

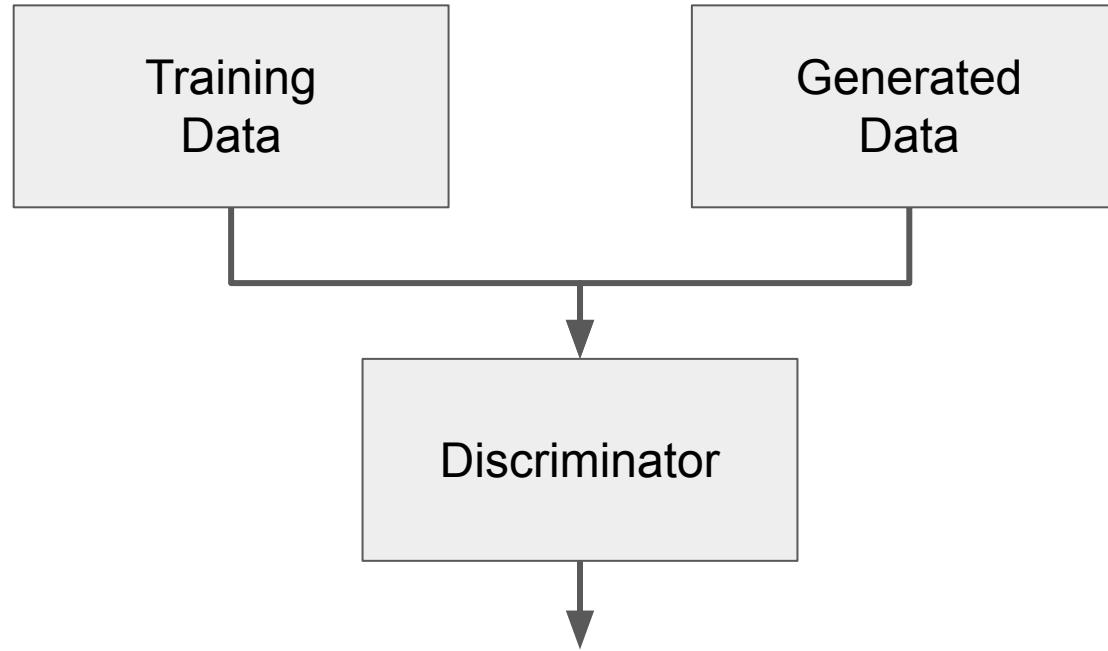
# MSBA 434: Advanced Workshop on Machine Learning

## Lecture 4: Generative Adversarial Networks

# Agenda

1. Adversarial Training
2. Case Study: DCGAN
3. Case Study: SRGAN
  
4. Case Study: BigGAN
5. AWS SageMaker
6. Assignment 4
  
7. Case Study: PGN
8. Case Study: StackGAN
9. Other Generative Tasks

# Discriminator Task



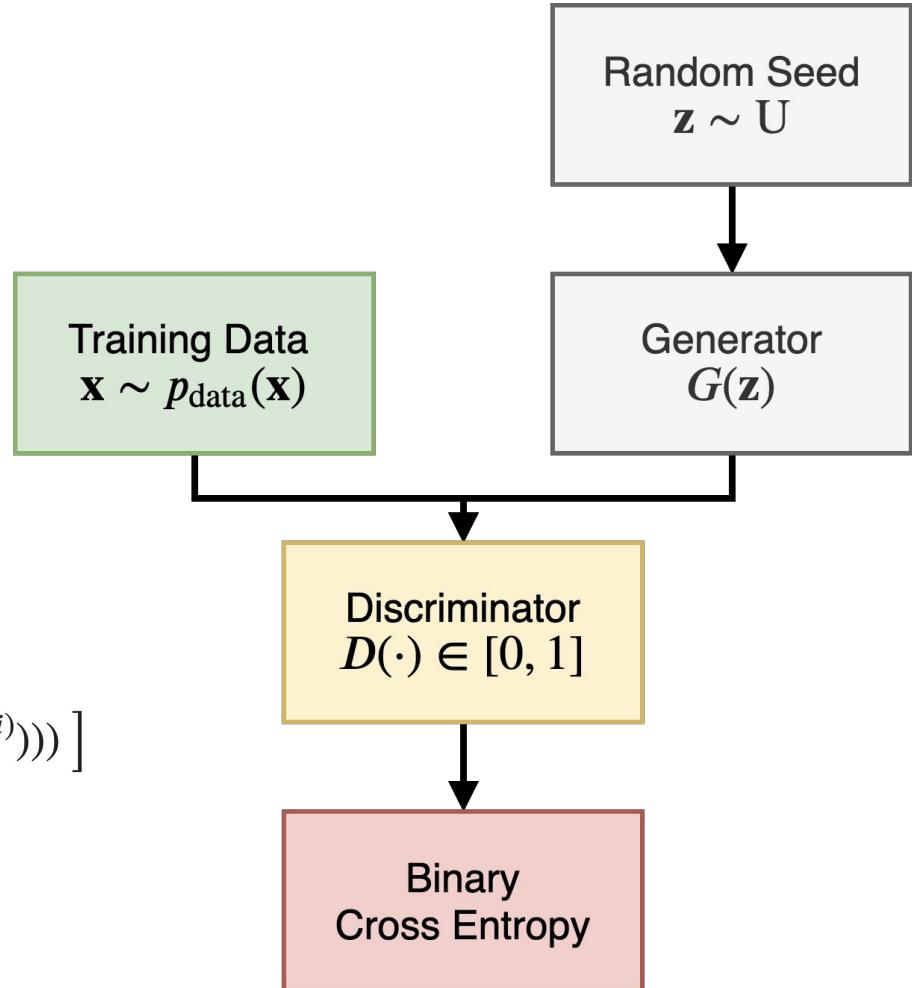
Goodfellow, I., et al. (2014)

# Discriminator Training

Discriminator given a batch of examples from **training data and the generator's output**.

Cost is the average binary cross-entropy across the batch:

$$C_D = -\frac{1}{m} \sum_{i=1}^m [ \log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) ]$$



# Discriminator Training

Discriminator given a batch of examples from **training data and the generator's output**.

Cost is the average binary cross-entropy across the batch:

$$C_D = -\frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log (1 - D(G(\mathbf{z}^{(i)}))) \right]$$

Random Seed  
 $\mathbf{z} \sim U$

Training Data  
 $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

Generator  
 $G(\mathbf{z})$

Discriminator  
 $D(\cdot) \in [0, 1]$

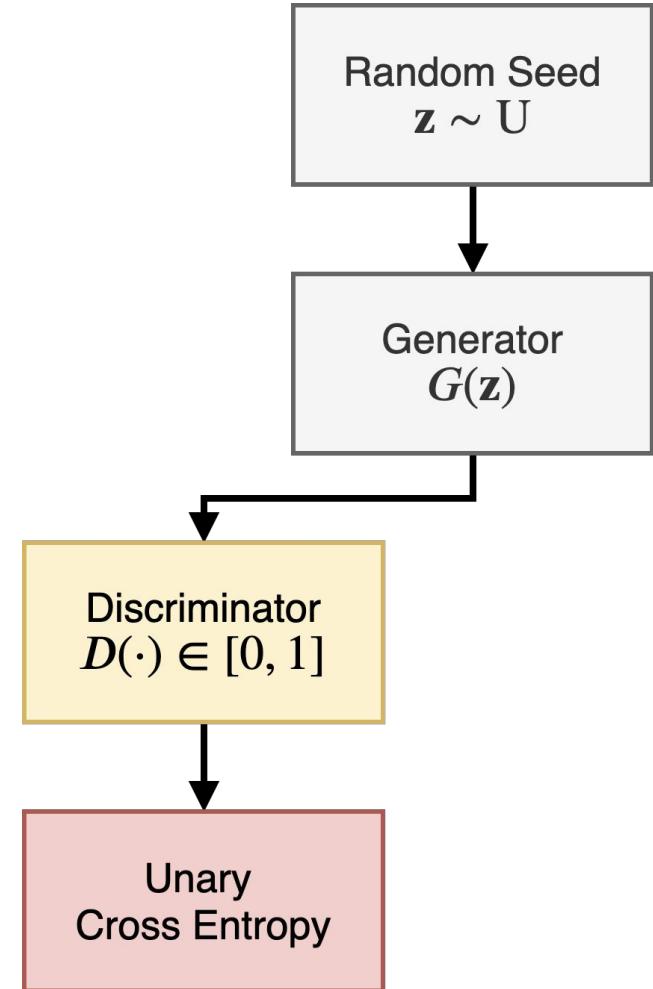
Binary  
Cross Entropy

# Generator Training

The discriminator is given a batch of examples from **the generator's output**, so it should always output 0.

