Generating Fake Images

Adrian Spataru

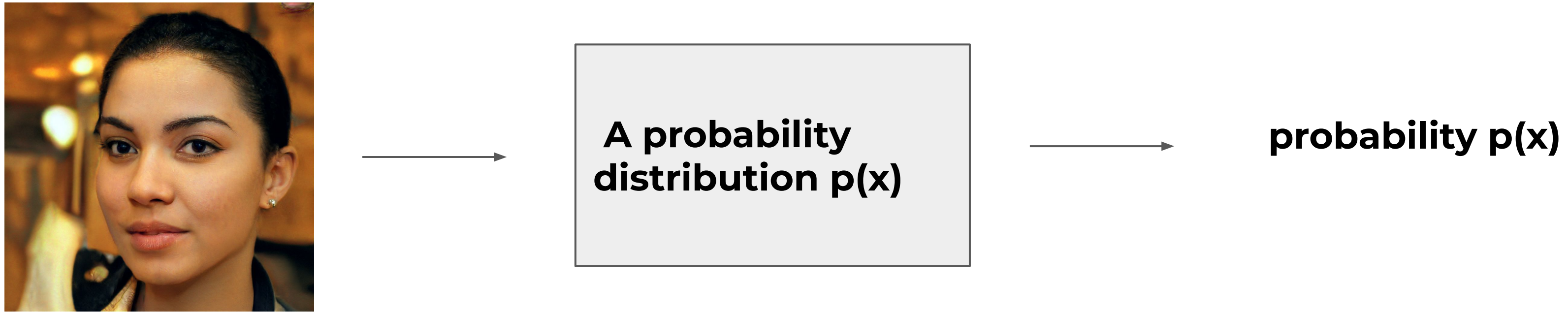
Machine Learning Graz

Statistical Generative Models

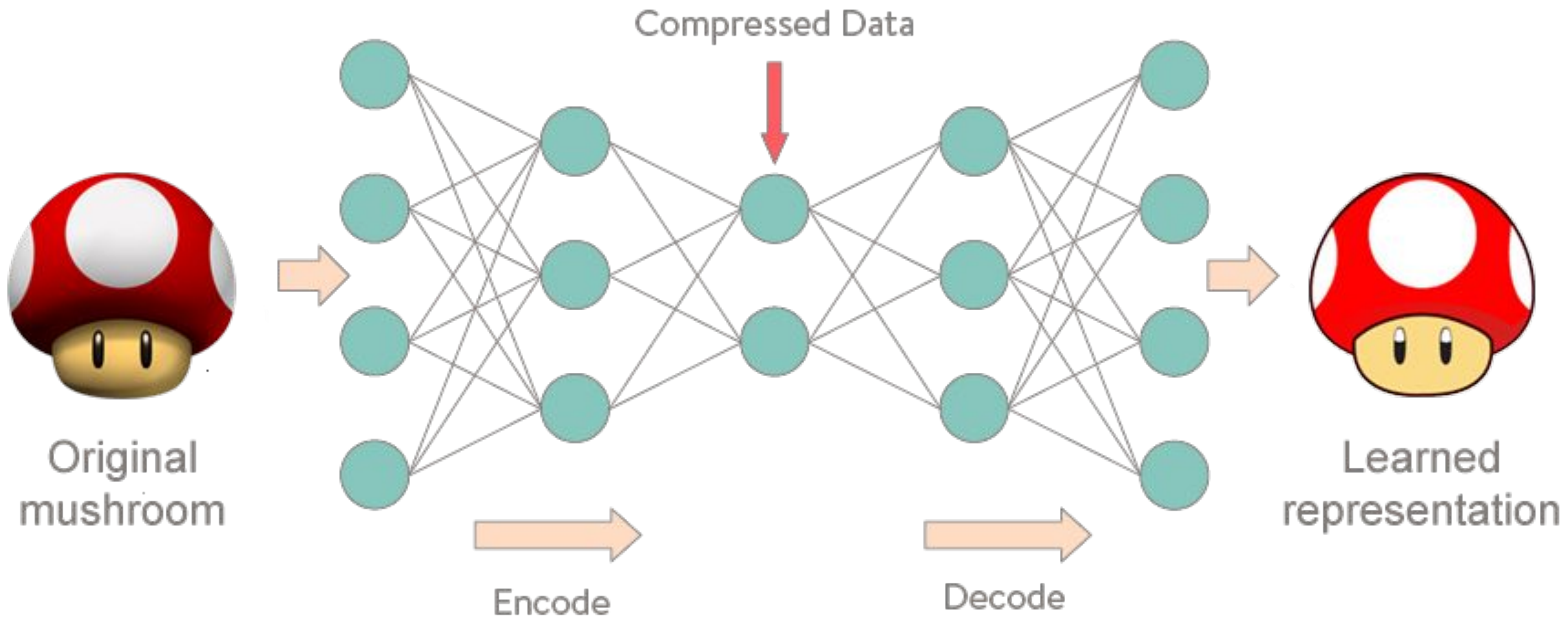
A statistical generative model is a probability distribution p(x)

- Data: samples (eg. Images of people)
- Prior Knowledge: parametric form (Gaussian?), loss function etc.

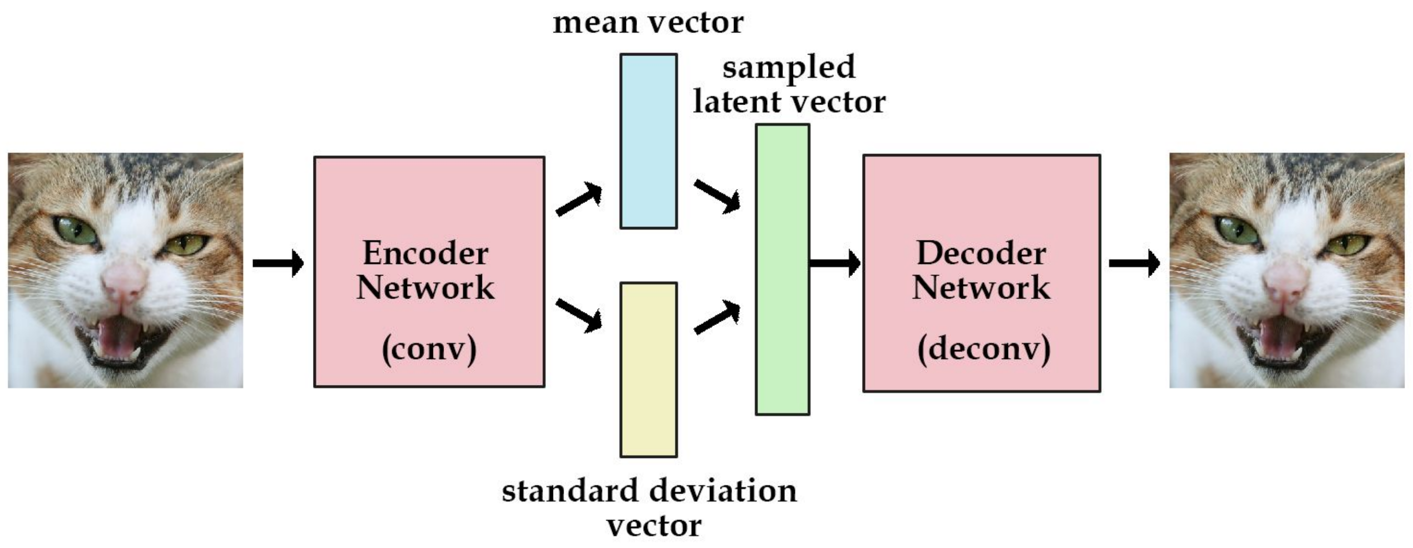
It is generative because sample from p(x) generates new images



