Motivation: Interpolating Faces, Generating Faces

Figures from http://casser.io/files/autoencoders.html
and https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Autoencoder

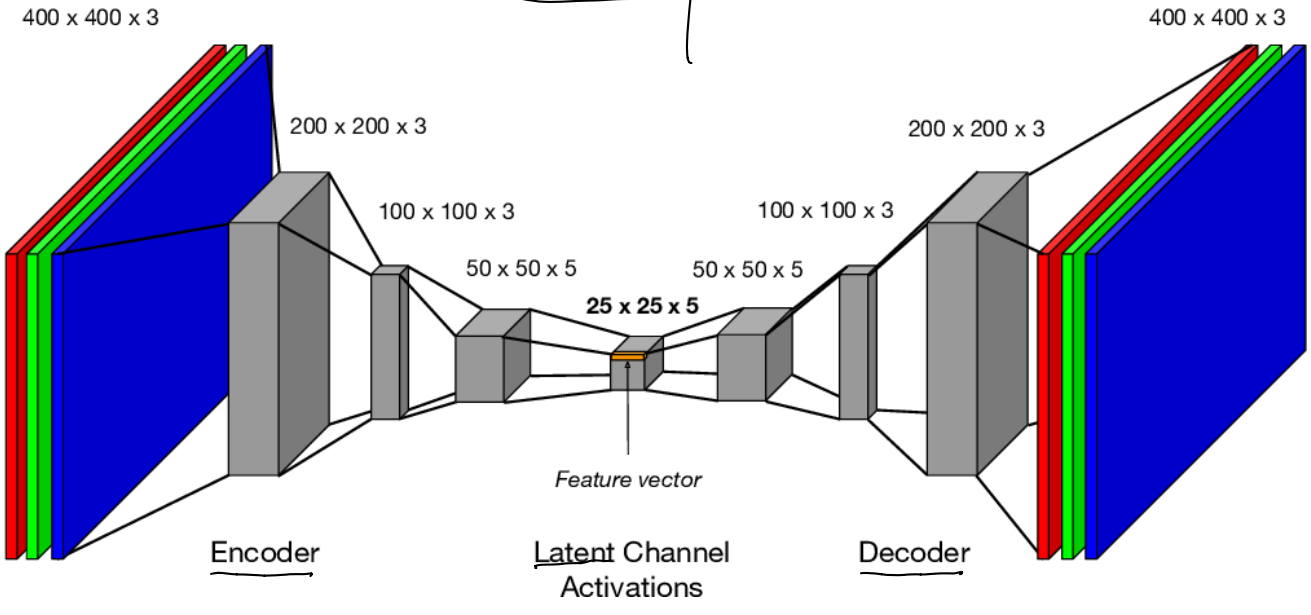
Can we develop a reduced encoding of an image that would let us reconstruct the original image?

Input x

Encoding
just a few
numbers!

