Building Generative Adversarial Networks in PyTorch

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understand Generative Adversarial Networks (GANs)

Candidate generation and evaluation

Role of generator

Role of discriminator

Generate handwritten digits using GANs trained on the MNIST dataset

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)

A new type of ML system (invented in 2014) that rely on two neural networks contesting in a game. Can be used to generate realistic images and videos of virtually anything.

Generative Adversarial Networks (GANs)

A new type of ML system (invented in 2014) that rely on two neural networks contesting in a game. Can be used to generate realistic images and videos of virtually anything.

Generative Adversarial Networks (GANs)

A new type of ML system (invented in 2014) that rely on two neural networks contesting in a game. Can be used to generate realistic images and videos of virtually anything.

Applications of GANs

AI-generated art

Human image synthesis

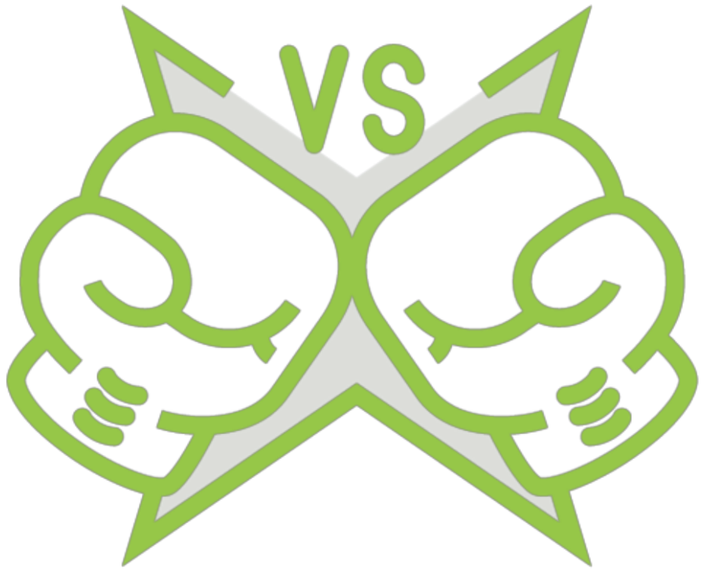
3D models from 2D images

Improve astronomical image

Traditional classification

Traditional regression

GANs



Considered part of unsupervised learning

Developed in 2014 by Ian Goodfellow et al

Two distinct contesting neural networks

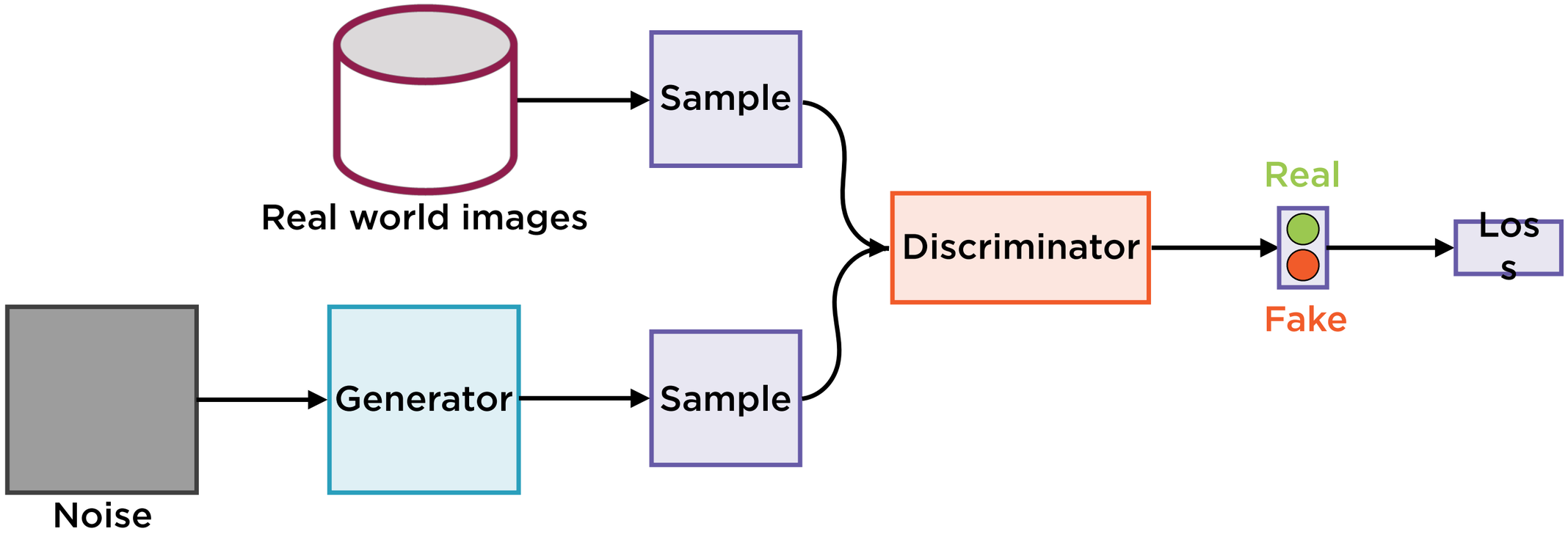
- Generative network
- Discriminative network