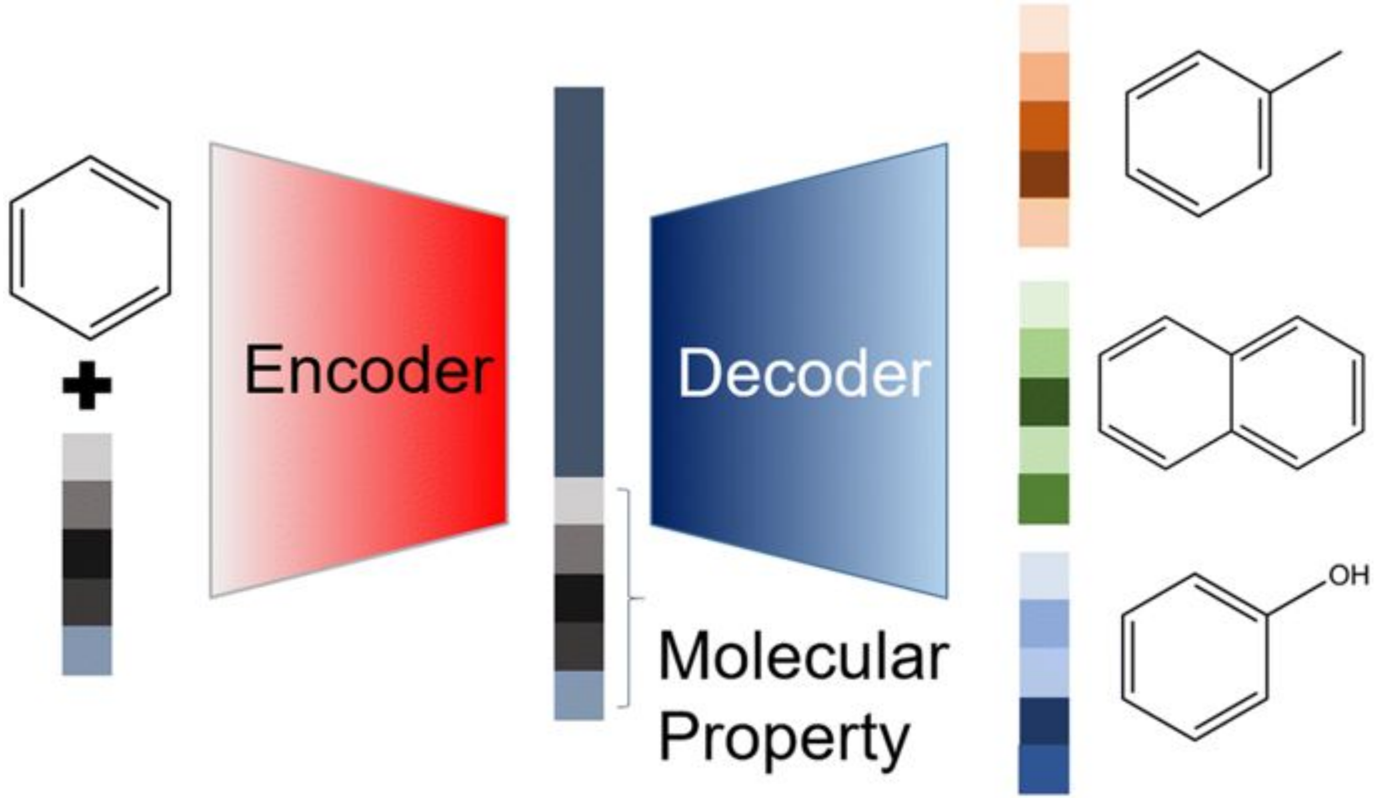
Autoencoder



Variational Autoencoder



Conditional Variational Autoencoder



PixelRNN

- See pixels as a sequence
- Use a Recurrent Neural Network to predict next pixel