Output $\tilde{x}$

CNN Autoencoder

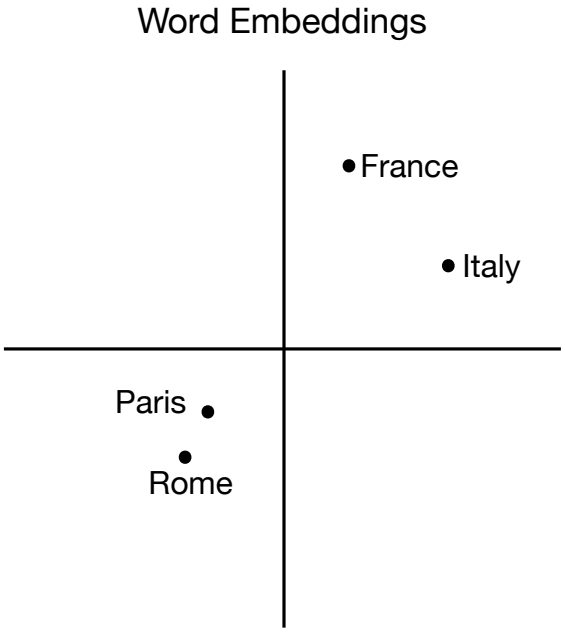


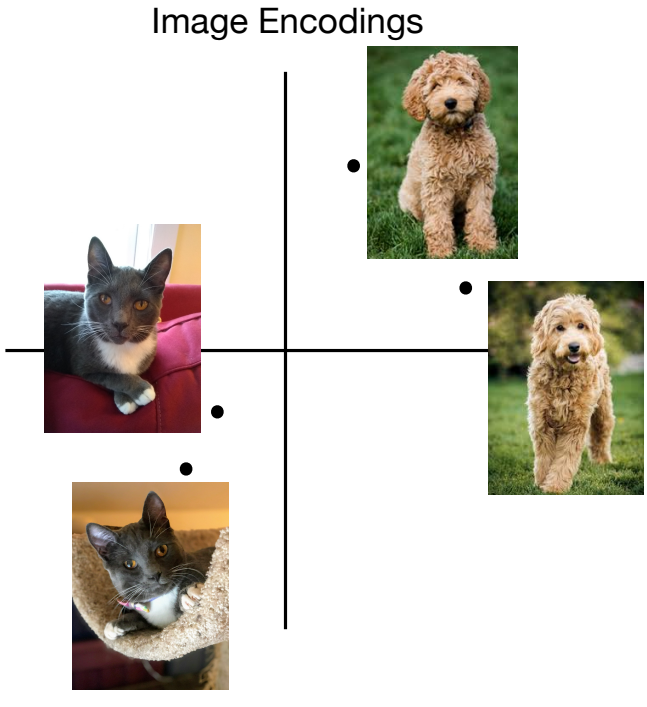
convolutional layers

transposed convolutional layers

Figure from Flasphler et al. "Feature discovery and visualization of robot mission data using convolutional autoencoders and Bayesian nonparametric topic models" (2017)

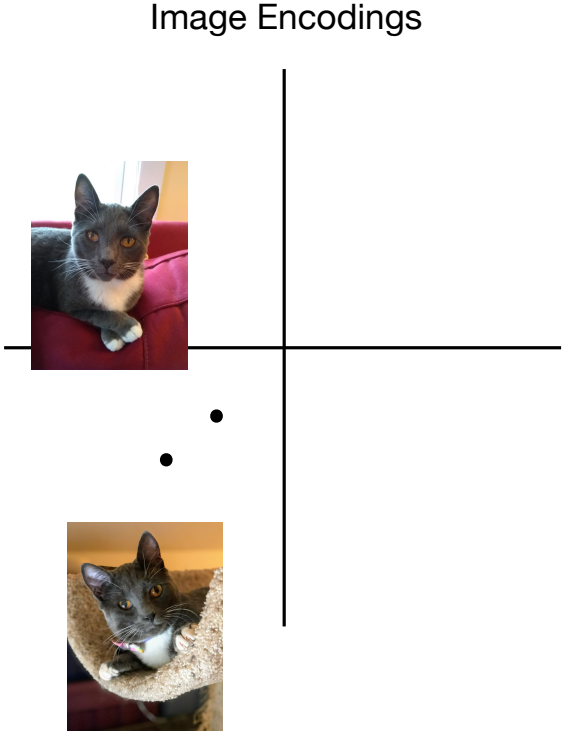
Why do This? Analogy to Word Embeddings





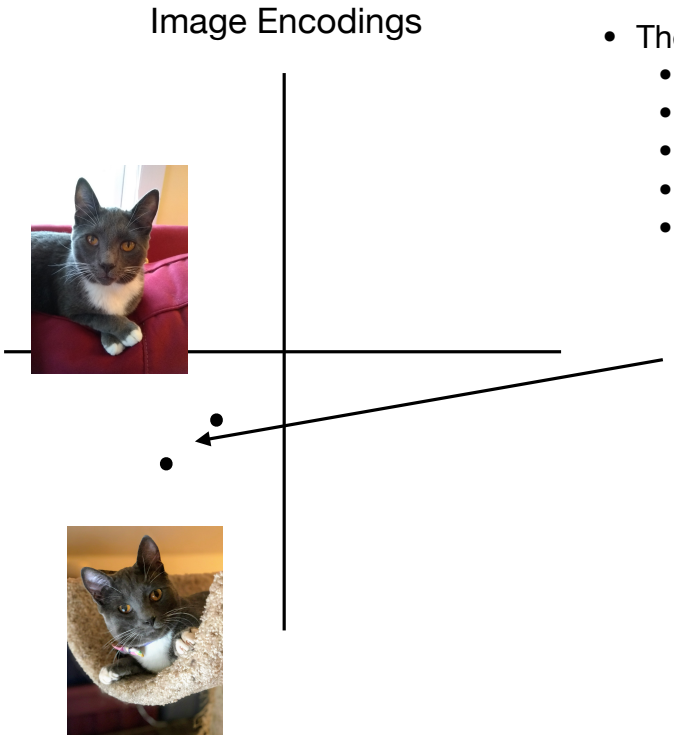
Dog pictures from http://www.ridgelinelabradoodles.com/breed-info.html

