[AI Theory&App] 04 Auto-encoders and Generative Adversarial Networks

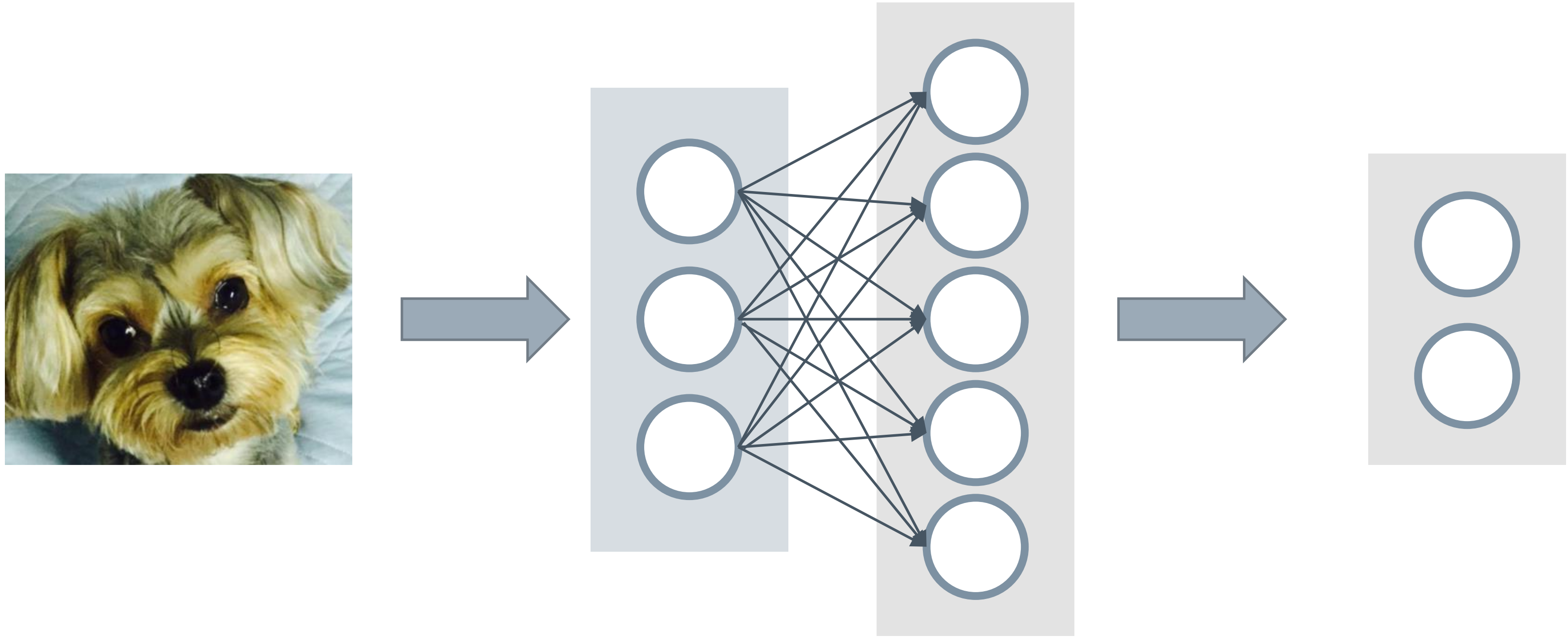
Instructor: Hao-Shang Ma

Department Of Computer Science And Information Engineering, NTUST

Auto-encoder

What are Auto-encoders

› Learn the latent representation for the input

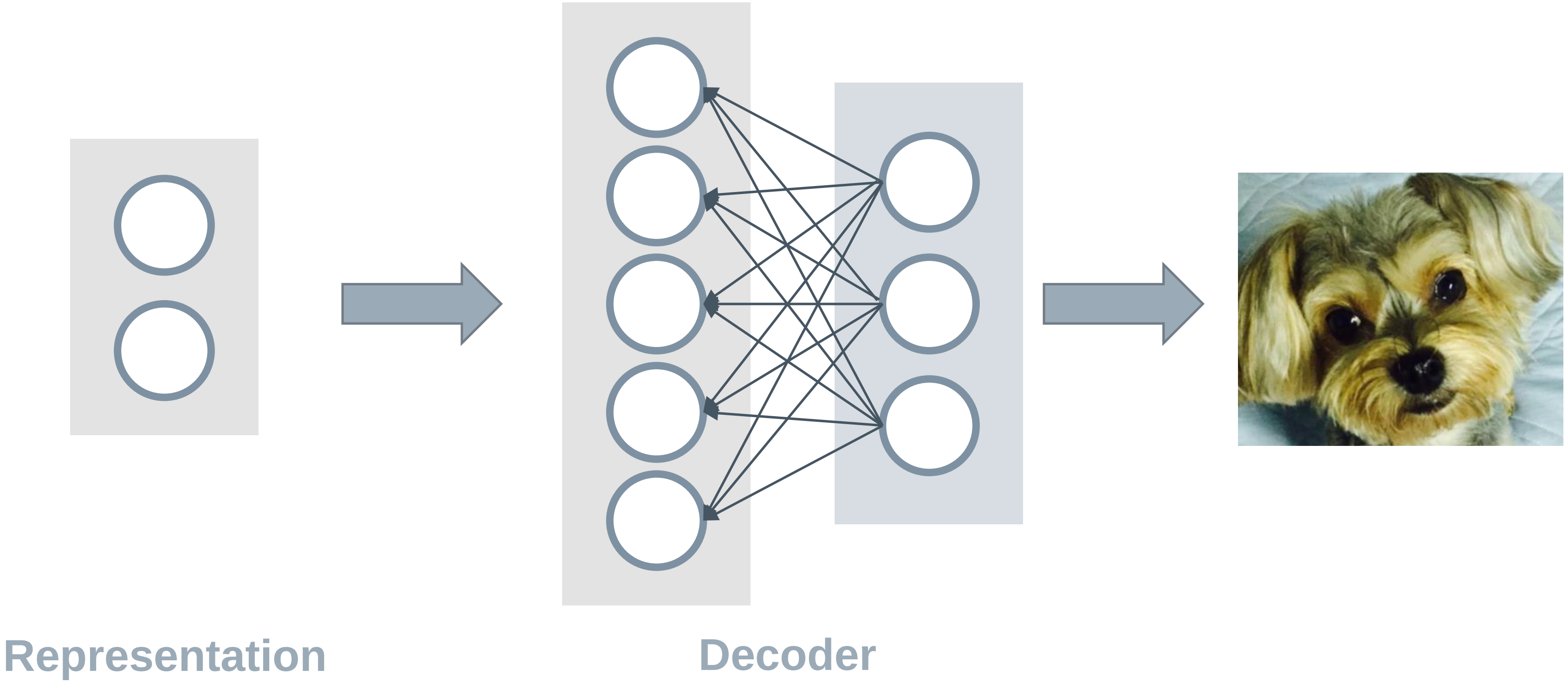


Encoder

Representation

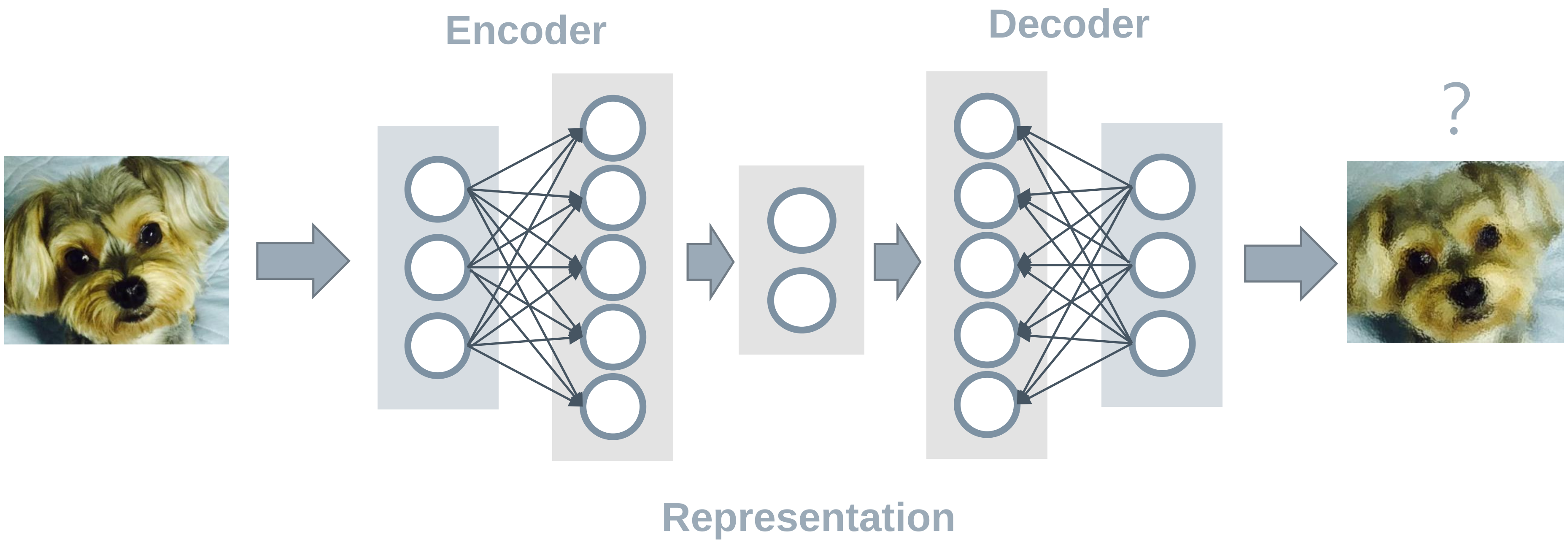
How to Check the Code

- Need a decoder to reconstruct the object by the latent representations



The Goal of Auto-encoders

- Check if the reconstructed object is similar to the original one



The Goals of Encoders and Decoders

- Encoder
  - Make the latent representation approximately represent the object
  - Build the representation for data
- Decoder
  - Make the latent representation approximately reconstruct the object
  - Generate data from the representation

