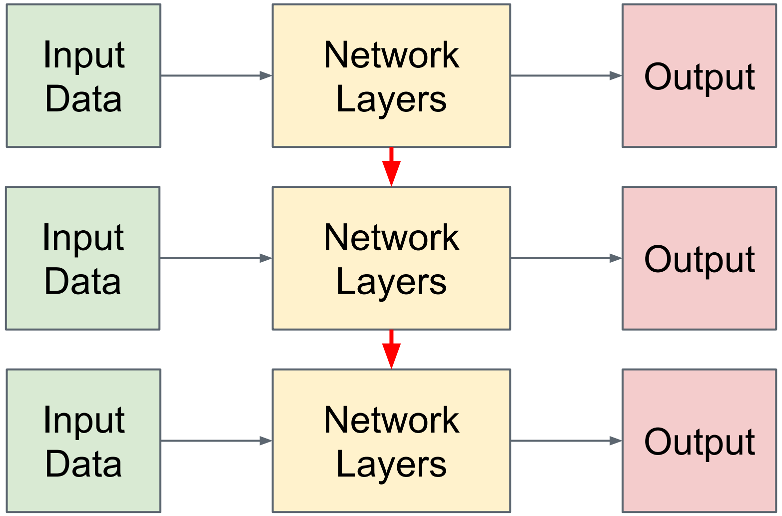
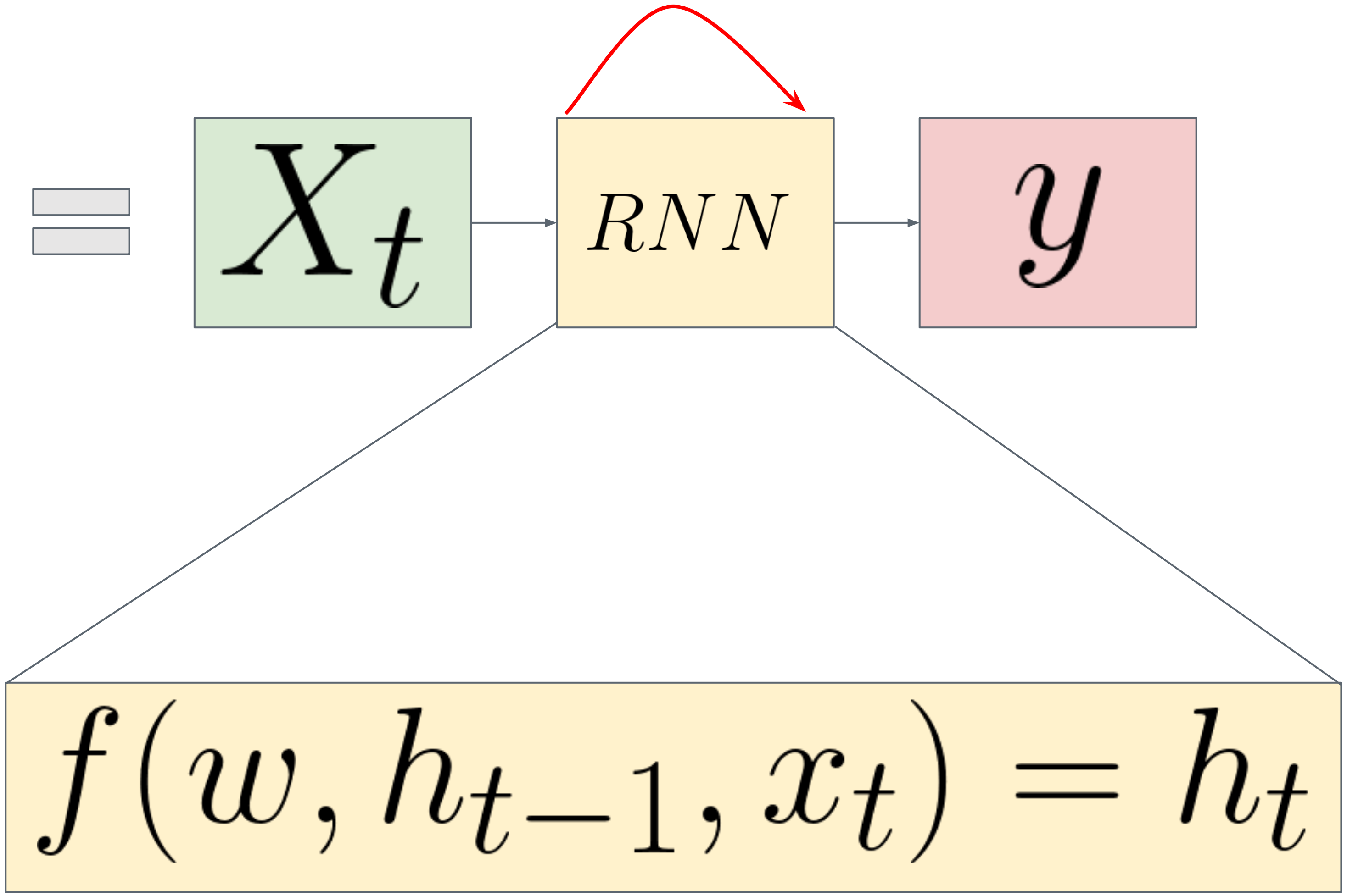
DATA 442: Neural Networks & Deep Learning

Dan Runfola – danr@wm.edu

icss.wm.edu/data442/





https://www.youtube.com/watch?v=_7SVJGTjUMI

Long short term memory network

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Simple RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Classification Models

Input

Output

Train

Loss Function,
and eventually
Model Weights

Classification Models

	Input	Output
Train		Loss Function, and eventually Model Weights
Test		“Cat”

Generative Models

Input

Output

Train

Loss Function,
and eventually
Model Weights

Test

“Cat”

Generative Models

Input

Output

Given some input of images, generate an output of samples drawn from the same distribution.


ZOOM IN... NOW ENHANCE!

Types of Generative Models

NADE/MADE

PixelRNN

Variational Autoencoder

Boltzmann Machine

GAN

GSN

Types of Generative Models

NADE/MADE

$\boxed{\text{PixelRNN}}$ Explicit Density Estimation

$\boxed{\text{Variational Autoencoder}}$ Approximate Density Estimation

Boltzmann Machine

$\boxed{\text{GAN}}$ Implicit Density Estimation

GSN

PixelCNN (..and PixelRNN)

- Fully Visible Belief Network

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

PixelCNN (..and PixelRNN)

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Really complex distribution
(conditioned on every other pixel in
an image!)

