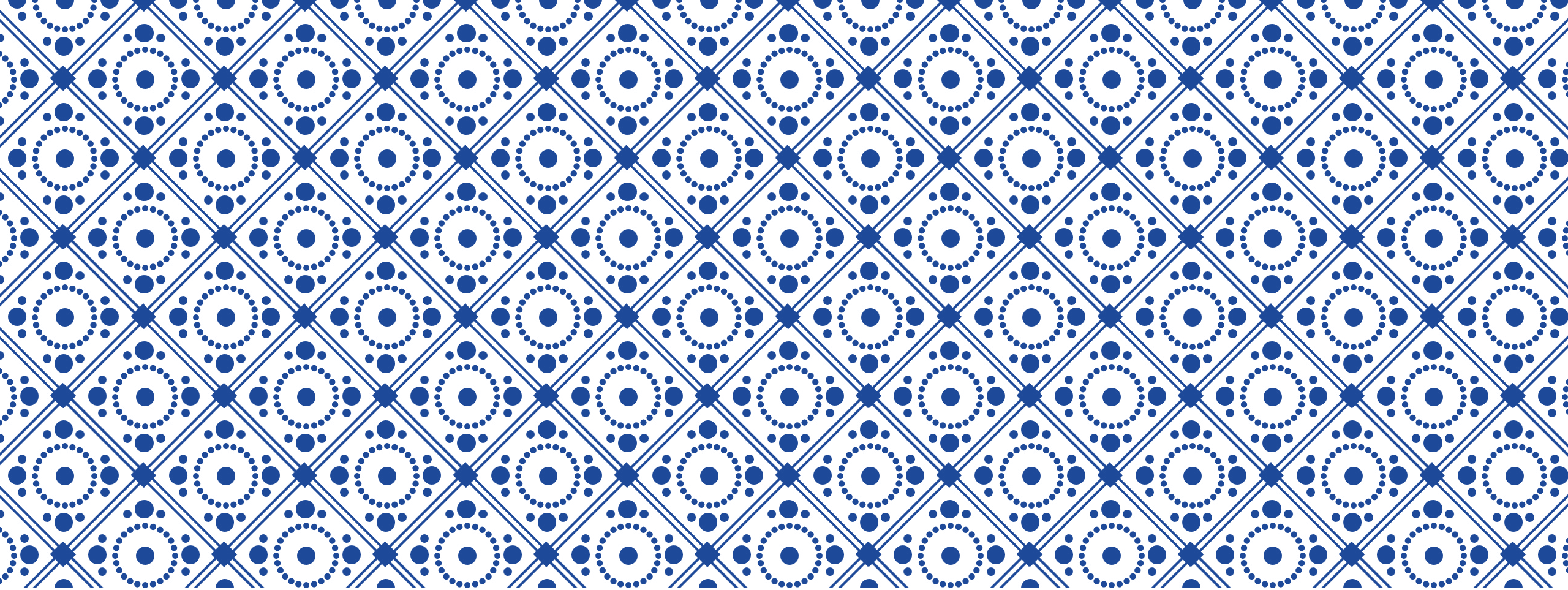


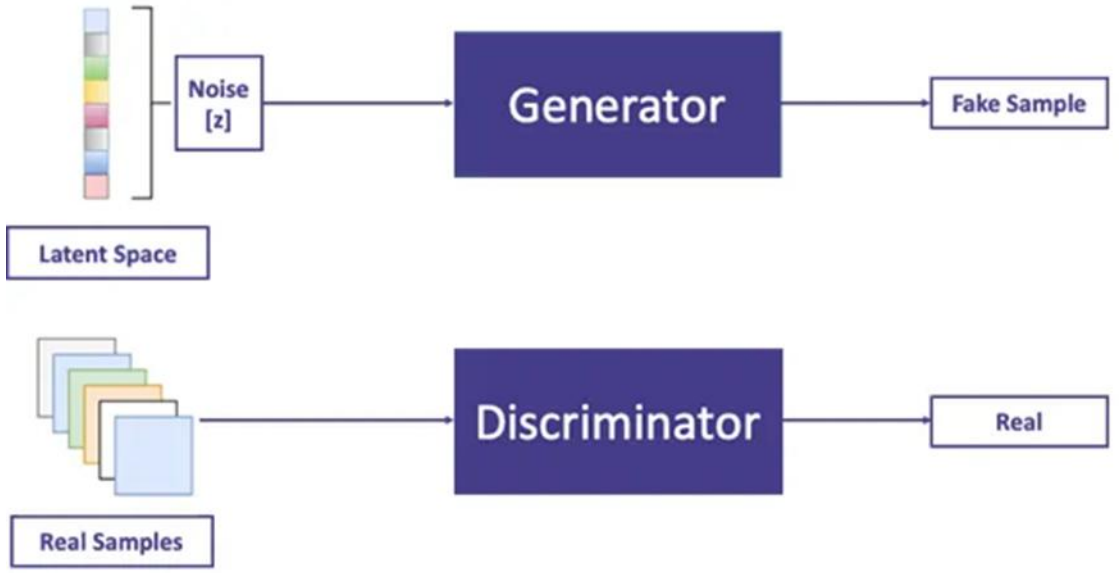
GENERATIVE ADVERSARIAL NETWORKS (GANS)

GENERATIVE ADVERSARIAL NETWORKS

- Generative adversarial networks (GANs), introduced in 2014 by Goodfellow et al. are a mechanism for learning latent spaces of images.
- They enable the generation of fairly realistic synthetic images by forcing the generated images to be statistically almost indistinguishable from real ones.
- Intuition: Painting forger vs Expert
  - The forger produces fake paintings of "masters", while the "expert" detects them.
  - As the forger gets more practice he starts producing more and more realistic "fakes" while the expert gets better and better at detecting "fakes."
  - In the end, they have some excellent "fake" paintings