Expressing Uncertainty via a Normal Distribution



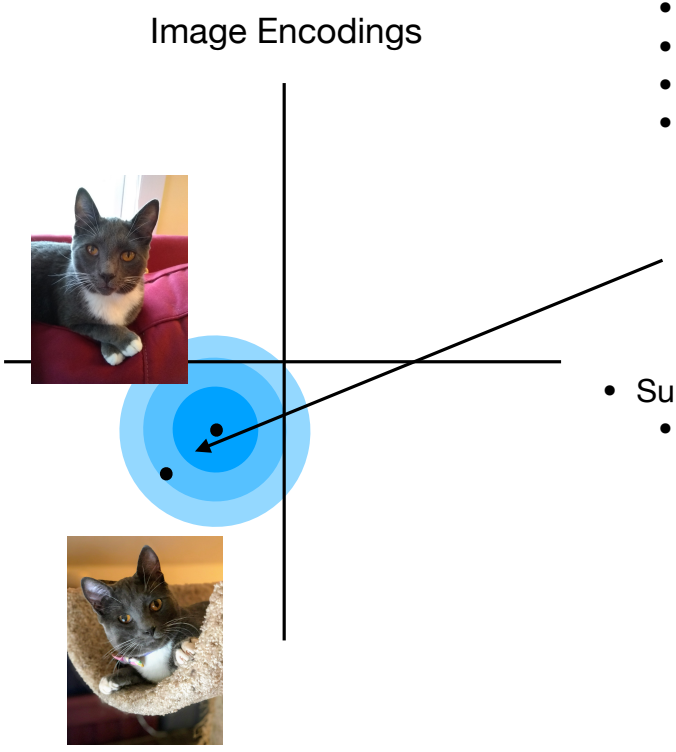
- The latent encoding space should be **continuous**
  - There is a space where Benedict lives
  - His neighbors are other grey and white Tuxedo cats,
  - Then still nearby are other cats
  - Dogs are a little further away
  - The Eiffel Tower is in a completely different location

Expressing Uncertainty via a Normal Distribution



- The latent encoding space should be **continuous**
  - There is a space where Benedict lives
  - His neighbors are other grey and white Tuxedo cats,
  - Then still nearby are other cats
  - Dogs are a little further away
  - The Eiffel Tower is in a completely different location

Expressing Uncertainty via a Normal Distribution



- The latent encoding space should be **continuous**
  - There is a space where Benedict lives
  - His neighbors are other grey and white Tuxedo cats,
  - Then still nearby are other cats
  - Dogs are a little further away
  - The Eiffel Tower is in a completely different location

This point should correspond
to another picture of Benedict

- Suppose my input was the upper picture of Benedict.
  - The blue circles are contours of a bivariate normal distribution representing my state of knowledge about where this picture of Benedict really exists in the latent space.

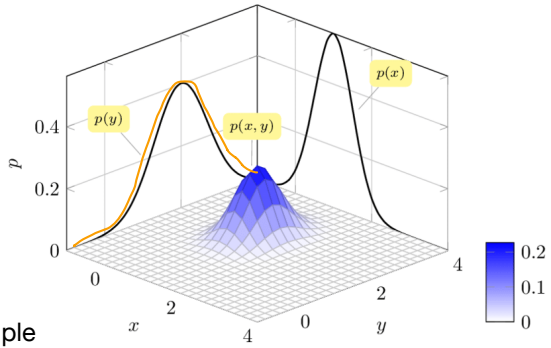
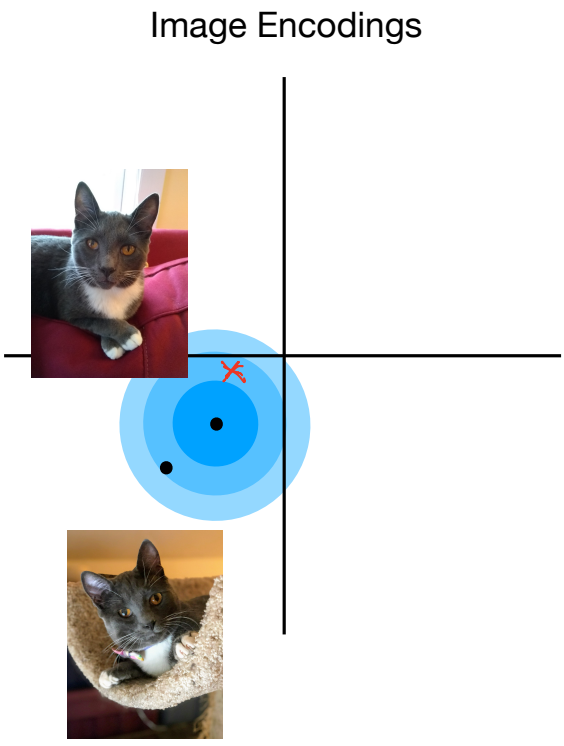


