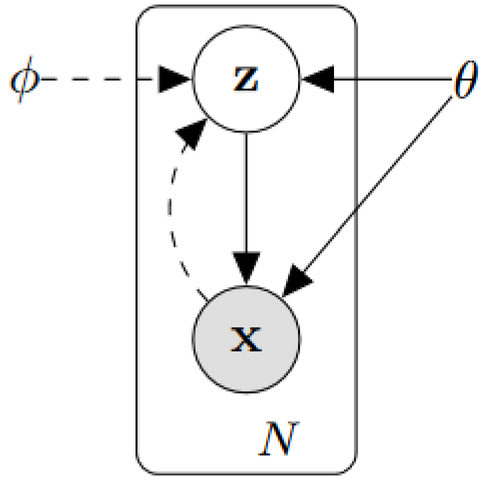
Lab: Beta Variational Autoencoder

Lab Objective:

In this lab, you will need to understand and reproduce the basic Variational Autoencoder (VAE), and then adjust the weight of KL divergence to train a beta-VAE on MNIST dataset to do disentanglement experiments.



Turn in:

Report:

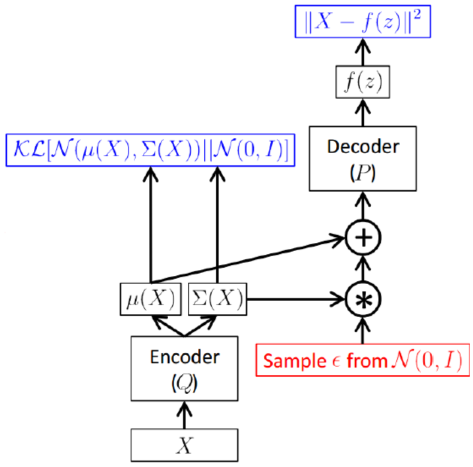
Demo:

Requirements:

1. Implement a basic Variational Autoencoder (VAE)
2. Adjust the weight of KL divergence to implement a beta-VAE
3. Adjust only one dimension of latent codes each time and fix others to show the ability of disentanglement

Implementation Details:

- VAE
    1. Recall the loss function of VAE:
    $$\mathcal{L}(\mathrm{X}, \mathrm{q}, \theta) = E_{Z\sim q(Z|X;\phi)} \log p(X|Z;\theta) - KL(q(Z|X;\phi)||p(Z))$$
    Where $\mathrm{q}(\mathrm{Z}|\mathrm{X};\phi)$ can be considered as encoder, and $\mathrm{p}(\mathrm{X}|\mathrm{Z};\theta)$ can be considered as decoder
    2. Reparametrized trick: train encoder and decoder jointly
    3. Log variance trick (in Pytorch examples): encoder output **log variance** rather than variance directly

