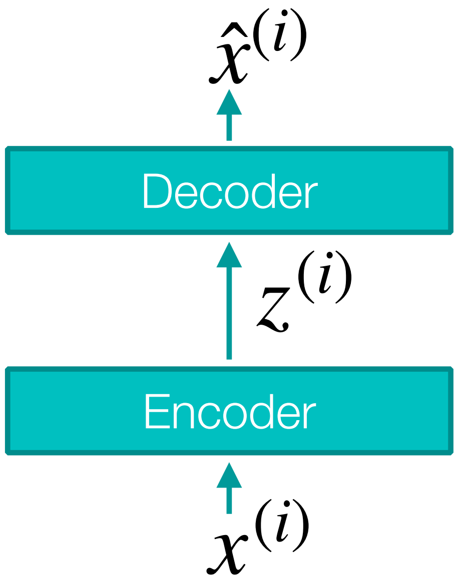
Contemporary Modeling

- DBNs and DBMs did not become very popular
    - Mathematics detracts from popular understanding
    - Often methods using sampling are not scalable
    - Cannot directly use Gradients (no Back Prop) 😢
- Popular method for calculating generative networks with Evidence Lower Bound (ELBO) approximation:
    - Variational Auto Encoding
        - Guaranteed NOT to find global minimum
        - But scalable and will converge in finite time

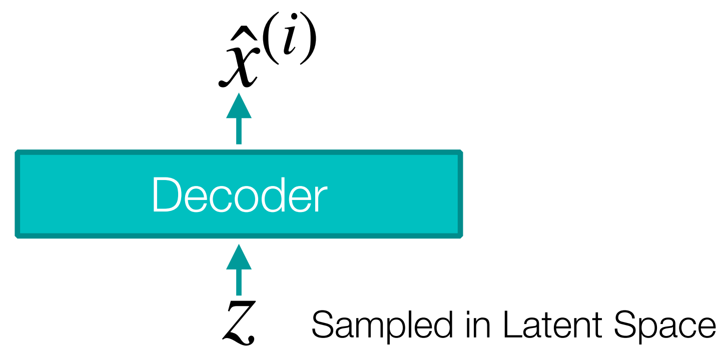
Variational Auto Encoding



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
- Learned space is not continuous
- Features could be highly correlated, related in complex ways
  - So, how should we sample from the latent space?
- Need to define some constraints on the latent space…

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

Reasonable constraints for p(z)?