Two Neural Networks

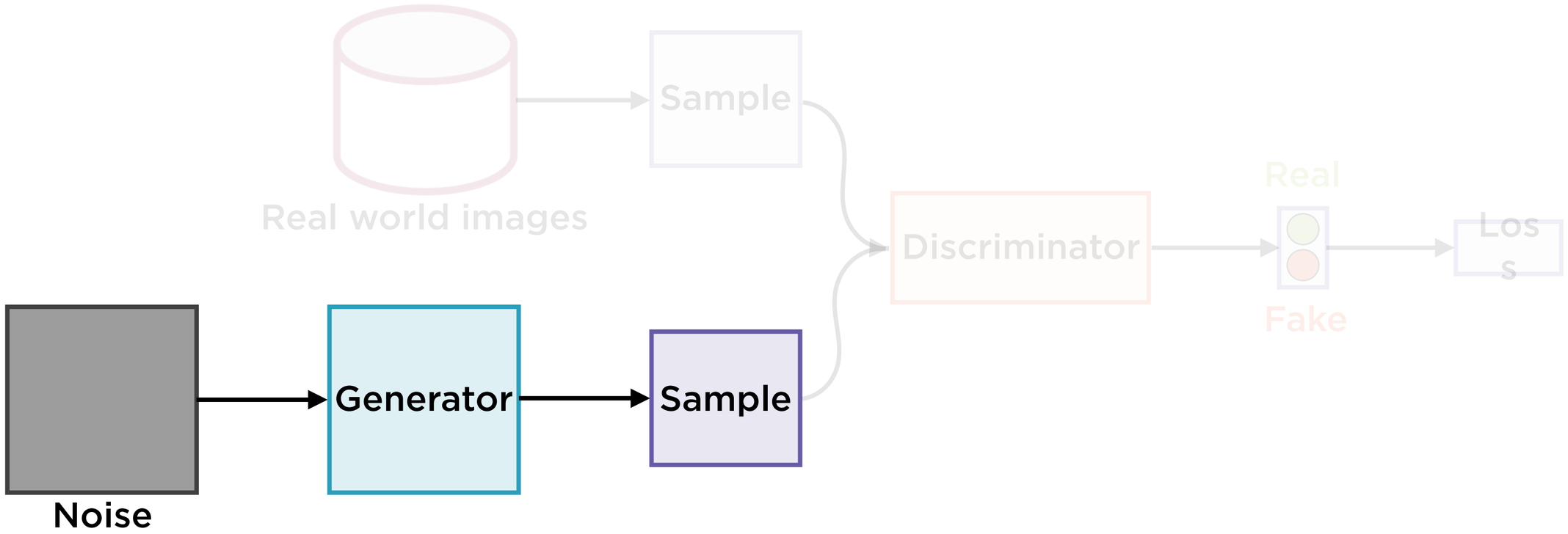
Generative Network
generates candidates

Discriminative Network
evaluates candidates

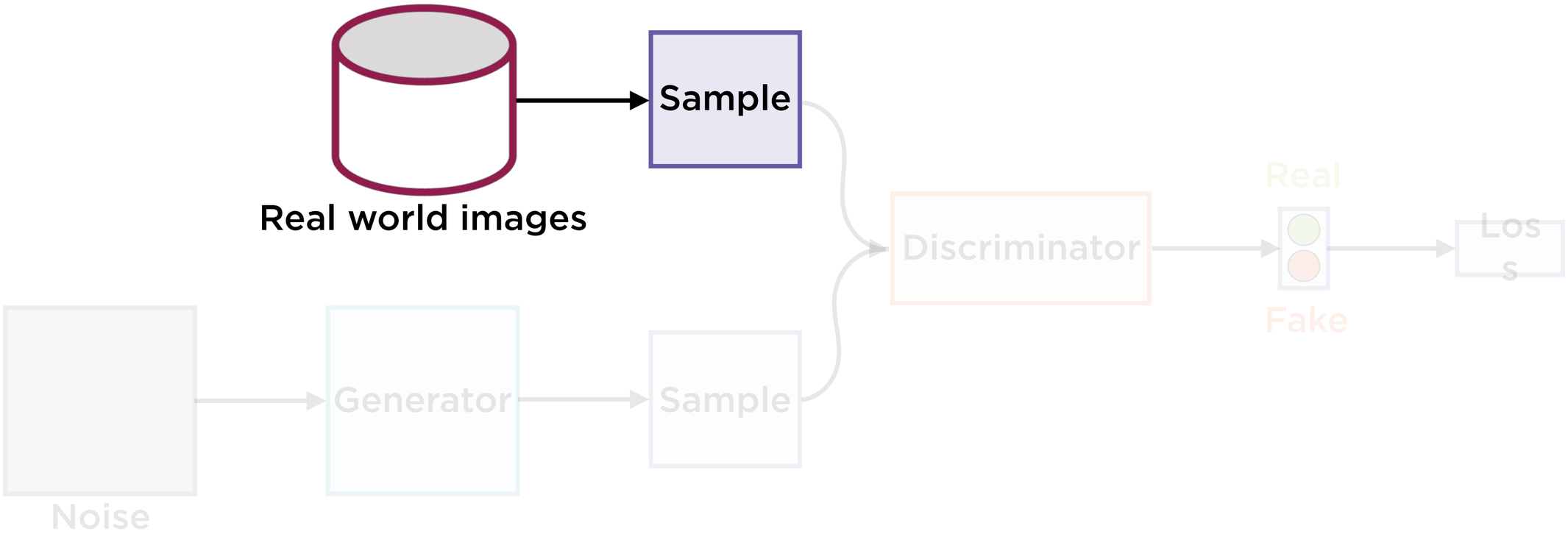
GANs


Real world images
Sample
Noise
Generator
Sample
Discriminator
Real
Fake
Los
s

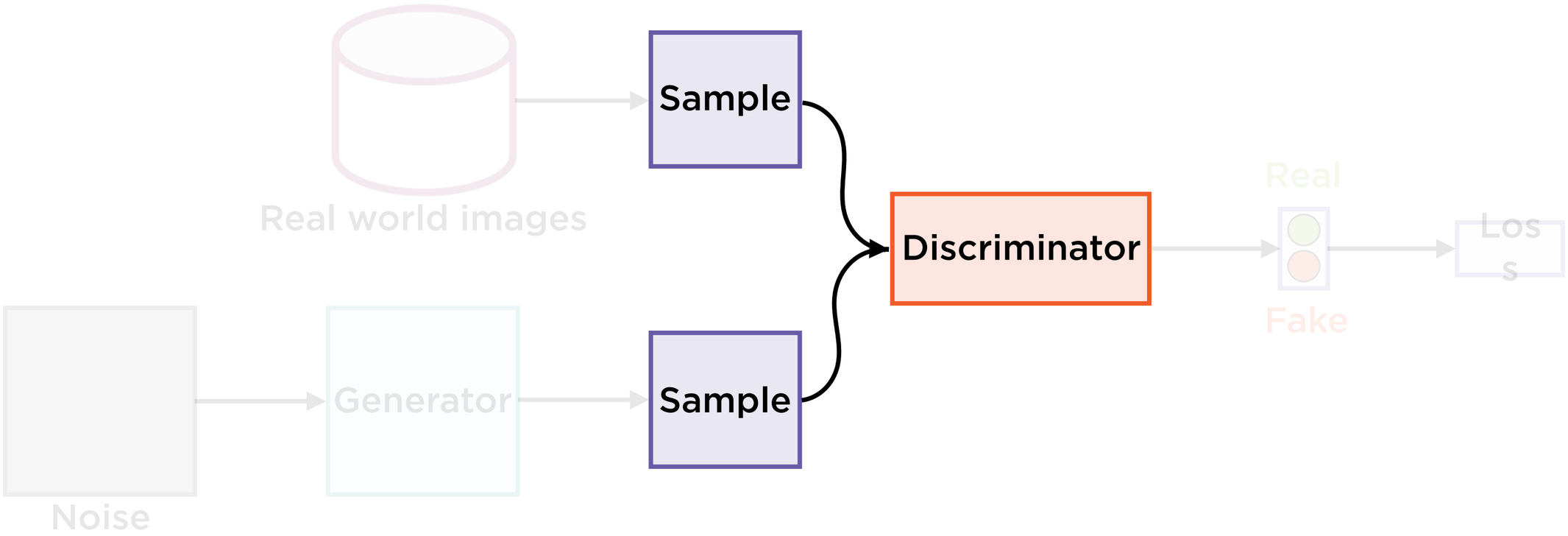
GANs


Real world images
Sample
Discriminator
Real
Fake
Los
s
Noise
Generator
Sample

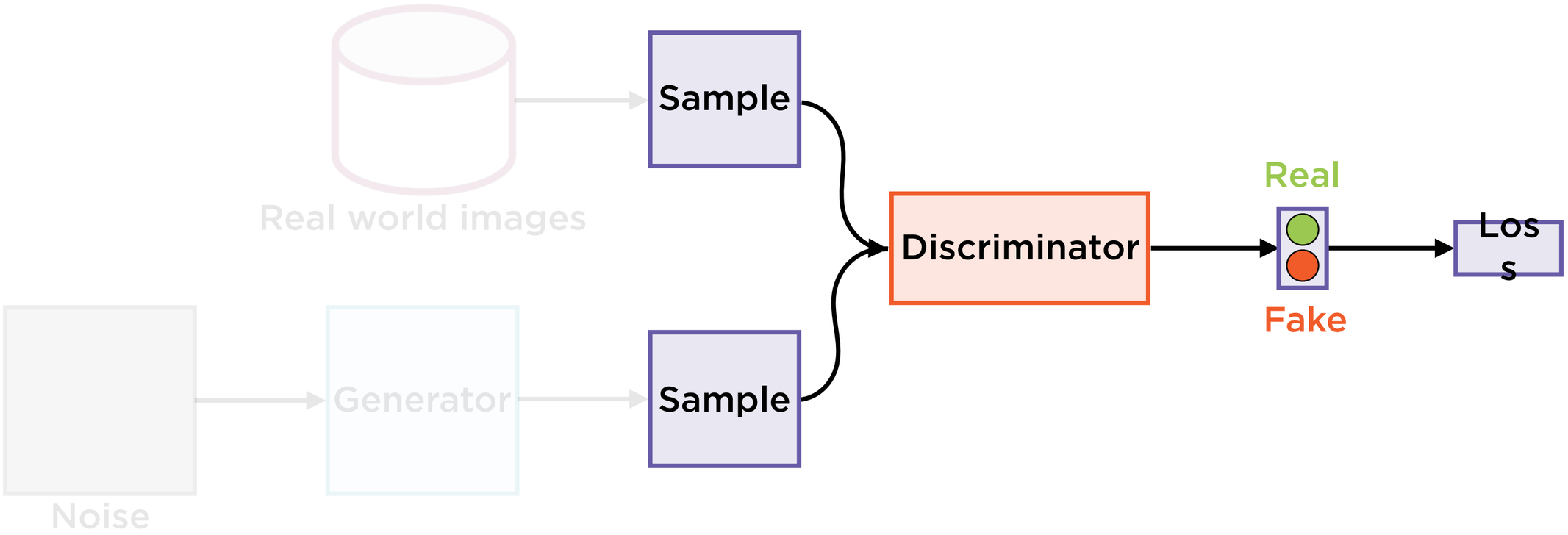
GANs


Real world images
Sample
Discriminator
Real
Fake
