



# GENERATIVE MACHINE LEARNING

**Creating noise from data is easy; creating data from noise is generative modeling.\***

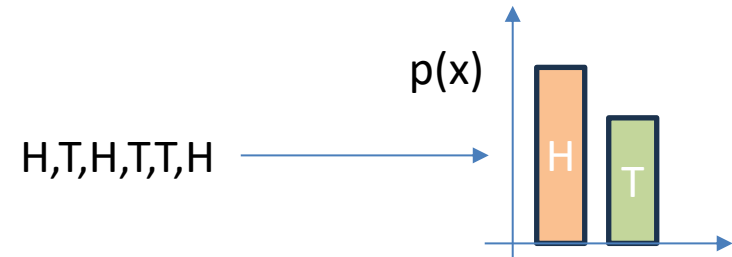
[\*] Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. "Score-Based Generative Modeling through Stochastic Differential Equations." arXiv, February 10, 2021. <https://doi.org/10.48550/arXiv.2011.13456>.

# Background: Introduction to Sampling

- **Empirical distribution Modelling:** Making a distribution from observations (Density Estimation)

– Example:

- Observations: {H,T,H,T,H}
- $P(H) = 3/5 = 0.6$ ,  $P(T) = 2/5 = 0.4$
- Shown as probability distribution (normalized histogram)

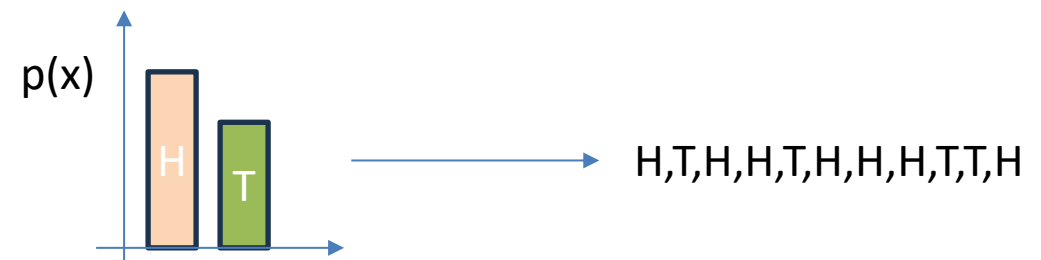


- **Sampling** from a distribution

– Assume you are given a probability distribution  $p(x)$ , then if you “sample” from it, you will be generating samples  $x$  which when observed will give you  $p(x)$

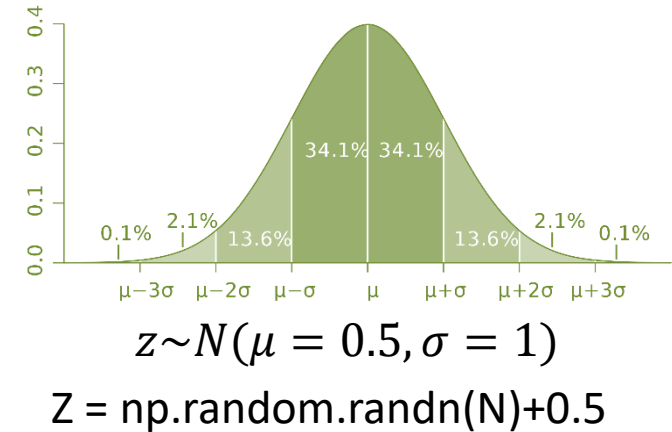
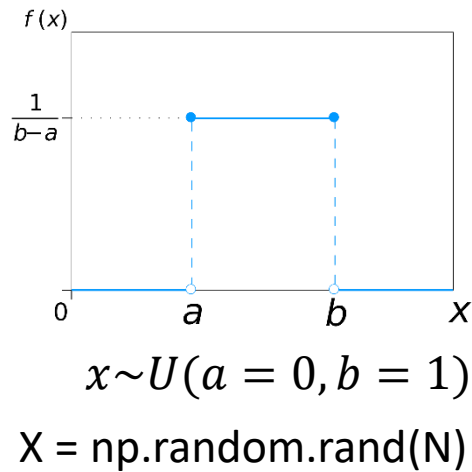
– Example

- Given:  $P(H) = 0.6$ ,  $P(T) = 0.4$
- Generated Samples: {H,T,H,T,H,T,H,H,T,H}

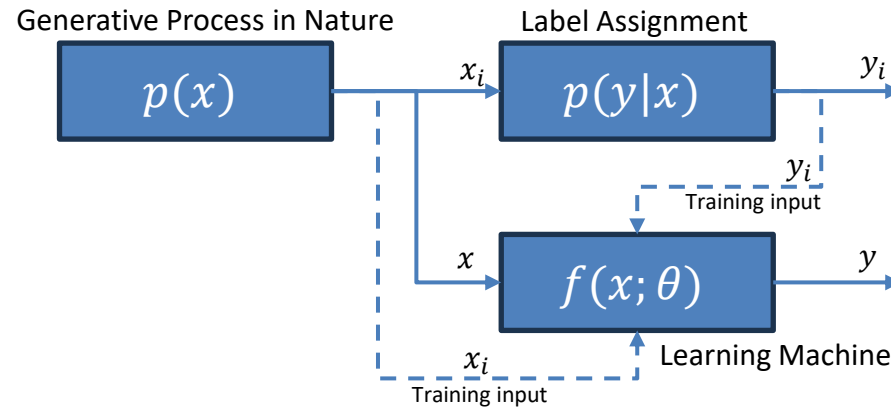
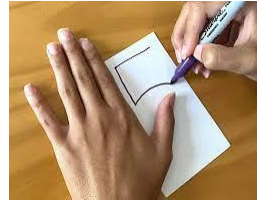


# Background: Generating samples

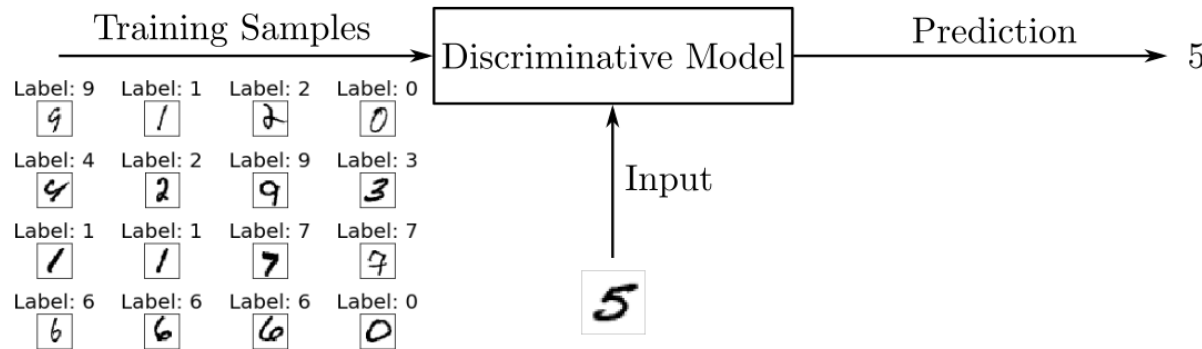
- Can we **generate** samples of a target distribution using samples from a source distribution as input?



# A generative look at Machine Learning

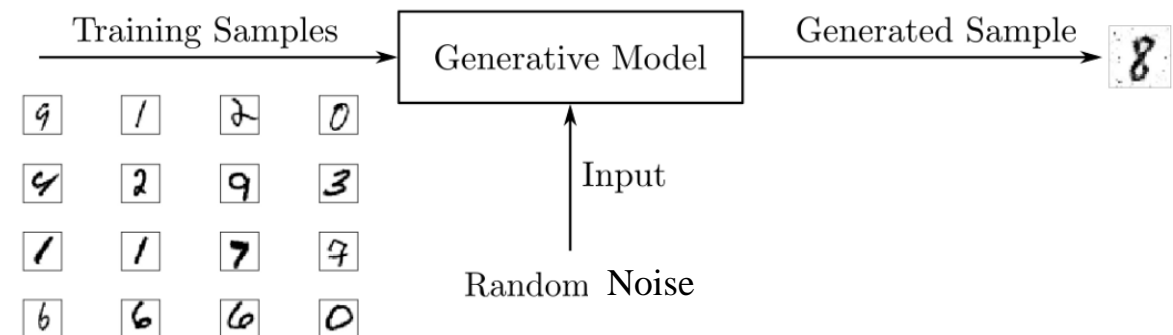


**Fundamental aim of a discriminative model**  
Learn a model of  $p(y|x)$  from observations



**Fundamental aim of a Generative Model**

Learn a model of  $p(x)$  or  $p(x|y)$  from observations to generate samples from random noise input