$\hat{x}^{(i)}$

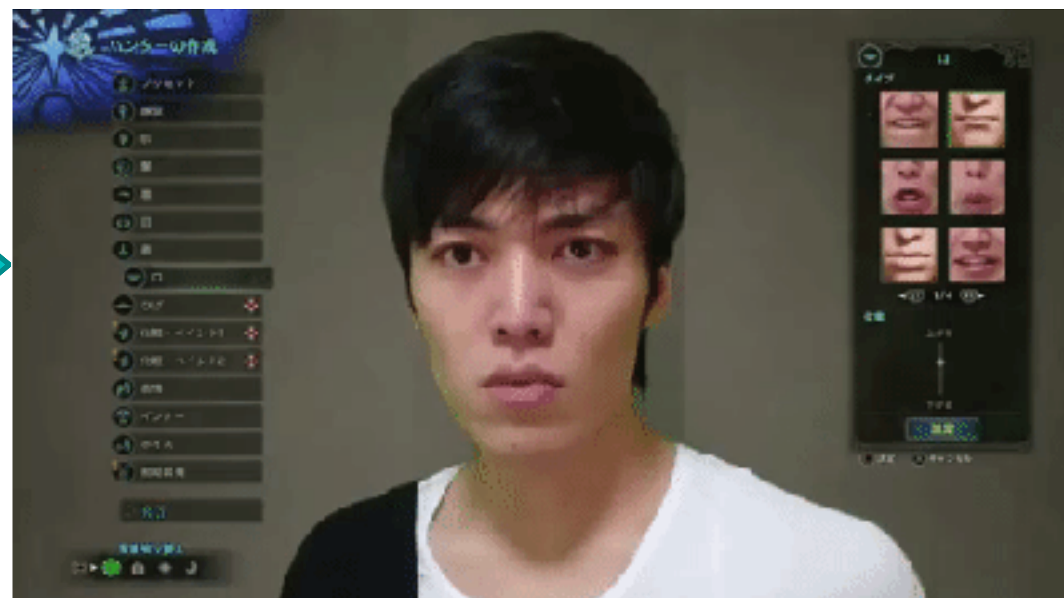
Decoder

z Sampled in Latent Space

- Should be simple, easy to sample from: **Normal**
- Each component should be i.i.d.: **Diag. Covariance**
  - Encourages features that may be semantic, like expert might select

Ideal:
Smilling
Hair Color
Skin Color
Eye Color
…
Nose Position

z → Transform with Decoder → $\hat{x}^{(i)}$



Optimizing

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

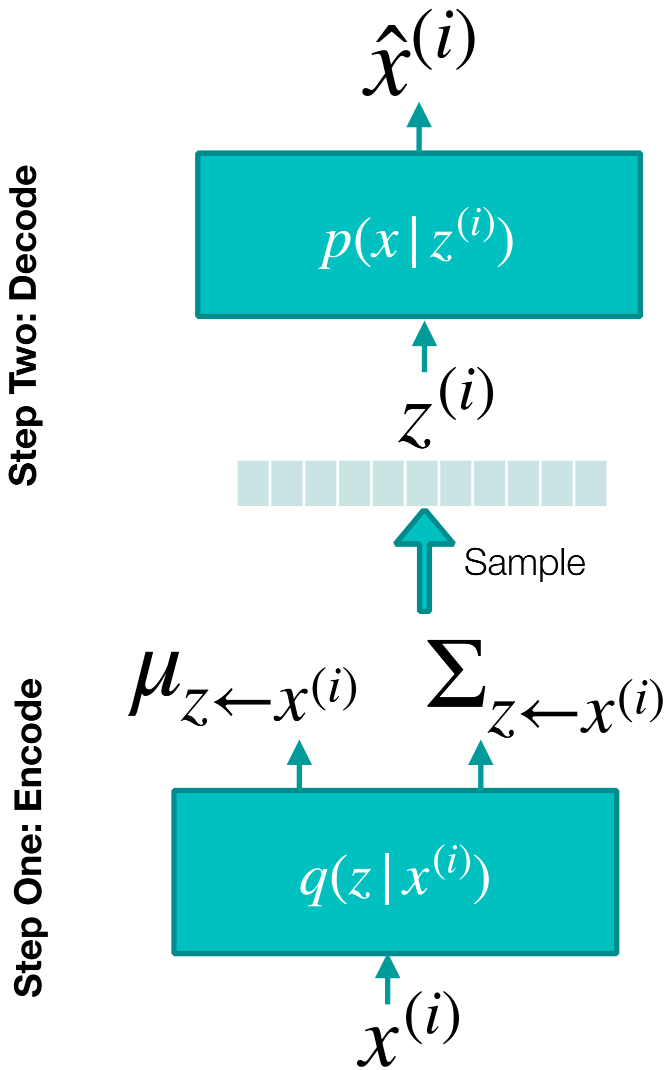
We need this inference in order to compute latent variable

$$p(x) = \int p(x|z)p(z)dz$$

Denominator is of this form

- We can't compute! **Intractable computation** for all "z"
- So let's define this with **variational inference**:
  - Only needs to work for z **with observed** $x^{(i)}$
  - 1. **Encode** observed $x^{(i)}$ using network $q(z \mid x^{(i)})$,
  - 2. Use $q(z \mid x^{(i)})$ to sample z appropriately, then **decode** with another neural network, $p(x^{(i)} \mid z^{(i)})$
  - 3. Use bounds to make $q(z \mid x^{(i)})$ as close as possible to what we think $p(z \mid x)$ might look like

Need a new formulation



Step Three: Make conditional p and q Similar

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

Step Four: Simplify Optimization

we can manipulate step three to yield the following approximation:

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

Intuition MLE: maximize probability of observed $x^{(i)}$ given function q

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

Need a new formulation

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

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

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

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

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

always non-negative

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

What have we really done here? Could we have motivated this differently?