Los
s
Noise
Generator
Sample

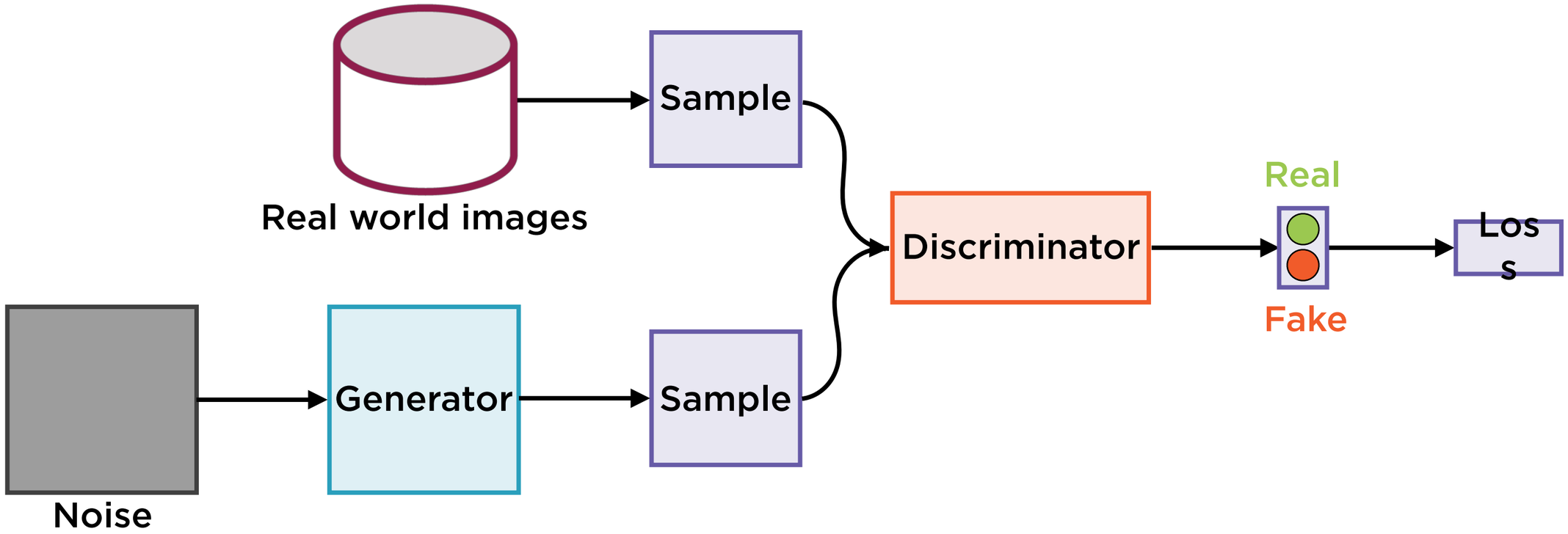
GANs


Real world images
Sample
Noise
Generator
Sample
Discriminator
Real
Fake
Los
s

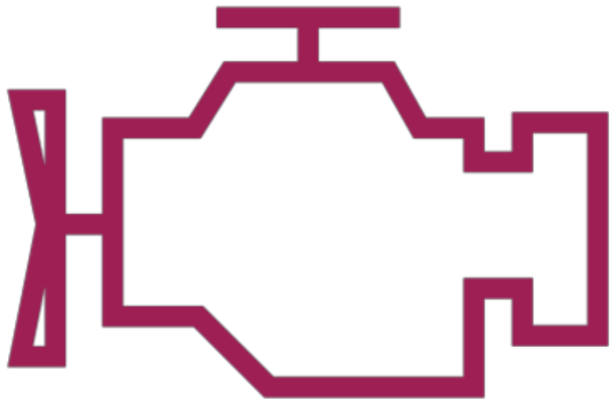
GANs


Real world images
Sample
Noise
Generator
Sample
Discriminator
Real
Fake
Los
s

GANs


Real world images
Sample
Noise
Generator
Sample
Discriminator
Real
Fake
Los
s

Generator

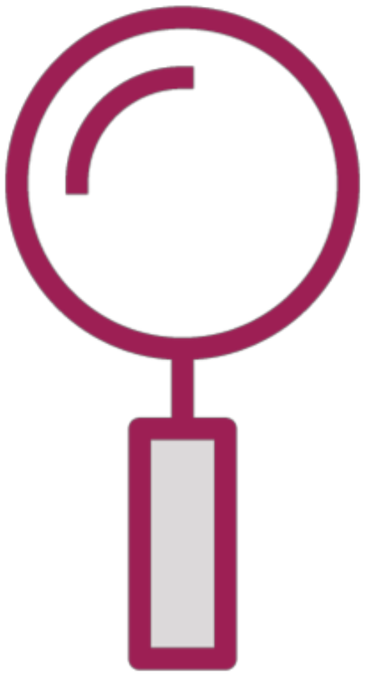


Generates data as realistically as possible

Trained to generate data similar to corpus