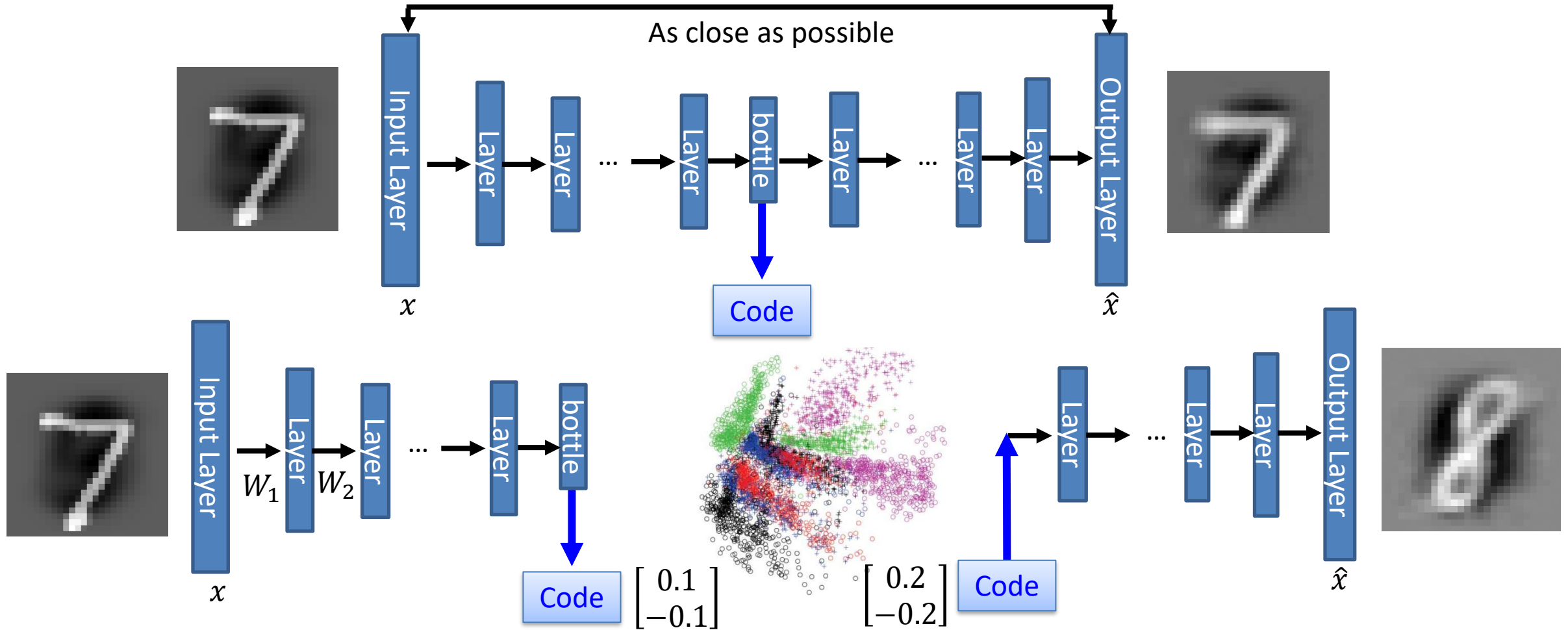
Lex Fridman. *Complete Statistical Theory of Learning (Vladimir Vapnik)* | MIT Deep Learning Series, 2020.  
<https://www.youtube.com/watch?v=Ow25mjFjSmg>.

# Generating data with machine learning

- Can we generate examples that follow the same distribution as a given set of examples using noise as input?
- Sampling from the multi-dimensional distribution of data
- How?
  - Density Modelling
    - Modelling the Probability of observing a given point  $p(x)$
    - Once I have an explicit or implicit  $p(x)$ , I can sample from that distribution to generate an example



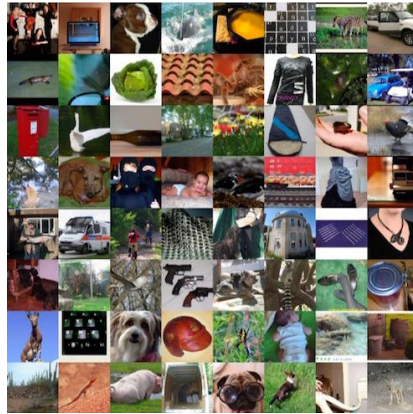
# Generating Data with Autoencoders



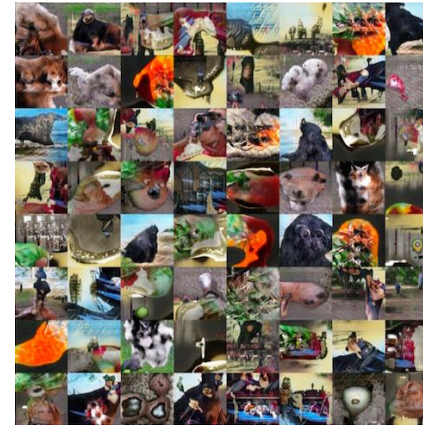
Tutorial Implementation: <https://github.com/foxtrotmike/CS909/blob/master/autoencoders.ipynb>

# Generative Models

- Can we build a model to approximate a data distribution from given examples?



Real image (training data)  $\sim p_{\text{data}}(x)$



Generated samples  $\sim p_{\text{model}}(x)$

Want to learn  $p_{\text{model}}(x)$  similar to  $p_{\text{data}}(x)$

**Density estimation:** a core problem in unsupervised learning

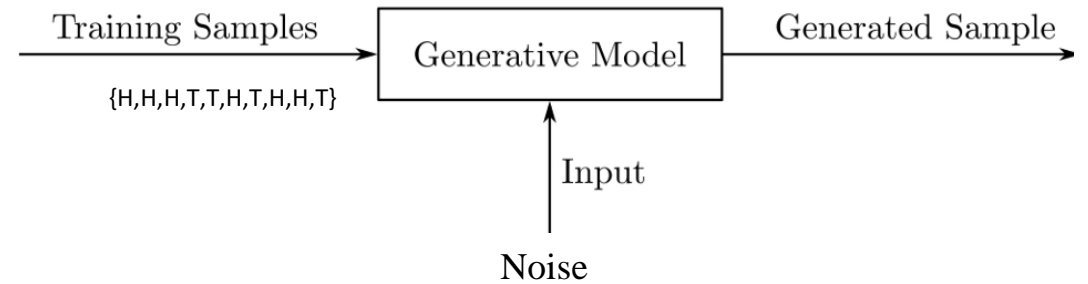
## Several flavors:

- **Explicit density estimation:** explicitly define and solve for  $p_{\text{model}}(x)$ 
  - Algorithms: Gaussian Mixture Models, Kernel Density Estimation, Variational Autoencoders
- **Implicit density estimation:** learn model that can sample from  $p_{\text{model}}(x)$  w/o explicitly defining it
  - Algorithms: Vanilla autoencoder, Generative adversarial networks (GANs), Diffusion Models, Normalizing Flows



# A Simple Generative Machine Learning Example

- Nature
  - A coin with  $p(x=H)=0.7$  and  $p(x=T)=0.3$
  - Generates data
- Given Data
  - $\{H,H,H,T,T,H,T,H,H,T\}$
- Goal of Generative Learning
  - Make a machine learning model that can generate data (heads or tails) that follows the same distribution as data from the real world or natural process.
  - The difference between the probability distributions of real and generated samples should be small



# REO for Generative Models

- Goal

- Given a set of real-world examples:  $x \sim p(x)$ .  $p(x)$  is not explicitly known.
- Learn parameters  $\theta$  of the model  $f(z; \theta)$  so that the examples generated by the model follow the same distribution as the real-world examples  $x \sim p(x)$