The generator's cost goes up when the discriminator thinks the samples are coming from the generator:

$$C_G = -\frac{1}{m} \sum_{i=1}^m \left[ \log (D(G(\mathbf{z}^{(i)}))) \right]$$

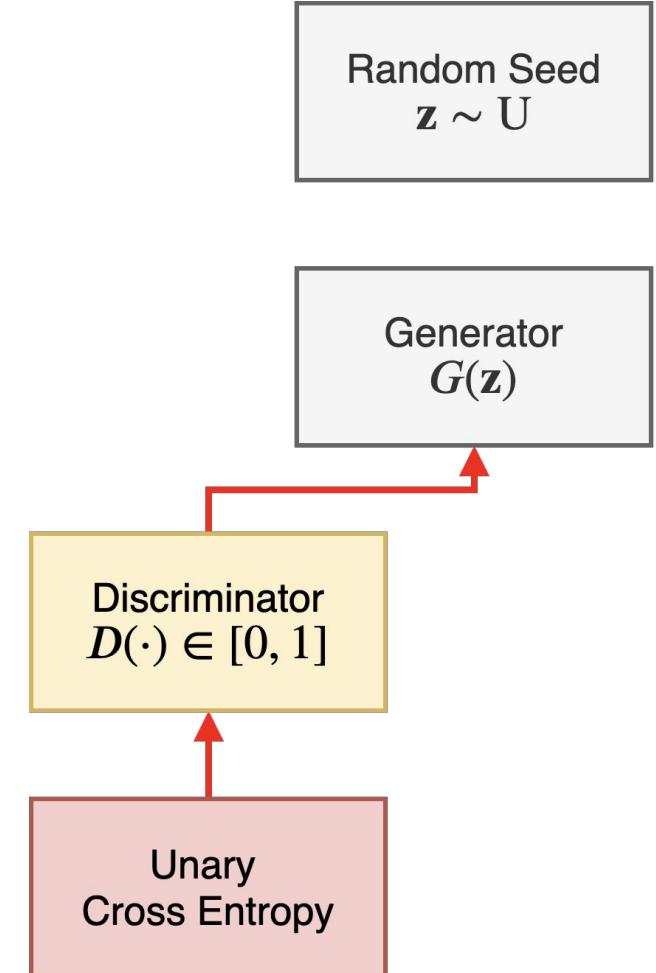


# Generator Training

The discriminator is given a batch of examples from **the generator's output**, so it should always output 0.

The generator's cost is the unary cross entropy, which penalizes the discriminator predicting 0:

$$C_G = -\frac{1}{m} \sum_{i=1}^m \left[ \log (D(G(\mathbf{z}^{(i)}))) \right]$$



# Training Procedure

1. Sample a batch of training and generated data
2. Train the discriminator for several steps
3. Sample a batch of noise
4. Train the generator
5. Repeat steps 1-4 until convergence.



Training one  
model at a time

# Combined Training

$$V(D, G) = \mathbb{E}_{\mathbf{x}} \left[ \log D(\mathbf{x}) \right] + \mathbb{E}_{\mathbf{z}} \left[ \log (1 - D(G(\mathbf{z}))) \right]$$


Rewards true positives      Rewards true negatives

$$\min_G \max_D V(D, G)$$

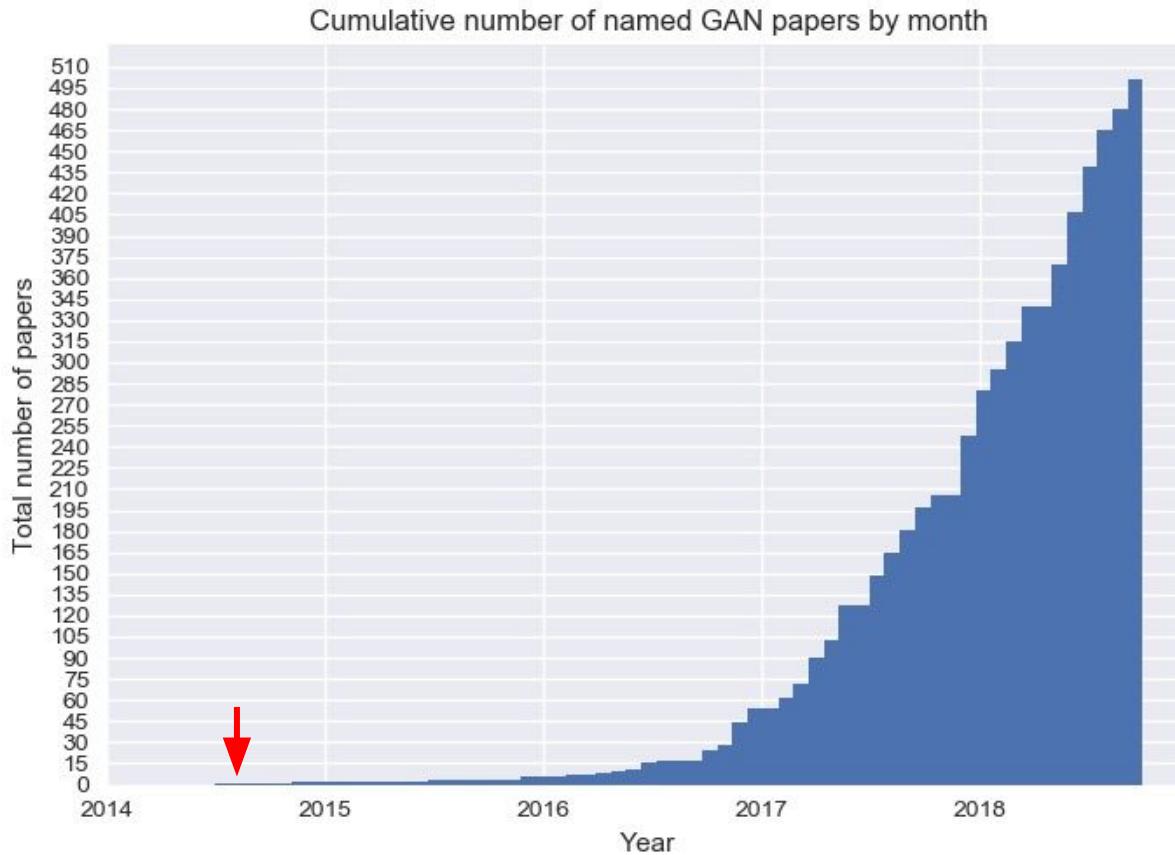
$$G(\mathbf{z}) \rightarrow p_{\text{data}}(\mathbf{x})$$