The Loss Function

Maximize through
Error of Reconstruction
Same as minimizing cross entropy

want $p(z)$ to be $\mathcal{N}(\mu = 0, \Sigma = I)$
because it makes nice latent space

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

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

Can get this by manipulating the KL for normal distribution

Determinant of diagonal matrix is simple. Motivates diagonal covariance…

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

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

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

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

The Covariance Output

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

Maximize through
Error of Reconstruction
Same as minimizing cross entropy

want $p(z)$ to be $\mathcal{N}(\mu = 0, \Sigma = I)$
because it makes nice latent space

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

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

raw covariance is not numerically stable because of underflow

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

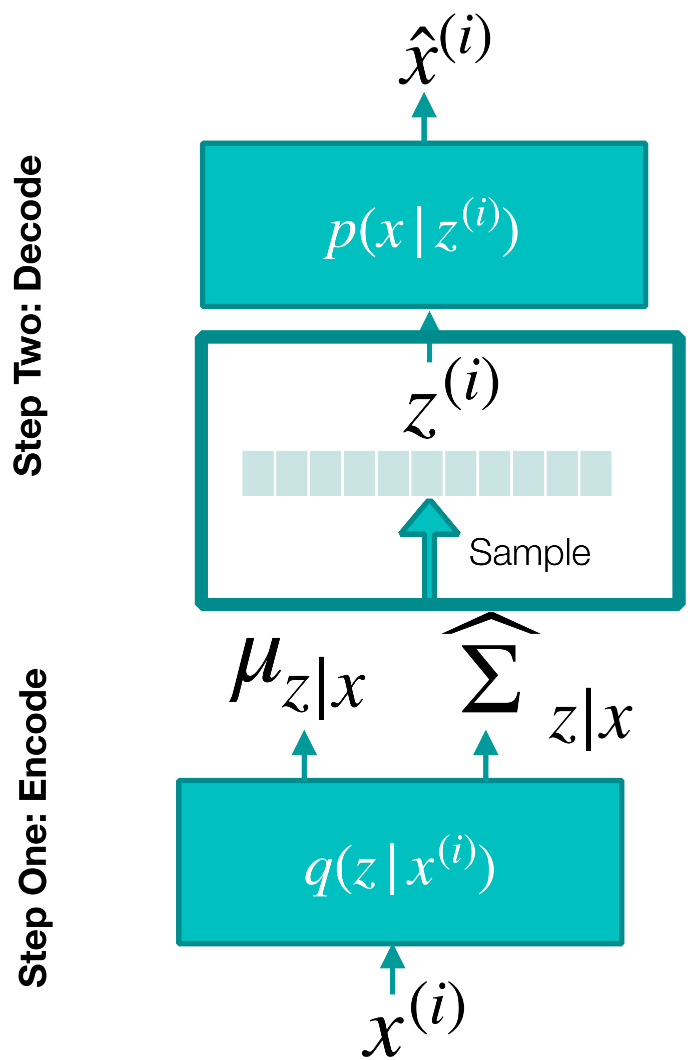
predicted by
$q(z\,|\,x^{(i)})$

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

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



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

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)
```

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

```
# Draw a latent point using a small random epsilon
z = z_mean + exp(z_log_variance) * epsilon
```

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

```
# Then decode z back to an image
reconstructed_img = decoder(z)
```

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