Seeks to fool discriminator

Discriminator

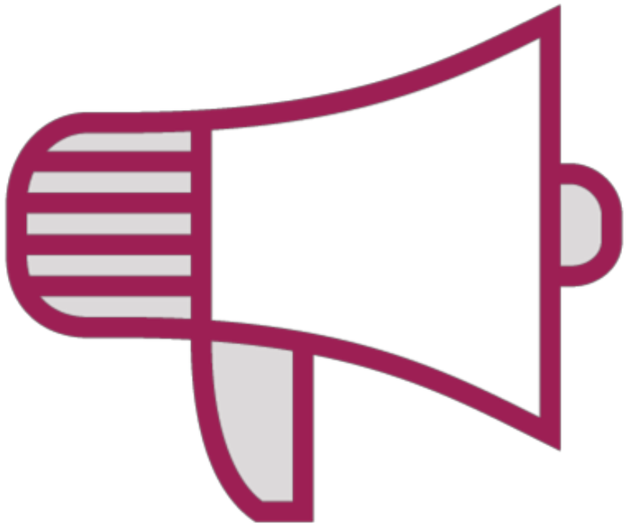


Generates probability that data is genuine

Classifies output of generator

Just like traditional classifier

Noise in GANs



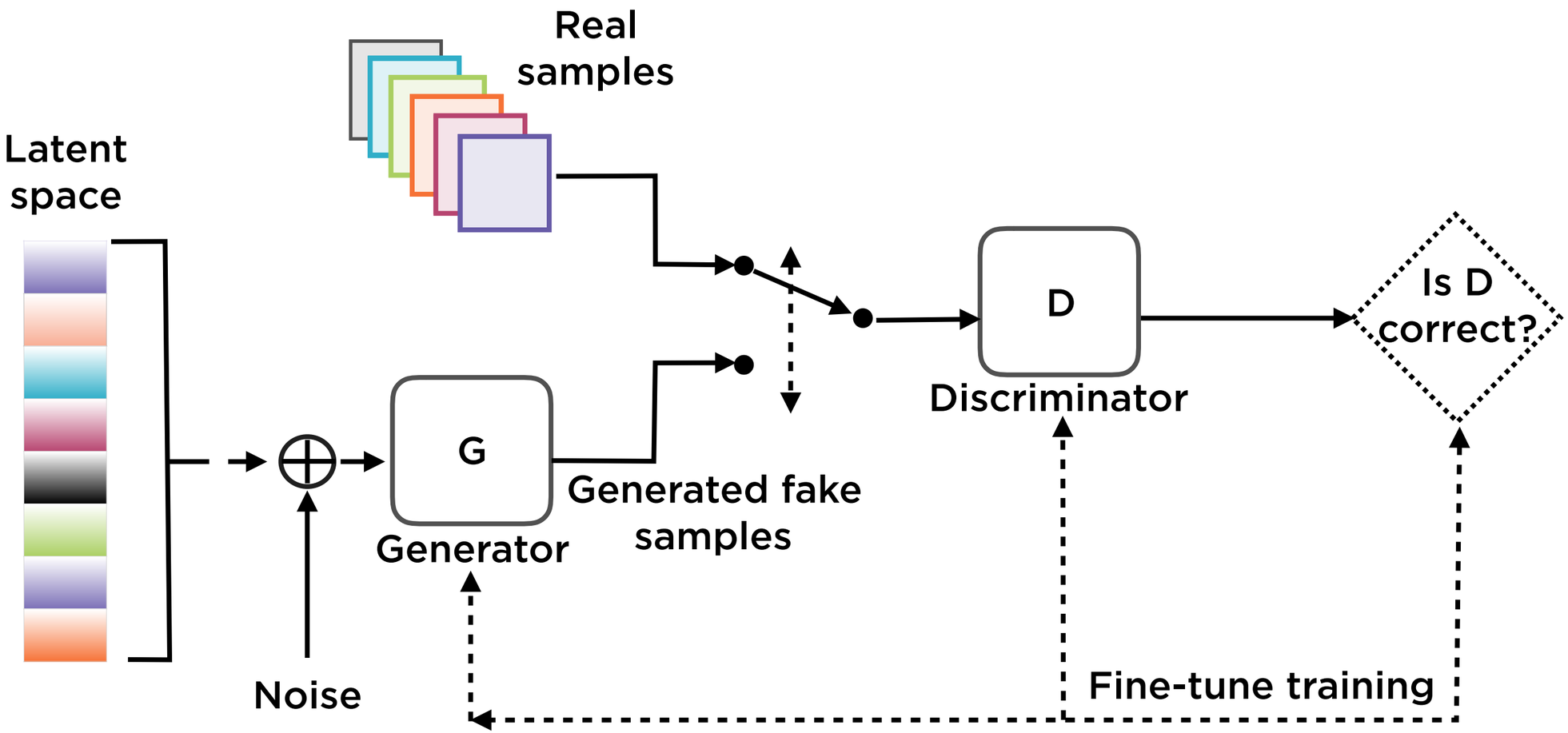
Requires function that generates noise

Create corpus of

- Real data points
- Noise function

Training a GAN

GANs


Real
samples
Latent
space
D
Is D
correct?
Discriminator
G
Generated fake
samples
Generator
Noise
Fine-tune training