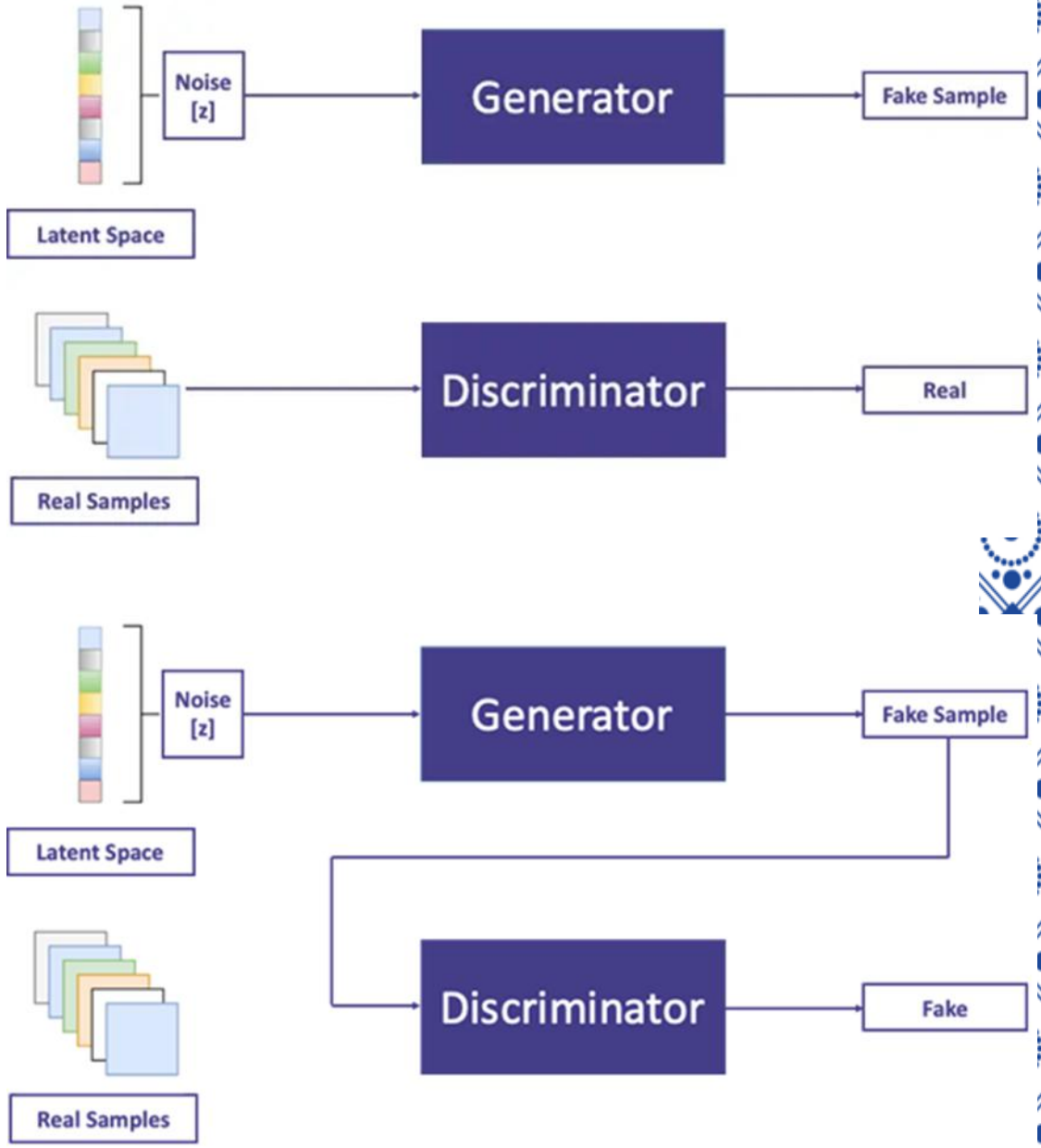
GENERATOR AND DISCRIMINATOR AS GAN BUILDING BLOCKS

- A GAN is made up of two parts
- Generator network—Takes as input a random vector (a random point in the latent space), and decodes it into a synthetic image
- Discriminator network (or adversary)—Takes as input an image (real or synthetic), and predicts whether the image came from the training set or was created by the generator network