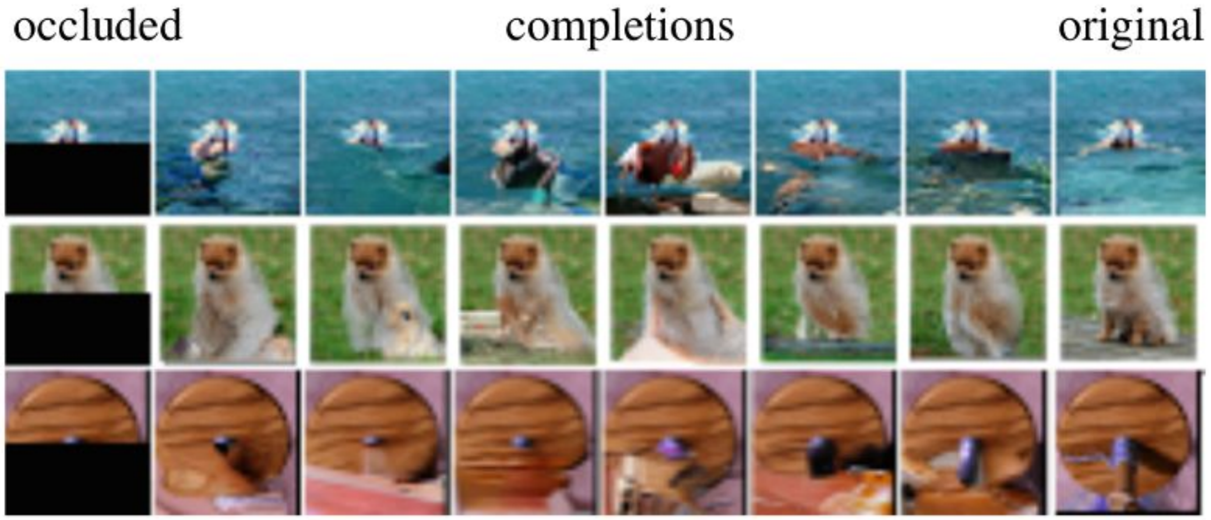
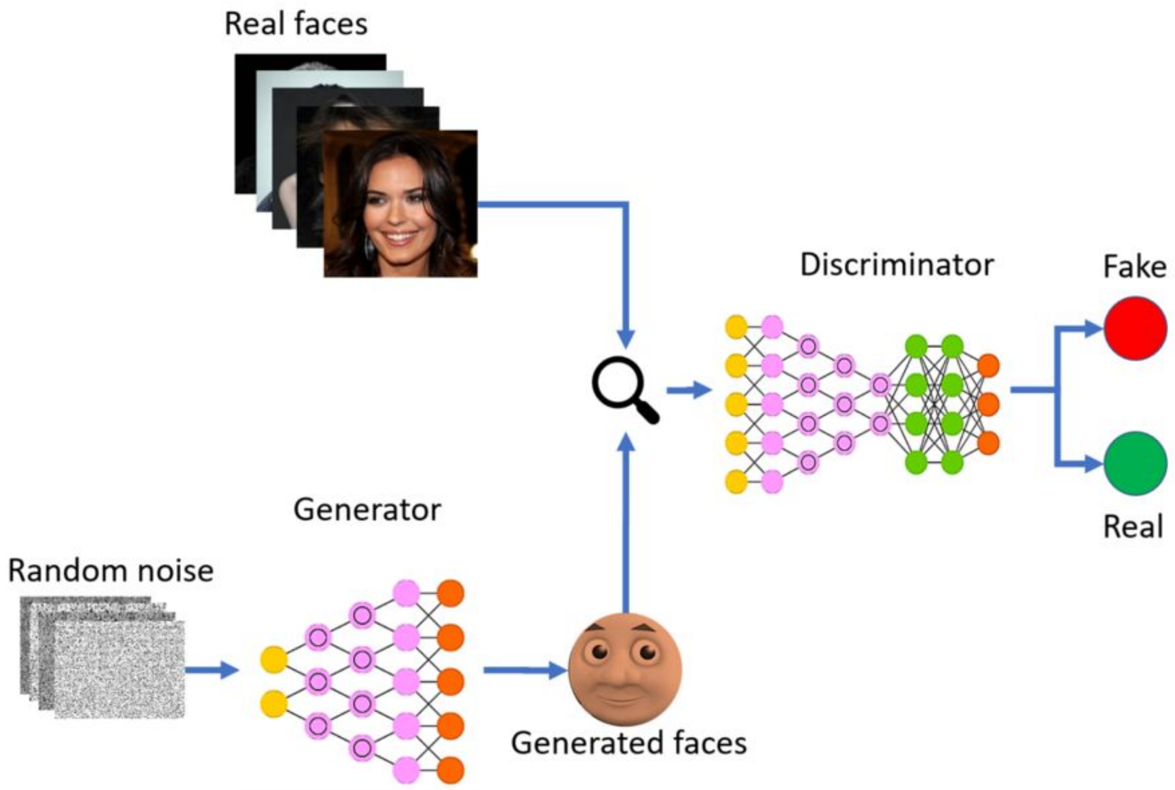
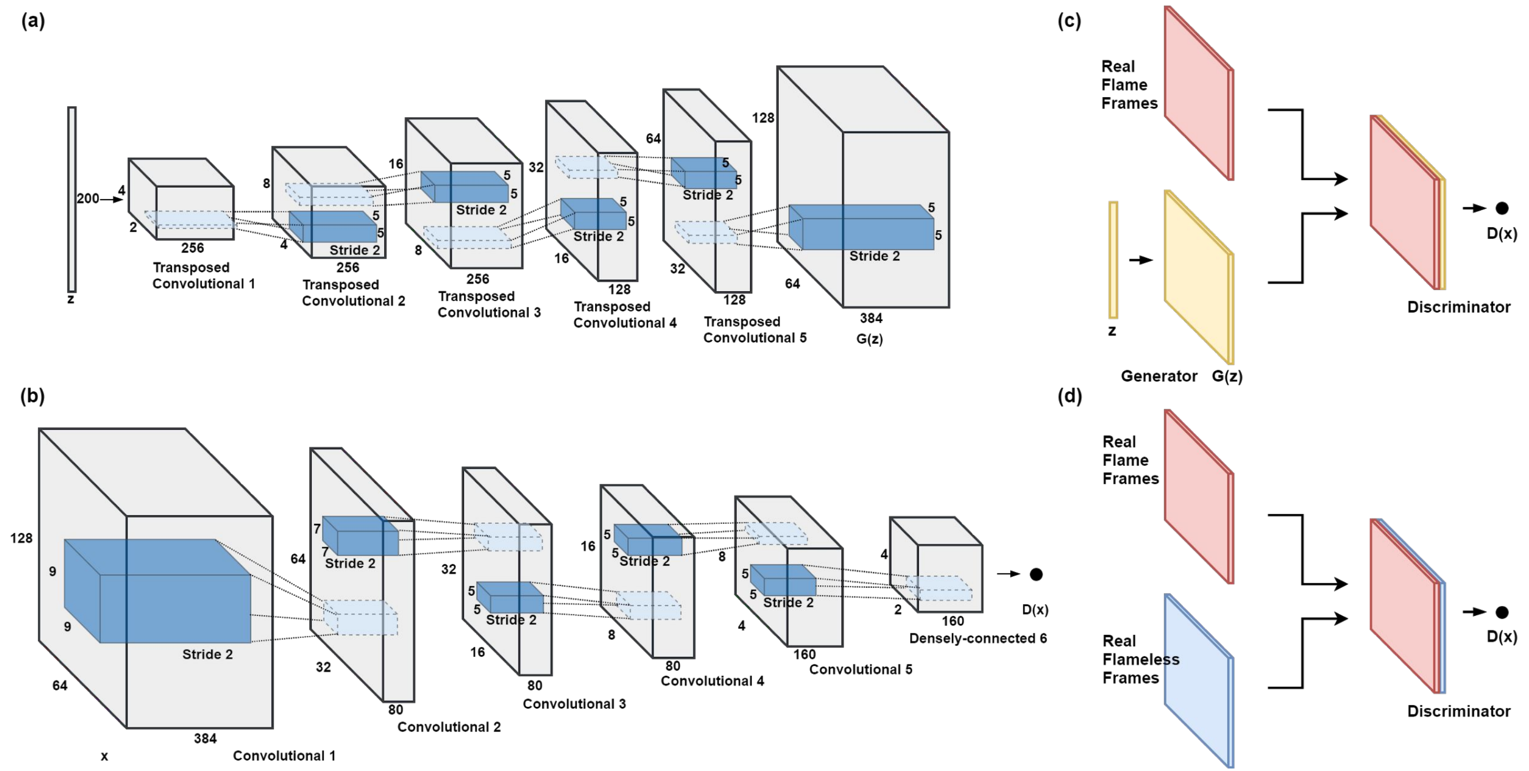


Figure 1. Image completions sampled from a PixelRNN.

Generative Adversarial Networks



DCGAN


(a)
z
200
4
2
256
Transposed Convolutional 1
8
4
5
5
Stride 2
256
Transposed Convolutional 2
16
8
5
5
Stride 2
256
Transposed Convolutional 3
32
16
5
5
Stride 2
128
Transposed Convolutional 4
64
32
5
5
Stride 2
128
Transposed Convolutional 5
128
64
5
5
Stride 2
384
G(z)
(b)
128
9
9
64
Stride 2
384
x
Convolutional 1
64
7
7
Stride 2
32
80
Convolutional 2
32
5
5
Stride 2
16
80
Convolutional 3
16
5
5
Stride 2
8
80
Convolutional 4
8
5
5
Stride 2
4
160
Convolutional 5
4
2
160
Densely-connected 6
D(x)
(c)
Real Flame Frames
z
Generator
G(z)
Discriminator
D(x)
(d)
Real Flame Frames
Real Flameless Frames
Discriminator
D(x)

DCGAN Techniques

- Use transposed convolution for upsampling.
- Eliminate fully connected layers.
- Use Batch normalization except the output layer for the generator and the input layer of the discriminator.
- Use ReLU in the generator except for the output which uses tanh.
- Use LeakyReLU in the discriminator.

Issues with GANs

- Setting up failure and bad initialization
- Problems with perspective
- Problems with global structures

Mode Collapse

- the discriminator essentially “wins” the game
- training gradients for the generator become less and less useful
- This happens when it generates the “same” sample all the time.
- Bigger the input/target size -> more likely it will happen

PROGAN (2017)

