Neural Computation Revision
Part 1

Jinming Duan

1. (Stochastic) Gradient Descent
2. Convolutional Neural Networks (CNNs)
3. Auto-Encoders (AEs)
4. Variational Auto-Encoders (VAEs)
5. Generative Adversarial Networks
6. Recurrent Neural Networks (RNNs)

(Stochastic) Gradient Descent

Gradient descent: optimisation

- Optimisation algorithm
  1. Start with a point $\boldsymbol{w}$ (initial guess)
  2. Find a direction $\boldsymbol{d}$ to move on
  3. Determine how far (η) to move along $\boldsymbol{d}$
  4. Update: $\boldsymbol{w} = \boldsymbol{w} + \eta\boldsymbol{d}$

Minimizing the cost is like finding the lowest point in a hilly landscape

Gradient descent: minimisation

- Gradient descent is one of the simplest, but very general algorithm for minimising an objective function $C(\boldsymbol{w})$ (first proposed by Cauchy in 1847)
- It is an iterative algorithm, starting from $\boldsymbol{w}^{(0)}$ and producing a new $\boldsymbol{w}^{(t+1)}$ at each iteration as:

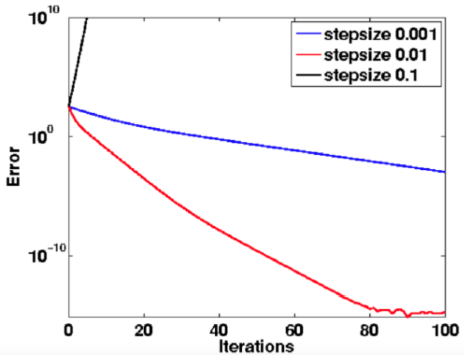
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \nabla C\left(\boldsymbol{w}^{(t)}\right)$$

where $t = 0,1, \dots, T$

- $\eta_t > 0$ is the <u>learning rate</u> or <u>step size</u>

Gradient descent: choosing a step size

- Choosing a good step-size is important
- If step size is too large, algorithm may never converge
- If step size is too small, convergence may be very slow
- May want a time-varying step size



Example Questions

- Know how to compute the gradient of a given function
- Understand the impact of using different values of step size
- Know how to use gradient descent for minimisation
- Least square regression with gradient descent

SGD: introduction

- GD is easy to implement since gradient computation is required
- GD is computationally expensive as it requires to go through all the examples
- Sum structure
  - $C(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} C_i(\boldsymbol{w})$, $C_i(\boldsymbol{w})$ corresponds to the loss for i^{th} example
- At the t^{th} iteration, we randomly choose an index i_t uniformly from $\{1,2,\ldots,n\}$
- We compute a stochastic gradient $\nabla C_i\left(\boldsymbol{w}^{(t)}\right)$
- We update the model as follows

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta_t \nabla C_i\left(\boldsymbol{w}^{(t)}\right)$$

Minibatch SGD: algorithm

- Let $\{\eta_t\}$ be a sequence of step sizes

Algorithm

1. Initialise the weights $\boldsymbol{w}^{(0)}$
2. For $t = 0,1,\ldots,T$
   - Randomly select a batch $B_t \subseteq \{1,2,\ldots,n\}$ of size b
   - Compute stochastic gradient $\nabla C_i\left(\boldsymbol{w}^{(t)}\right)$ with $i \in B_t$ and update

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \frac{\eta_t}{b}\sum_{i\in B_t}\nabla C_i\left(\boldsymbol{w}^{(t)}\right)$$

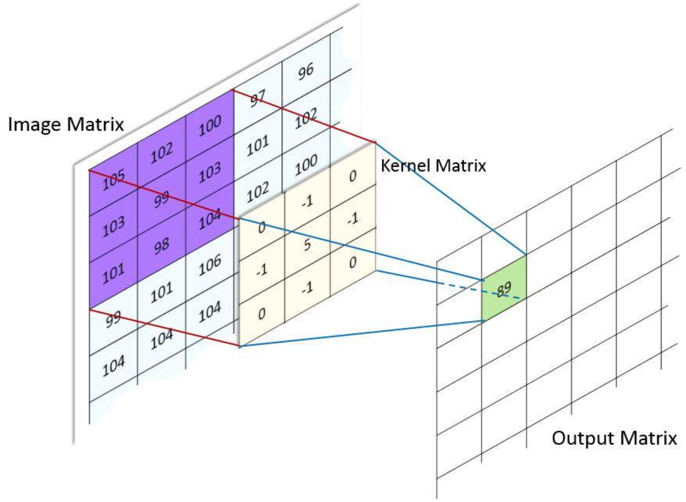
Example Questions

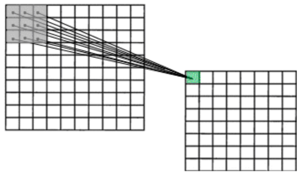
- Stochastic gradient descent implementation details
- Effects of learning rate
- Extension to minibatch SGD (ways to sample minibatch)
- Connections between GD, SGD, minibatch SGD
- How is SGD used for linear classification problems