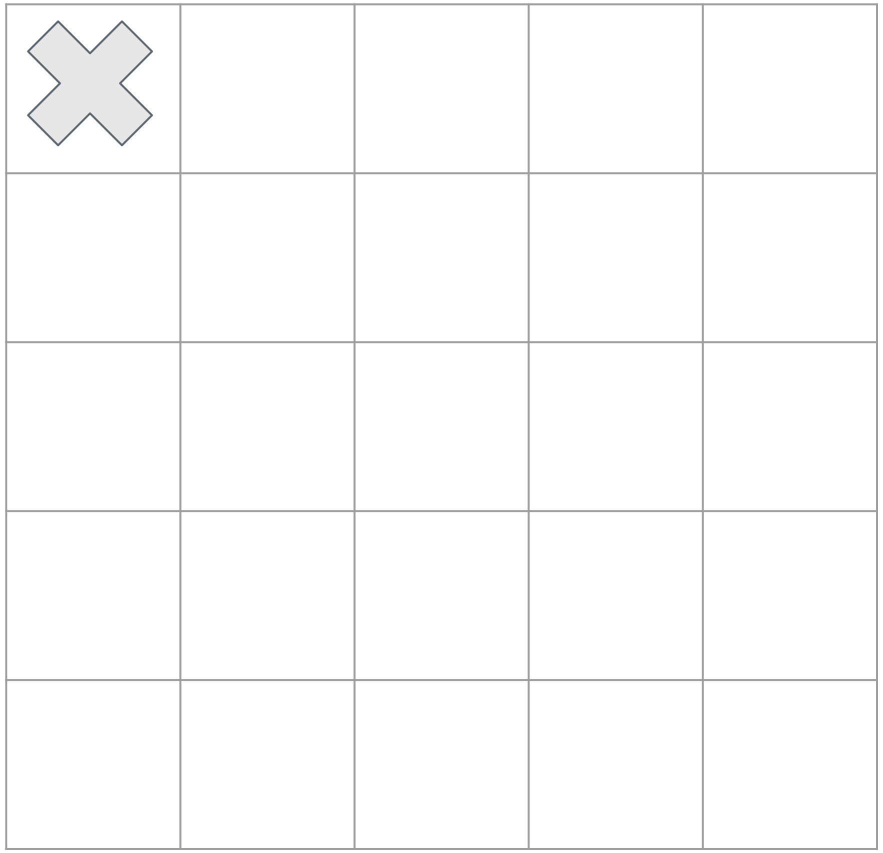
PixelRNN

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1})$$

Starts by generating an image pixel in the upper-left hand corner.

Every pixel is then fed in, in order, following a RNN approach.

Goal is to model the probability distribution of all pixels by fitting the RNN.



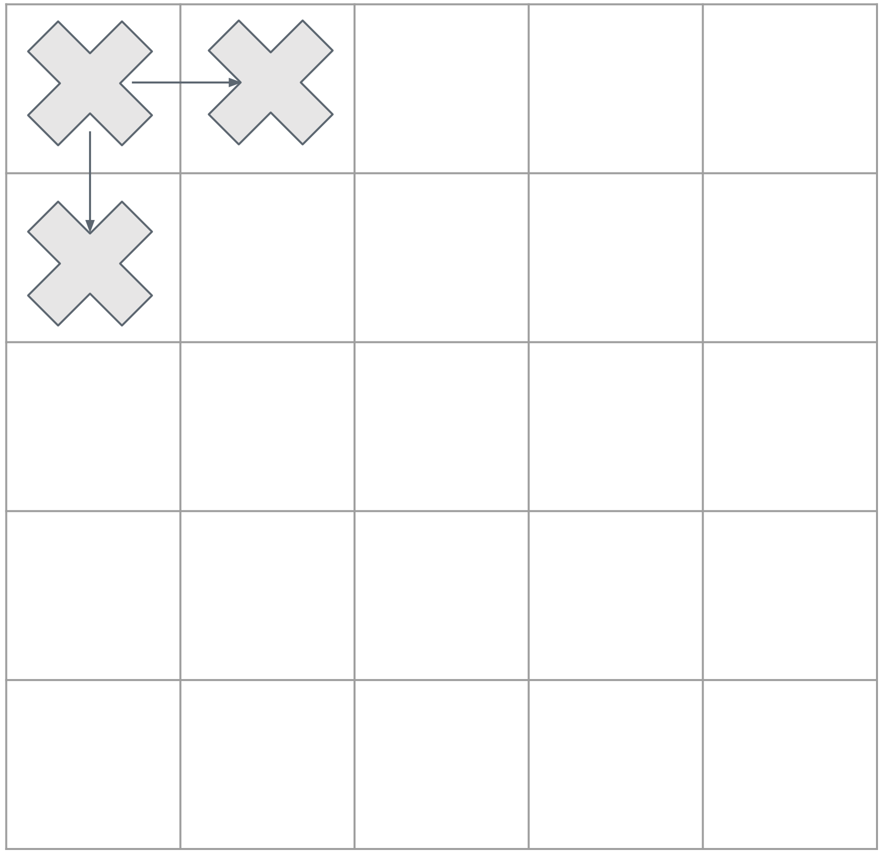
PixelRNN

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Starts by generating an image pixel in the upper-left hand corner.

Every pixel is then fed in, in order, following a RNN approach.

Goal is to model the probability distribution of all pixels by fitting the RNN.



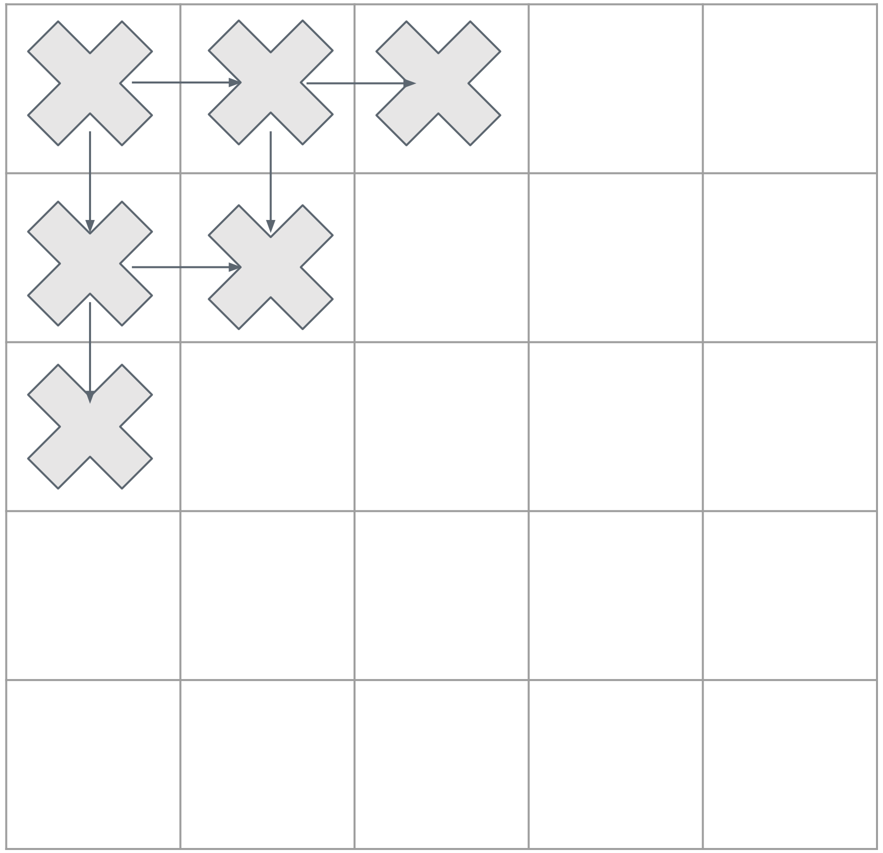
PixelRNN

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

Starts by generating an image pixel in the upper-left hand corner.

Every pixel is then fed in, in order, following a RNN approach.

Goal is to model the probability distribution of all pixels by fitting the RNN.