- Representation:  $x = f(z; \theta)$  with  $z \sim \text{Noise}$

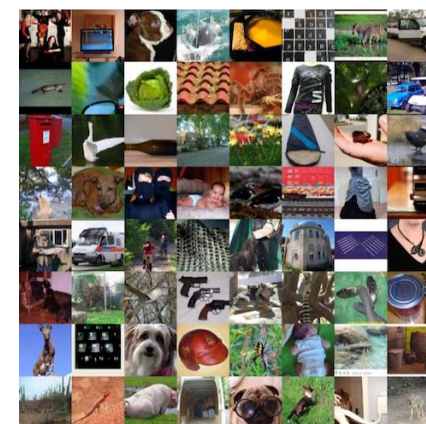
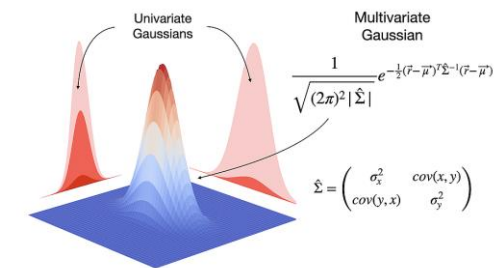
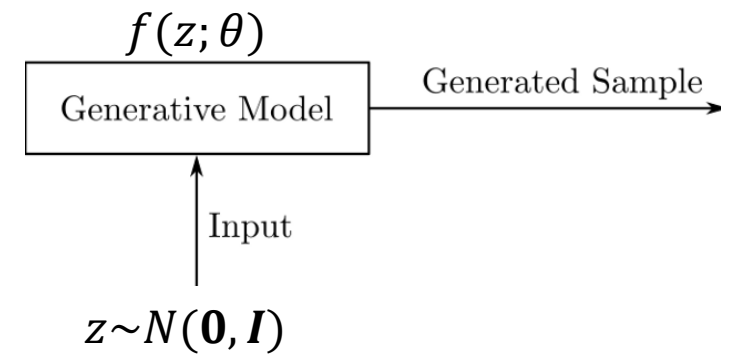
- Let's denote the distribution of examples generated by this model as  $p_\theta(x)$ .
- Note that the model may not have an explicit internal formula for this distribution.

- Evaluation:

- Differences between the probability distribution of  $x$  in nature  $p(x)$  and of the generated samples  $p_\theta(x)$  from  $f(z; \theta)$ 
  - That is, if I sample from  $p(x)$  or if I sample from  $p_\theta(x)$ , the real and generated samples are similar

- Optimization

- Use gradient descent to optimize for  $\theta$



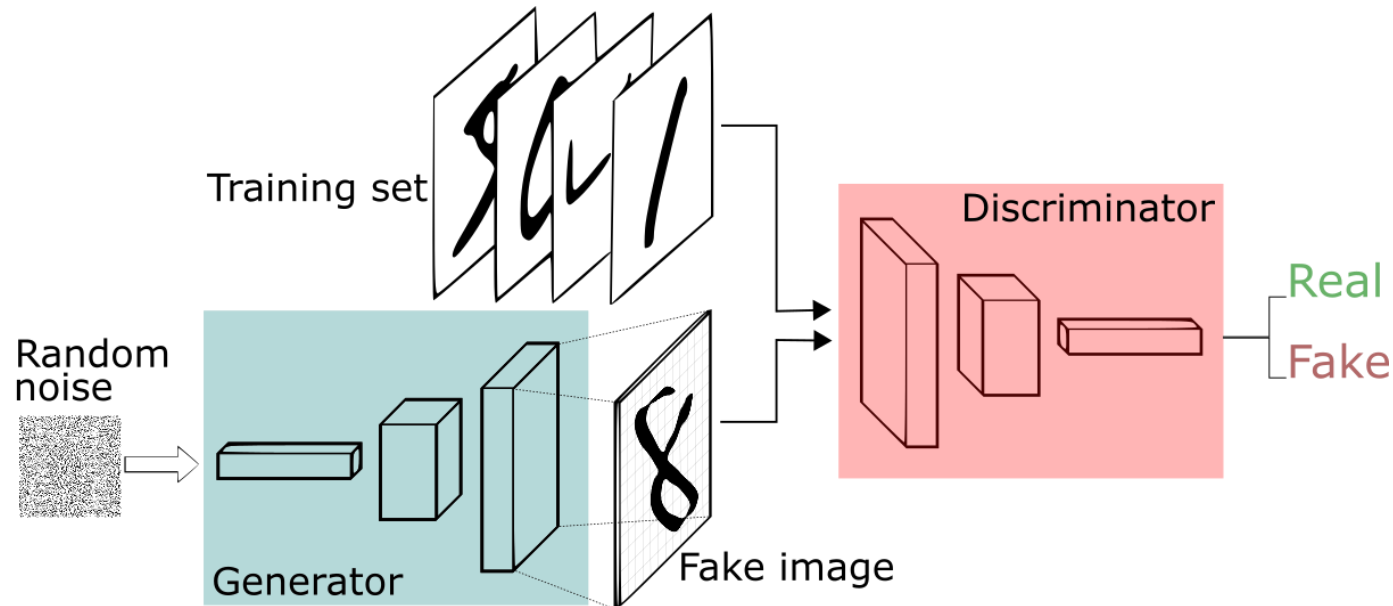
Real image (training data)  $\sim p(x)$   
 $p(x)$  is not given.



Generated samples  $\sim p_\theta(x)$   
 $p_\theta(x)$  may be implicit.

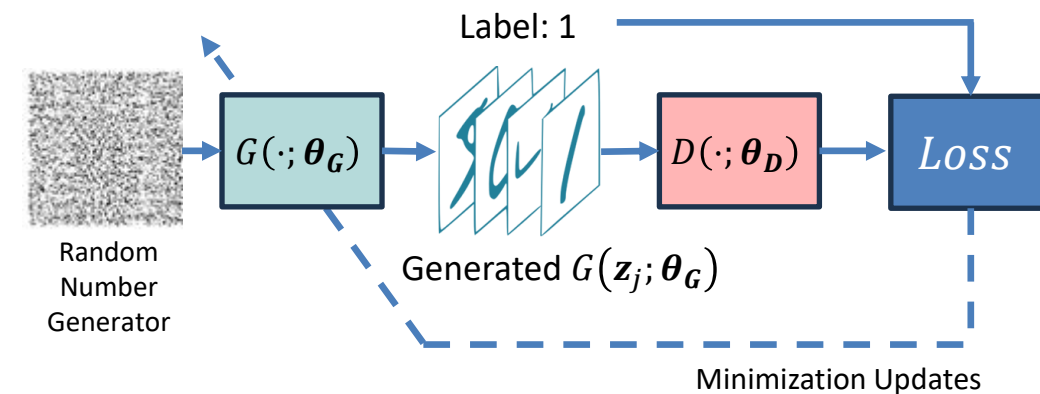
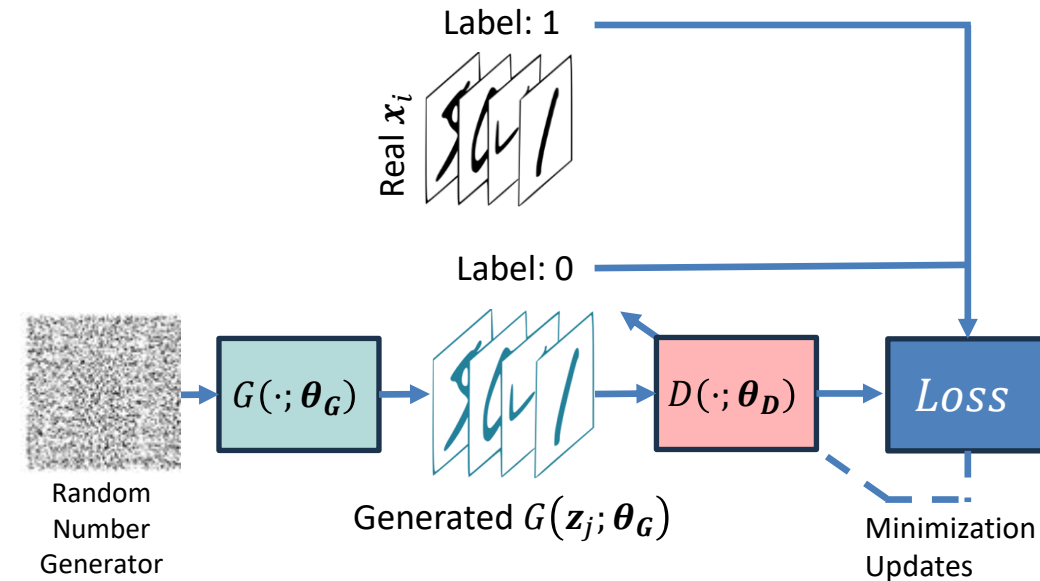
# Generative Adversarial Networks

- Use “Adversarial Training” to train a generator and discriminator simultaneously
- Generator: Generate samples from noise
- Discriminator: Detect “fake” or generated samples



# Adversarial Training in a GAN

- GAN Training the goal is to:
  - **Train the discriminator** to be good at detecting fakes
    - Simple classification: Discriminator should produce 1 for real and 0 for generated
    - $\min_{\theta_D} \sum_{x_i \in R} l(D(x_i; \theta_D), 1) + \sum_{z_j \sim N} l(D(G(z_j; \theta_G); \theta_D), 0)$
  - **Train the generator** to be so good that the discriminator labels generated samples as “Real”
    - The generator exploits the discriminator’s ability or knowledge to distinguish between real and generated samples to its advantage
    - The generator is optimized such that the discriminator produces 1 for generated examples
    - $\min_{\theta_G} \sum_{z_j \sim N} l(D(G(z_j; \theta_G); \theta_D), 1)$
    - OR equivalently, the generator is optimized such that the discriminator generates errors in classifying generated examples (note the max below)
    - $\max_{\theta_G} \sum_{z_j \sim N} l(D(G(z_j; \theta_G); \theta_D), 0)$
- Can also add additional loss terms for quality/realism etc.



