aCubeIT

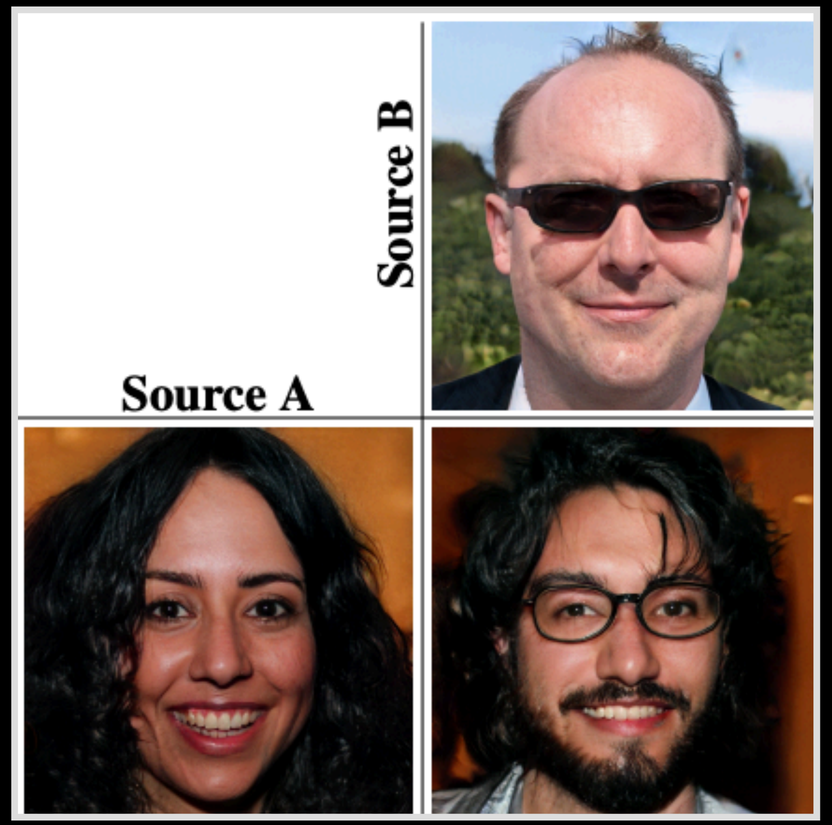
Generative Adversarial Networks

—

L03 - Deep Learning for Life Sciences Course

Generative Adversarial Networks (GANs)

Creating content



Which face is fake ?

Karras et al., 2019. *Analyzing and improving the image quality of stylegan.*

Generative Adversarial Networks (GANs)

2014

2015

2016

2017

2018

2019

Generative Adversarial Networks (GANs)


2014


2015


2016


2017


2018


2019

- Generative modelling → progress in facial image generation

Generative Adversarial Networks (GANs)


2014


2015


2016


2017


2018


2019

- Generative modelling → progress in facial image generation
- Newer and more sophisticated methods every year

Supervised Learning

Supervised Learning

1	...	...
2	...	...
3	...	...

Training Data

Supervised Learning

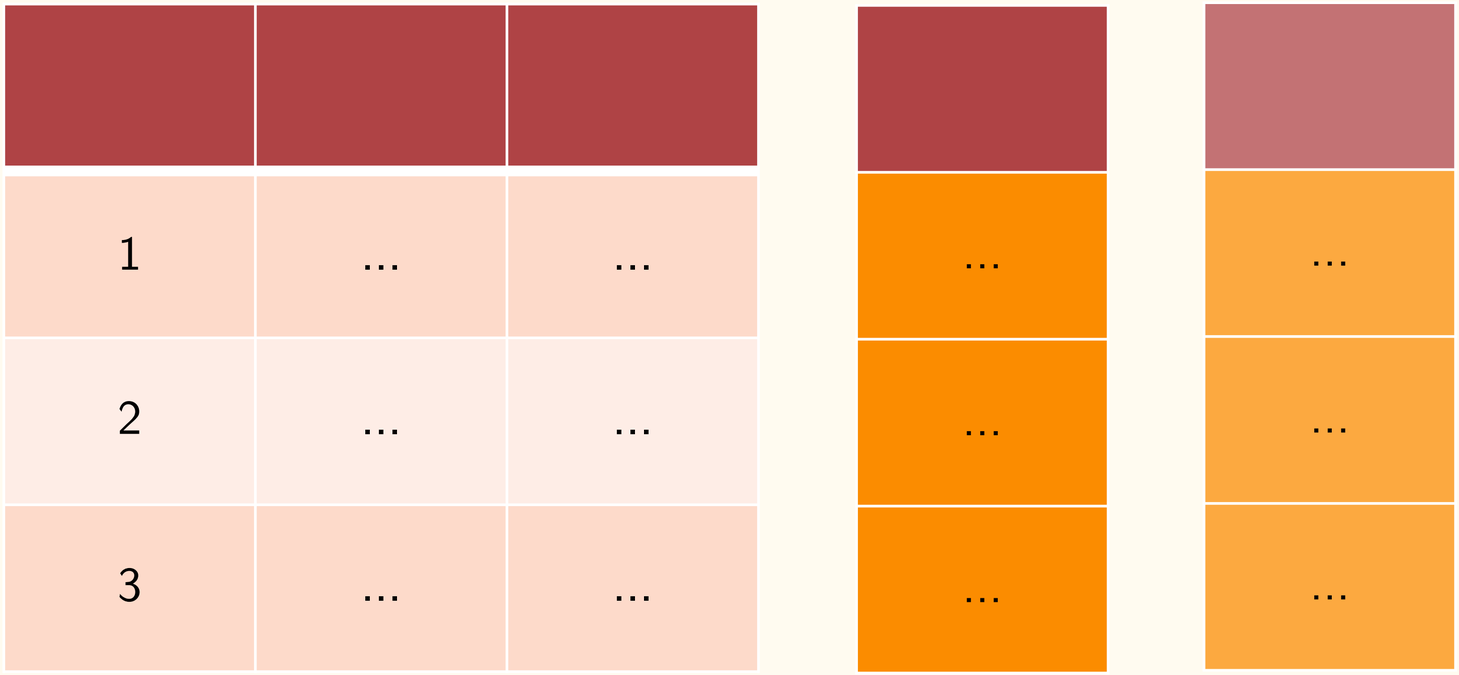
1	...	...
2	...	...
3	...	...

Training Data

...
...
...

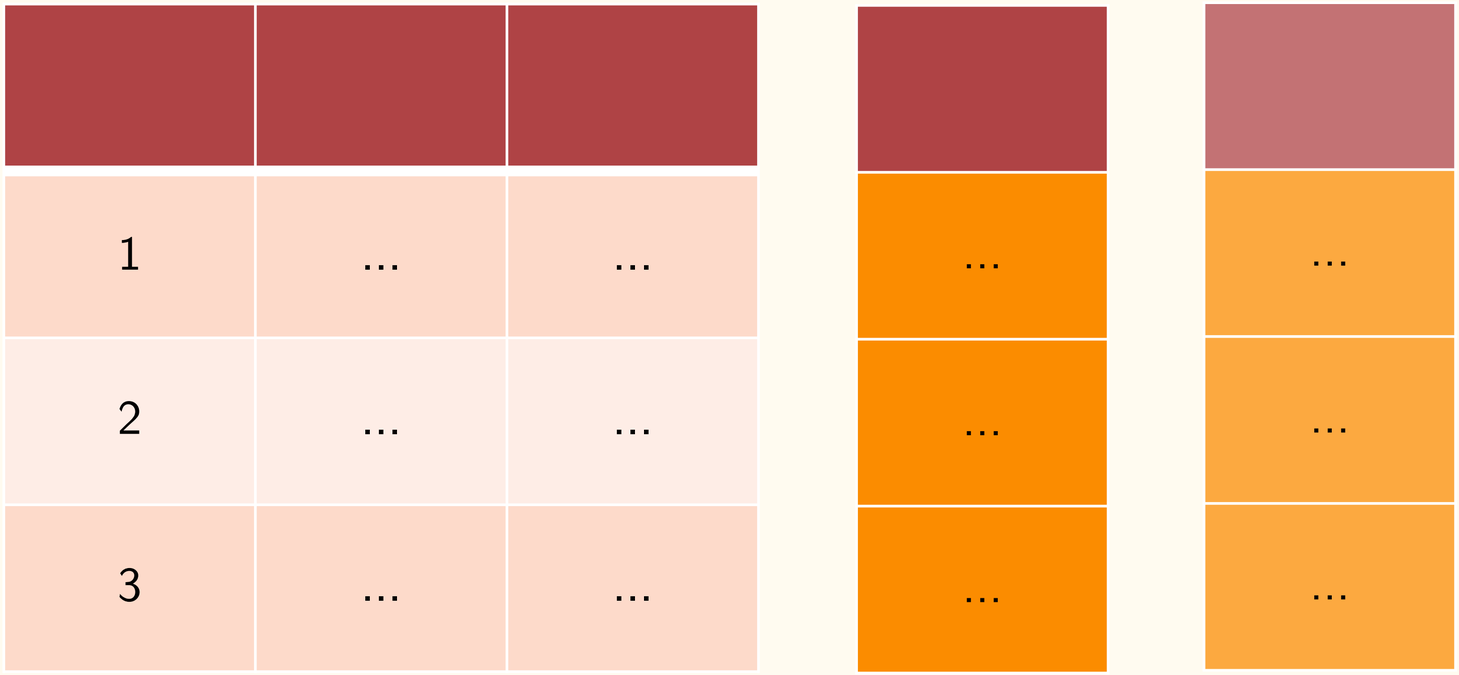
Labels

Supervised Learning



Training Data Labels Predictions

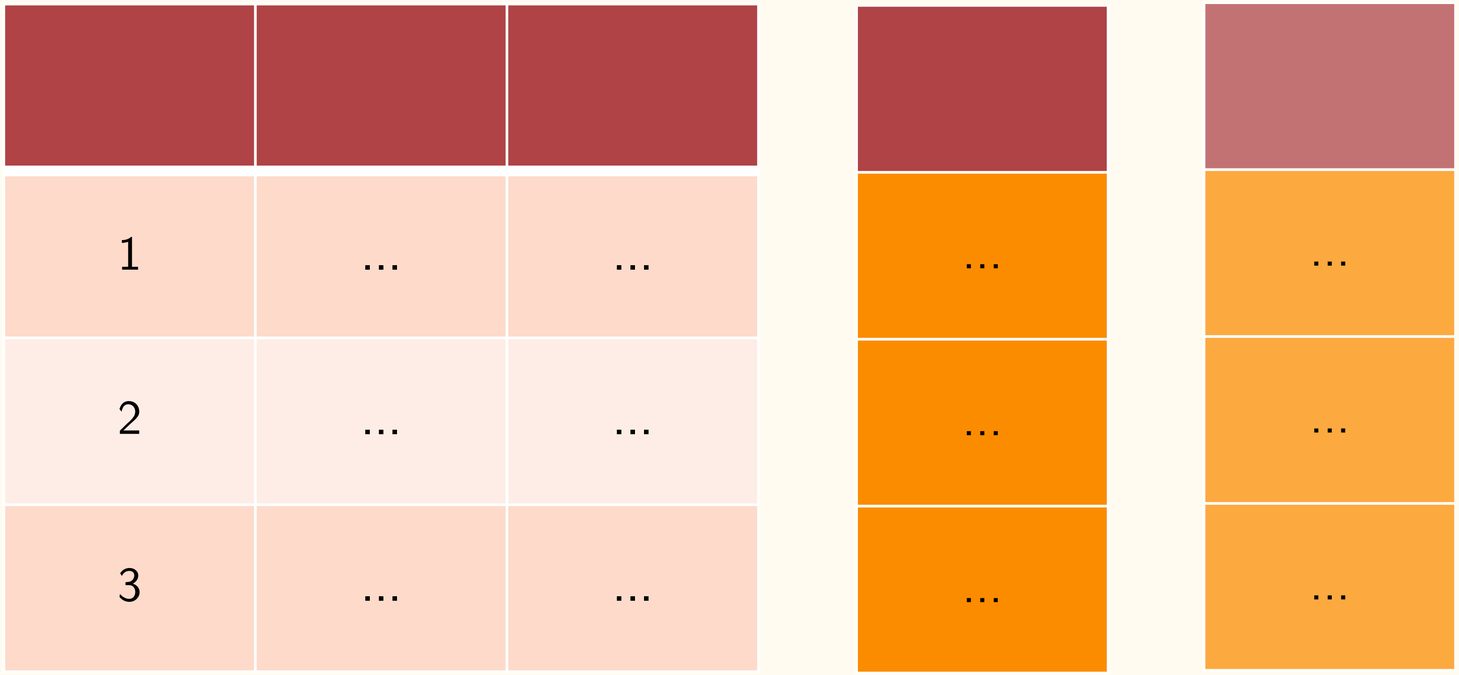
Supervised Learning



Training Data Labels Predictions

So far we've seen models which:

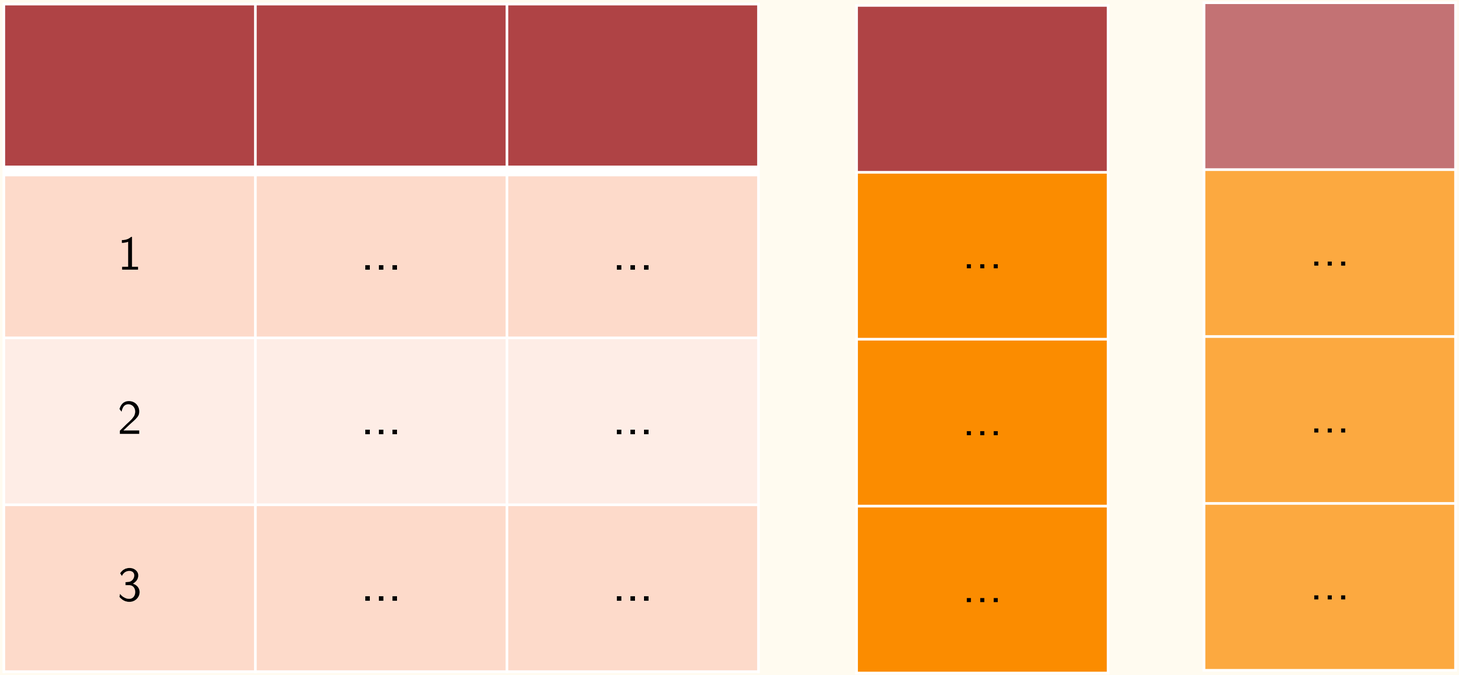
Supervised Learning



Training Data Labels Predictions

So far we've seen models which:

Supervised Learning

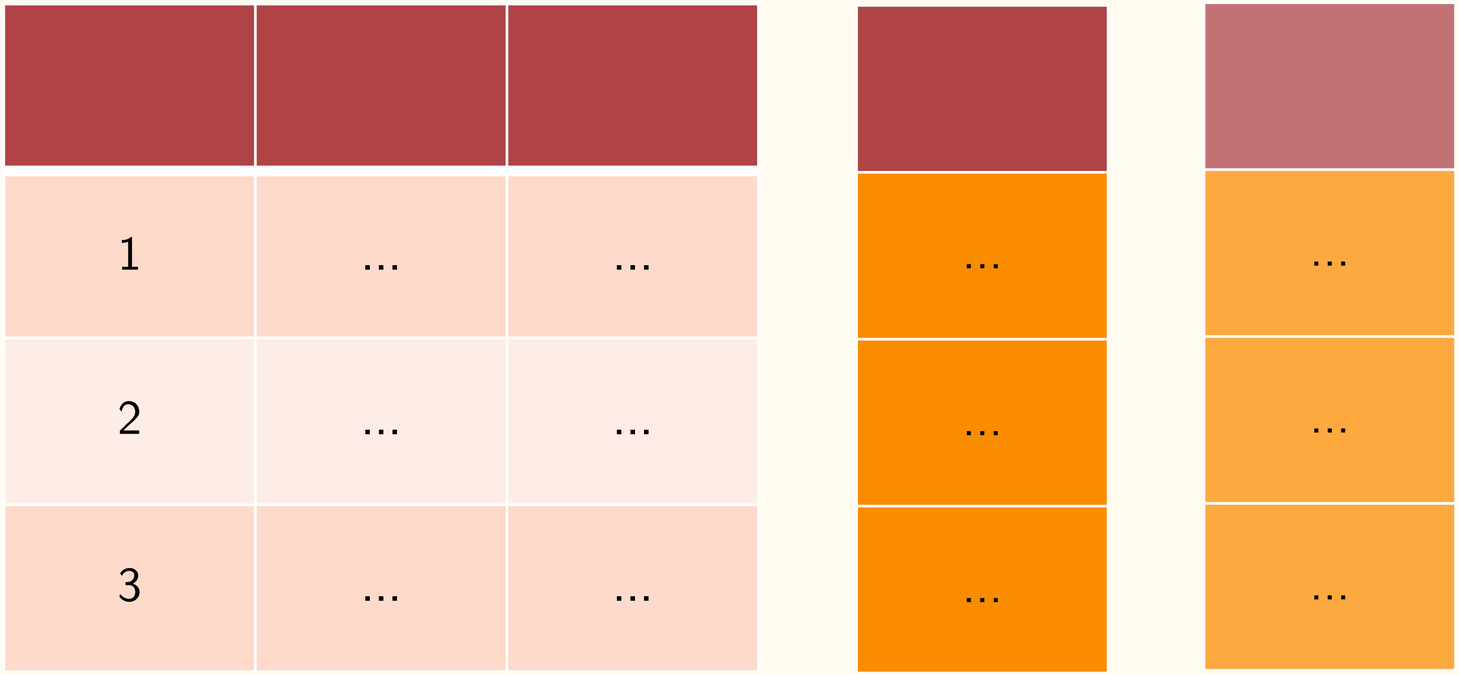


Training Data Labels Predictions

So far we've seen models which:

- Take an *input*

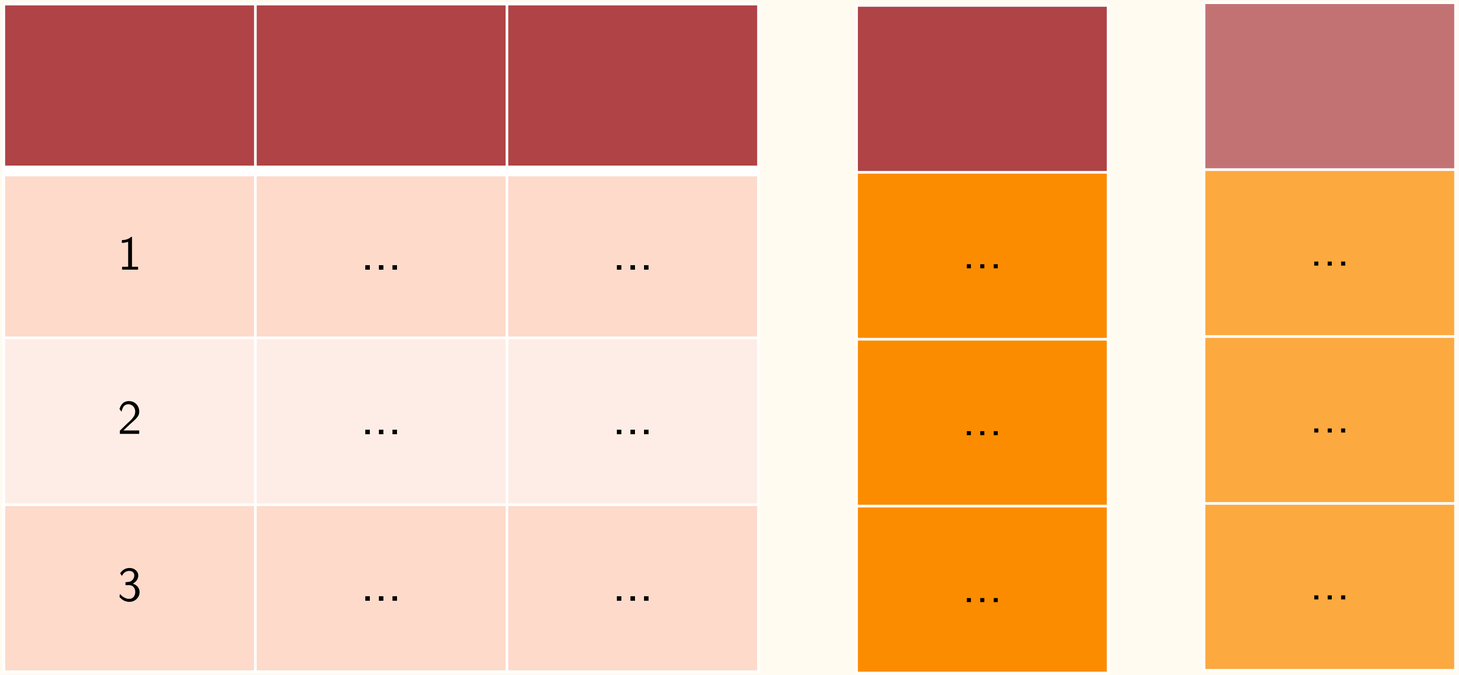
Supervised Learning



So far we've seen models which:

- Take an *input*
- Produce an *output*

Supervised Learning

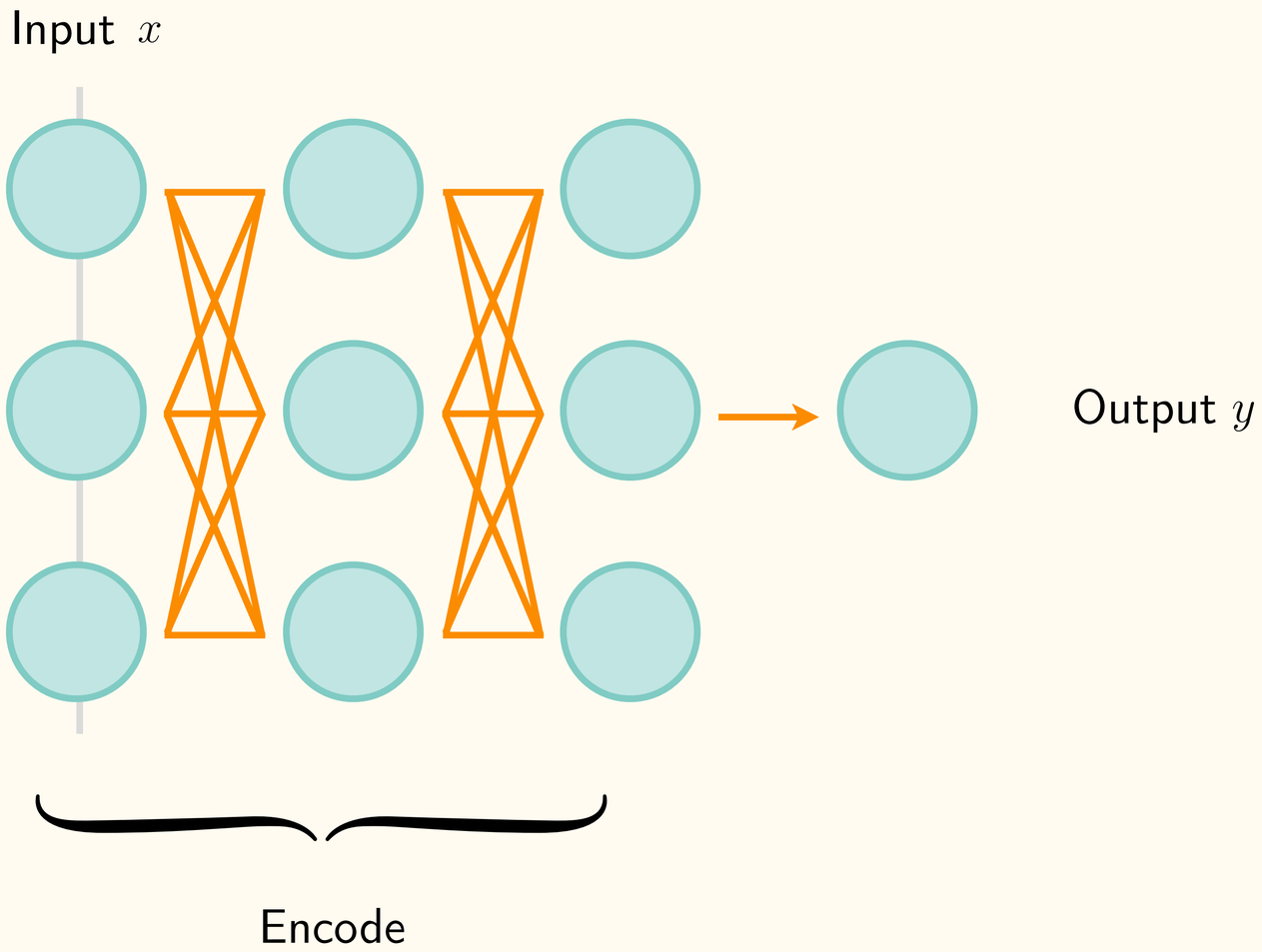


Training Data Labels Predictions

So far we've seen models which:

- Take an *input*
- Produce an *output*
- Optimised with a *loss-function*

Supervised Learning



So far we've seen models which:

- Take an *input*
- Produce an *output*
- Optimised with a *loss-function*

Unsupervised Learning

Unsupervised Learning

- Generative models are different.

Unsupervised Learning

- Generative models are different.

Unsupervised Learning

- Generative models are different.

- Instead of taking a sample as *input*, they produce a sample as *output*.

Unsupervised Learning

Unsupervised Learning

- Train model on photographs of cats

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images
- Train it on known drug molecules

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images

- Train it on known drug molecules
  - Generate new ‘drug-like’ molecules to use as candidates in a virtual screen.

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images
- Train it on known drug molecules
  - Generate new ‘drug-like’ molecules to use as candidates in a virtual screen.
- A generative model is trained on samples which are drawn from a:

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images

- Train it on known drug molecules
  - Generate new ‘drug-like’ molecules to use as candidates in a virtual screen.

- A generative model is trained on samples which are drawn from a:

probability distribution

Unsupervised Learning

- Train model on photographs of cats
  - Learn to produce new cat-like images

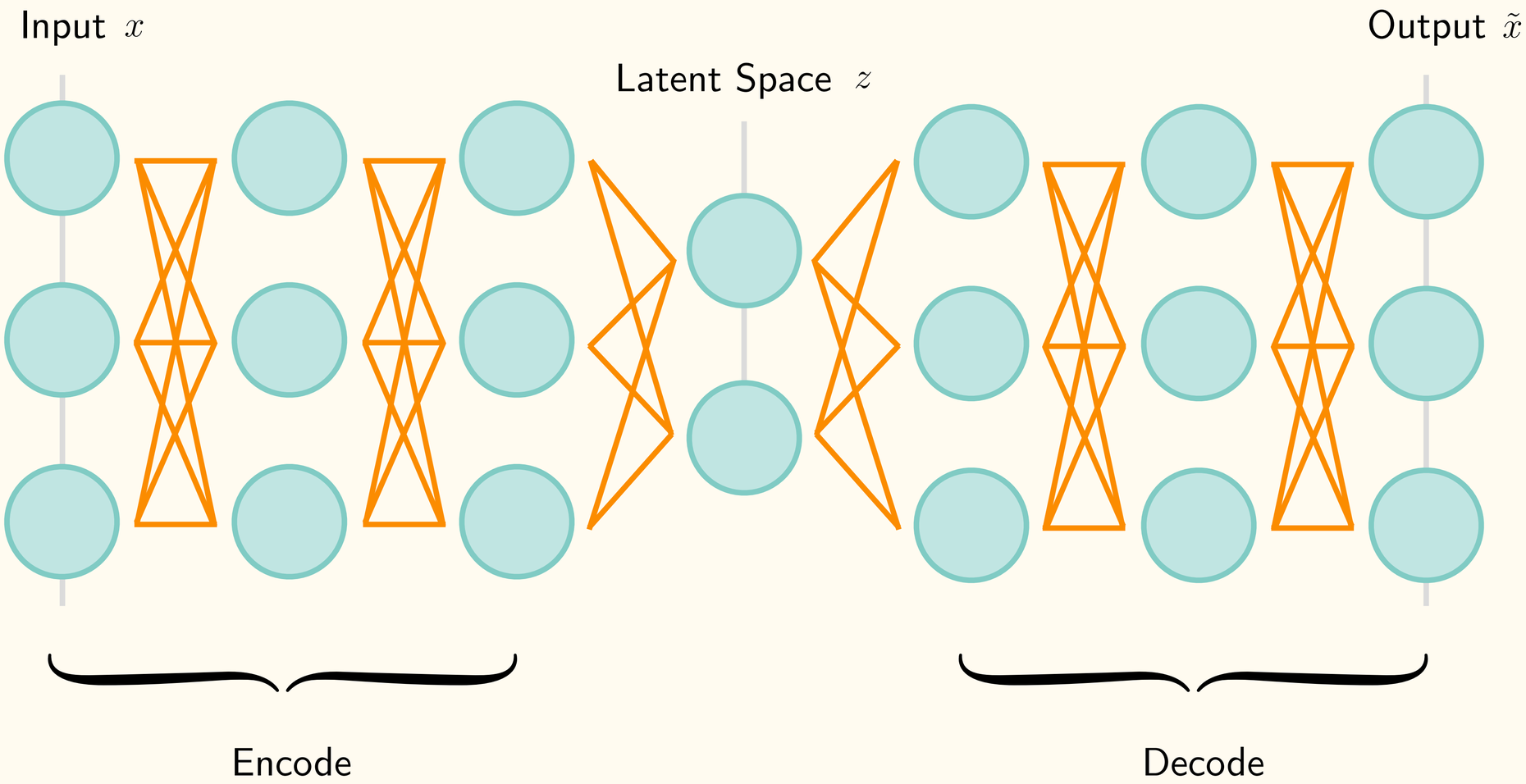
- Train it on known drug molecules
  - Generate new ‘drug-like’ molecules to use as candidates in a virtual screen.

- A generative model is trained on samples which are drawn from a:

probability distribution

- Its job is to produce new samples from that same probability distribution.

Auto-Encoder



Auto-Encoder

Auto-Encoder

- **Encoder**: samples → compressed representations

Auto-Encoder

- **Encoder**: samples → compressed representations
- **Decoder**: compressed representations in latent space → original samples

Auto-Encoder

- **Encoder**: samples → compressed representations
- **Decoder**: compressed representations in latent space → original samples

Auto-Encoder

- **Encoder**: samples → compressed representations
- **Decoder**: compressed representations in latent space → original samples

- What if we take *random vectors* in the latent space (picking a random value for each component of the vector) and pass them through the decoder ?

Auto-Encoder

- **Encoder**: samples → compressed representations
- **Decoder**: compressed representations in latent space → original samples

- What if we take *random vectors* in the latent space (picking a random value for each component of the vector) and pass them through the decoder ?

Auto-Encoder

- **Encoder**: samples → compressed representations
- **Decoder**: compressed representations in latent space → original samples

- What if we take *random vectors* in the latent space (picking a random value for each component of the vector) and pass them through the decoder ?

- If everything goes well, the decoder should produce a completely *new sample* that *resembles* the ones it was trained on.

Auto-Encoder

Auto-Encoder

- Problem ?

Auto-Encoder

- Problem ?
  - Doesn't work very well.

Auto-Encoder

- Problem ?
  - Doesn't work very well.

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.
- But if we pick a vector from *outside* that region ?

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.
- But if we pick a vector from *outside* that region ?
  - Output could look nothing like the training samples

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.
- But if we pick a vector from *outside* that region ?
  - Output could look nothing like the training samples

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.
- But if we pick a vector from *outside* that region ?
  - Output could look nothing like the training samples

- Decoder has only learned to work for the *particular* latent vectors from encoder

Auto-Encoder

- Problem ?
  - Doesn't work very well.

- Encoder may only produce vectors in a *small* region of the latent space.
- But if we pick a vector from *outside* that region ?
  - Output could look nothing like the training samples

- Decoder has only learned to work for the *particular* latent vectors from encoder
  - Not for *arbitrary* ones.

Variational Auto-Encoder (VAE)

Variational Auto-Encoder (VAE)

- Solution ?

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

- Add a term to the loss function that *forces* the latent vectors to follow a specified distribution, eg N ~ (0, 1)

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

- Add a term to the loss function that *forces* the latent vectors to follow a specified distribution, eg N ~ (0, 1)
  - We can *expect* the decoder to work well on them

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

- Add a term to the loss function that *forces* the latent vectors to follow a specified distribution, eg N ~ (0, 1)
  - We can *expect* the decoder to work well on them

Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

- Add a term to the loss function that *forces* the latent vectors to follow a specified distribution, eg N ~ (0, 1)
  - We can *expect* the decoder to work well on them

- We add random noise to the latent vector

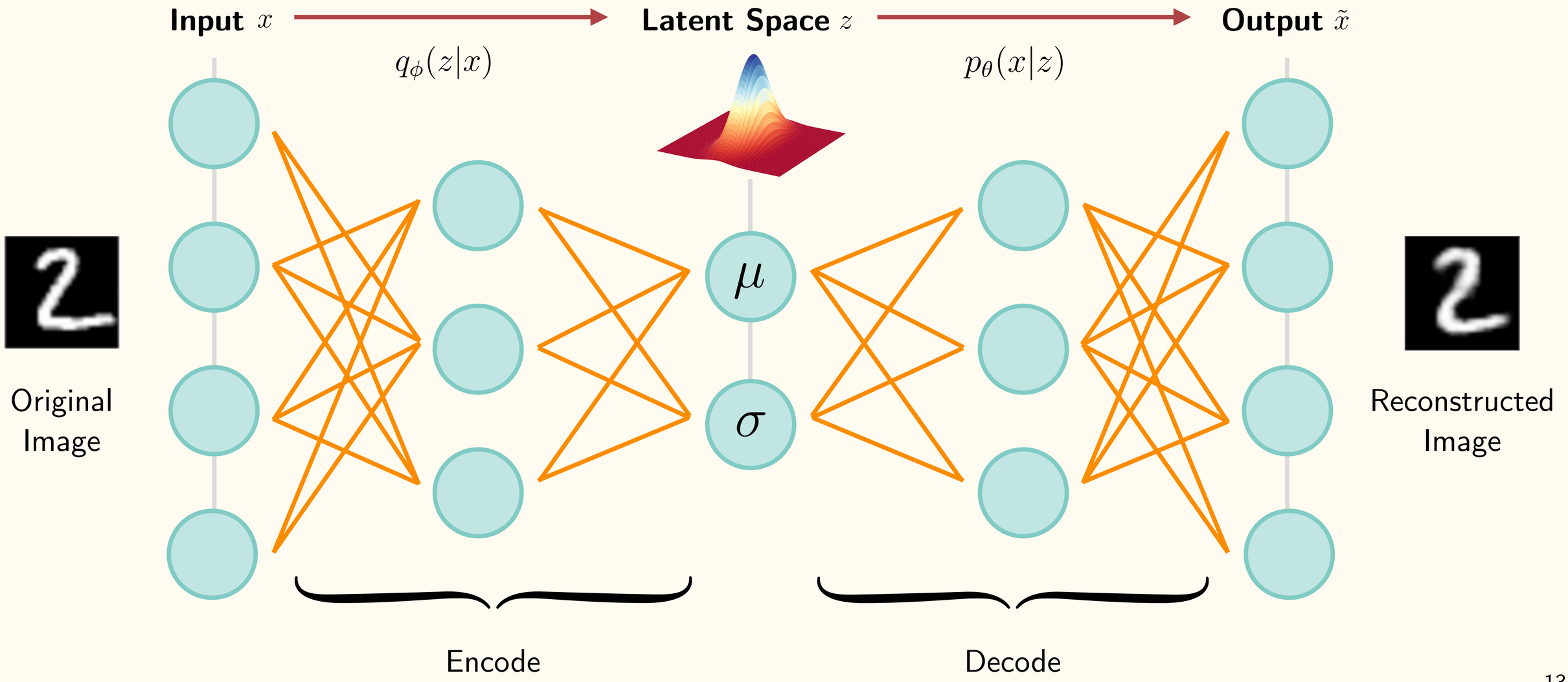
Variational Auto-Encoder (VAE)

- Solution ?
  - Variational Auto-Encoder

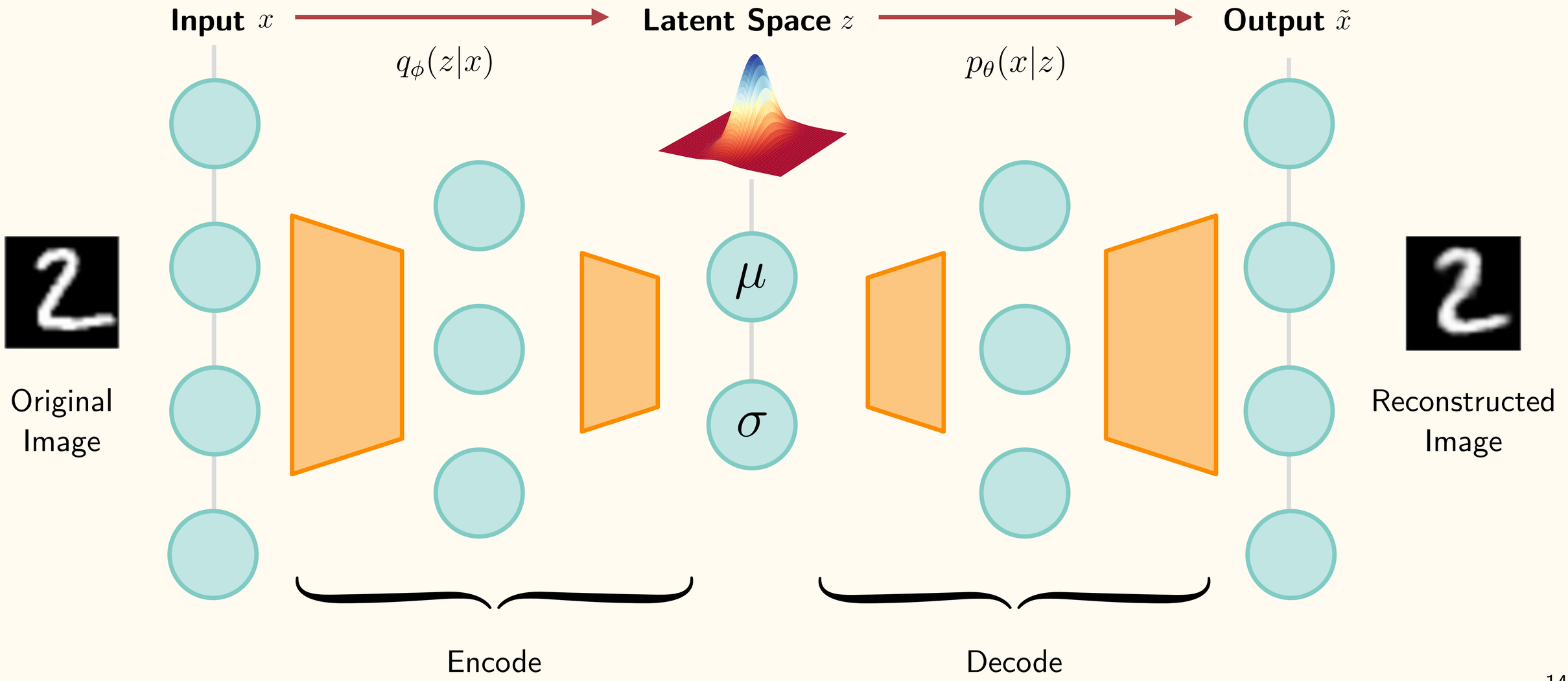
- Add a term to the loss function that *forces* the latent vectors to follow a specified distribution, eg N ~ (0, 1)
  - We can *expect* the decoder to work well on them

- We add random noise to the latent vector
  - Prevents decoder from being too sensitive to details

Variational Auto-Encoder (VAE)

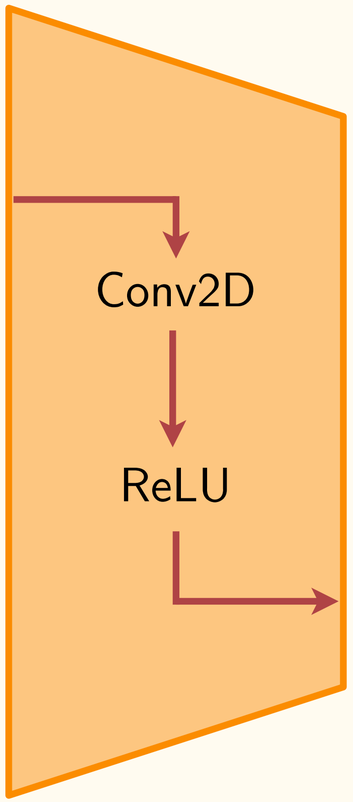


Simplified VAE Representation

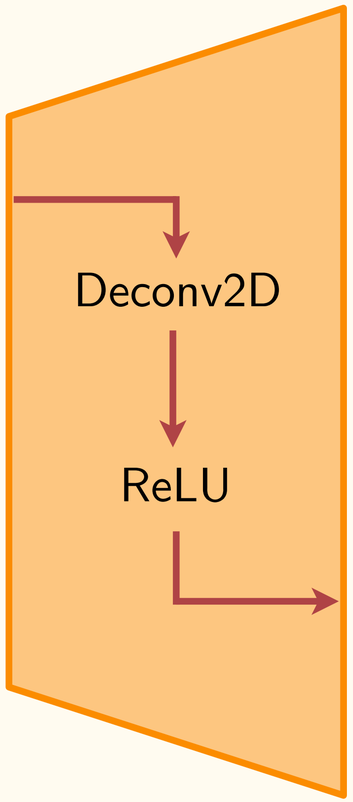


Encoder-Decoder

Encode

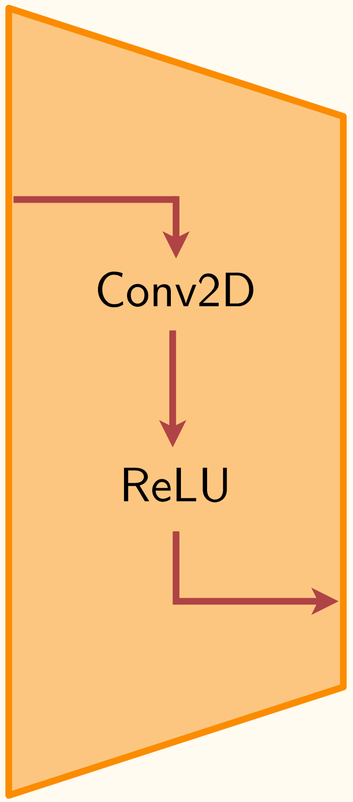


Decode



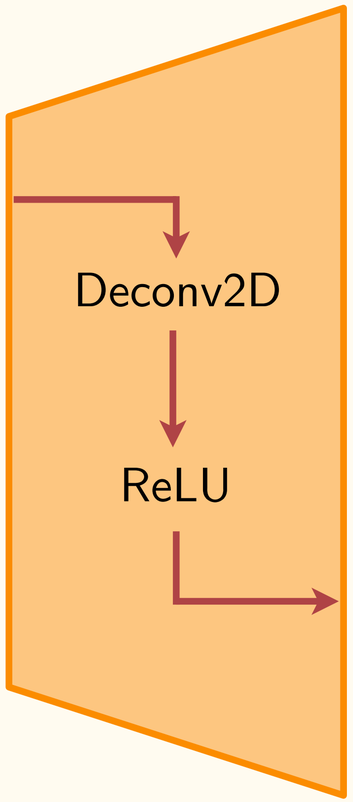
Encoder-Decoder

Encode



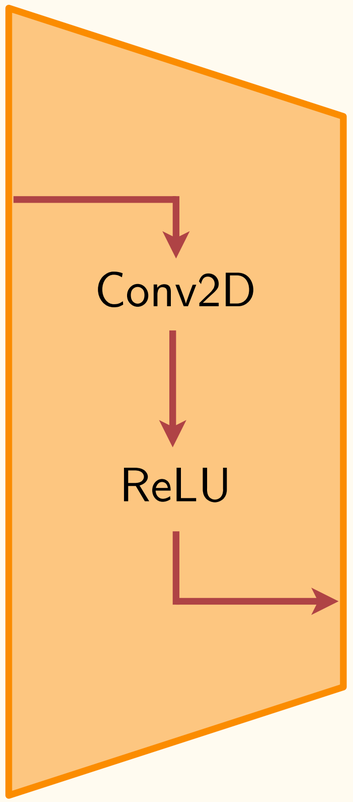
- An **encoder** is just a convolutional operation coupled with an activation function

Decode



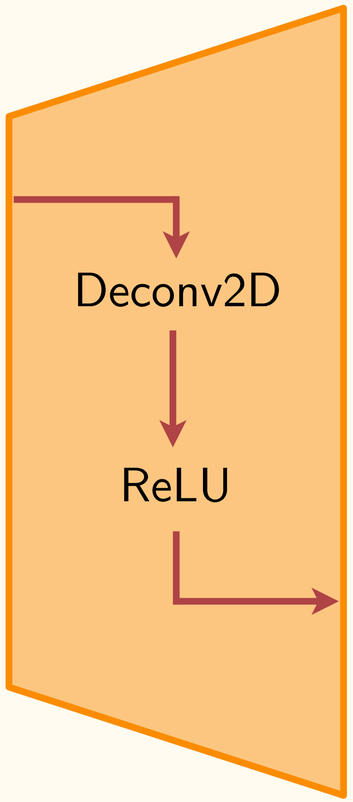
Encoder-Decoder

Encode

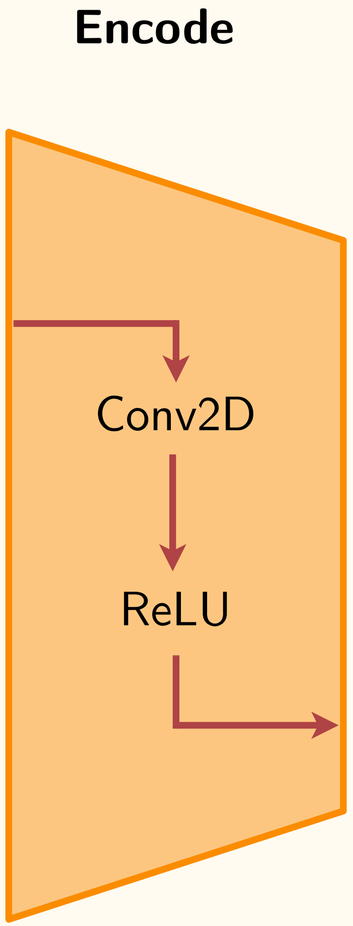


- An **encoder** is just a convolutional operation coupled with an activation function

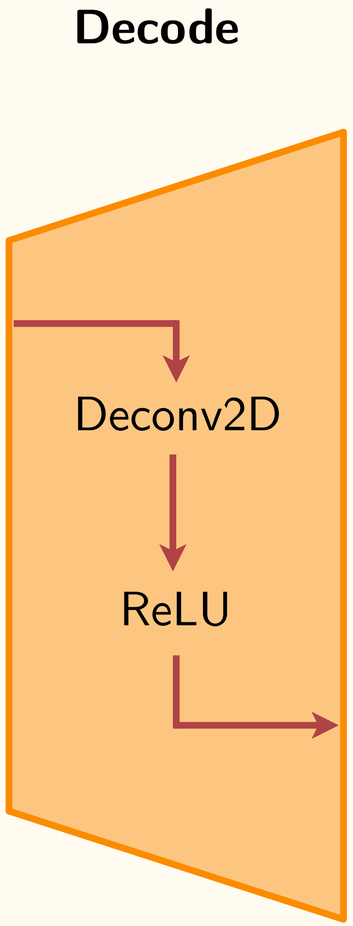
Decode



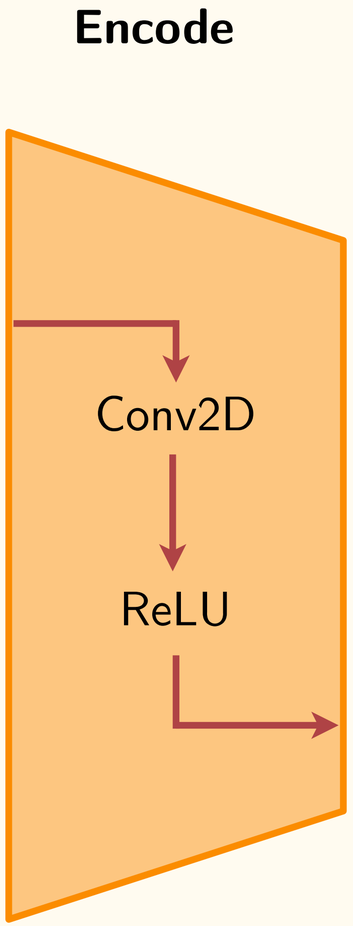
Encoder-Decoder



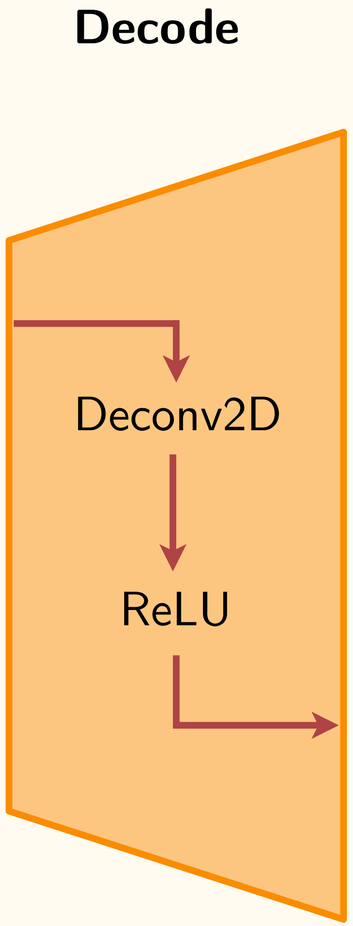
- An **encoder** is just a convolutional operation coupled with an activation function
- A **decoder** is de-convolutional operation and activation function



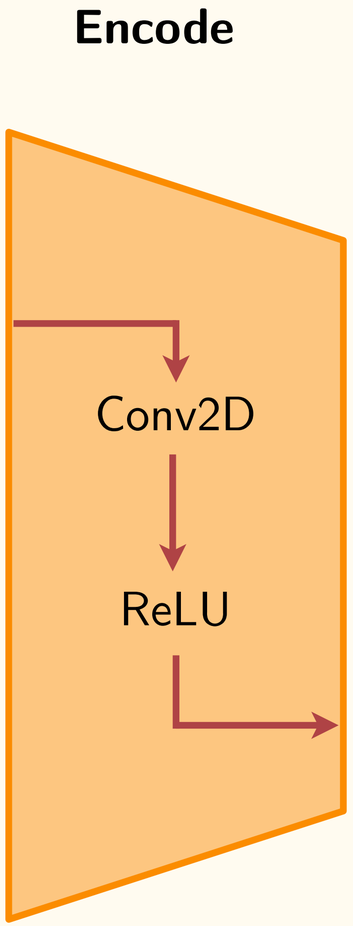
Encoder-Decoder



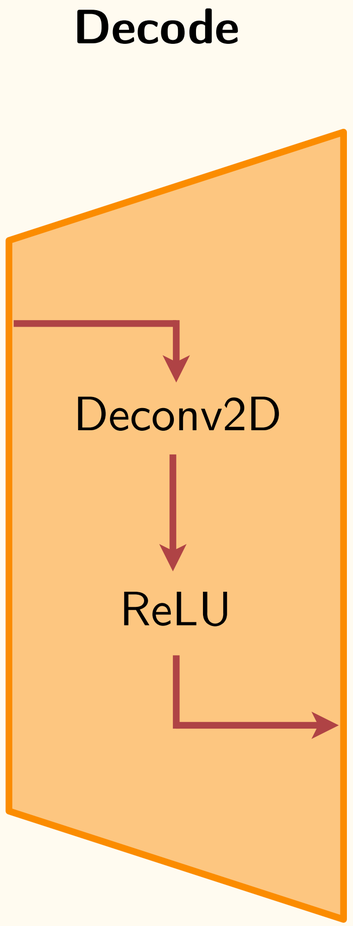
- An **encoder** is just a convolutional operation coupled with an activation function
- A **decoder** is de-convolutional operation and activation function



Encoder-Decoder

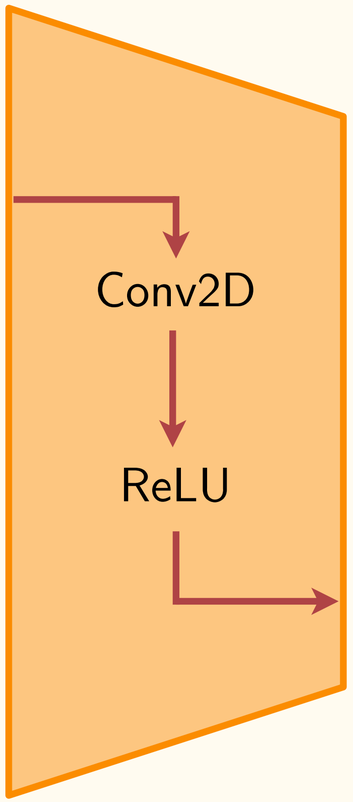


- An **encoder** is just a convolutional operation coupled with an activation function
- A **decoder** is de-convolutional operation and activation function
- De-conv2D is just a **transposed** Conv2D



Encoder-Decoder

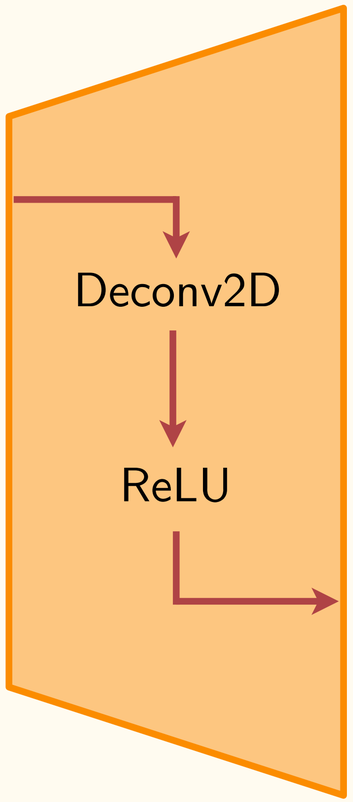
Encode



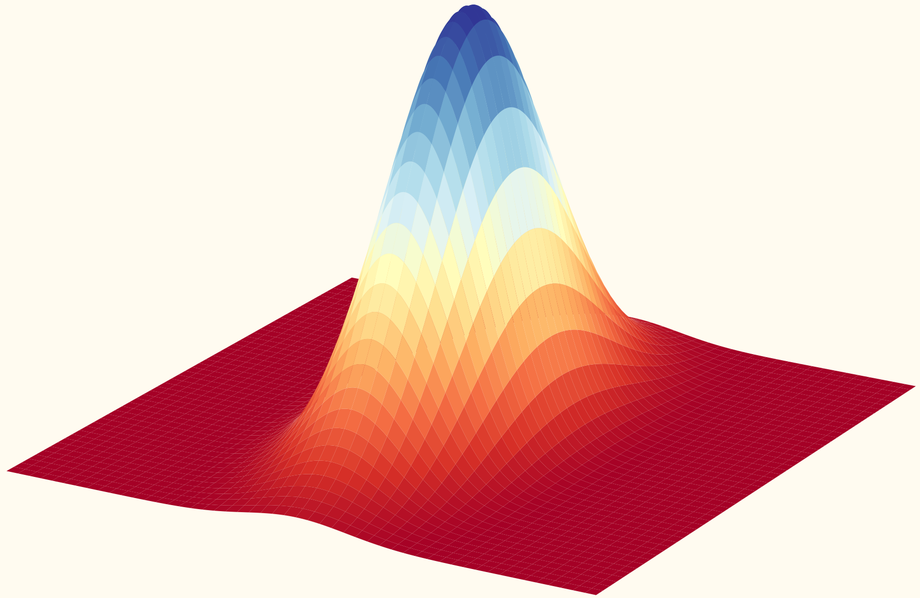

```
# Encoder
tf.nn.conv2d()
tf.nn.relu()

# Decoder
tf.nn.conv2d_transpose()
tf.nn.relu()
```

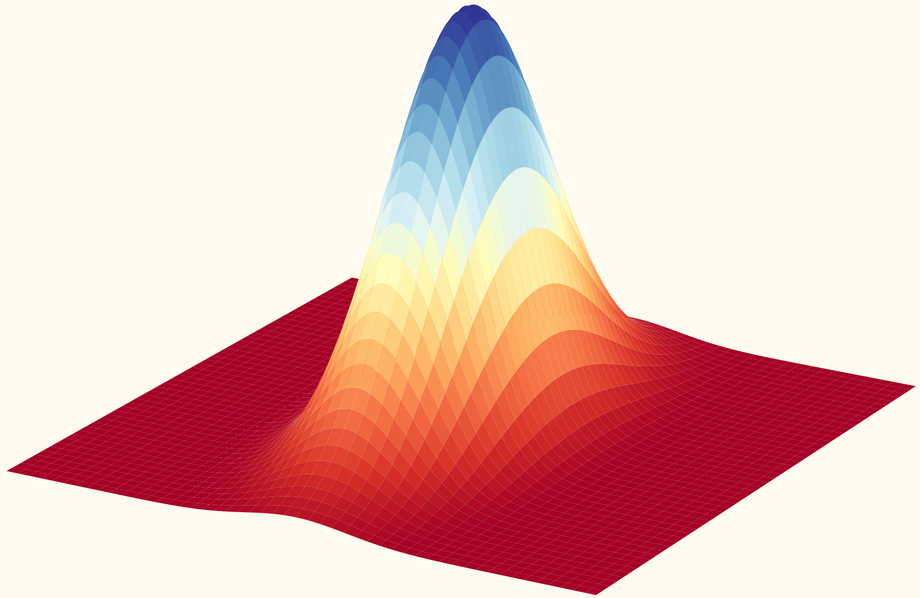
Decode



Latent Distribution

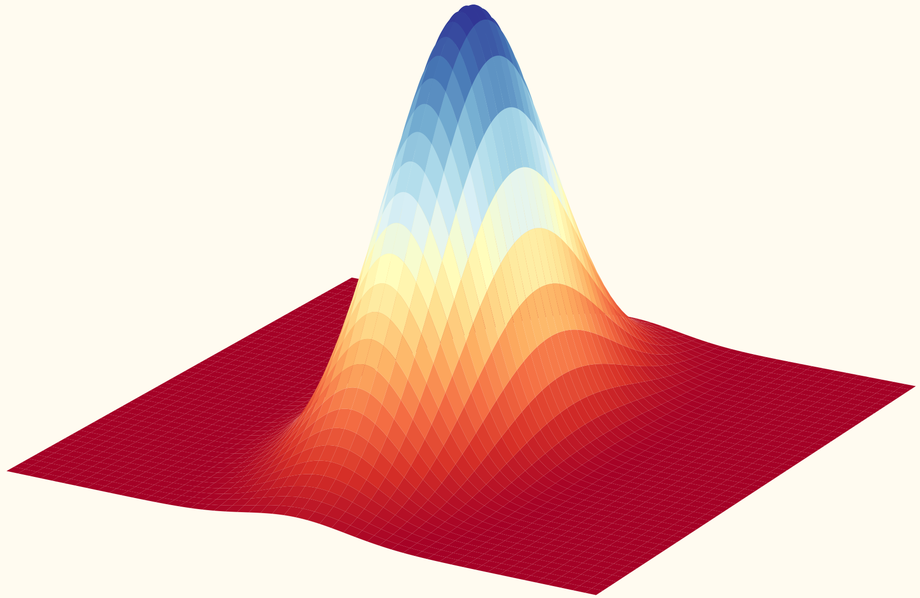


Latent Distribution



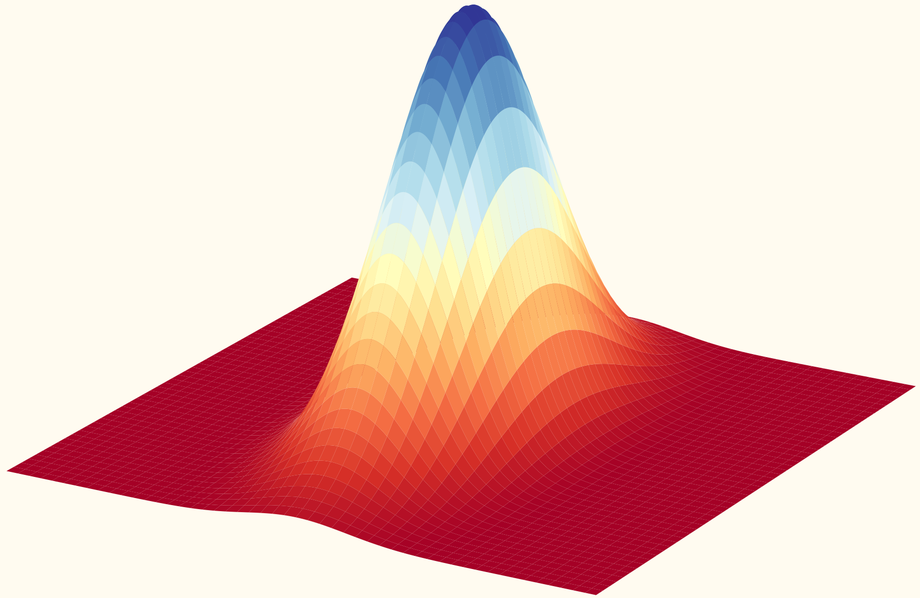
- The latent vector $z \sim (\mu, \sigma)$

Latent Distribution



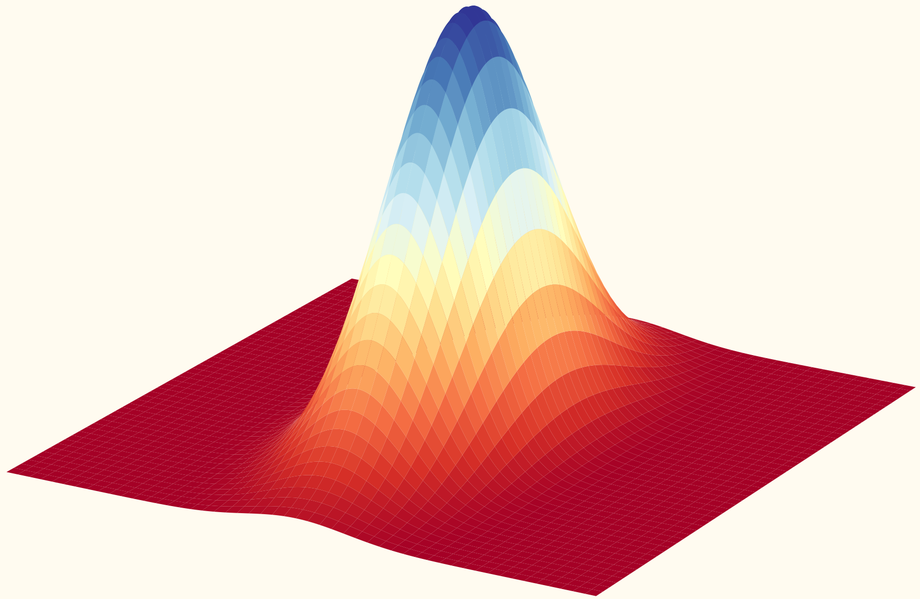
- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

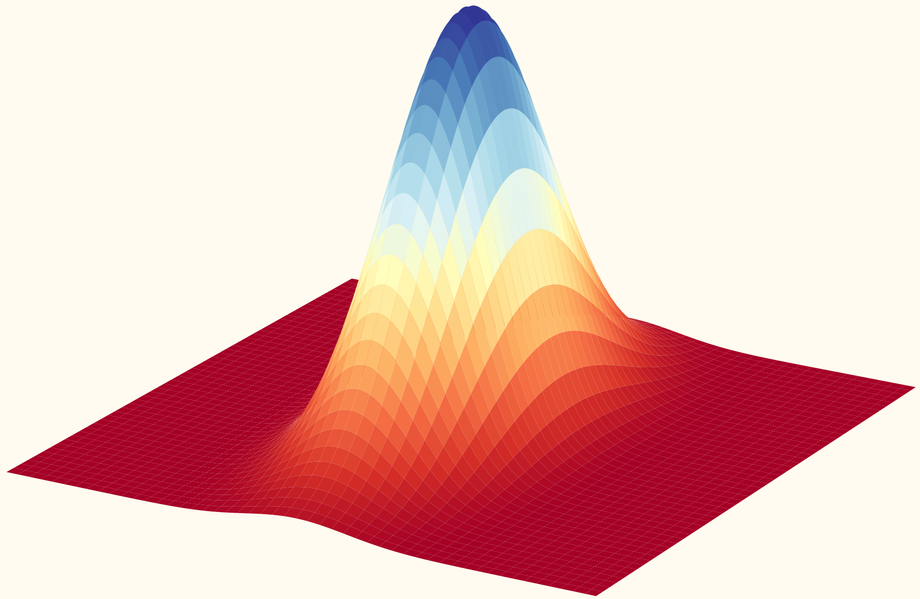
Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

- q_Φ(z|x) encodes x to z

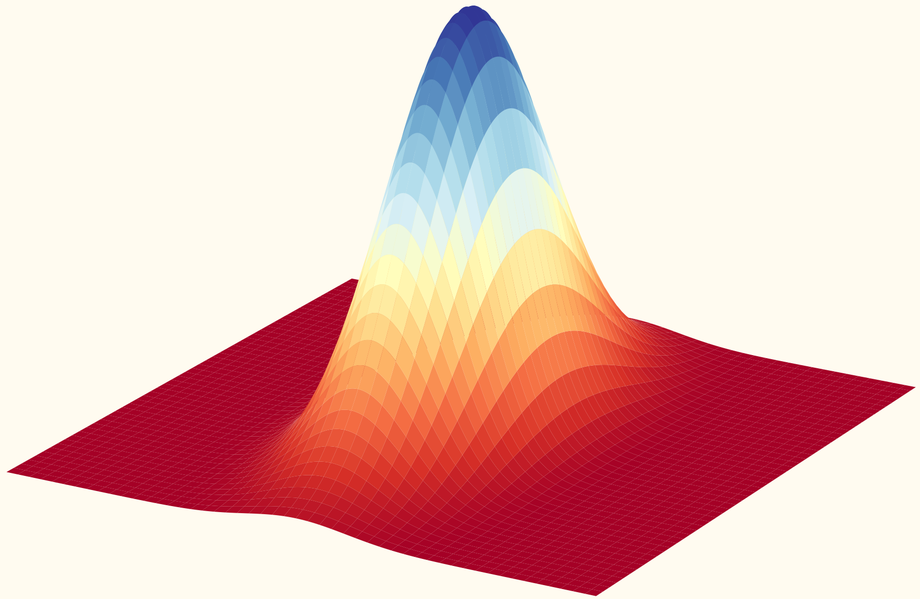
Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

- q_Φ(z|x) encodes x to z
- p_θ(x|z) decodes z to x

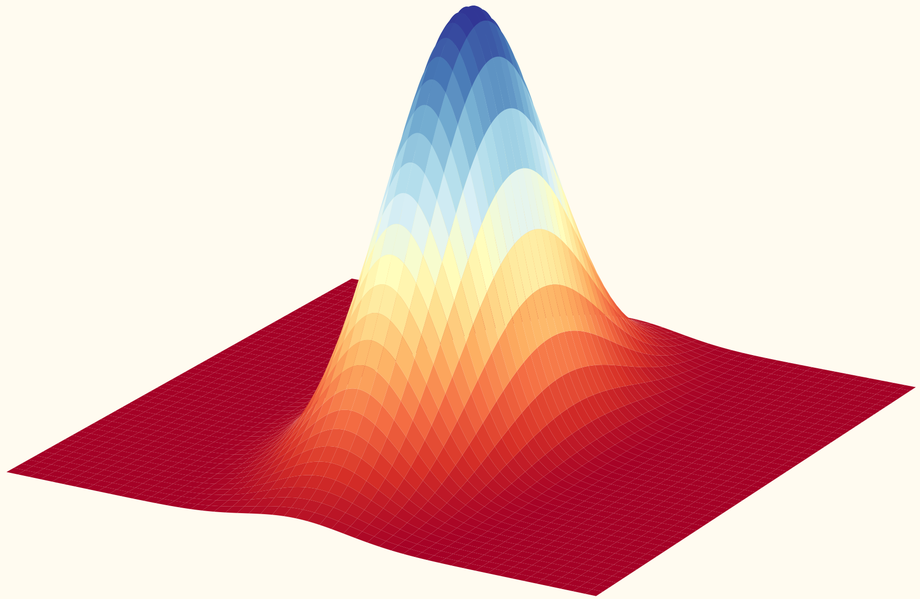
Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

- q_Φ(z|x) encodes x to z
- p_θ(x|z) decodes z to x
- θ and Φ represent weights and biases

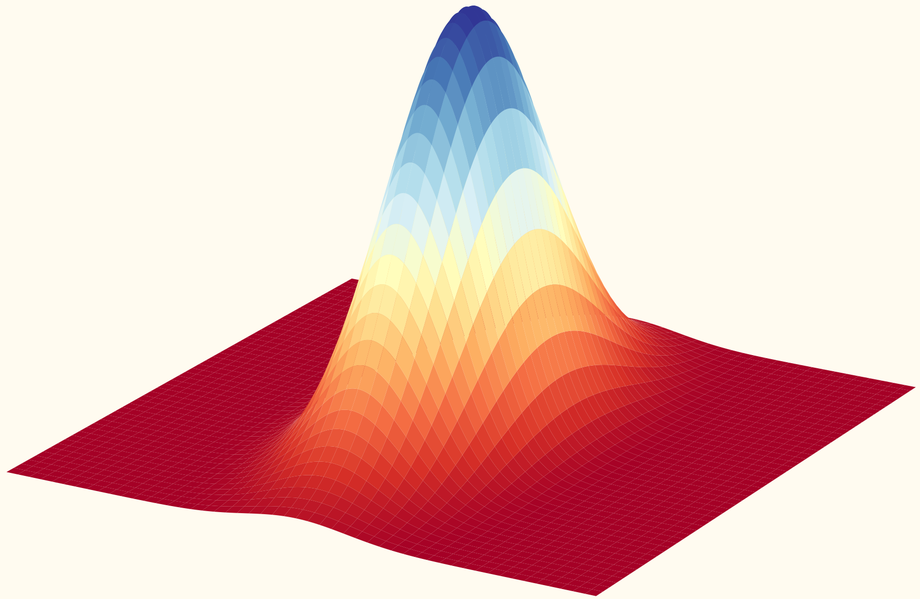
Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

- q_Φ(z|x) encodes x to z
- p_θ(x|z) decodes z to x
- θ and Φ represent weights and biases

Latent Distribution



- The latent vector $z \sim (\mu, \sigma)$
- Compressed representation of input x

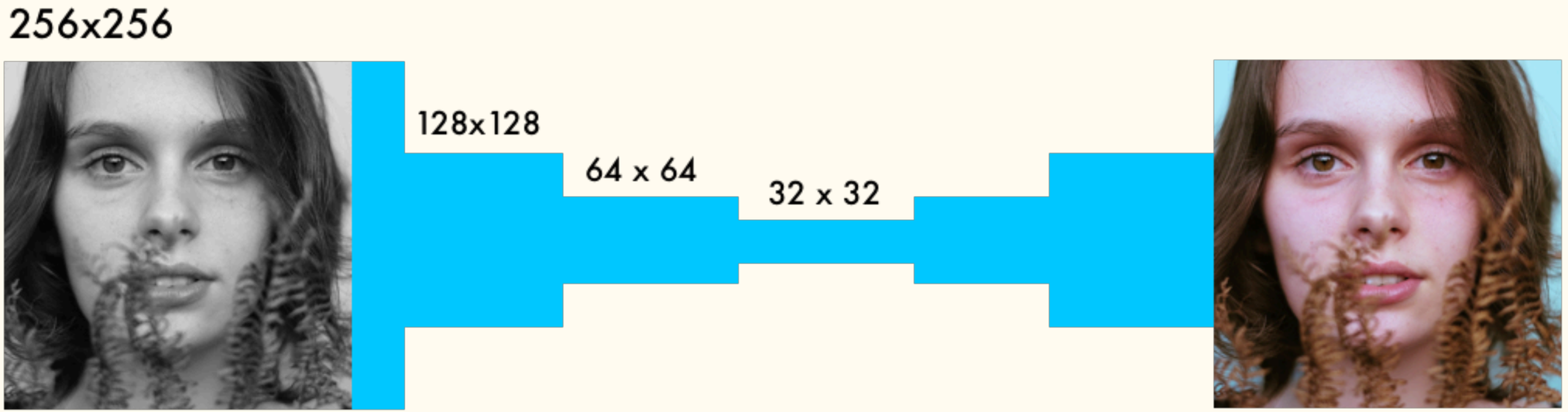
- q_Φ(z|x) encodes x to z
- p_θ(x|z) decodes z to x
- θ and Φ represent weights and biases

- Remember: we **sample** from the distribution z → lossy reconstruction

VAE Applications

Colorize B&W images

- Encoder-Decoder architecture
- Inception ResNet V2 Classifier



Emil Wallner, 2017. *Colorizing B&W Photos with Neural Networks.* FloydHub.

Generative Adversarial Network

Generative Adversarial Network

- Works similarly to VAEs

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples
  - ‘Generator’ instead of decoder

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples
  - ‘Generator’ instead of decoder

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples
  - ‘Generator’ instead of decoder

- But trains in a different way:

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples
  - ‘Generator’ instead of decoder

- But trains in a different way:
  - Passes random vectors into the generator

Generative Adversarial Network

- Works similarly to VAEs
  - Uses encoder-decoder system
  - Converts latent vectors into samples
  - ‘Generator’ instead of decoder

- But trains in a different way:
  - Passes random vectors into the generator
  - Directly evaluates the outputs on how well they follow the expected distribution

Generative Adversarial Network

Generative Adversarial Network

- Essentially create a loss function to measure:

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples
  - Use that loss function to optimise the model

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples
  - Use that loss function to optimise the model

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples
  - ▸ Use that loss function to optimise the model

- How to make such a loss-function ?

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples
  - Use that loss function to optimise the model

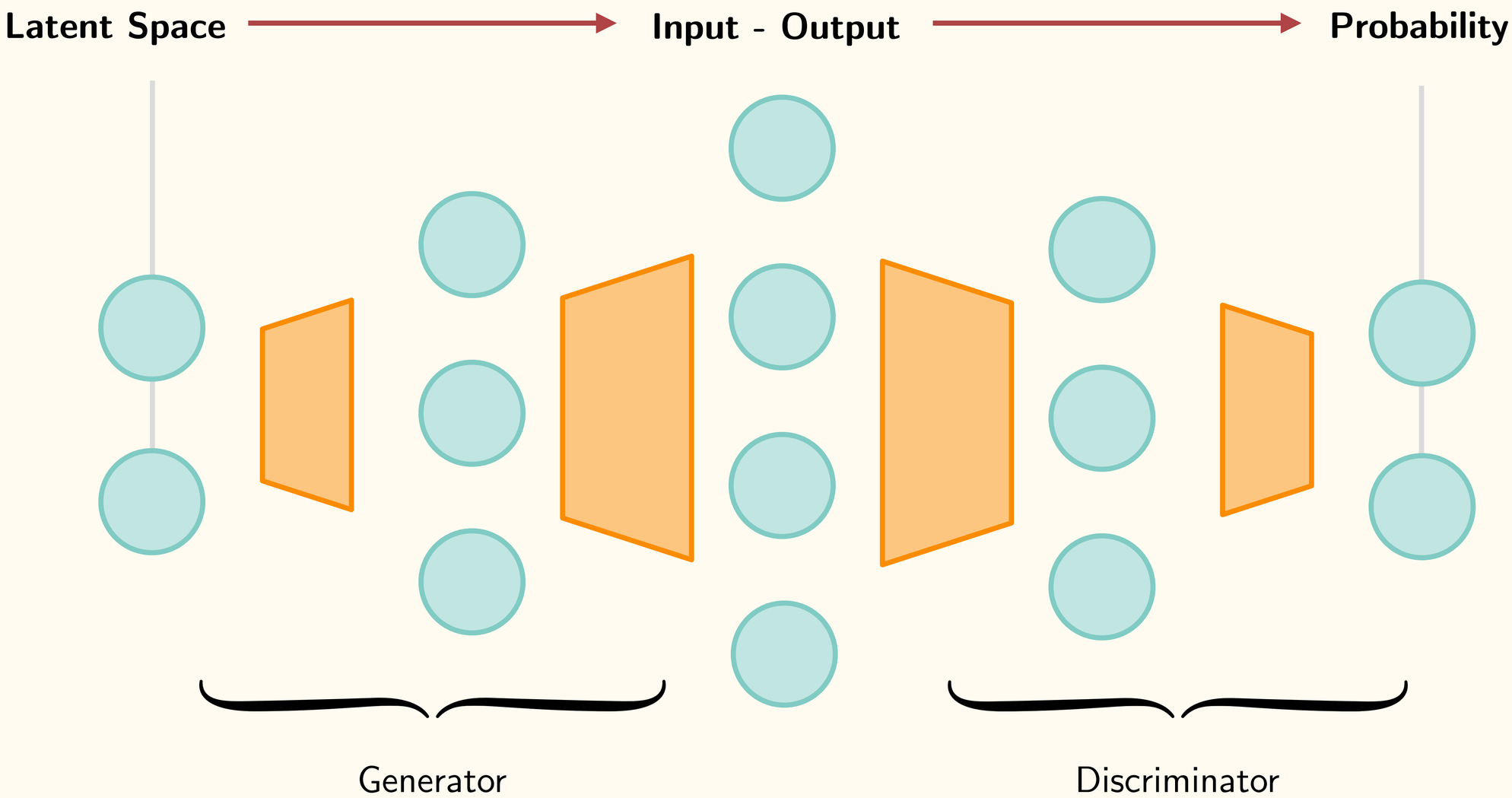
- How to make such a loss-function ?
  - You don't.

Generative Adversarial Network

- Essentially create a loss function to measure:
  - How well generated samples match the training samples
  - Use that loss function to optimise the model

- How to make such a loss-function ?
  - You don't.
  - GAN learns the loss function from the data.

Generative Adversarial Network



Generative Adversarial Network

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples
- **Discriminator**: tries to *distinguish* between

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples
- **Discriminator**: tries to *distinguish* between 

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples

- **Discriminator**: tries to *distinguish* between 
  - The generated samples from real training samples

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples

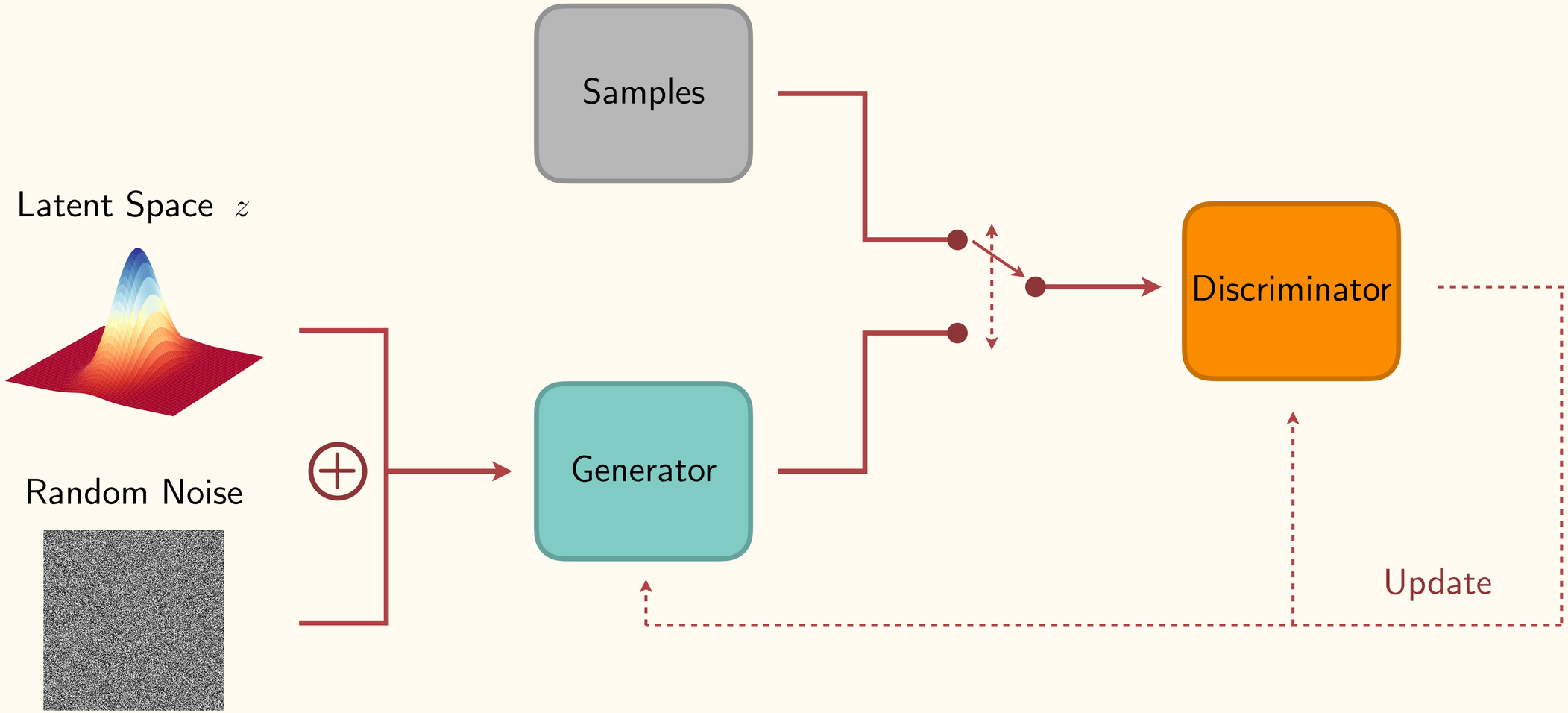
- **Discriminator**: tries to *distinguish* between 
  - The generated samples from real training samples
  - It takes a sample as input and outputs a probability of being a real sample

Generative Adversarial Network

- **Generator**: takes random vectors → generates synthetic samples

- **Discriminator**: tries to *distinguish* between 
  - The generated samples from real training samples
  - It takes a sample as input and outputs a **probability** of being a real sample
    - This acts as a loss function for the generator.

Generative Adversarial Network



Generative Adversarial Network

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

- Generator tries to ‘fool’ discriminator by presenting realistic samples

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

- Generator tries to 'fool' discriminator by presenting realistic samples
- Discriminator tries to get better at distinguishing real from fake samples

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

- Generator tries to 'fool' discriminator by presenting realistic samples
- Discriminator tries to get better at distinguishing real from fake samples

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

- Generator tries to 'fool' discriminator by presenting realistic samples
- Discriminator tries to get better at distinguishing real from fake samples

Generative *Adversarial* Network**s**

Generative Adversarial Network

- **Generator** and **discriminator** are run *simultaneously*.

- Generator tries to 'fool' discriminator by presenting realistic samples
- Discriminator tries to get better at distinguishing real from fake samples

Generative *Adversarial* Network**s**

GAN Applications

Colorize B&W images

- Even more advanced and realistic
- GAN based architecture

Jason Antic, 2018. *DeOldify: Colorizing and Restoring Old Images and Videos with Deep Learning*. FloydHub.

VAE vs GANs

VAE vs GANs

Roughly:

VAE vs GANs

Roughly:

VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples

VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples 

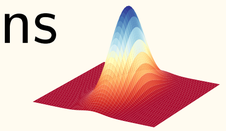
VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions

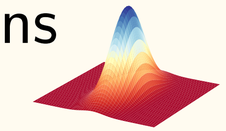
VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

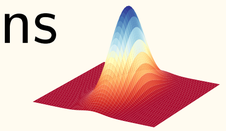
VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

VAE vs GANs

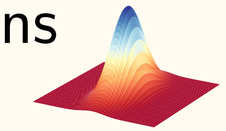
Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

- *Individual* samples generated by GANs more closely resemble training samples

VAE vs GANs

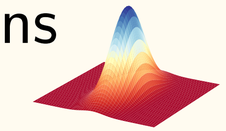
Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

- *Individual* samples generated by GANs more closely resemble training samples
- *Range* of samples generated by VAE more closely match range of training samples

VAE vs GANs

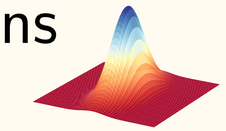
Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

- *Individual* samples generated by GANs more closely resemble training samples
- *Range* of samples generated by VAE more closely match range of training samples

VAE vs GANs

Roughly:

- GANs tend to produce higher-quality samples 
- VAEs tend to produce higher-quality distributions 

- *Individual* samples generated by GANs more closely resemble training samples
- *Range* of samples generated by VAE more closely match range of training samples

Very active field of research → Things changing constantly !

Application: Predicting Drug-Target Binding Affinity

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

- 97M compounds reported by the PubChem database

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

- 97M compounds reported by the PubChem database
- 12K drug entries reported by the DrugBank are considered

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

- 97M compounds reported by the PubChem database
- 12K drug entries reported by the DrugBank are considered

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

Application: Predicting Drug-Target Binding Affinity

- Challenging in Drug Discovery
- Supervised-learning based methods require labeled data on large scale
  - Expensive
  - Difficult

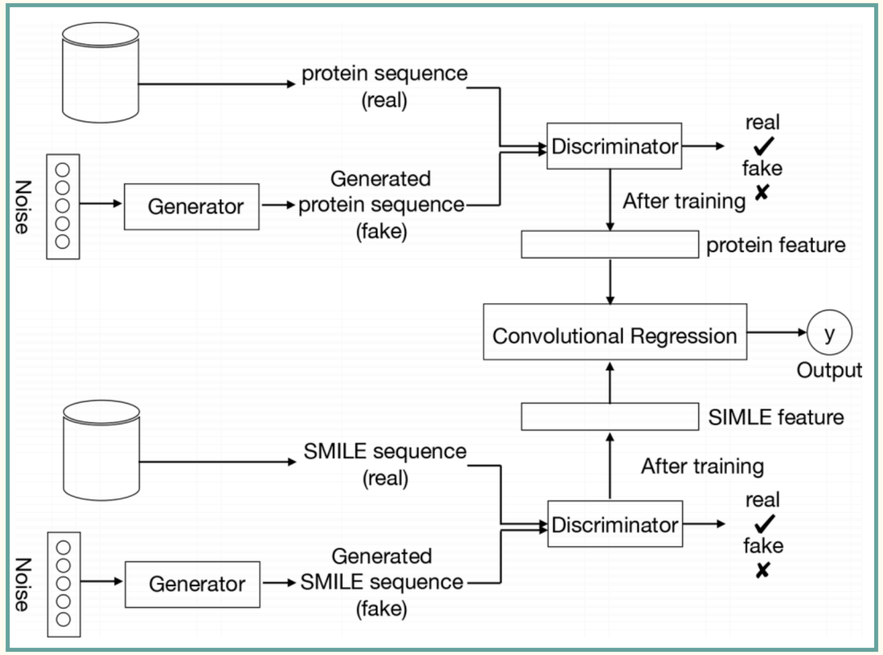
- 97M compounds reported by the PubChem database
- 12K drug entries reported by the DrugBank are considered

- Semi-supervised GAN based methods can help [1]

[1] Zhao Lingling et al., 2020. *GANsDTA: Predicting Drug-Target Binding Affinity Using GANs*. Frontiers in Genetics, Volume 10.

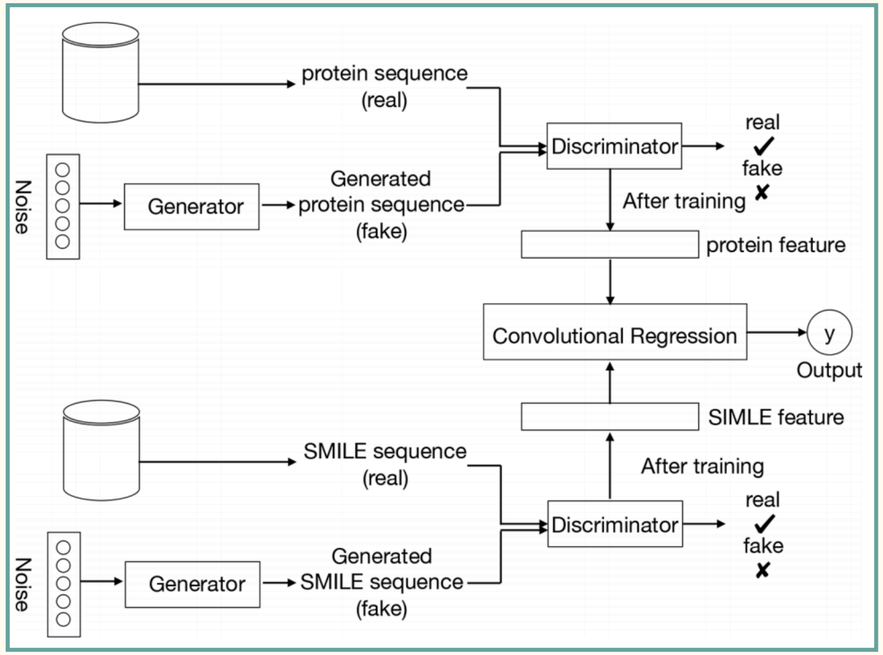
Application: Predicting Drug-Target Binding Affinity

Application: Predicting Drug-Target Binding Affinity



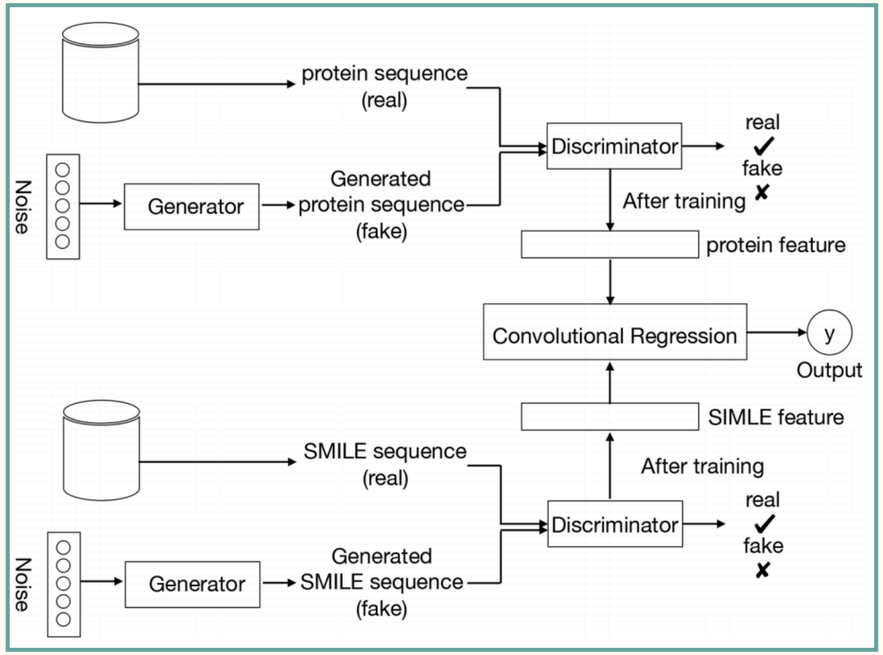
Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set



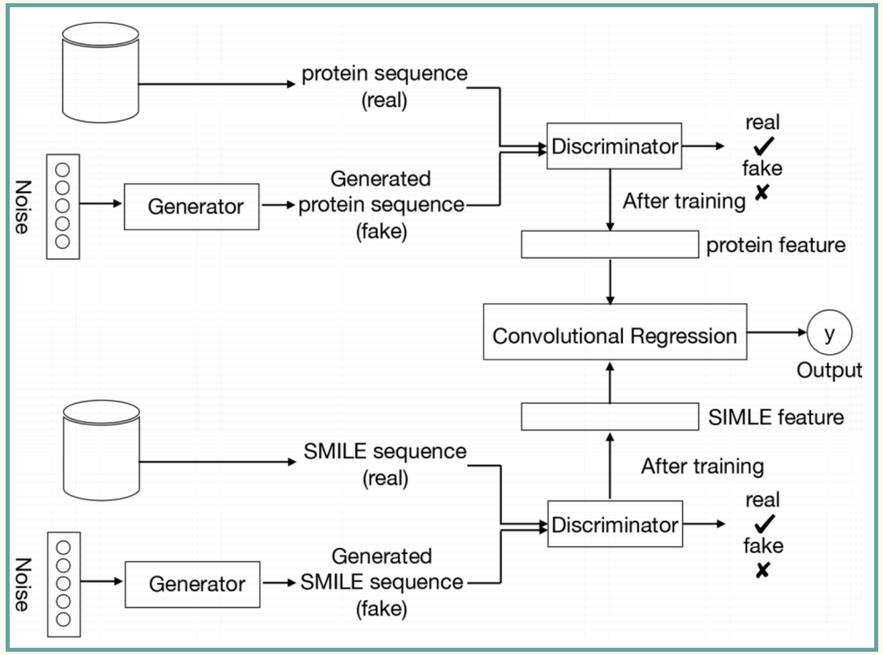
Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded



Application: Predicting Drug-Target Binding Affinity

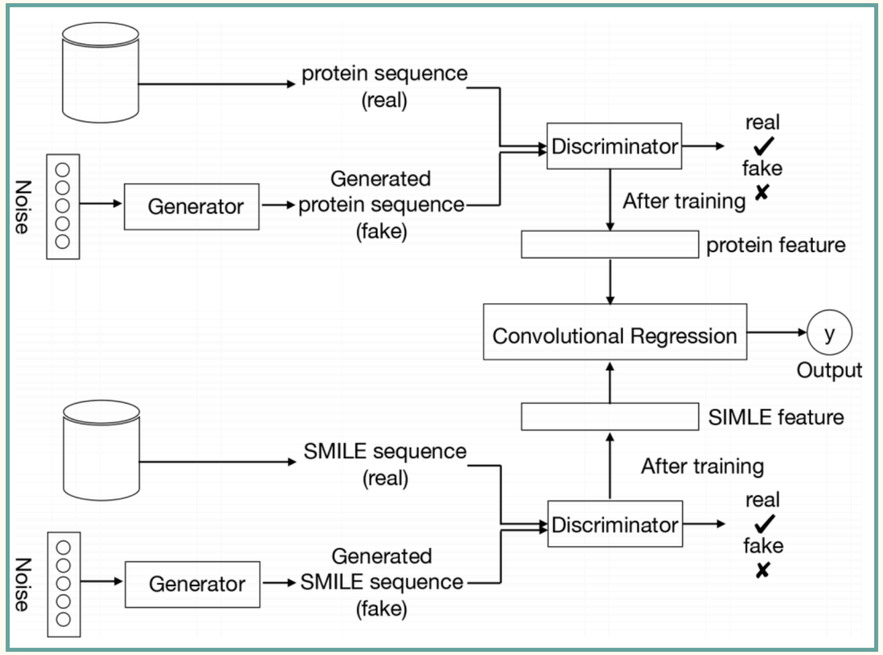
1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded



Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

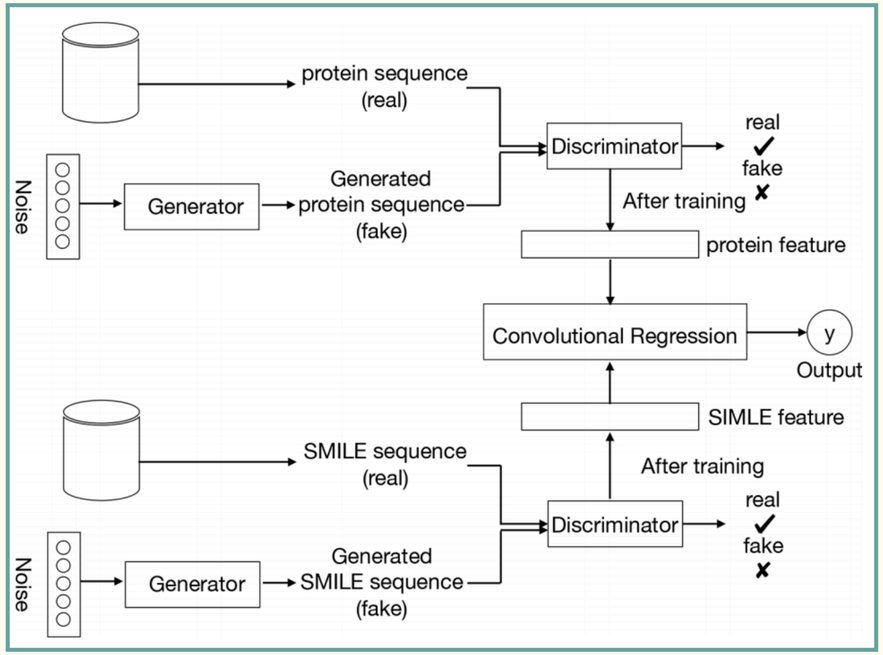
2. Two independent GANs are applied



Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

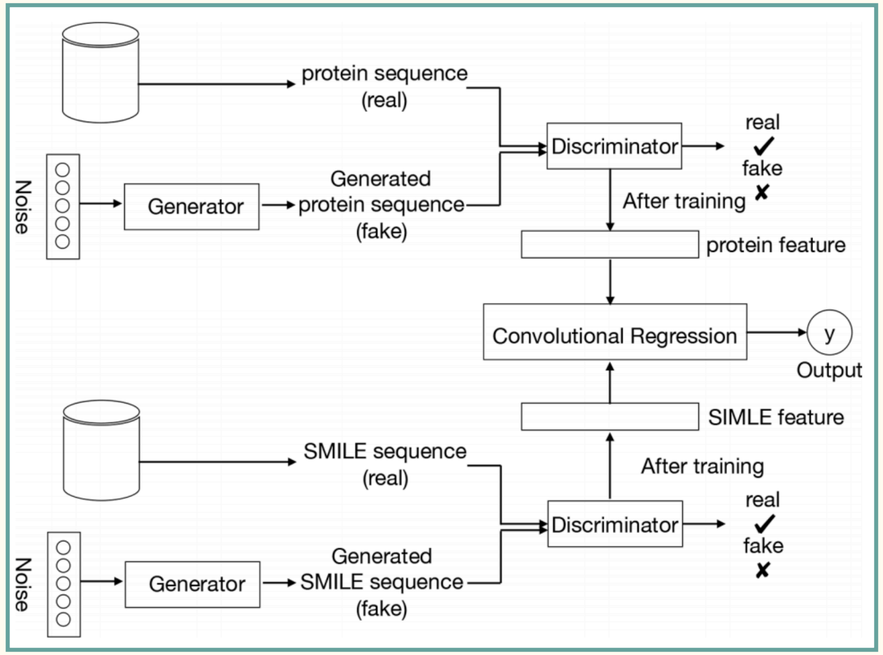
2. Two independent GANs are applied
   - Generate the fake samples



Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

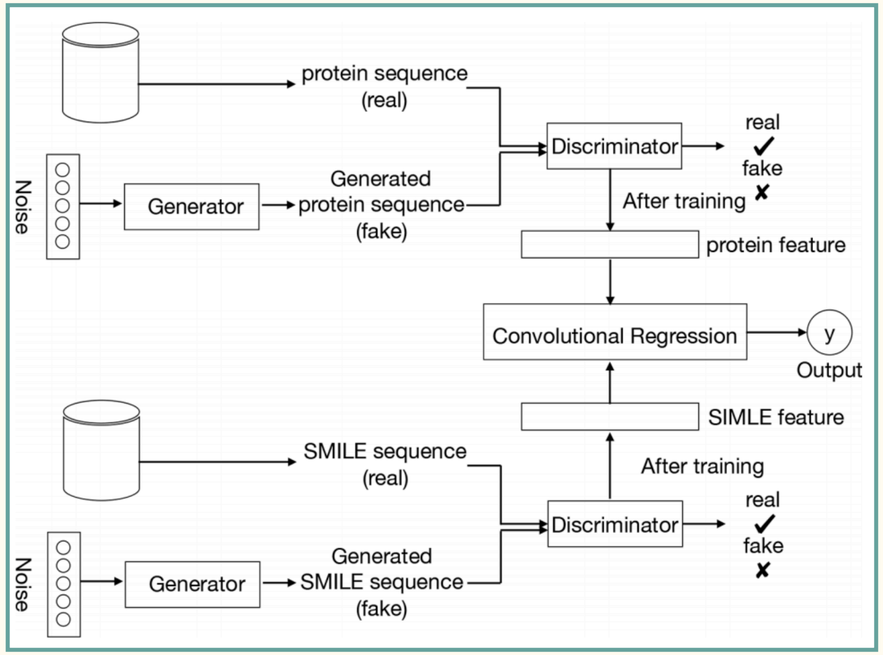


Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

3. Use GANs' trained discriminator

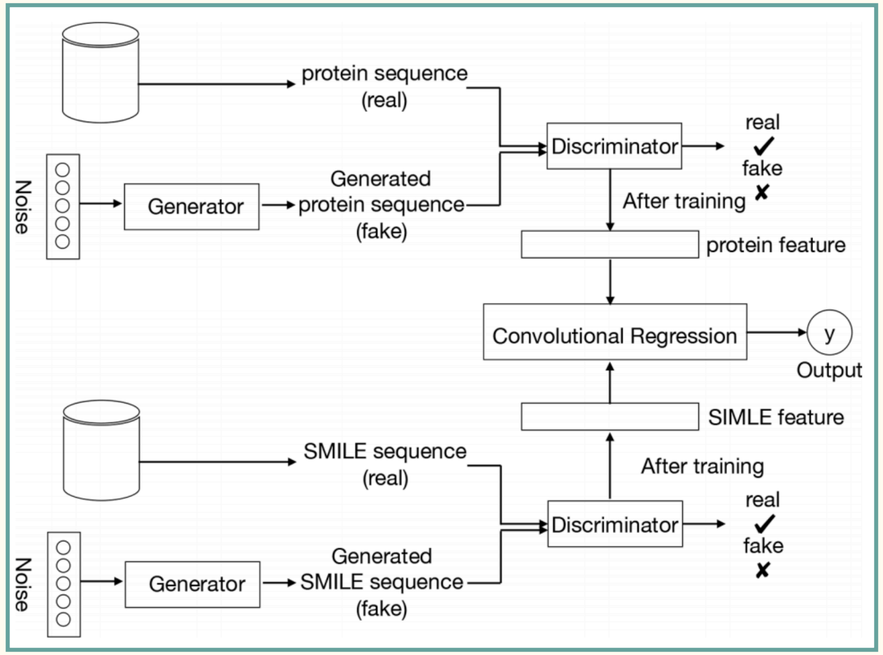


Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

3. Use GANs' trained discriminator
   - Project labeled datasets into a feature latent space

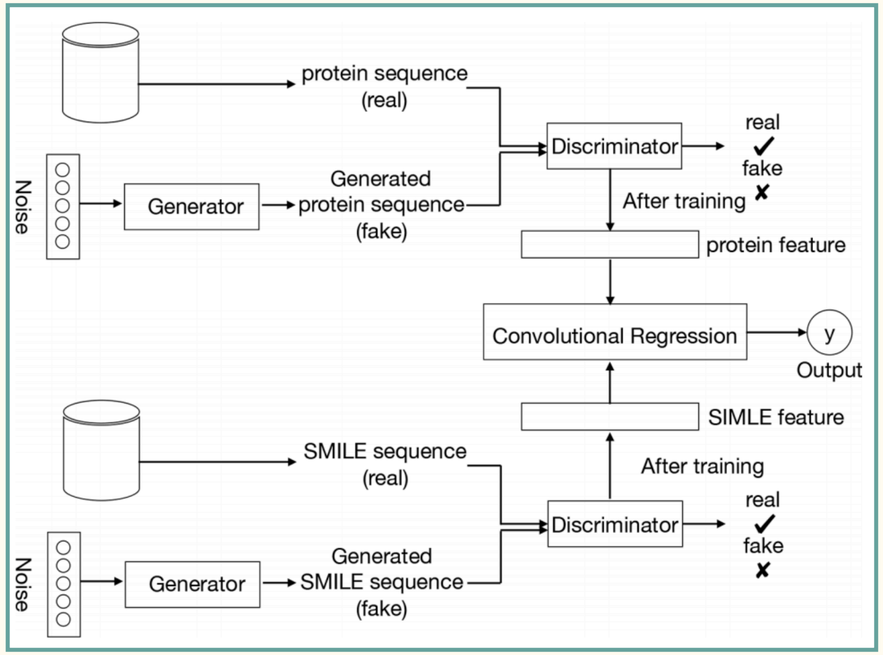


Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

3. Use GANs' trained discriminator
   - Project labeled datasets into a feature latent space



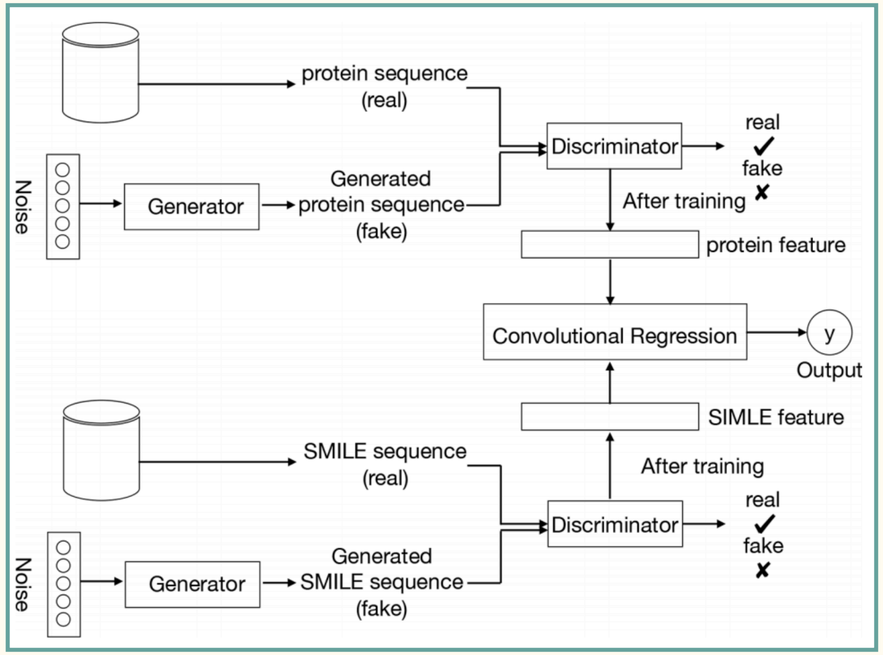
Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

3. Use GANs' trained discriminator
   - Project labeled datasets into a feature latent space

4. Train a convolutional regression based on this feature



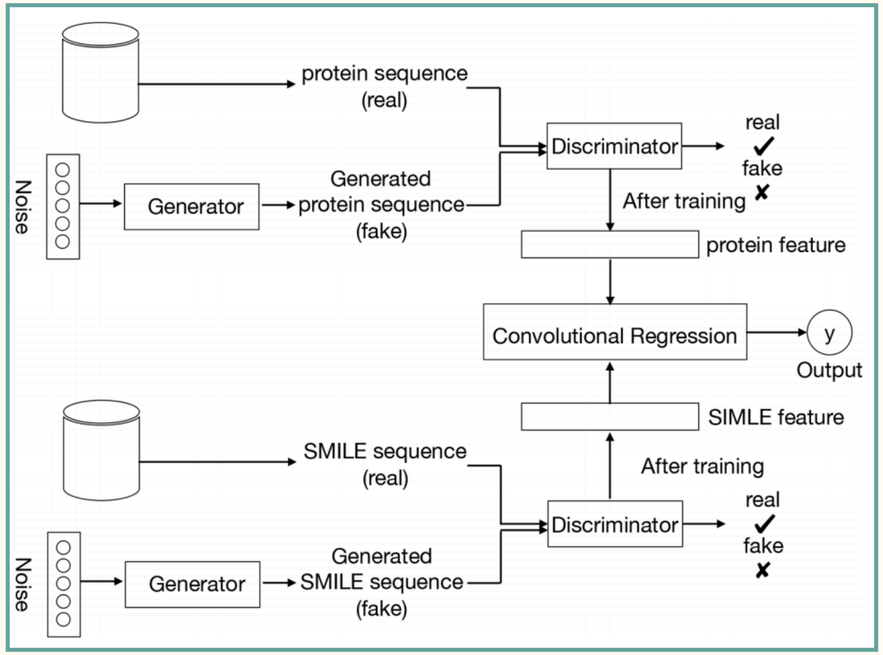
Application: Predicting Drug-Target Binding Affinity

1. Train the GANs on the unlabelled data set
   - Compound SMILES and protein sequences are encoded

2. Two independent GANs are applied
   - Generate the fake samples

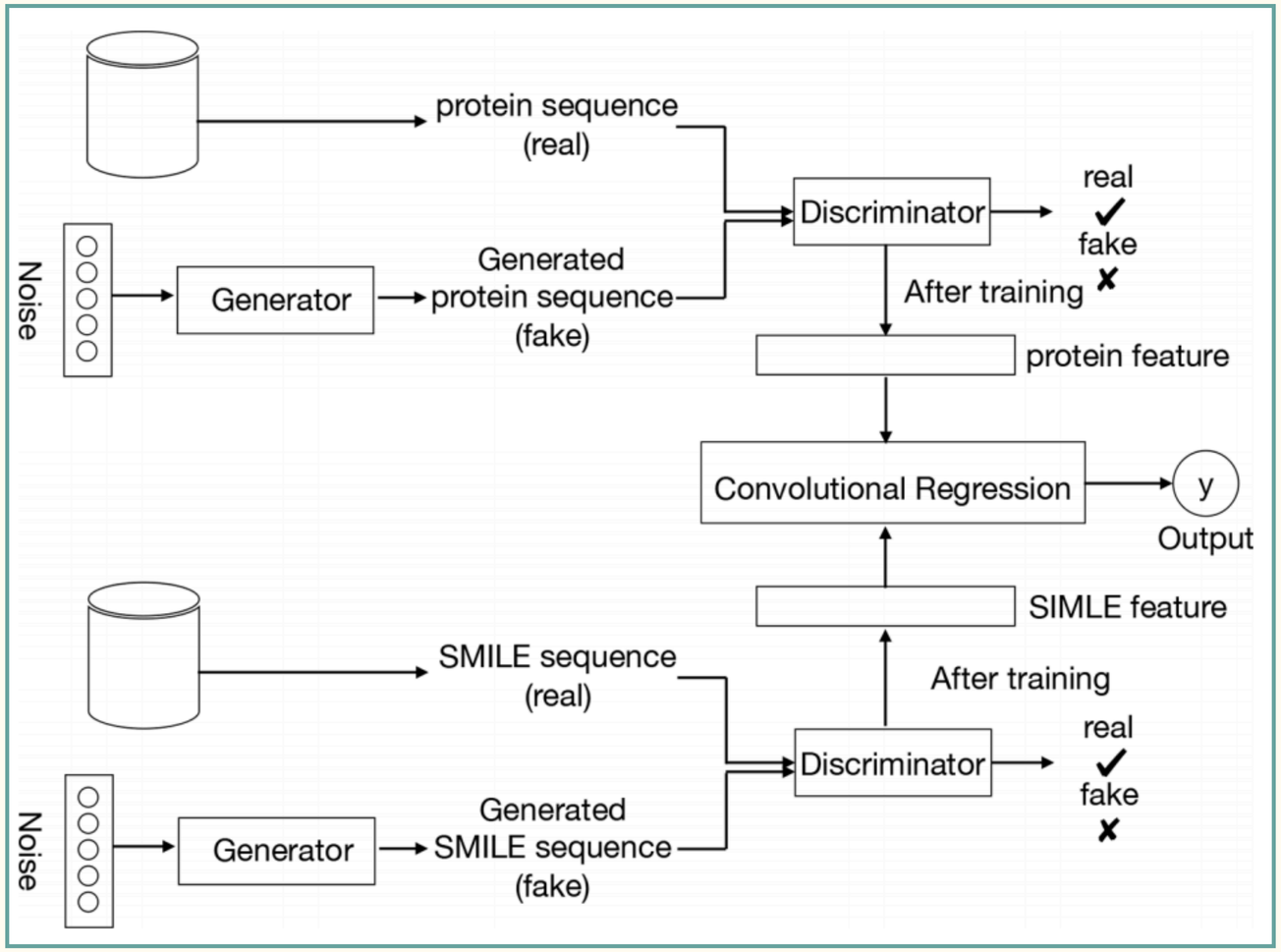
3. Use GANs' trained discriminator
   - Project labeled datasets into a feature latent space

4. Train a convolutional regression based on this feature
   - Predict the DT binding affinity



Application: Predicting Drug-Target Binding Affinity

Pipeline:



Future Work

Future Work

- Remember the progress !

Future Work

- Remember the progress !

2014

2015

2016

2017

2018

2019

Future Work

- Remember the progress !

2014

2015

2016

2017

2018

2019

Future Work

- Remember the progress !

- Next: generative videos


2014


2015


2016


2017


2018


2019

Future Work

- Remember the progress !

- Next: generative videos
  - High-res embryonic development ?


2014


2015


2016


2017


2018


2019

Future Work

- Remember the progress !

- Next: generative videos
  - High-res embryonic development ?
  - Deepfakes ?


2014


2015


2016


2017


2018


2019

Future Work

- Remember the progress !

- Next: generative videos
  - High-res embryonic development ?
  - Deepfakes ?


2014


2015


2016


2017


2018


2019

Future Work

- Remember the progress !

- Next: generative videos
  - High-res embryonic development ?
  - Deepfakes ?

- Evolving quickly


2014


2015


2016


2017


2018


2019

Thank you !

aCubeIT

Ackruti Trade Center, MIDC, Mumbai, IND

www.acubeit.com

+91 9892512129

prakash@acubeit.com