Figure from Selvan “Bayesian tracking of multiple point targets using Expectation Maximization” (2015)

Expressing Uncertainty via a Normal Distribution



- The blue circles are contours of a bivariate normal distribution representing my state of knowledge about where this picture of Benedict really exists in the latent space.

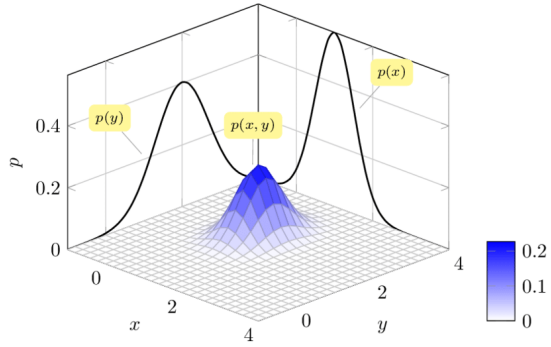
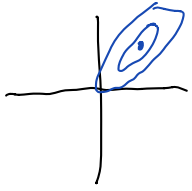
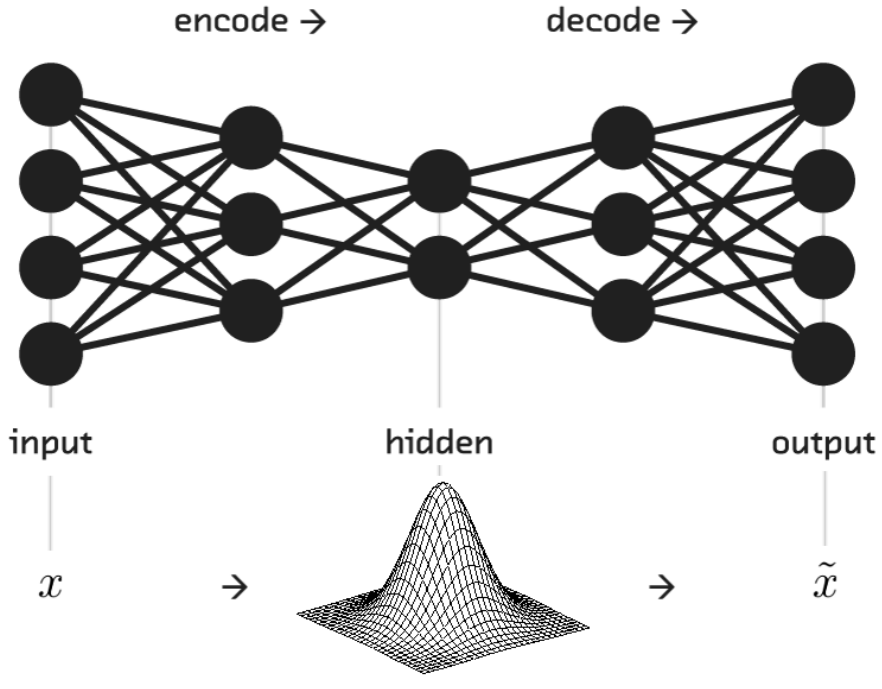


Figure from Selvan “Bayesian tracking of multiple point targets using Expectation Maximization” (2015)

- To specify this distribution, we need 4 parameters:
  - Mean along horizontal axis
  - Mean along vertical axis
  - Standard deviation along horizontal axis
  - Standard deviation along vertical axis
  - (optional fifth parameter: correlation; we ignore this)