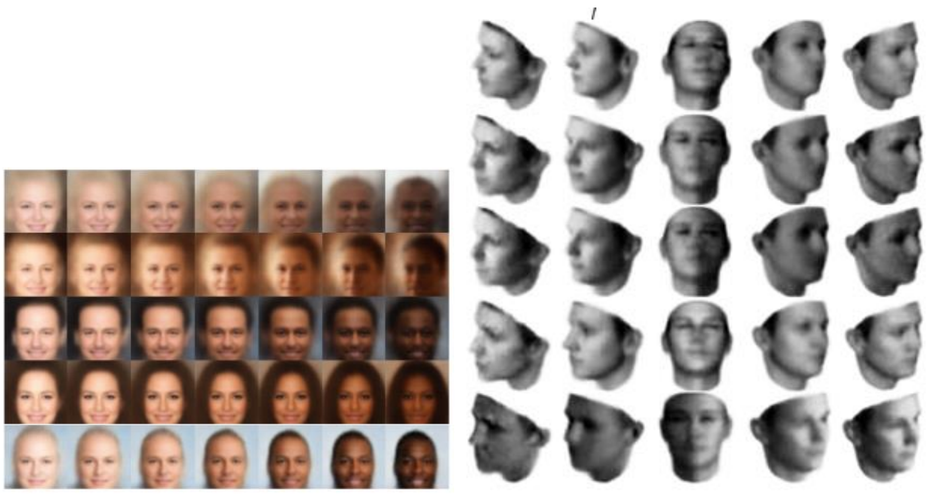


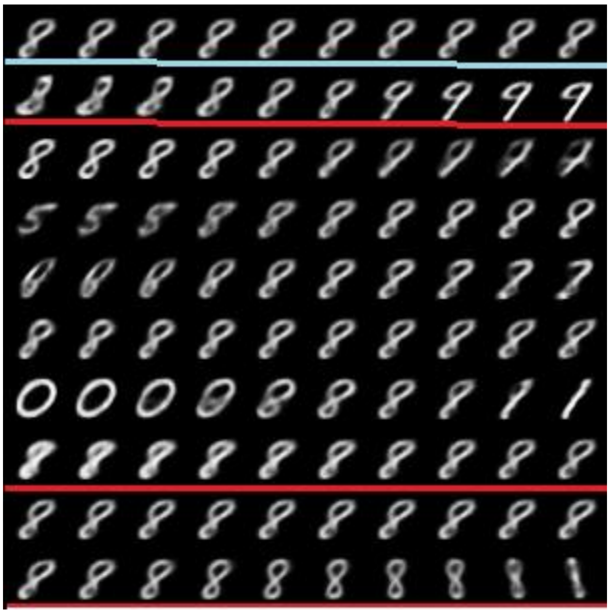
$$\underbrace{E_{\boldsymbol{Z} \sim q(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}')} p(\boldsymbol{X}|\boldsymbol{Z};\boldsymbol{\theta})}_{\text{Re-parameterization for end-to-end training}} \quad - \mathrm{KL}(q(\boldsymbol{Z}|\boldsymbol{X};\boldsymbol{\theta}')||p(\boldsymbol{Z}))$$

- Beta-VAE
  1. Increase the weight of KL divergence to enhance the ability of disentanglement:

$$\mathcal{F}(\theta, \phi, \beta; \mathbf{x}, \mathbf{z}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \, D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

- Disentanglement
  1. A disentangled representation can be defined as one where single latent units are sensitive to changes in single generative factors, while being relatively invariant to changes in other factors
  2. In your lab, after training your model, you should first encode one image to latent codes, and then change latent codes from -2 to 2 uniformly along one axis each time





- Model Architecture
  1. Encoder

Name	Type	Input	Activation	Shape
Fc1	Fully-connected	Input_data	ReLU	(784, 400)
Fc21 (mean)	Fully-connected	Fc1	None	(400, 10)
Fc22 (logvar)	Fully-connected	Fc1	None	(400, 10)

  2. Decoder

Name	Type	Input	Activation	Shape
Fc3	Fully-connected	Gaussian noise	ReLU	(10, 400)
Fc4 (rec)	Fully-connected	Fc3	Sigmoid	(400, 784)

- Hyper-parameters
  1. Batch size: 128
  2. Max epochs: 100
  3. Beta: 6
  4. Learning rate: 1e-3
  5. Latent code size = 10
  6. Optimizer: RMSprop

Reference

1. Auto-Encoding Variational Bayes:
   https://arxiv.org/abs/1312.6114

2. β -VAE: L EARNING B ASIC V ISUAL C ONCEPTS WITH A C ONSTRAINED V ARIATIONAL F RAMEWORK:
   https://openreview.net/pdf?id=Sy2fzU9gl
3. Pytorch VAE example:
   https://github.com/pytorch/examples/blob/master/vae/main.py

Bonus

- Train your model and do disentanglement experiments on different datasets (5%)

Report Spec [black: Demo, Gray: No Demo]

1. Introduction (15%, 15%)
2. Experiment setups:
   A. How you implement VAE (10%, 10%)
   B. How you implement beta-VAE (10%, 10%)
   C. How you do disentanglement experiments (10%, 10%)
3. Results:
   A. Results of your disentanglement experiments (20%, 30%)
4. Discussion (15%, 25%)
   A. Train your model and do disentanglement experiments on different datasets (5%)
5. Demo (20%)