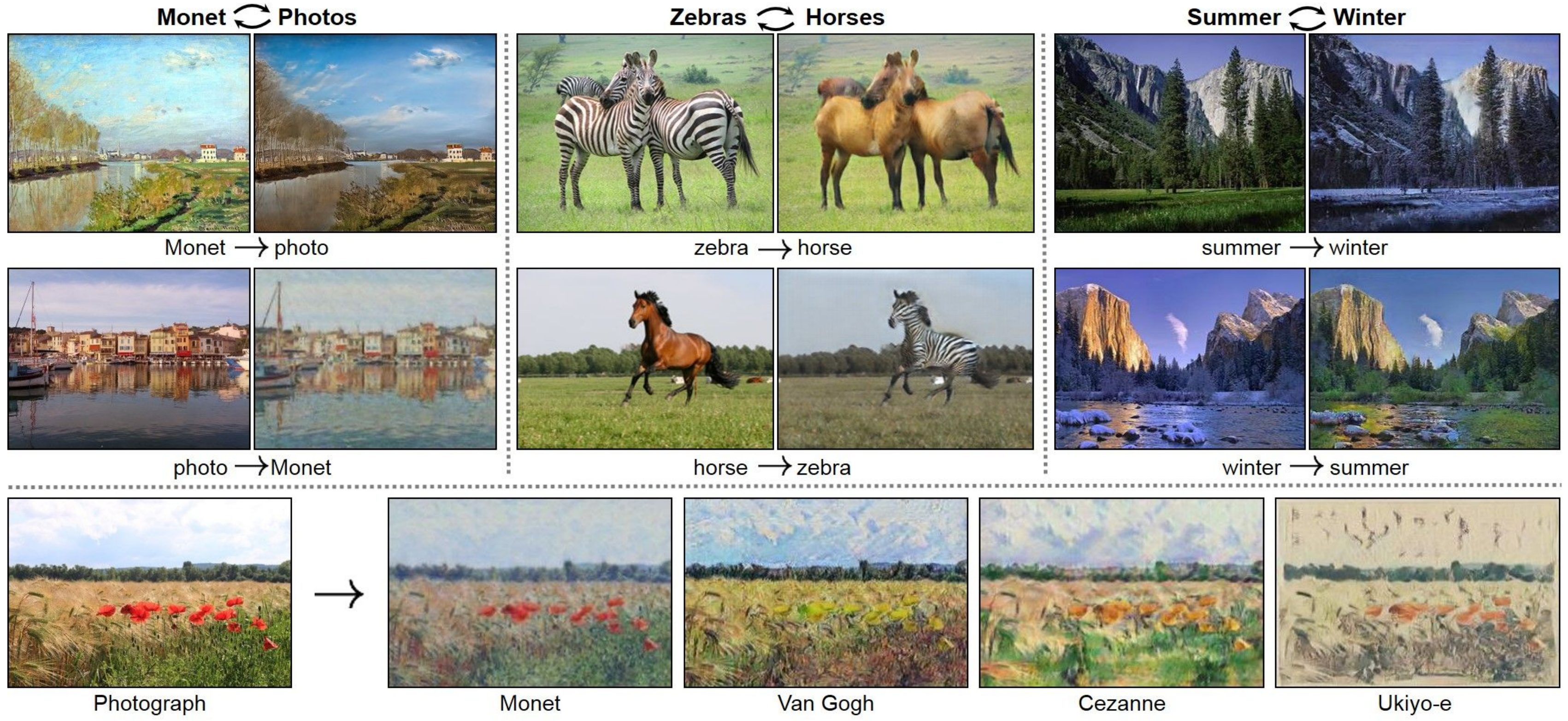
PROGAN - Intuition

- Gradually increasing the resolution
- the networks to learn a *much simpler piece* of the overall problem

Minibatch Standard Deviation

- Give the discriminator statistics on the batch data.
  - the standard deviations of the feature map pixels across the batch
- GANs tend to sample low variance samples.
- This forces the generator to give variance similar to the real data.

CycleGAN

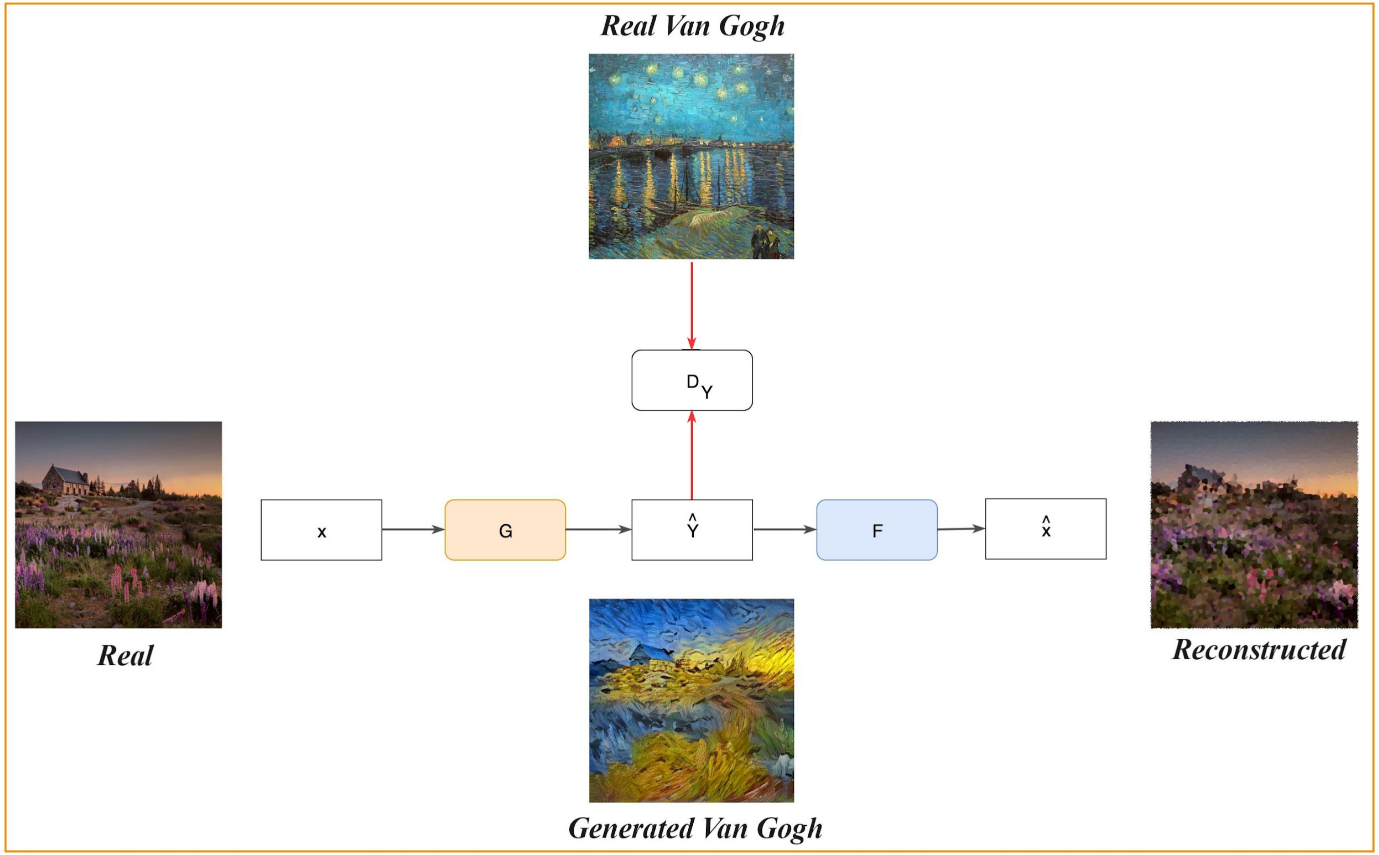

Monet ⇄ Photos
Zebras ⇄ Horses
Summer ⇄ Winter
Monet → photo
zebra → horse
summer → winter
photo → Monet
horse → zebra
winter → summer
Photograph
Monet
Van Gogh
Cezanne
Ukiyo-e

Cycle Consistency

- Transform picture from one domain to another
- A generator G to convert a real image to a target domain
- A generator F to convert from target domain to original
- A discriminator for indentifying real or fake target domain pictures
- This is the Cycle consistency loss which measures the L1-norm reconstruction cost for the real image (x → y → reconstructed x) and the Monet paintings (y → x → reconstructed y)