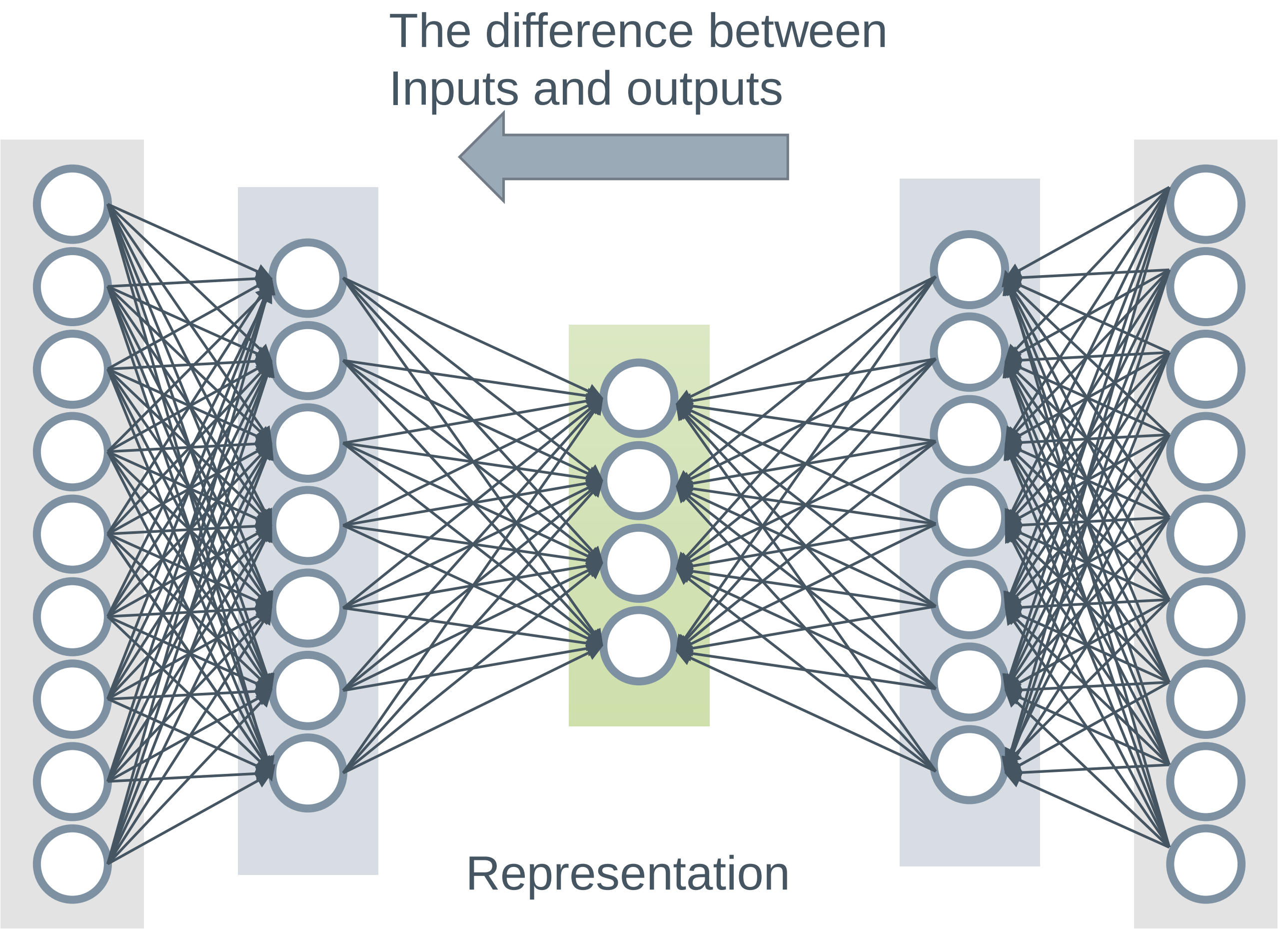
How to Train Auto-encoders

- The auto-encoders can be treated as a DNN
  - Input is a vector
  - Output is a vector with the same dimension as the input
  - Errors are the differences between inputs and outputs

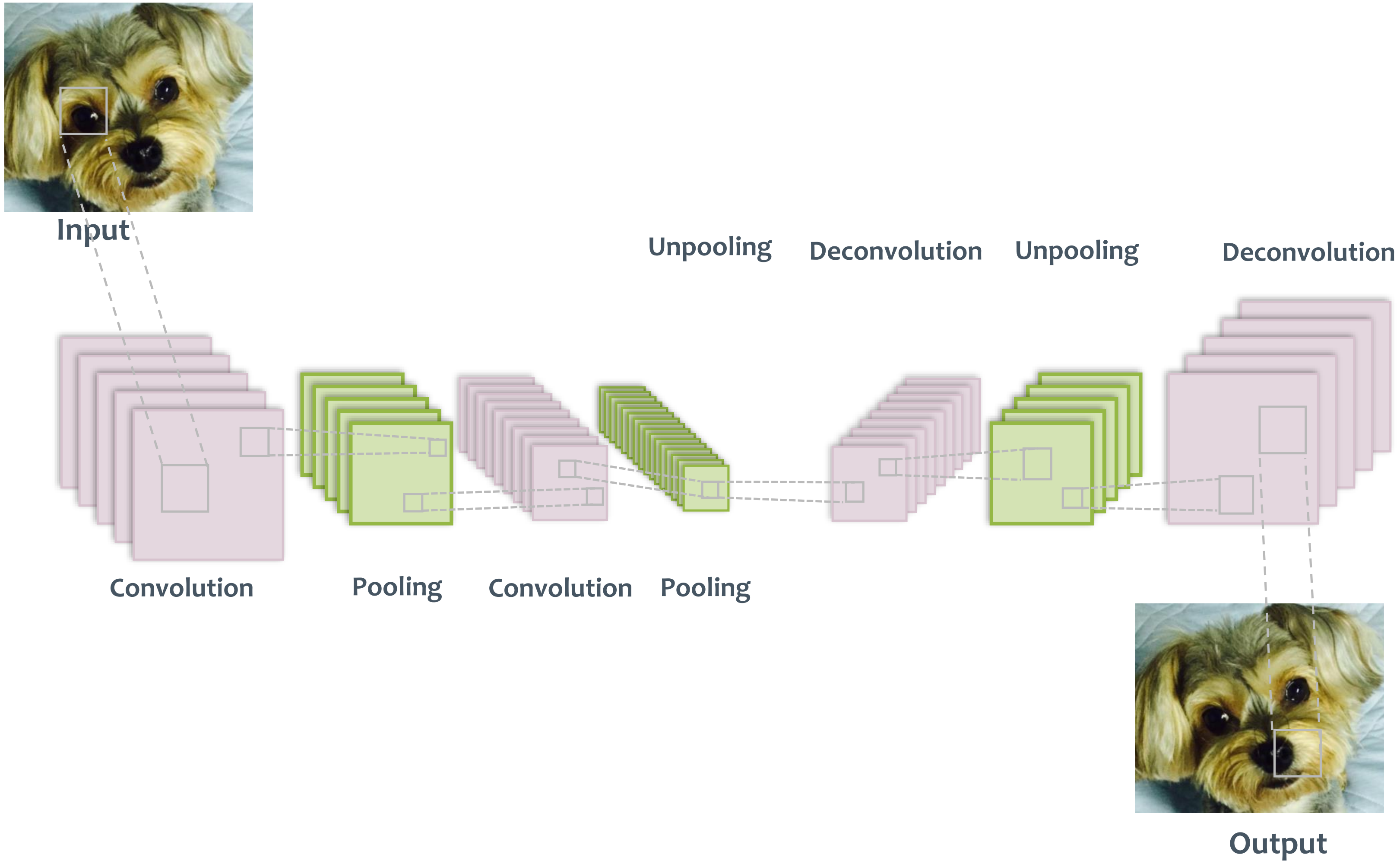
MSE $$E = \sum_{i=1}^{n} \left\| y_i - \hat{y}_i \right\|^2$$

Cross Entropy $$E = -\sum_{i=1}^{n} \left(y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

