Deep Generative Modeling

Saqib Shamsi

Unsupervised Learning

- No labels are provided during training
- General objective: inferring a function to describe hidden structure from unlabeled data
  - Clustering (discrete labels)
  - Representation/feature learning (continuous vectors)
  - Dimensionality reduction (lower-dimensional representation)
  - Density estimation (continuous probability)

Unconditional Data Generation

- **Problem:** Given some high dimensional data, generate new samples from the same distribution.

Training data ~ $p_{data}(x)$

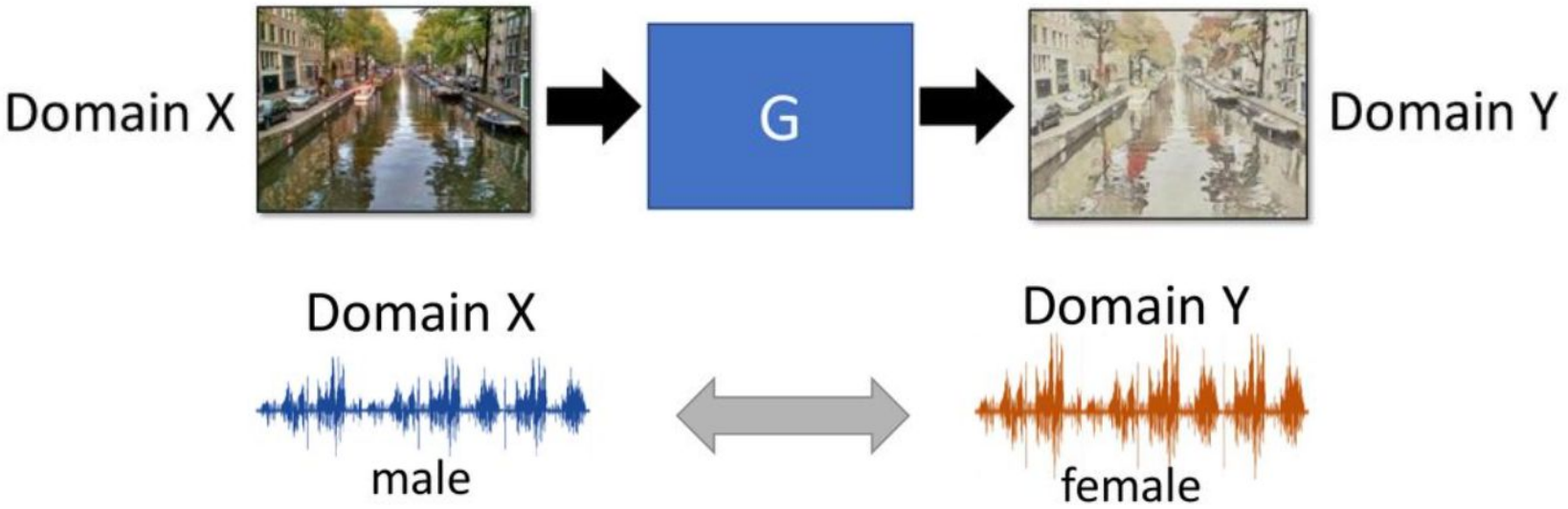
Generated samples ~ $p_{model}(x)$

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

Source

Conditional Data Generation

- **Problem:** Given some high dimensional data, generate new data subject to some constraint.



Source

Unconditional Image Generation

What About This Solution?

```
import glob

def augment(image):
    """
    Applies random rotations, brightness, contrast,
    flips, shearing, colour-shift and noise
    """
    # Code for augmenting `image`
    return augmented_image


def generate(given_data_path):
    images = glob.glob(given_data_path + '/*.jpg')
    for image in images:
        yield augment(image)
```

What About This One?

```
def generate_all(num_pixels, intensity_levels):
    """
    Generates all possible images with the number
    of pixels equal to `num_pixels` and of given
    `intensity_levels`
    """
    # Code to generate all possible images
    return all_images


def main():
    DESIRED_SIZE = 100 * 100 * 3
    INTENSITIES = 256
    all_images = generate_all(DESIRED_SIZE, INTENSITIES)
    # Pick whatever images you want from `all_images`
```

What About This One?

```
def generate_all(num_pixels, intensity_levels):
    """
    Generates all possible images with the number
    of pixels equal to `num_pixels` and of given
    `intensity_levels`
    """
    # Code to generate all possible images
    return all_images


def main():
    DESIRED_SIZE = 100 * 100 * 3
    INTENSITIES = 256
    all_images = generate_all(DESIRED_SIZE, INTENSITIES)
    # Pick whatever images you want from `all_images`
```

1.58×10^{72245}

What About This One?

```
def generate_all(num_pixels, intensity_levels):
    """
    Generates all possible images with the number
    of pixels equal to `num_pixels` and of given
    `intensity_levels`
    """
    # Code to generate all possible images
    return all_images


def main():
    DESIRED_SIZE = 100 * 100 * 3
    INTENSITIES = 256
    all_images = generate_all(DESIRED_SIZE, INTENSITIES)
    # Pick whatever images you want from `all_images`
```

Estimated #protons + #electrons in the observable universe: 10^{80}

$$1.58 \times 10^{72247}$$

What About This One?

```
def generate_all(num_pixels, intensity_levels):
    """
    Generates all possible images with the number
    of pixels equal to `num_pixels` and of given
    `intensity_levels`
    """
    # Code to generate all possible images
    return all_images
```

Moreover,
Undesired Images

```
def main():
    DESIRED_SIZE = 100 * 100 * 3
    INTENSITIES = 256
    all_images = generate_all(DESIRED_SIZE, INTENSITIES)
    # Pick whatever images you want from `all_images`
```

What We Want

- Meaningfully different samples. Not the same data in augmented form
  - Samples that have the same underlying pattern as the training data, but should be different from the training data
  - Eg, digits in a different handwriting, cats in new poses, paintings with different scenery and brushstrokes
- The generation process must be tractable

Density Estimation

- **Goal:** Estimate the probability density function for given data.

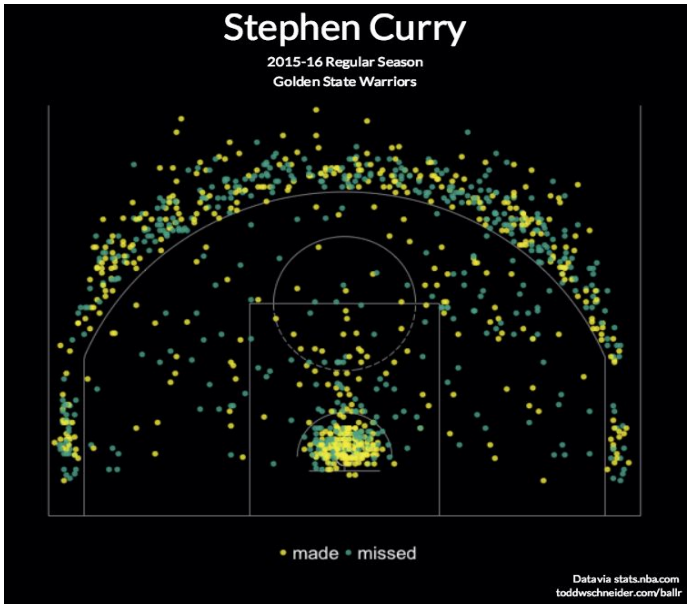


Image Source

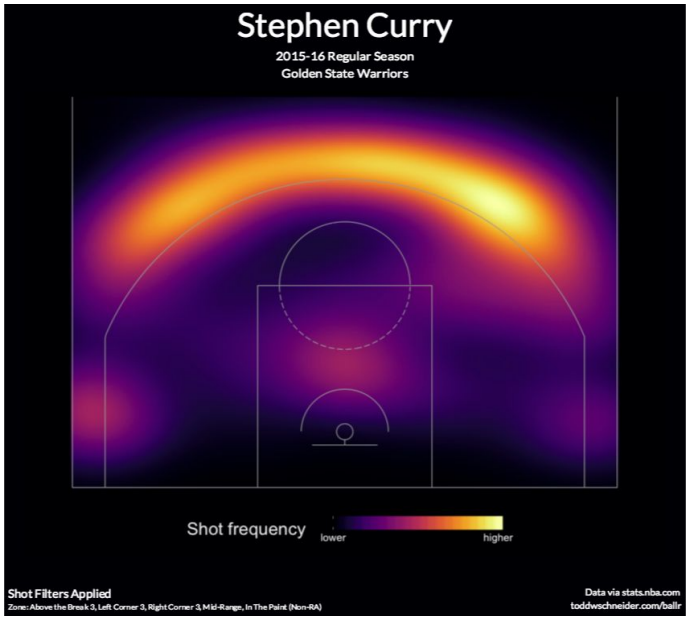
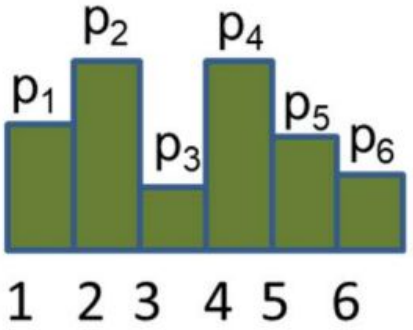


Image Source

Generative Model

- Simple Generative Models:
  - **Multinomial PMF:** For discrete data. Can be modeled as a probability table.
    - Probability of observing a data point can be obtained directly from the table
  - **Gaussian PDF:** For continuous data. Distribution is parameterized by mean μ and standard deviation σ
    - Probability of observing a data point x can be obtained by

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



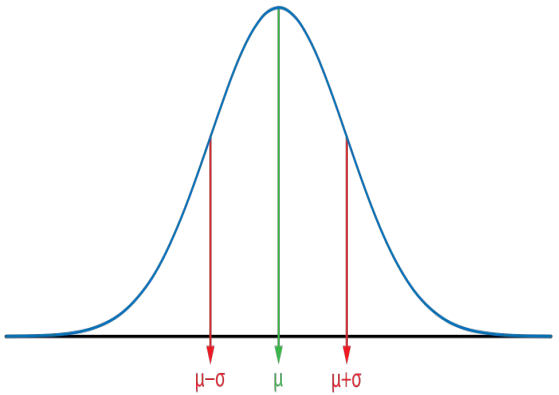


Image Source

Source

Sampling From a Distribution

- Since our goal was to generate new data, we have to sample from the density function.
- Sampling from a multinomial PMF:
  - From the model probabilities p_1, ..., p_k, compute the cumulative distribution

$$Fi = p_1 + \cdots + p_i \quad \text{for all } i \in \{1, ..., k\}$$

  - Draw a uniform random number u ~ [0, 1]
  - Return the smallest i such that $u \leq F_i$
- We have algorithms to sample from a normal distribution given μ and σ
- We have algorithms to sample from uniform and other simple distributions

Source

Generative Models for ‘Simple’ Data

- We are given some data $X = \{x_i\}_{i=1}^{N}$
- We choose a model P(x;w) (that we can sample from later) to model the distribution of X
- We estimate `w` using some method such that P(x;w) best fits the observations X, hoping that it will account for data not in X
- Finally, we sample from P to generate new data such that the new points generated follow the same pattern as X

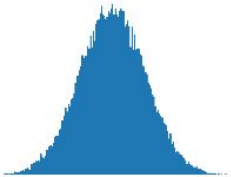
What About High Dimensional Data?

- We are interested in generating high dimensional data like images, audio etc
- We can sample from simple distributions but they do not capture the distribution of high dimensional data well
- Neural networks have the capacity to model the distribution of high dimensional data, but how do we sample from them?
- Can we combine the two above?