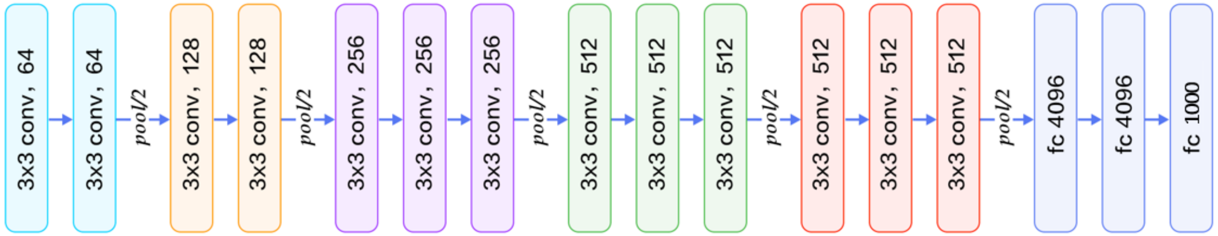
Convolutional Neural Networks

Convolutional Networks

- Convolutional Neural Networks
  - Popular for image recognition and computer vision, etc.
  - How to train a network
  - Convolution operation
    - Convolutional layers
- Pooling stage (e.g. max-pooling, downsampling, upsampling, transpose convolution, etc.)
- Non-linearity (e.g. Relu, leaky Relu, Tanh, etc.)
- Number of parameters and size of feature maps
- Backpropagation (gradients)
- Data processing (min-max, z-score, Tanh, etc)







Example Questions

- Compute the convolution between an input and a kernel
- State some important properties of CNNs relative to fully connected NNs
- Compute number of parameters or size of outputs
- What are typical applications of CNNs?

AutoEncoders

Unsupervised Learning

Available data: $x_1, \ldots, x_N \sim p_{data}(x)$

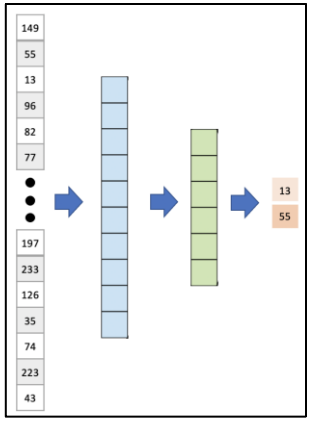
Goal: Learn "**useful features**" of the data / Learn "**the structure**" of the data.

Useful for:

- Dimensionality Reduction (compression)
- Clustering
- Generation / Synthesis
- Learn from loads unlabeled data, when labeled data are limited
- Probability Density Estimation
- …