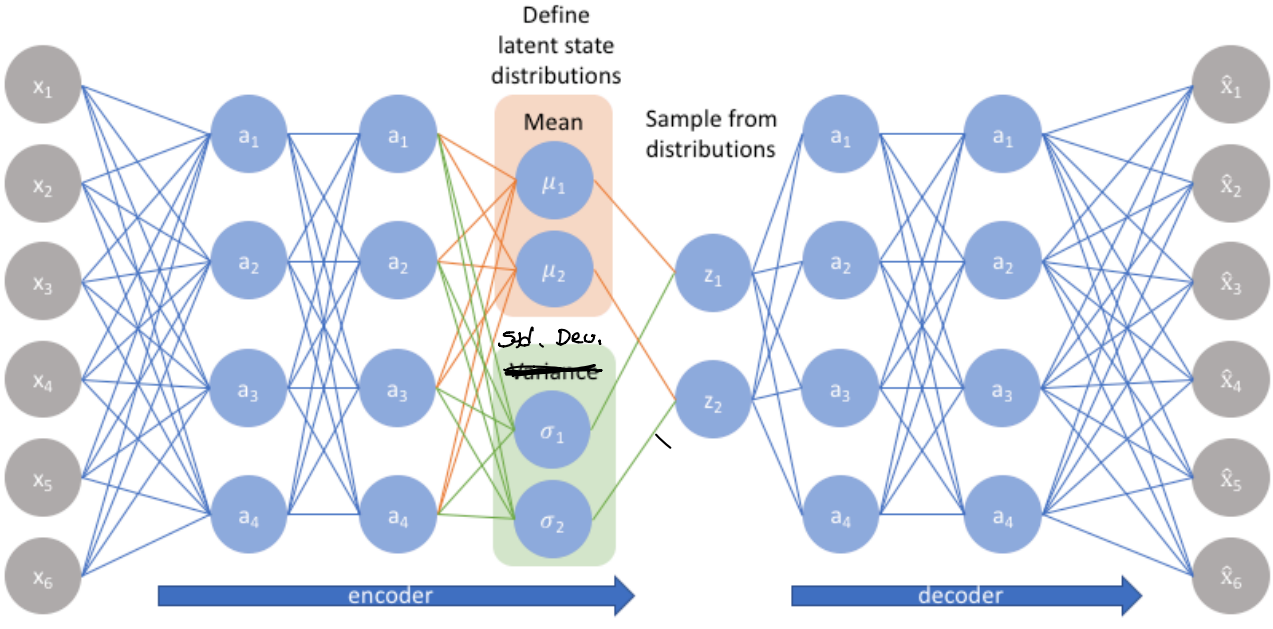
Variational AutoEncoders (VAE)



Sparse encoding of x with uncertainty
All points in this region should generate a similar $\tilde{x}$.
`

Figure from https://towardsdatascience.com/what-the-heck-are-vae-gans-17b86023588a

Variational AutoEncoders (VAE)



- Encoder network outputs parameters of the normal distribution!
- We draw a sample from that normal distribution (a location in the latent space)
- We feed that into the decoder network and ask it to recreate the original input

Figure from https://www.jeremyjordan.me/variational-autoencoders/

Sampling from a Bivariate Normal

If $\varepsilon \sim \text{Normal}(0,1)$
then if
$Z = \mu + \sigma\varepsilon$,
$Z \sim \text{Normal}(\mu, \sigma^2)$

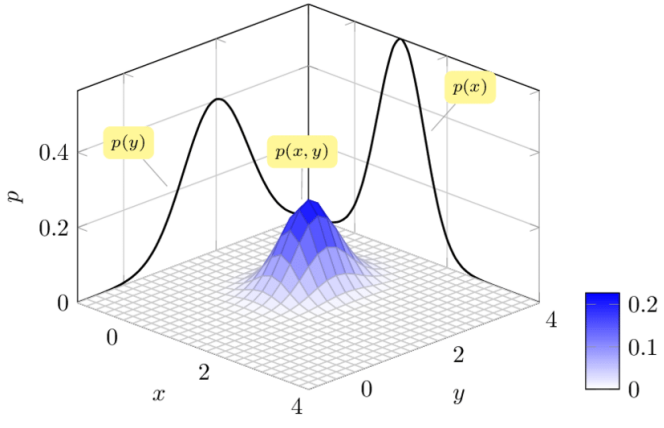


Figure from Selvan "Bayesian tracking of multiple point targets using Expectation Maximization" (2015)