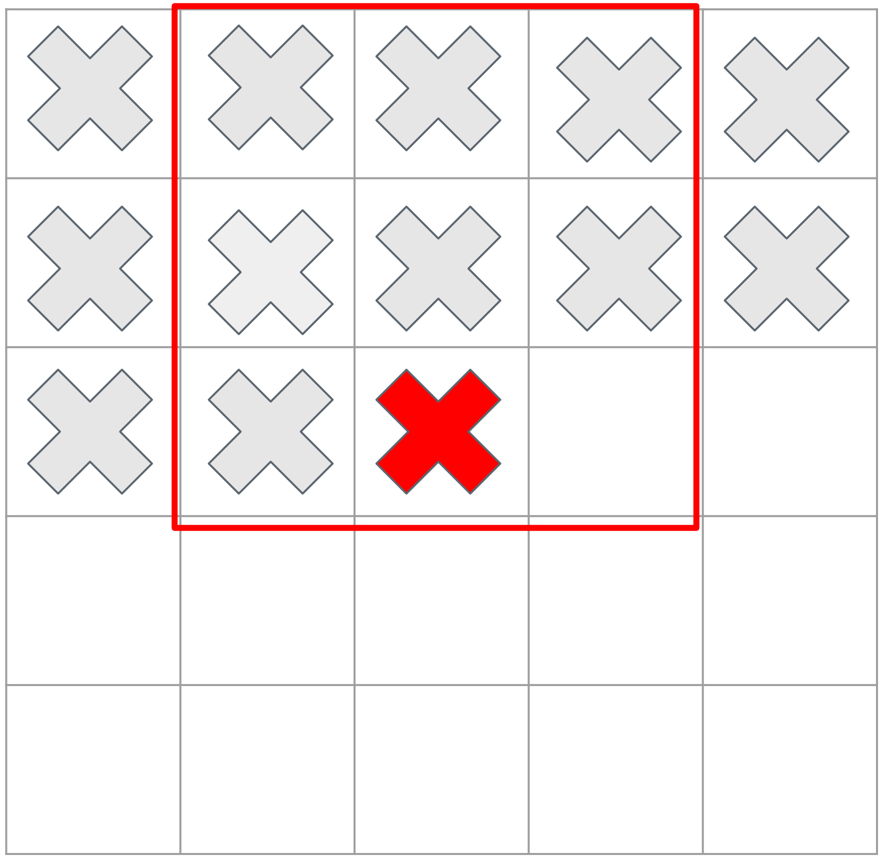


PixelCNN

$$p(image) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1})$$

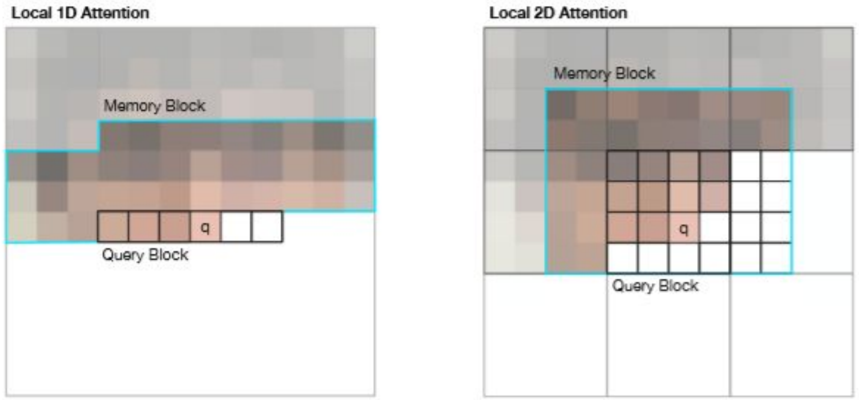
Generates the probability distribution based on filters.

Each pixel is generated based on it's own local neighborhood.



PixelRNN/PixelCNN

- Explicitly calculates likelihood
  - Can help in understanding model performance
- Relatively Slow
- Makes fairly believable images
- Area of extensive inquiry - PixelCNN+; PixelCNN++, PixelCNN 2.0, and many more.
  - Concept of 'Attention'

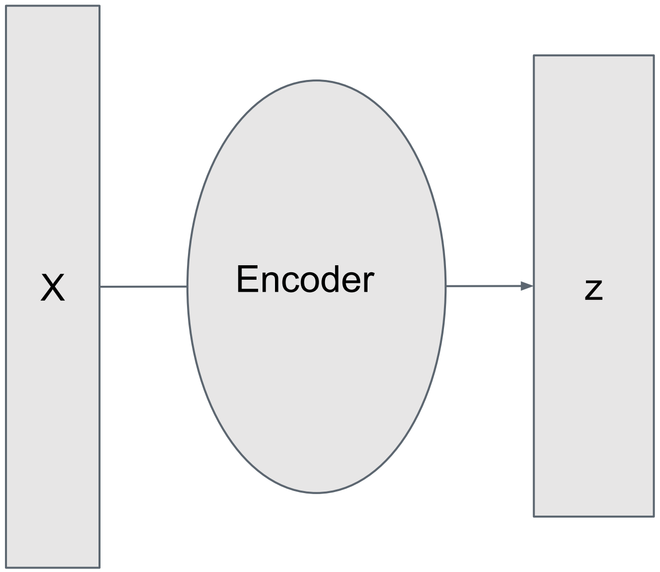


http://cs231n.stanford.edu/reports/2016/pdfs/211_Report.pdf

Variational Autoencoders (VAE)

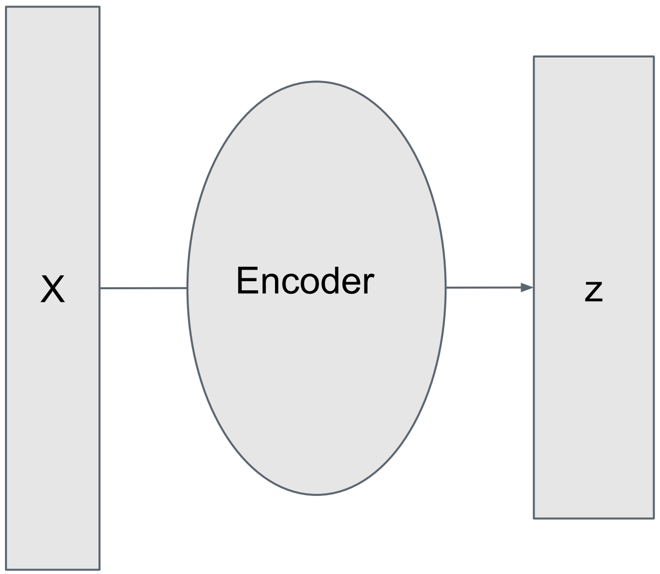
$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)



$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)

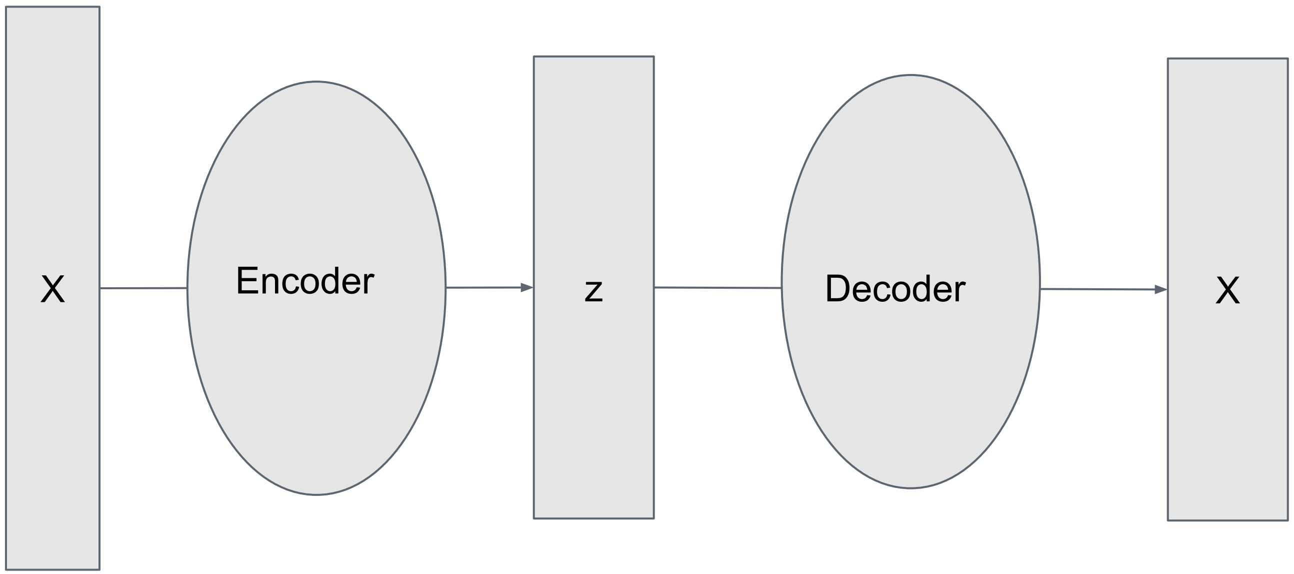


Encoder can be:
A Linear Model
A Neural Network
a CNN
..or nearly anything else.