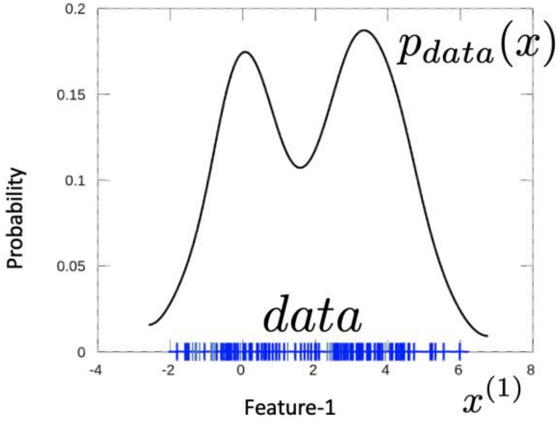
Generation / Synthesis

Dimensionality Reduction



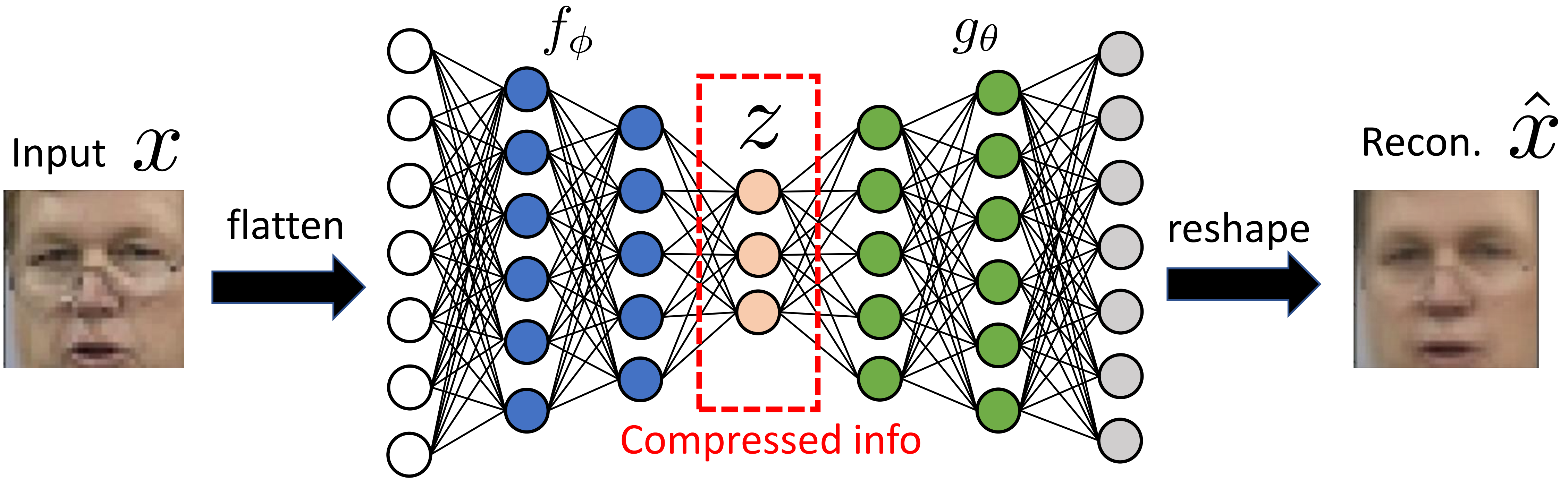
Clustering

Probability Density Estimation



Auto-Encoders with bottleneck

$$f_\phi : \mathcal{X} \in \mathbb{R}^d \rightarrow \mathcal{Z} \in \mathbb{R}^v, \text{ where } v < d$$



What we learned about Auto-Encoders:

Part 1:

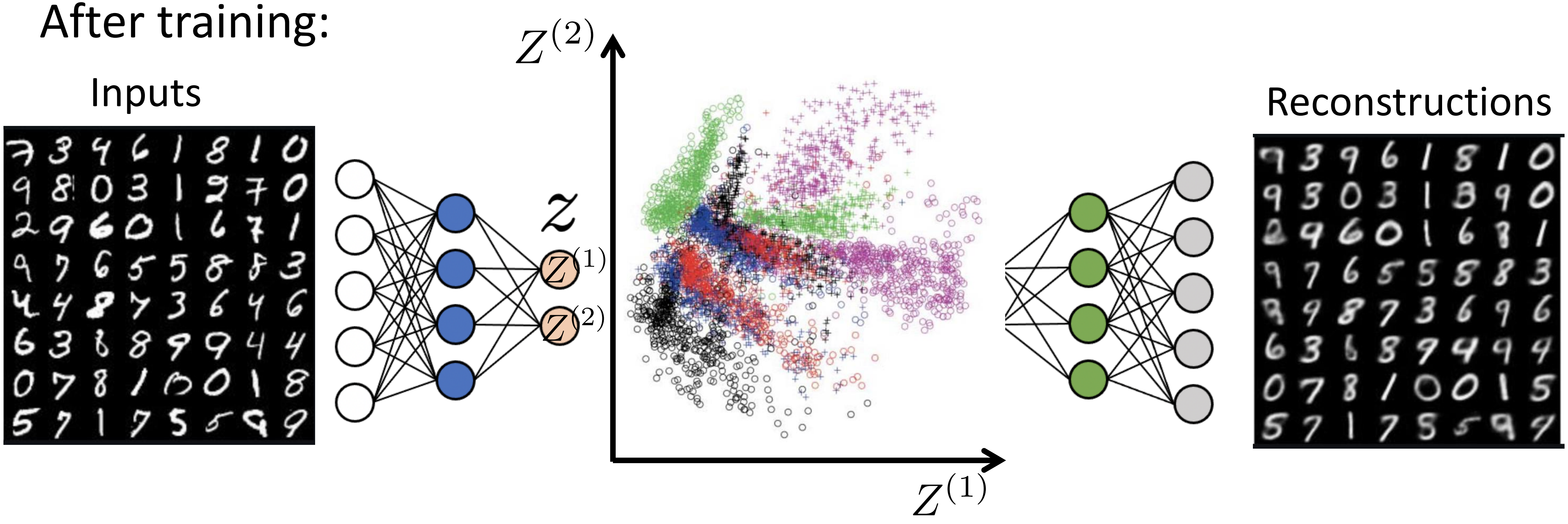
- What is the basic auto-encoder
- How to train auto-encoders
- What is an AE with a bottleneck layer and why to use it
- What feature representations do they learn

Part 2:

- What can they be used for (dimensionality reduction, clustering, pretraining,…)
- What AEs are not good at and why

AEs for learning clusters of data

After training:



From: Hinton and Salakhutdinov, Reducing the Dimensionality of Data with Neural Networks, Science, 2006

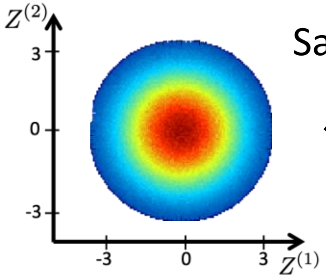
Variational AutoEncoders

What is a Generative Model?