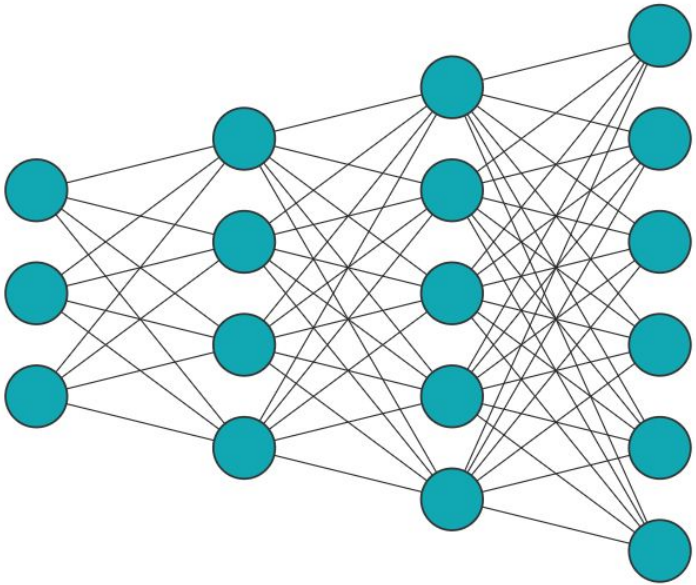
Mapping Distributions

- We create a function `f` such that it maps a simple distribution Z={z} to a high dimensional distribution X={x}. f: z -> x
- `f` would be a neural network which has the capacity to model the distribution of `X`
- When we want to sample new data, we sample z from Z, provide it as input to f, which gives a data point x, hoping the the generated x is different from the ones in X

Putting it all Together



z~p(z)



f(z; w)

x~p(x)

Learning Network Parameters

- **Frameworks to learn the parameters of the generative network `f`:**
  - Generative Adversarial Networks (GAN)
  - Variational Autoencoders (VAE)

Learning Network Parameters

Generative Adversarial Networks

- Goodfellow et al, 2014
- Goal is to model the data distribution
- Trained using a pair of networks playing a minmax game
- They are termed as “adversaries” because they have conflicting loss functions
- New data are sampled by “seeding” the trained generator

GAN Architecture

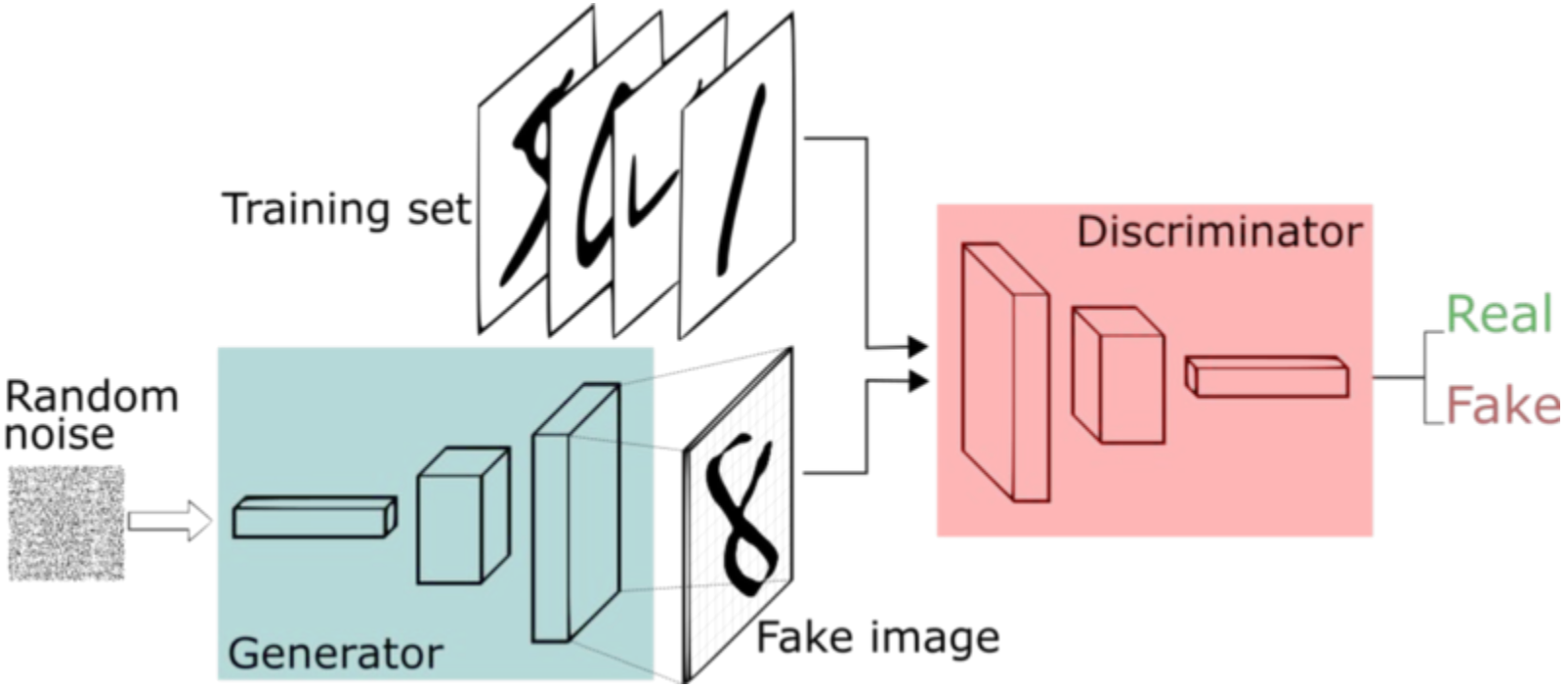


Image Source

Vanilla GAN Loss

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- Minmax game being played between a generator (G) and a discriminator (D)
- D tries to maximize the log-likelihood for the binary classification problem of real (1) vs fake (0)
- G tries to minimize the log-probability of its samples being classified as fake (0) by D (“fooling” the discriminator)

Goodfellow, Ian J. et al. Generative Adversarial Nets. *NIPS* (2014).

Vanilla GAN Training

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

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

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Goodfellow, Ian J. et al. Generative Adversarial Nets. *NIPS* (2014).

Problems with Training Vanilla GANs

- Non-convergence
- A balancing act
  - If the discriminator gets too strong too fast, generator does not learn
  - If the discriminator is weak generator does not receive reliable feedback
- Too sensitive to the choice of hyperparameters. Require “hacks” to behave well
- Mode Collapse: generated samples lack diversity

Non-Convergence

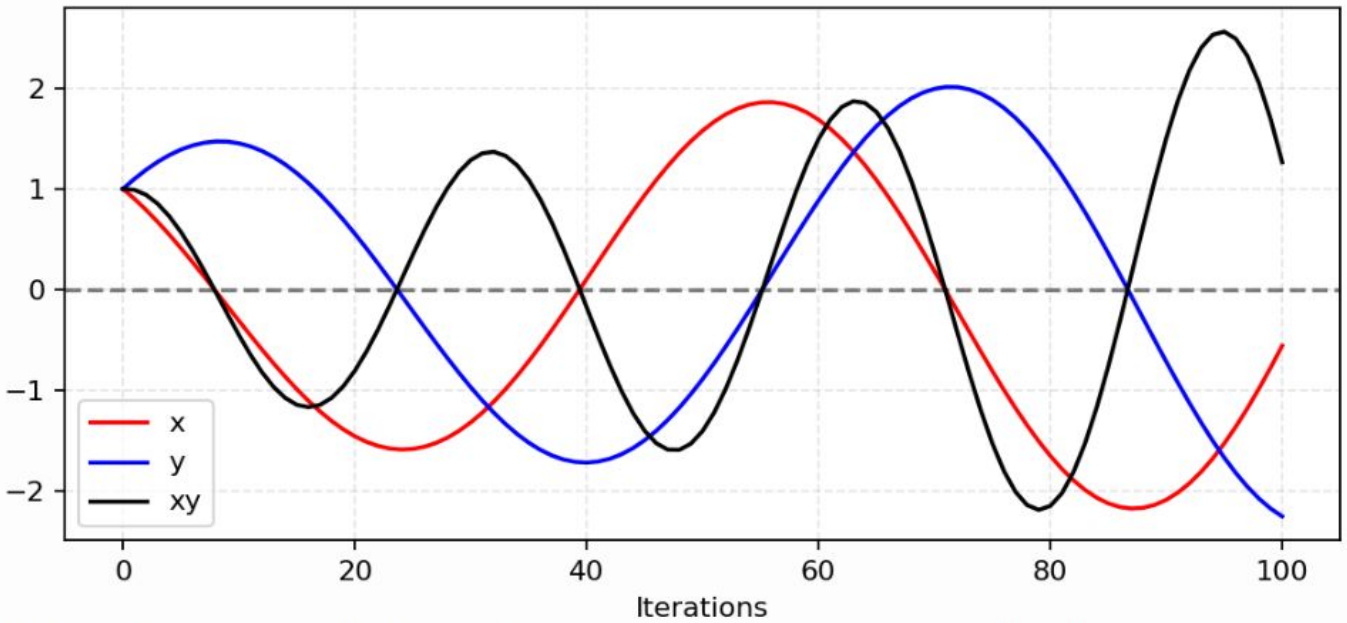
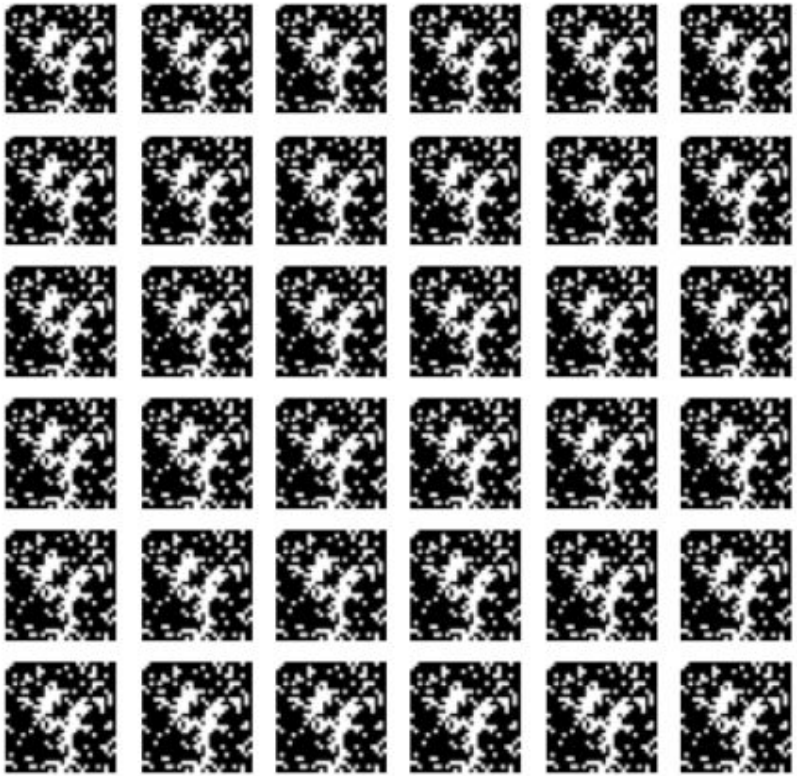


Fig. 3. A simulation of our example for updating x to minimize xy and updating y to minimize $-xy$. The learning rate $\eta = 0.1$. With more iterations, the oscillation grows more and more unstable.

Weng, Lilian. From GAN to WGAN. *ArXiv* abs/1904.08994 (2019)

Mode Collapse



Source

Learning Network Parameters

Variational Autoencoders

- Kingma and Welling, 2013
- Goal is to compress dataset information to a latent space
- Autoencoder whose encodings distribution is regularised
- Trained using gradient based methods
- New data are sampled by “seeding” the latent space