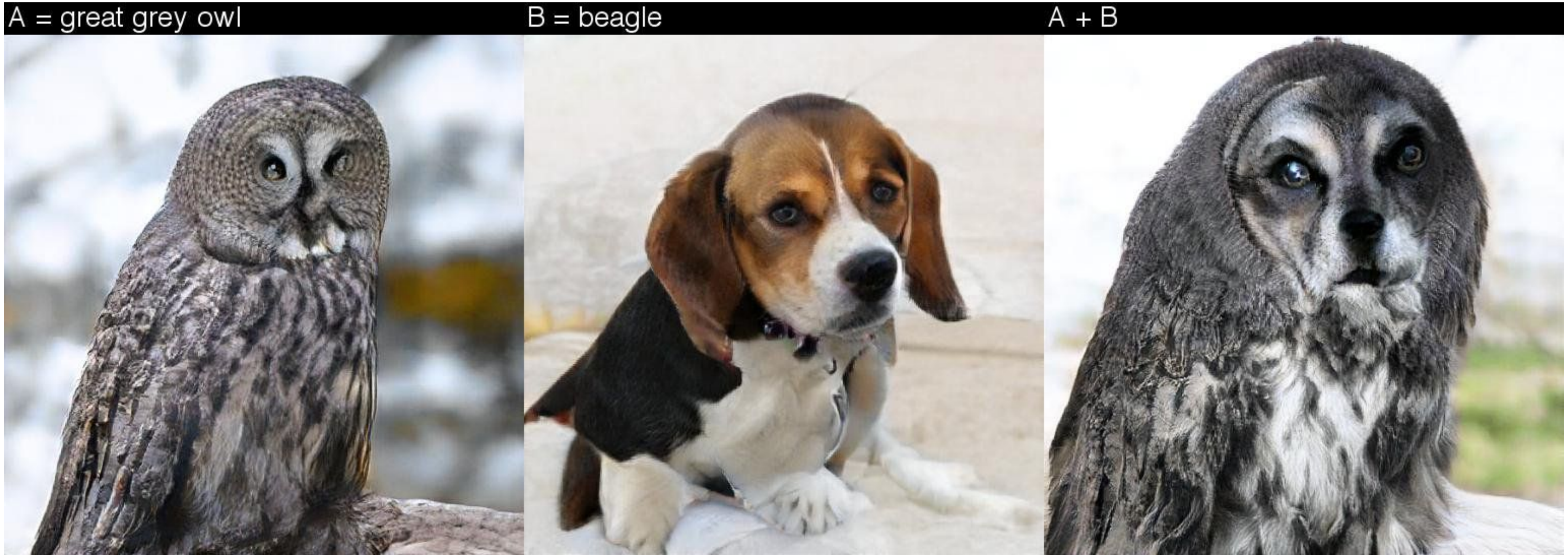
$$\mathcal{L}_{\text{cyc}}(G,F) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1]$$

$$\mathcal{L}(G,F,D_X,D_Y) = \mathcal{L}_{\text{GAN}}(G,D_Y,X,Y) + \mathcal{L}_{\text{GAN}}(F,D_X,Y,X) + \lambda\mathcal{L}_{\text{cyc}}(G,F),$$


Real Van Gogh
D_Y
Real
x
G
$\hat{Y}$
F
$\hat{x}$
Reconstructed
Generated Van Gogh

BigGAN

- Introduces techniques for scaling GANs
- Introduces hierarchical latent values with noise
- Proposed model has 350M Parameters

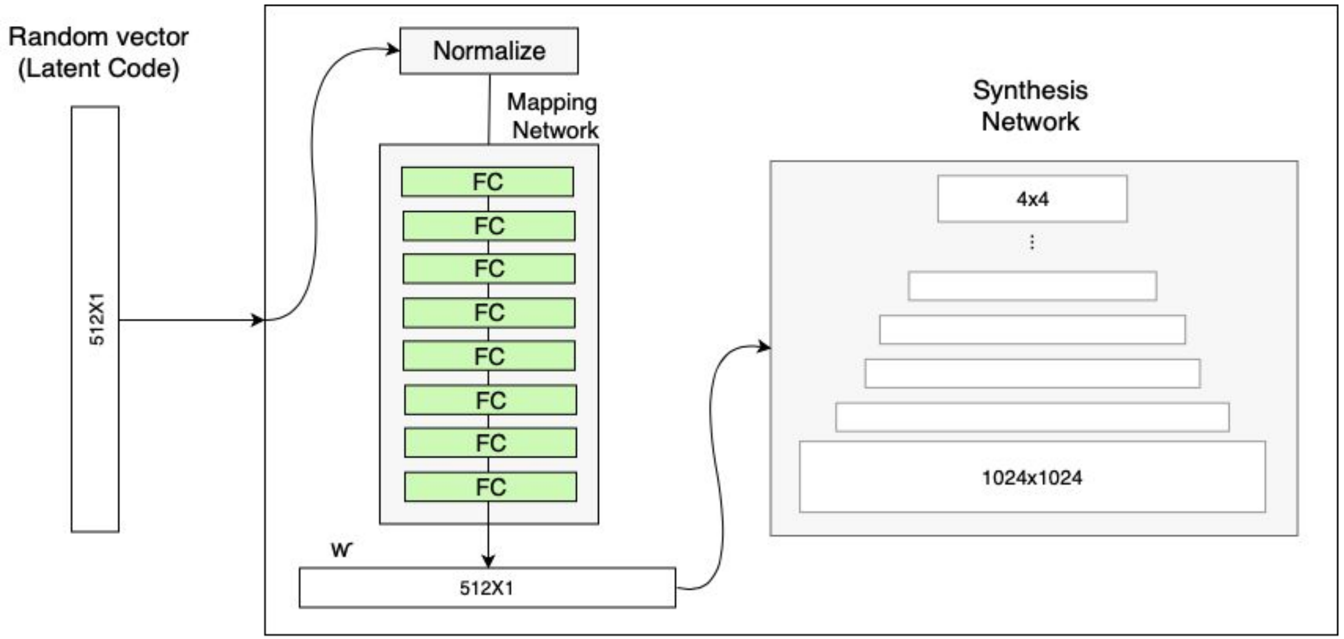


StyleGAN

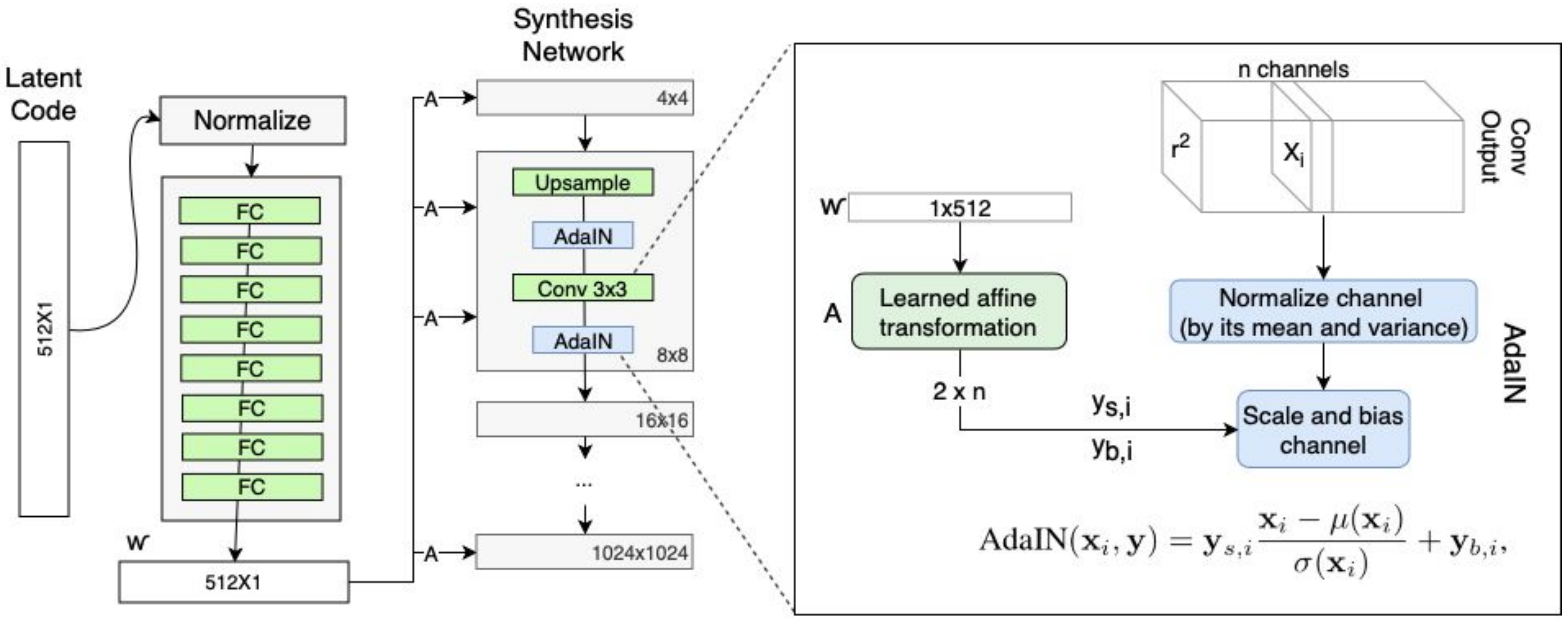
- PROGAN 2.0
- Further increase control in generating images
- Make the Features Disentangled as possible
- modifying the input of each level separately, it controls the visual features that are expressed in that level, from coarse features (pose, face shape) to fine details (hair color), without affecting other levels.