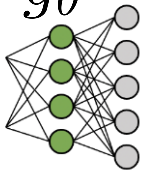
"A generative model describes **how a dataset is generated**, in terms of a **probabilistic model**.
By **<u>sampling</u>** from this model, we are **<u>able to generate new data</u>**."

Generative Deep Learning, by David Foster

Prior distribution of z
$p(z)$



Sampling $\rightsquigarrow$ $\tilde{z}$ g_θ



$\tilde{x}$ Generated new data point

Generative models: $g : \mathcal{Z} \rightarrow X$

We assume: z "causes" x

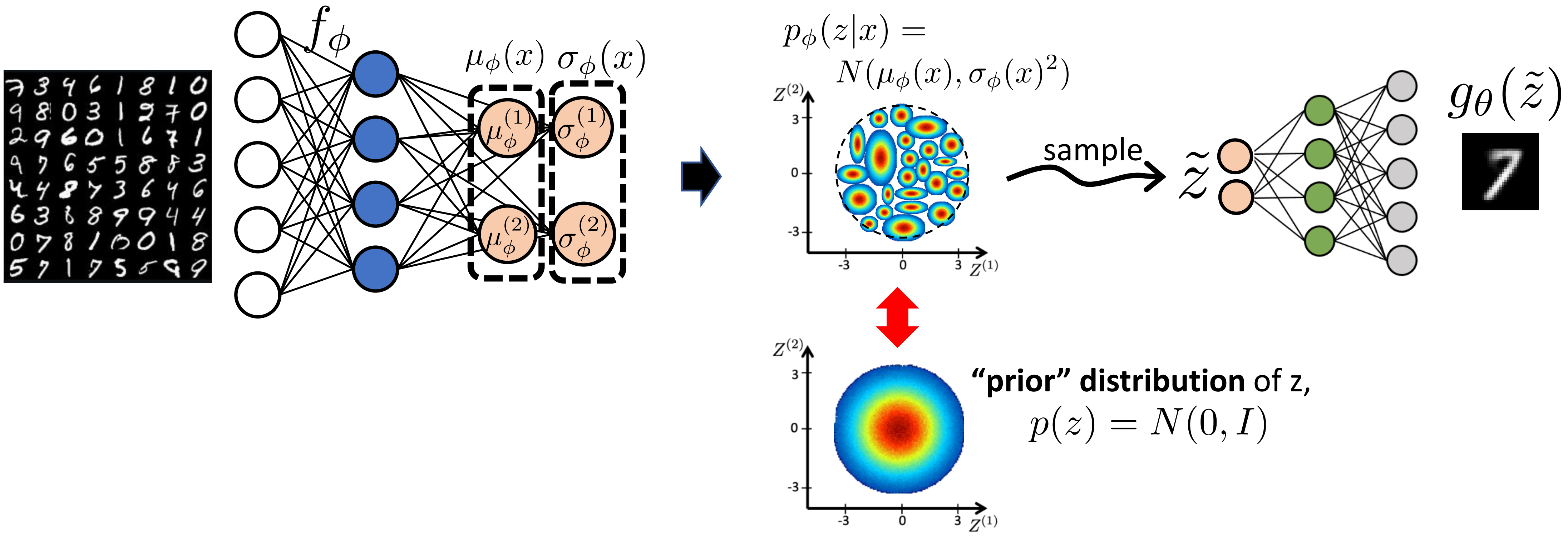
z: content, lighting, zoom …
x: the photo