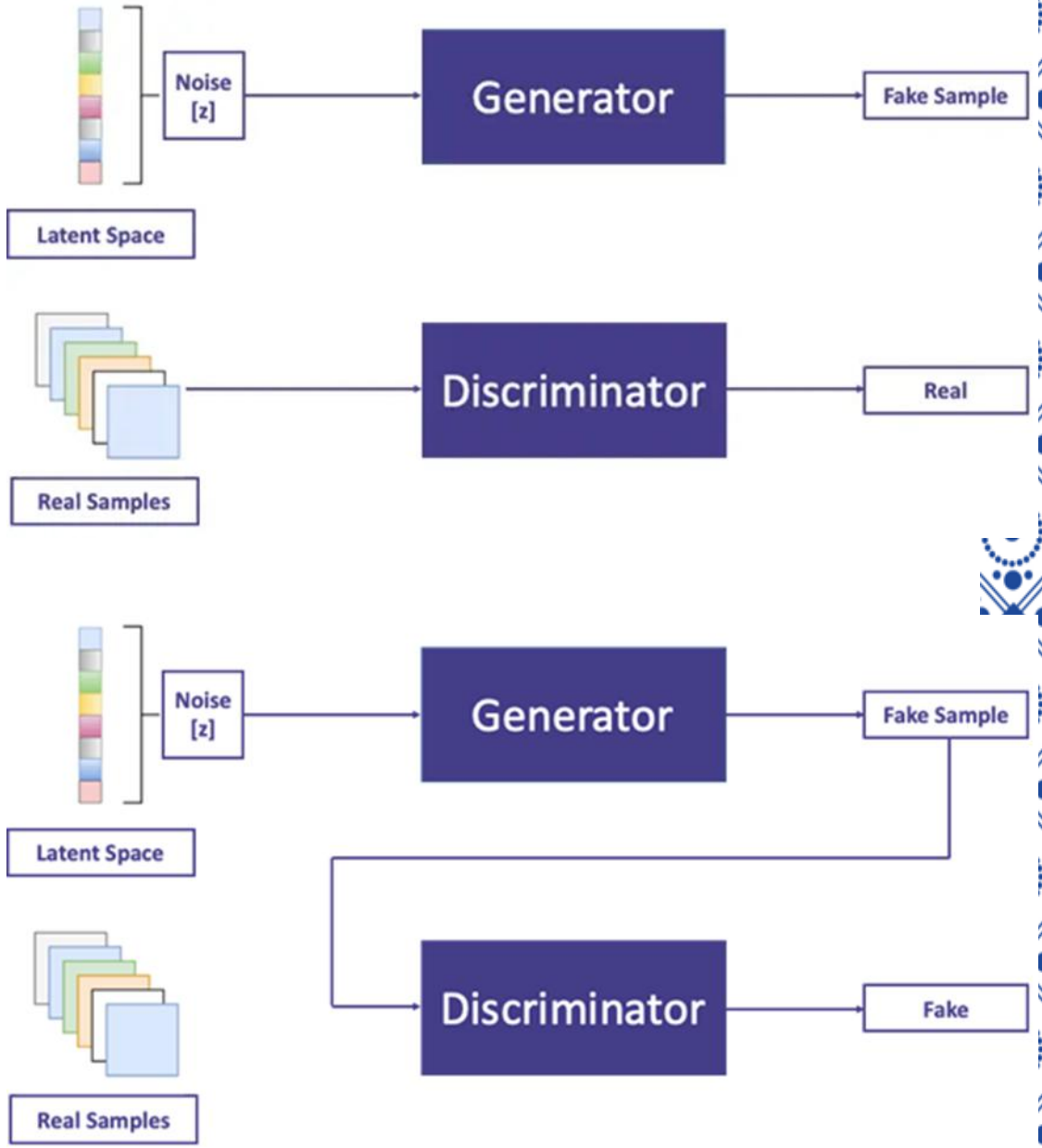
GENERATOR AND DISCRIMINATOR AS GAN BUILDING BLOCKS

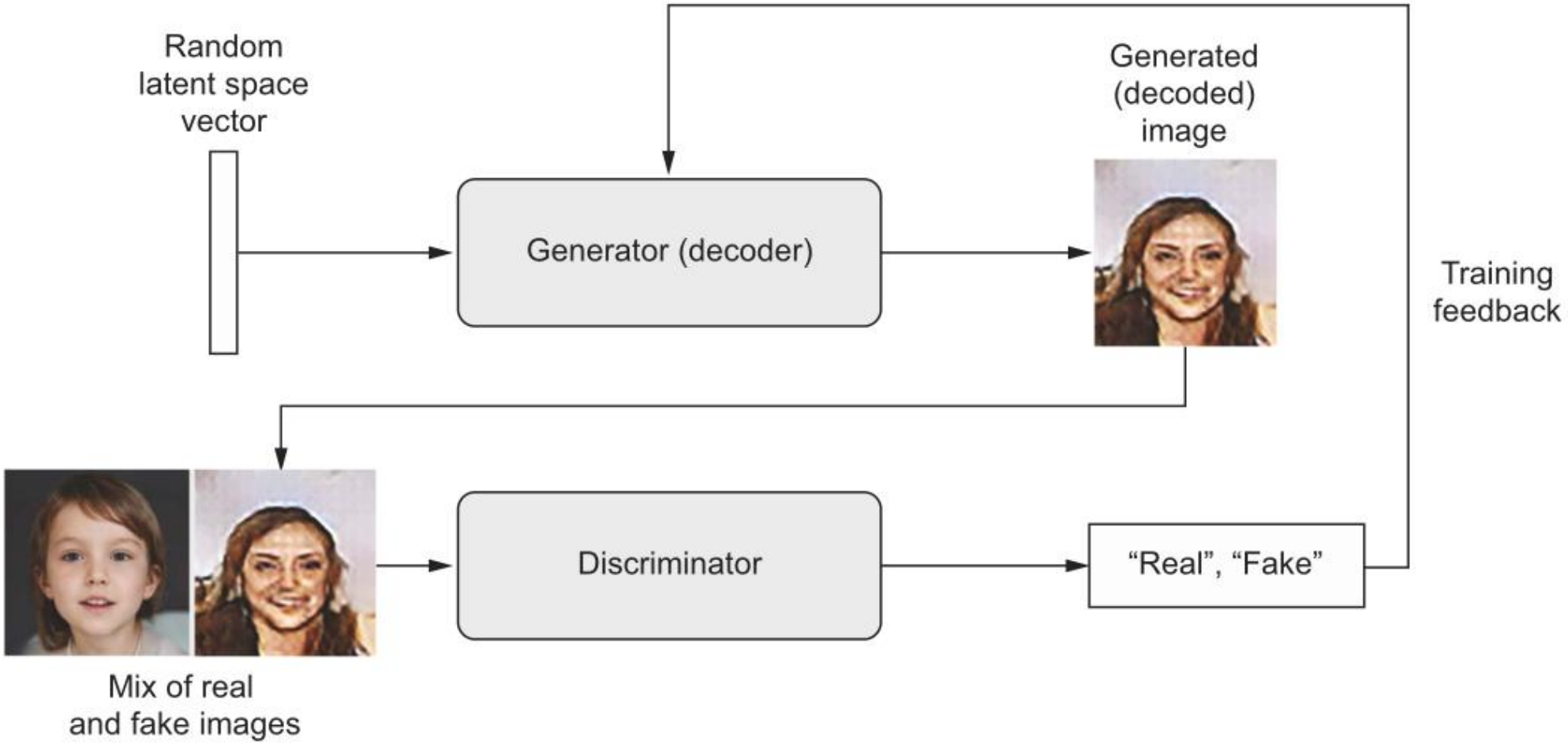
- The generator network is trained to be able to fool the discriminator network
  - It evolves toward generating increasingly realistic images as with increased training
  - Generates artificial images that look indistinguishable from real ones
  - Idea is that it should be impossible for the discriminator network to tell the two apart.
- At the same time, the discriminator is constantly adapting to:
  - the gradually improving capabilities of the generator
  - setting a high bar of realism for the generated images.



GENERATOR AND DISCRIMINATOR AS GAN BUILDING BLOCKS

- Once training is over, the generator is capable of turning any point in its input space into a believable image.
- The GAN latent space has fewer explicit guarantees of meaningful structure; in particular, it isn't continuous.