How to Train Auto-encoders



Auto-encoder by CNN



Unpooling

pooling

0.1	0.7	1.5	0.1
-0.3	0.5	0.2	-0.7
0.4	0.9	0.3	0.6
0.8	-0.1	2.1	0.8

→

0.7	1.5
0.9	2.1

unpooling

0.7	1.5
0.9	2.1

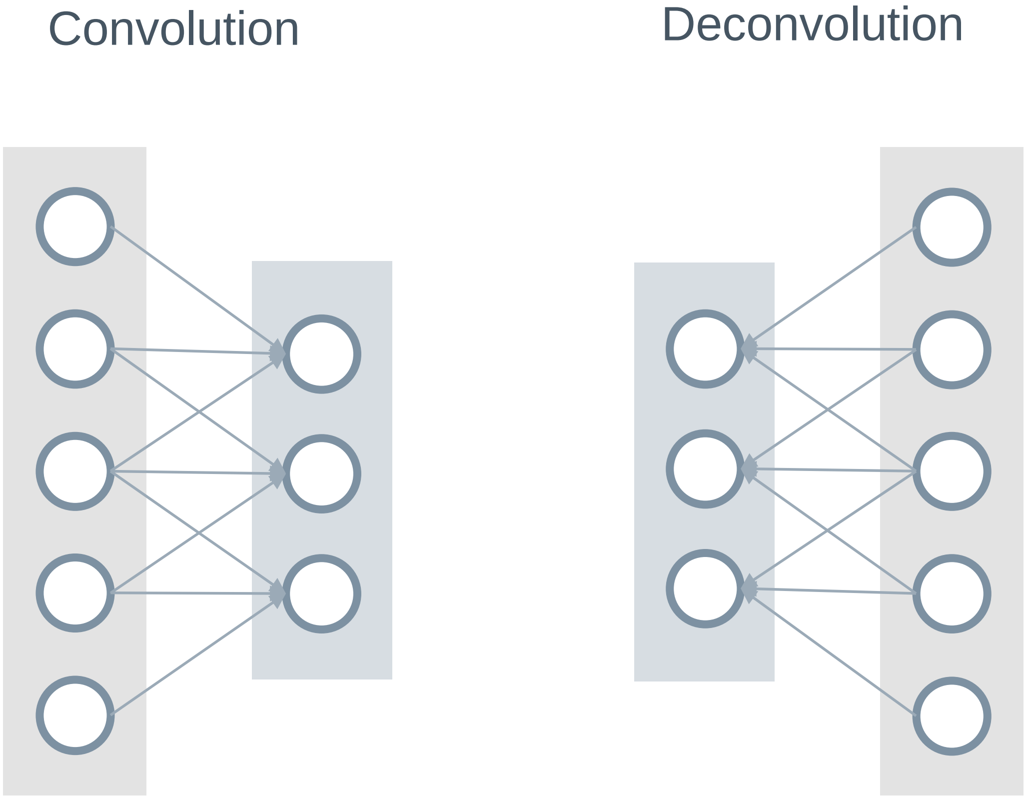
→

0	0.7	1.5	0
0	0	0	0
0	0.9	0	0
0	0	2.1	0

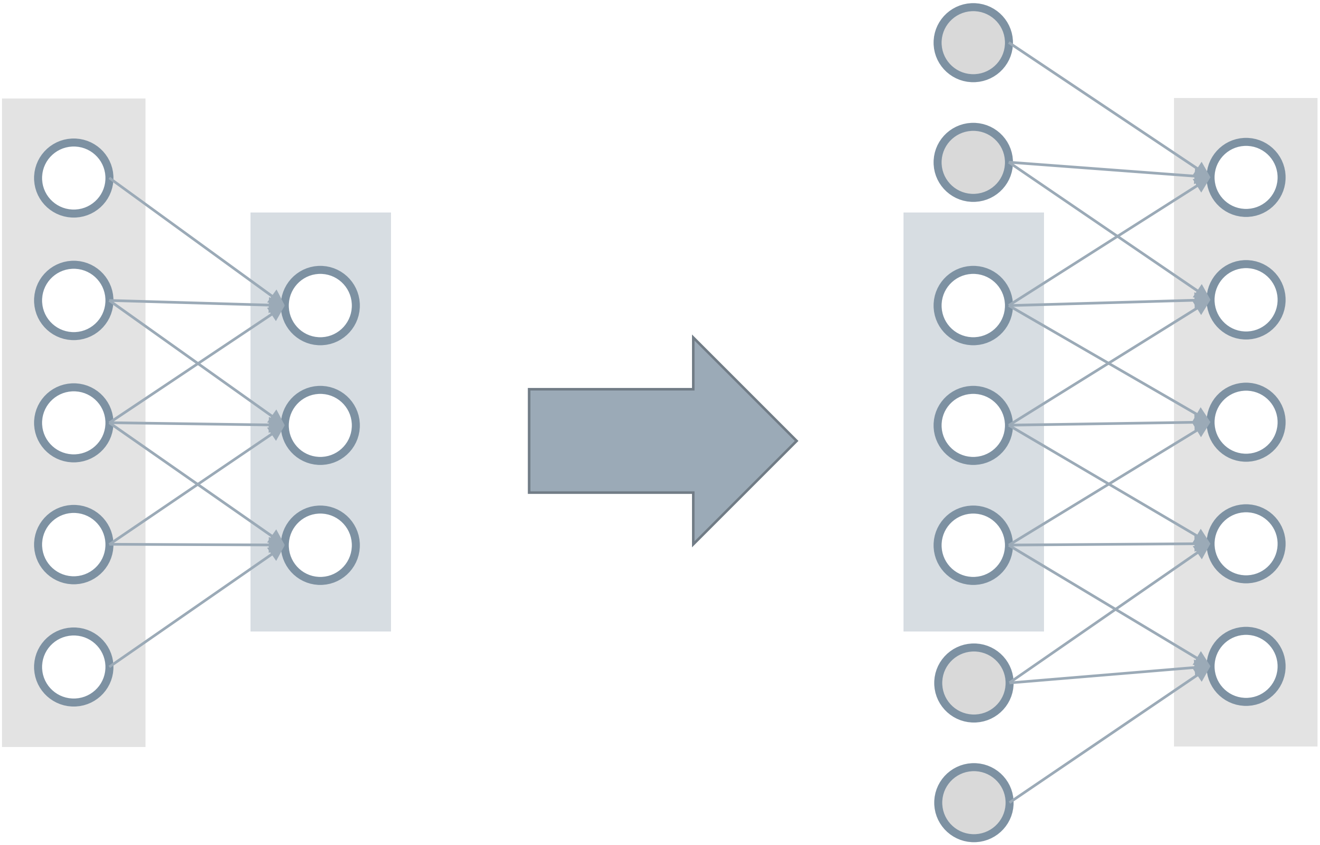
0	1	1	0
0	0	0	0
0	1	0	0
0	0	1	0

The pooling position

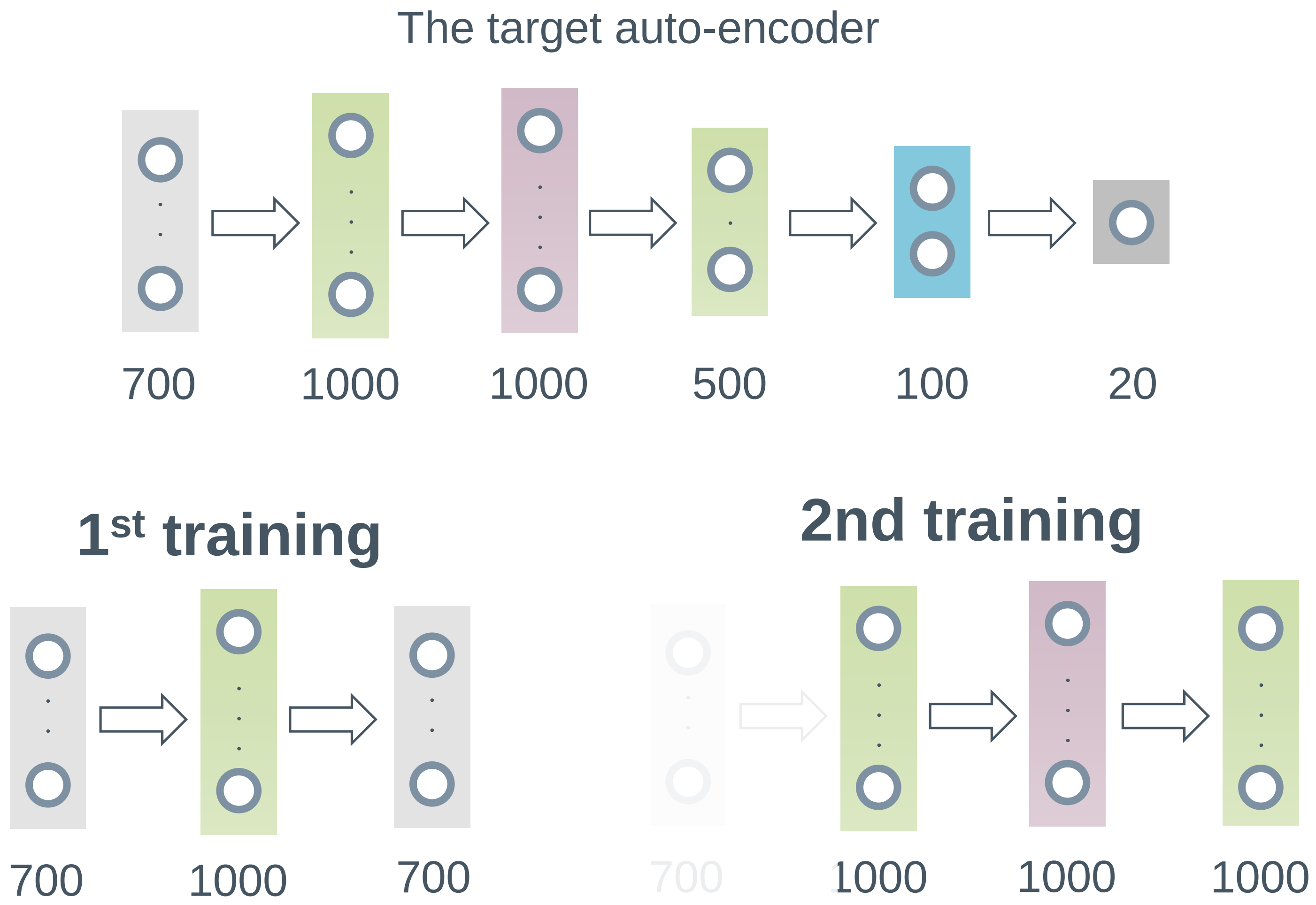
Deconvolution



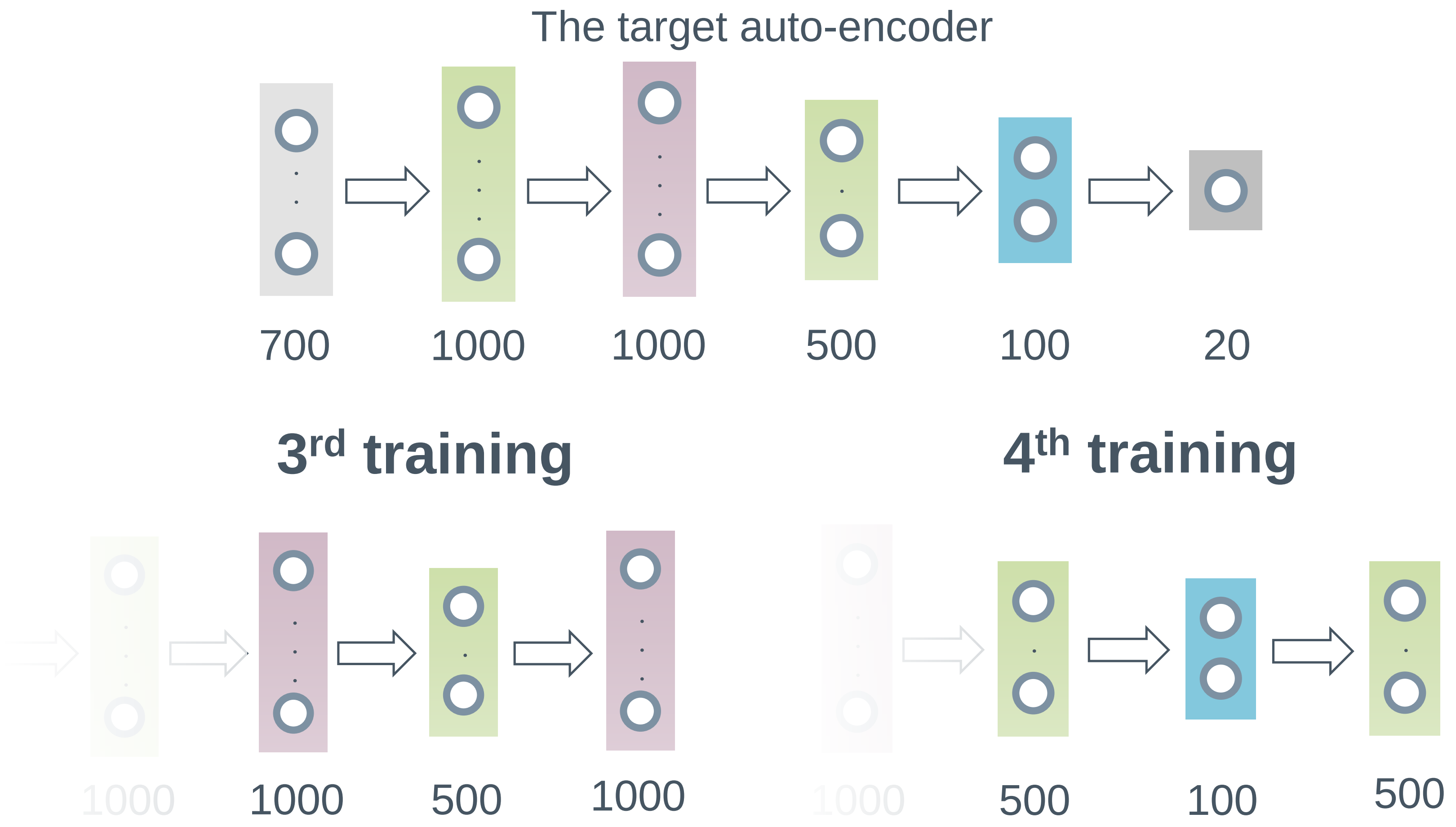
Using Convolution to Do Deconvolution



Pre-trained DNN as Auto-encoders

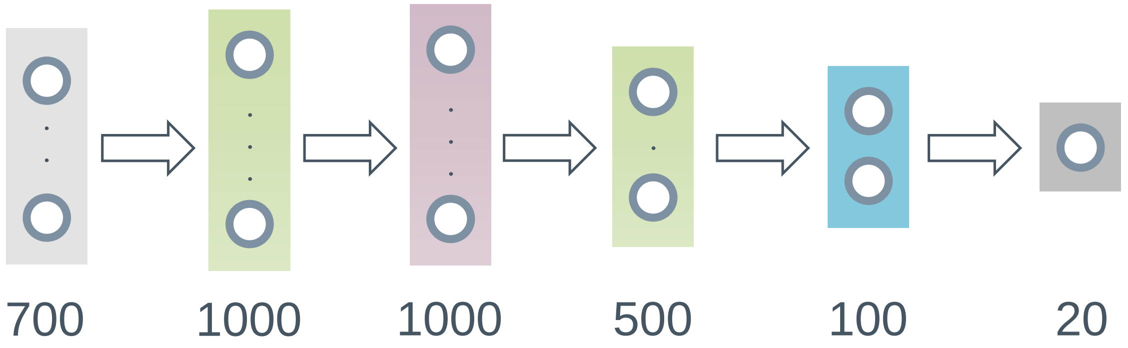


Pre-trained DNN as Auto-encoder

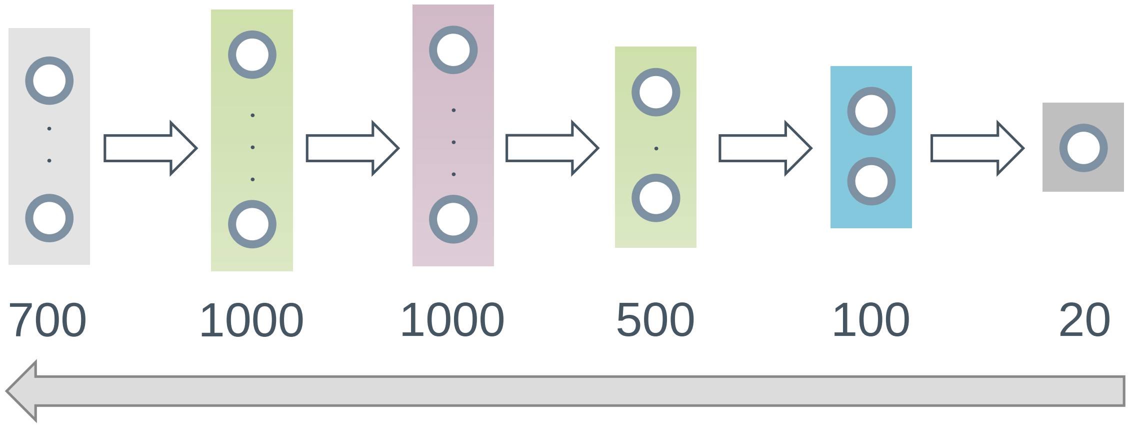


Pre-trained DNN as Auto-encoders

The target auto-encoder



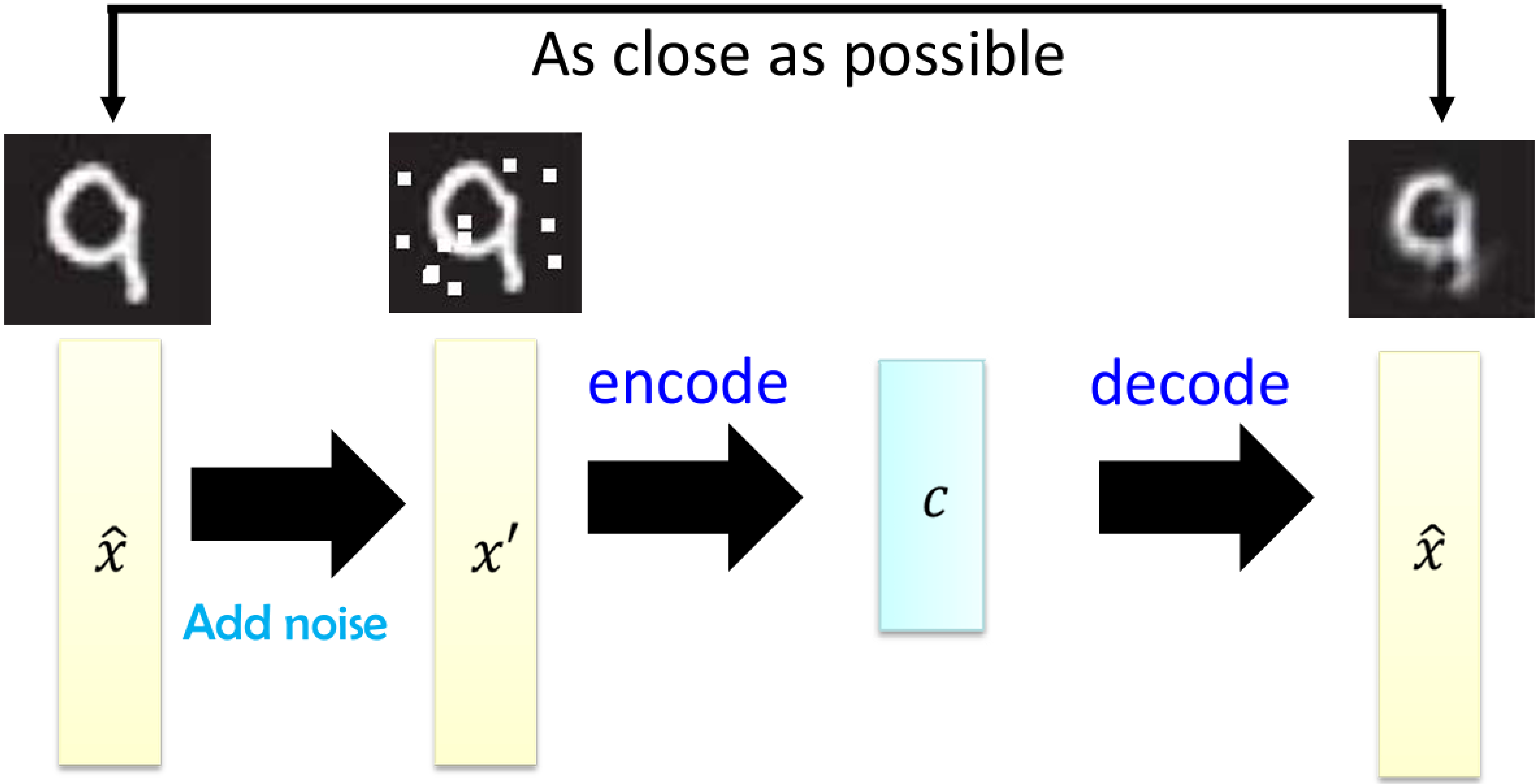
5th training



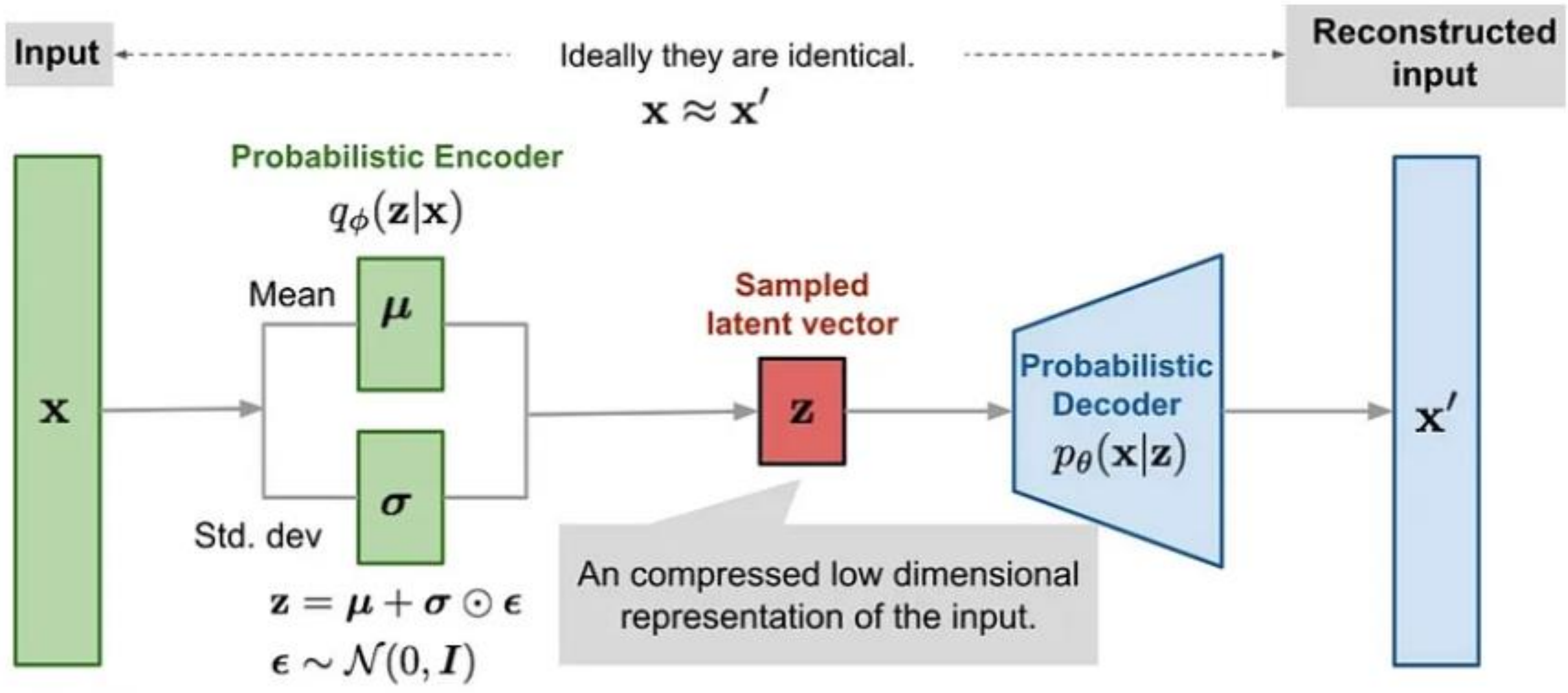
Fine tune the weights

Different Auto-encoder

De-noising auto-encoder

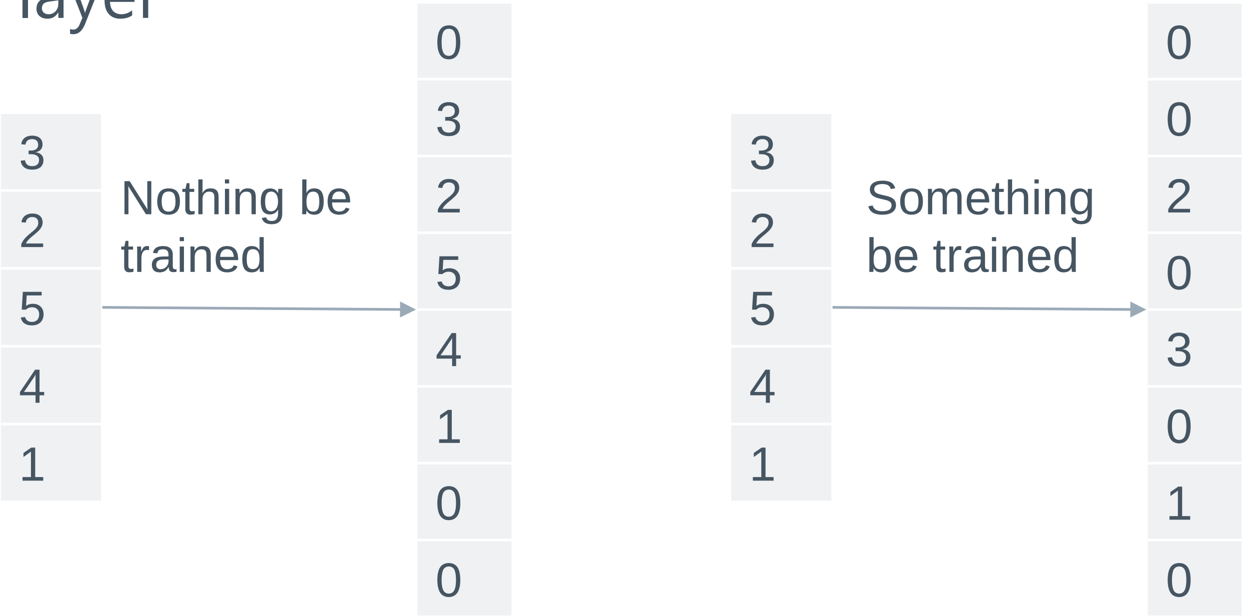


Variational Auto-Encoder



Sparse Auto-Encoder