$$D(G(\mathbf{z})) \rightarrow 0.5$$

# Generator Output



# GAN Zoo



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

# Case Study: Deep Convolutional GAN

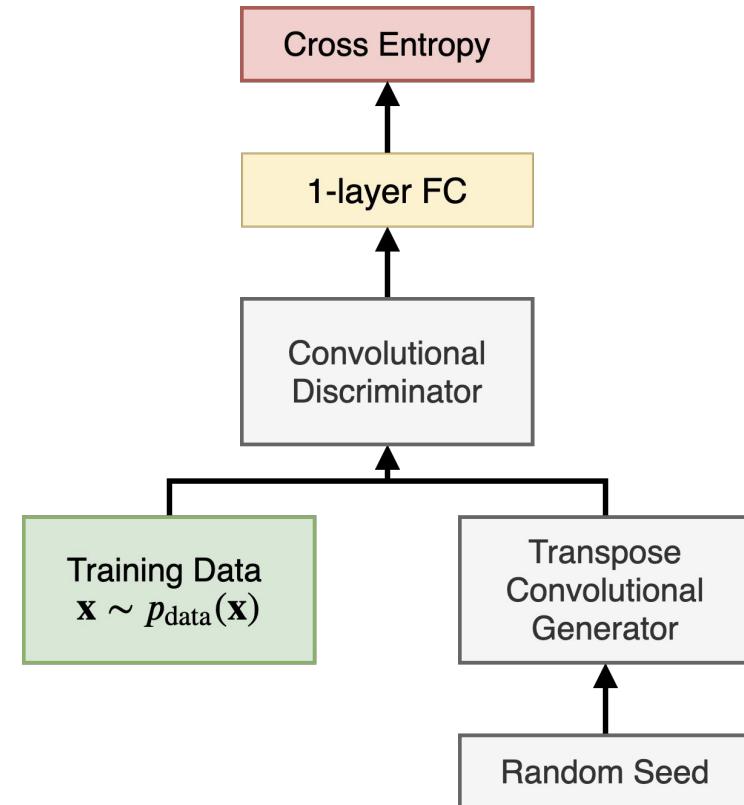
## Convolutional Discriminator

Strides instead of pooling

Global average instead of deep  
FC classifier

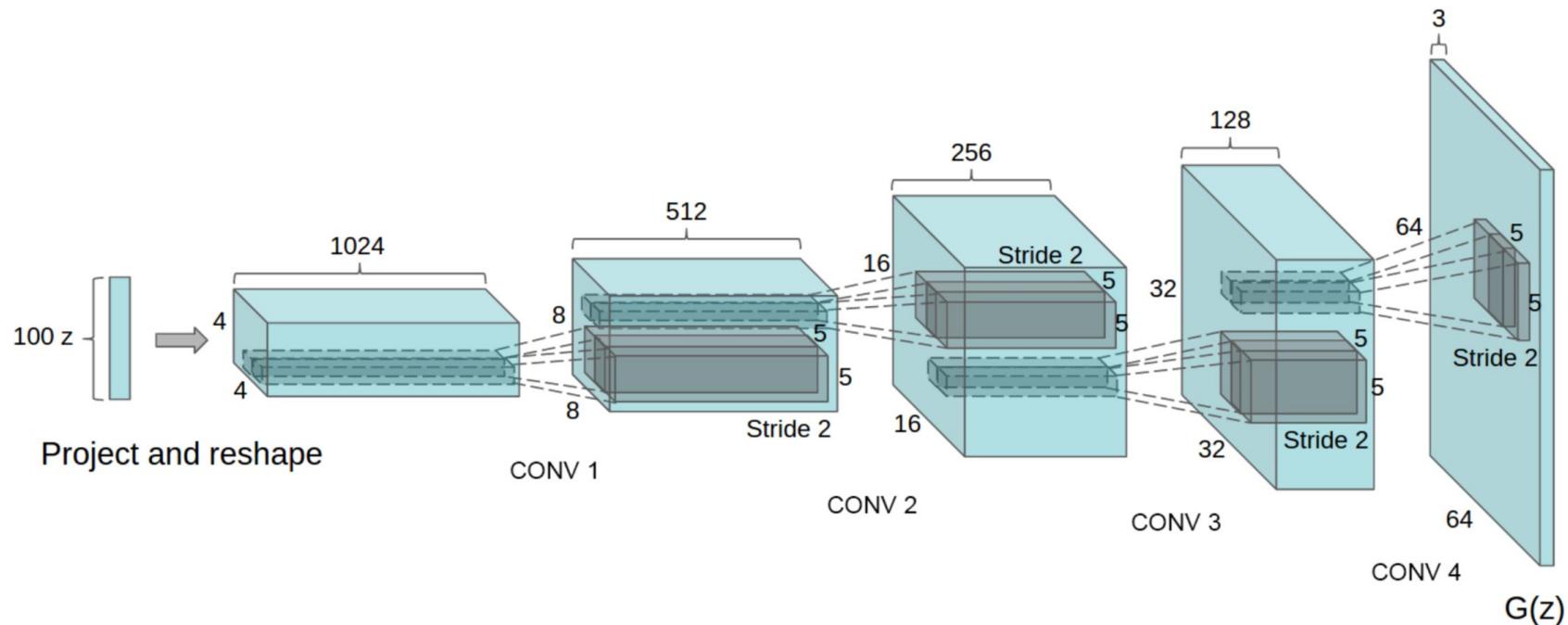
## Convolutional Generator

Transpose convolutions



Radford, A., et al. (2016)

# DCGAN: Generator



Radford, A., et al. (2016)

# Transpose Convolution

Input: 2x2

2	-1
0	1

Filter: 2x2



10	20
30	40

Output: 3x3

20	40	
60	80	

# Transpose Convolution

Input: 2x2

2	-1
0	1

Filter: 2x2



10	20
30	40

Output: 3x3

20	30	-20
60	50	-40

# Transpose Convolution

Input: 2x2

2	-1
0	1

Filter: 2x2



10	20
30	40

Output: 3x3

20	30	-20
70	70	-40
0	0	

# Transpose Convolution

Input: 2x2

2	-1
0	1

Filter: 2x2



10	20
30	40

Output: 3x3

20	30	-20
70	80	-30
0	30	40

Also called fractionally-strided convolution.

Linear algebra details: <https://arxiv.org/pdf/1603.07285.pdf>

# DCGAN: Generated Samples



Trained on LSUN bedroom images.

Radford, A., et al. (2016)

# DCGAN: Latent Space

Interpolation between  $z$  vectors with and without windows in the generated image.



# DCGAN: Latent Space



-

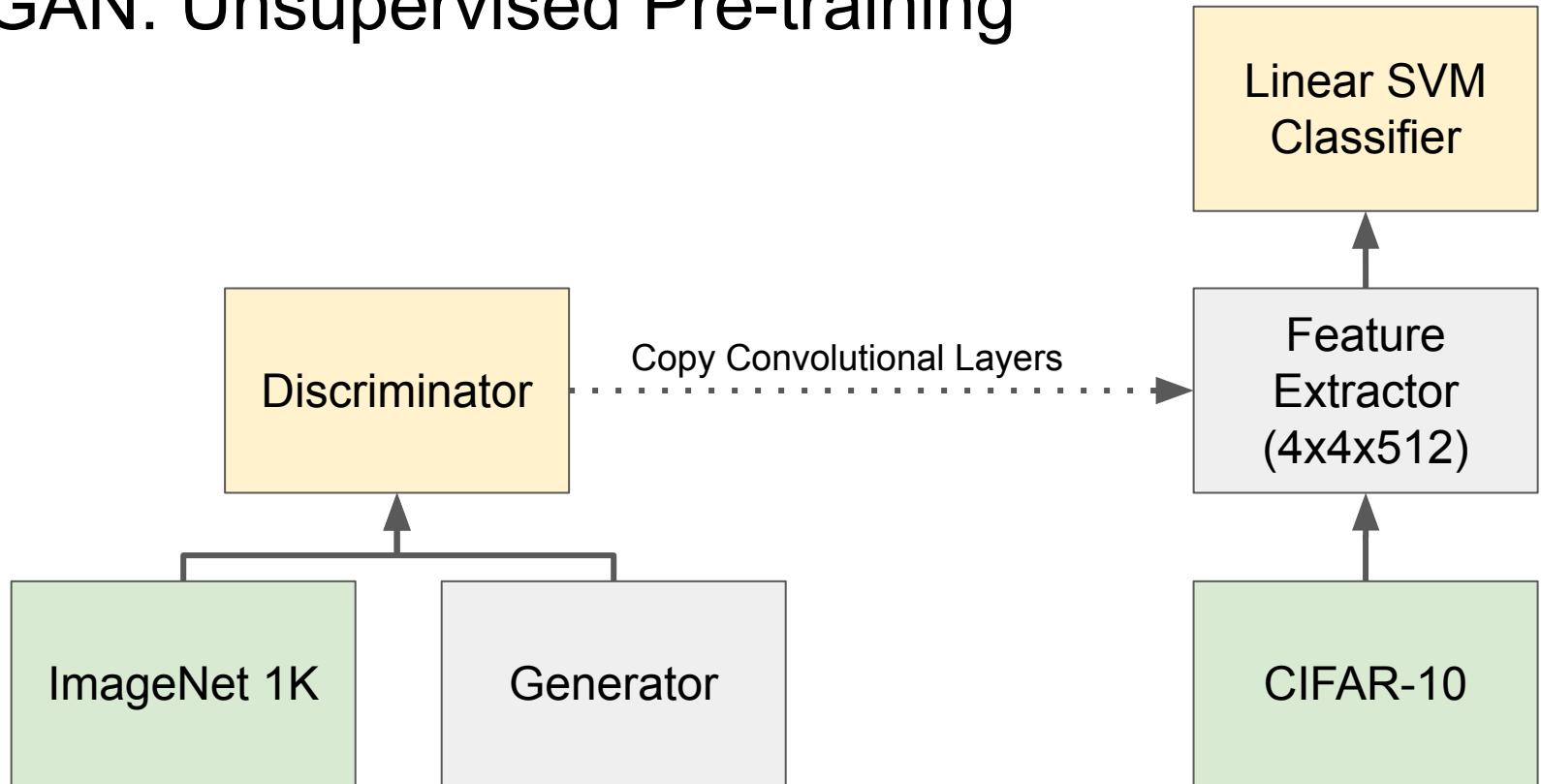
+

=



Radford, A., et al. (2016)

# DCGAN: Unsupervised Pre-training



# Case Study: SuperResolution GAN

## Supervised Task

Given a low-resolution image,  
**reconstruct the high-resolution**  
image.