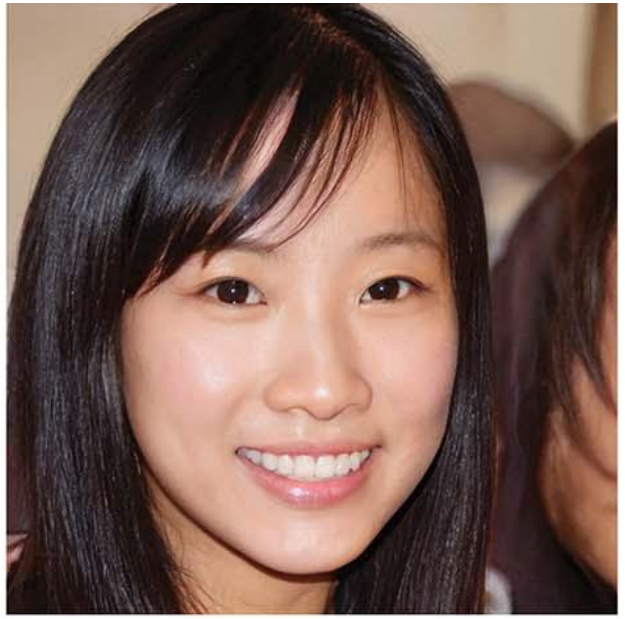



Random
latent space
vector
Generator (decoder)
Generated
(decoded)
image
Training
feedback
Discriminator
“Real”, “Fake”
Mix of real
and fake images

GENERATIVE ADVERSARIAL NETWORKS

- A GAN is a system where the optimization minimum isn't fixed, unlike in other networks like CNN, etc. where we try to achieve zero error.
- Normally, gradient descent consists of rolling down hills in a static loss landscape.
- But with a GAN, every step taken down the hill changes the entire landscape a little.
- It's a dynamic system where the optimization process is seeking not a minimum, but an equilibrium between two forces – an equilibrium between the discriminator and the generator
- For this reason, GANs are very difficult to train
  - getting a GAN to work requires lots of careful tuning of the model architecture and training parameters.

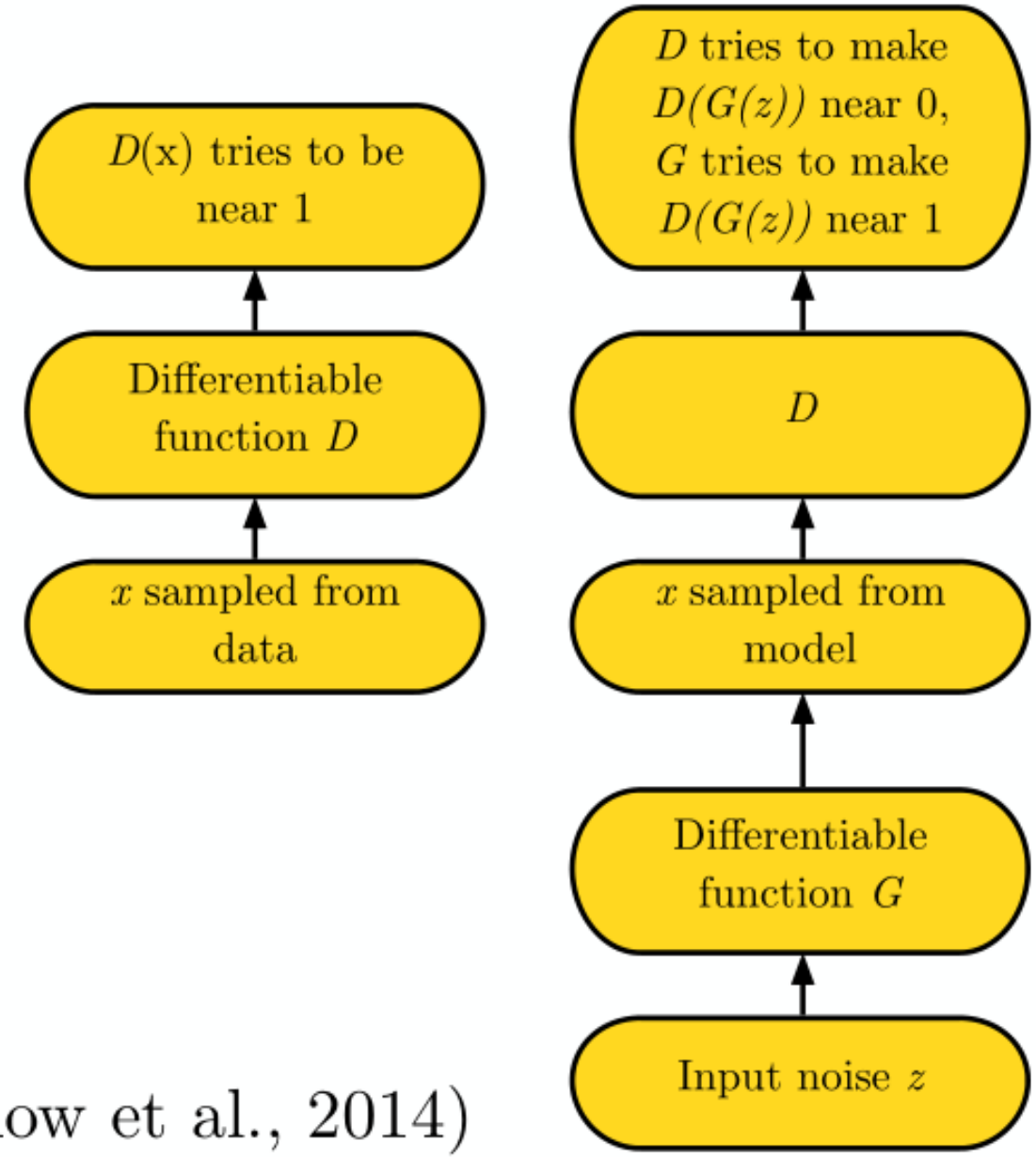




Realistic Fakes

Latent space dwellers. Images generated by https://thispersondoesnotexist.com using a StyleGAN2 model. (Image credit: Phillip Wang is the website author. The model used is the StyleGAN2 model from Karras et al., https://arxiv.org/abs/1912.04958.)