https://thispersondoesnotexist.com/

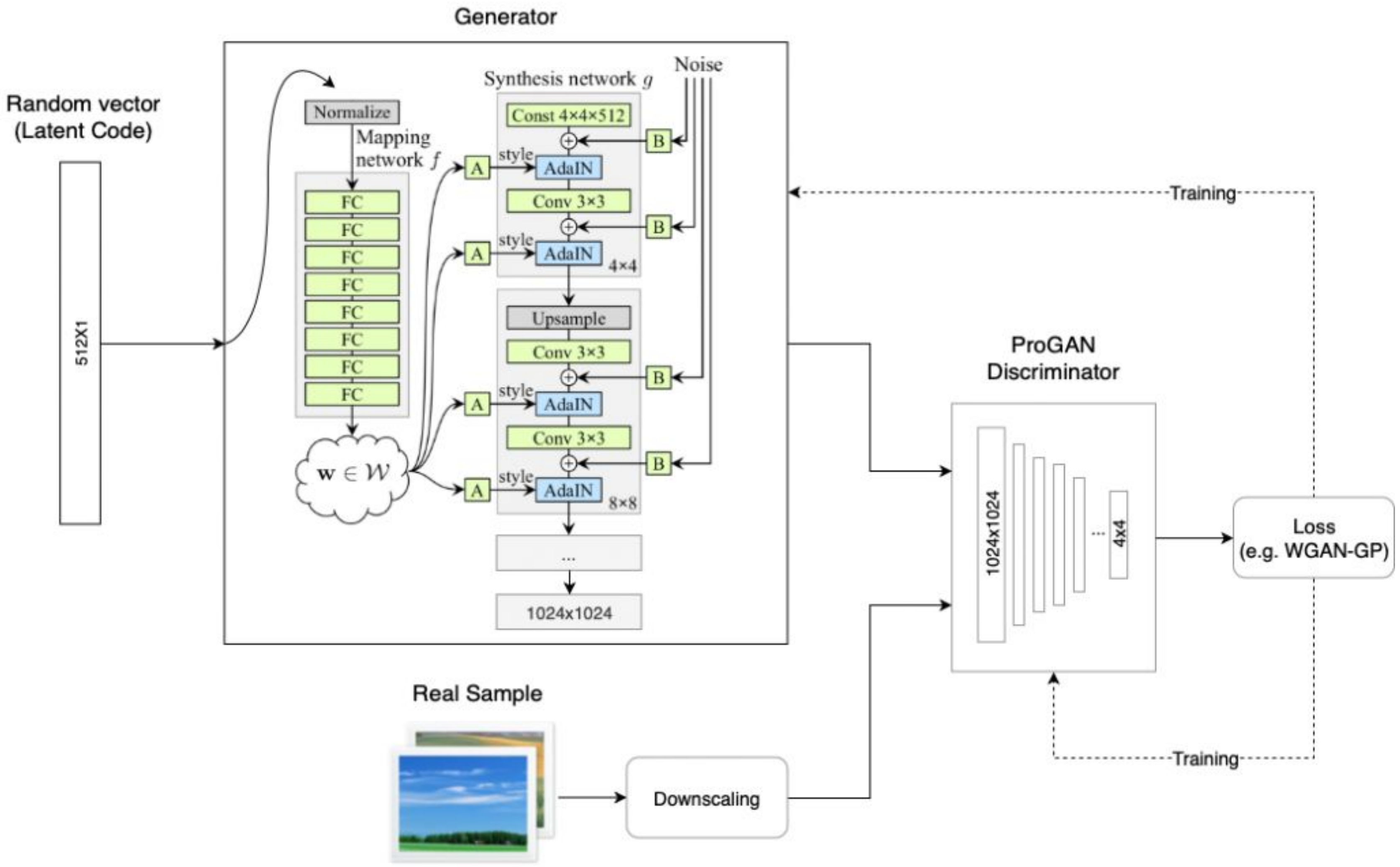
Mapping Network



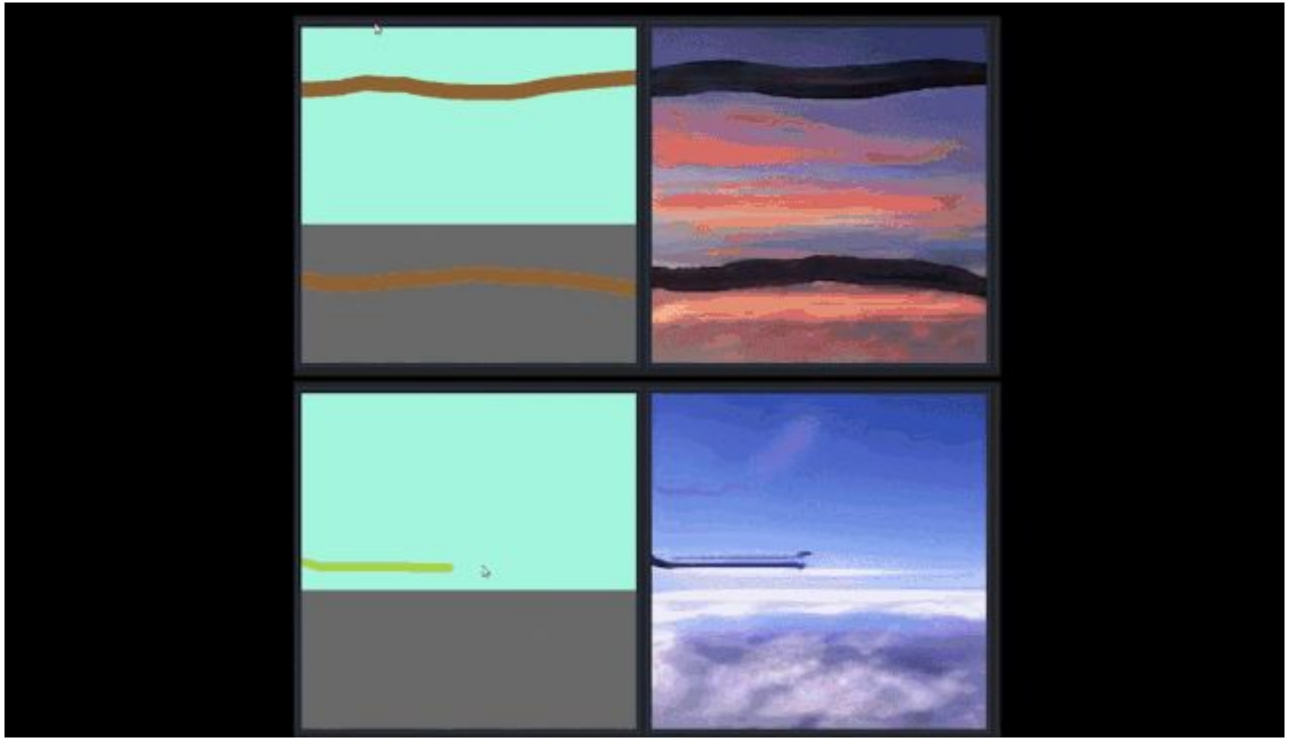
Adaptive Instance Normalization



$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i}\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$


Generator
Random vector (Latent Code)
512X1
Normalize
Mapping network f
FC
FC
FC
FC
FC
FC
FC
FC
$\mathbf{w} \in \mathcal{W}$
Synthesis network g
Noise
Const 4×4×512
style
AdaIN
Conv 3×3
4×4
Upsample
8×8
A
B
...
1024x1024
Training
ProGAN Discriminator
1024x1024
4x4
Loss (e.g. WGAN-GP)
Real Sample
Downscaling
Training