Historically, it started as a linear model.

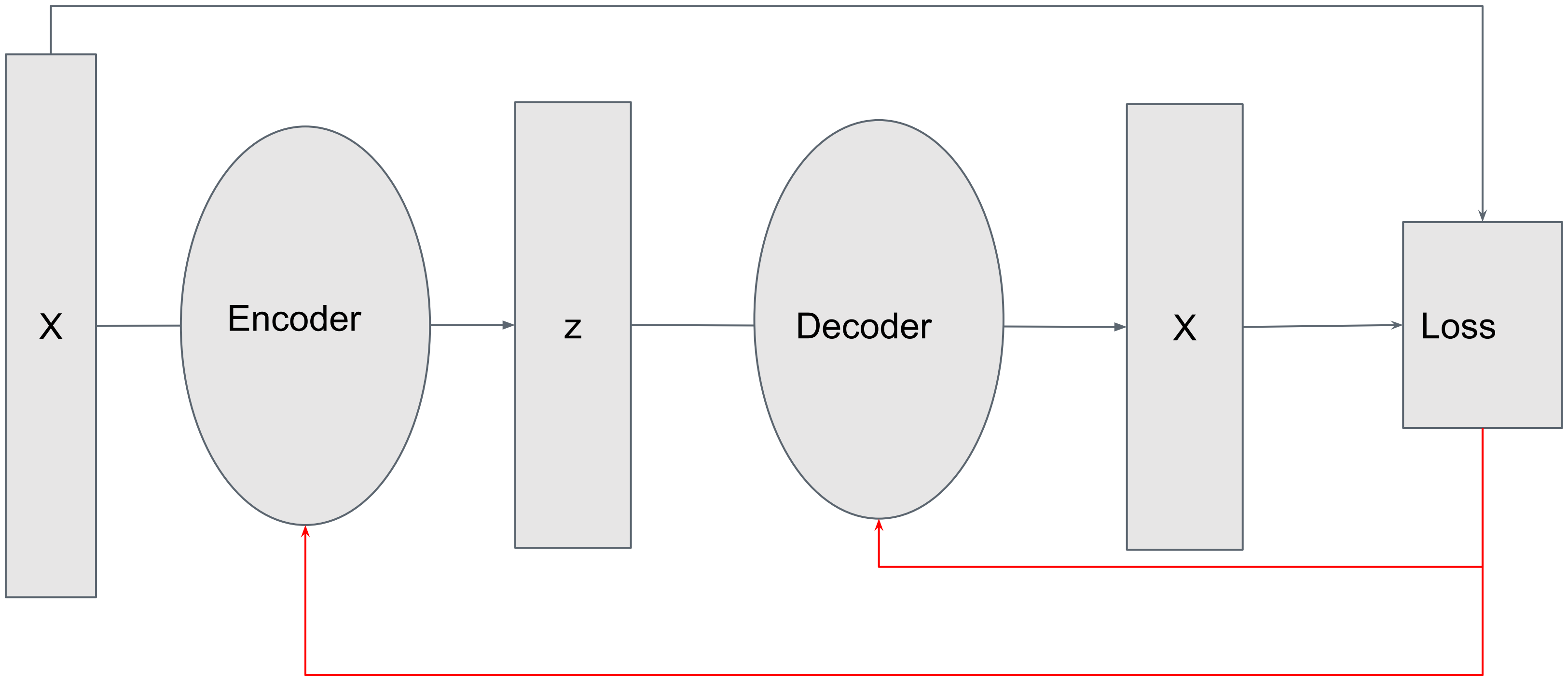
$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)



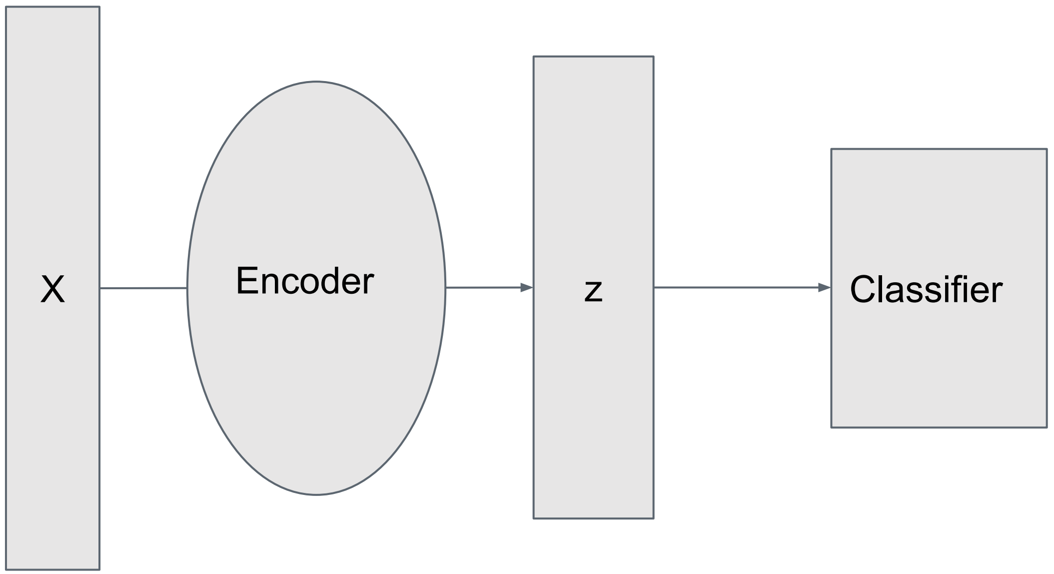
$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)



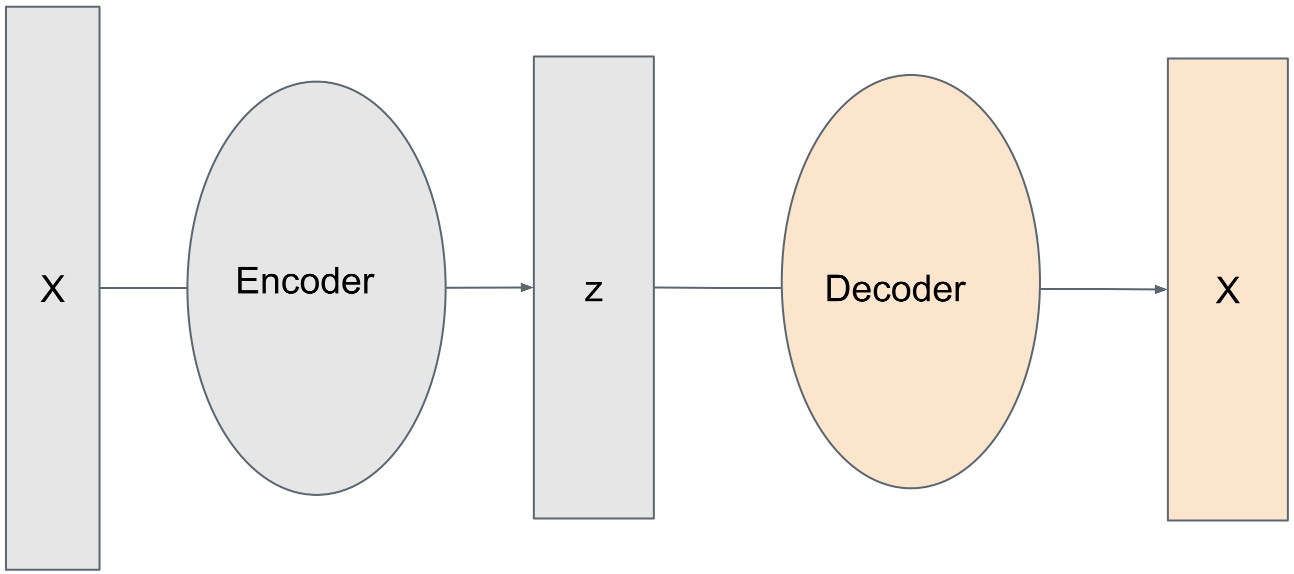
$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)