Training a GAN

Start with corpus of real points as well as noise

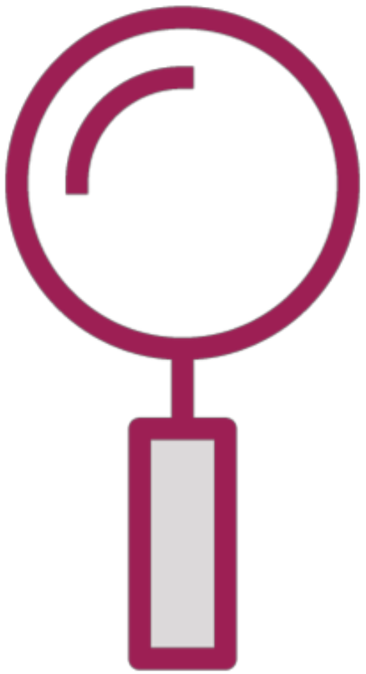
Train discriminator to tell them apart

Generate new noise points

Train generator to produce data that fools the discriminator

Repeat using optimizer

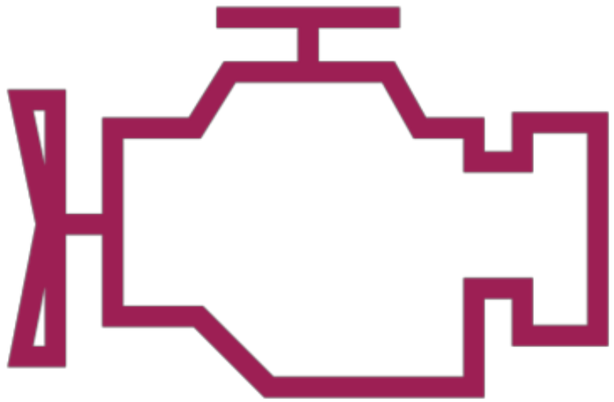
Discriminator



Maximizes probability of real data being classified as real

Minimizes probability of fake data being classified as real

Generator



Maximizes probability of fake data being classified as real

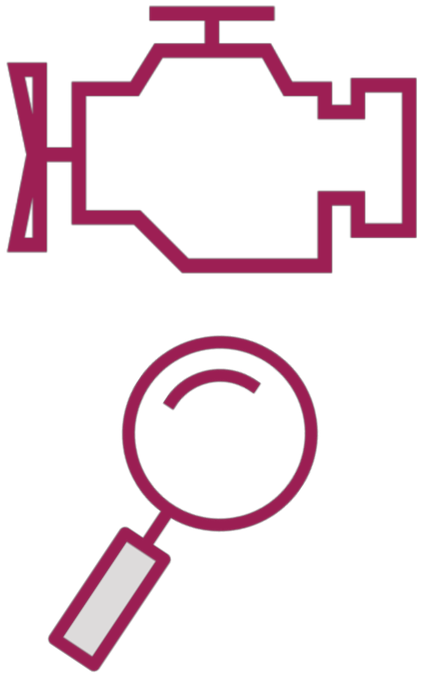
Loss Functions

Need optimizers for both networks

Loss function used is Binary Cross-Entropy (BCE) Loss

Used to heavily penalize incorrect classifications

Generator and Discriminator



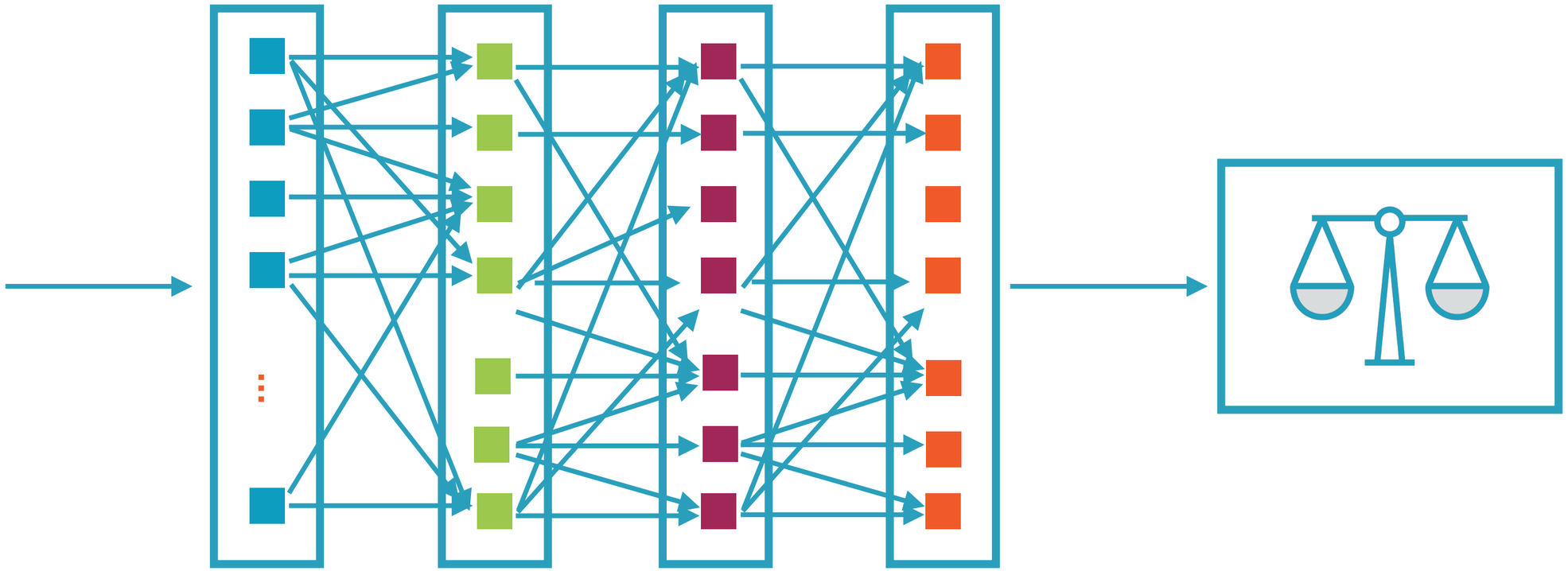
Adversaries during training

At some point generator will generate realistic data

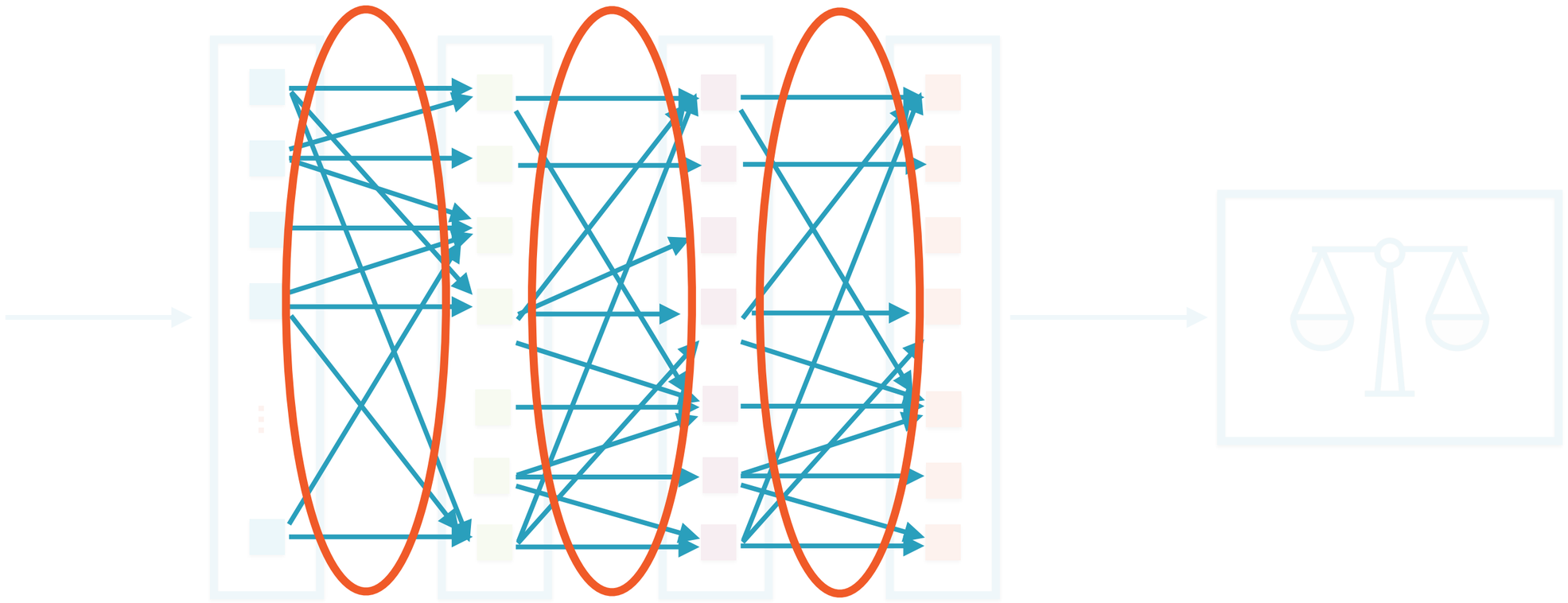
Consistently fool the discriminator

Leaky ReLU Activation

A Neural Network

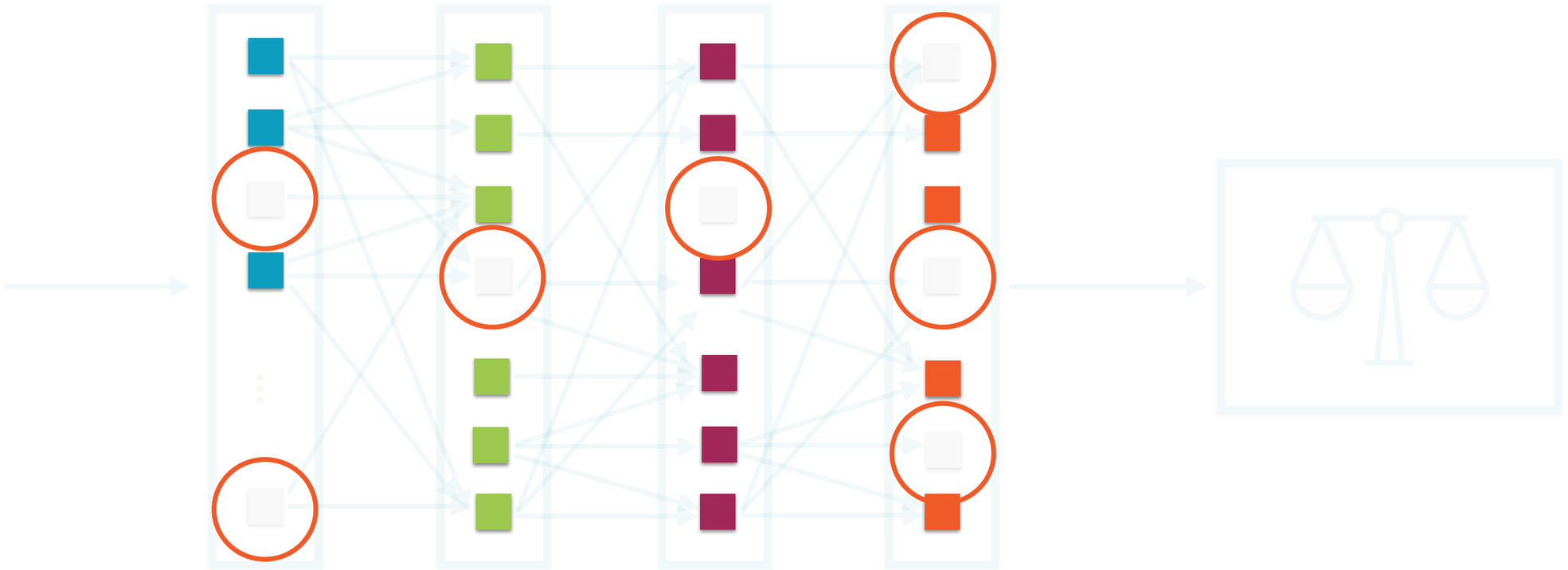


Unresponsive Neurons



What if the weights of the connections do not change in response to changing input?

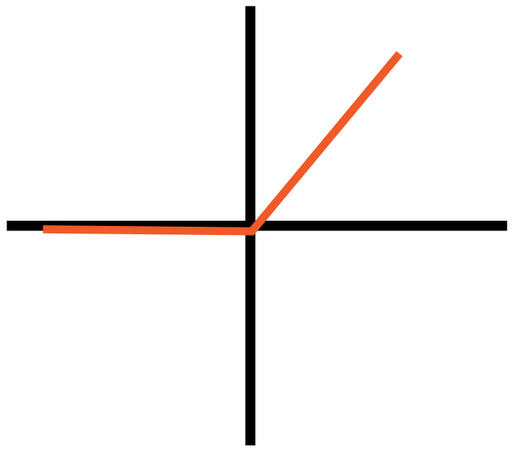
Unresponsive Neurons



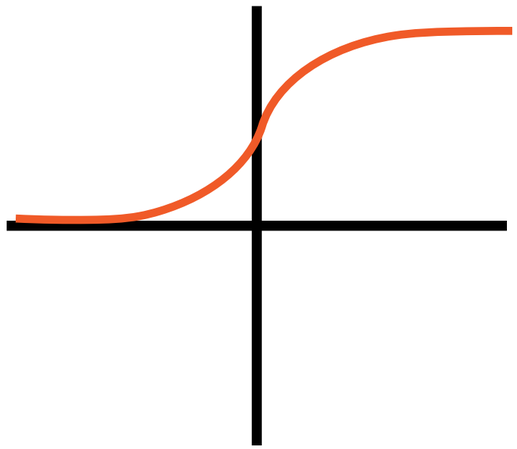
Neurons may be dead

Activation Functions

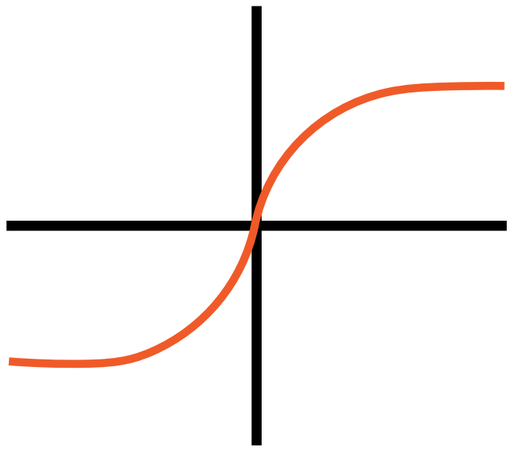
reLU



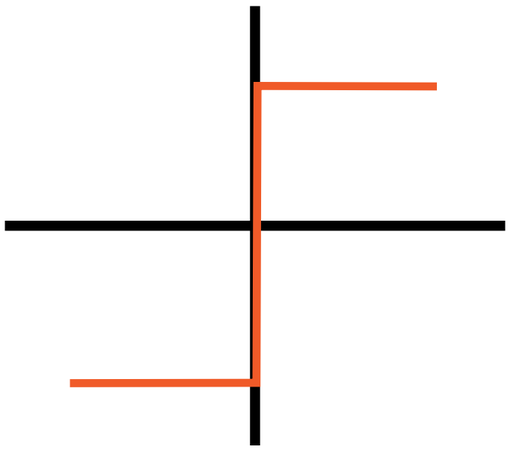
logit



tanh

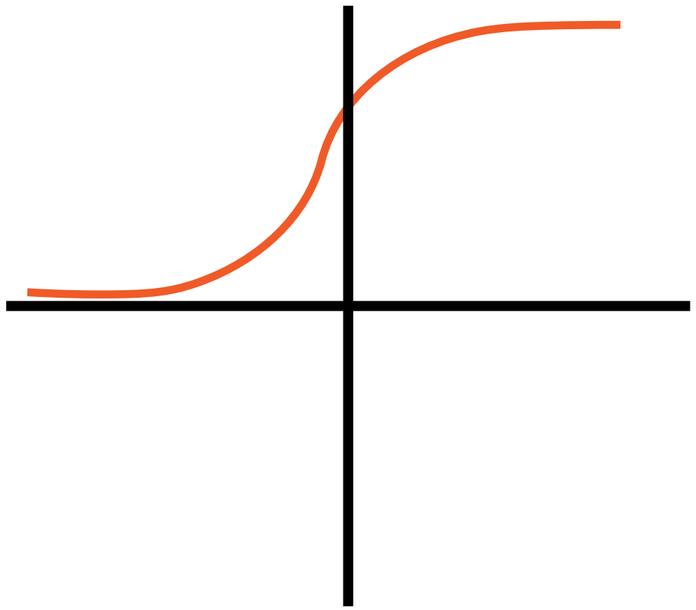


step



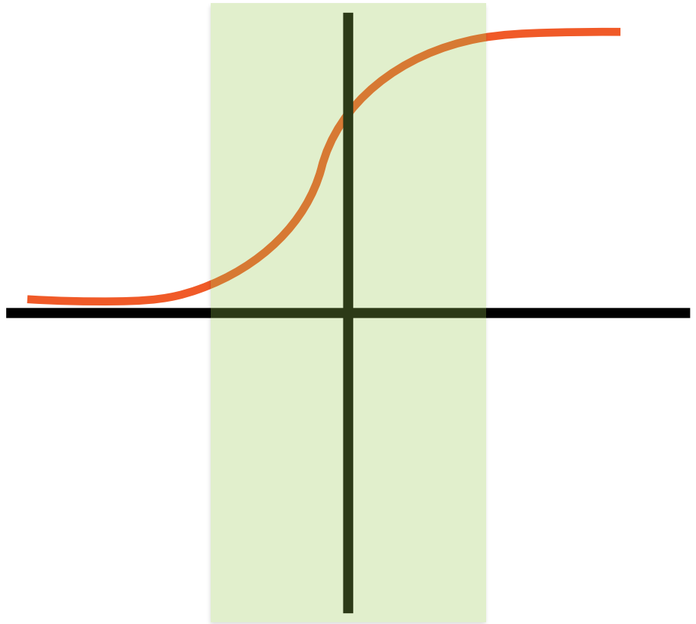
Various choices of activation functions exist and drive the design of your neural network