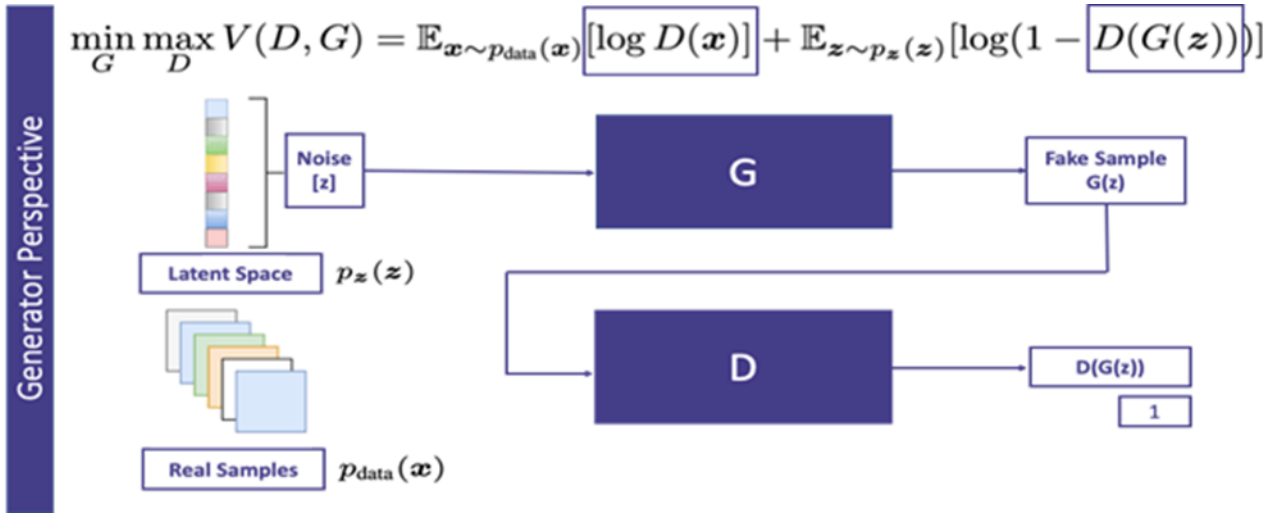
Adversarial Nets Framework



(Goodfellow et al., 2014)

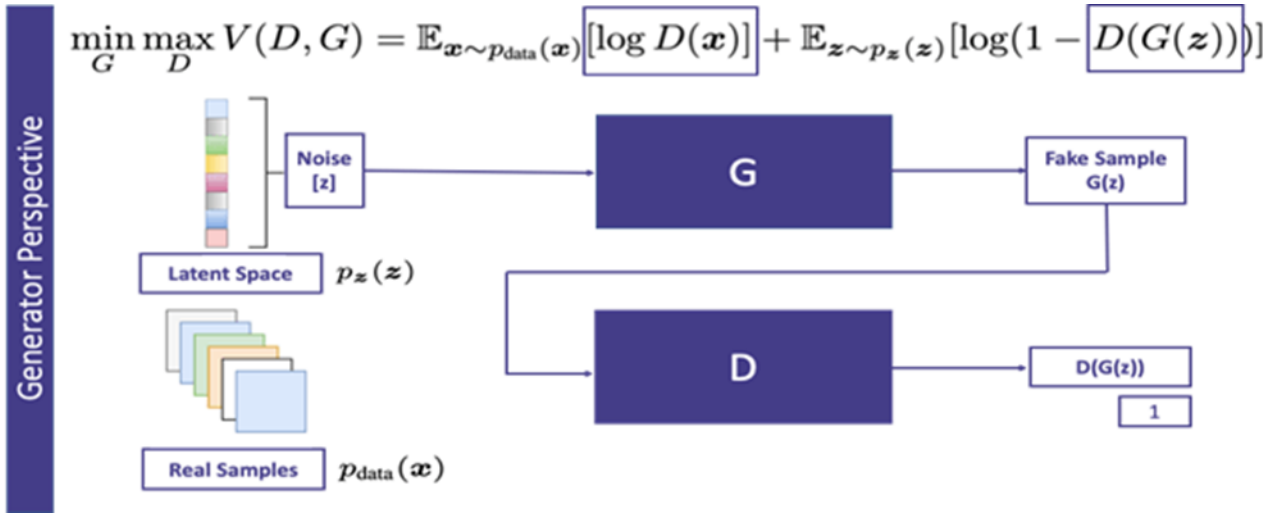
OPTIMIZATION IN GANS [1]

- In GANs the optimization is a min-max optimization formulation where:
  - the Generator wants to minimize the objective function whereas
  - the Discriminator wants to maximize the same objective function.
- The Discriminator function is termed as D and the Generator function is termed as G.
- P_z is the probability distribution of the latent space which is usually a random Gaussian distribution.
- P_{data} is the probability distribution of the training dataset.
- When x is sampled from P_{data}, the Discriminator wants to classify it as a real sample. G(z) is a generated sample when G(z) is given as input to the Discriminator, it wants to classify it as a fake one.



OPTIMIZATION IN GANS [2]

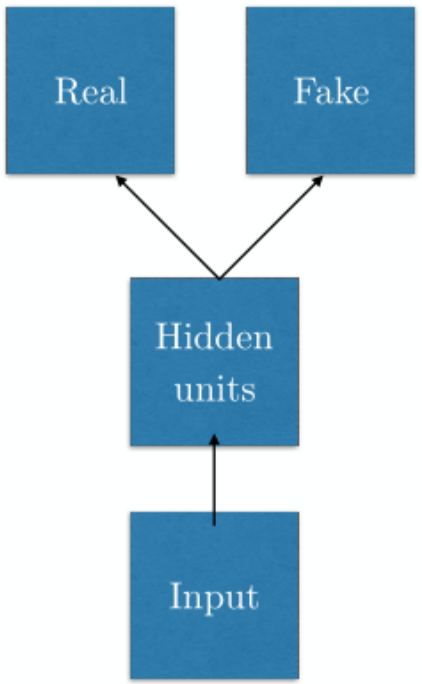
- ✓In GANs the optimization is a min-max optimization formulation where:
  - ✓the Generator wants to minimize the objective function whereas
  - ✓the Discriminator wants to maximize the same objective function.
- ✓The Discriminator wants to drive the likelihood of D(G(z)) to 0. Hence it wants to maximize (1-D(G(z)))
- ✓The Generator wants to force the likelihood of D(G(z)) to 1 so that Discriminator makes a mistake in calling out a generated sample as real. Hence Generator wants to minimize (1-D(G(z)).