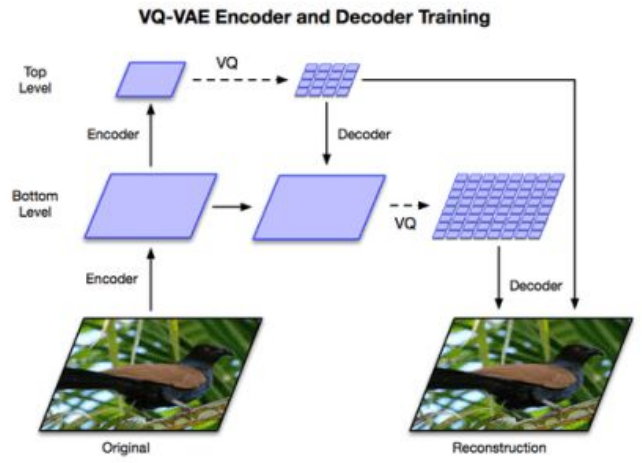
GauGAN



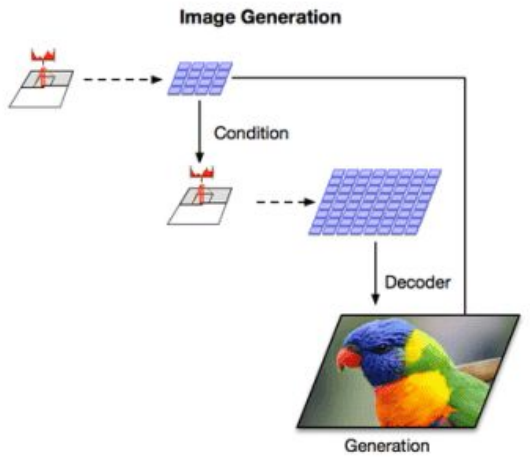
Going beyond GAN?

- Models are getting too crazy at this point
- Can we make models more efficient with new discoveries?

VQ-VAE-2

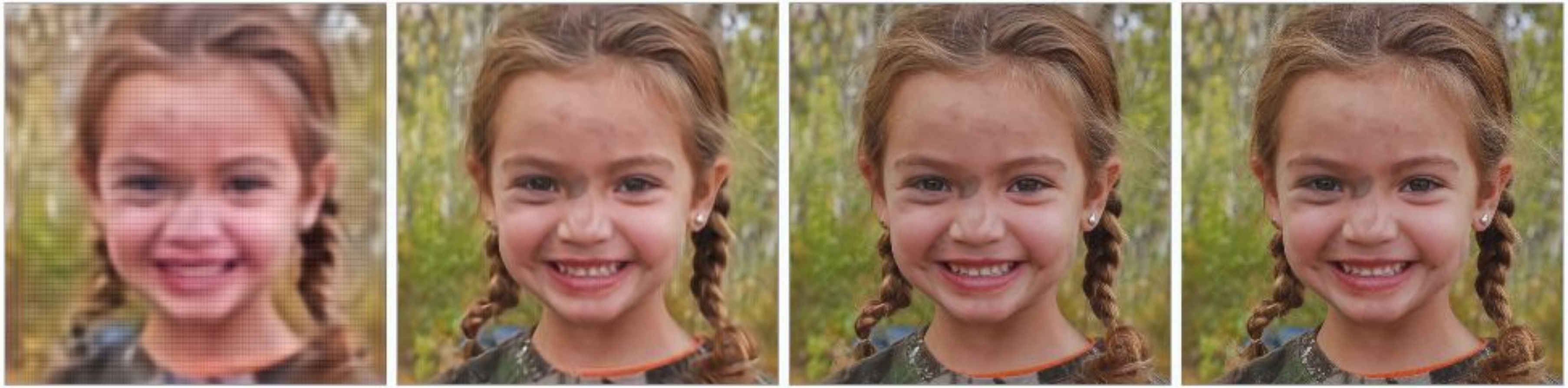


(a) Overview of the architecture of our hierarchical VQ-VAE. The encoders and decoders consist of deep neural networks. The input to the model is a 256×256 image that is compressed to quantized latent maps of size 64×64 and 32×32 for the *bottom* and *top* levels, respectively. The decoder reconstructs the image from the two latent maps.



(b) Multi-stage image generation. The top-level PixelCNN prior is conditioned on the class label, the bottom level PixelCNN is conditioned on the class label as well as the first level code. Thanks to the feed-forward decoder, the mapping between latents to pixels is fast. (The example image with a parrot is generated with this model).

Hierarchical Latent Values


h_{top}
$h_{\text{top}}, h_{\text{middle}}$
$h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$
Original

VQ-VAE-2 Samples

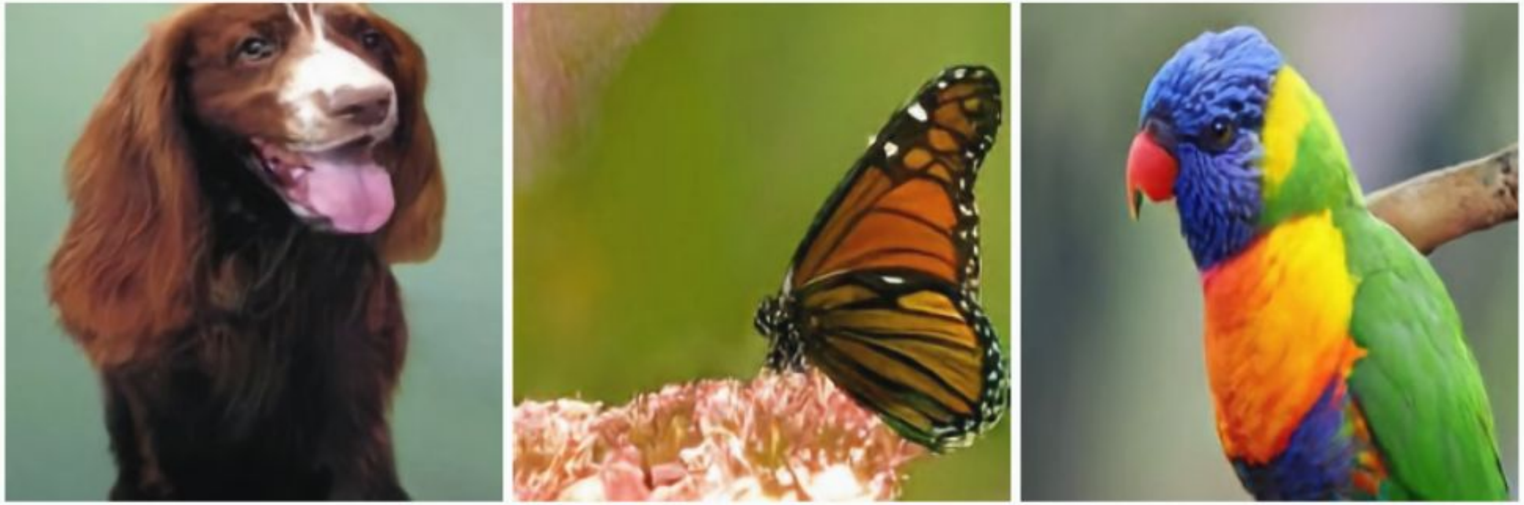
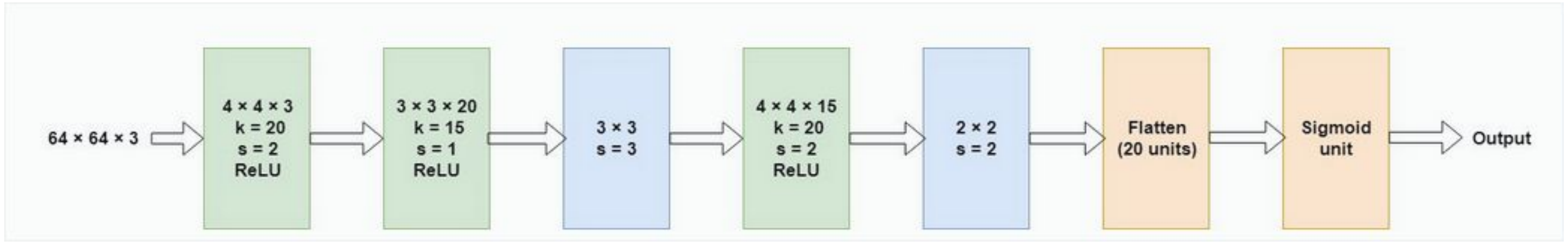


Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

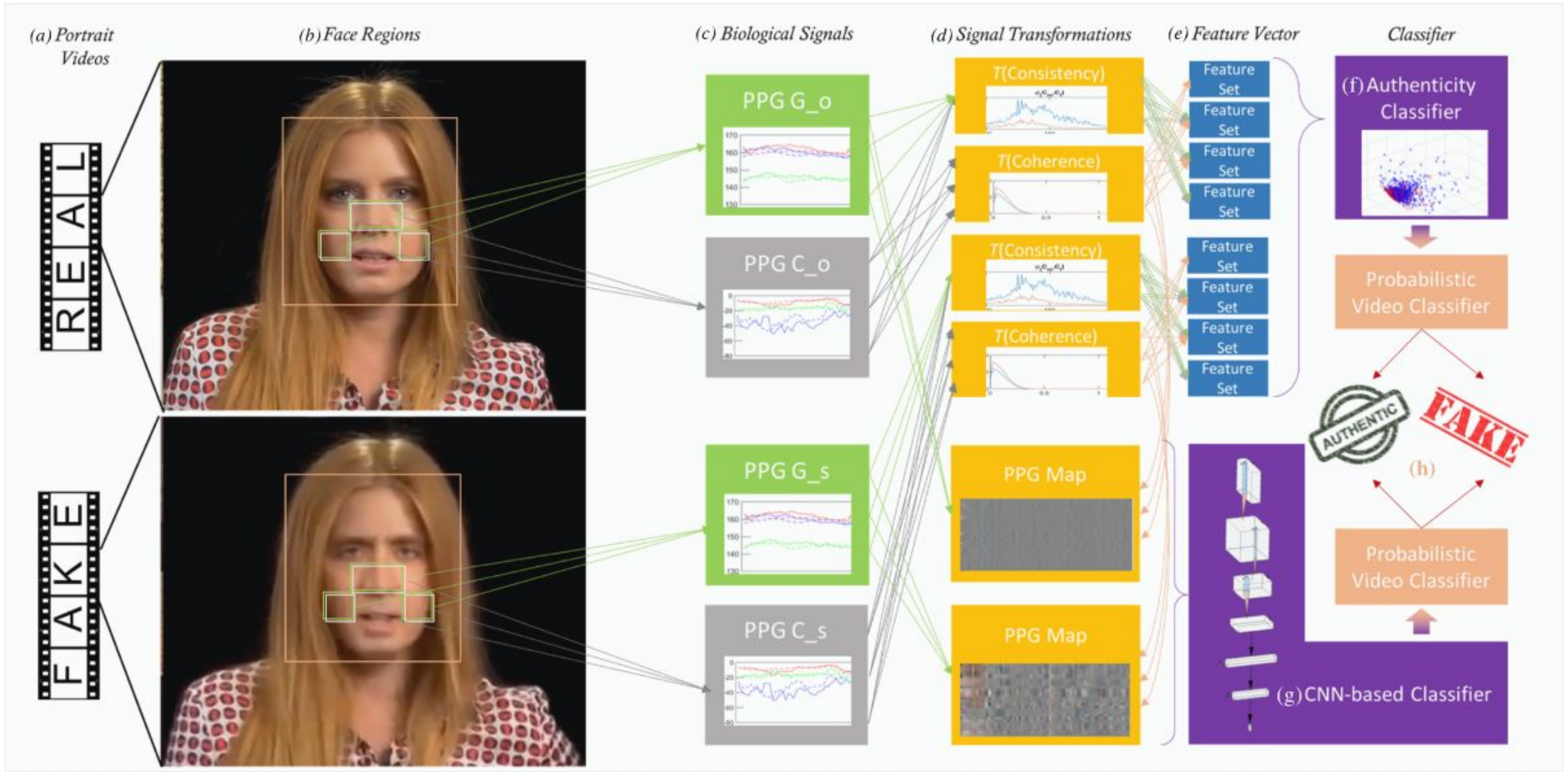
Detecting Fakes

Deep Image Forgery Detection

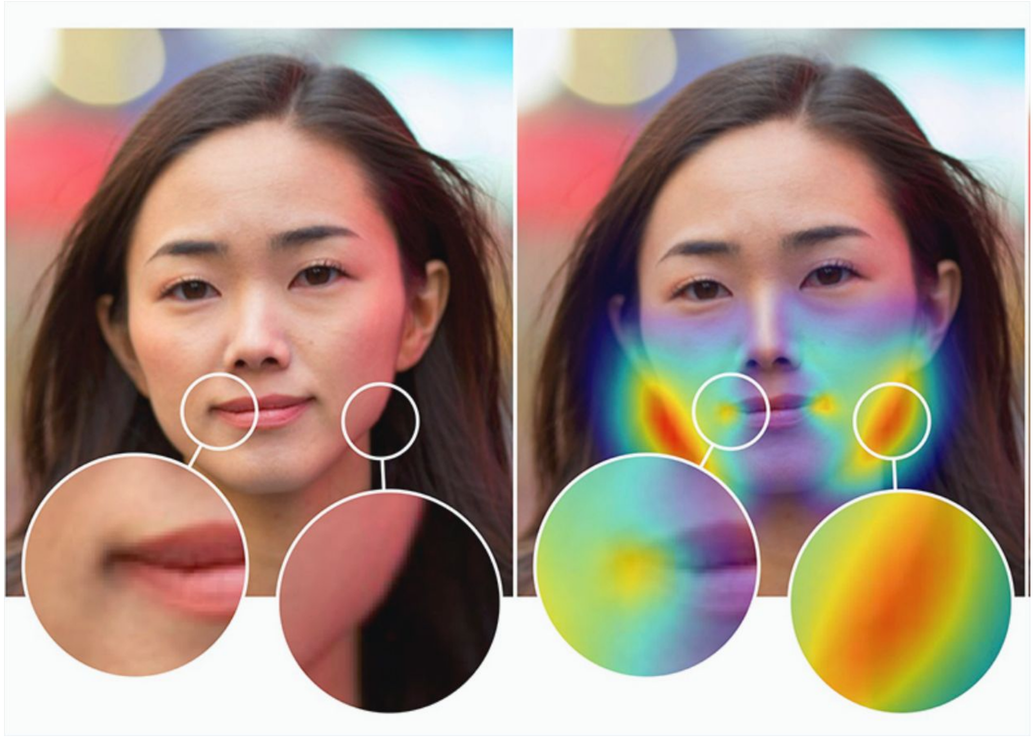
- Build a CNN Image Classifier (Fake/Not Fake)
- or/and create a mask to mark fake region.



Fakecatcher



Adobe Photoshop Detector



https://thisxdoesnotexist.com/

References

- Tutorial on Variational Autoencoders -https://arxiv.org/abs/1606.05908
- DCGAN - https://arxiv.org/abs/1511.06434
- PixelRNN - https://arxiv.org/abs/1601.06759
- Generative Adversarial Networks - https://arxiv.org/abs/1406.2661
- PROGAN - https://arxiv.org/abs/1710.10196
- CycleGAN - https://arxiv.org/abs/1703.10593
- BigGAN - https://arxiv.org/abs/1809.11096
- StyleGAN - https://arxiv.org/abs/1812.04948
- GauGAN - https://nvlabs.github.io/SPADE/
- VQ-VAE-2 - https://arxiv.org/abs/1906.00446
- Fakecatcher - https://arxiv.org/abs/1901.02212