$$p(image) = \int p(z)p(x|z)dz$$

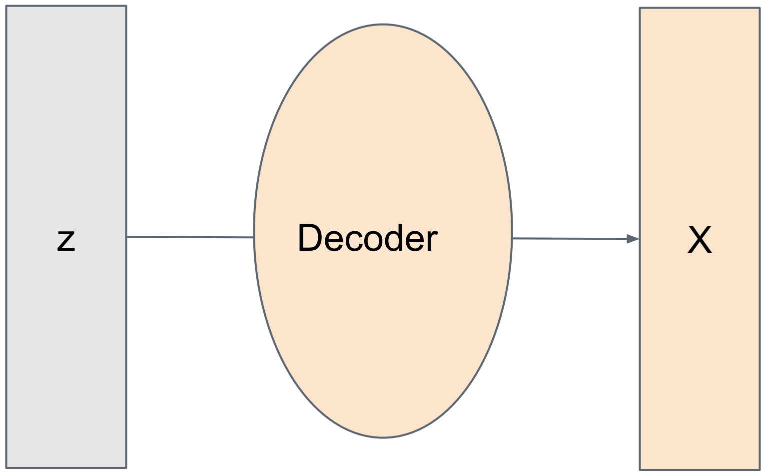
Variational Autoencoders (VAE)

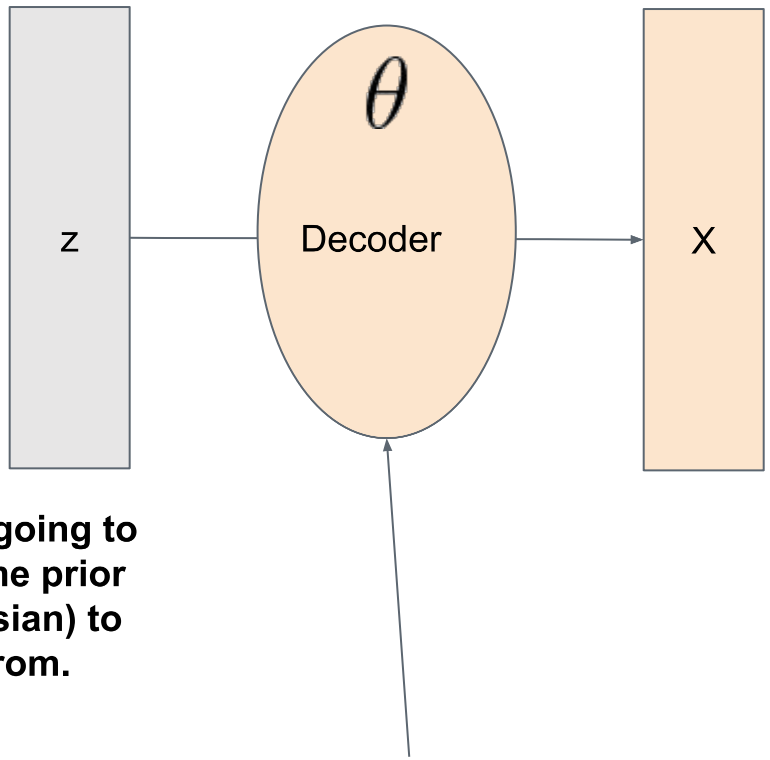


$$p(image) = \int p(z)p(x|z)dz$$

Variational Autoencoders (VAE)

- We assume that all of our data is generated from some unknown (i.e., latent) *z*.

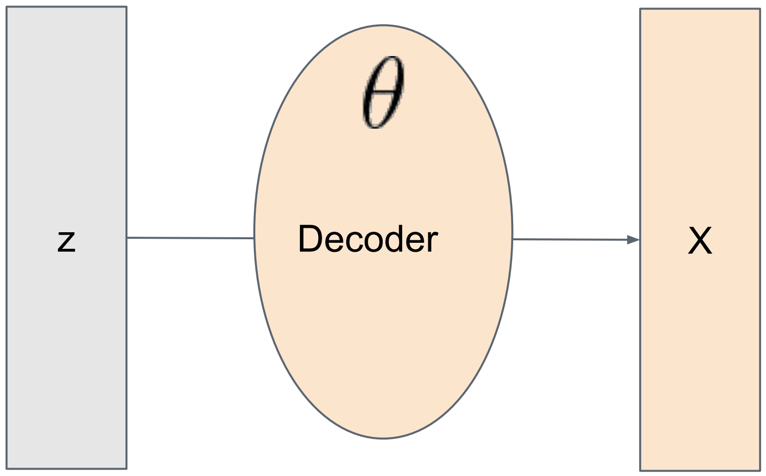




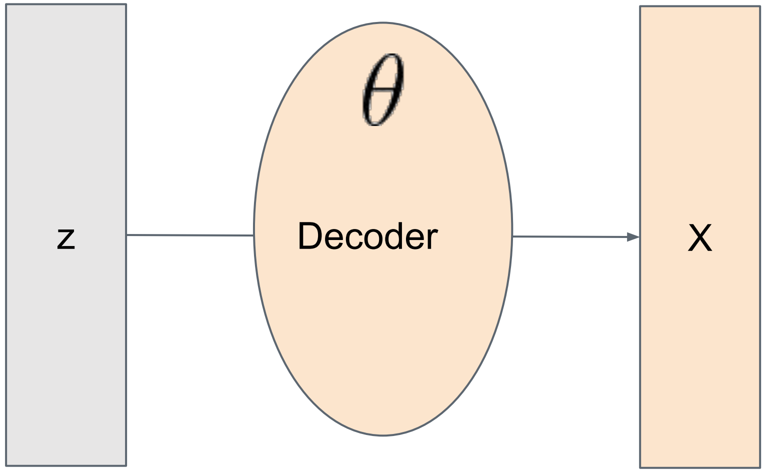
1 **Because we're assuming our data is generated from some underlying Z, we don't actually know what this Decoder is. We're assuming it exists, but need to solve for it.**

2 **For Z, we're going to assume some prior (i.e., a gaussian) to sample from.**

3 **Here, we're going to sample from X | Z, generally represented by a neural network.**