Autoencoder

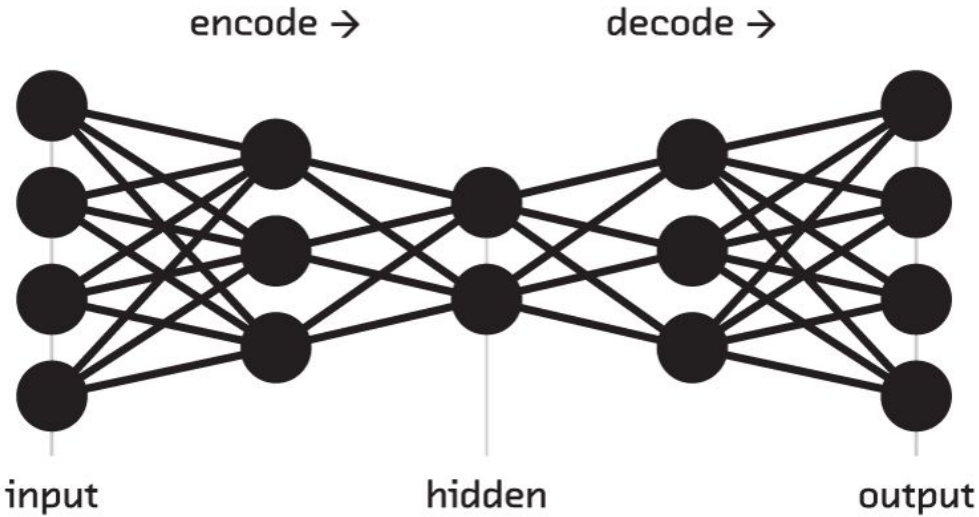
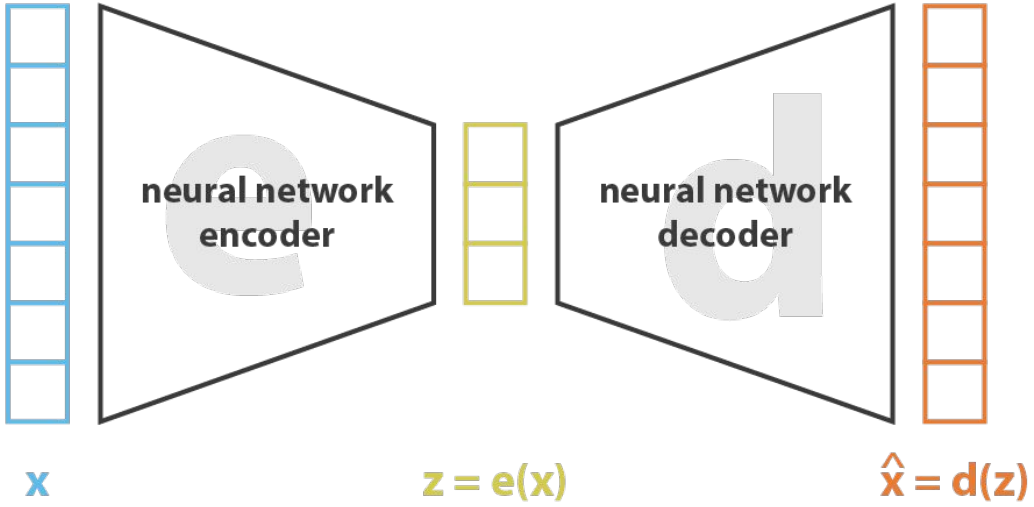


Image Source

Autoencoder Loss



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Image Source

Variational Autoencoder

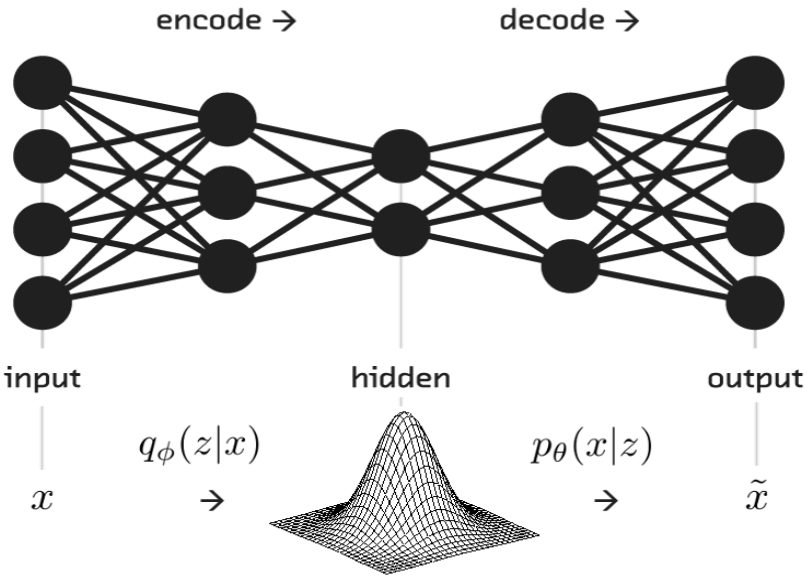
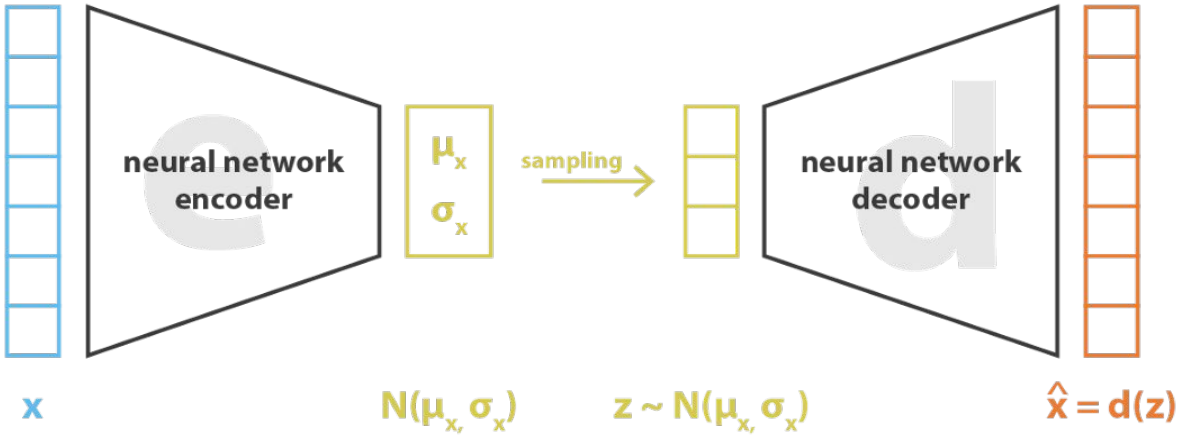


Image Source

Variational Autoencoder Loss



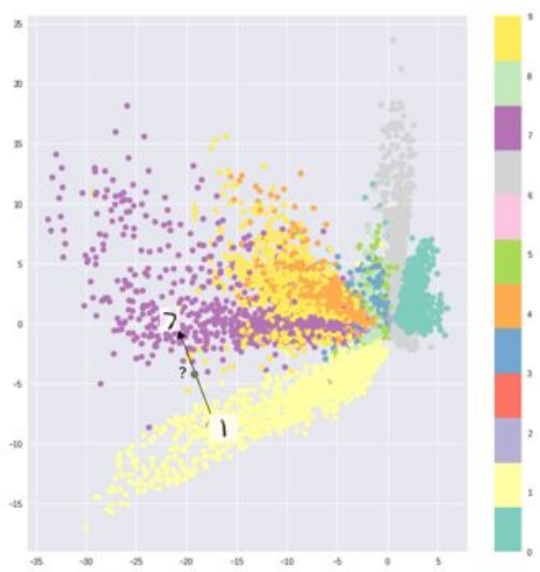
$$\text{loss} = ||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Image Source

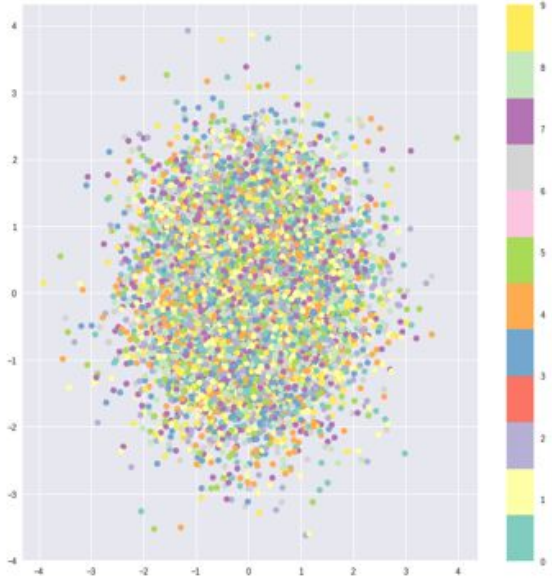
$$-\frac{1}{2}\sum(1 + log(\sigma_x^2) - \mu_x^2 - \sigma_x^2)$$

The Need to Regularize the Latent Space

Only reconstruction loss



Only KL divergence



Combination

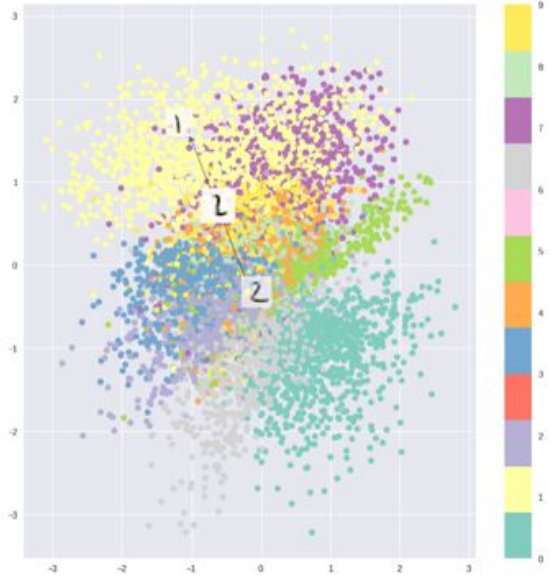


Image Source

Reparameterization Trick

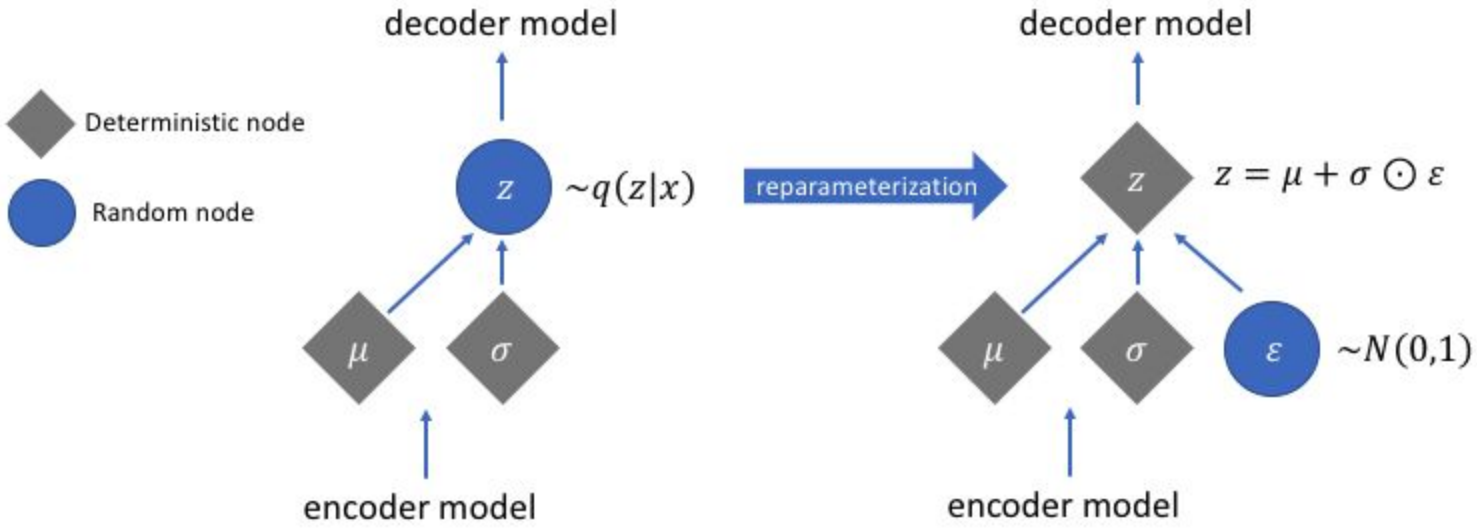
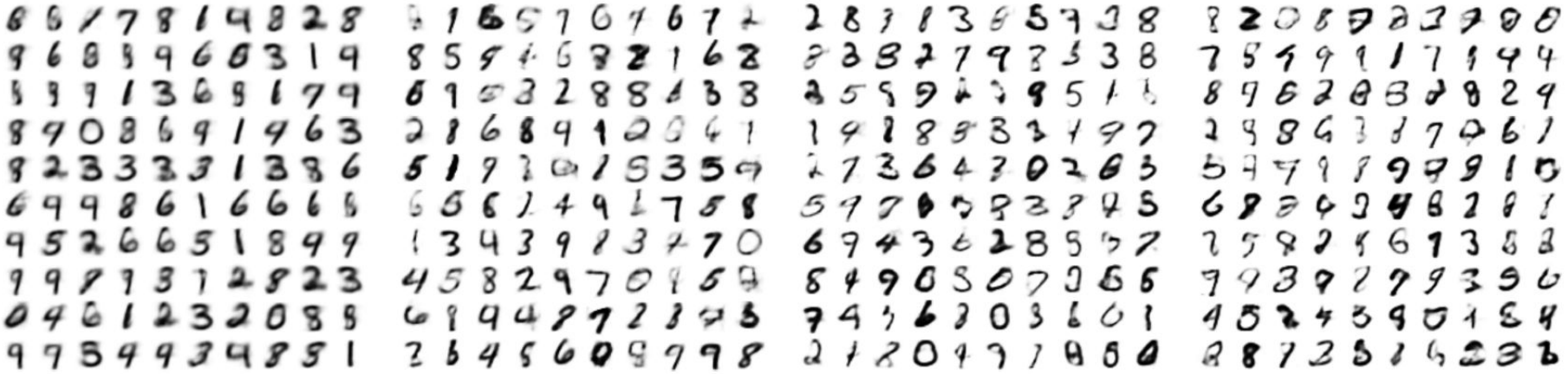