## Prior Art

Residual Transpose CNN with  
MSE loss has shown great  
performance on this task but the  
**reconstructions are blurry.**

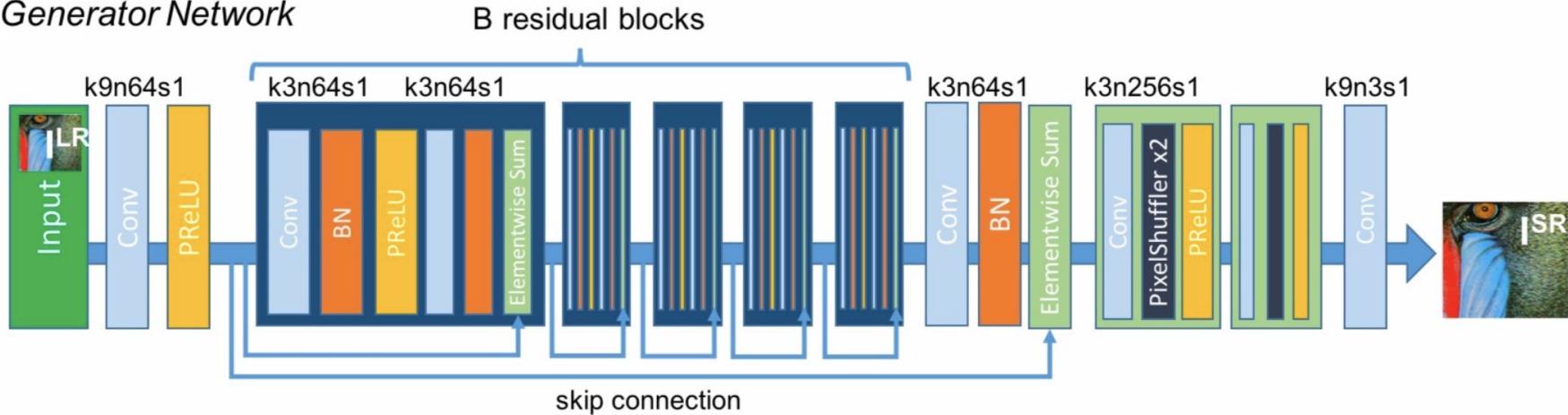


Bicubic

Original

# SRGAN: Generator CNN

*Generator Network*



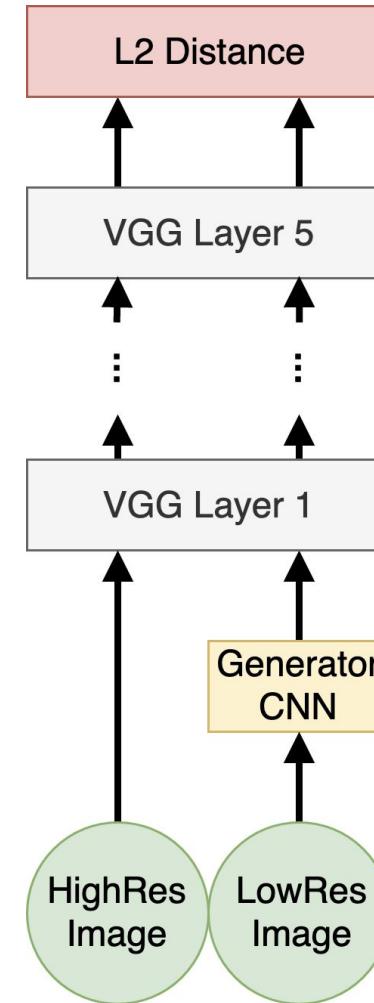
Similar to **ResNet**, but uses **Transposed Convolutions** to increase the spatial dimensions.

# SRGAN: VGG Loss

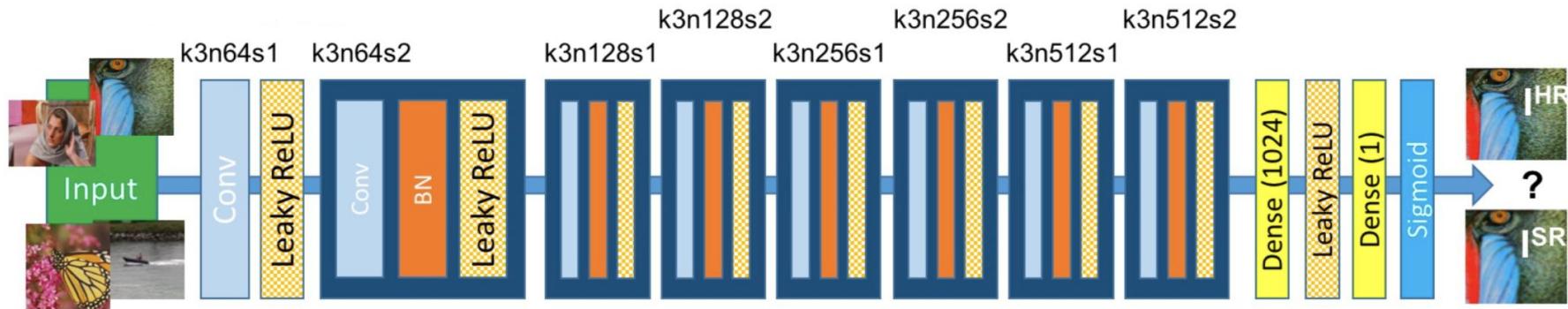
Comparison of reconstructed vs the original image is done using **pre-trained VGG19**.

Both images go through several convolutional layers of VGG19, and the **final activation volumes** are compared.

The loss is the **L2 distance** between the activation volumes of the reconstruction vs the original image.



# SRGAN: Discriminator CNN



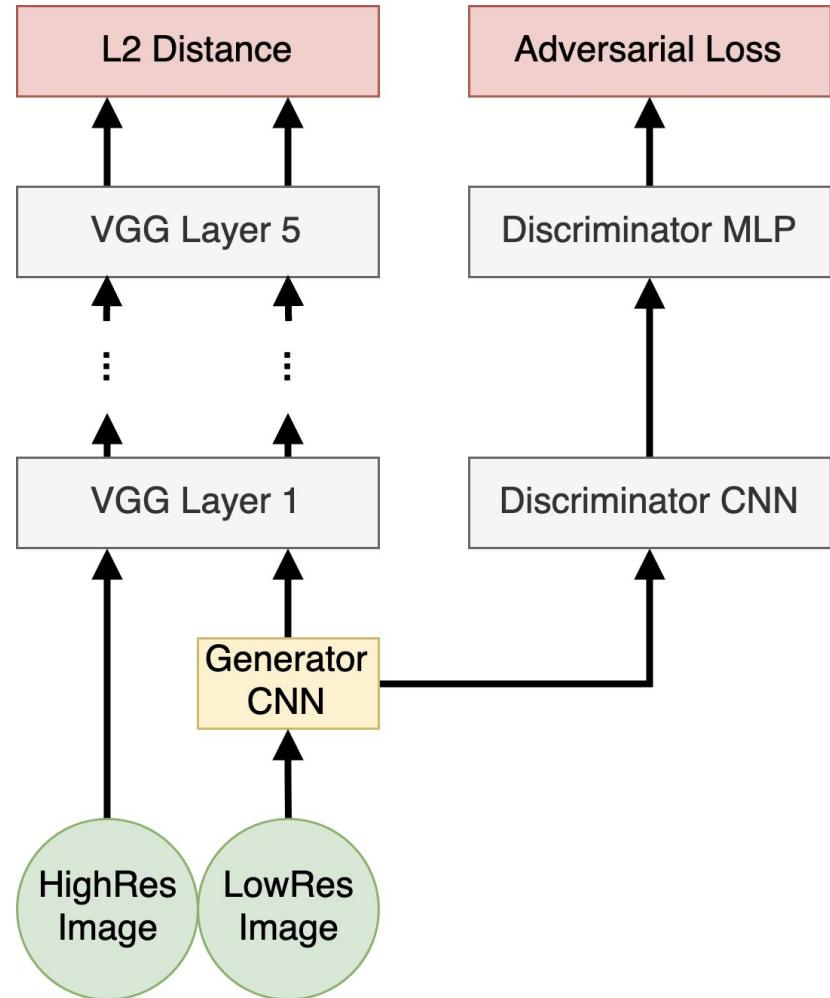
1. Fully-convolutional CNN
2. MLP Classifier
3. Binary Cross Entropy.

# SRGAN: Forward Pass

## Dual Loss

Weighted sum of **Adversarial Loss** and **VGG Activation Loss**

Authors find the AL+VGG combo to work better than individual losses or MSE.