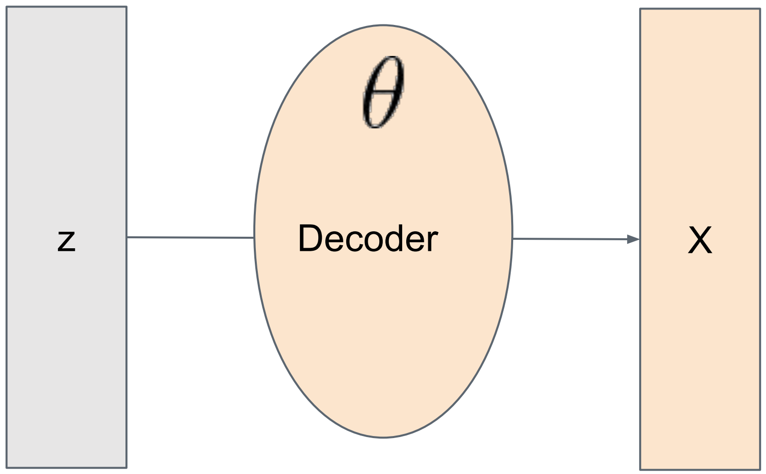


$$p(image) = \int p(z)p(x|z)dz$$

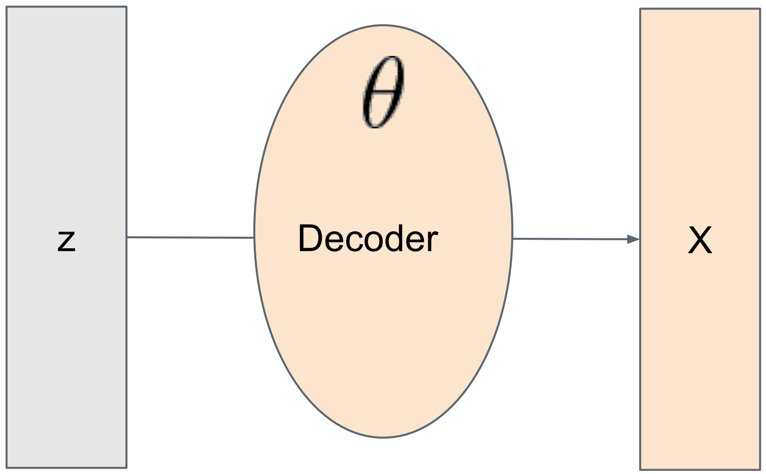


$$p(image) = \int \boxed{p(z)} p(x|z) dz$$

Gaussian Prior



$$p(image) = \int \underbrace{\boxed{p(z)}}_{\text{Gaussian Prior}} \underbrace{\boxed{p(x|z)}}_{\text{Decoder Neural Net}} dz$$

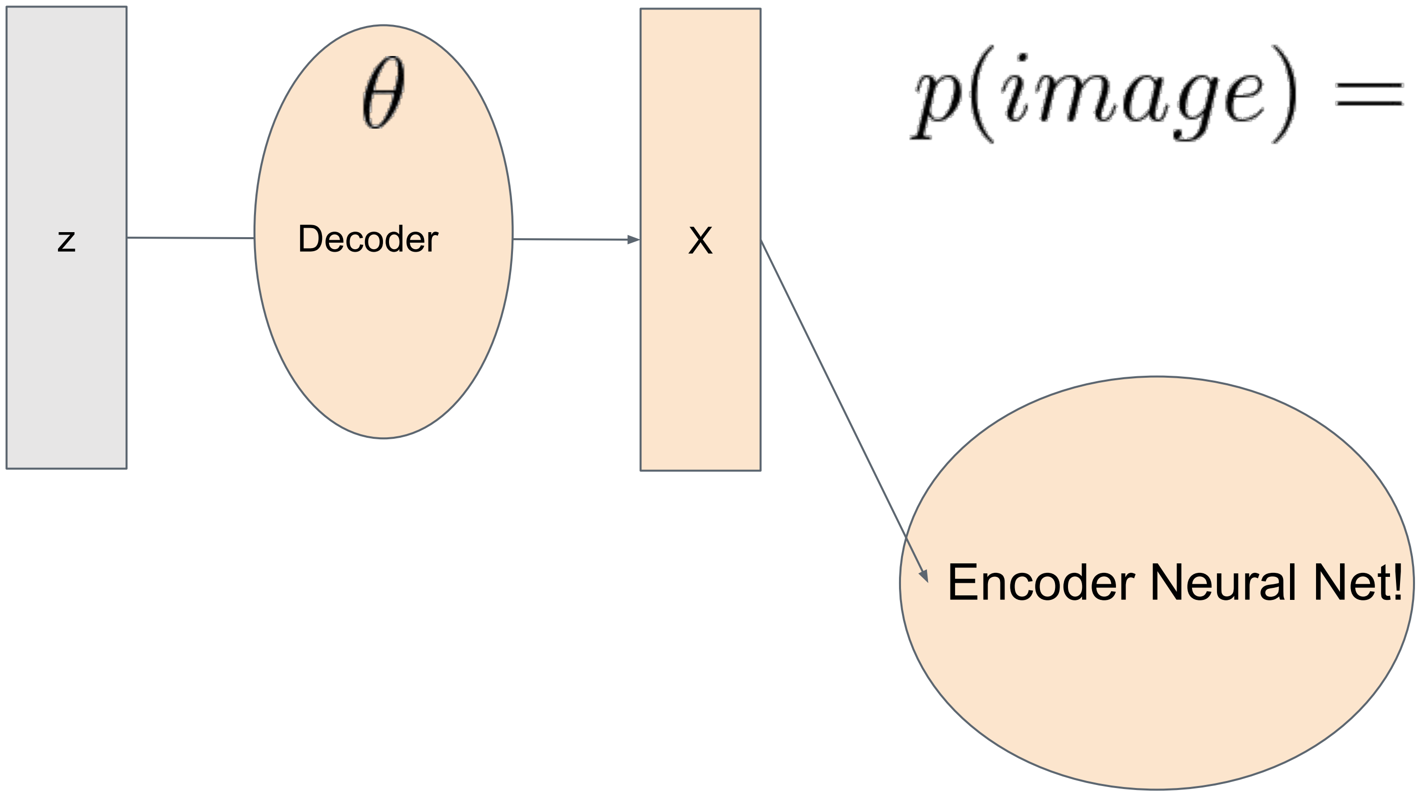


$$p(image) = \boxed{\int} \boxed{p(z)} \boxed{p(x|z)} dz$$

Gaussian Prior

Decoder Neural Net

p(x|z) can't be calculated for every possible Z!



$$p(image) = \int p(z)\boxed{p(x|z)}dz$$

Decoder Neural Net

Variational Autoencoders (VAE)

https://www.compthree.com/blog/autoencoder/

Generative Adversarial Networks

What if we decide not to try and explicitly solve for the probability densities, and instead just sample the space created by Z?

Solution: GANs, or 2-player games, where both players are neural networks.

GAN

The Players:

Red Team: Generator Network

Tries to generate real-looking images.

Blue Team: Discriminator Network

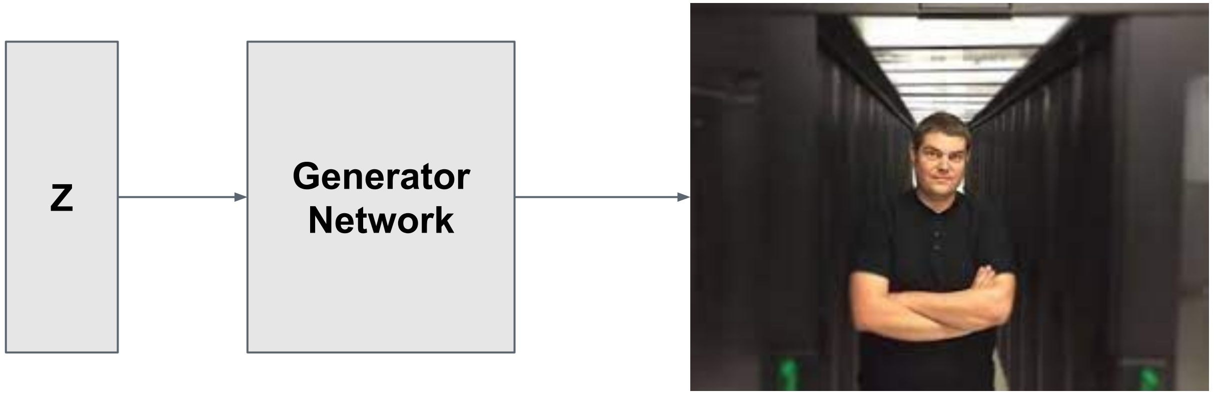
Tries to distinguish between real and fake images.