# GAN Tutorial

 Open in Colab

## A Barebones GAN in PyTorch for generating coin flips

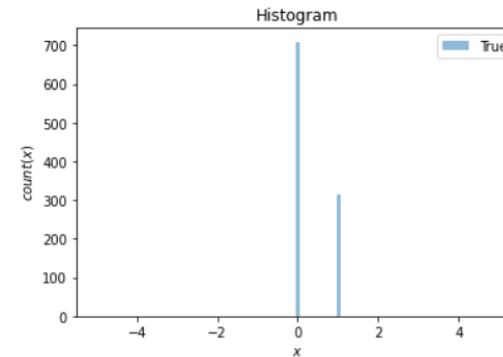
By Fayyaz Minhas

Let's consider a very simple coin toss as a process that generates coin flips with a probability of 0.3 of producing heads. We can describe the underlying probability distribution for this generative process (coin toss) as  $p(x)$  where  $x \in \{H = 1, T = 0\}$  is sampled from  $p(x)$ , i.e.,  $x \sim p(x)$ . We would like to use a Generative Adversarial Network (GAN) to model this process using a number of data samples or observations from the original process for training. Specifically, we would like to have a GAN with such a generator that you (and its discriminator) wouldn't be able to tell if a series of coin tosses has been generated using the GAN or the underlying true process! In more mathematical terms, we would like to train a generative model  $x = G(z; \theta_G)$  that can generate samples  $x$  using Normally distributed random input ( $z \sim \mathcal{N}(0, 1)$ ) such that the probability distribution of these generated samples  $p_G(x)$  is close to  $p(x)$  without knowing  $p(x)$  in advance or explicitly modelling  $p_G(x)$ .

Using a GAN is an overkill for this simple task and there are much simpler and more effective ways of modelling this simple problem. However, this GAN based solution is intended to help you understand how GANs can model complex densities implicitly and can be used to generate samples that mimic the true or natural generative process.

We first simulate the coin toss and generate 1024 training samples below. The histogram shows the (sample estimate of) the true density.

```
]: """  
A toy GAN to generate coin tosses  
"""  
  
# Let's model the natural density and generate some data using that  
  
import torch  
from torch import nn  
  
import math  
import matplotlib.pyplot as plt  
import numpy as np  
train_data_length = 1024  
def cointoss(t):  
    phead = 0.3  
    return 1.0*(t
```

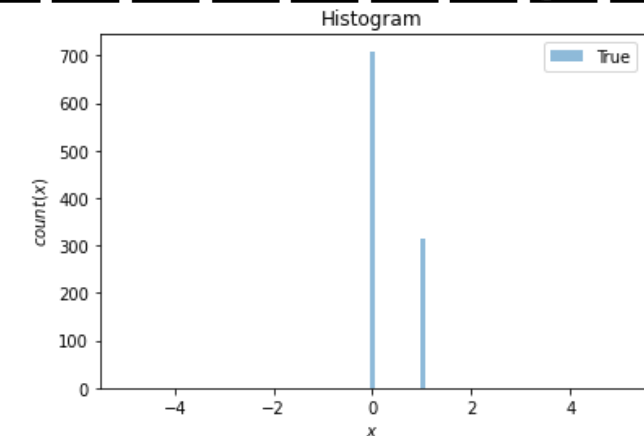
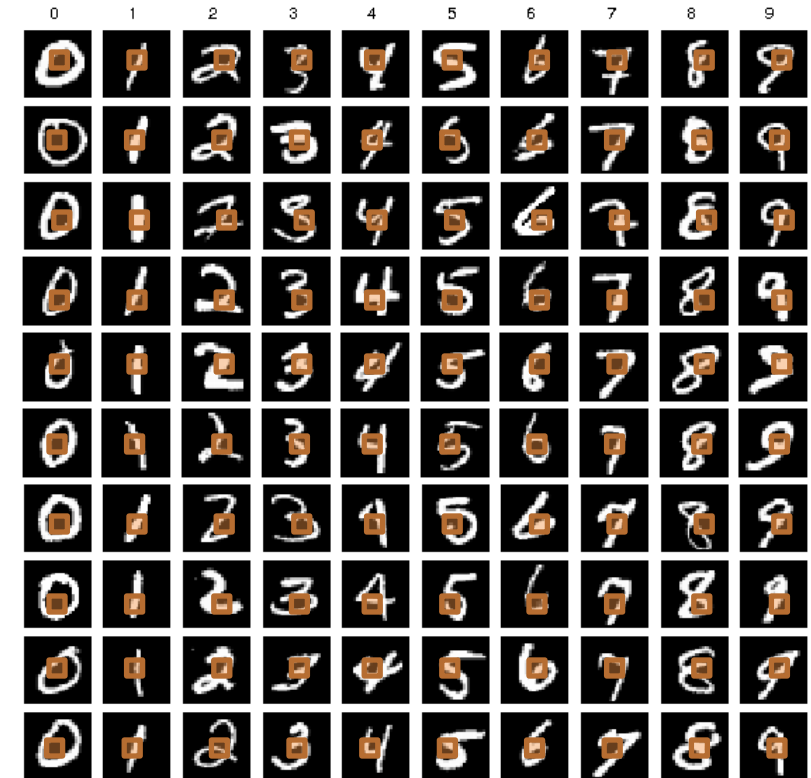


Example Data tensor([[0., 0., 0., ..., 0., 1., 1.]])

<https://github.com/foxtrotmike/CS909/blob/master/simpleGAN.ipynb>

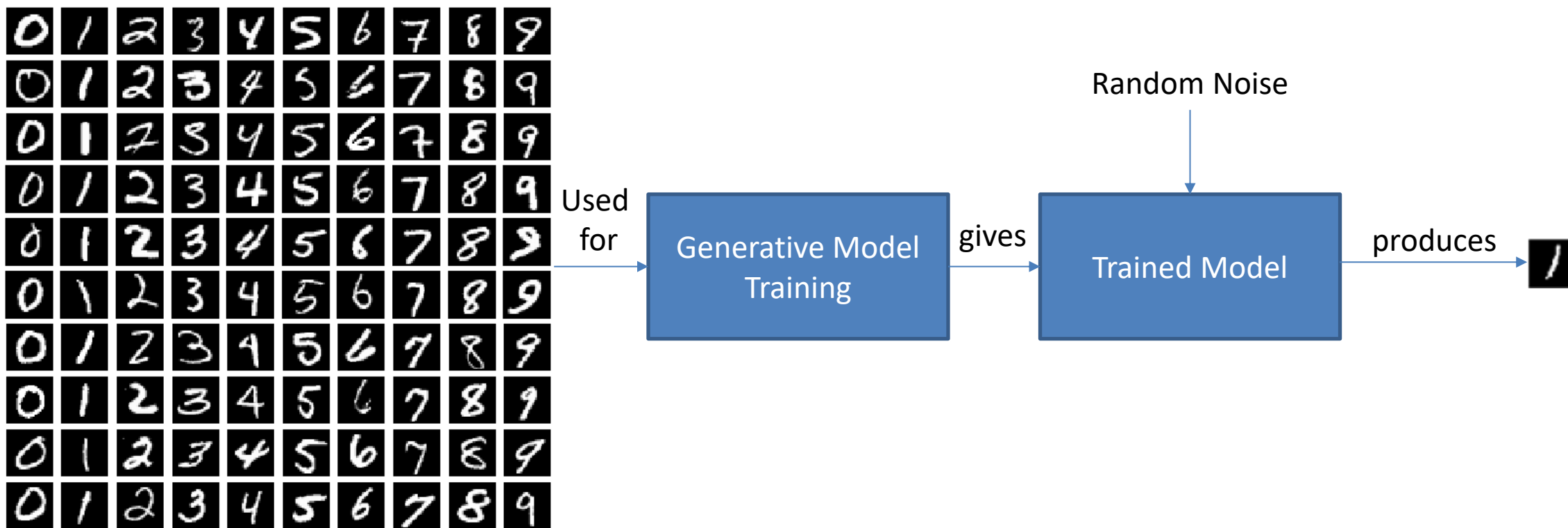
# How to go from generating coin flips to images?