- If the dimension of hidden layers is higher than the input layer
  - Need some regularizations to make sure the sparsity of the hidden layer



How to Design a Good Auto-encoder

The Hyper Parameters

- The layers of encoder and decoder
- The dimensions of each layer
  - The dimensions of the code should be lower than the input
- Pre-train or not

Weight Sharing

- Regularization
  - Weight sharing acts as a form of regularization.
  - By constraining the encoder and decoder to use the same weights
  - reduces the model's capacity, preventing overfitting
- Faster Training
  - Weight sharing can make training faster and more stable because the encoder and decoder effectively cooperate during training
  - The model may converge more quickly with shared weights.

Main Applications

- Data Compression
  - Autoencoders can be used to compress data, such as images, text, or audio, into a lower-dimensional representation.
- Image Denoising
  - Autoencoders can be used to remove noise from images.
- Dimensionality Reduction
  - Autoencoders are employed to reduce the dimensionality of data while preserving its essential features
  - This is useful in data visualization, feature selection, and simplifying machine learning models

Main Applications

› Generation
  – Variational Autoencoders (VAEs), a type of auto-encoder, are used to generate new data samples.

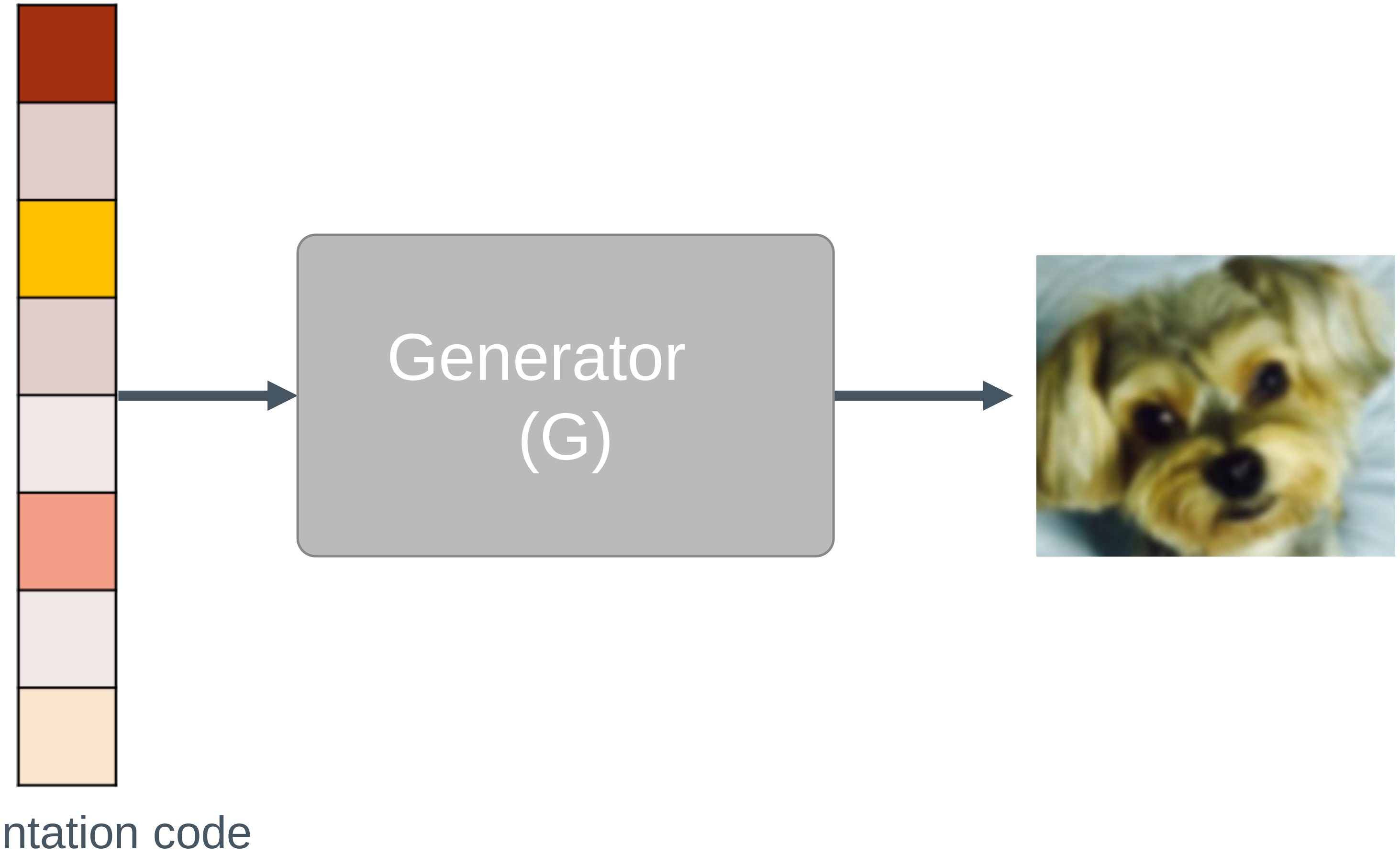
› Text Summarization
  – Autoencoders can be used for text summarization tasks.
  – By encoding the input text into a lower-dimensional representation and then decoding it, they can generate concise summaries of documents.

Generative Adversarial Network

The Creation of NNs

- Traditional NNs is a supervised learning model
  - Learn knowledge from labeled data
  - The knowledge is based on the label
  - Cannot have some creativity
- How about creating data by machine?
  - Auto-encoder
    - The decoder in auto-encoder

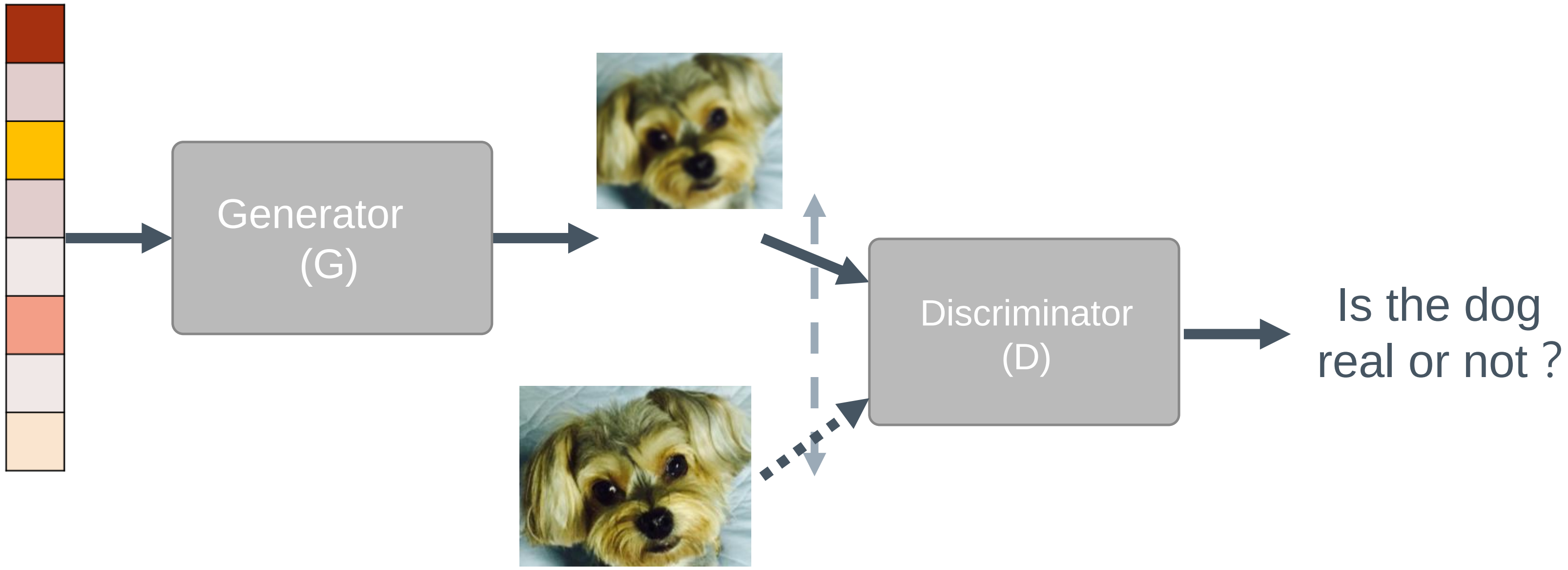
The Generator in GANs



Is Generated Data Real Enough?