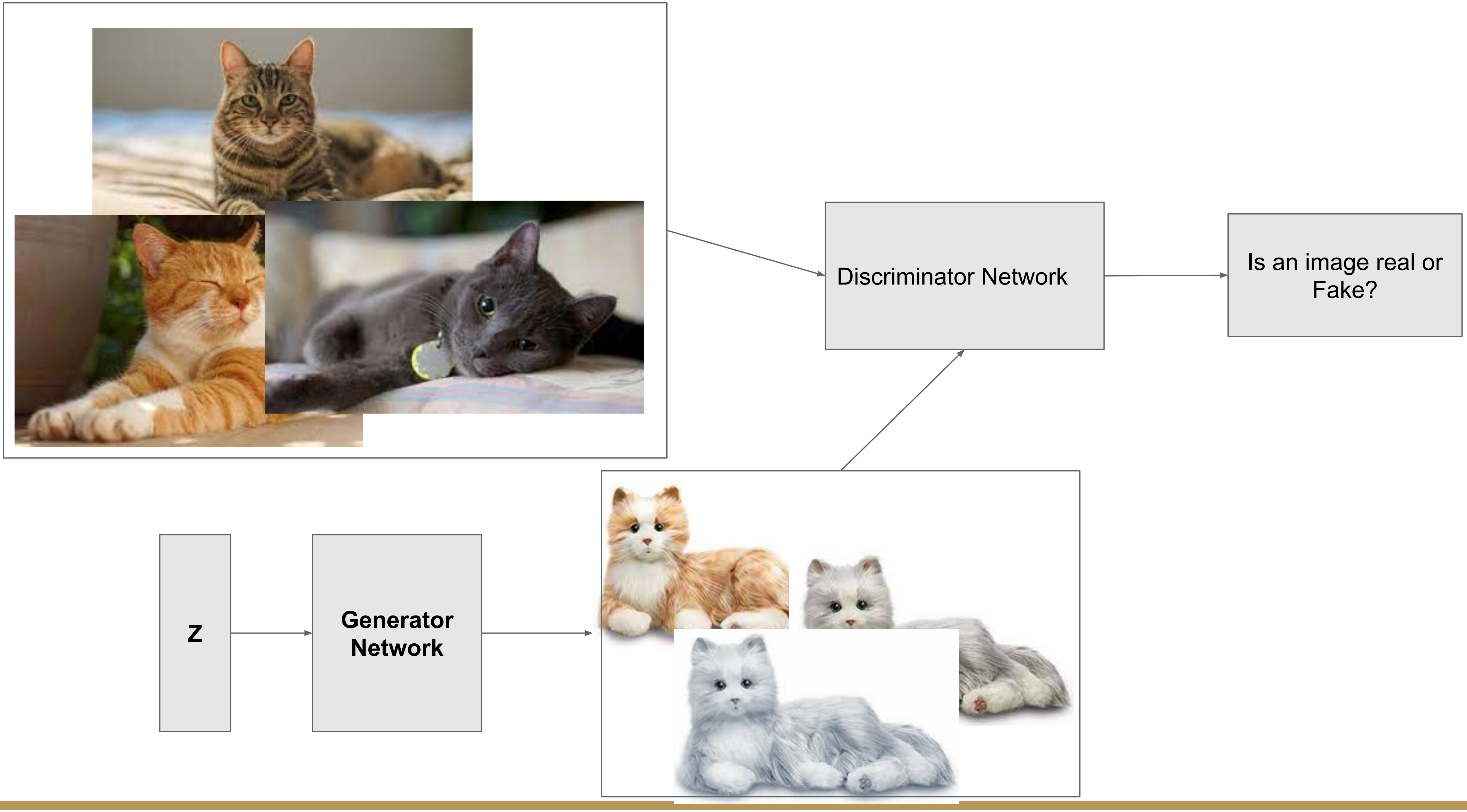
GAN

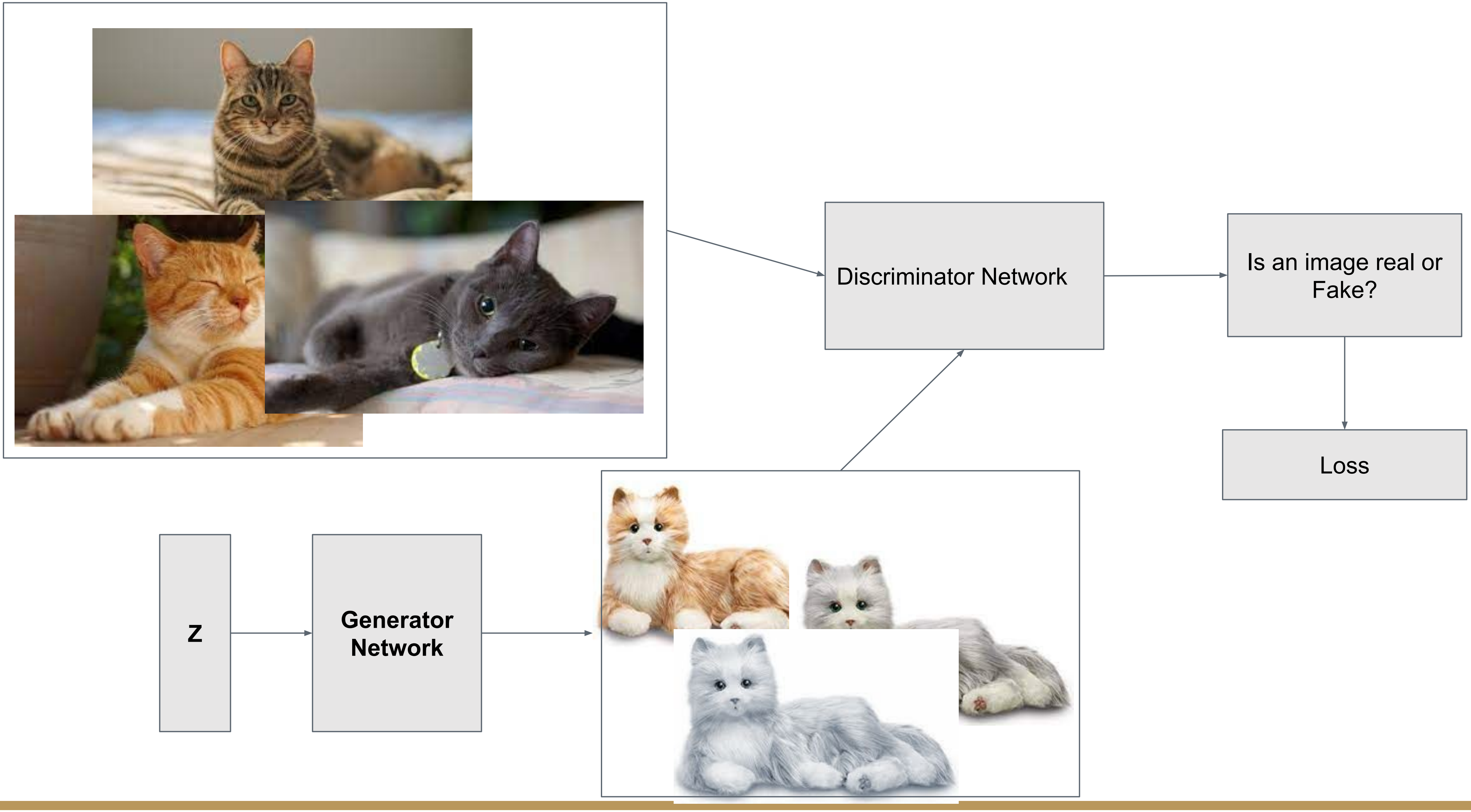
Intuition:

(A) Sample from random noise (i.e., gaussian).

(B) Learn a transformation (neural network) that modifies noise to our training data distribution.