Image Source

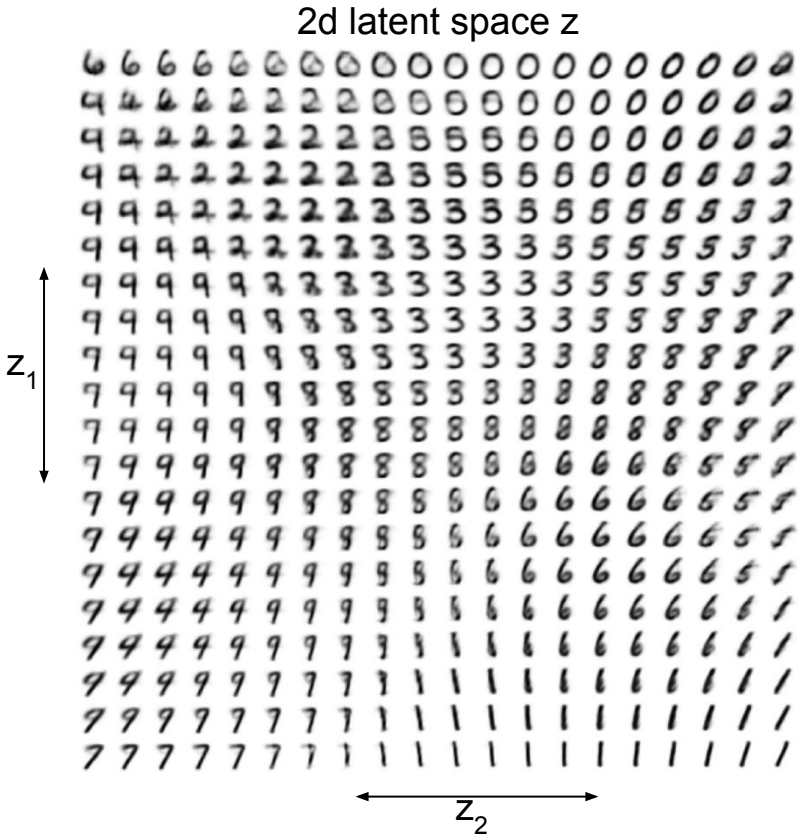
Data Generation



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Kingma, D.P., & Welling, M. Auto-Encoding Variational Bayes. ICLR (2014)

Data Generation



Kingma, D.P., & Welling, M. Auto-Encoding Variational Bayes. ICLR (2014)

Applications

Data Generation

GAN Progress on Face Generation

Link

Generated Face Images

Karras, Tero et al. Analyzing and Improving the Image Quality of StyleGAN. *ArXiv* abs/1912.04958 (2019)

Want More Samples?

Generated Car Images

Karras, Tero et al. Analyzing and Improving the Image Quality of StyleGAN. *ArXiv* abs/1912.04958 (2019)

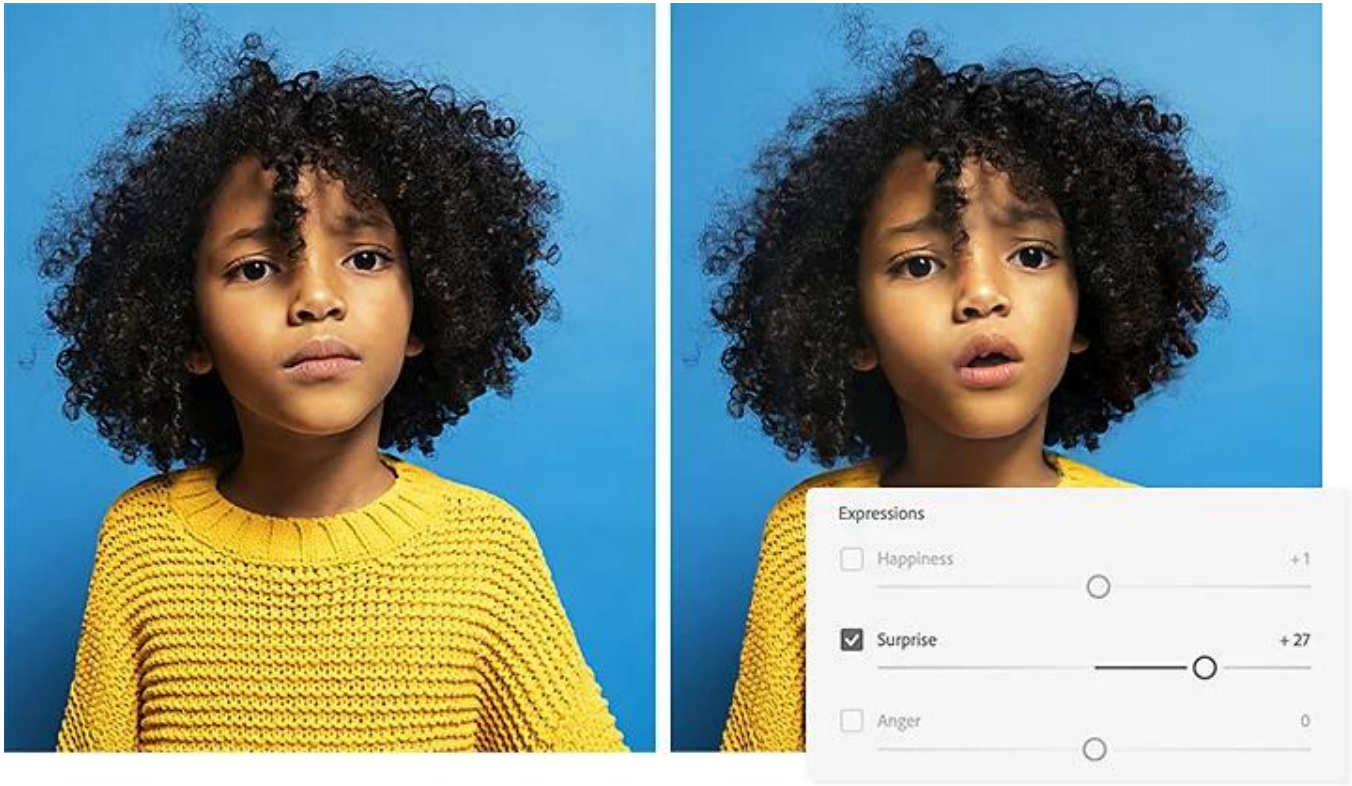
Generated Cat and Church Images

Karras, Tero et al. Analyzing and Improving the Image Quality of StyleGAN. *ArXiv* abs/1912.04958 (2019)

It's raining cats and cats

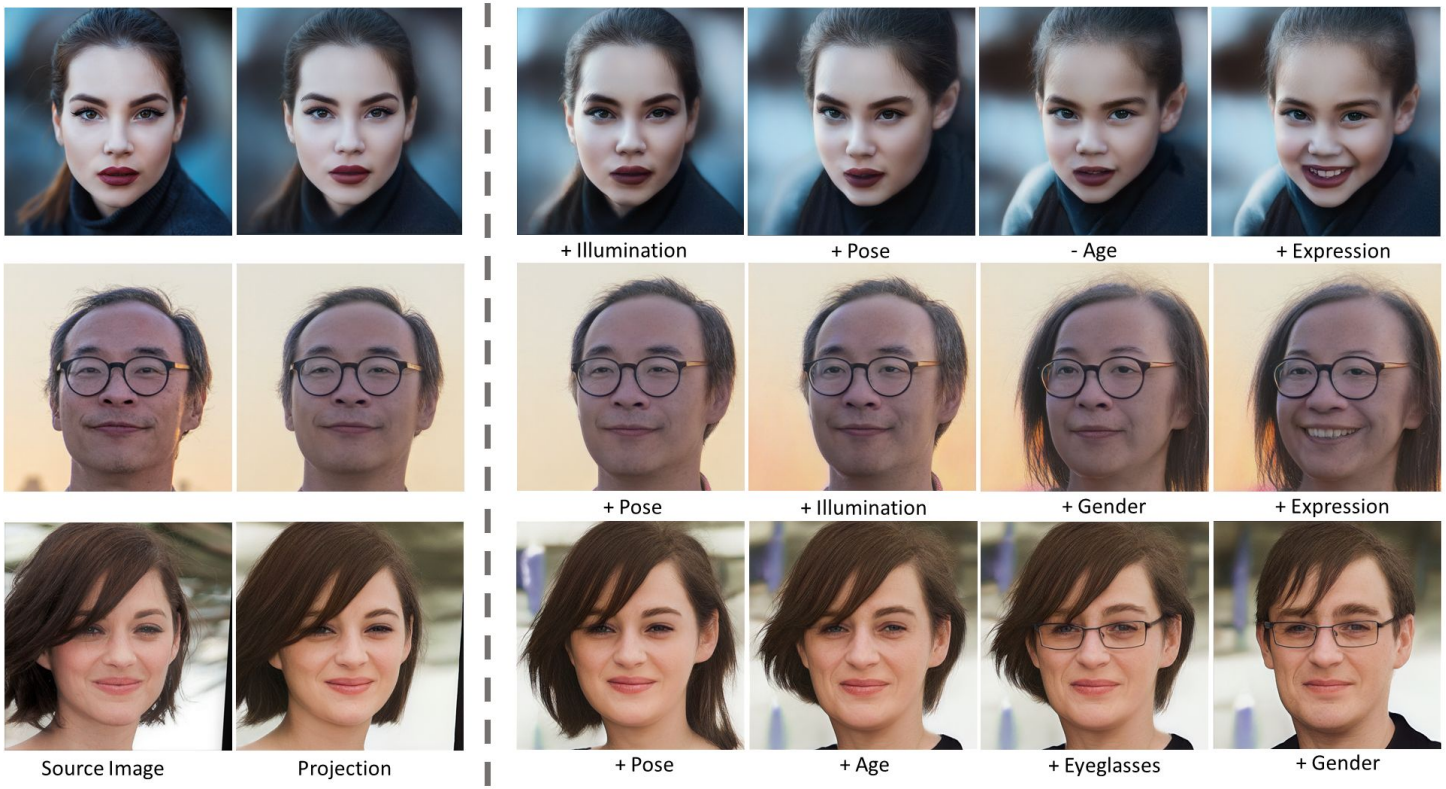
Image Editing

Image Editing



Adobe Neural Filters

Image Editing



StyleFlow

Image to Embedding

Algorithm 1: Latent Space Embedding for GANs

Input: An image $I \in \mathbb{R}^{n \times m \times 3}$ to embed; a pre-trained generator $G(\cdot)$.

Output: The embedded latent code w^* and the embedded image $G(w^*)$ optimzed via F'.

Initialize latent code $w^* = w$;

while *not converged* **do**

  $L \leftarrow L_{percept}(G(w^*), I) + \frac{\lambda}{N}\|G(w^*) - I\|_2^2$;

  $w^* \leftarrow w^* - \eta F'(\nabla_{w^*} L)$;

end

Abdal, R., Qin, Y., & Wonka, P. (2020). Image2StyleGAN++: How to Edit the Embedded Images? *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8293-8302.