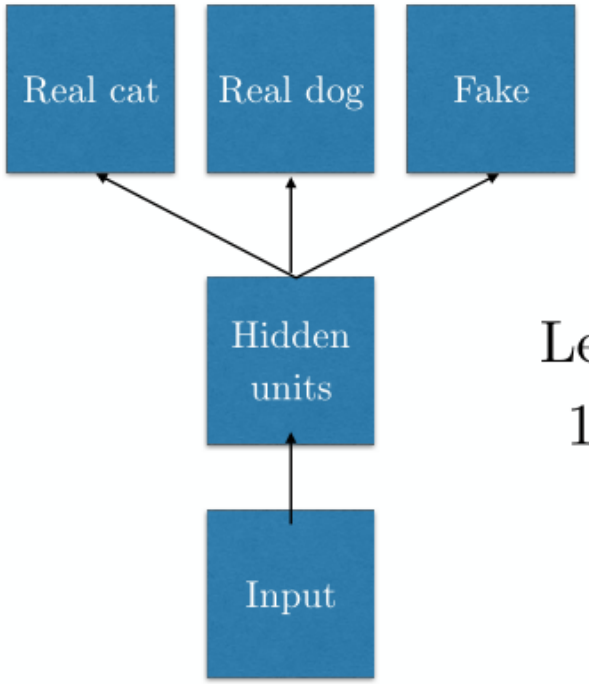


APPLICATIONS OF GANS

- Semi-supervised Learning
- Model-based optimization
- Extreme personalization
- Program synthesis

Supervised Discriminator for Semi-Supervised Learning





Learn to read with 100 labels rather than 60,000

(Odena 2016, Salimans et al 2016)

Semi-Supervised Classification

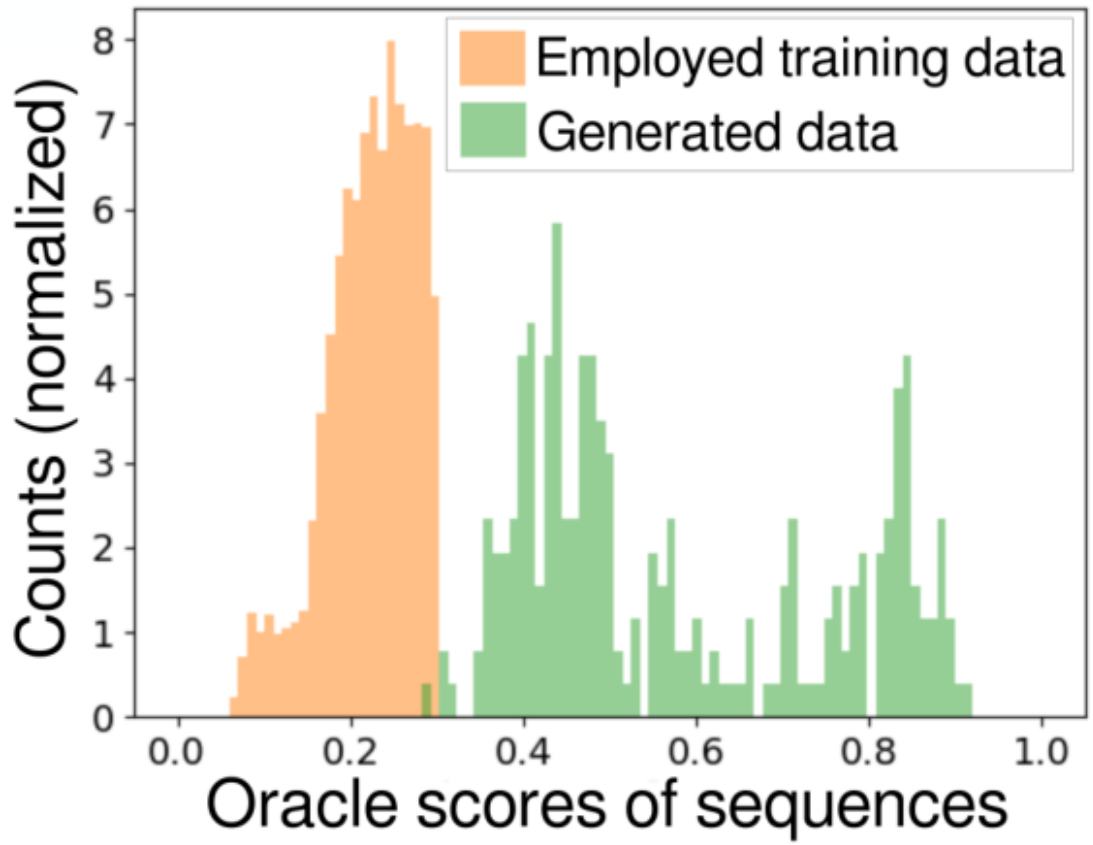
MNIST: 100 training labels -> 80 test mistakes