- Sample $\varepsilon_1 \sim$ **Normal**$(0,1)$ and $\varepsilon_2 \sim$ **Normal**$(0,1)$
- Set $z_1 = \mu_1 + \sigma_1\varepsilon_1$ and $z_2 = \mu_2 + \sigma_2\varepsilon_2$

- Note: σ_1 and σ_2 must be positive! Enforce this with a transformation.
  - Model actually outputs a_1, and we set $\sigma_1 = \exp(a_1)$

Loss for Variational AutoEncoders

- Two networks generating different outputs; each has a contribution to loss

- Start with decoder network:
  - It generates an output that is supposed to look similar to the input (e.g. pixel values for an image)
  - Use whatever loss appropriate to data type
  - For example, if outputting numeric values for pixels, mean squared error

$$J^{(i),Decoder}(b,w) = \underbrace{\frac{1}{n_h n_w n_c} \sum_{i=1}^{n_h} \sum_{j=1}^{n_w} \sum_{c=1}^{n_c} (x_{i,j,c}^{(i)} - \tilde{x}_{i,j,c}^{(i)})^2}_{\text{reconstruction loss}}$$

- Encoder network:
  - Derivation beyond the scope of this class (approximation to Bayesian posterior distribution)

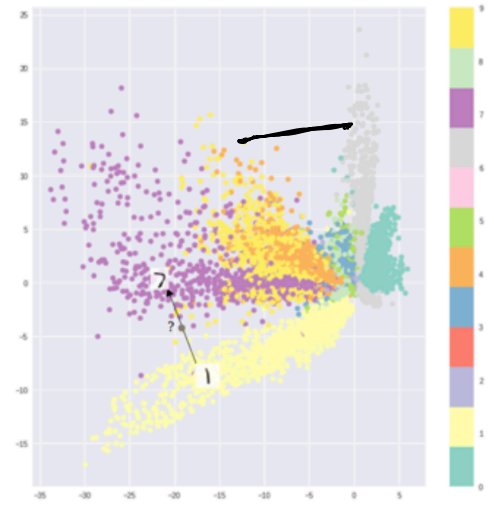
$$J^{(i),Encoder}(b,w) = \frac{1}{2} \sum_{d=1}^{D} \left[(\mu_d^{(i)})^2 + (\sigma_d^{(i)})^2 - \log\{(\sigma_d^{(i)})^2\} - 1 \right]$$

  - Here, D is the dimension of the latent space (2 in examples above)
  - Note the effect is to regularize the model: shrink all means for different groups towards 0

Intuition about Loss for Variational AutoEncoders

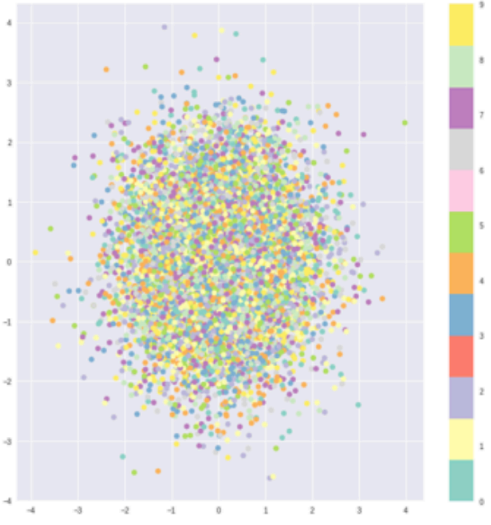
(Decoder)

Only reconstruction loss

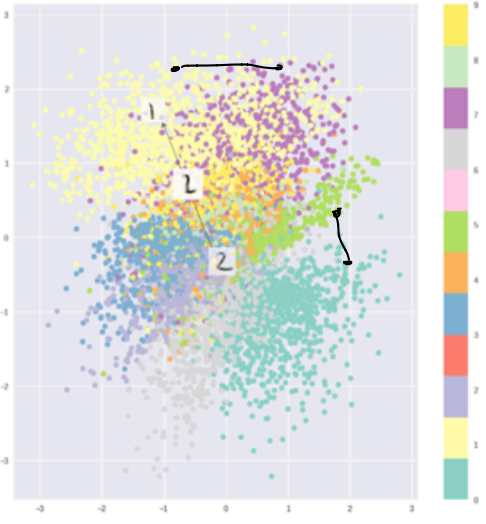


(Encoder)

Only KL divergence



Combination



Figures from https://www.jeremyjordan.me/variational-autoencoders/
and https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf