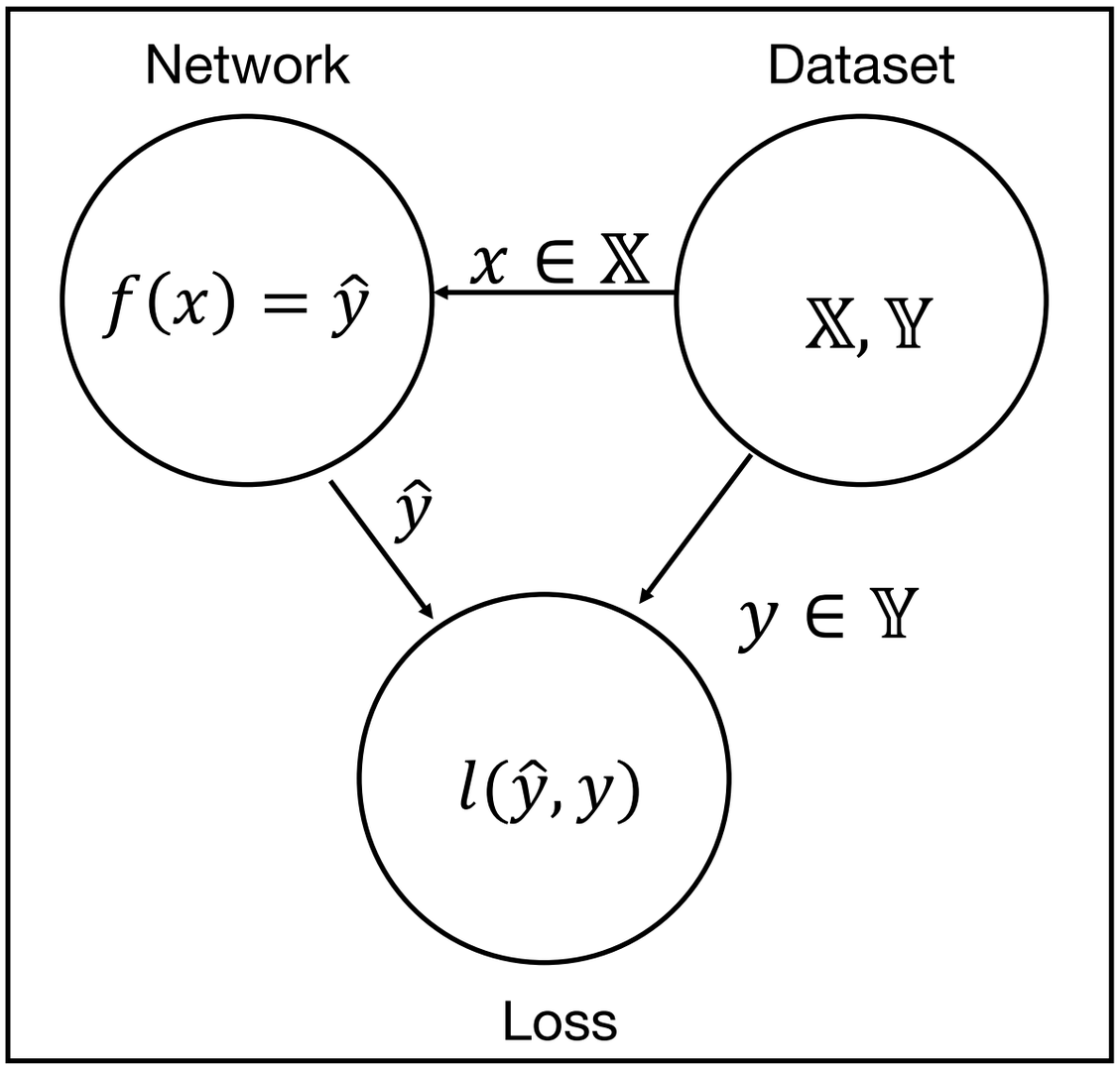
Deep learning

Unsupervised learning, VAE, GAN

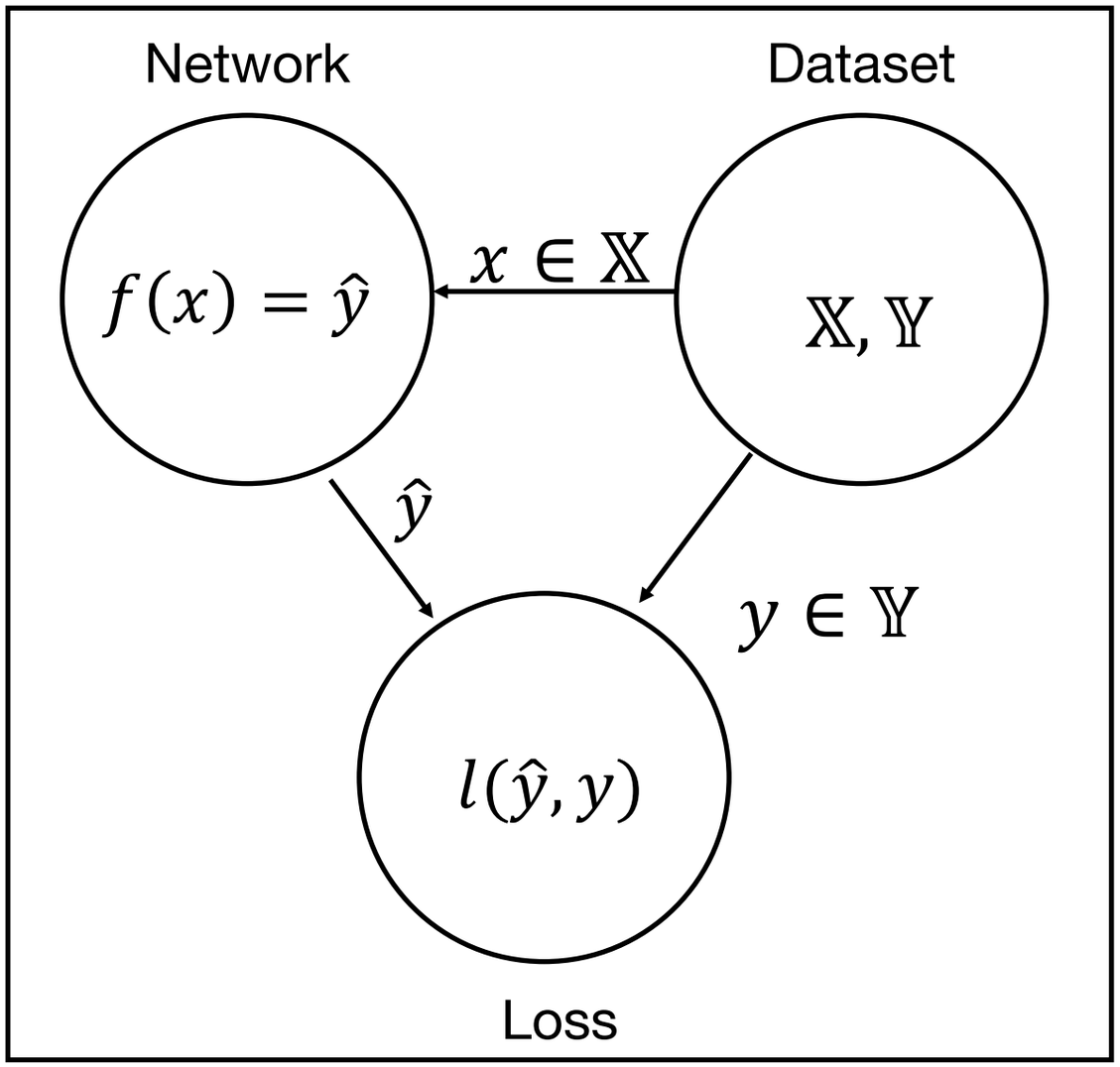
Supervised learning

- The task of learning a function given input-output pairs.
- Our dataset needs to be painstakingly **labeled by humans**.
- The **loss function** drives the learning.
  - **Regression**
  - **Classification**
  - ...



Unsupervised learning

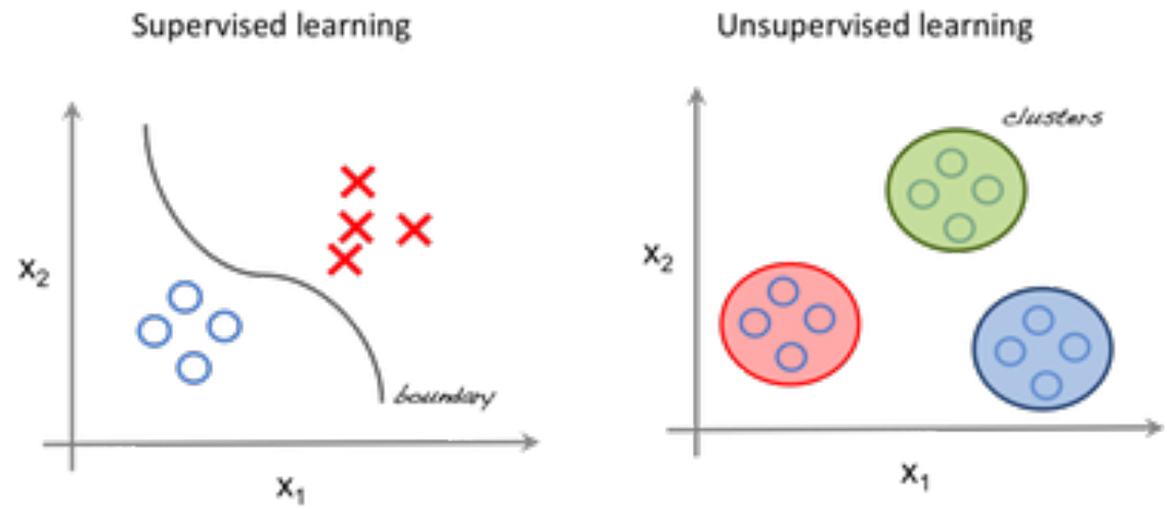
- Can we avoid labelling the data?
  - It can be costly.
    - Children do not need every example to be labelled.
- How then do we **evaluate** the output of the network?
- What is our **output**?



Unsupervised learning

Tasks

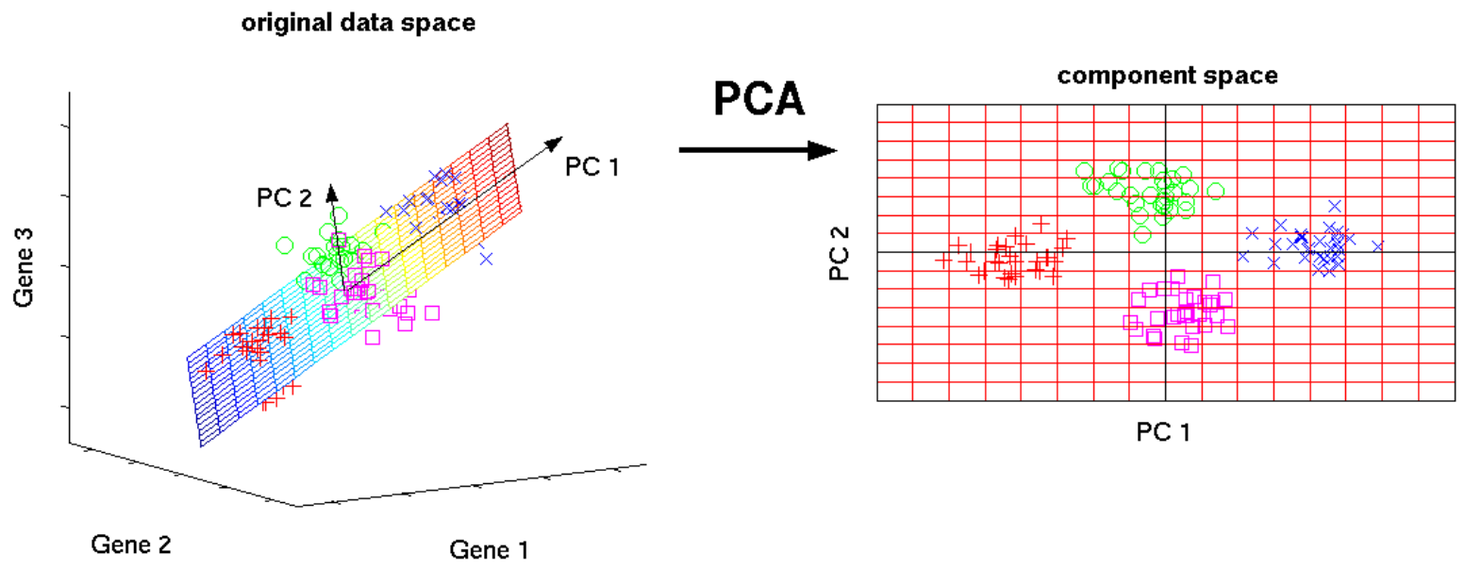
- We try to learn relationships between the data.
  - **Clustering / Probability density estimation**
  - **Dimensionality reduction**
  - ...
- We tend to do this by estimating the probability distribution of the data.



Unsupervised learning

ML algorithms

- There are many unsupervised learning ML algorithms.
  - Principal component analysis (**PCA**)
  - **k-means** clustering

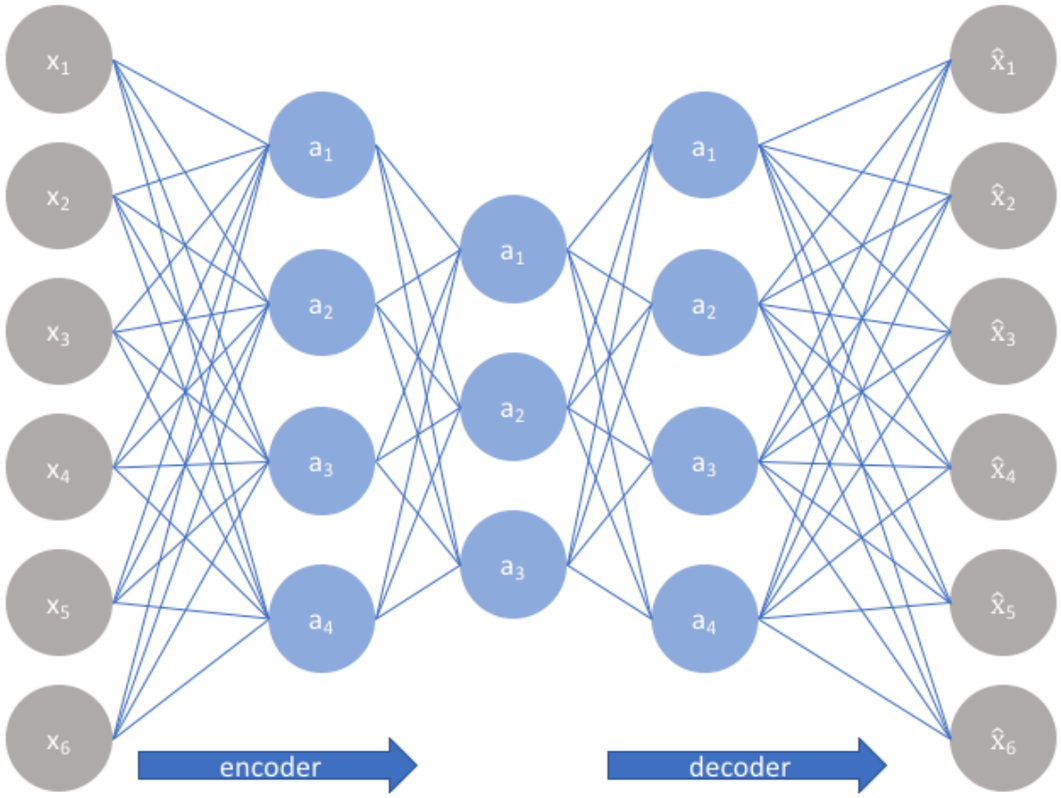


Source: http://www.nlpca.org/pca_principal_component_analysis.html

Unsupervised learning

DL models

- DL models include
- **Autoencoders** (Hinton, 2006, seminal paper)
  - Dimensionality reduction
  - Anomaly detection
  - Feature extractors
- Generative Adversarial Networks (**GANs**) (Ian Goodfellow et al., 2014)
  - Probability Density estimation
  - Generative model



Autoencoder

Source: https://en.wikipedia.org/wiki/Autoencoder

Autoencoder

- Autoencoder f attempts to learn the identity function

$$f(x) = x$$

  - Given input x the autoencoder outputs $\hat{x}$ which should be as close to x as possible

- Consists of an **encoder** and **decoder** pair.

$$f(x) = d(e(x))$$

- The output z of the encoder has lower dimension than x

  - The autoencoder compresses the input to a smaller representation.

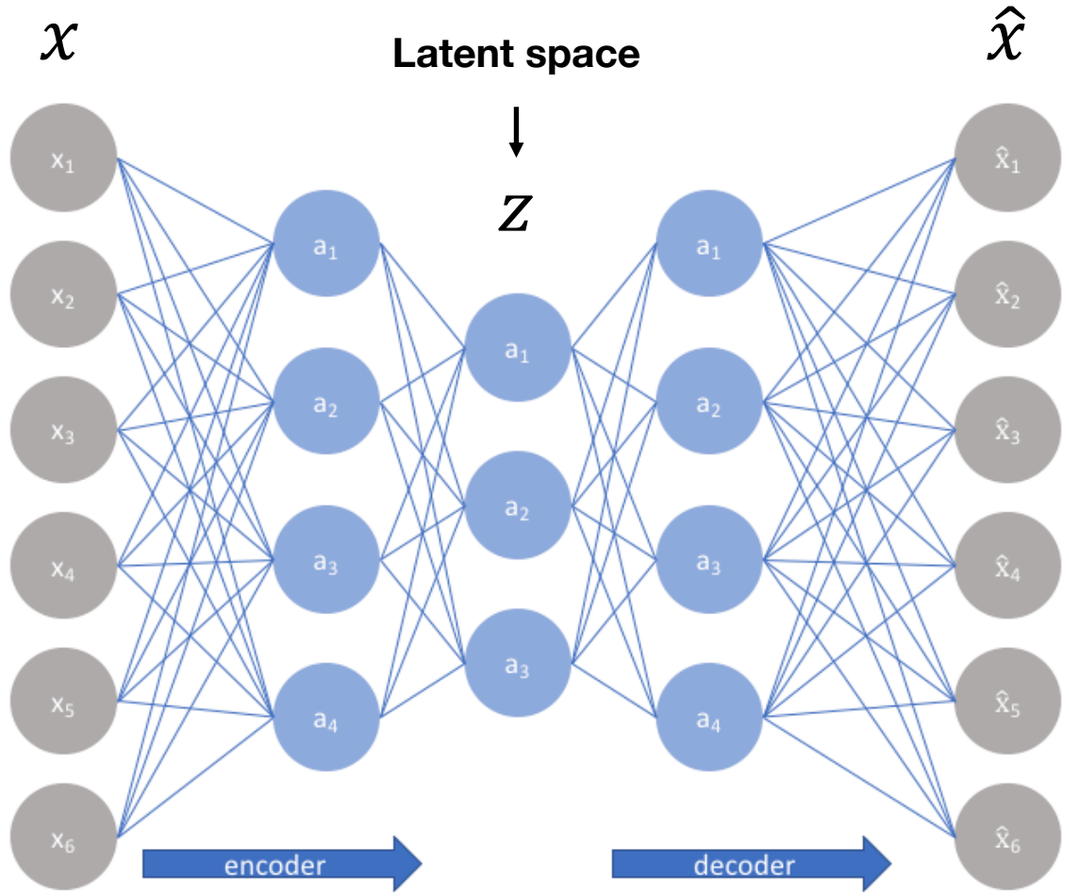
- Autoencoders reconstruct the input after going through this information bottleneck.

Encoder

$$e(x) = z$$

Decoder

$$d(z) = \hat{\mathrm{x}}$$

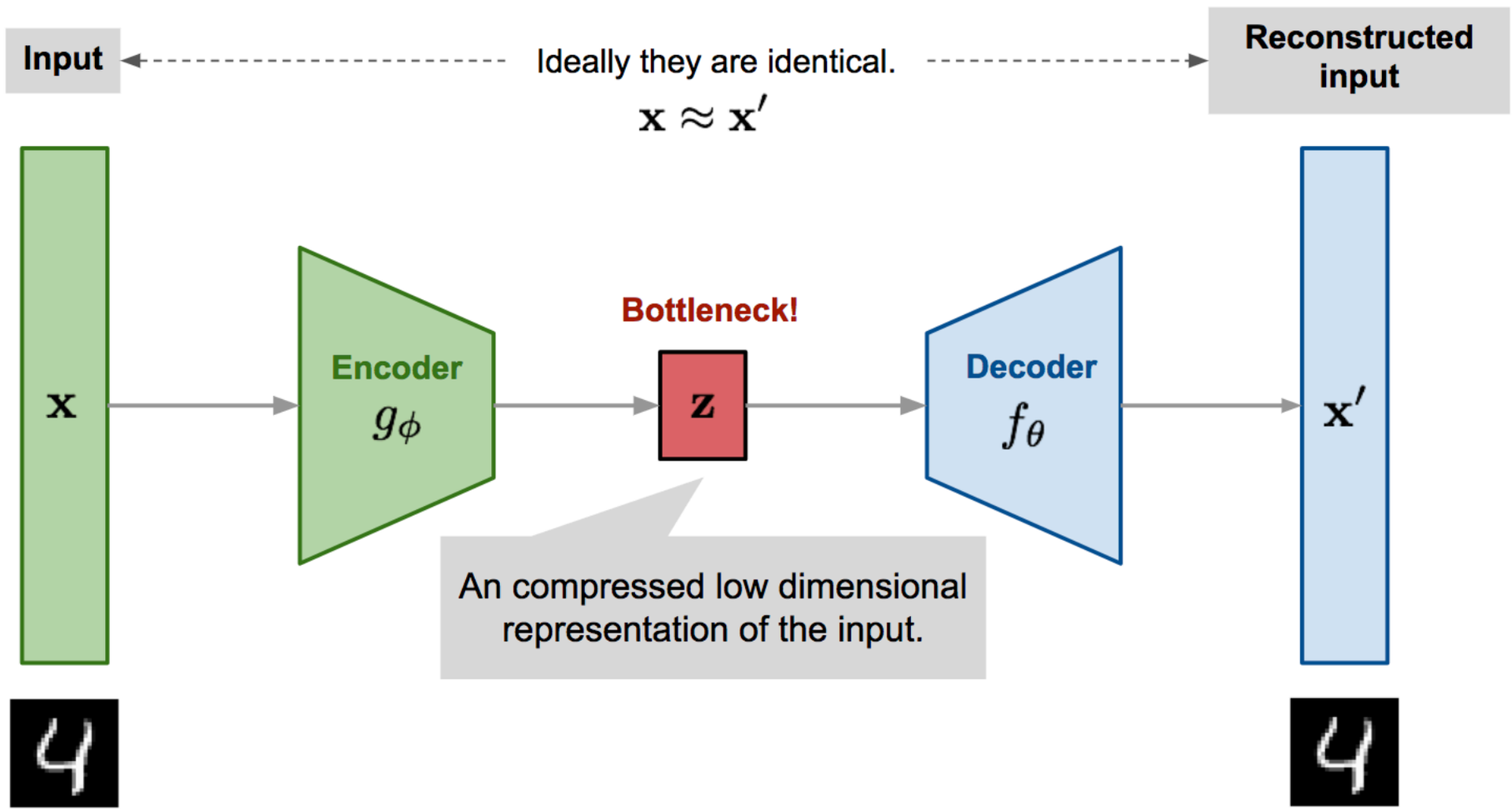


Source: https://en.wikipedia.org/wiki/Autoencoder

Autoencoder

Compression

- We get a compressed representation/encoding of the input

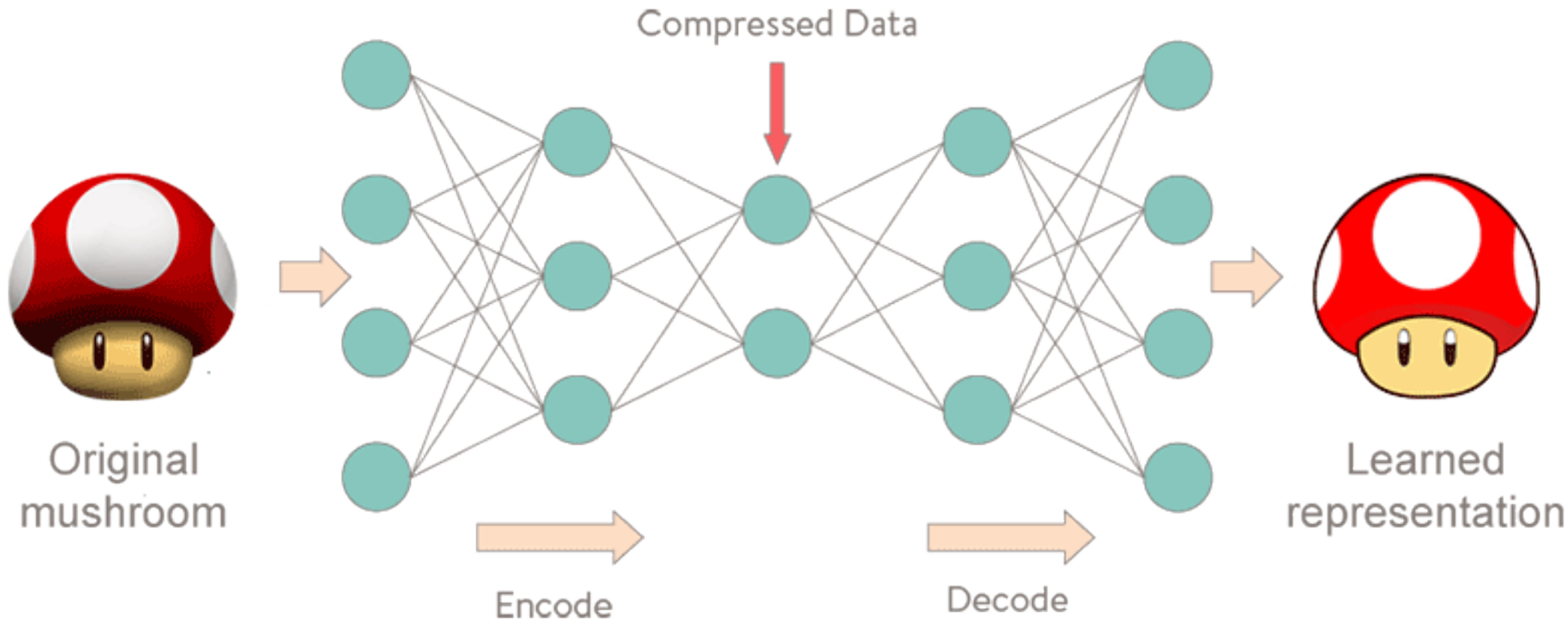


Source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Autoencoder

Compression

- A lossy compression.

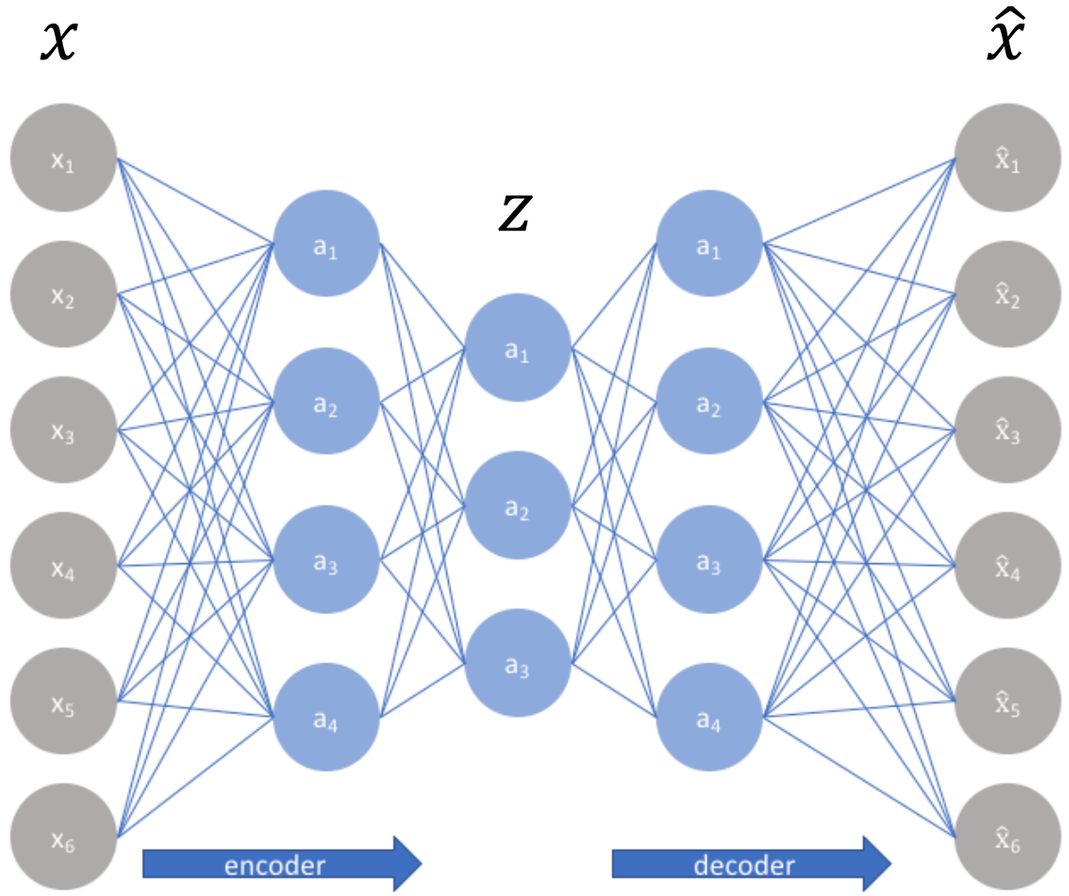


Source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Autoencoder

Applications

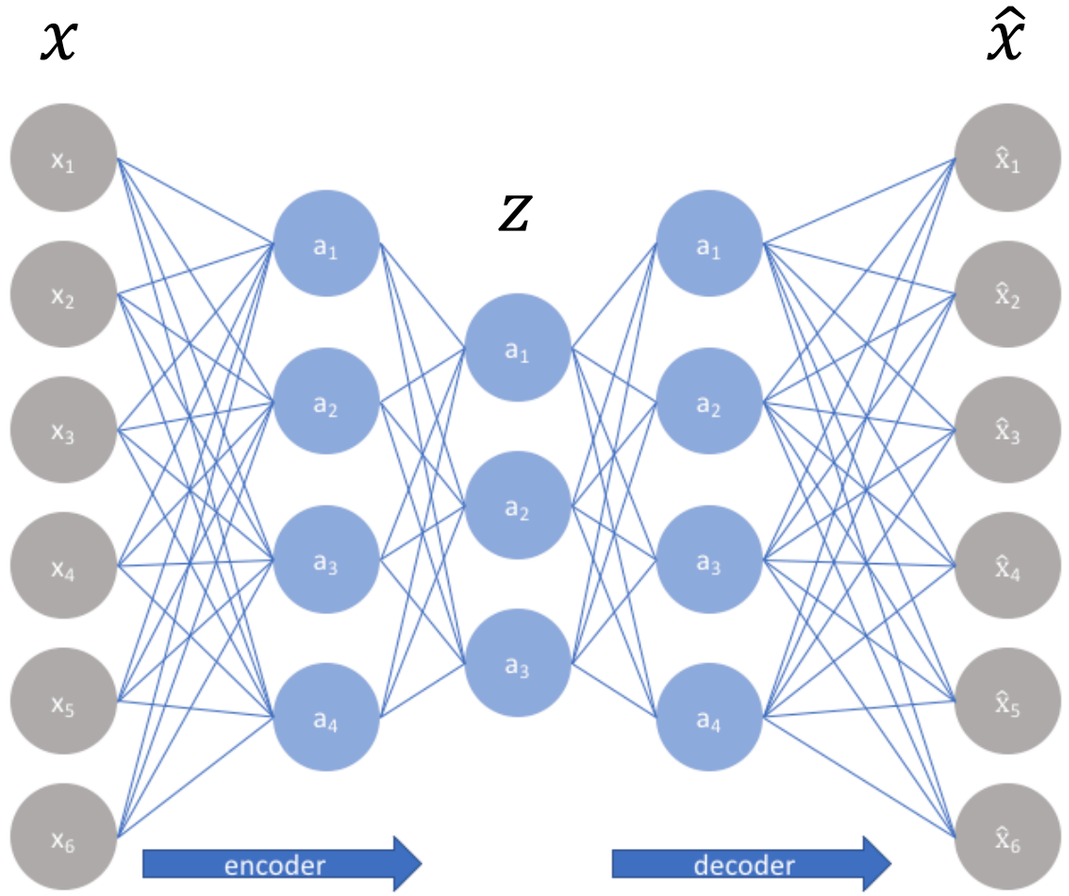
- Why is this useful?
  - Dimensionality reduction / Feature extraction.
  - Can be more expressive than PCA
    - PCA is linear
    - Autoencoder has multiple layers; non-linear activations
- Apply "classical" algorithms on the compressed representation (z), e.g. k-means.



Source: https://en.wikipedia.org/wiki/Autoencoder

Autoencoder
Applications

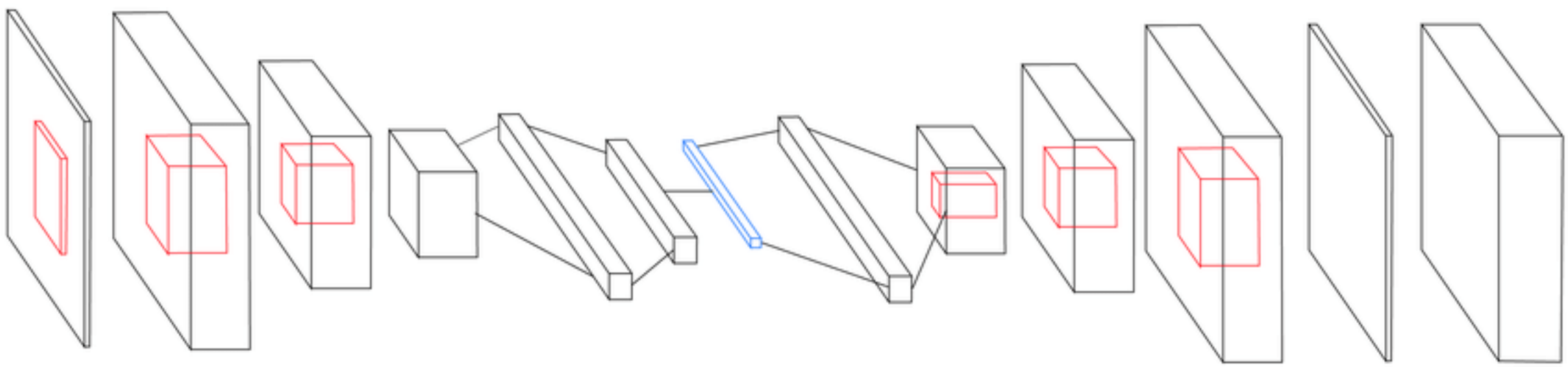
- Why is this useful?
  - Anomaly detection
  - We have learnt to represent common features compactly
  - Anomalies have different features
  - High reconstruction error = anomaly.



Source: https://en.wikipedia.org/wiki/Autoencoder

Autoencoder
Implementations

- The encoder and decoder can be.
  - Fully connected (dense) with many layers
  - Convolutional.
  - ...



Source: Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning (2016, Tang et al.)

Autoencoder
Implementations

- Code example: https://blog.keras.io/building-autoencoders-in-keras.html

Autoencoder

Denoising autoencoder (2008, Vincent et al.)

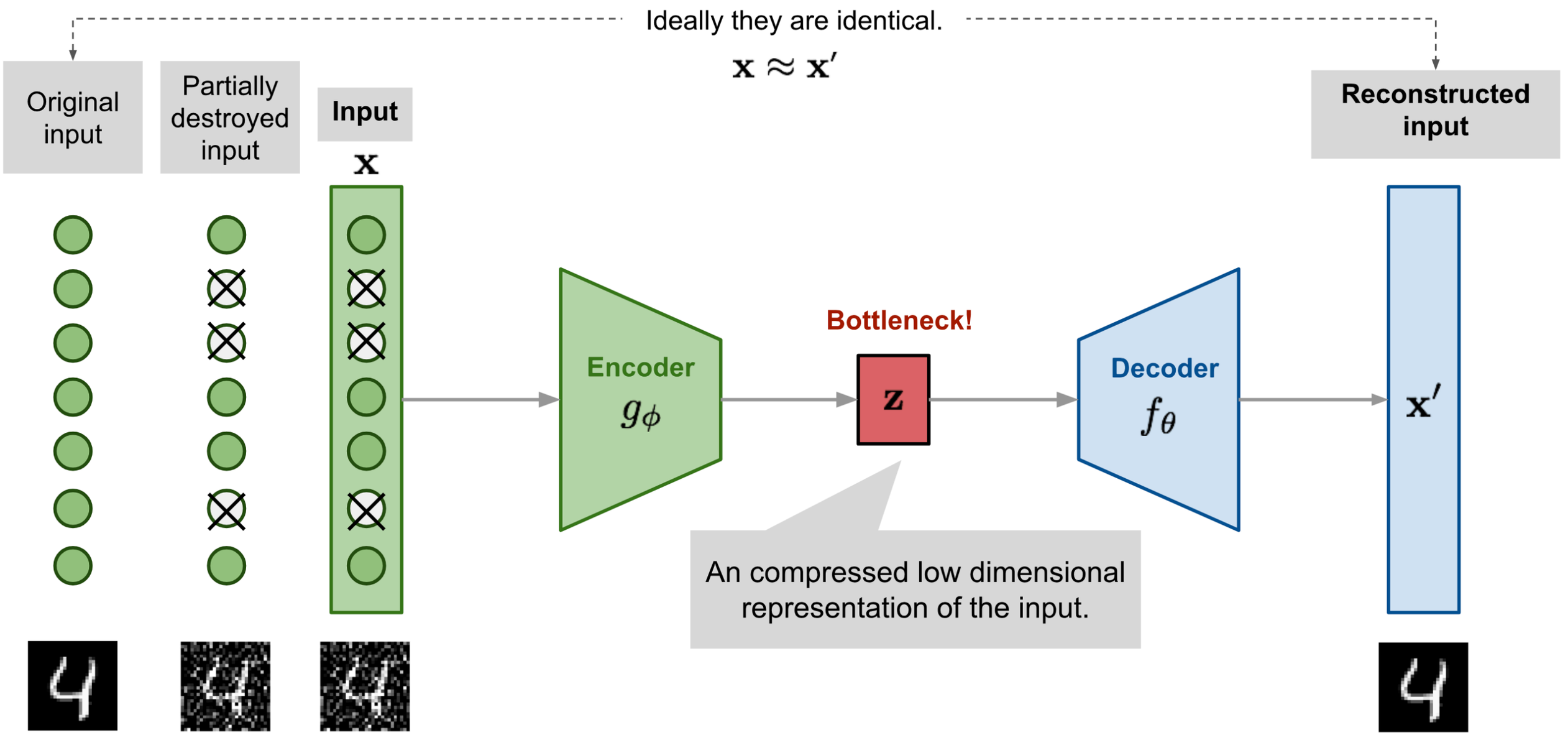
- Instead of feeding the input x we feed a corrupted input $\tilde{x}$, e.g. by adding random noise to x.

- We want the autoencoder to remove the corruption from the input.

- The loss is based on the output of the model and the uncorrupted image.

Autoencoder

Denoising autoencoder



Source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Autoencoder

Denoising autoencoder

- Why is this useful?
  - We learn to detect more robust features.
  - Precursor to dropout (2012, Hinton et al.)

Source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Autoencoder

other autoencoders

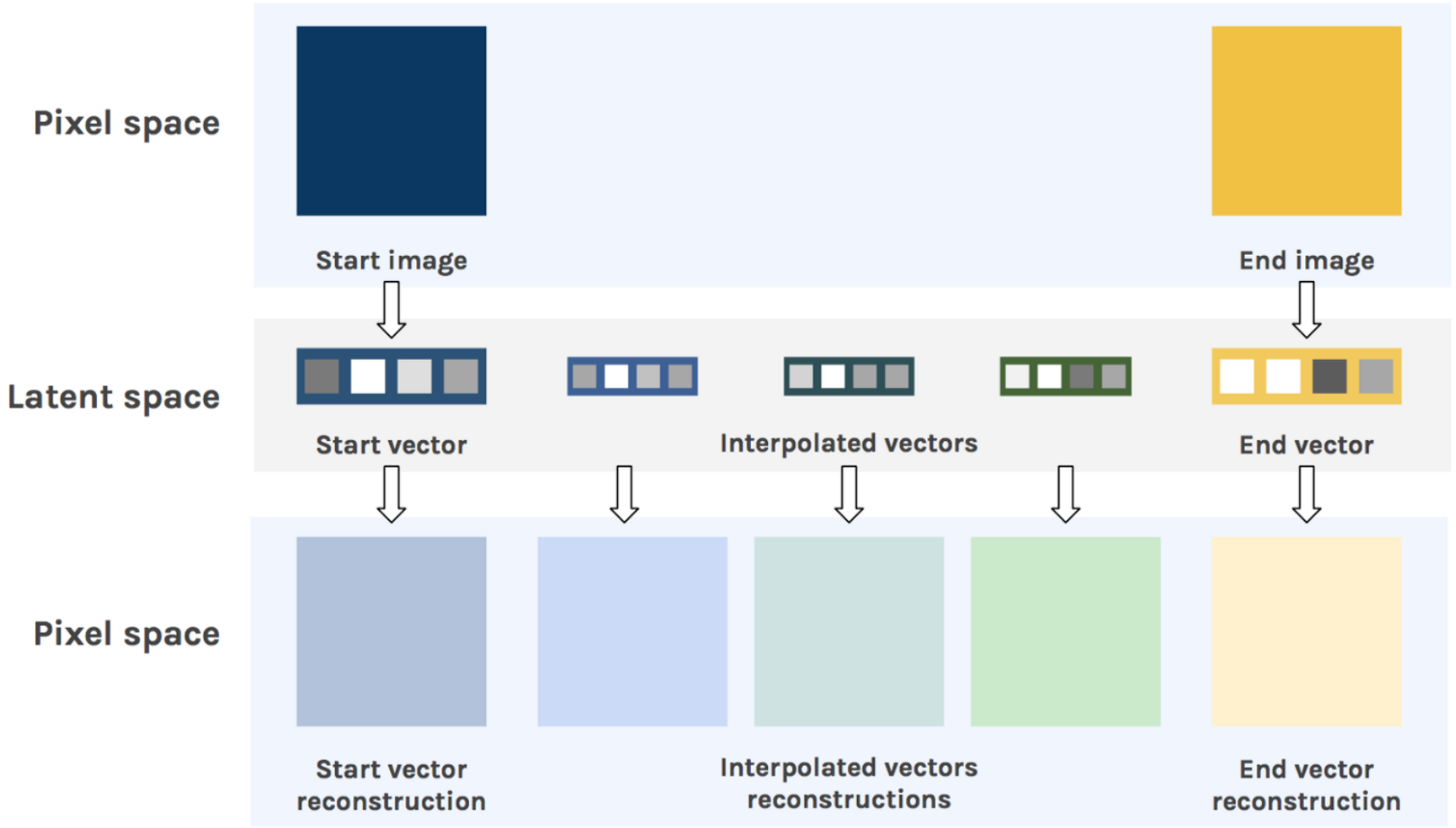
- There are more variations of autoencoders which apply different constraints on the learnt representation, the **latent space**.
- These constraints are applied by **adding terms to the loss function** (like regularisation).
- They are sometimes called regularised autoencoders.
- Take-home message, to apply different constraints, add terms to your loss function. This is very common.

Autoencoder

Interpolation

- Can we get novel images by interpolation in the latent space?

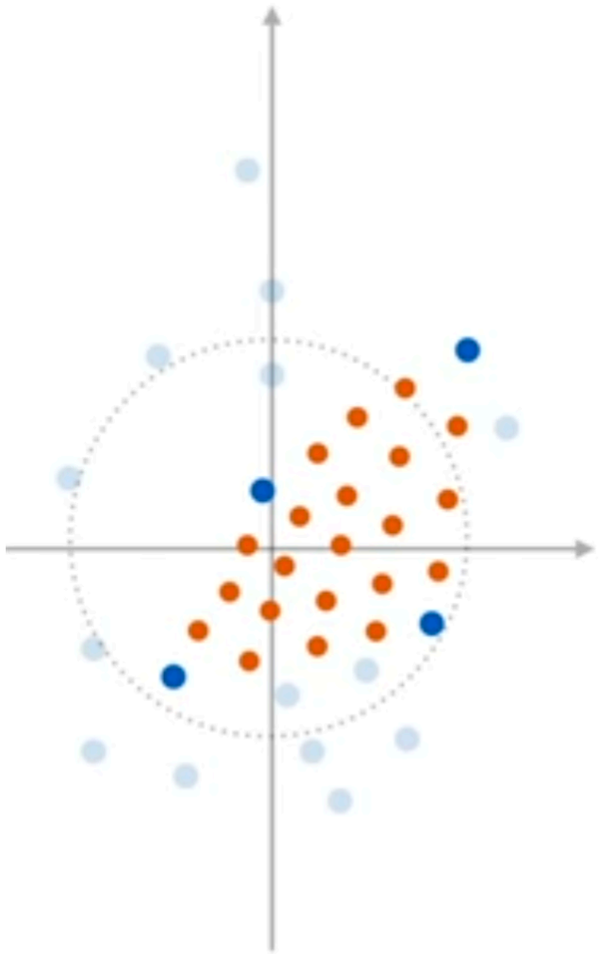
- Interpolation is the process of creating new datapoints between existing datapoints.



Source: https://hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df

Autoencoder

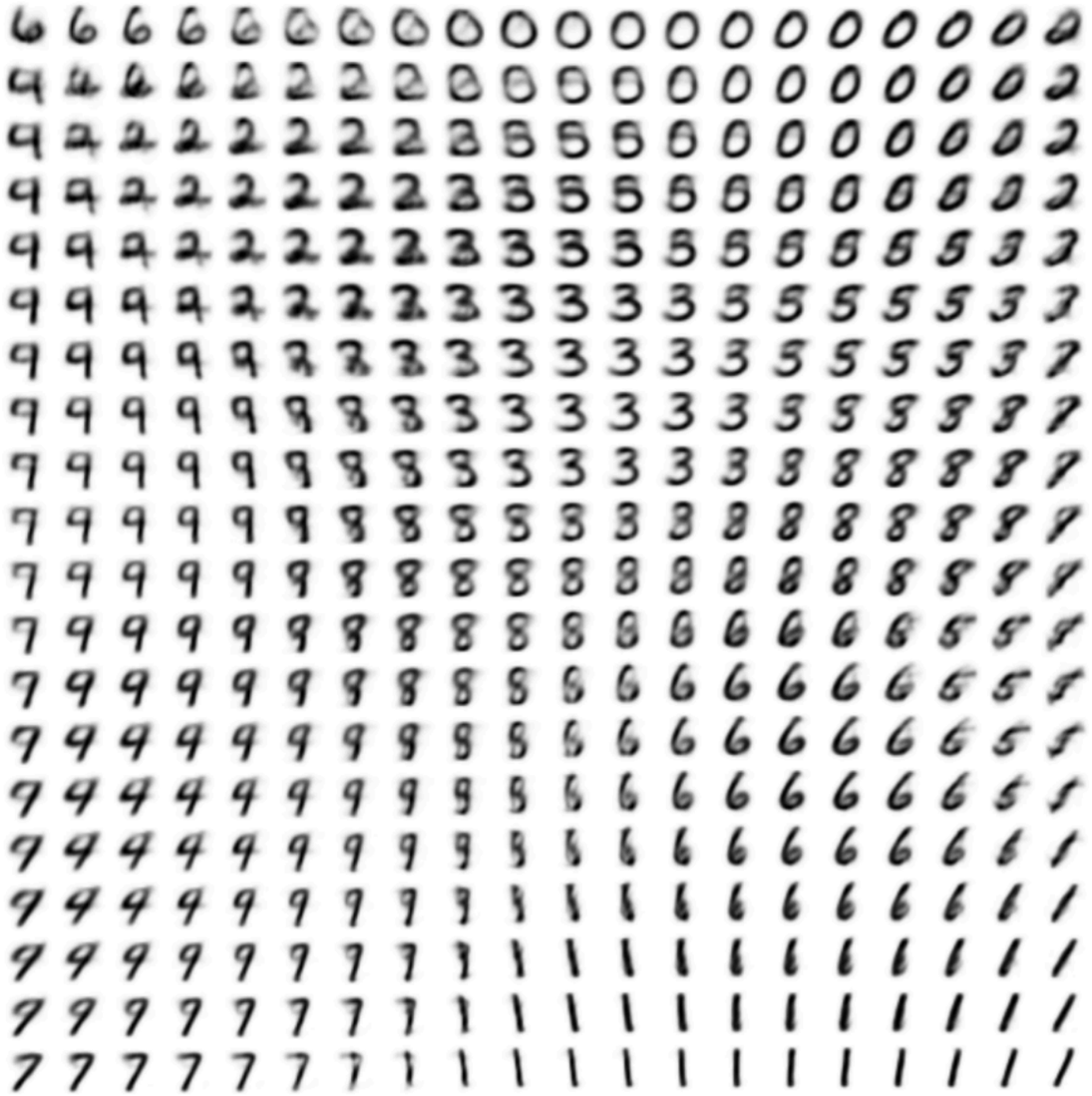
Interpolation



Source: Peter Bloem, VU ML course 2019.

Autoencoder

Interpolation



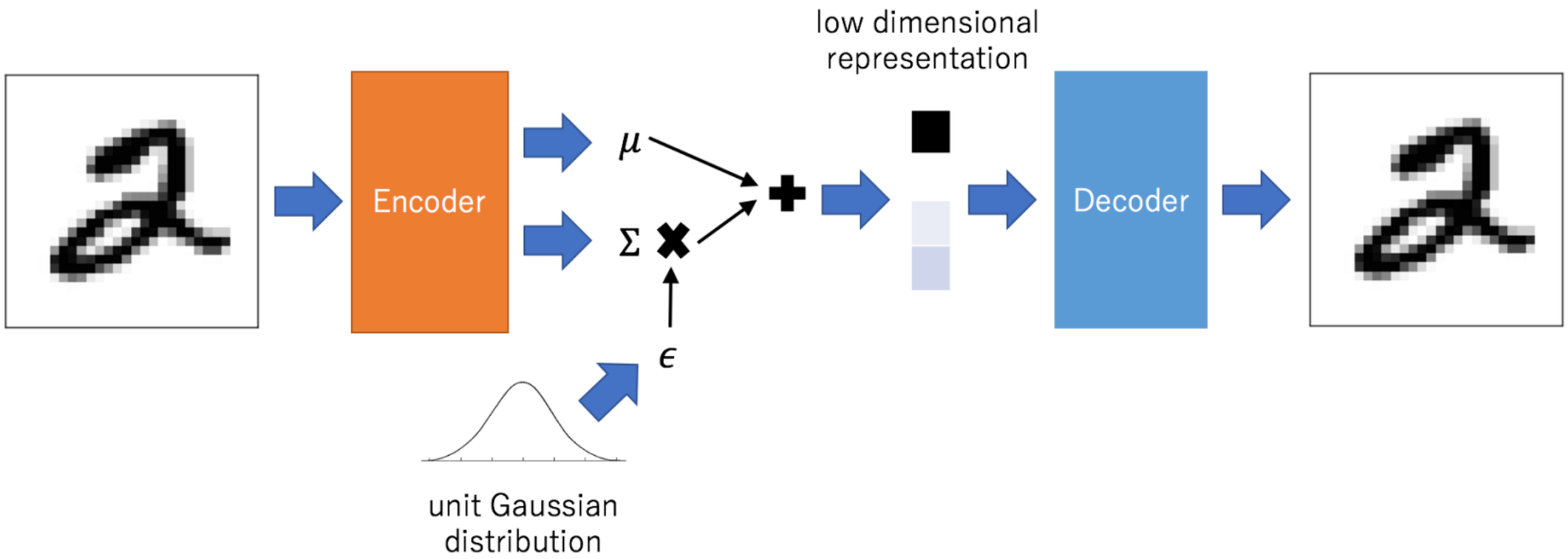
Source: https://ermongroup.github.io/cs228-notes/extras/vae/

Variational Autoencoder

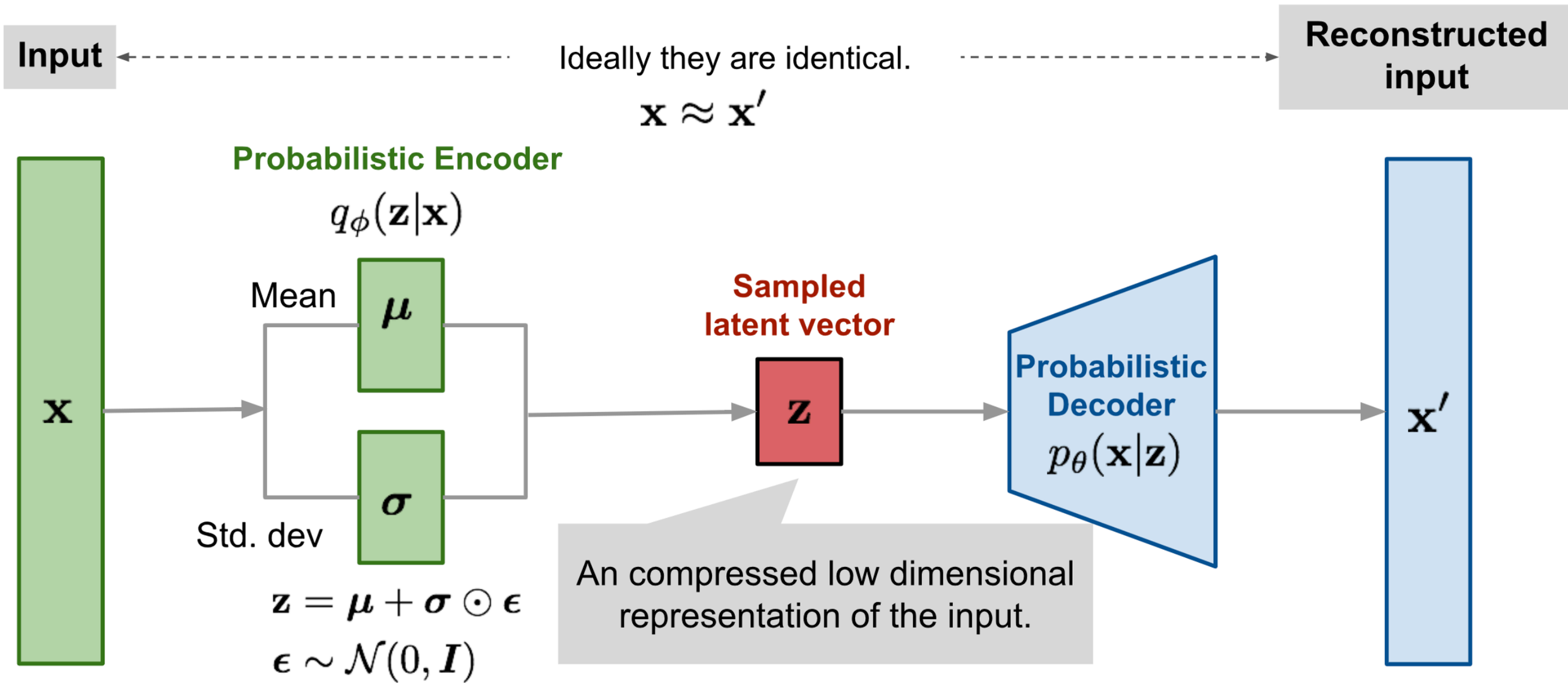
- The latent space in normal autoencoders does not allow us to do this well.
- The space is "not smooth" so the output is very distorted.
    - When we decode some latent representations we get garbage out.
- The variational autoencoder (VAE) (2014, Kingma & Welling) allows us to do this by adding some randomness to our latent space.
- By making the decoder deal with some randomness, the latent space becomes more smooth.

Variational Autoencoder

- The encoder outputs the parameters (mean and standard deviation) for a multinomial (many dimensions) normal distribution.
- We then **sample** a vector as the representation for the input.
- We have just added some random elements to our model.



Variational Autoencoder



Variational Autoencoder

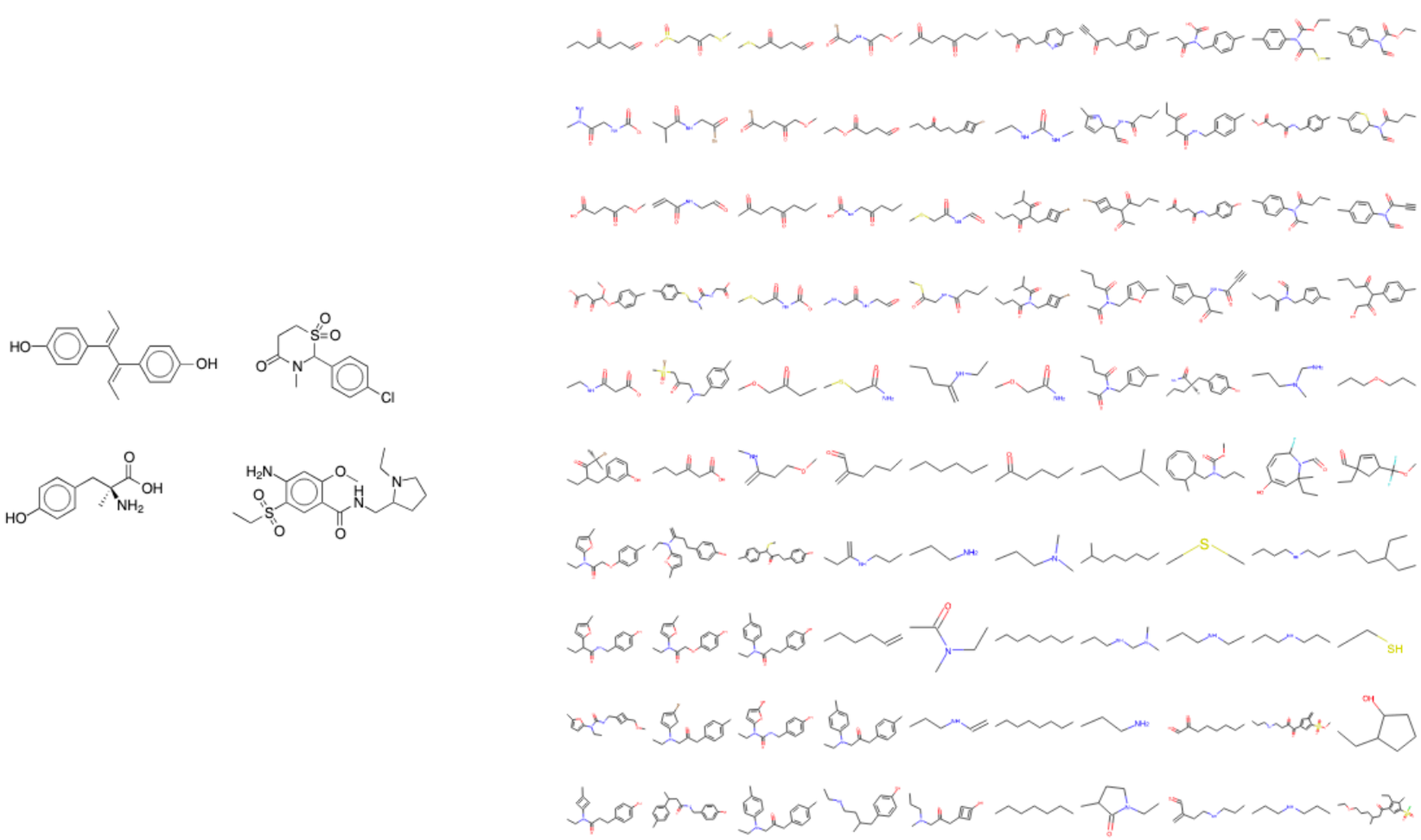


Figure 3: Interpolation. Two-dimensional interpolation between four random drugs. *Left* Starting molecules encoded, whose decodings correspond to the respective four corners of the figure to the right. *Right* Decodings of interpolating linearly between the latent representations of the four molecules to the right.

Source: Automatic Chemical Designusing Variational Autoencoders (2016, Gómez-Bombarelli et al.)

Variational Autoencoder

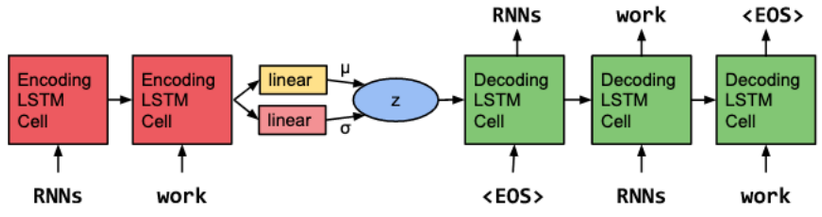


Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned dictionary of embedding vectors.

Variational Autoencoder

i went to the store to buy some groceries .
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

Table 1: Sentences produced by greedily decoding from points between two sentence encodings with a conventional autoencoder. The intermediate sentences are not plausible English.

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

Source: Generating Sentences from a Continuous Space (2016, Bowman et al.)

Variational Autoencoder

- The latent space in VAEs is (mostly) smooth in the sense that if we interpolate in that space we get something novel, something which was not trained on but looks "real".

- To make sure that the latent space follows the rules which we expect (e.g. grammatical rules), we need to add extra terms to our loss function, so that when something breaks these rules, additional loss is applied.

Hands-on

Go to https://jupyter.lisa.surfsara.nl:8000

or https://dba.projects.sda.surfsara.nl/

Notebook: `07-autoencoder.ipynb`

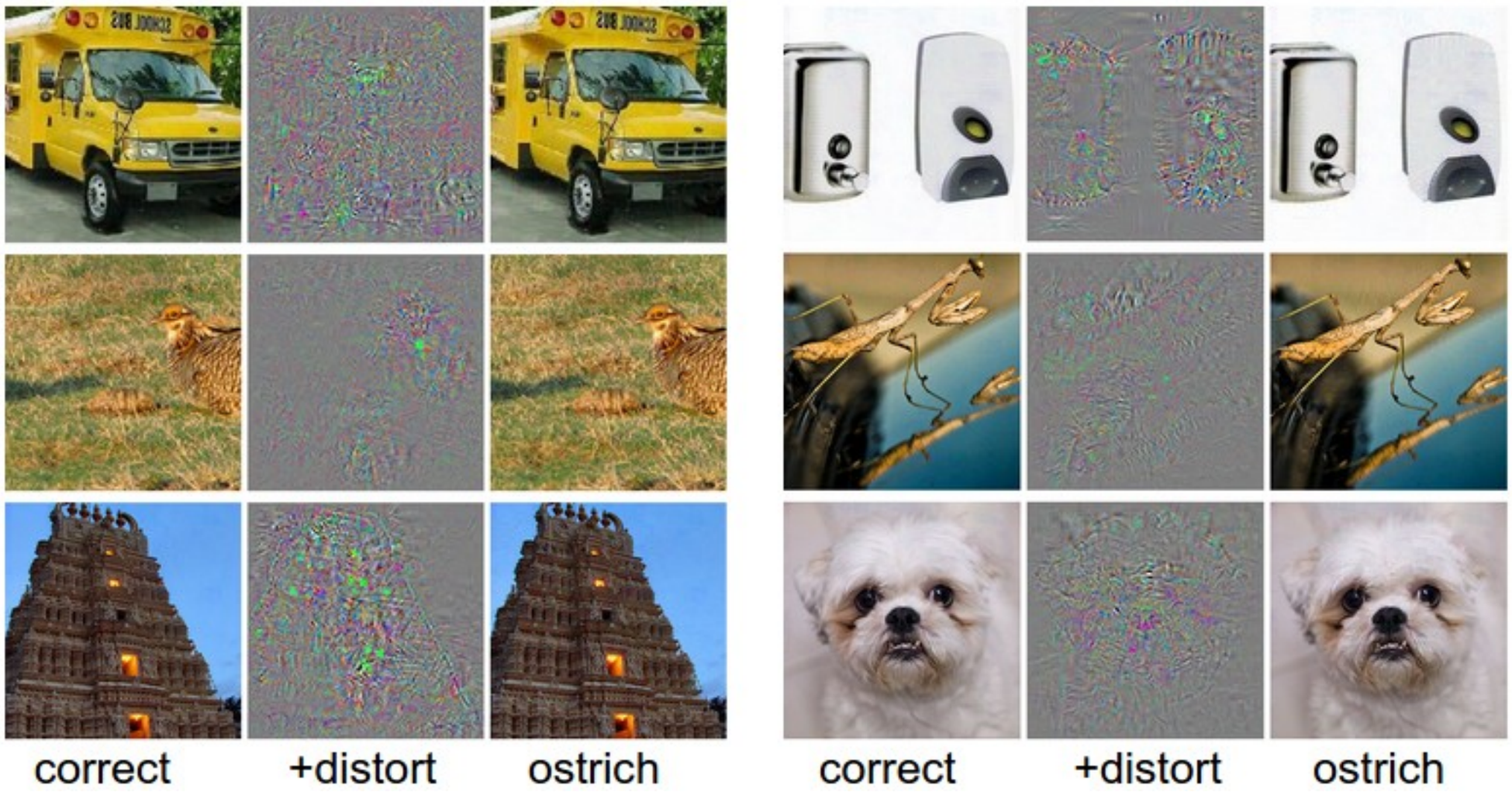
19:30-20:15

Adversarial attacks

- All classifiers are deterministic, most DL models are after they have been trained.
    - We just saw VAEs which are not deterministic, but they are not classifiers.
- People can "game" these classifiers.

Adversarial attacks

- Starting from 2014, researchers conducted adversarial attacks.
- https://www.youtube.com/watch?v=PrU9R6eFNTs

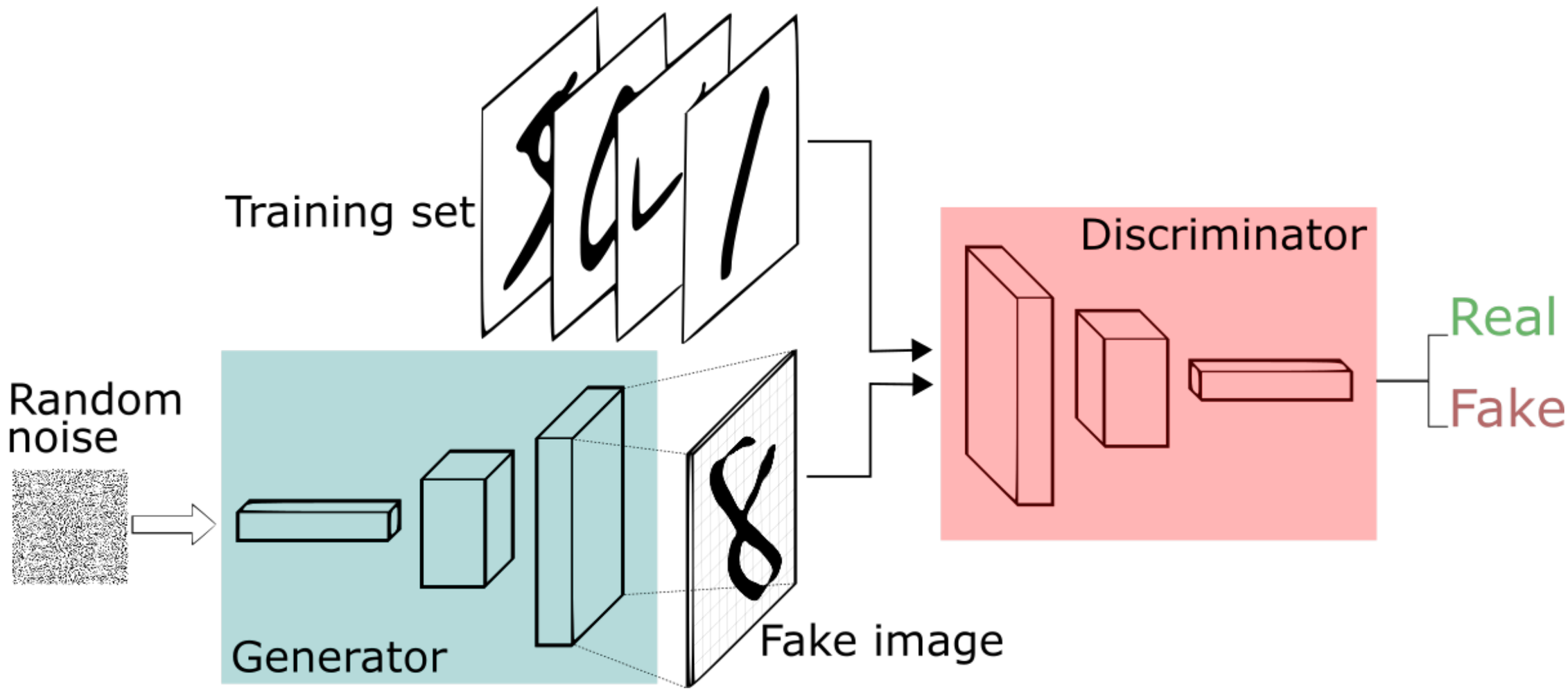


Source: https://karpathy.github.io/2015/03/30/breaking-convnets/

Adversarial attacks

- To begin with classifiers were trained additionally on the adversarial images, but that does not scale.
- Instead, let us try to **detect real and fake images**, then we can always detect adversarial attacks.
- We create fake images from random noise, this is the **generator**.
- We then train a classifier to detect whether the images are from the generator (fake) or whether they are real, this is the **discriminator**.
- The network is then in two parts, the **generator** and the **discriminator**.
- The generator wants the discriminator to think that the images are real, while the discriminator wants to tell real images from fake.
- It is a min-max game.

GANs



Source: https://medium.freecodecamp.org/an-intuitive-introduction-to-generative-adversarial-networks-gans-7a2264a81394

GANs

- For each input image we do two weight updates, one to update the weights of the generator and another for the discriminator.

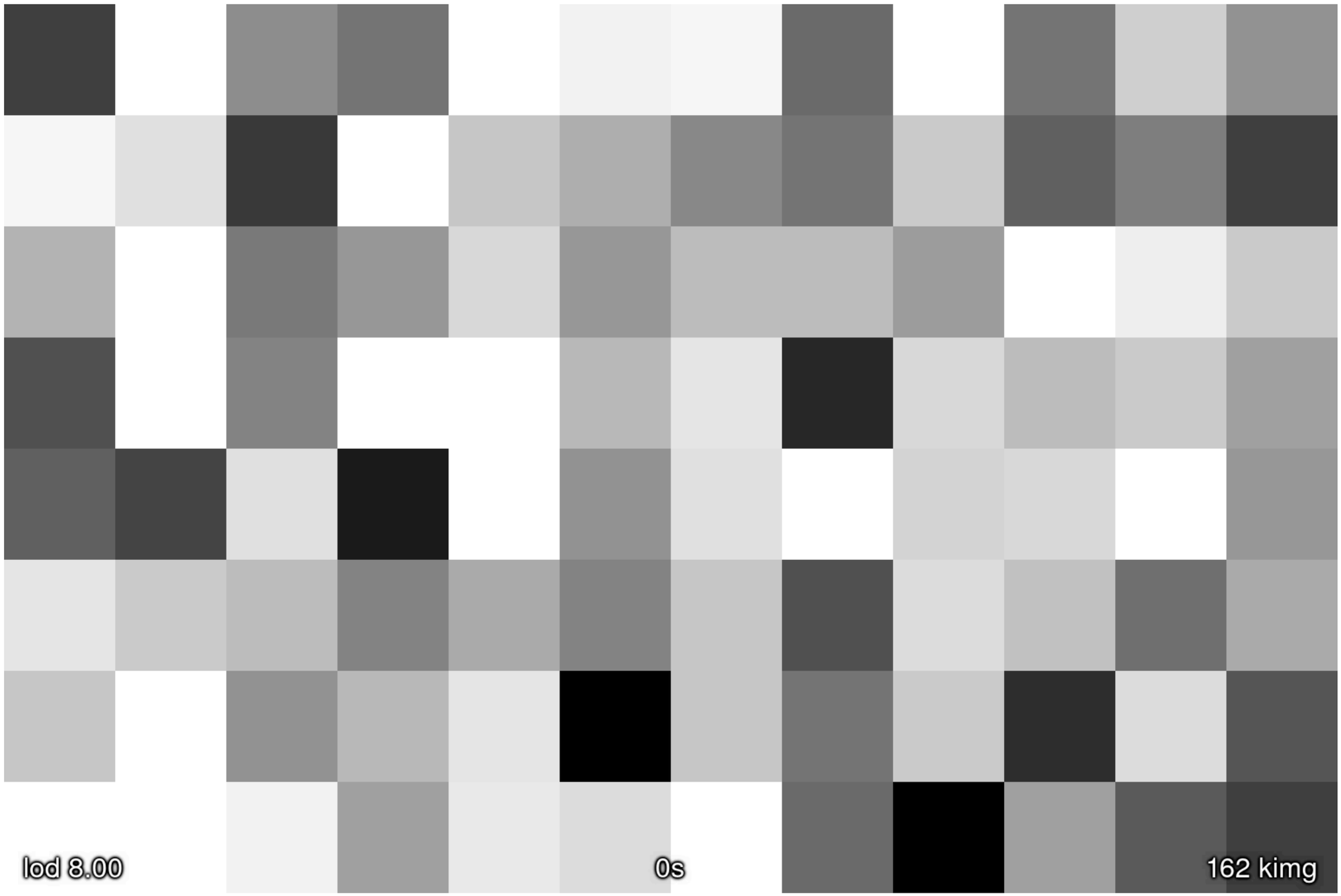
- As time goes on during training, the generator gets better and better and in the end the discriminator is can only guess.

- Then we have a generator which can create realistic images from the same domain as the training data.

GANs

Source: https://www.jeremyjordan.me/convnet-architectures/

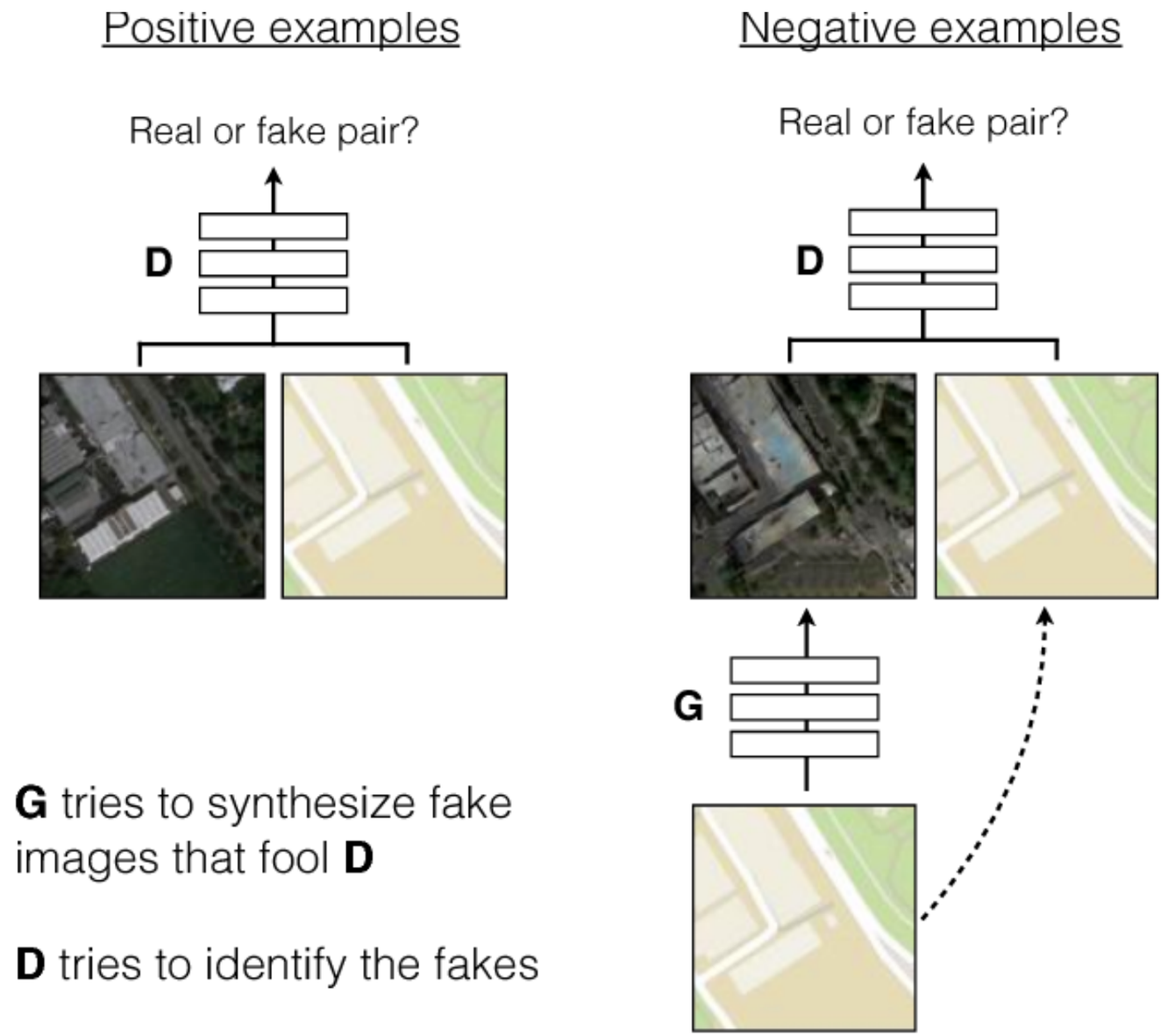
GANs



Source: https://www.jeremyjordan.me/convnet-architectures/

Conditional GANs

- Instead of asking whether and image is fake or real, we ask for a pair of images.
- The generated image (left) may be very realistic, but if it doesn't match with the condition (right), the detector will identify it as fake!



Source: Image-to-Image Translation with Conditional Adversarial Networks (2016, Isola et al.)

Conditional GANs

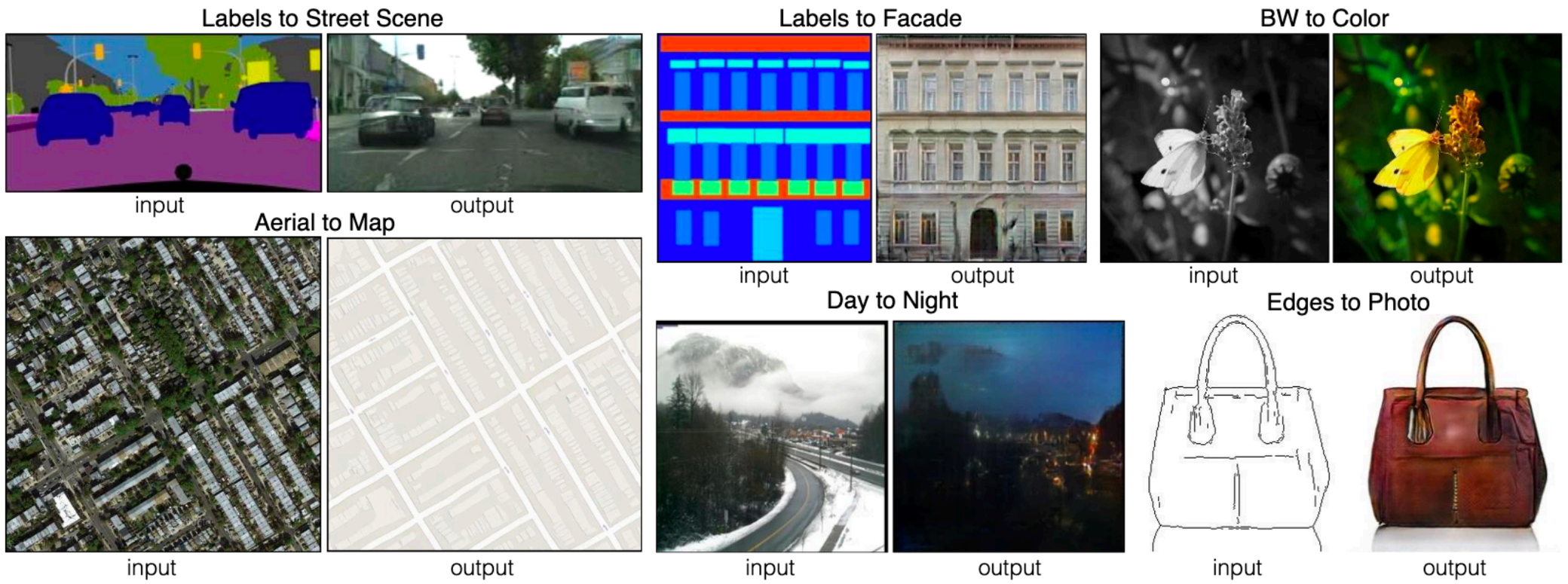


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

Source: Image-to-Image Translation with Conditional Adversarial Networks (2016, Isola et al.)

cycle GANs

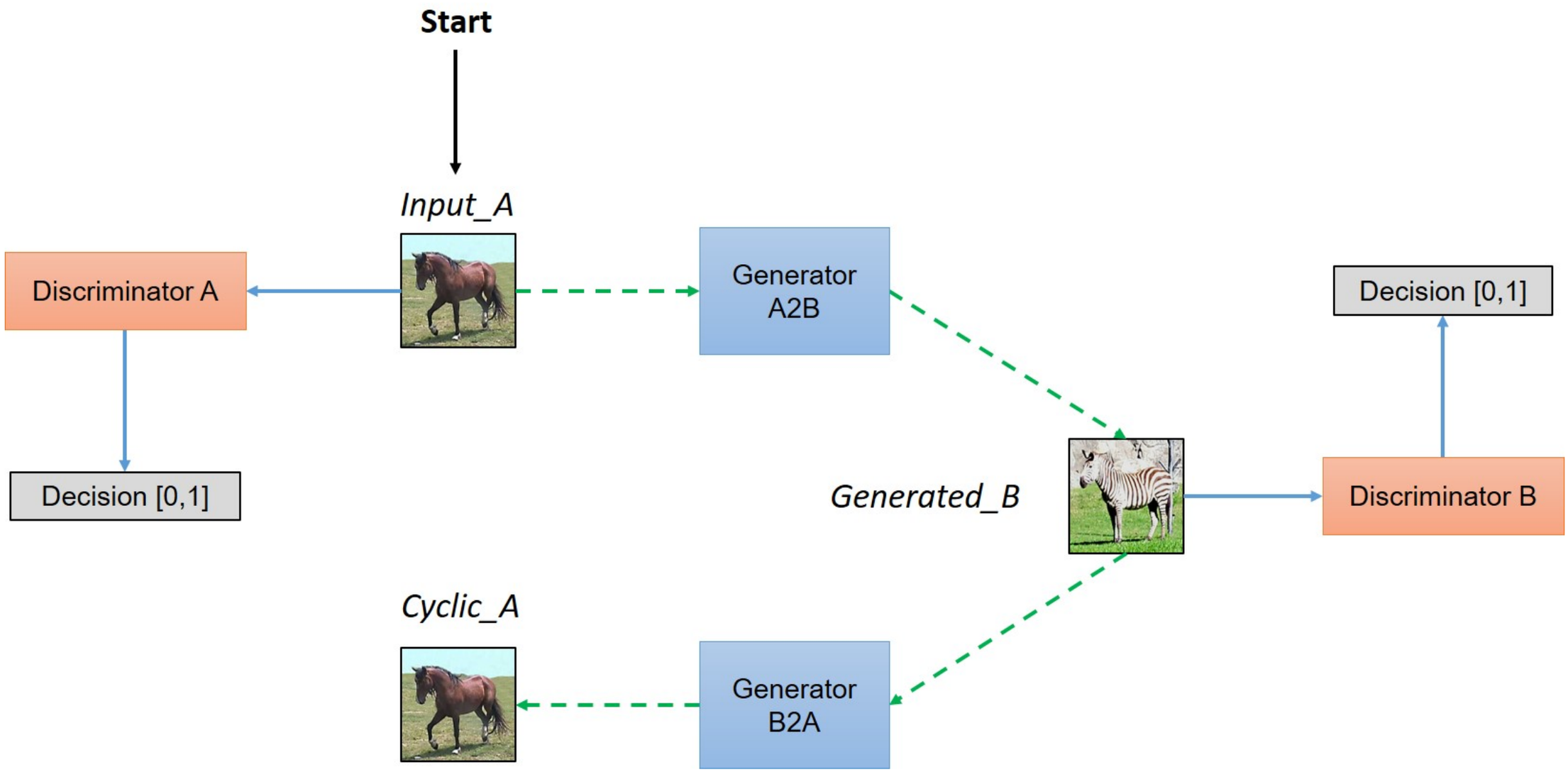
- The problem with conditional GANs is that for each outline there must be a correctly filled in image to train on.

- Having to find a training data which consists of pairs of images is hard, and sometimes they do not even exist!

- Cycle GANs address this problem by considering mappings between two distinct datasets.

Source: Unpaired Image-to-Image Translationusing Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

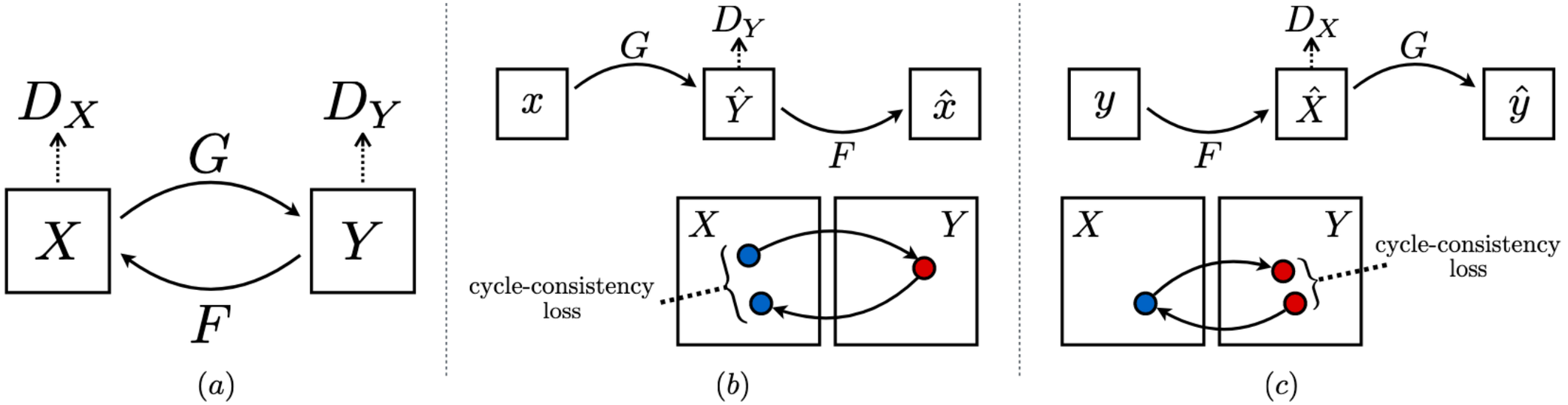
cycle GANs



Source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

cycle GANs

- The loss function is not just the adversarial loss, but also the cycle-consistency loss.



Source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

cycle GANs

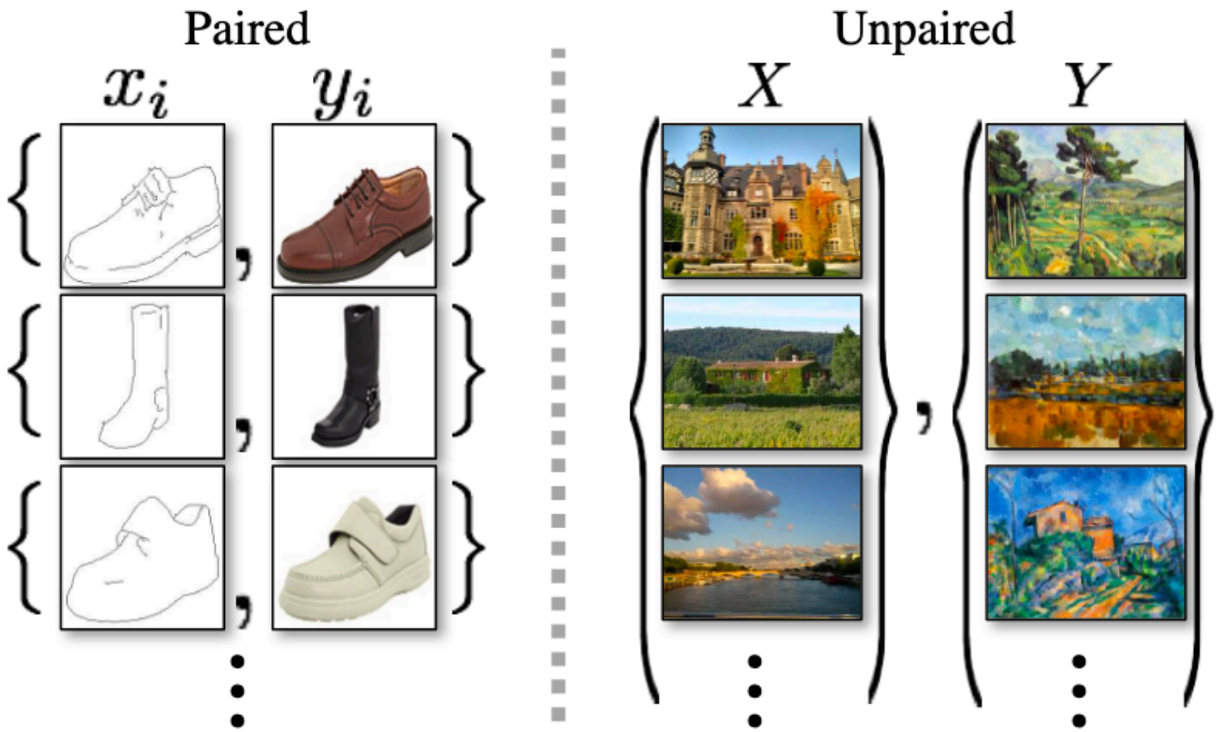


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

Source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

cycle GANs

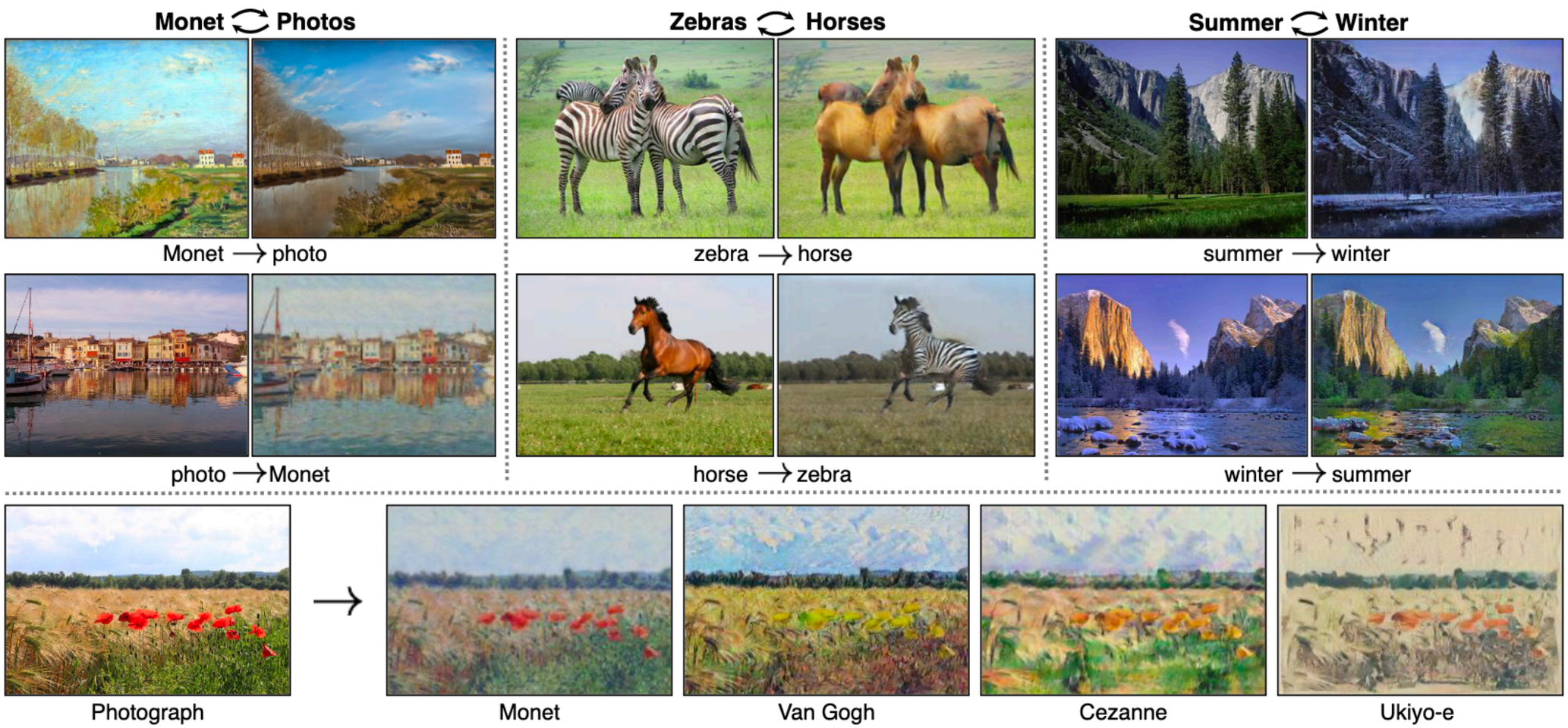
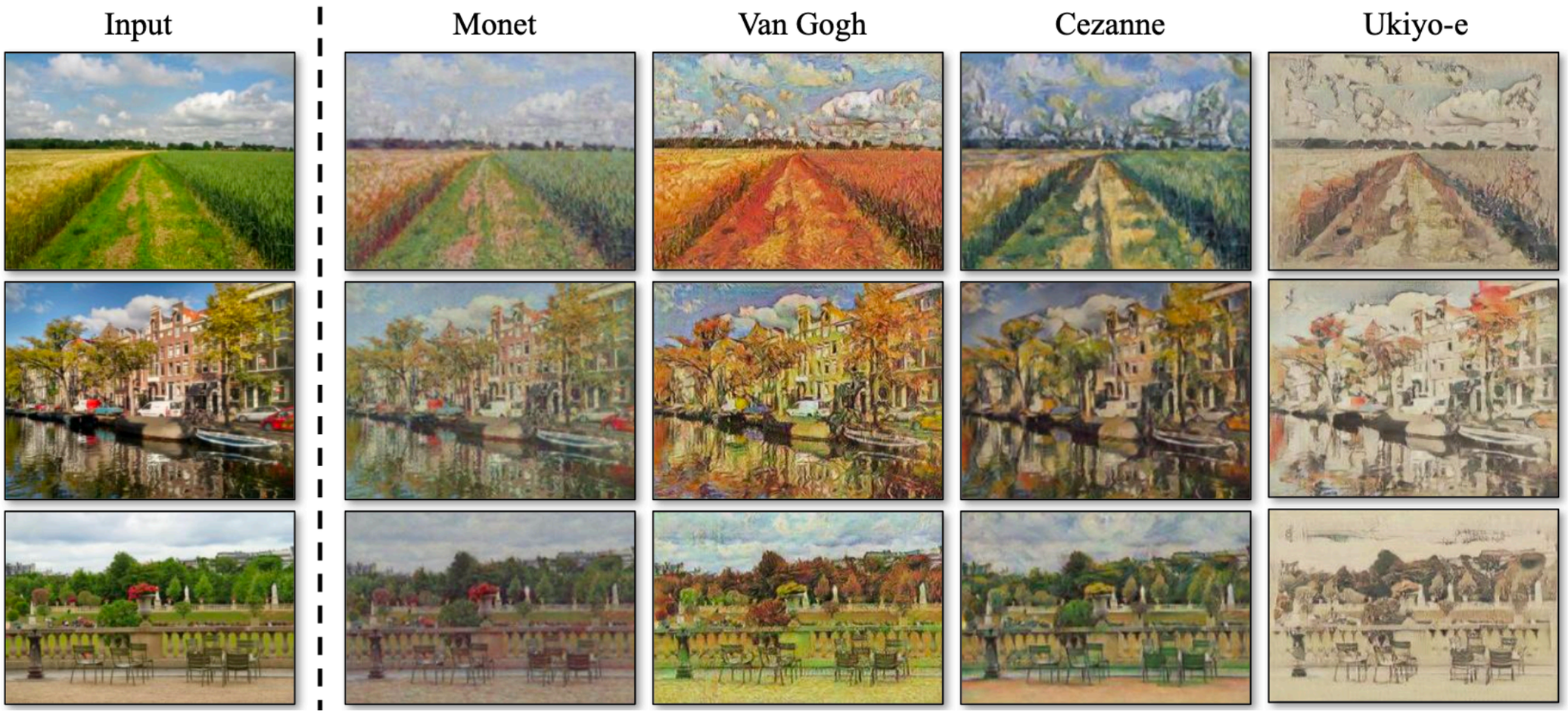


Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

Source: Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

SURF

cycle GANs



Source: Unpaired Image-to-Image Translationusing Cycle-Consistent Adversarial Networks (2017, Zhu et al.)

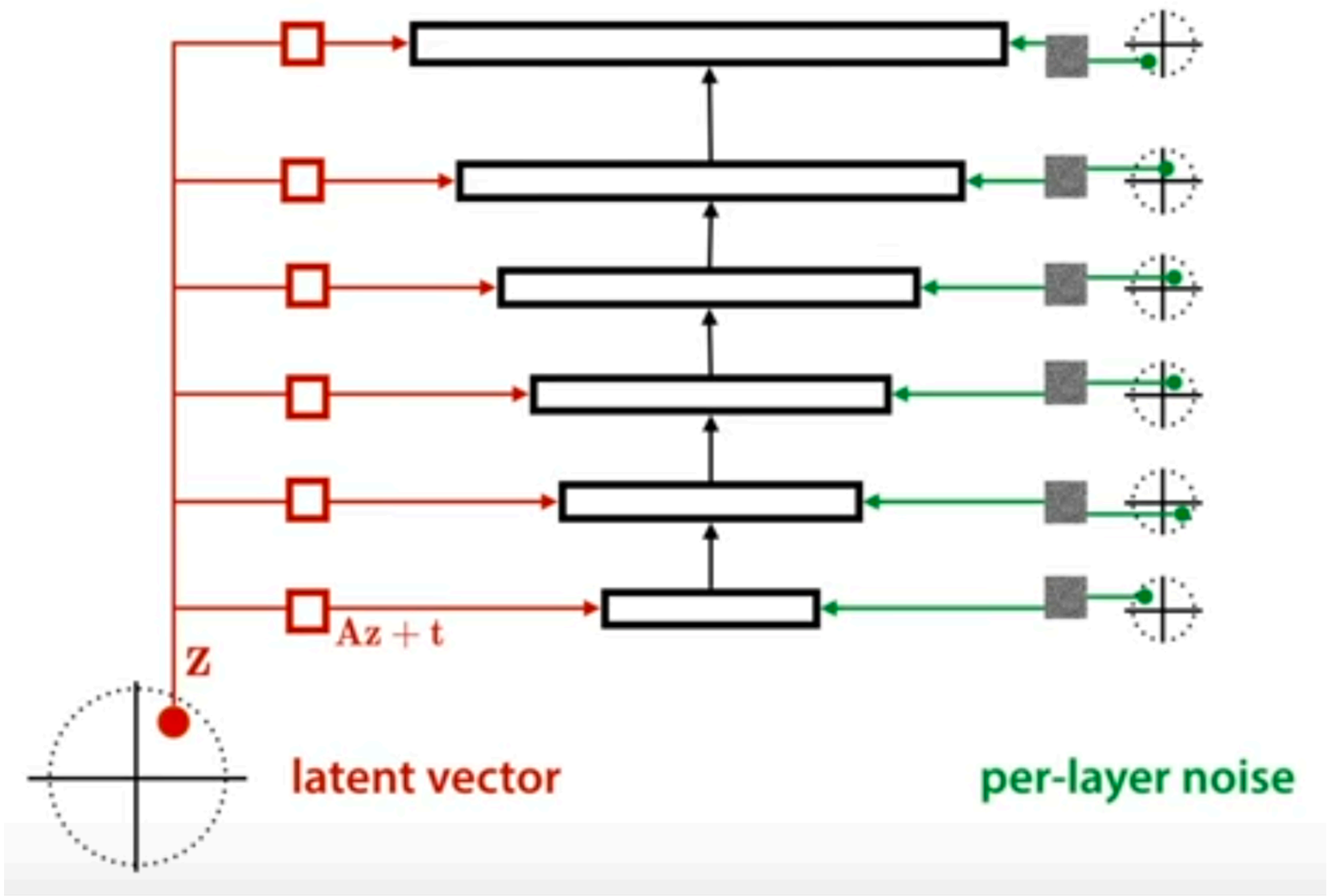
cycle GANs

- https://resources.wolframcloud.com/NeuralNetRepository/tasktype/Image-Processing/
- https://github.com/junyanz/CycleGAN

style GANs

- Images from GANs can be blurry and lack finer details.
- Style GANs address this problem by adding **random noise at multiple stages** in the image generation process.
- First, a **single random vector is sampled** and used **everywhere** in the generation process, this is the overarching style of the image.
- Then **multiple others** are sampled during the process.

cycle GANs



Source: A Style-Based Generator Architecture for Generative Adversarial Networks (2018, Karras et al.)

style GANs

Source: A Style-Based Generator Architecture for Generative Adversarial Networks (2018, Karras et al.)

VAEs vs GANs

- Both learn the distribution of the data, i.e. they are generative.
- GANs are better for images,
- VAEs work better for data with some structure (music, language)
- VAEs have a strong mathematical foundation, GANs do not.
  - Unknown what is being optimised for GANs.

Thank you!

- Thank you for your interest, you were a very engaging group!
- Remaining questions?
- Please fill out the feedback form!