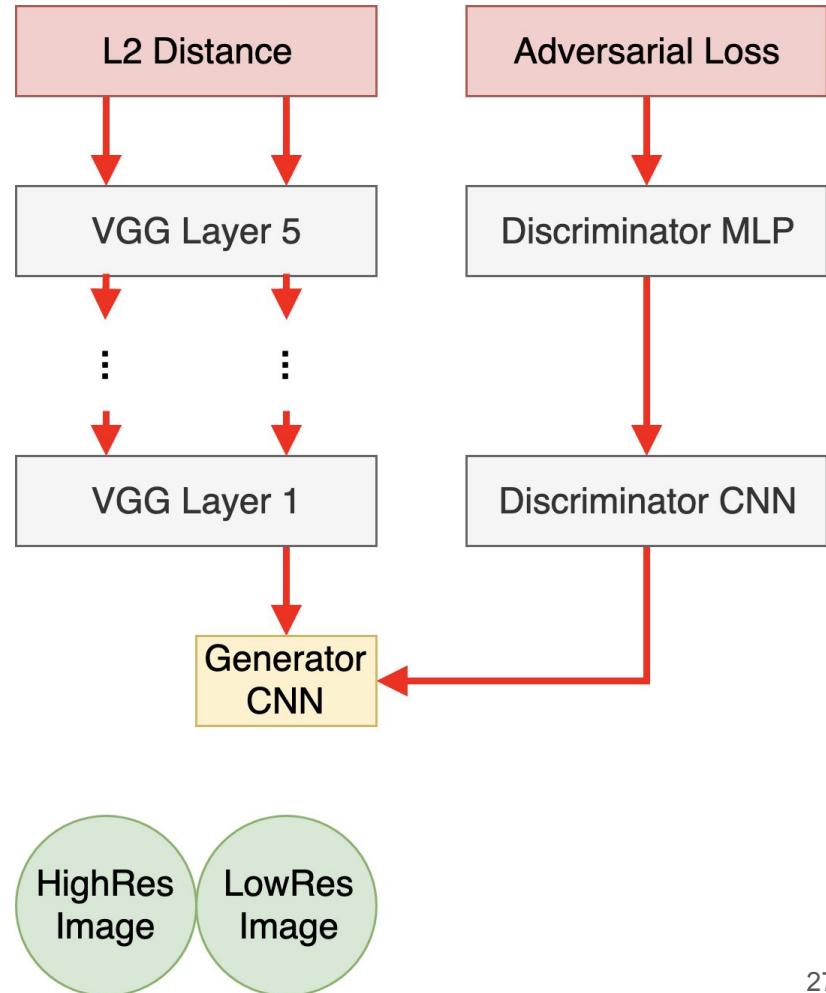


# SRGAN: Backprop

## Dual Loss

Weighted sum of **Adversarial Loss** and **VGG Activation Loss**

Authors find the AL+VGG combo to work better than individual losses or MSE.



# SRGAN: Examples



Bicubic



MSE



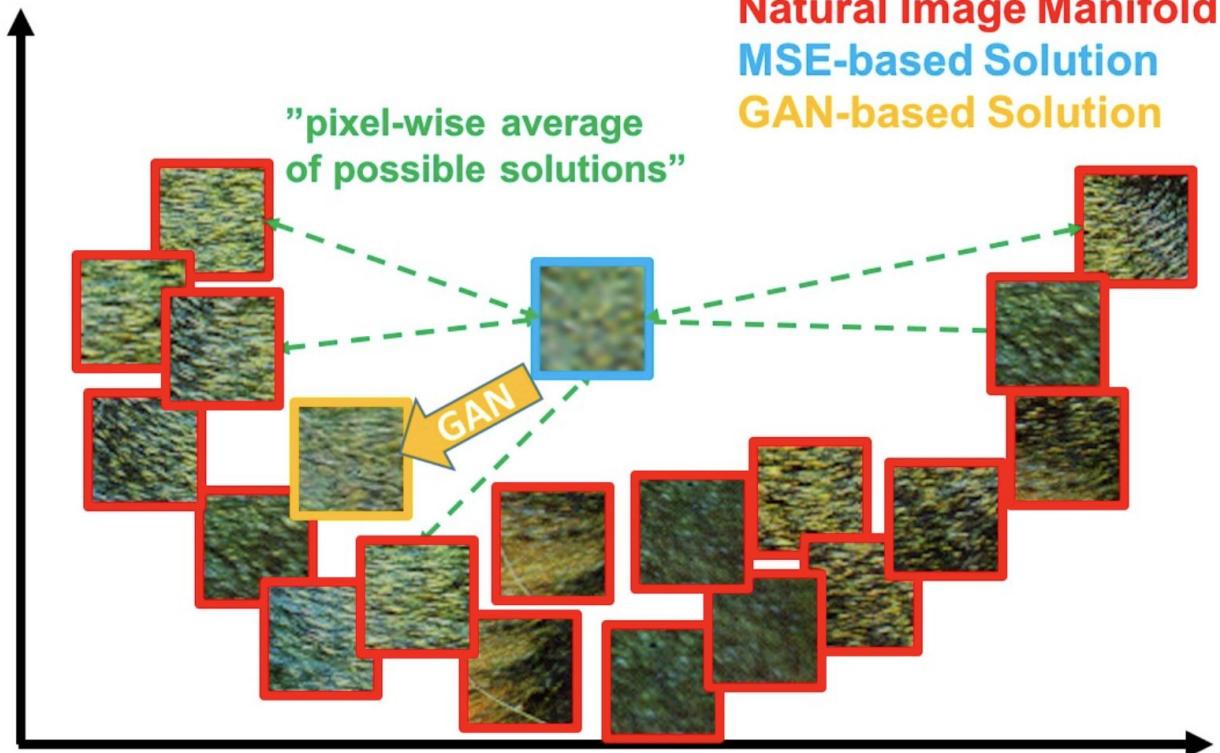
VGG+AL



Original

# SRGAN: What's wrong with MSE?

A mean is often a bad approximator of any of the samples from multi-modal distributions.



# Case Study: BigGAN



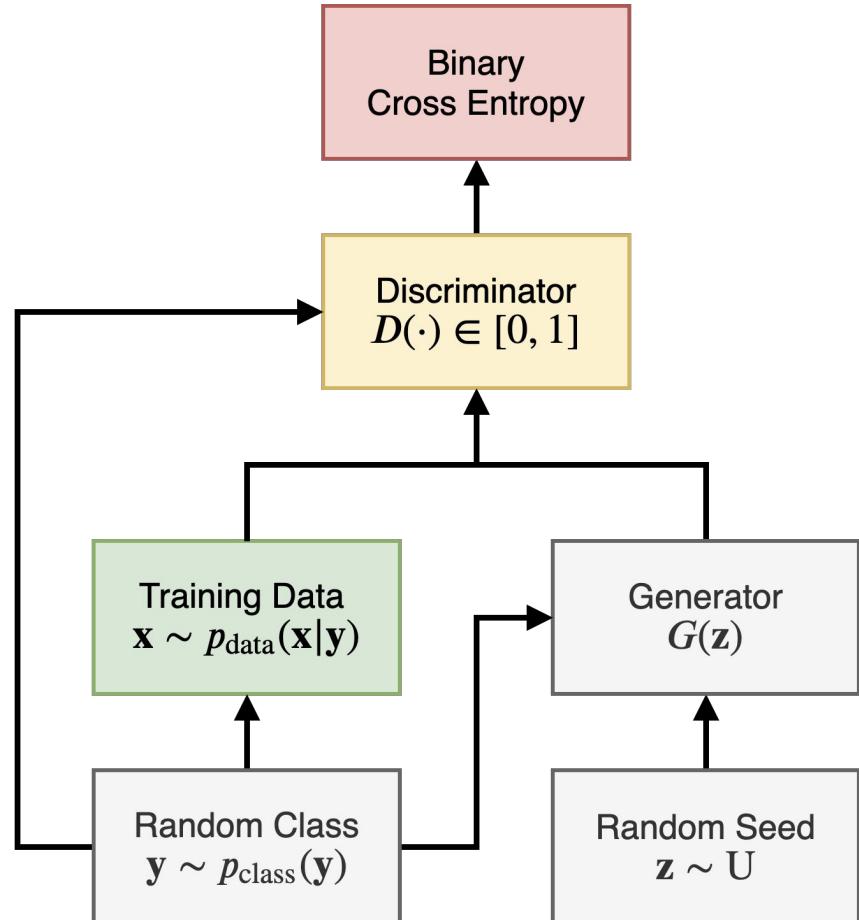
Which one of these images is real?