- Discriminator
  - Used to distinguish the true input object from the generated fake one
- The goal of G is to confuse the discriminator

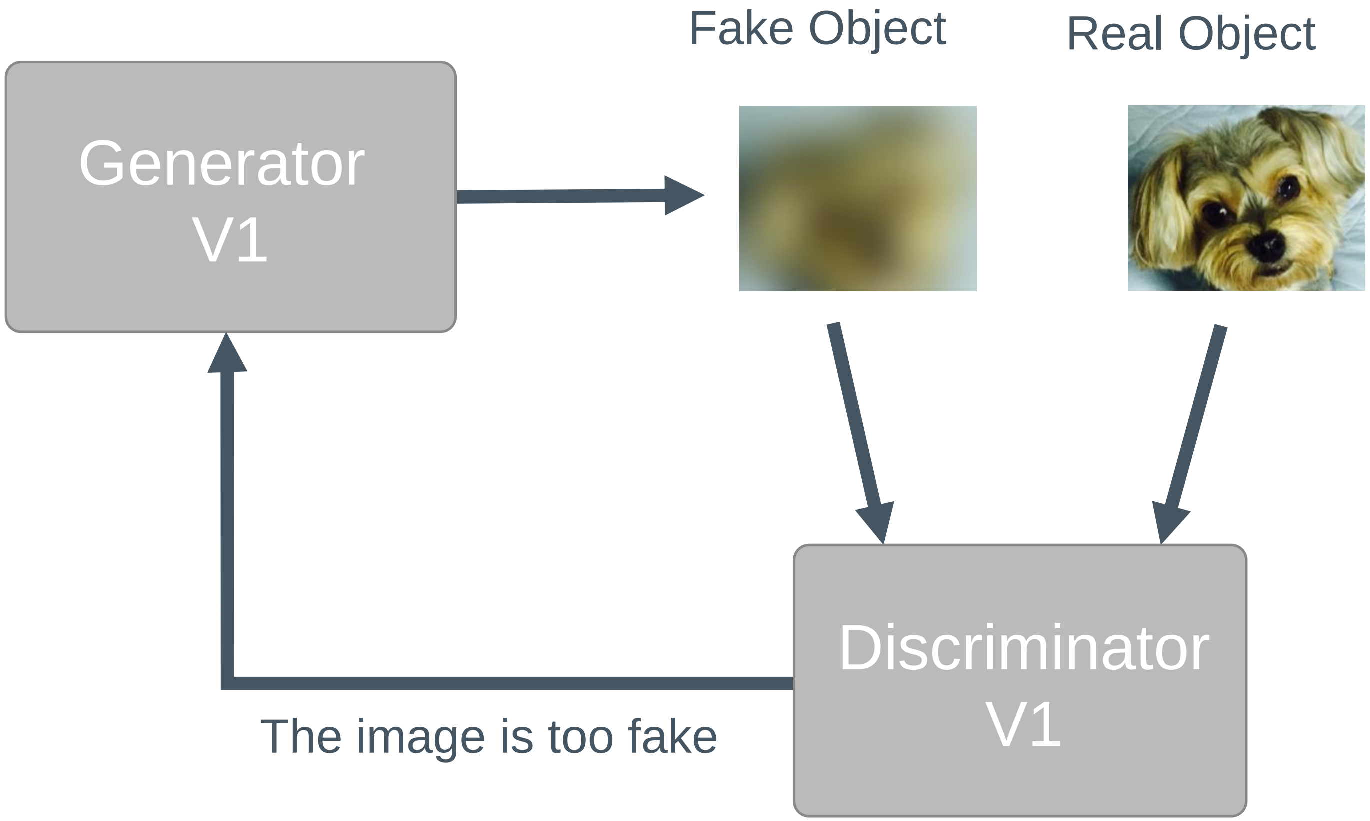
A Brief Overview of GANs



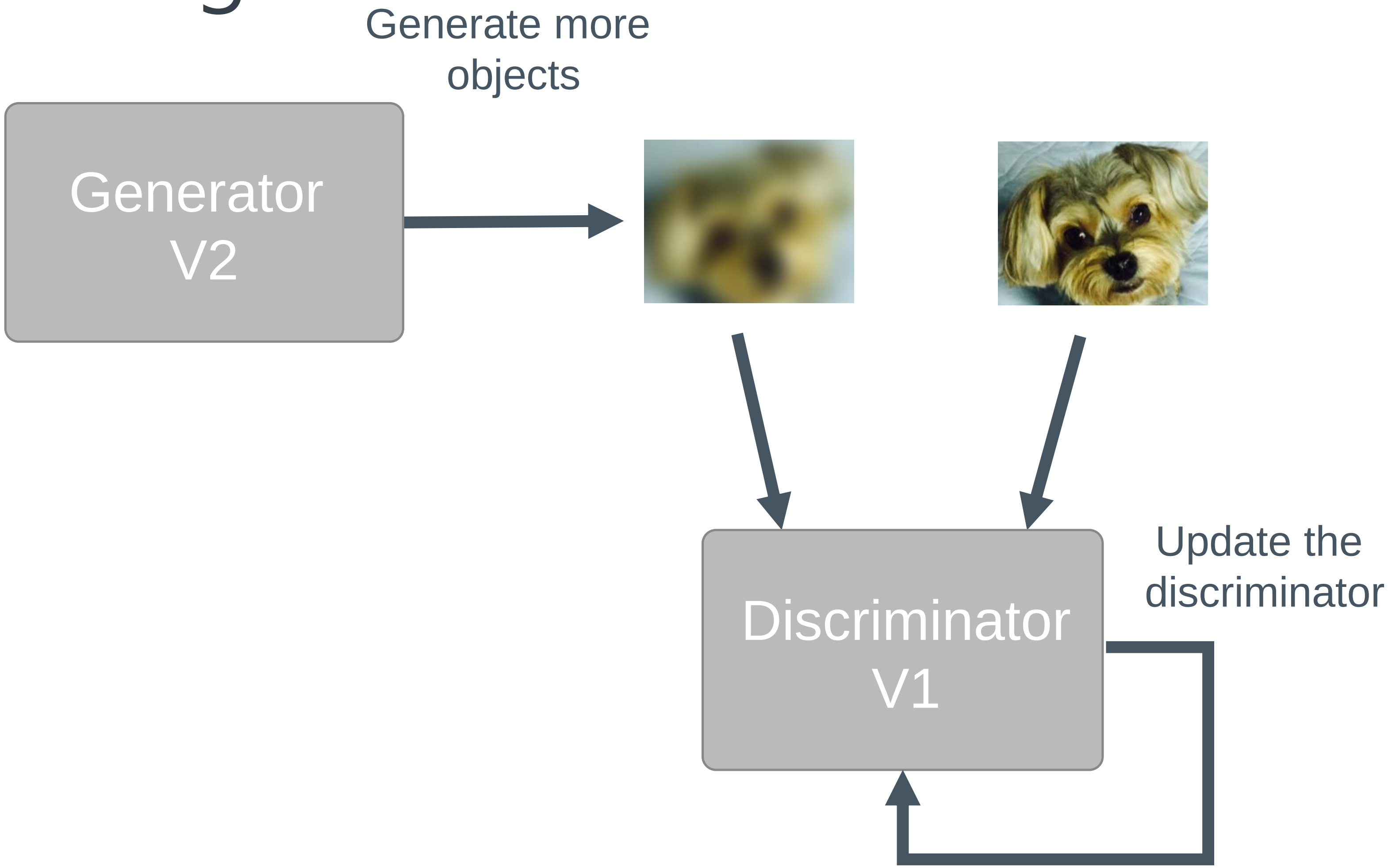
The Training Objective

- Generator
  - Generate the object very close to the real object
- Discriminator
  - Correctly distinguish the true or fake objects