```
# Instantiate a model
model = Model(input_img, reconstructed_img)
```

```
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

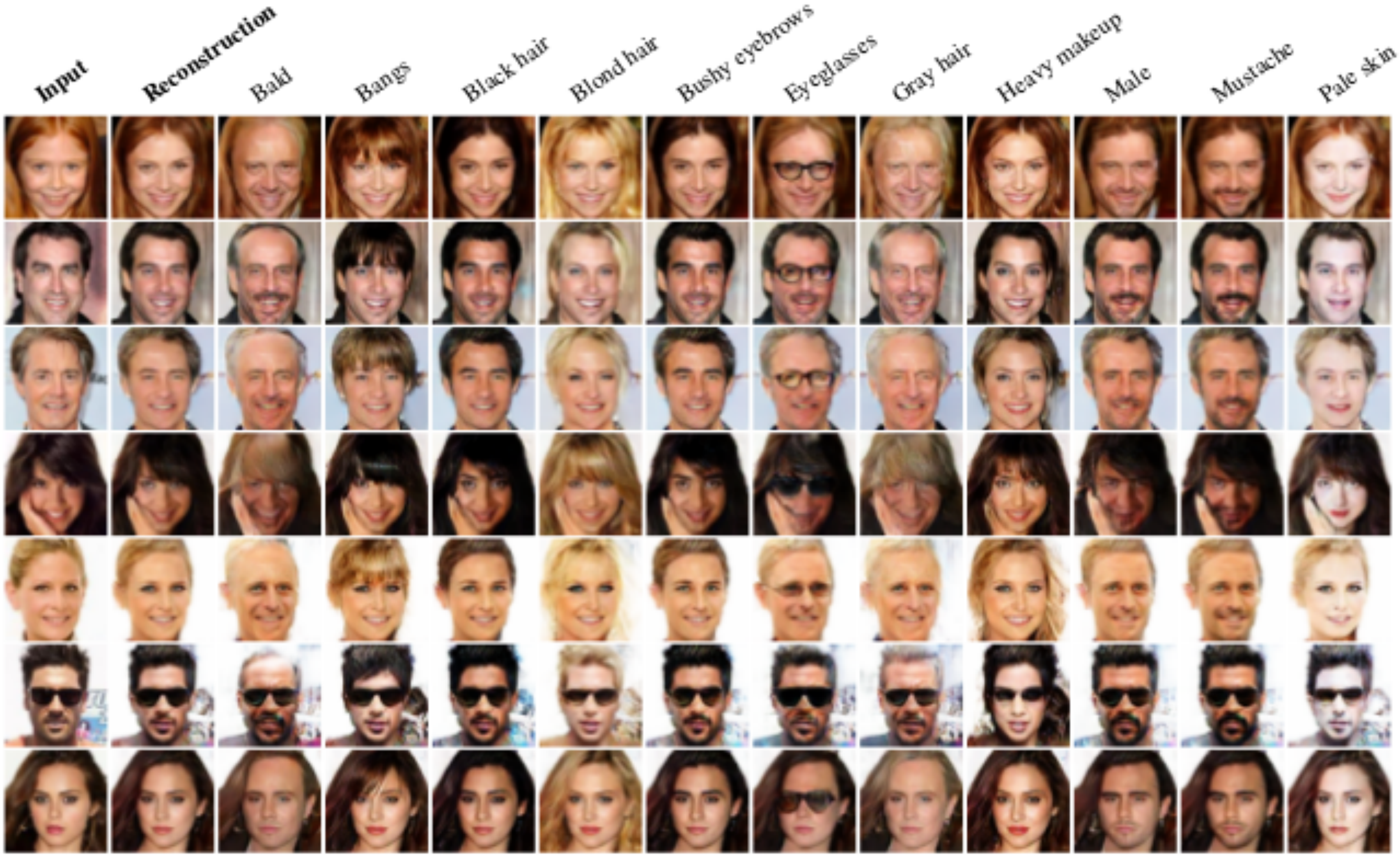
$$= -\mathbf{E}_{q(z|x^{(i)})}\left[\log p(x^{(i)}\,|\,z)\right] - \lambda \sum_k 1 + \widehat{\Sigma(x^{(i)})} - \mu(x^{(i)})^2 - \exp(\widehat{\Sigma(x^{(i)})})$$

VAE Examples

Encoding faces, then finding the "z" that relates to smiling.

VAE Examples

Different, automatically found z, latent variables



VAEs in Keras

Sampling from variational auto encoder using MNIST

Demo by Francois Chollet

In Master Repo: 07a VAEs in Keras.ipynb

Follow Along: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/8.4-generating-images-with-vaes.ipynb