Activation Functions



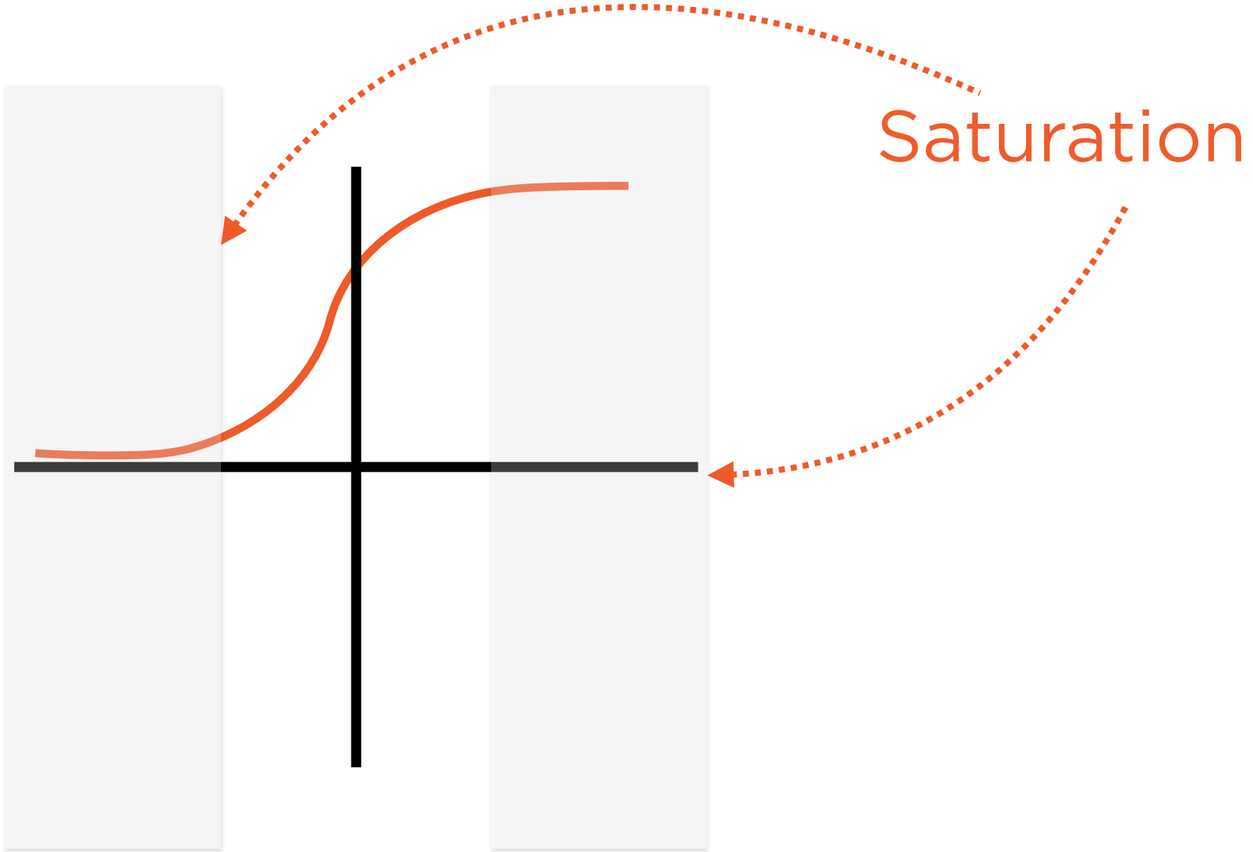
Consider an S-shaped (sigmoid) activation function

Activation Functions



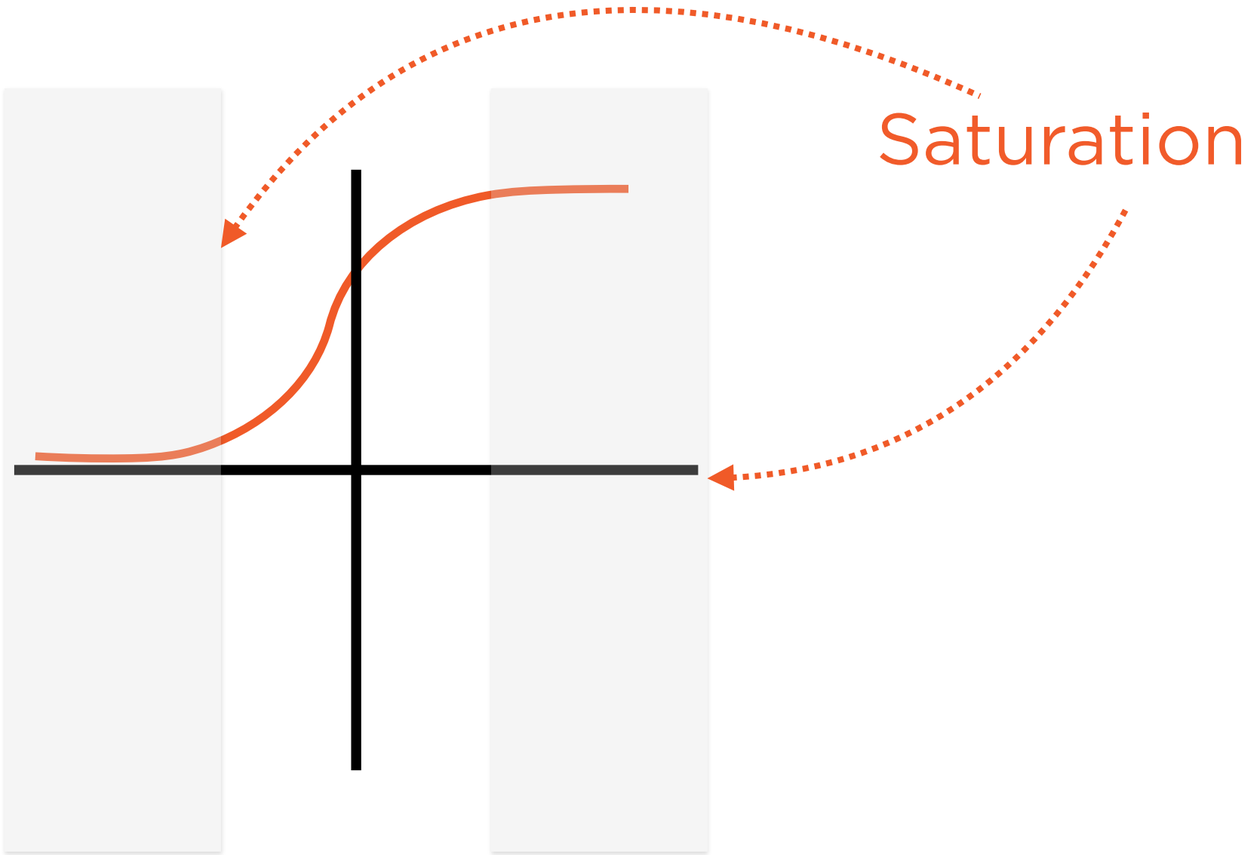
This is the active or responsive region of the function