z "causes" x

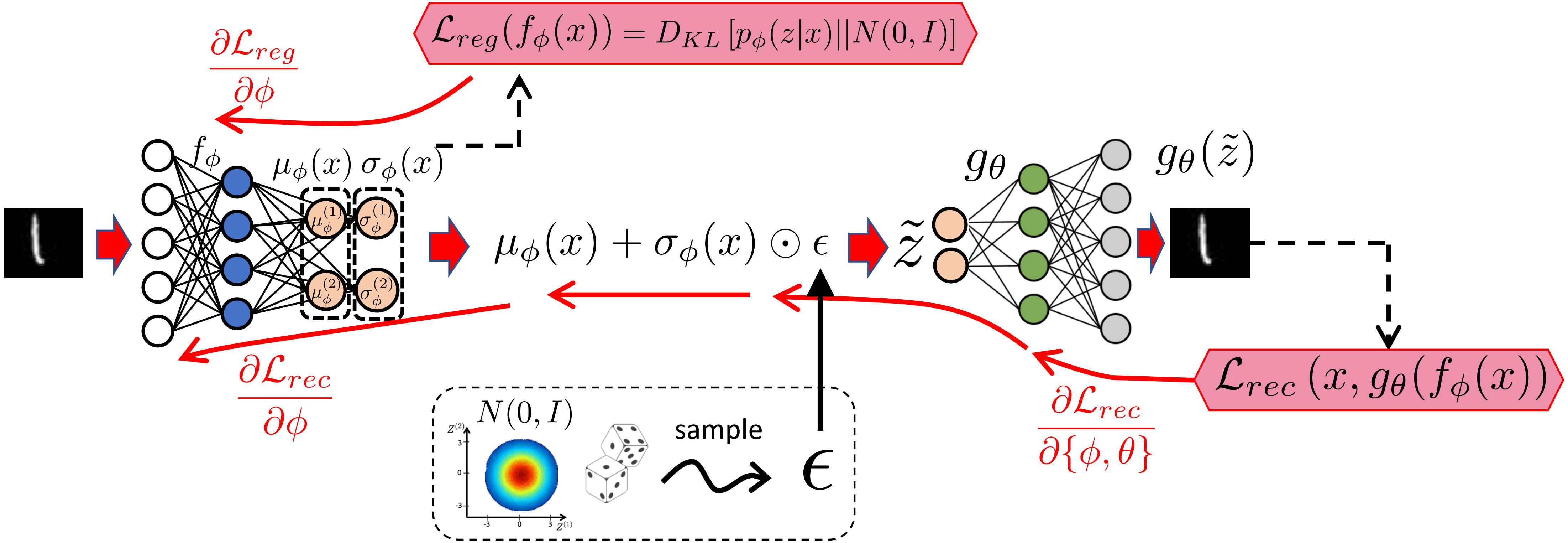
$z \rightarrow x$

Variational Auto-Encoder

Predicted **“conditional”** or
“posterior” of z given x



Variational Auto-Encoders and Re-parameterization trick



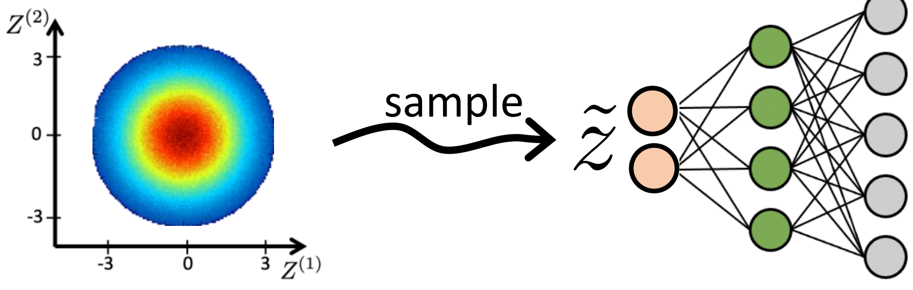
Reparameterization trick

Kingma & Welling, Auto-encoding variational bayes, ICLR 2014
Rezende et al, Stochastic Backpropagation and Approximate Inference in Deep Generative Models, ICML 2014

VAE: Generating new data by sampling from prior

“prior” distribution of z,

$$p(z) = N(0, I)$$

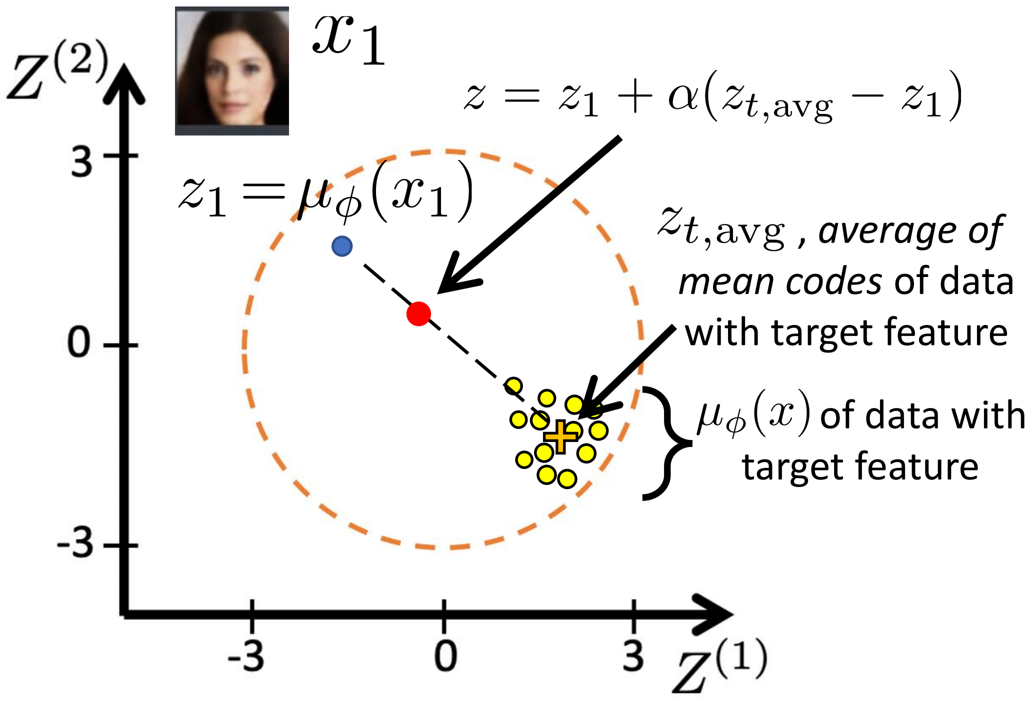


What we learned about VAEs:

Part 1: The “simple” explanation of a VAE, as a regularized AE
- What is a VAE
- How to train VAEs
- How do different terms of training loss influence what VAE learns
- How does a VAE relate to the basic AE

Part 2: Applications of VAE for…
- Generation of new data points
- Modifying data via interpolating between 2 inputs
- Modifying specific feature of an input
- Compression => Reconstruction

Altering specific features of data with VAE



Algorithm:

1. **Encode** original input x and use predicted $\mu_\phi(x)$ as its code:
   E.g. $z_1 = \mu_\phi(x_1)$
2. Identify all training samples that have the desired "target" characteristic. E.g. blondes. Assume these are $x_{t,1}, x_{t,2}, \dots$
3. **Encode** all training samples with the target characteristic.
   Use mean of the Gaussian predicted by encoder as the code.
   E.g. $z_{t,1} = \mu_\phi(x_{t,1}),\ z_{t,2} = \mu_\phi(x_{t,2}), \dots$
4. Compute **average** value of codes of all samples with target characteristic: $z_{t,\text{avg}} = average(z_{t,1}, z_{t,2}, \dots)$
5. Create **new z** code by **interpolation**
   E.g. $z = z_1 + \alpha(z_{t,\text{avg}} - z_1)$
6. **Decode z** with decoder.

Possible with more than 1 target features, similarly to algo on Slide 7.

$g_\theta(\mu_\phi(x_1))$ $\quad g_\theta\left(z_1 + \alpha(z_{t,\text{avg}} - z_1)\right)$ Increasing α

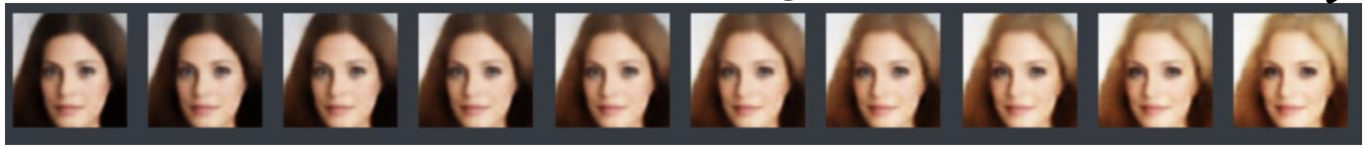
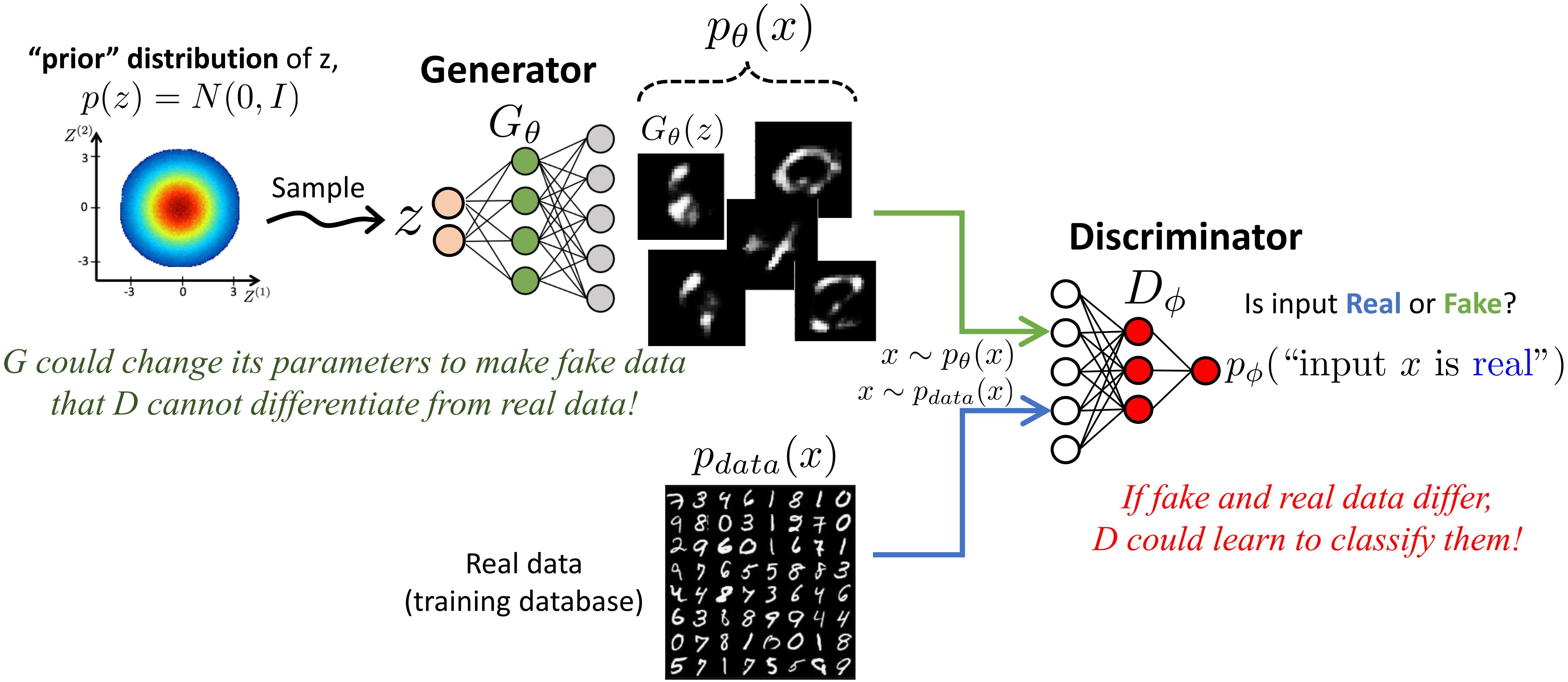