Image to Embedding

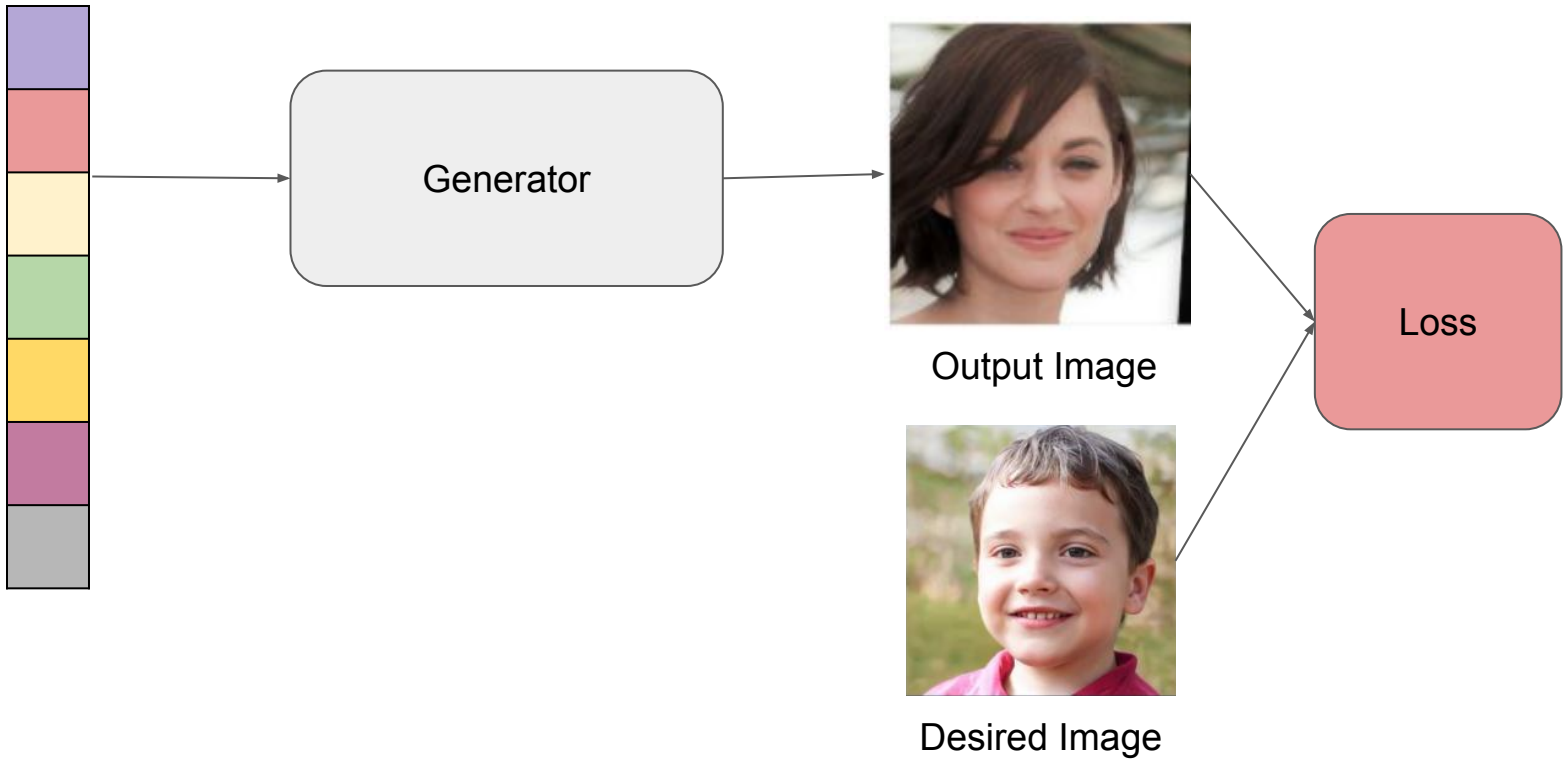
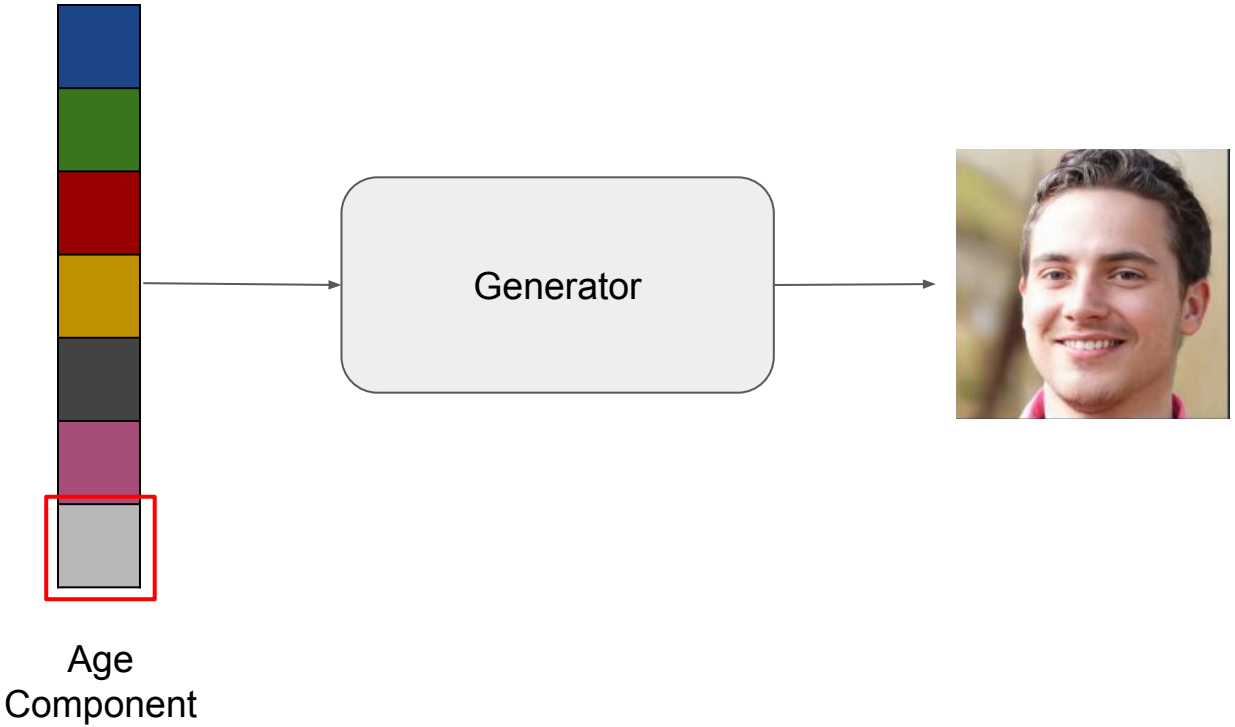


Image Transformation

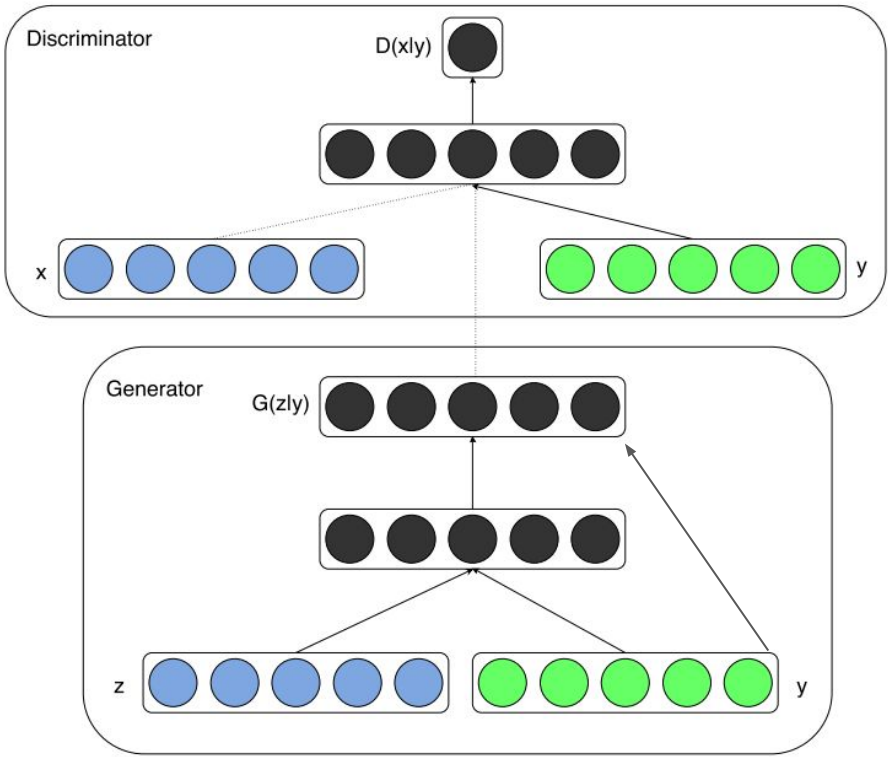


Conditional Image Generation

Conditional Generation

- In an unconditional model, there is no control over the characteristics of the data being generated
- A conditional model generates fake samples that satisfy some condition (a class label, base image, tags, text description, etc), rather than generating generic samples

Vanilla cGANs



Mirza, Mehdi and Simon Osindero. Conditional Generative Adversarial Nets. *ArXiv* abs/1411.1784 (2014)

BigGAN

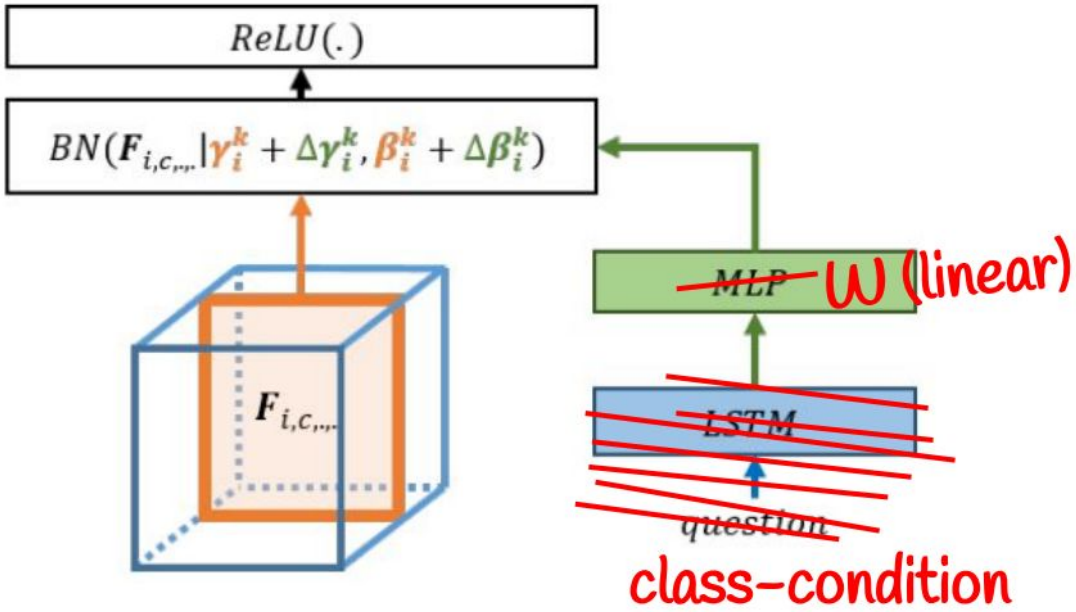


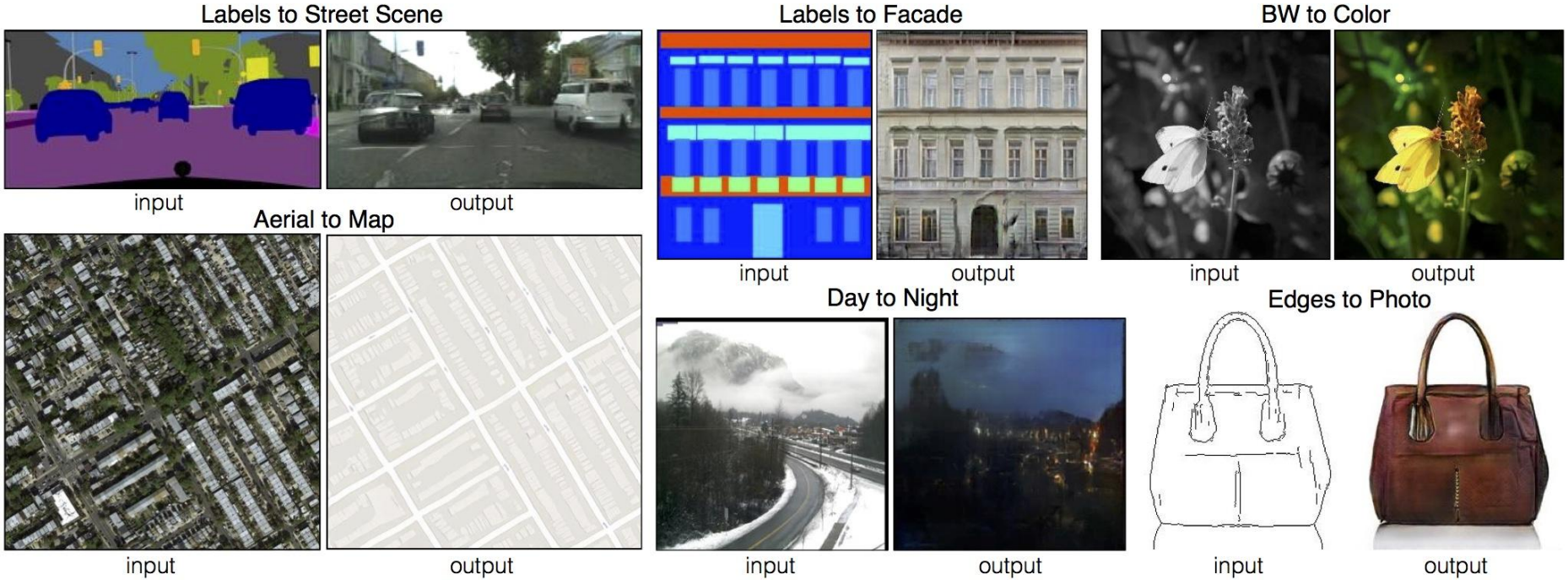
Image Source

- Among other things, use class-conditional batchnorm layers

BigGAN

Brock, Andrew et al. Large Scale GAN Training for High Fidelity Natural Image Synthesis. *ArXiv* abs/1809.11096 (2018)