- Assume you are given B&W images for training a GAN to generate more images like that.
- Let's look at a single pixel location in each image
  - We have a distribution of pixel values across all images at that location
    - We would like our GAN to generate data according to that distribution at that pixel location
    - Naïve idea: Have multiple GANs – one for each pixel location
      - Assumes each pixel is independent of the other
      - Computationally intensive
    - We can train a single GAN to generate a multi-dimensional probability distribution by using a multi-output generator.



# Unconditional vs Conditional Generation

- Unconditional Generative Modelling
  - Simple model the probability distribution of the data  $p(x)$ 
    - Example: Generating images without paying any regard to the digit

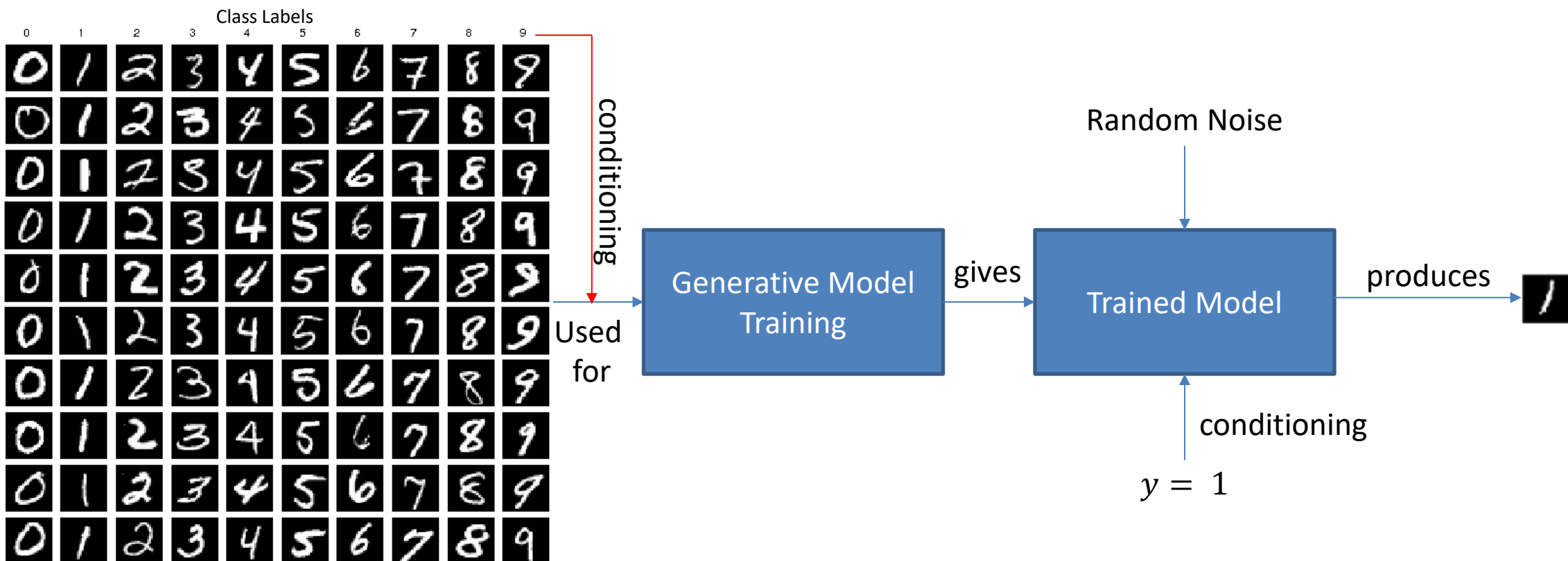


# Unconditional vs Conditional Generation

- Conditional Generative Modelling

- Model the distribution  $p(x|y)$  of data  $x$  conditioned on a variable  $y$

- Example: Generating images for a given digit





# GANs Applications

- GANs have some impressive applications
  - Synthetic Image Generation
  - Speech Generation
  - Image to Image Translation
  - Style Transfer
  - Deep Fakes

Barebones GAN

<https://github.com/foxtrotmike/CS909/blob/master/simpleGAN.ipynb>

Raevskiy, Mikhail. "Write Your First Generative Adversarial Network Model on PyTorch." Medium, August 31, 2020.

<https://medium.com/dev-genius/write-your-first-generative-adversarial-network-model-on-pytorch-7dc0c7c892c7>.



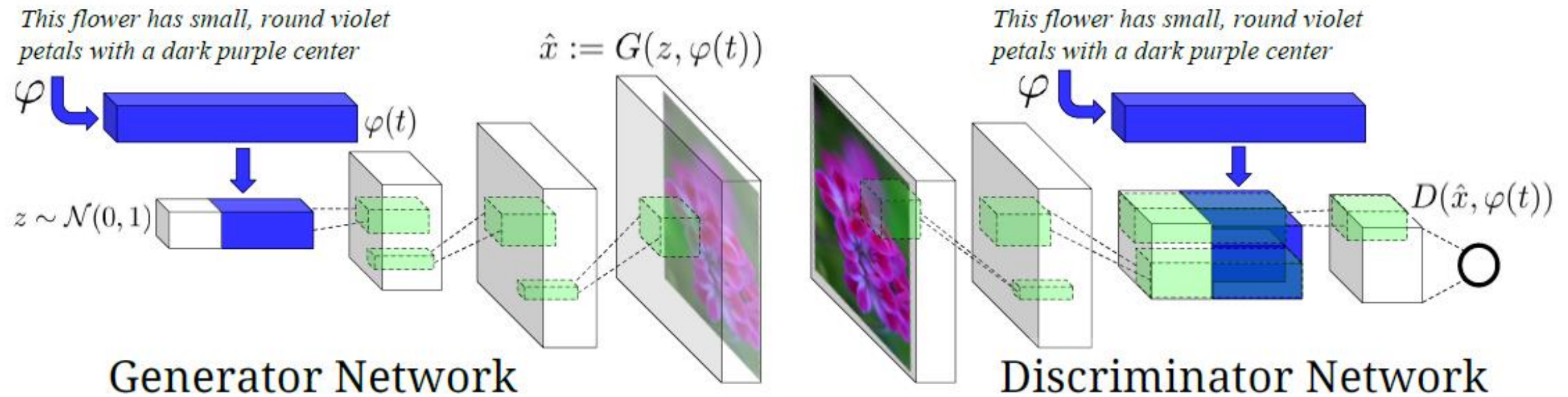
Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. <https://arxiv.org/abs/1701.00160>  
<https://github.com/eriklindernoren/PyTorch-GAN>  
<https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>  
<https://affinelayer.com/pixsrv/>