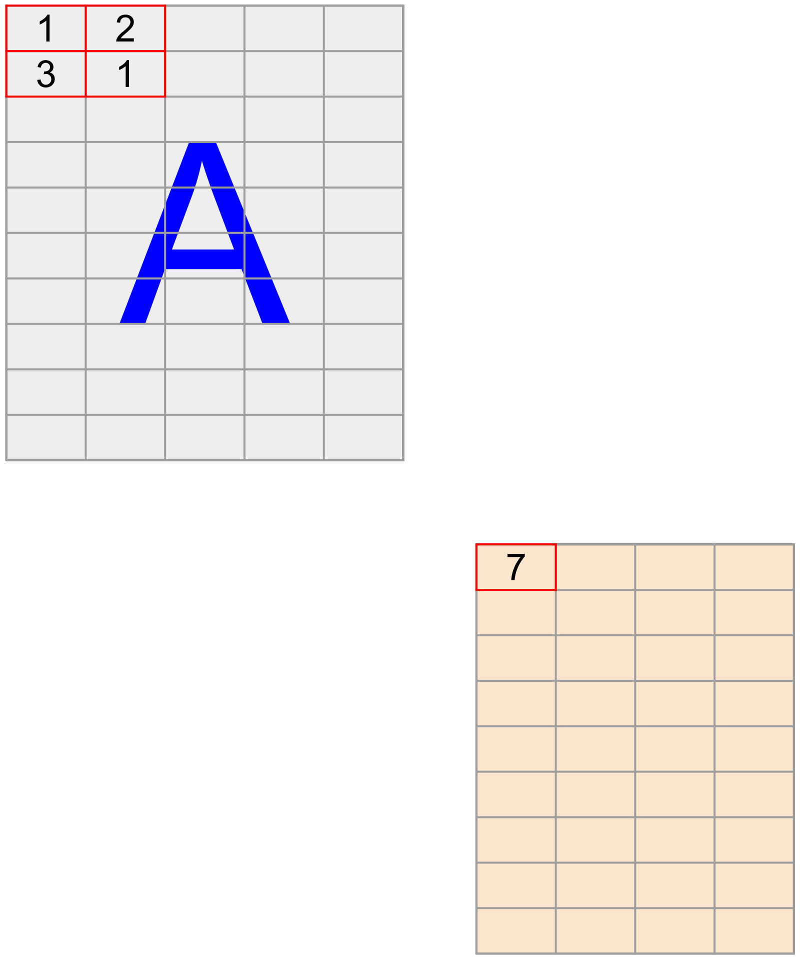

Discriminator Network
Is an image real or Fake?
z
Generator Network


Discriminator Network
Is an image real or Fake?
Loss
Z
Generator Network

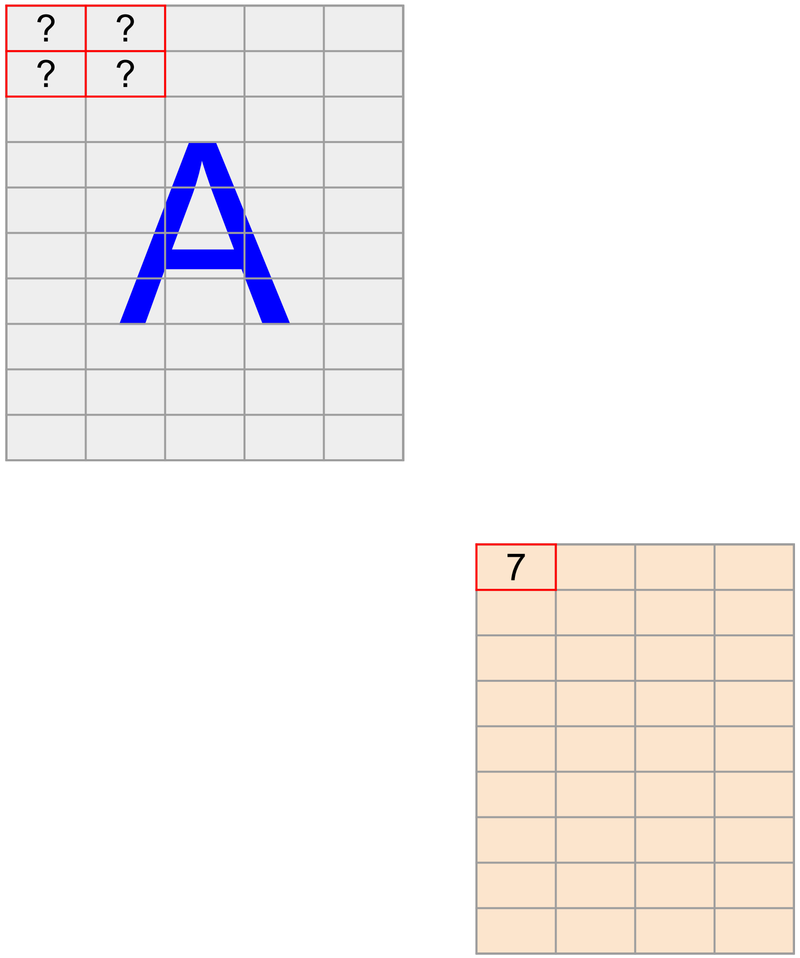
Defining GAN Architecture

- Interlinking two neural networks is a very unstable proposition.
- Easy to end up generating a lot of noise from noise, without finding helpful solutions, if your discriminator or generator are poorly specified.
- Generator is going to be focused on upsampling from the noise (Z) - more on this next slide.
- Discriminator is going to be a traditional CNN.

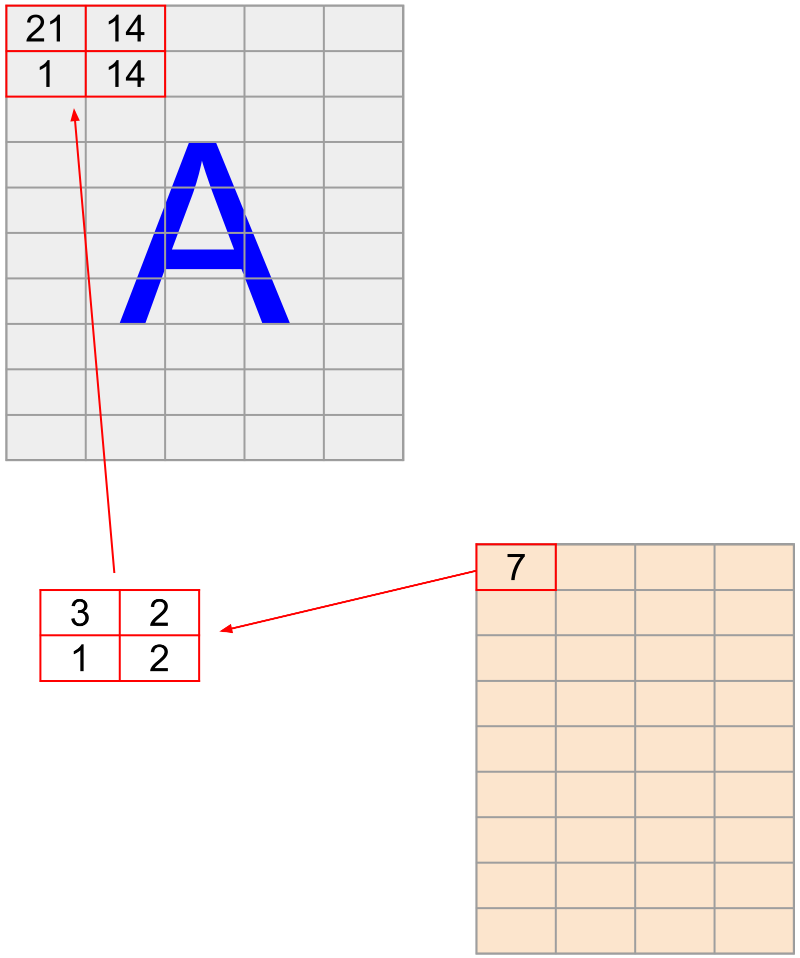
Generator Networks



Generator Networks



Generator Networks



Rules of Thumb

- Use batch normalization
- Fully connected layers can be troublesome in deeper architectures
- LeakyReLU is important in the discriminator; ReLU in the generator.

Recap

- What is a Generative Network?
- PixelRNN / PixelCNN
- Variational AutoEncoders (VAE)
- Generative Adversarial Networks (GAN)