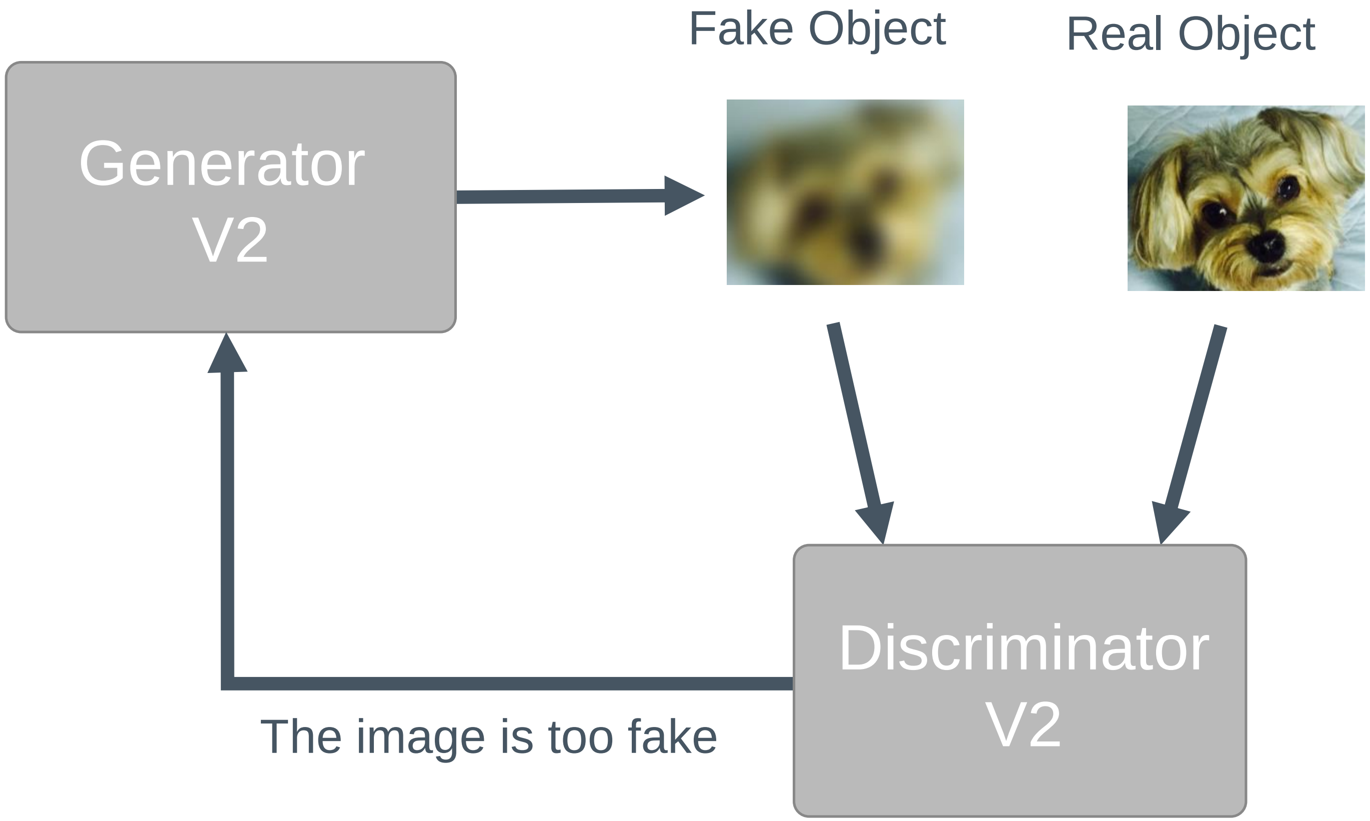
The Training Process of GANs



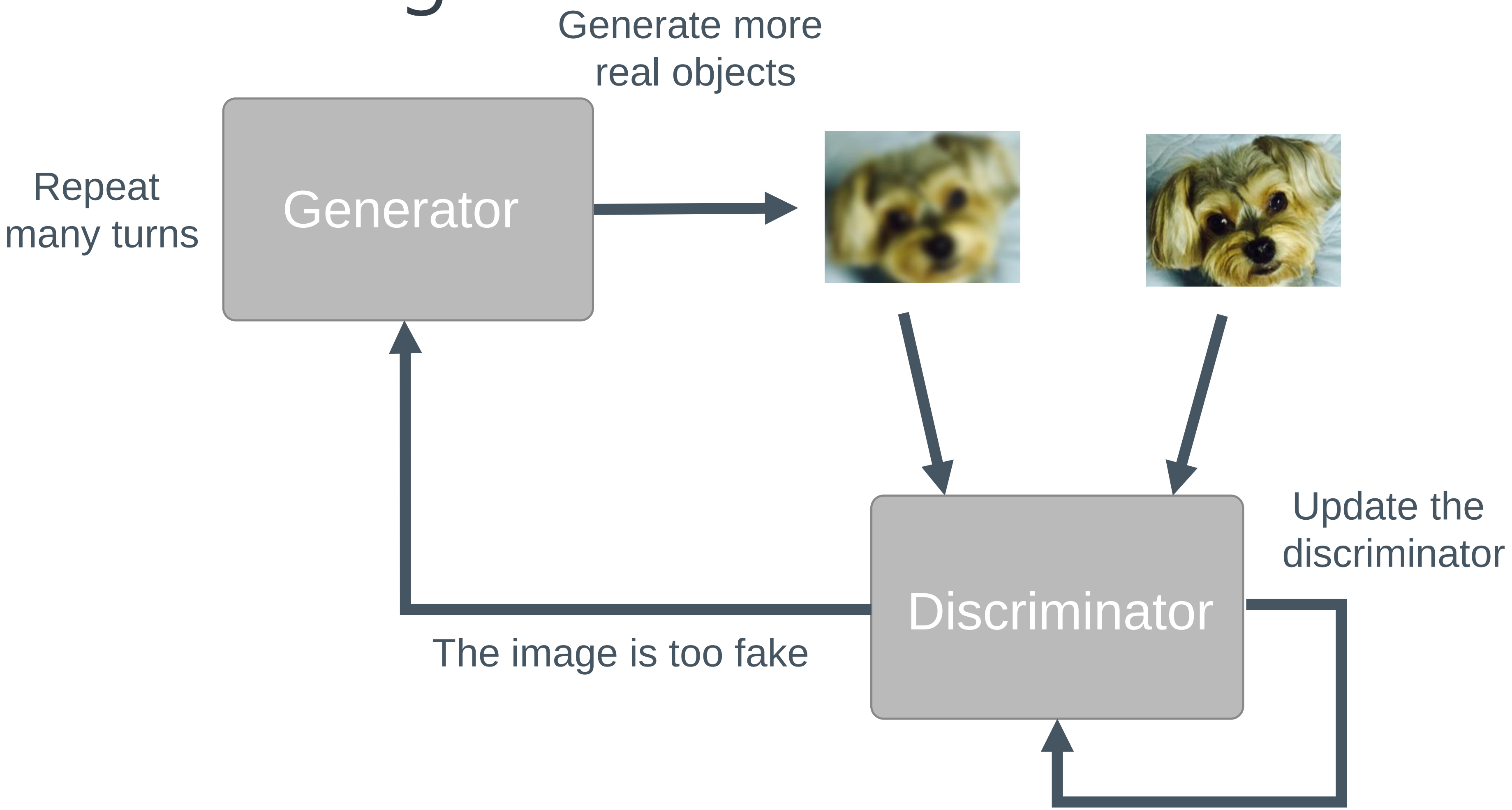
The Training Process of GANs



The Training Process of GANs



The Training Process of GANs



How to Train GANs

- For every iteration
  - We train the discriminator first
    - By a given generator
    - Do many rounds to create a powerful discriminator
  - Then we train the generator
    - By a given discriminator
    - Only one iteration to prevent overfitting

Training On Discriminators

- The discriminator is a binary classifier
  - Classify if the input is real or not
- Use cross-entropy as the error of the discriminator
  - Back-propagation to train discriminator
  - Lower cross-entropy is better

Training On Generators

- The goal of generator is to generate the data close to the real data
  - Let discriminator unable to classify the real objects well

- Using the cross-entropy of the discriminator as the error
  - Back-propagation to train the generators
  - Higher cross-entropy is better

In Practice

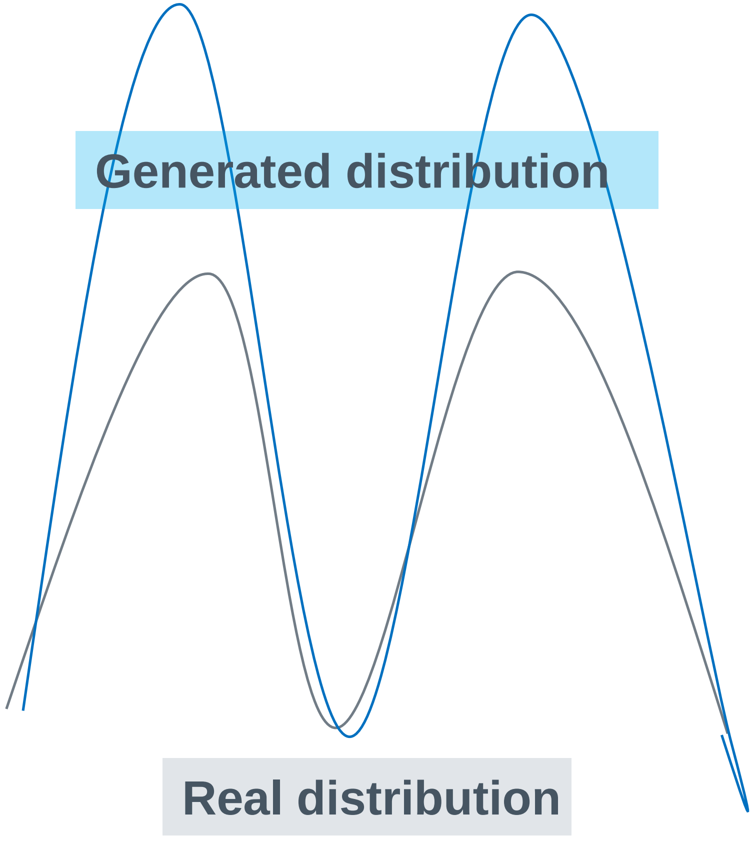
- We will use reverse KL-divergence instead of KL-divergence
  - To make the training faster

The Problems in GANs

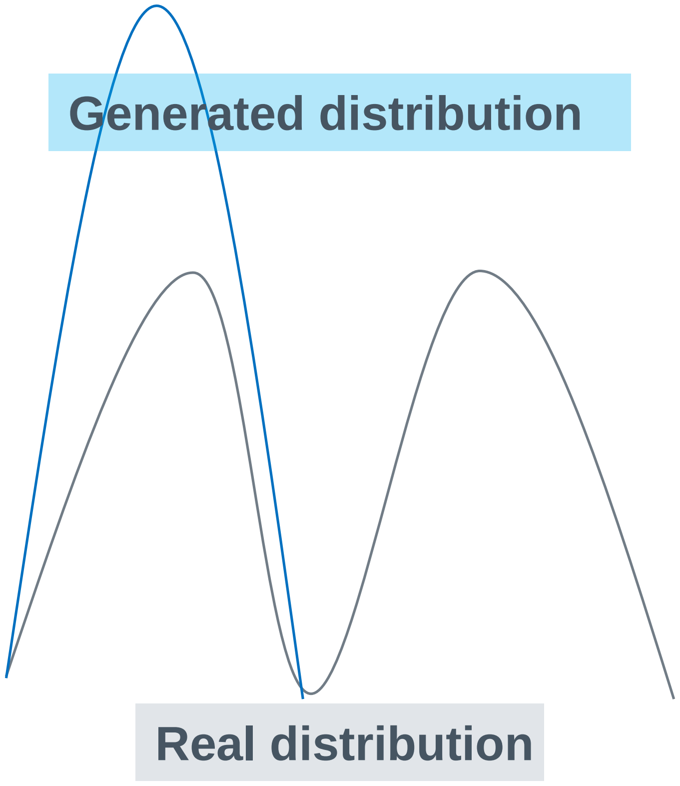
- The balance between generator and discriminator
- Mode collapse
- Gradient vanishing