# The GAN Zoo

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks




<https://github.com/hindupuravinash/the-gan-zoo>

# Text-to-Image Synthesis

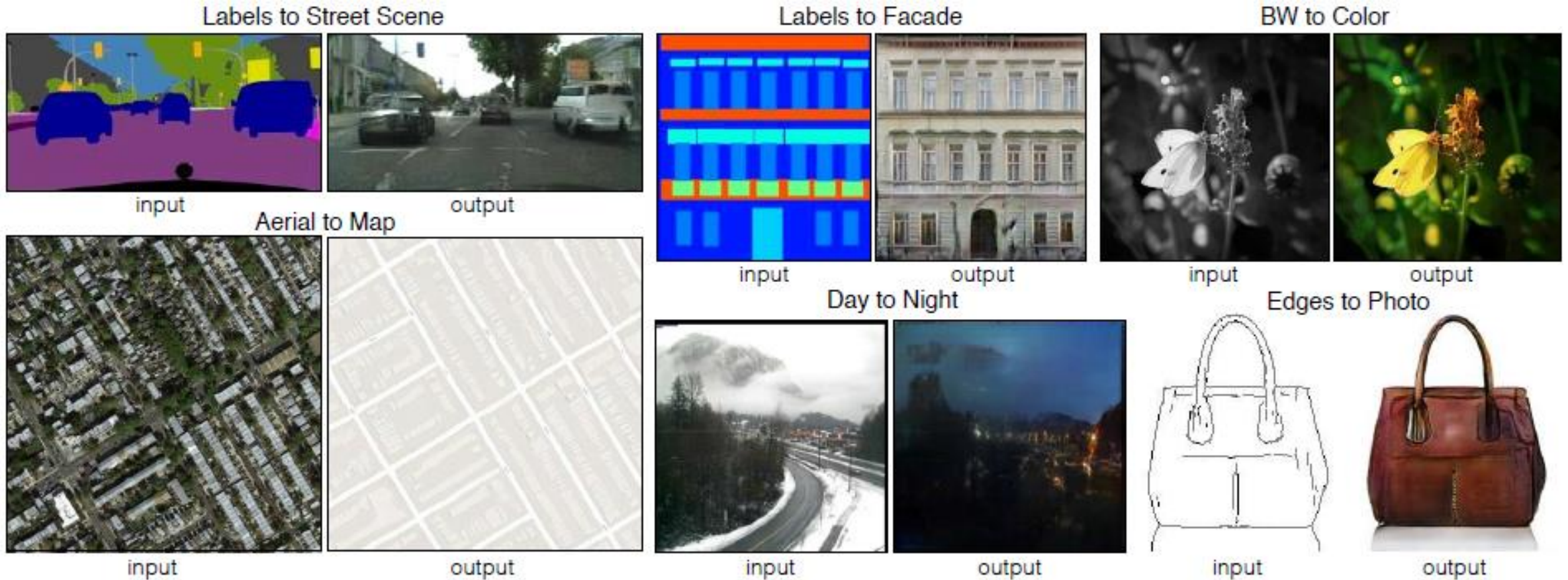


- S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, H. Lee, "Generative Adversarial Text-to-Image Synthesis", ICML 2016
- H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, D. Metaxas, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks", arXiv preprint, 2016
- S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, H. Lee, "Learning What and Where to Draw", NIPS 2016

# Text to Image – Results

Caption	Image
a pitcher is about to throw the ball to the batter	
a group of people on skis stand in the snow	
a man in a wet suit riding a surfboard on a wave	

# Image-to-image Translation



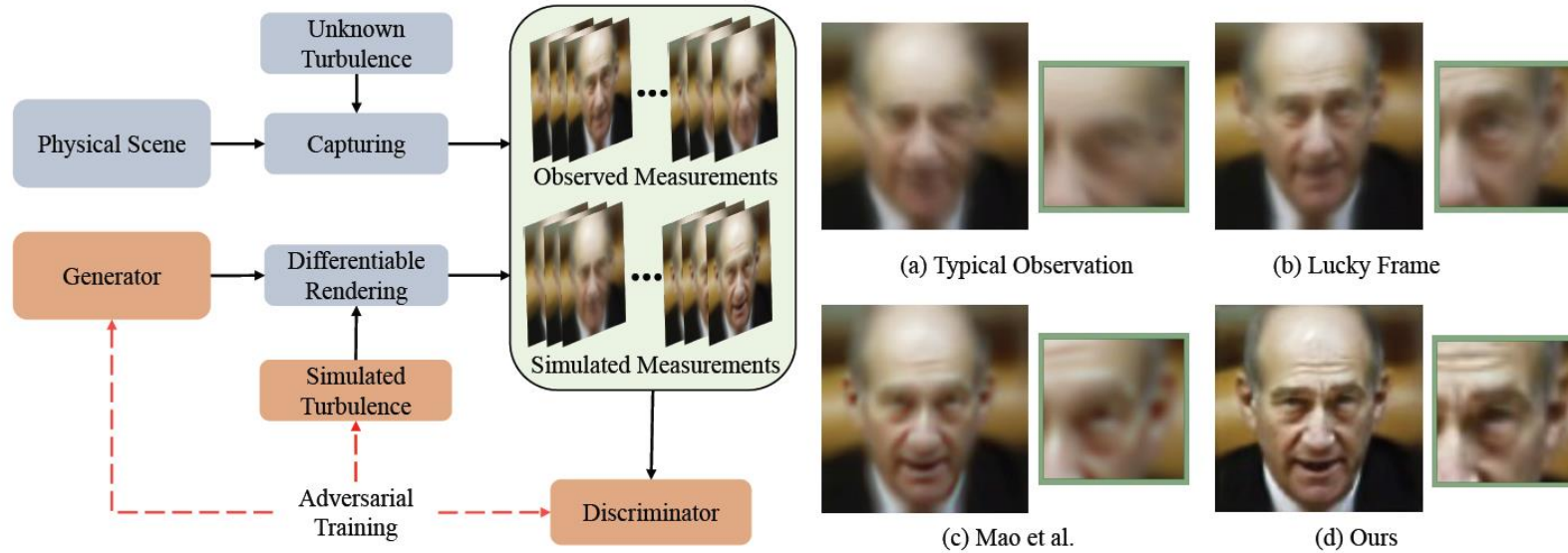
P. Isola, J.-Y. Zhu, T. Zhou, A.A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks", arXiv preprint, 2016