# BigGAN

Class-conditional generation

High-resolution images  
512 x 512

Quality vs diversity trade-off  
via noise truncation

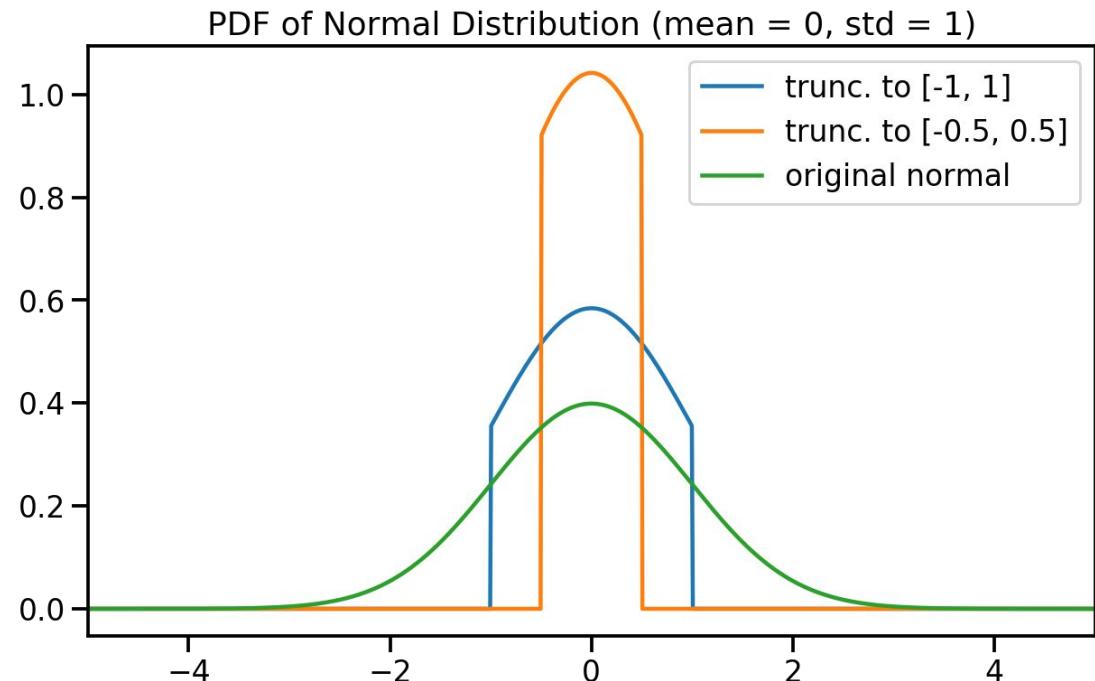


# BigGAN: Noise Truncation

**Properties of  
symmetrically truncated  
normal distribution:**

1. excludes low-probability events
2. preserves mean
3. lowers variance

Enables trade-off of diversity for quality.



# BigGAN: Diversity

Larger noise space allows to generate diverse sets of same-class images.

Since low-probability noise was less frequently checked by the discriminator, its quality is lower.

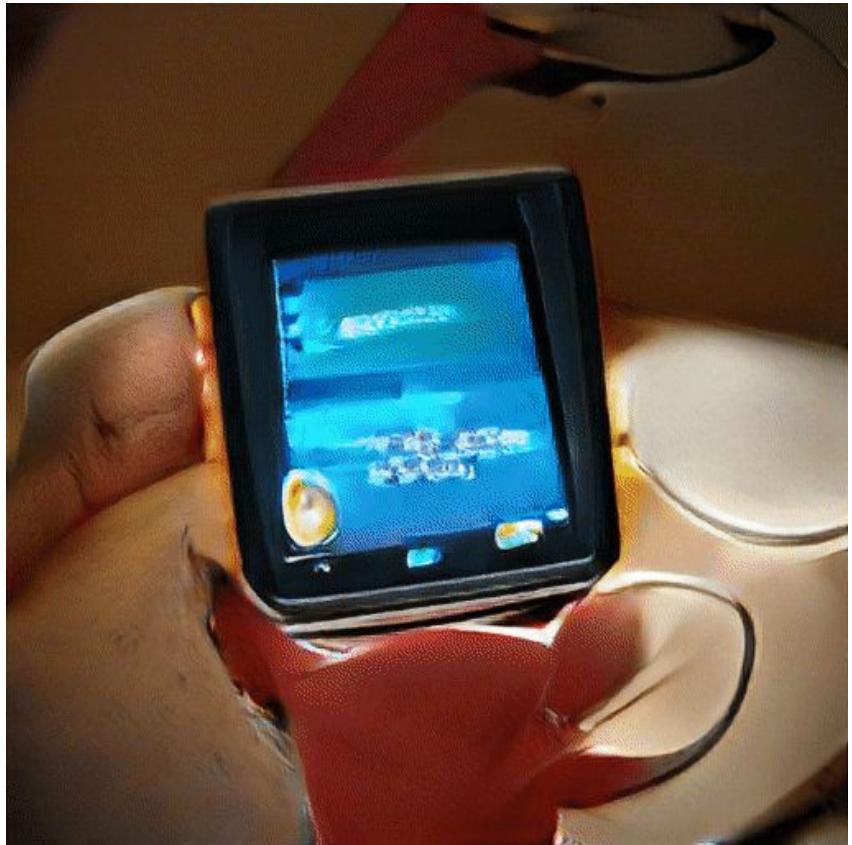
Diversity and quality are a trade off.



# BigGAN: Class Interpolation

1. Randomly choose a noise vector
2. Choose two classes  
(terrier and tiger in this example)
3. Linearly interpolate  
one-hot-encoded class vectors
4. Feed the constant-noise vector, and  
the class vectors to the generator.





iPod to ice cream



Train to golf cart

# Demo

## AWS SageMaker + BigGAN.ipynb

# Assignment 4

1. BigGAN model
  - a. Optional reading: <https://arxiv.org/abs/1809.11096>
2. Explore the relationship between truncation and image properties
  - a. See the assignment template notebook for details
3. Deliverables:
  - a. Jupyter notebook with task solutions
  - b. Fully executed and saved with images
4. Due by EOD October 28th

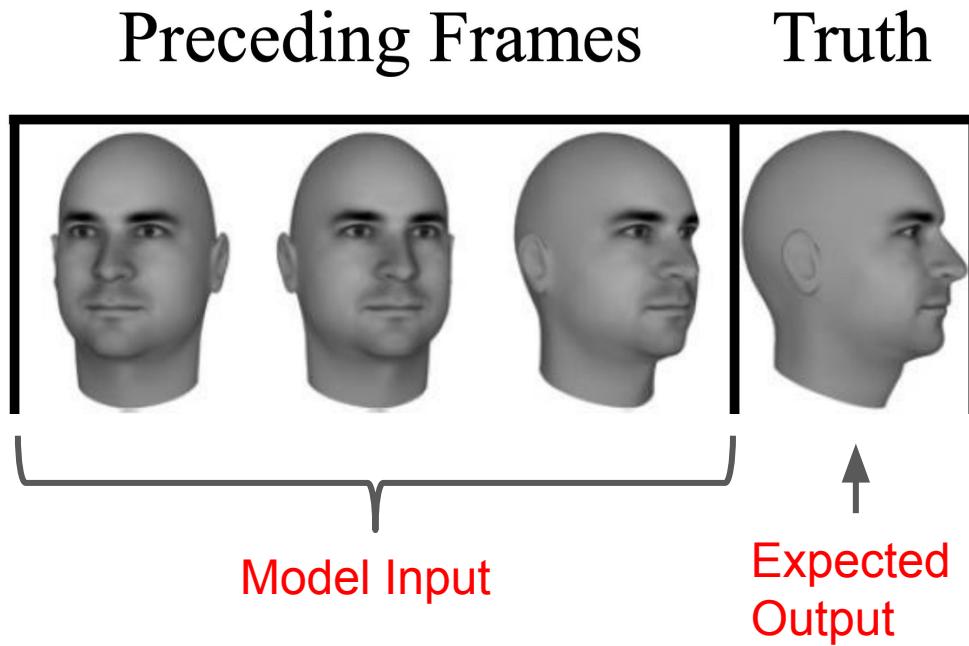
# Case Study: Predictive Generative Network

## Task

Predict the **next frame in a sequence** given last 3 frames

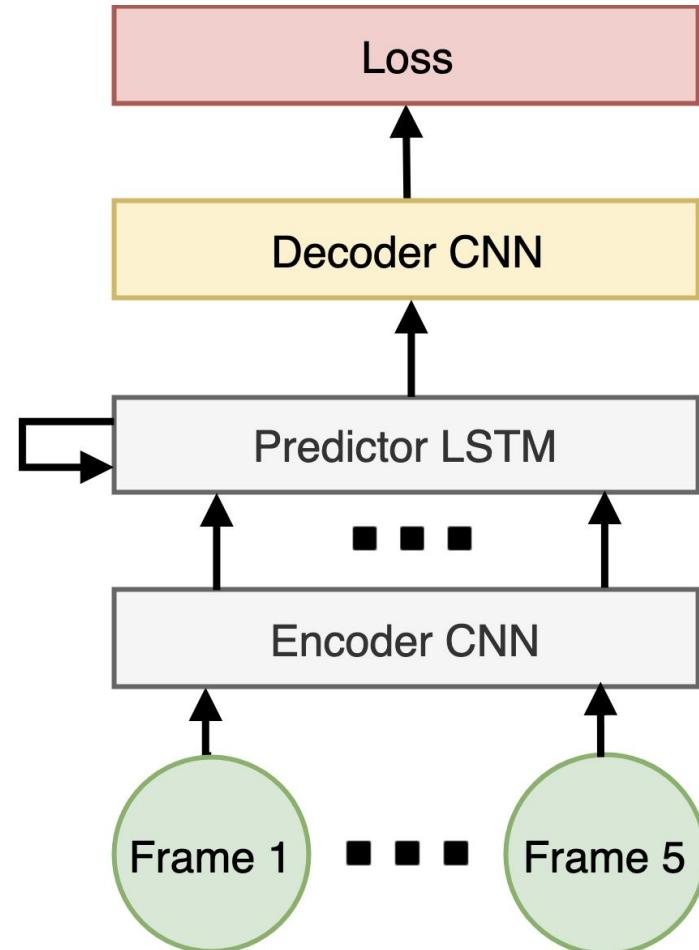
Every sequence has a **randomly generated face** rotating around Y axis.