Paired Image-to-Image Translation



Isola, P., Zhu, J., Zhou, T., & Efros, A.A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *CVPR (2017)*

Paired Image-to-Image Translation

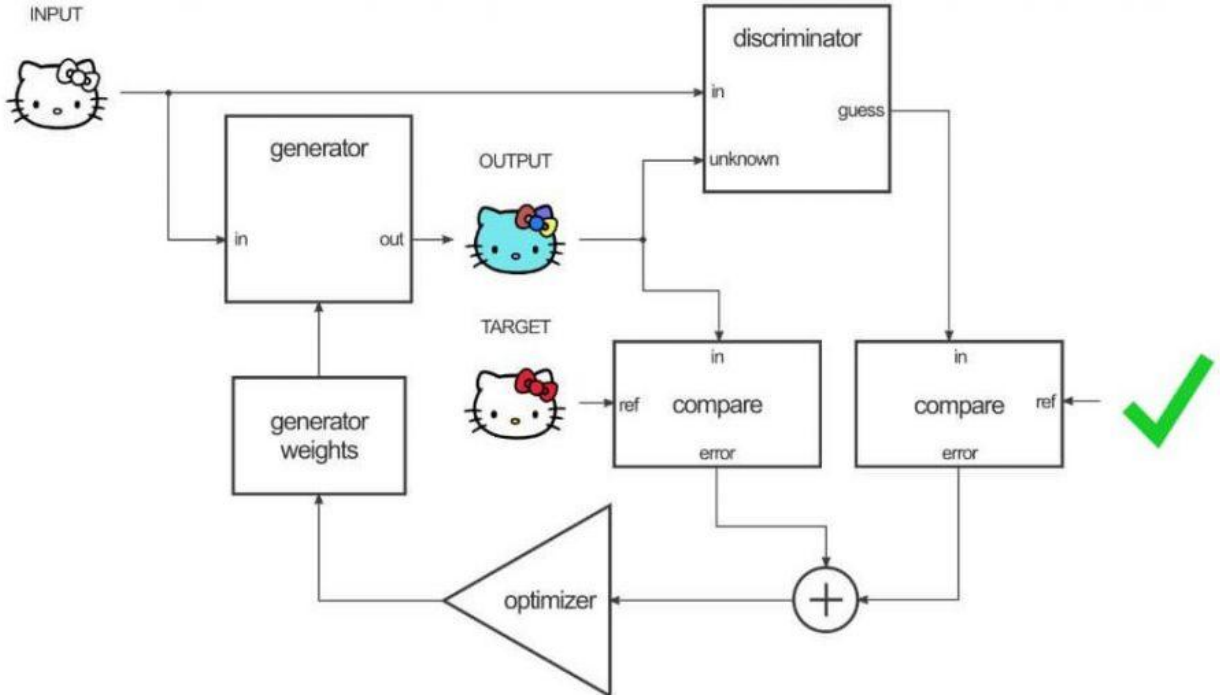


Image Source

Paired Image-to-Image Translation

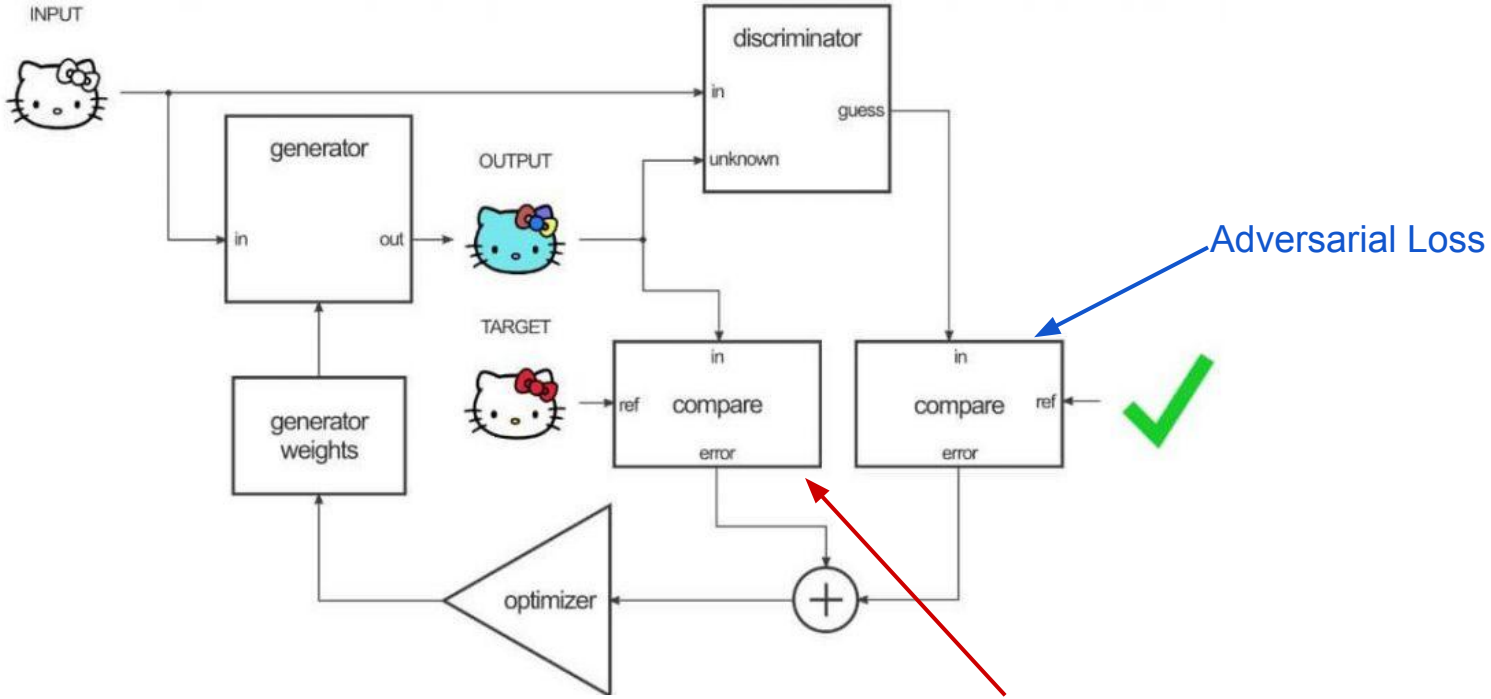
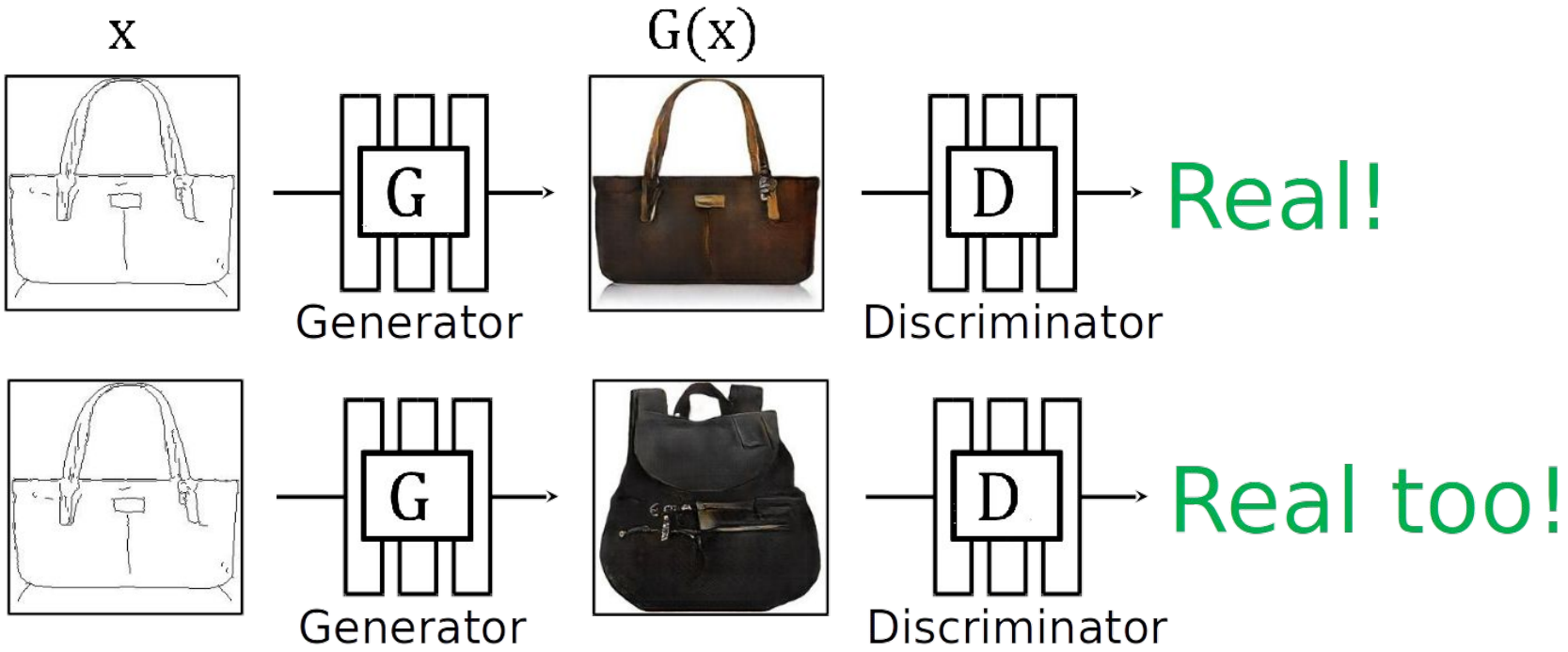


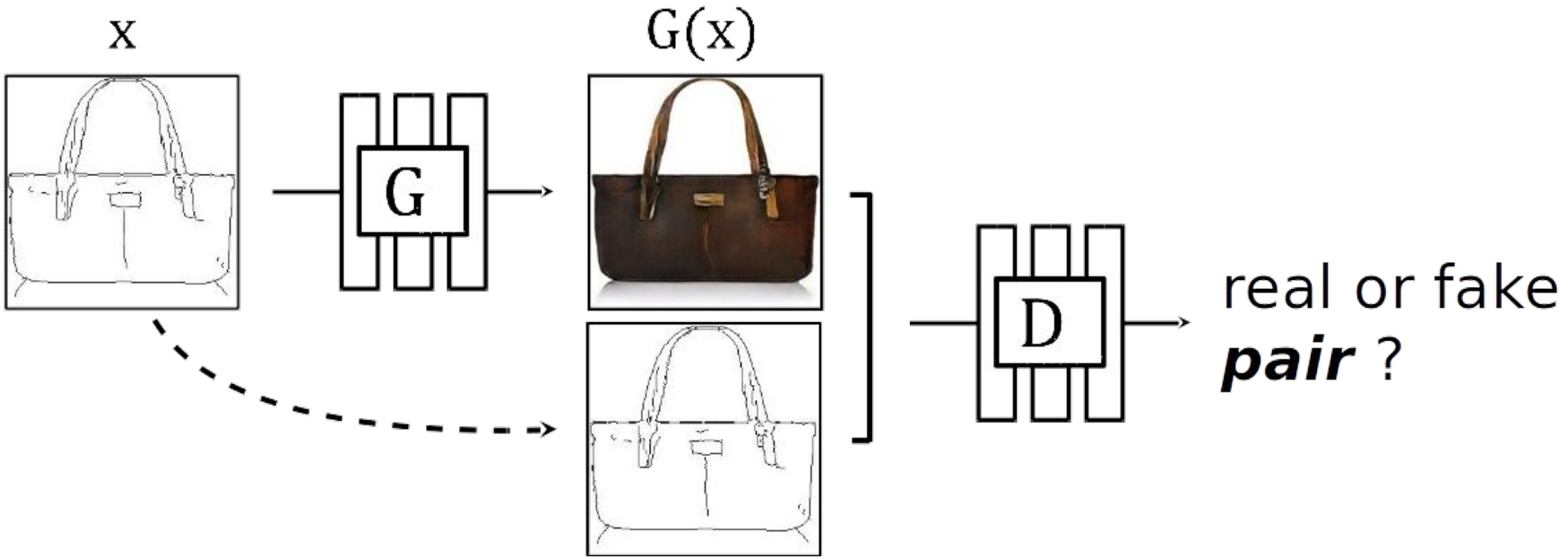
Image Source

Why cGAN?