# Unpaired Transformation – Cycle GAN, Disco GAN

Transform an object from one domain to another without paired data



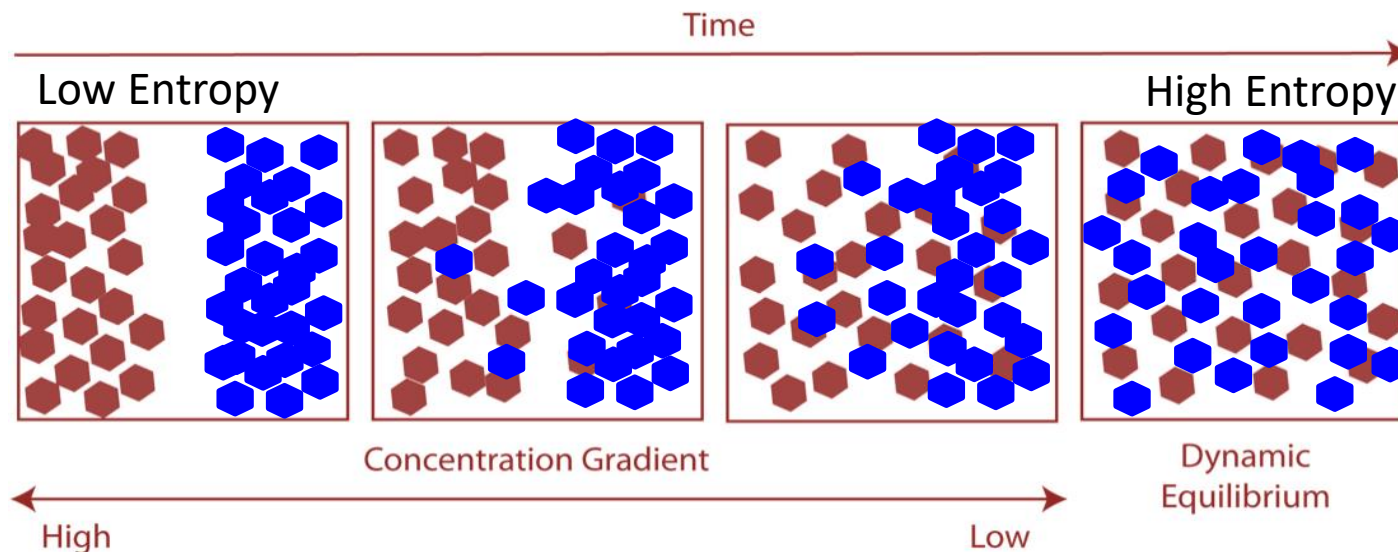
# TurbuGAN



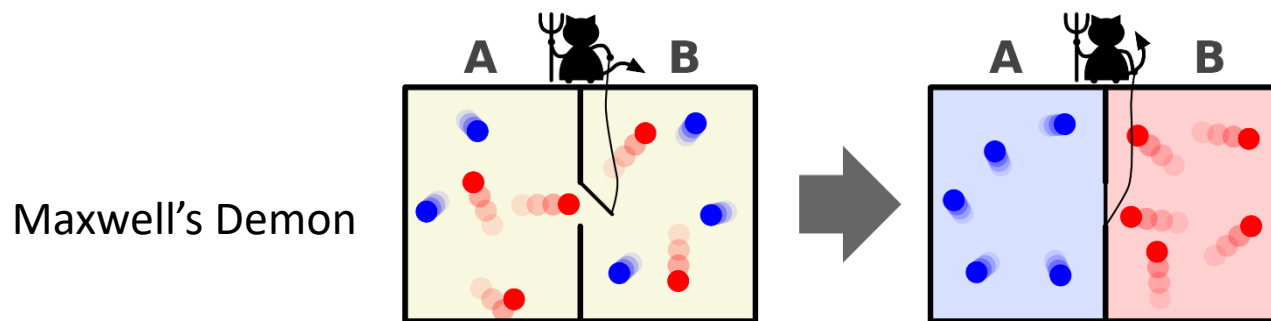
Feng, Brandon Yushan, Mingyang Xie, and Christopher A. Metzler. "TurbuGAN: An Adversarial Learning Approach to Spatially-Varying Multiframe Blind Deconvolution with Applications to Imaging Through Turbulence." arXiv, August 22, 2022. <https://doi.org/10.48550/arXiv.2203.06764>.

# Diffusion Models

- What is diffusion?

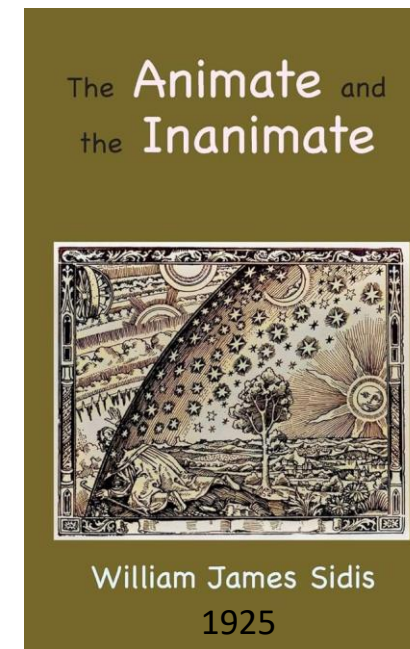


- Can we learn to reverse it?



Maxwell's Demon

[https://en.wikipedia.org/wiki/Maxwell%27s\\_demon](https://en.wikipedia.org/wiki/Maxwell%27s_demon)



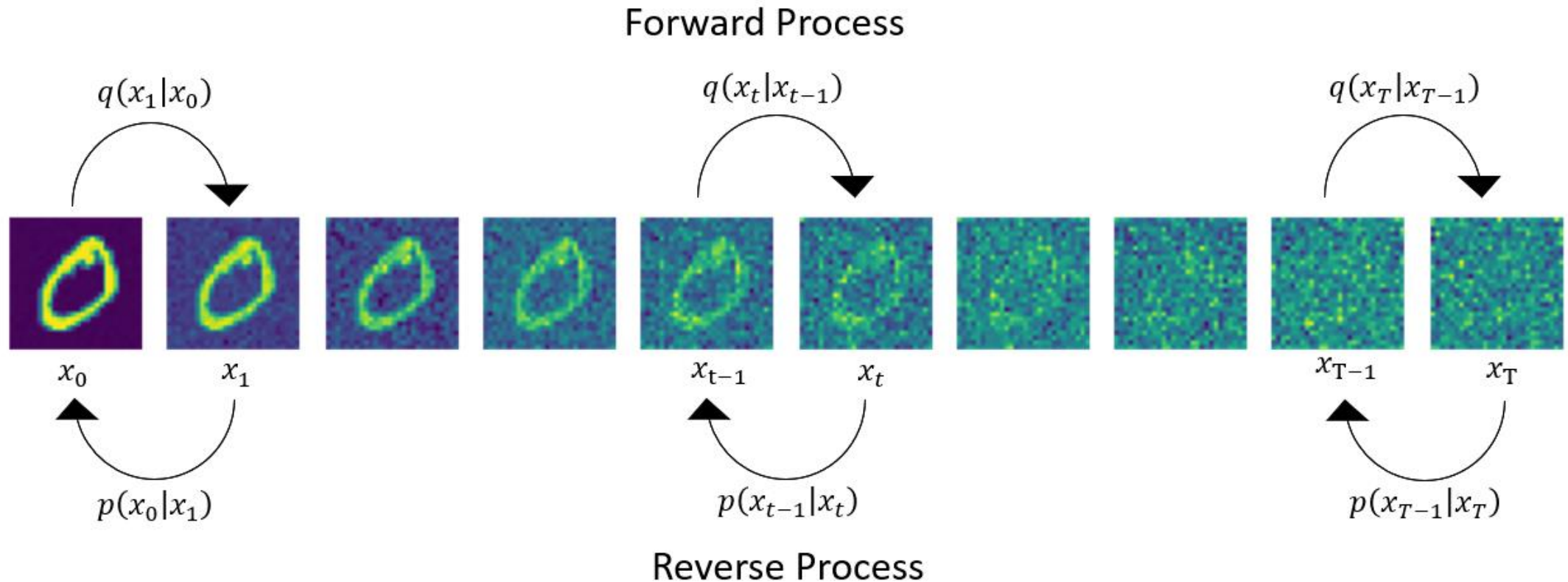
PREFACE

This work sets forth a theory which is speculative in nature, there being no verifying experiments. It is based on the idea of the reversibility of everything in time; that is, that every type of process has its time-image, a corresponding process which is its exact reverse with respect to time. This accounts for all physical laws but one, namely, the second law of thermodynamics. This law has been found during the nineteenth century to be a source of a great deal of difficulty. The eminent physicist, Clerk-Maxwell, in the middle of the nineteenth century, while giving a proof of that law, admitted that reversals are possible by imagining a "sorting demon" who could sort out the smaller particles, and separate the slower ones from the faster ones. This second law of thermodynamics brought in the idea of energy-level, of unavailable energy (or "entropy" as it was called by Clausius) which was constantly increasing.



# Diffusion Models

- Main idea: Learn to reverse a “diffusion” process



Tutorial: <https://github.com/wgrgwrght/Simple-Diffusion/blob/main/SimpleDiffusion.ipynb>

Dhariwal, Prafulla, and Alex Nichol. “Diffusion Models Beat GANs on Image Synthesis.” arXiv, June 1, 2021. <https://doi.org/10.48550/arXiv.2105.05233>.

Nichol, Alex, and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models.” arXiv, February 18, 2021. <https://doi.org/10.48550/arXiv.2102.09672>.