SVHN: 1,000 training labels -> 4.3% test error

CIFAR-10: 4,000 labels -> 14.4% test error

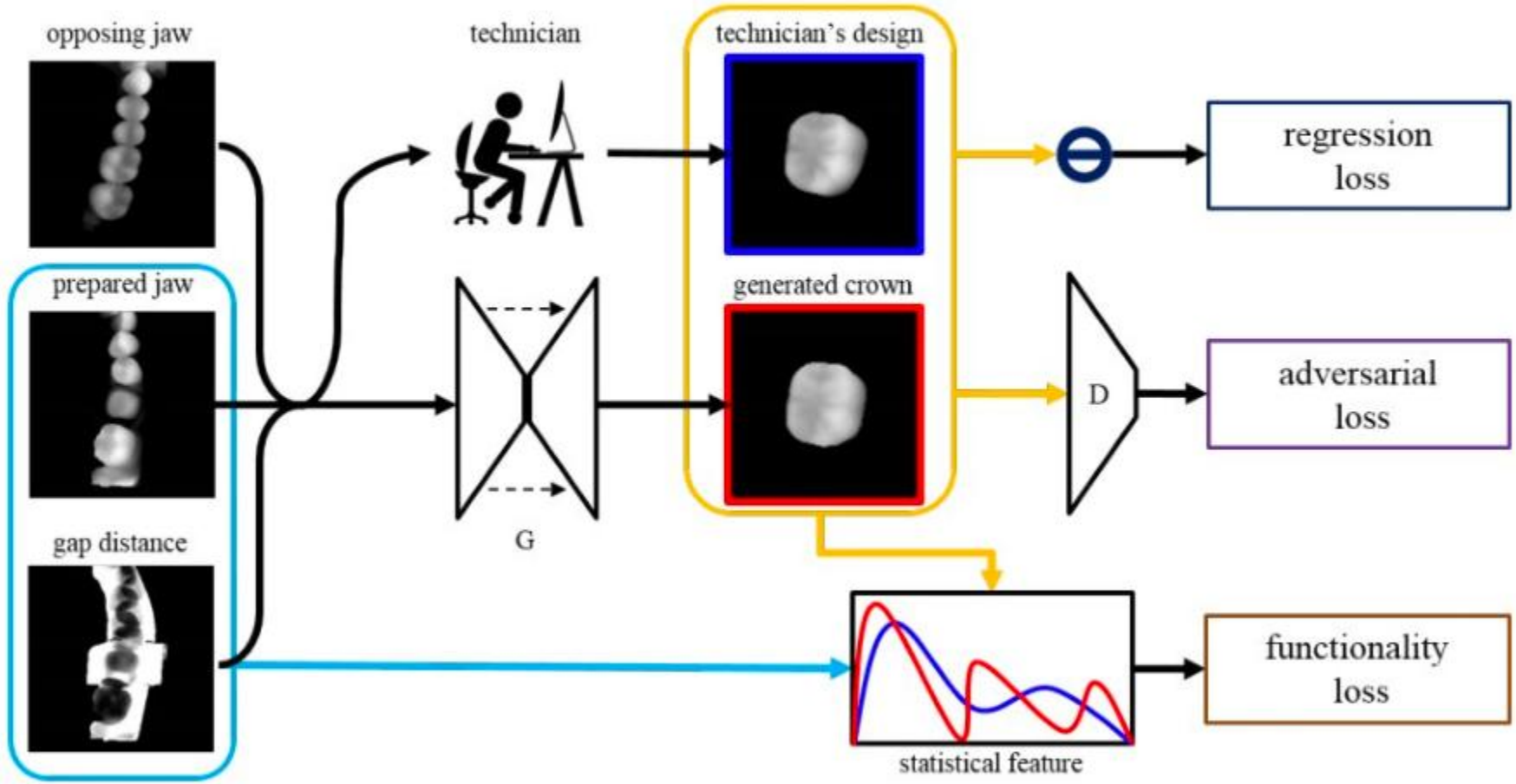
(Dai et al 2017)

Designing DNA to optimize protein binding



(Killoran et al, 2017)