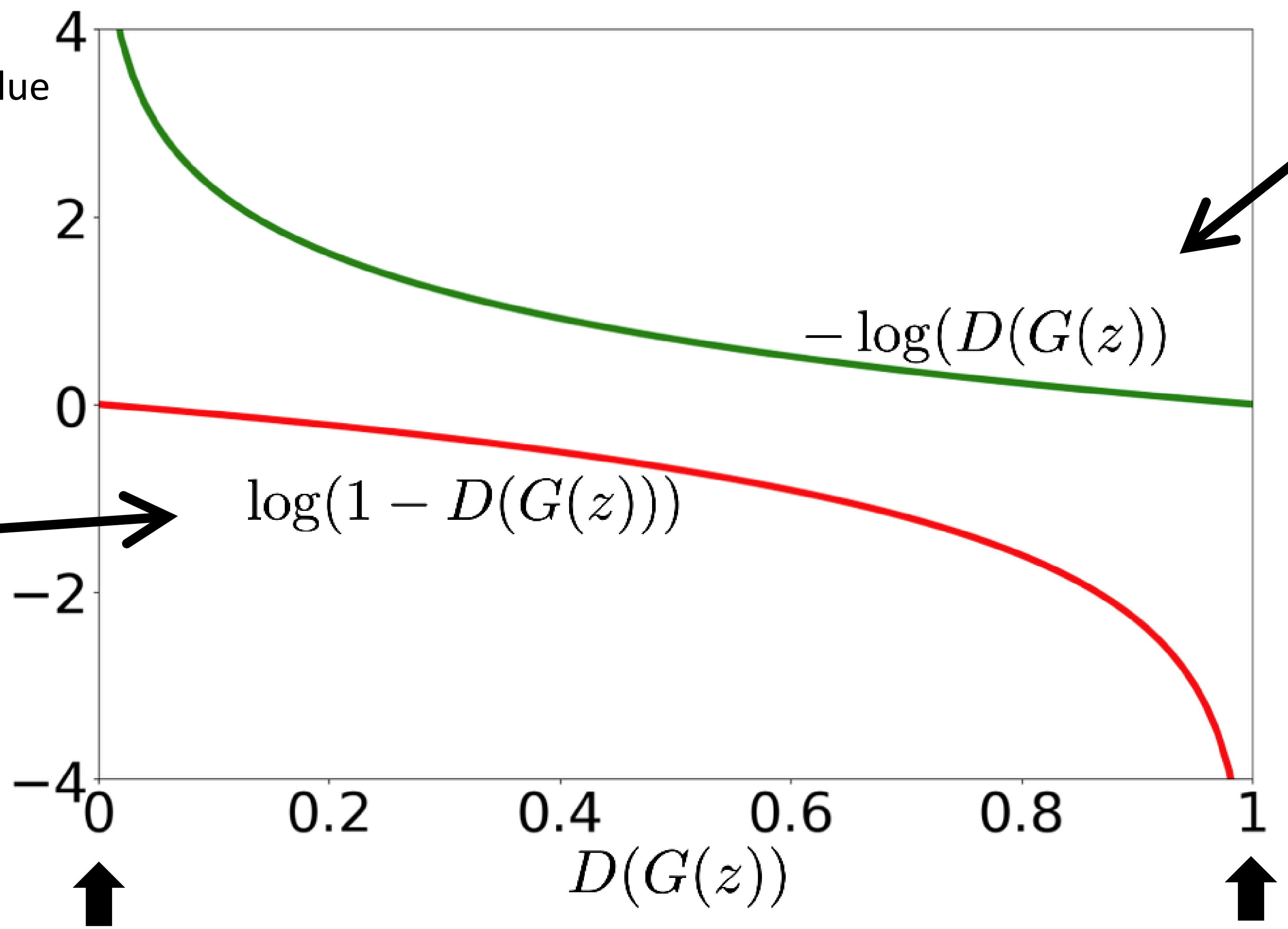
Image from: Steven Flores (link)