Credits: Jun-Yan Zhu

Why cGAN?



Credits: Jun-Yan Zhu

Effect of Different Losses

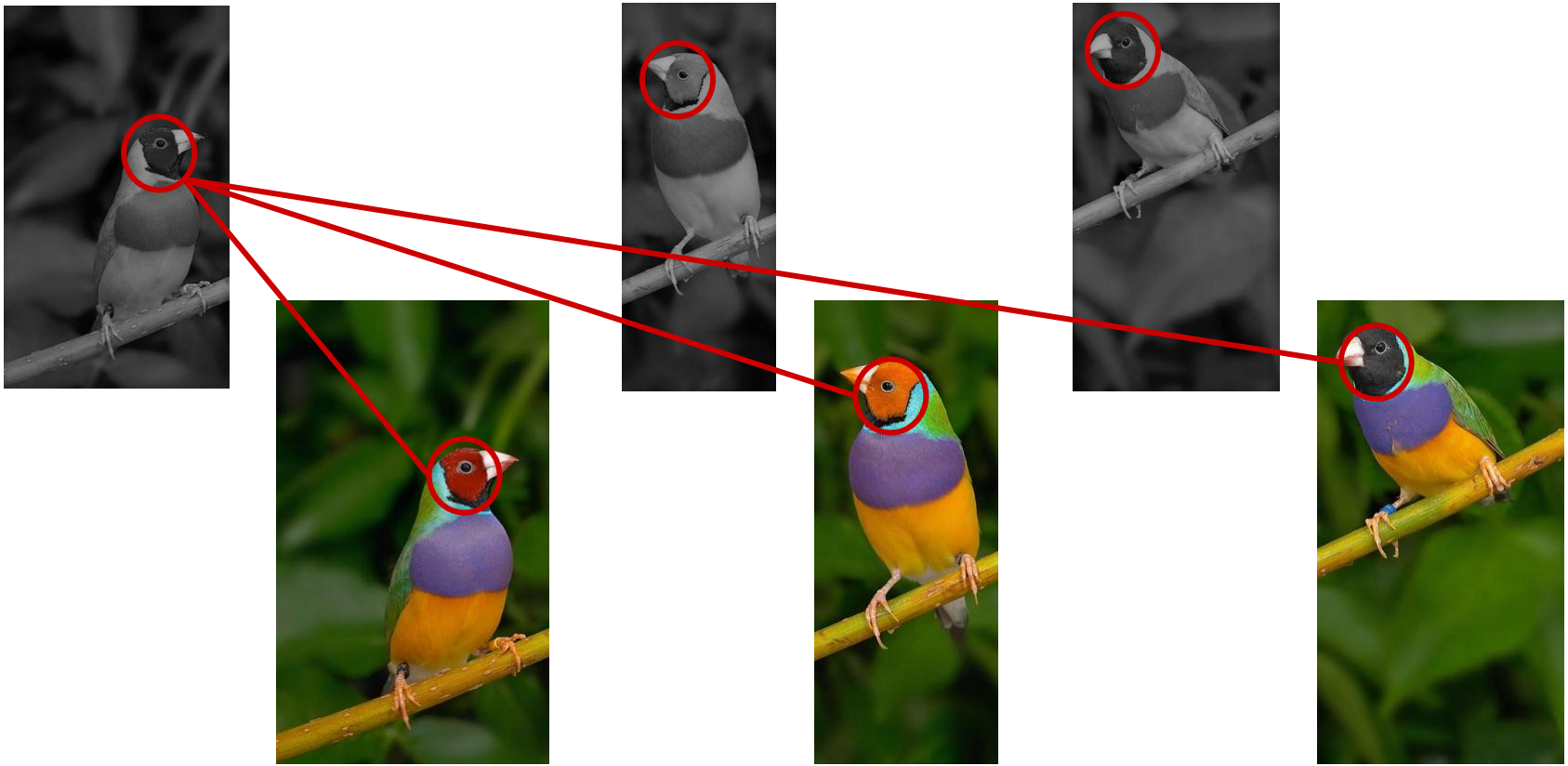
Isola, P., Zhu, J., Zhou, T., & Efros, A.A. (2016). Image-to-Image Translation with Conditional Adversarial Networks. *CVPR (2017)*

Notion of Adversarial Loss

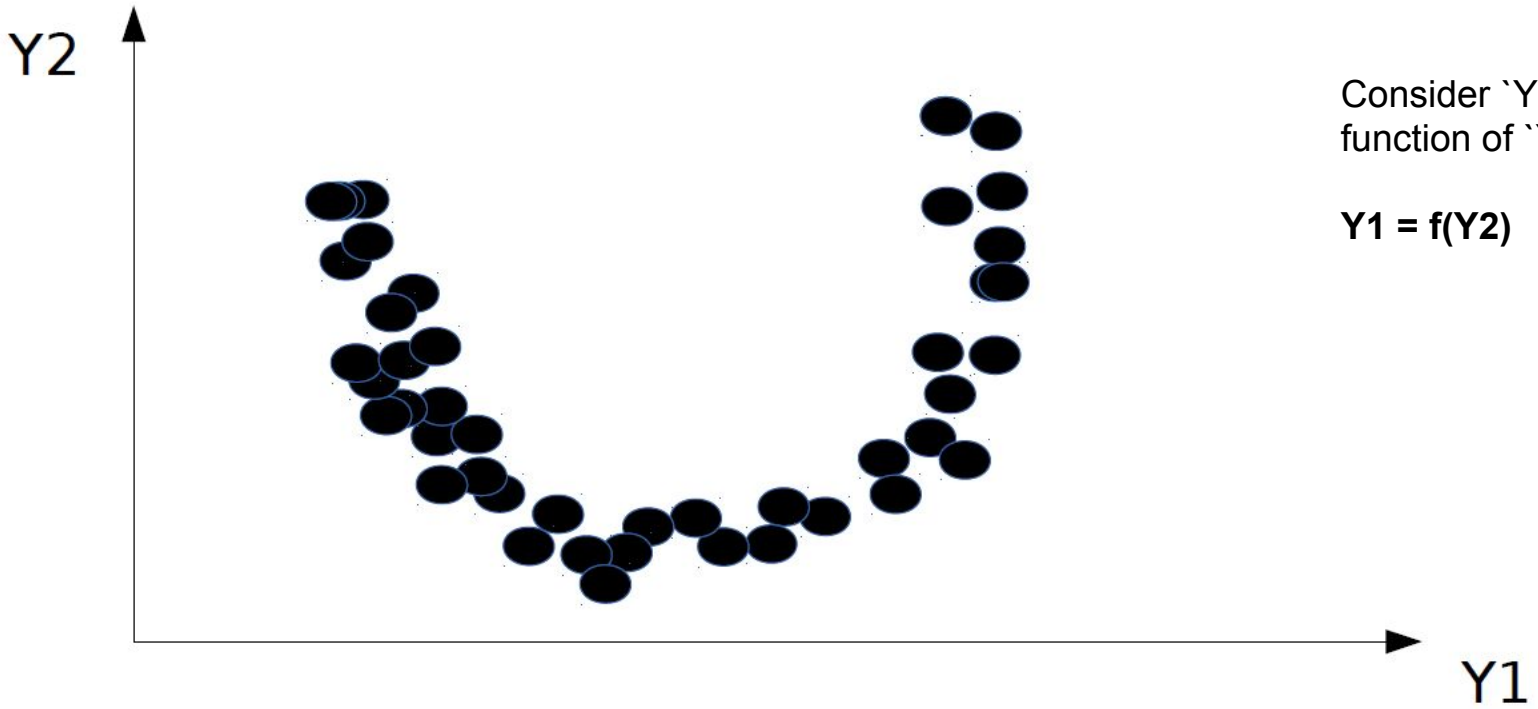
Understanding Blurry Results Due to L1/L2

Image Source

Understanding Blurry Results Due to L1/L2



Understanding Blurry Results Due to L1/L2

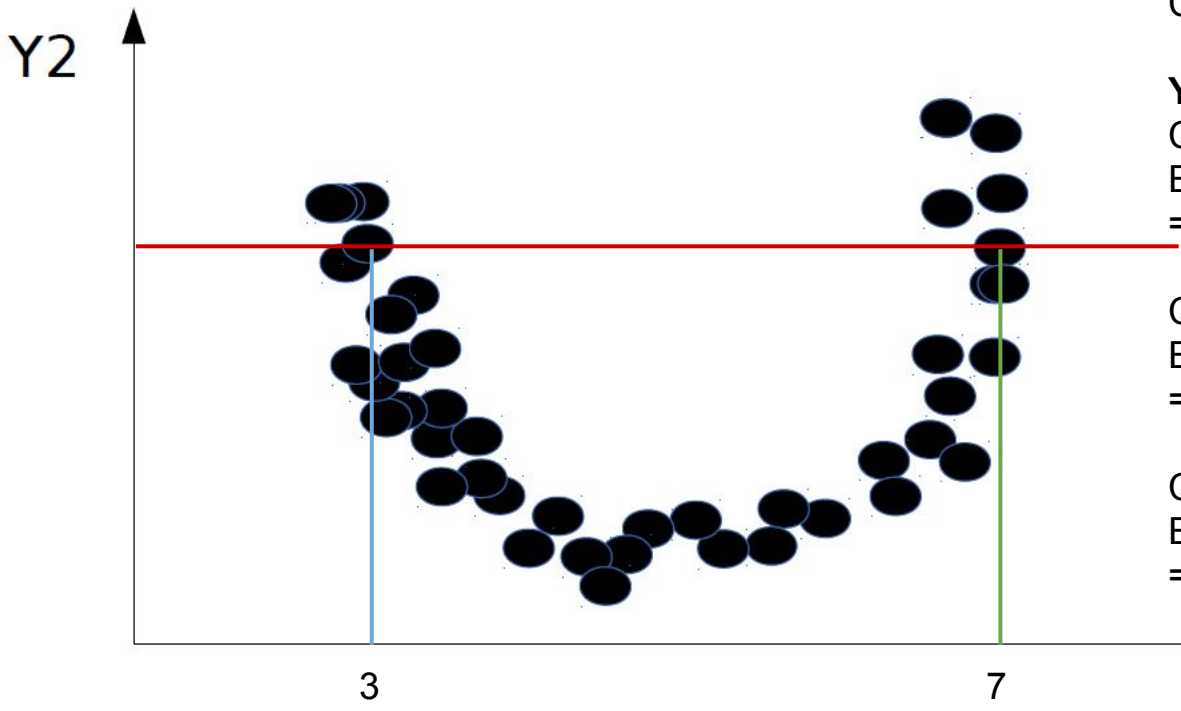


Consider `Y1` as a function of `Y2`

Y1 = f(Y2)