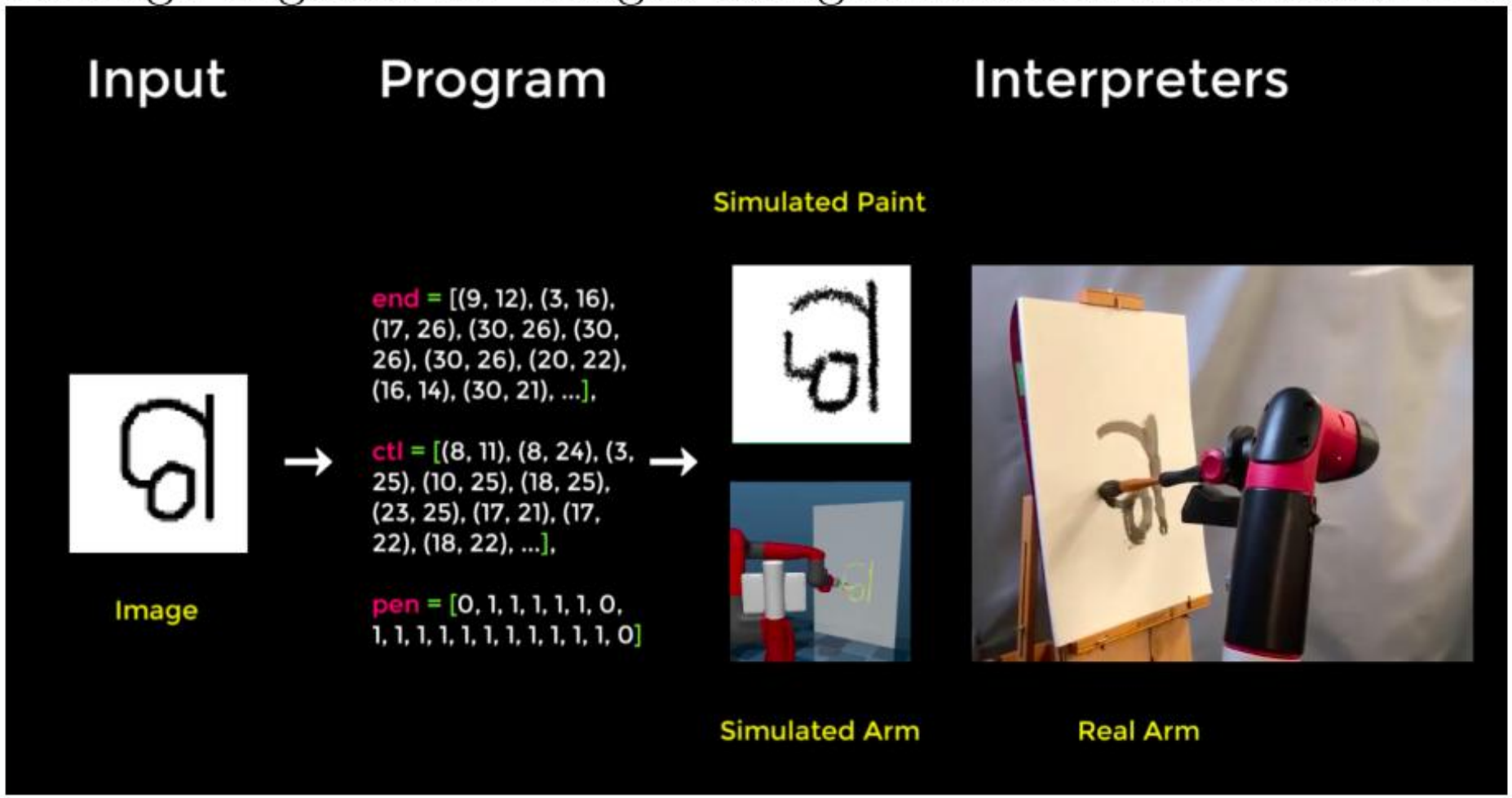
Personalized GANufacturing



(Hwang et al 2018)

SPIRAL

Synthesizing Programs for Images Using Reinforced Adversarial Learning



(Ganin et al, 2018)

OTHER APPLICATIONS

- Planning
- World Models for RL agents
- Fairness and Privacy
- Missing data
- Training data for other agents
- Inference in other probabilistic models
- Domain adaptation
- Imitation Learning

A SCHEMATIC GAN IMPLEMENTATION

Implementation of a GAN in Keras in its barest form.

The specific implementation here in this demonstration is a deep convolutional GAN (DCGAN): a very basic GAN where the generator and discriminator are deep convnets

The GAN is trained on images from the Large-scale CelebFaces Attributes dataset (known as CelebA), a dataset of 200,000 faces of celebrities (http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html)

- Here we use only around a 1,000 of those images

To speed up training, the images are resized to 64 × 64,

- This enables us to learn to generate 64 × 64 images of human faces.

A SCHEMATIC GAN IMPLEMENTATION

A generator network maps vectors of shape (latent_dim) to images of shape (64, 64, 3).

A discriminator network maps images of shape (64, 64, 3) to a binary score estimating the probability that the image is real.

A GAN network chains the generator and the discriminator together: gan(x) =discriminator(generator(x)).

- Thus, this GAN network maps latent space vectors to the discriminator's assessment of the realism of these latent vectors as decoded by the generator.

The discriminator is trained using examples of real and fake images along with "real"/"fake" labels, just as we train any regular image-classification model.

To train the generator, we use the gradients of the generator's weights with regard to the loss of the GAN model.

- This means that at every step, **we move the weights of the generator in a direction that makes the discriminator more likely to classify as "real" the images decoded by the generator**.

In other words, we train the generator to fool the discriminator.

LIMITATIONS OF GANS

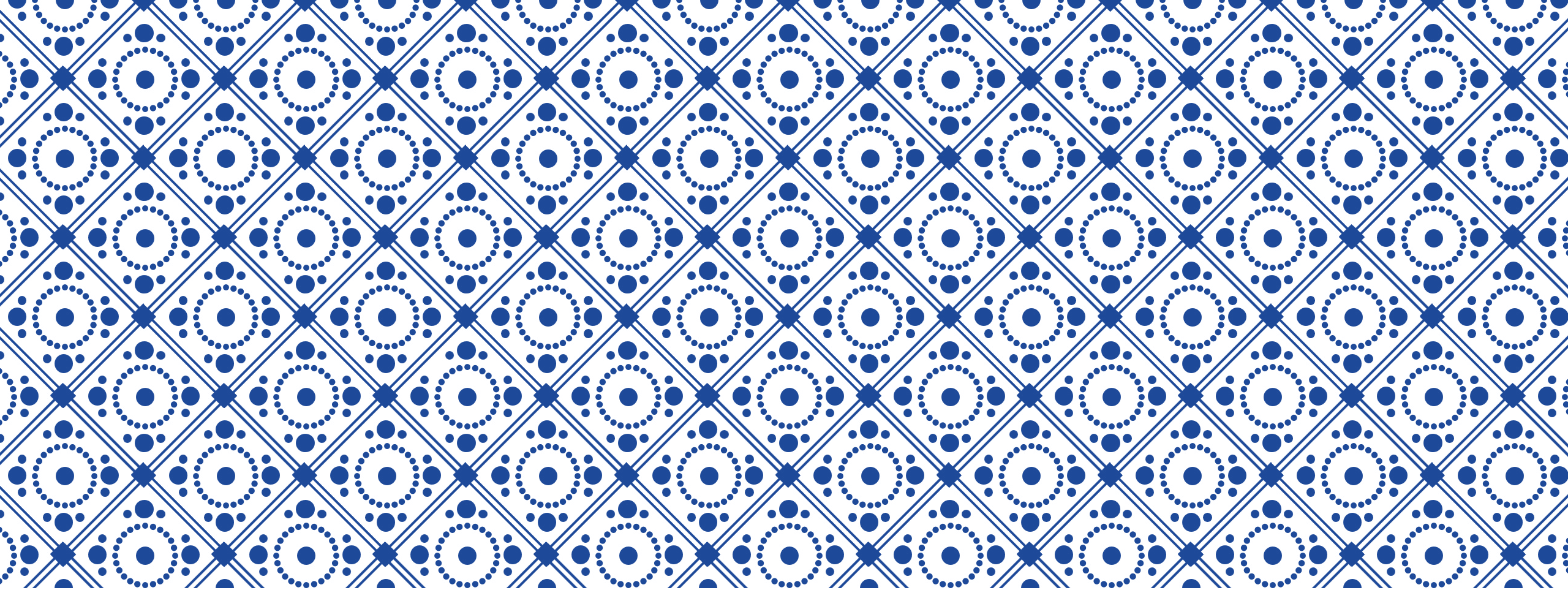
Difficult to train: You need to provide different types of data continuously to check if it works accurately or not.

- The discriminator nearly always wins
- Sometimes training makes it worse
- Additional data sometimes does not improve the output

Generating results from text or speech is very complex

It is not clear whether they actually generate a distribution

Generality Penalty: For a given problem, the application tailored solutions may work better.



THANKS