Generative Adversarial Networks

Generative Adversarial Networks



Losses for training the Generator G



Loss value

Commonly used in practice

Theoretically motivated loss

D is certain that input G(z) is Fake (correctly)

D is certain that input G(z) is Real (it got tricked by G)

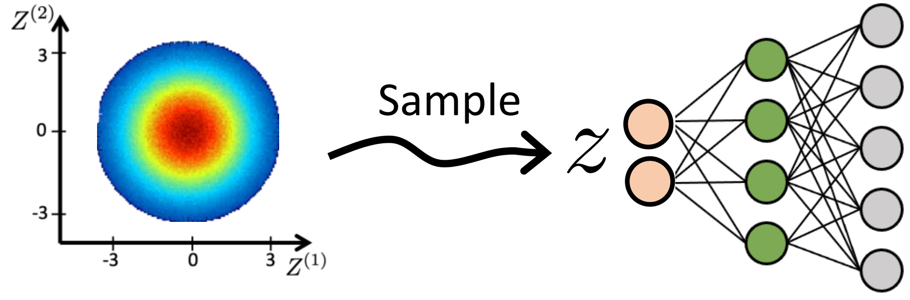
Generating new data points with trained GAN

"prior" distribution of z,

$p(z) = N(0, I)$

Generator

G_θ



To generate new data from a GAN:
Step 1. sample *z* from prior *N(0,I)*
Step 2. "Decode" using *G*, obtaining *G(z)*, the generated sample.

What we learned about GANs:

- What is the GAN (architecture)
- The min-max GAN objective
- How to train a GAN
- Two loss functions for training the Generator (theoretical/practical)
- Use of GANs for Generation of new datapoints

OPTIONAL – NON ASSESSED:

- *Issues of GANs*
- *Extensions / more advanced models*

Assessed material:

Anything written or discussed in:

- Slide decks
- Pre-recorded videos
- Lectures
- Tutorials (not code / implementation, but understanding of models & behaviour)

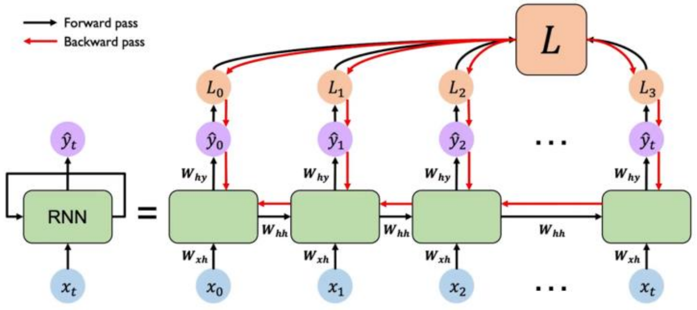
NOT assessed:

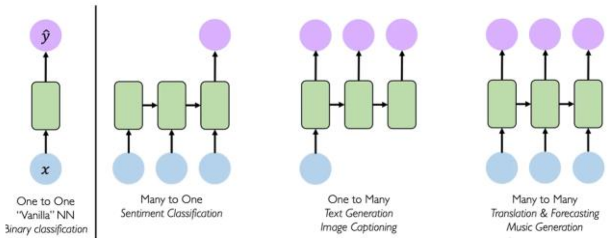
- Material in “further reading” that is not already part of the above assessed material.
- “Optional” material on VAEs (probabilistic derivation) and GANs (issues and advanced models).

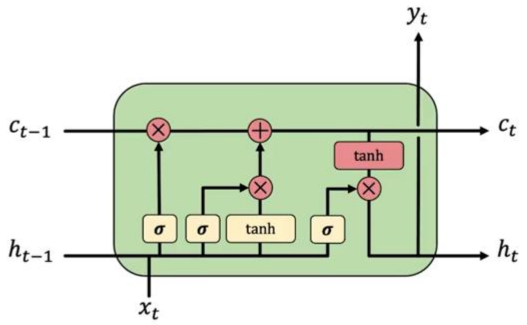
Recurrent Neural Networks

Recurrent Networks

- Recurrent Neural Networks
  - Popular for modeling sequence data, etc.
    - Vanilla RNN
    - Sequence modeling applications (one to one, many to one, etc)
- Neurons with recurrence (unfold RNNs, shared weights, computational graph)
- Design criteria (variable length inputs, long term dependenices, maintain information about order, shared parameters across the sequence)
- BPTT (exploding/vanishing gradients, solutions)
- LSTMs (forget, store, update, output)







Example Questions

- How is an RNN handling variable length inputs
- How is hidden state updated at each time step
- Whats an RNN’s application in terms of inputs and outputs
- Methods to tackle long term dependencies issues
- Compute the outputs of an RNN, given some inputs
- How do LSTMs work? Why is it better?

Compulsory Material

- All material not marked optional on the “Modules” page on Canvas is compulsory reading, e.g.,
  - Lecture notes
  - Reading lists (only contents covered in lectures)
  - Videos
  - Exercises, lab sheets
- Exam can cover anything covered by compulsory material

Thank you!