Saturating Activation Functions



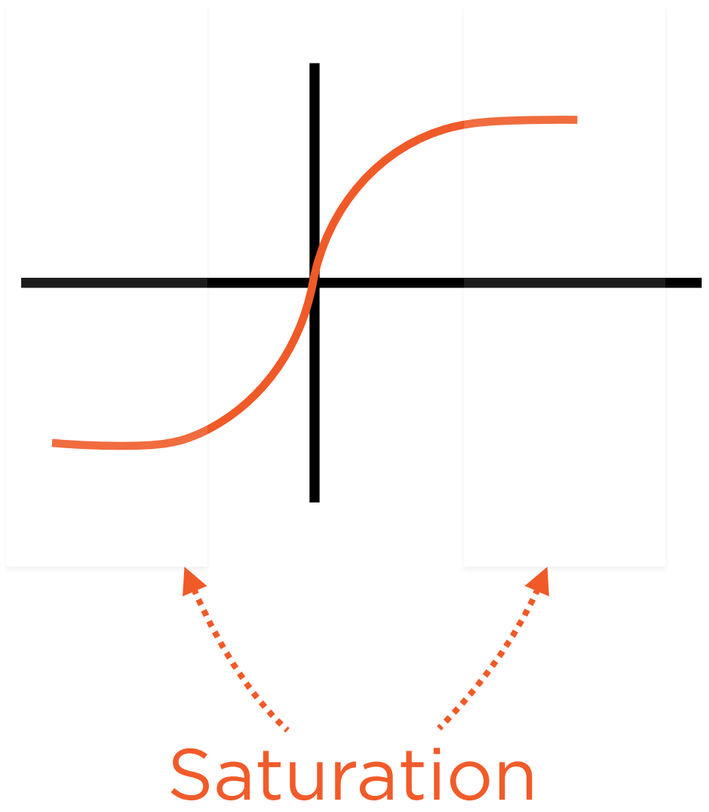
The activation function saturates at either end

Saturating Activation Functions



If a neuron operates within these saturation regions throughout training it might become unresponsive

Dying Neurons



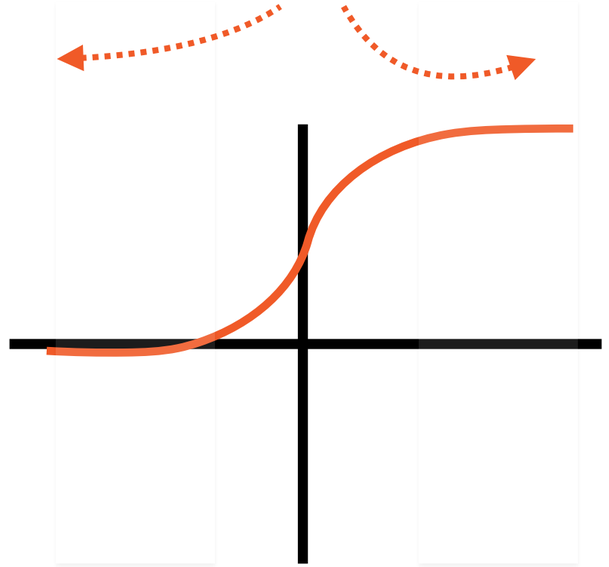
Neuron might become unresponsive - output won't change as input changes

If this continues throughout training, neuron is "dead"

Saturation of neuron occurs at both ends of S-curve, for instance

Saturating Activation Functions

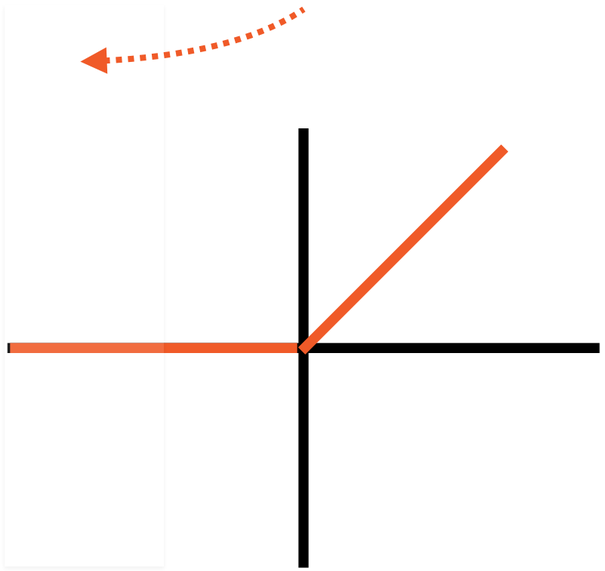
Saturation



Logit Activation

Saturates for very large and very small values of input

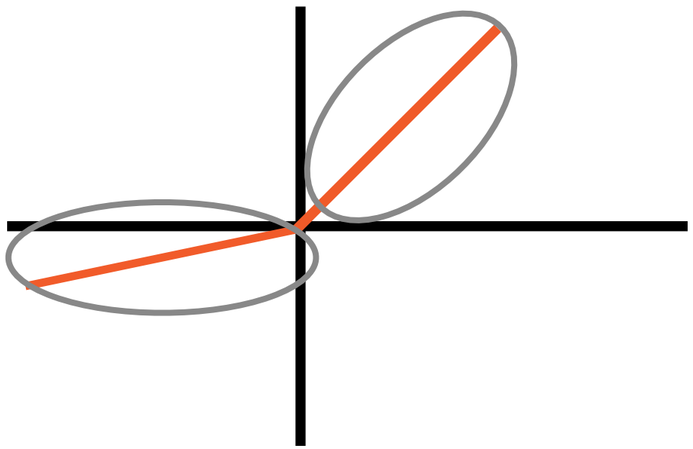
Saturation



ReLU Activation

Saturates for very small (negative values) of inputs

Leaky ReLU Activation



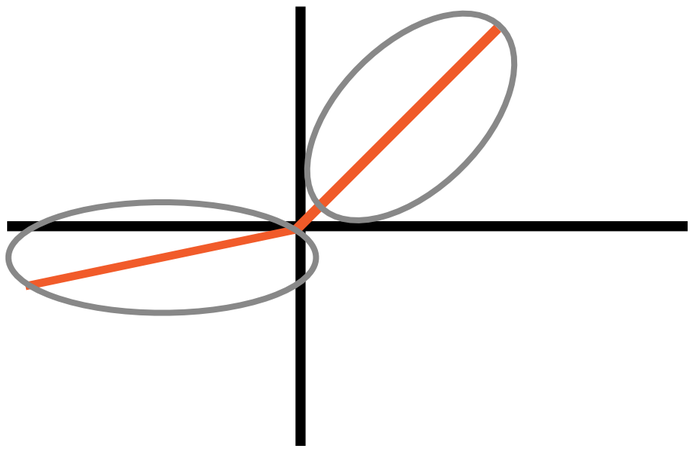
Mitigates dying-neuron problem of ReLU

For positive values, output same as input (like ReLU)

For negative values, output is non-zero, but close to zero

Leaky ReLU found to out-perform strict ReLU

Leaky ReLU Activation



$\text{LeakyRelu}_{\alpha}(x) = \max(x, \alpha x)$

α is a hyper parameter

Typically 0.01

Small non-zero value of α ensures that neuron does not “die”

Demo

Using Generative Adversarial Networks (GANs) to generate handwritten digits

Summary

Understand Generative Adversarial Networks (GANs)

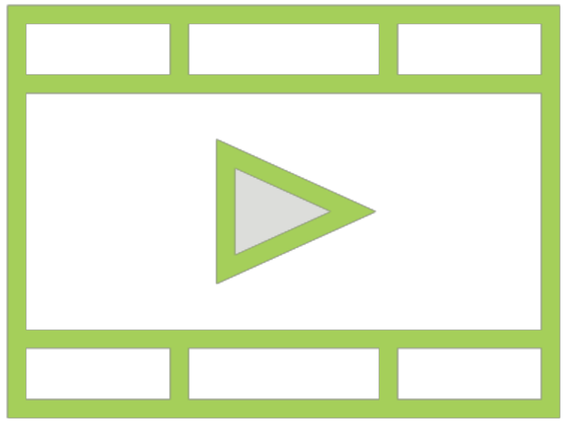
Candidate generation and evaluation

Role of generator

Role of discriminator

Generate handwritten digits using GANs trained on the MNIST dataset

Related Courses



Building Features from Image Data

Image Classification with PyTorch

Expediting Deep Learning with Transfer Learning: PyTorch Playbook