Initial angle and angular velocity are random across sequences.

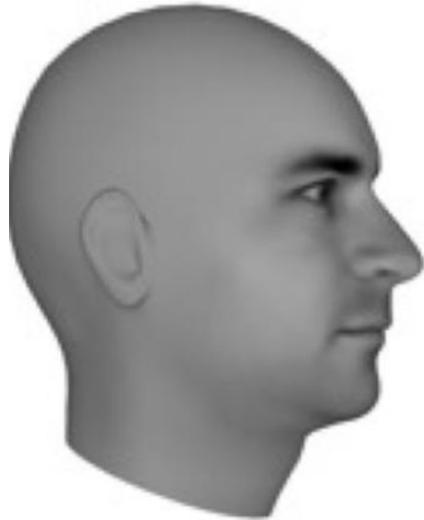


# PGN: Generator Model

1. Encoder CNN **extracts each frame's features** into a low-dimensional encoding.
2. Predictor LSTM **consumes each encoding sequentially**, and produces an encoding for the target frame.
3. Decoder CNN **converts the encoding** into the target frame.



# PGN: Mean Squared Error



Target



Prediction



Target



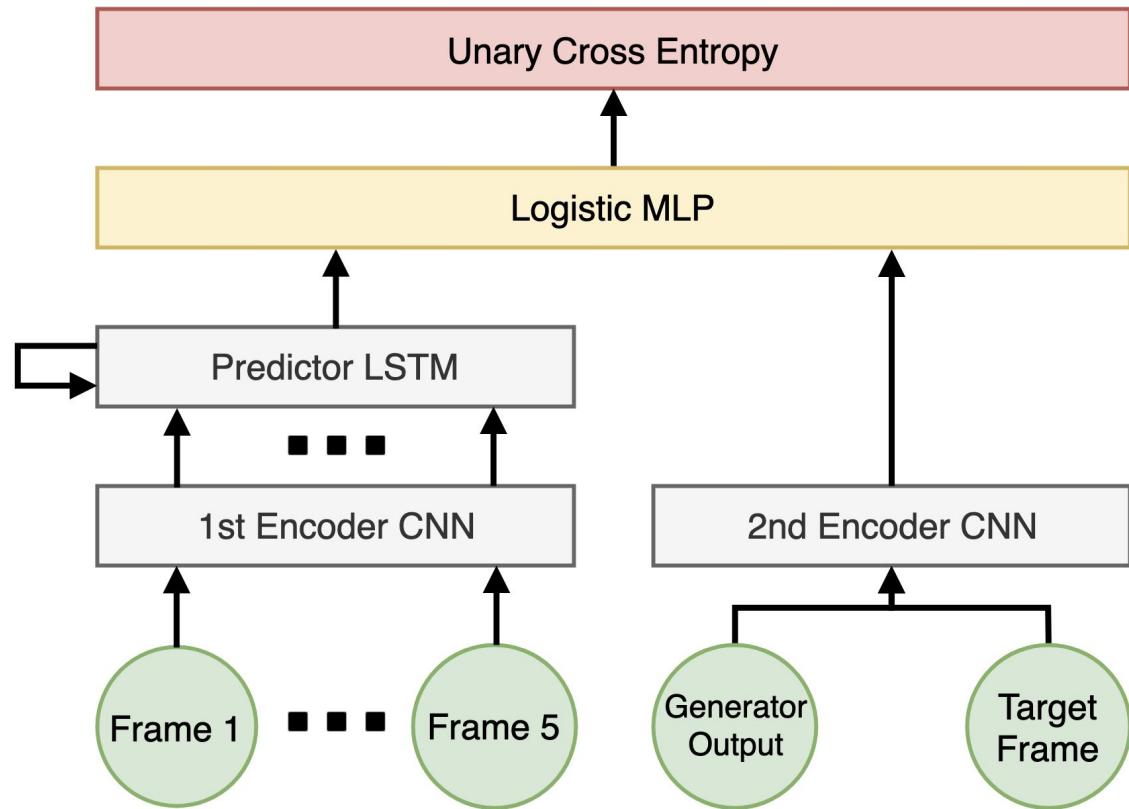
Prediction

# PGN: Discriminator

1st Encoder CNN and Predictor LSTM form **conditional expectations about the target frame.**

2nd Encoder CNN extracts features from the adversarial input.

Logistic MLP determines whether the adversarial input is the true target frame.



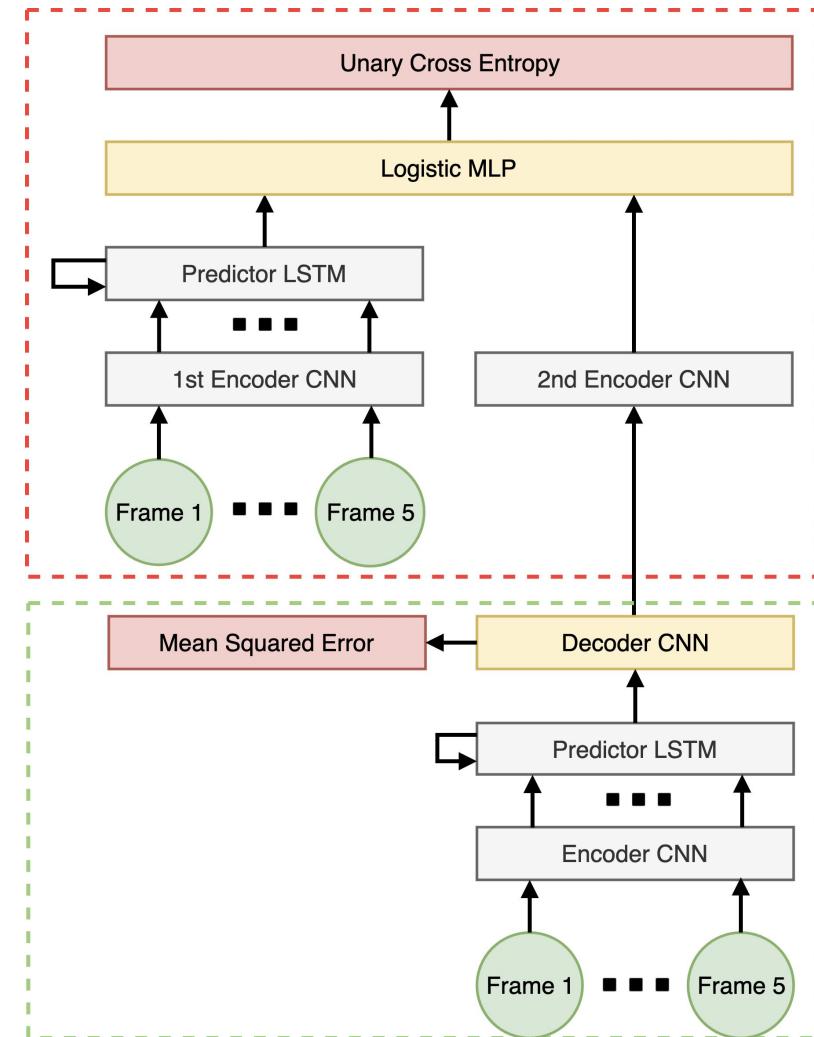
# PGN: Generator Training

The generator is trained with **dual loss**:

- MSE against the target frame
- Adversarial loss

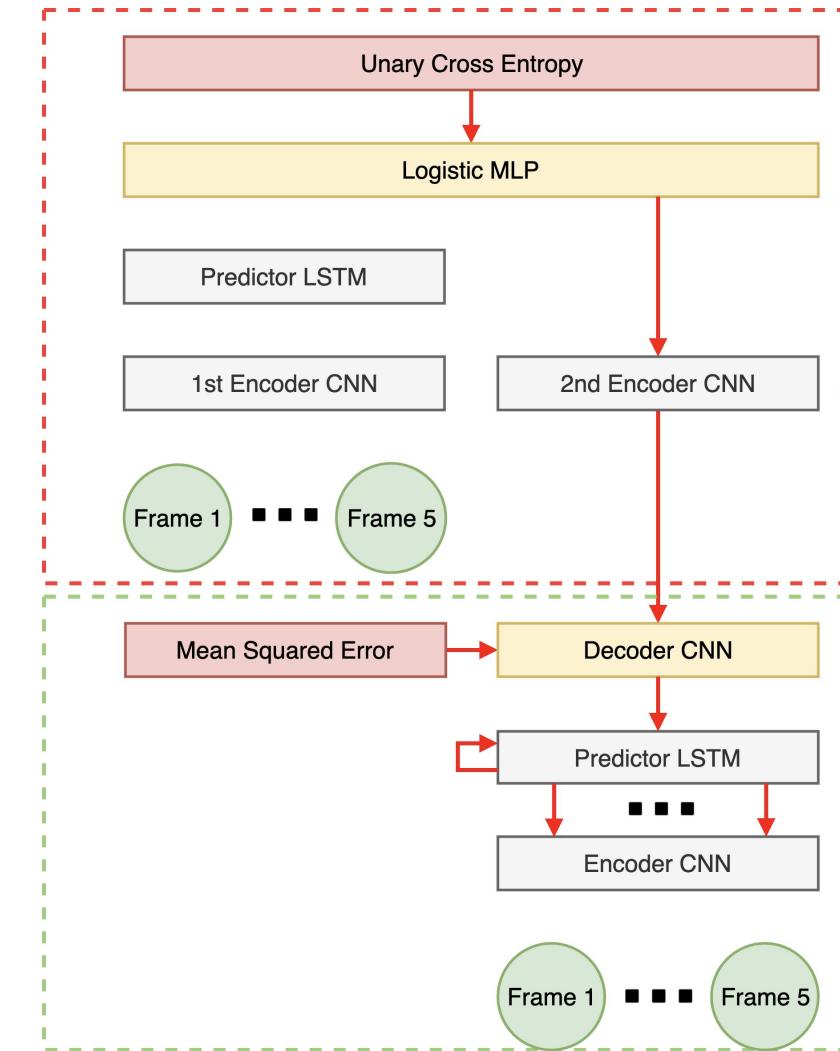
The final cost is a **weighted sum** of the two losses. Weights are hyperparams.

None of the LSTMs or CNNs share parameters across layers.



# PGN: Generator Backprop

During backward propagation gradient flows through only a part of the discriminator.



# PGN: Dual Loss

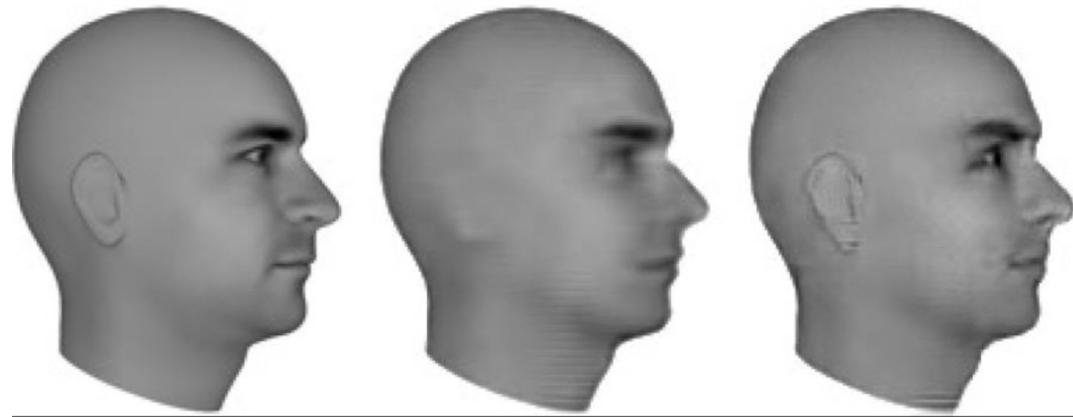
Model trained with Dual Loss outputs **sharp frames** which more closely resemble the target frames.

Read the paper for more predictive task examples.

Truth

MSE

AL/MSE



# Text to Synthetic Image

this small bird has a pink breast and crown, and black primaries and secondaries.



this magnificent fellow is almost all black with a red crest, and white cheek patch.



# Case Study: StackGAN

Multi-stage  
generator/discriminator:  
1. High-level sketch  
2. Fine details

Embeddings for text  
conditioning but no  
recurrent layers

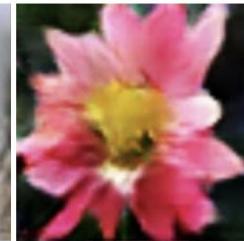
Zhang, H., et al. (2016)

This bird is white with some black on its head and wings, and has a long orange beak

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

(a) StackGAN  
Stage-I  
64x64  
images



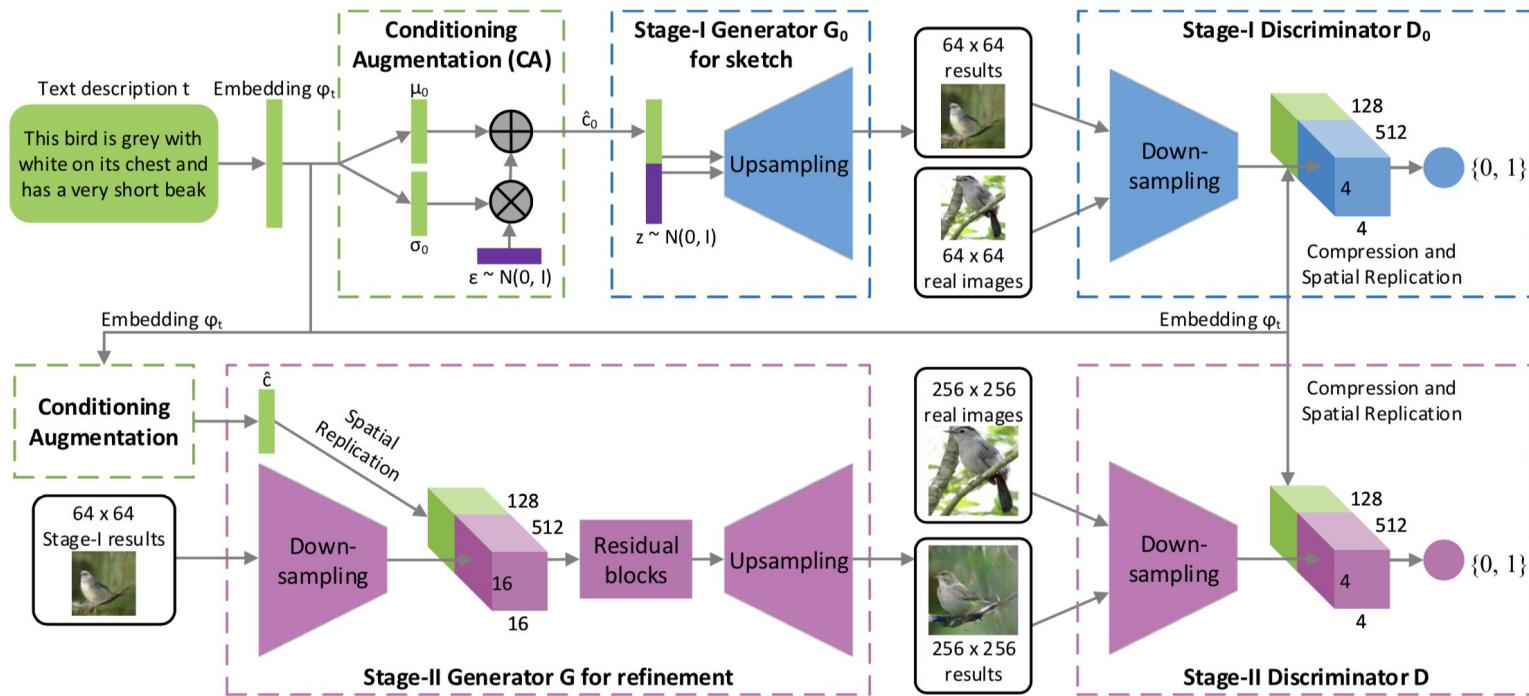
(b) StackGAN  
Stage-II  
256x256  
images



(c) Vanilla GAN  
256x256  
images



# StackGAN: Architecture



# References

1. Generative Adversarial Nets (Goodfellow, I., et al. 2014)
2. Generative Adversarial Text to Image Synthesis (Redd, S., et al., 2016)
3. Large Scale GAN Training for High Fidelity Natural Images Synthesis (Brock, A., et al., 2019)
4. NIPS 2016 Tutorial: Generative Adversarial Networks (Goodfellow, I., 2016)
5. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (Ledig, C., et al., 2017)
6. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks (Zhang, H., et al., 2017)
7. Unsupervised Learning of Visual Structure using Predictive Generative Networks (Lotter, W., et al., 2016)
8. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford, A., Metz, L., 2016)

# Thank you

Final Q&A Time