# Diffusion Models

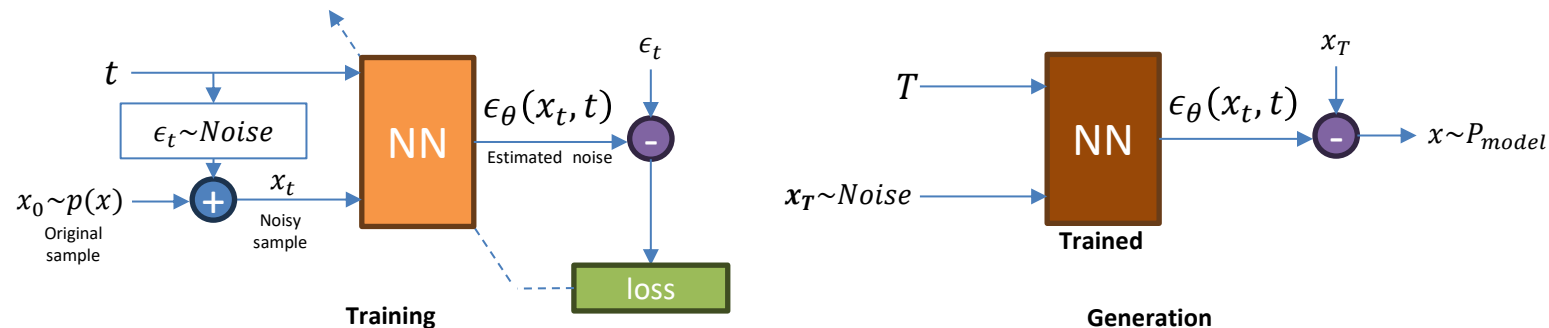
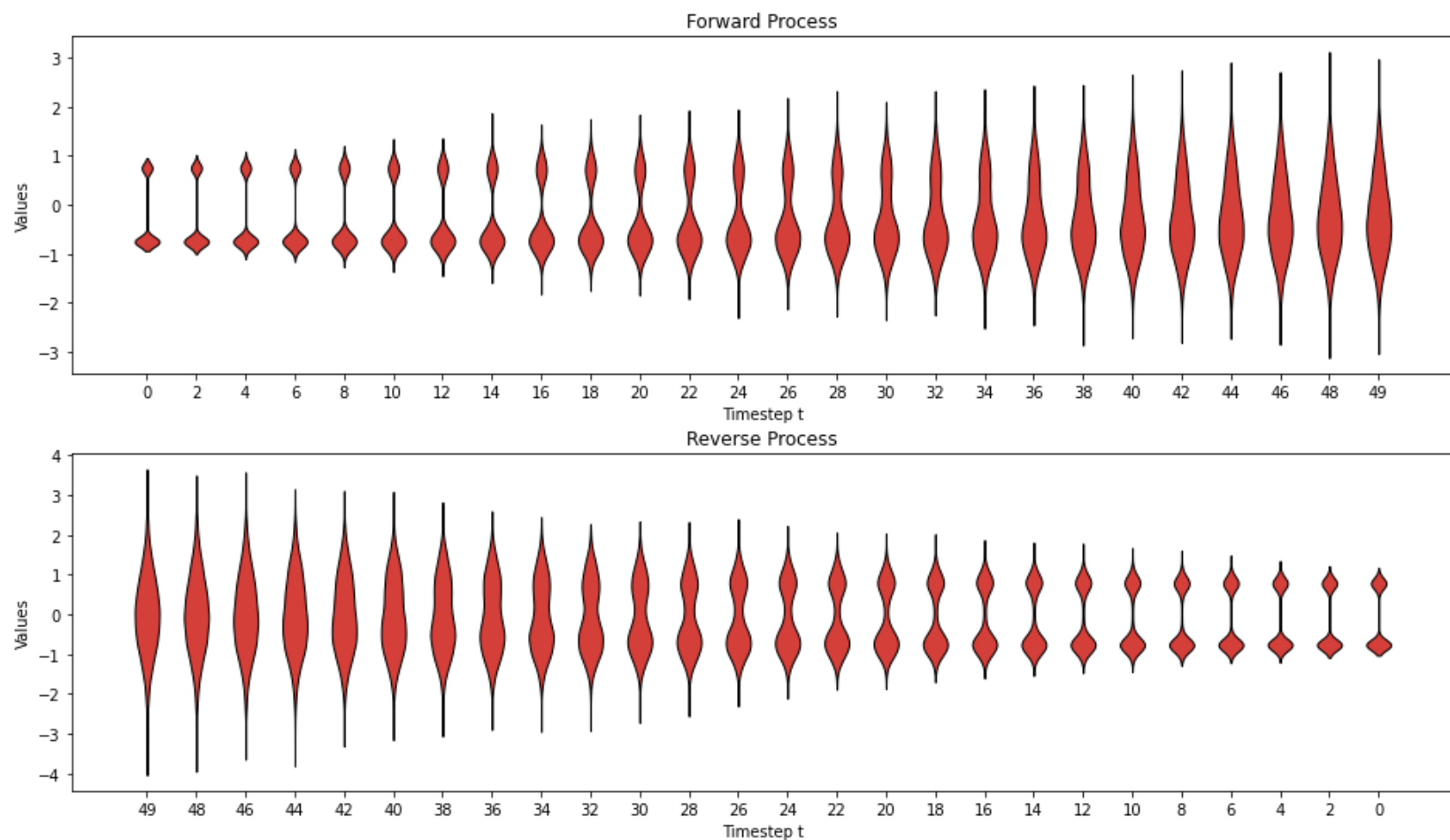
- **Generation by learning to reverse entropy**
- **Forward Process:** Generate noisy signals from data
  - Data distribution gets gradually converted to noise
- **Reverse Process:** Learn to denoise
  - Using a neural network  $\epsilon_{\theta}(x_t, t)$  with weights  $\theta$  which takes the noisy data  $x_t$  as input along with the time step  $t$  (and possibly other "conditioning" variables) to output an estimate of the noise  $\epsilon_t$  that has been added to  $x_0$  to generate  $x_t$ . This is achieved by solving the following optimization problem:

$$\min_{\theta} E_{t, x_0 \in} |\epsilon_t - \epsilon_{\theta}(x_t, t)|^2$$

- **Generation:** Once the neural network is trained, we can generate data using:

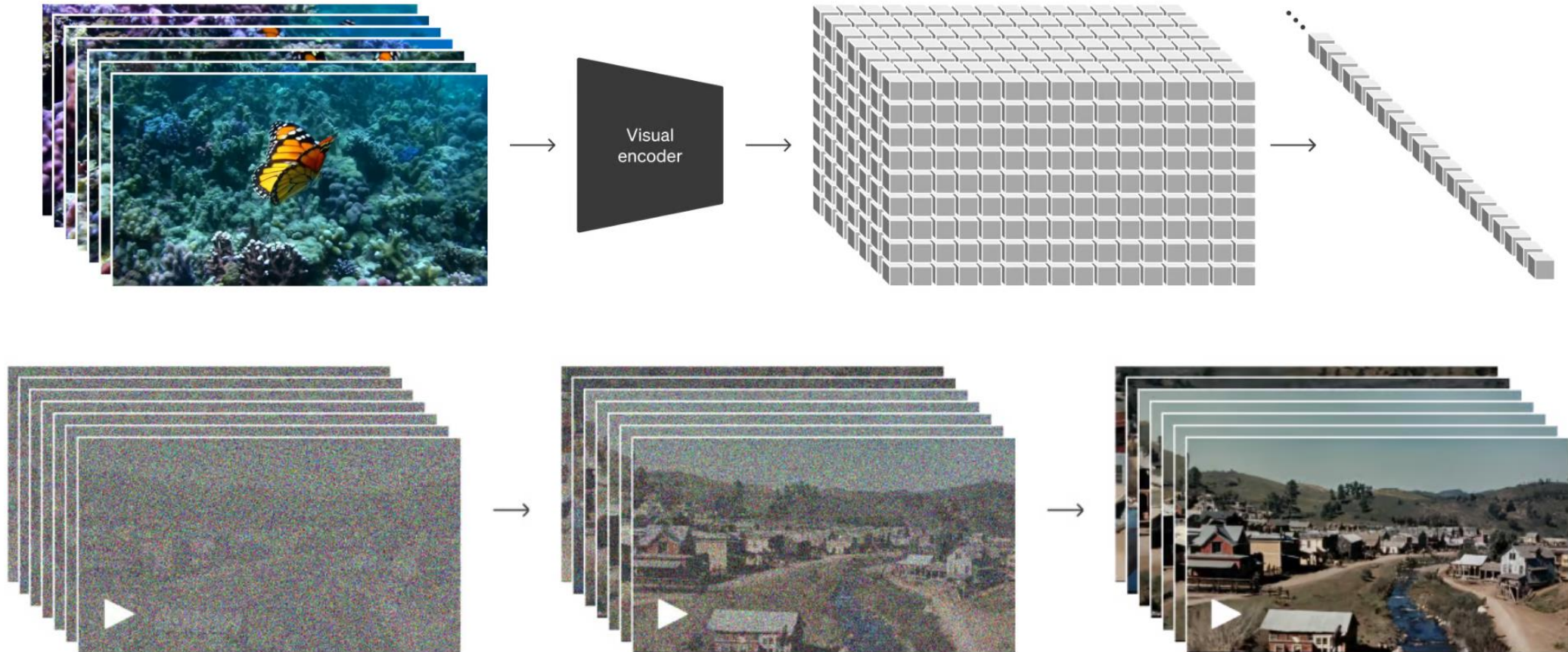
$$x = x_T - \epsilon_{\theta}(x_T, T) \text{ with } x_T \sim N(0, 1)$$

- Can be improved by operating in a compressed or latent space: **Latent diffusion**



Simplest Diffusion Tutorial: <https://github.com/wgrgwrgt/Simple-Diffusion/blob/main/SimpleDiffusion.ipynb>

# SORA: Diffusion Transformer



<https://openai.com/research/video-generation-models-as-world-simulators>

# Application of Generative Modelling