Mode Collapse

We hope that



In real case

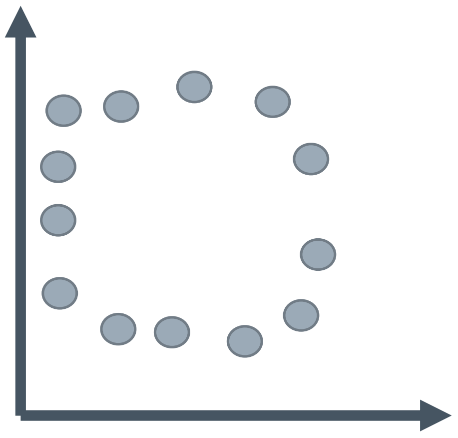


Mode Collapse

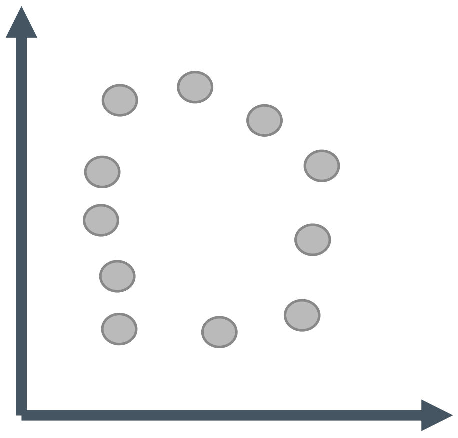
- Do not generate different kinds of objects
  - Will lead KL or JS divergence larger
  - The loss will be larger
- Tend to generate the same real objects
  - Will lead the smallest loss
- The generated object is very similar
  - Loss the diversity of the objects

Another Example for Mode Collapse

Real distribution



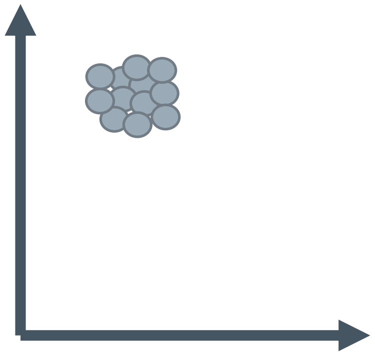
Ideal generated distribution



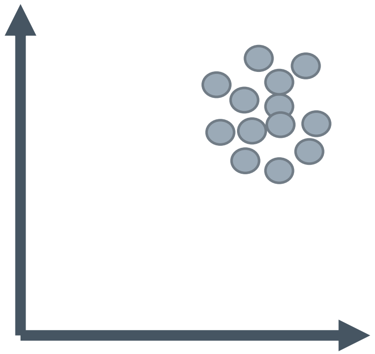
Another Example for Mode Collapse

Actually……

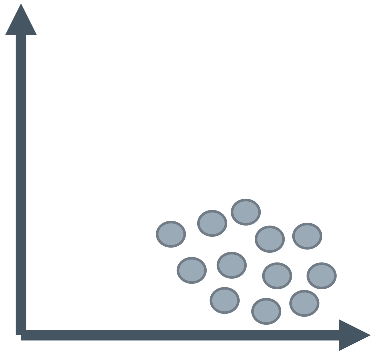
100^{th} iteration



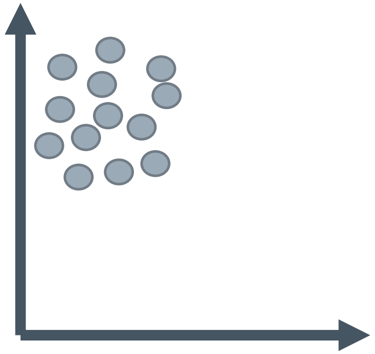
500^{th} iteration

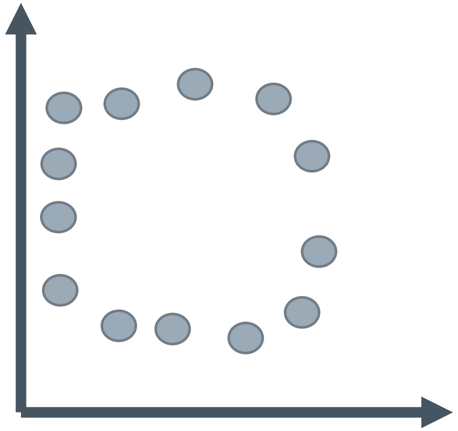


1000^{th} iteration



1500^{th} iteration





An infinite loop……
Cannot converge

Gradient Vanishing

1st iteration

P_G P_{data}

$JS(P_G||P_{data}) = log2$

Not become better

50th iteration

P_G P_{data}

$JS(P_G||P_{data}) = log2$

Too fast and hard

100th iteration

$P_G = P_{data}$

$JS(P_G||P_{data}) = 0$

If We Change The Distance Measurement?

1st iteration

P_G P_{data}

d_0

$$W(P_G||P_{data}) = d_0$$

50th iteration

P_G P_{data}

d_{50}

$$W(P_G||P_{data}) = d_{50}$$

100th iteration

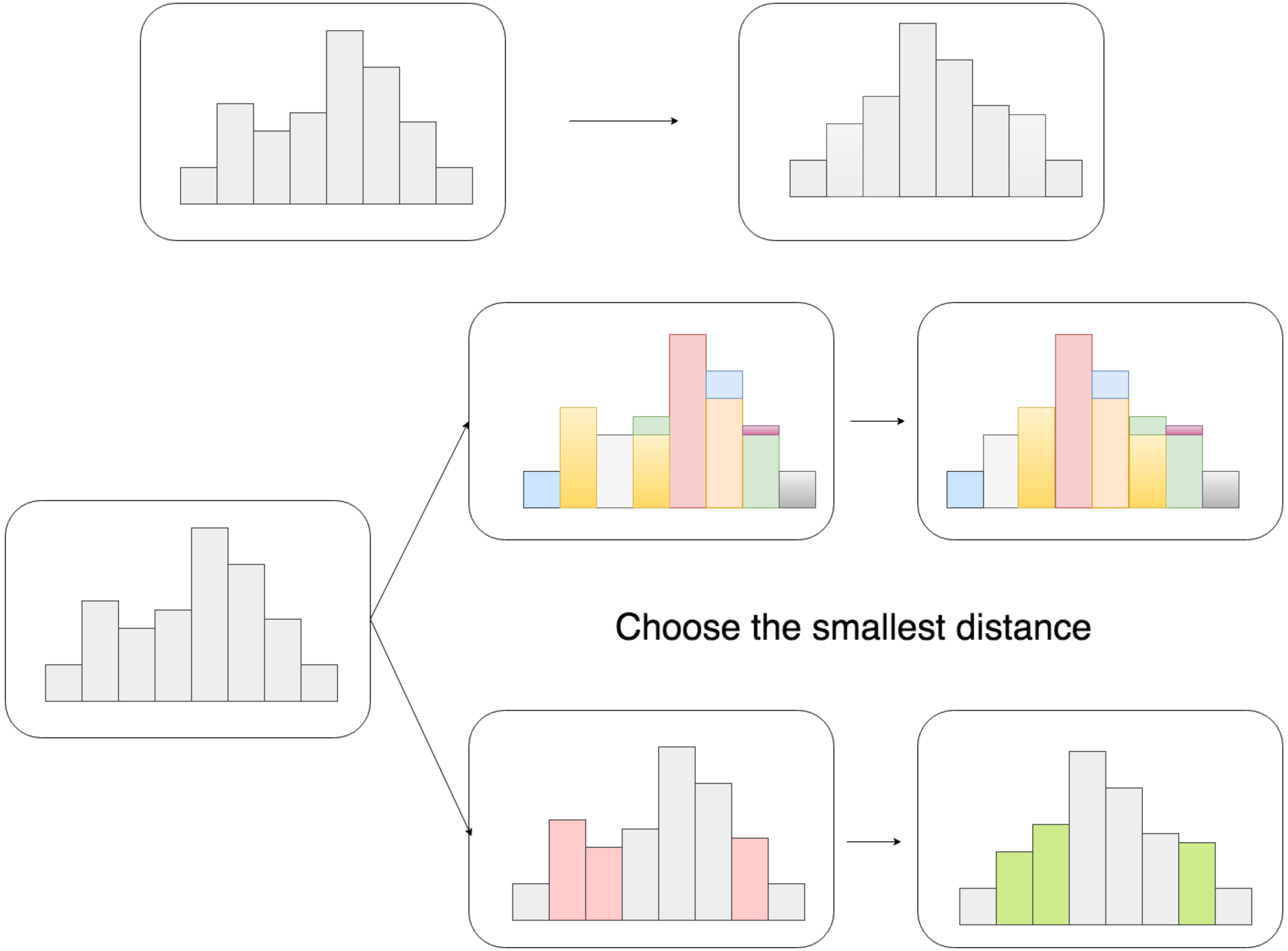
$P_G = P_{data}$

$$W(P_G||P_{data}) = 0$$

Wasserstein Distance

- Also known as Earth Mover Distance
- The distance from a distribution to another distribution
- More simplify：
  - Two distributions are places
  - There is a pile of earth on the first distribution
  - We want to move the earth to the second distribution
  - The average distance is Earth Mover Distance

Earth Mover Distance



Wasserstein Generative Adversarial Network (WGAN)

- Using Wasserstein distance instead of KL divergence or JS divergence
- Can solve the problems on GAN
  - The balance on generator and discriminator
  - Mode collapse
  - Gradient vanishing

The Modifications in WGAN

- Using Wasserstein distance instead of KL divergence or JS divergence

- Without sigmoid activation in the output layer
- Using real error instead log error
- Weight clipping
  - If $w < -c$, let $w = -c$
  - If $w > c$, let $w = c$
- Use SGD instead of momentum based optimization

The Notions and Suggestions

- The performance of generator and discriminator
  - The discriminator should not be very strong
- The diversity of inputs and outputs
  - Especially the diversity of outputs
  - To prevent mode collapse
- Consider to use WGAN instead of GANs
  - Prevent the problems in GANs