Image Credits: Prof Yann LeCun

Understanding Blurry Results Due to L1/L2



Consider `Y1` as a function of `Y2`

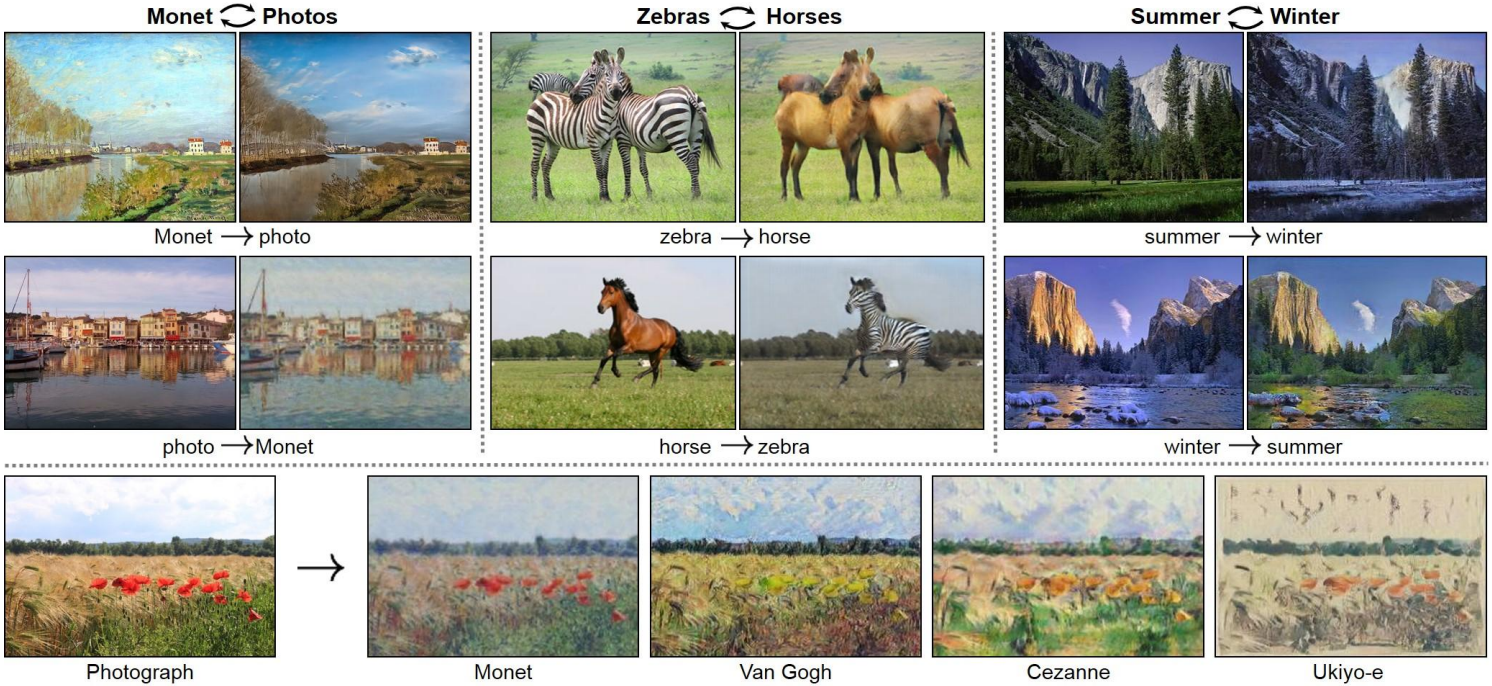
Y1 = f(Y2)
Choose blue
Error = 0.5 * (error(b, b) + error(b, g))
= 0.5 * (0 + 16) = 8

Choose green
Error = 0.5 * (error(g, b) + error(g, g))
= 0.5 * (16 + 0) = 8

Choose Mid-point
Error = 0.5 * ((error(m, b) + error(m, g))
= 0.5 * (4 + 4) = 4

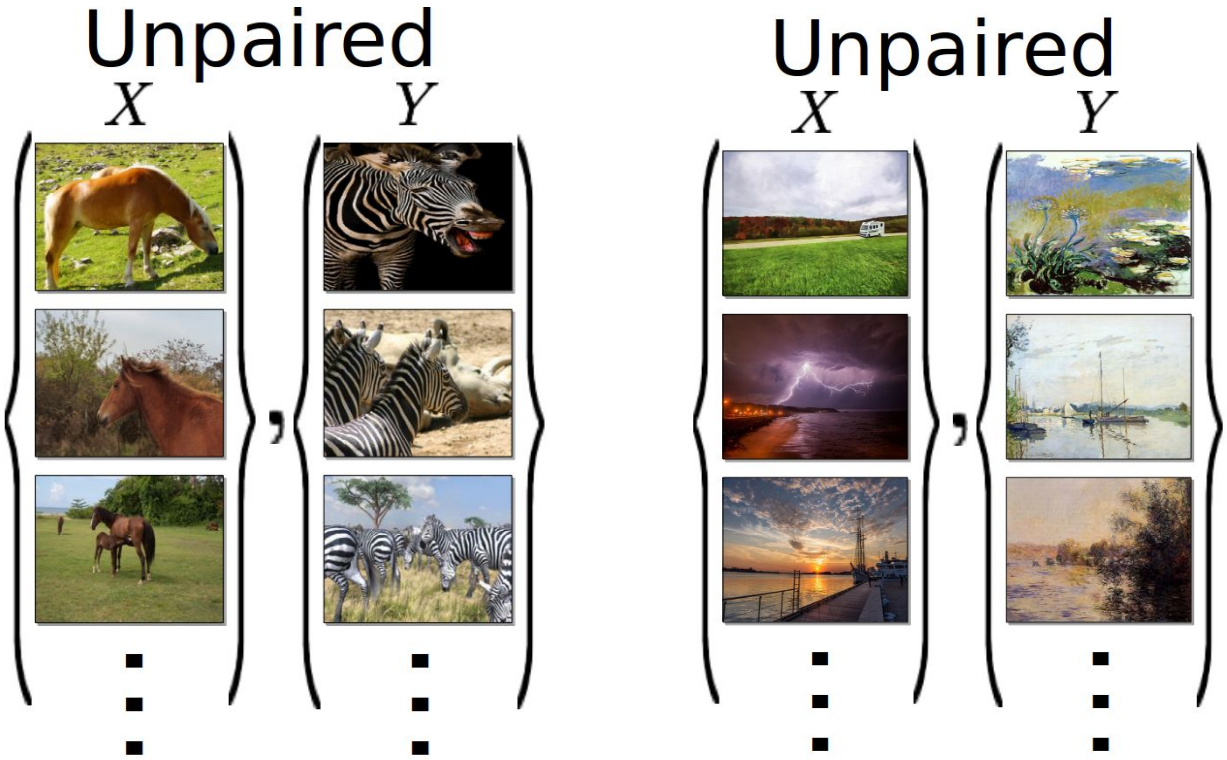
Image Credits: Prof Yann LeCun

Unpaired Image-to-Image Translation



Zhu, J., Park, T., Isola, P., & Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *ICCV (2017)*

Snapshot of Data

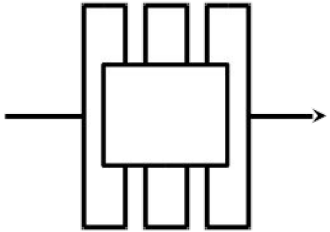


Credits: Jun-Yan Zhu

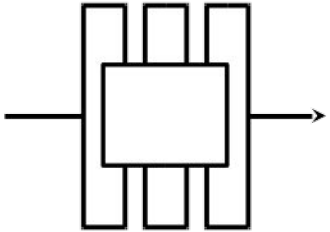
What About This Method?

- Train the generator to generate something from horses
- Sample a generated batch, and a batch of zebras
- Ask the discriminator to differentiate between them
- Train like a normal GAN

Problem

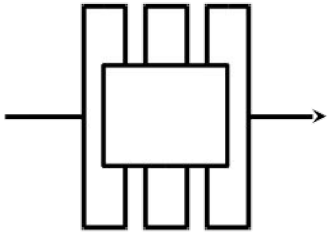


Generator

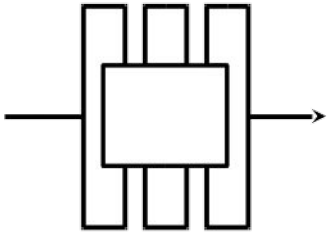


Discriminator

Real!



Generator

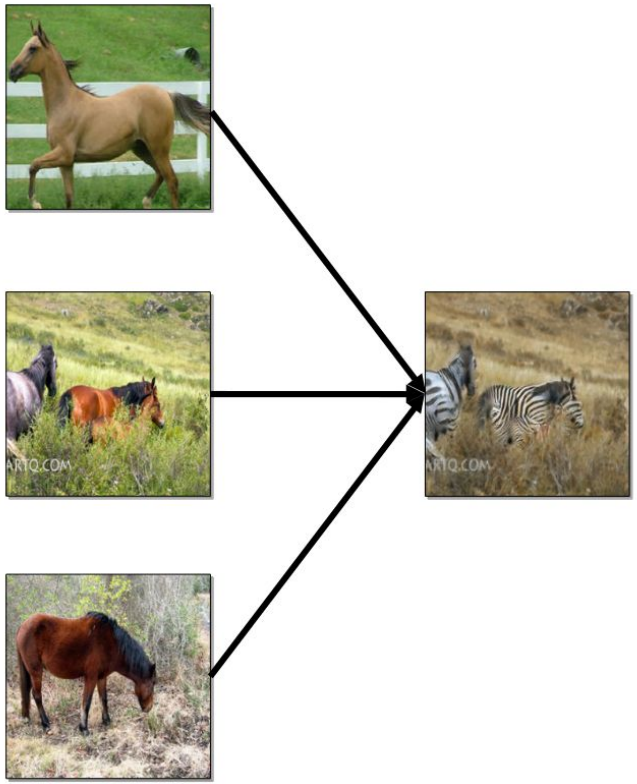


Discriminator

Real too!

Credits: Jun-Yan Zhu

Moreover



mode collapse!

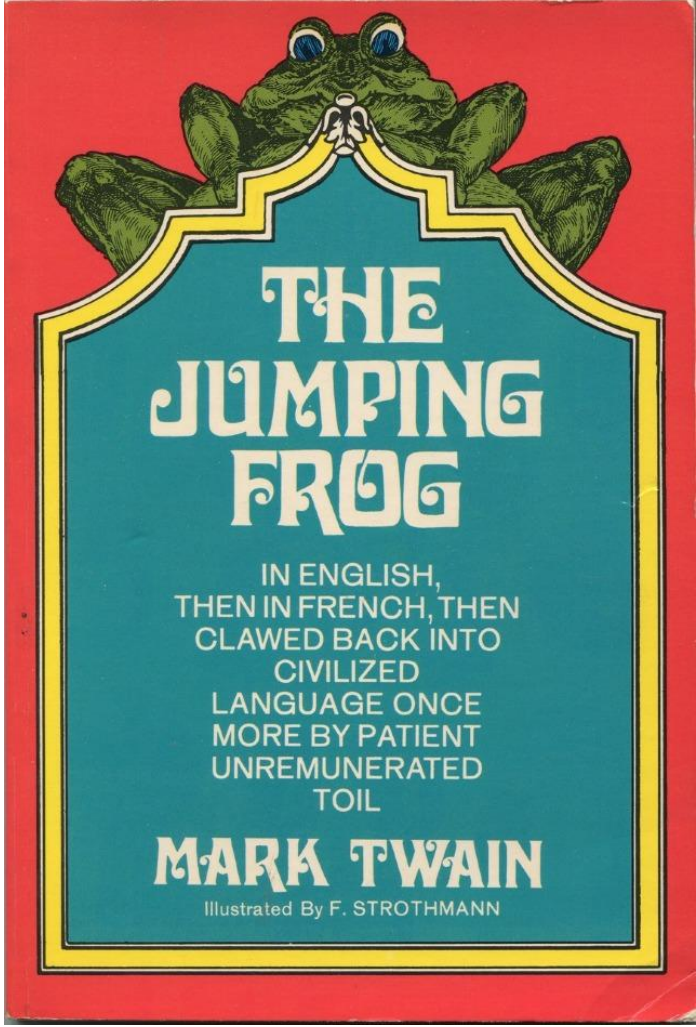
Credits: Jun-Yan Zhu

Cycle Consistency

Original: *“There was a feller here once by the name of Jim Smiley, in the winter of ’49 or maybe it was the spring of ’50 I don’t recollect exactly”*

Back Translation: *“It there was one time here an individual known under the name of Jim Smiley; it was in the winter ’49, possibly well at the spring of ’50, I no me recollect not exactly.”*

— Mark Twain



Slide Inspired by Prof Alexei Efros’ Lecture: Self-supervision, Meta-supervision, Curiosity: Making Computers Study Harder

Image Source

Cycle Consistency

- **G(F(x)) = x**, and
- **F(G(y)) = y**
- F and G are inverses of each other
- F and G are neural networks
- We will learn a function and its inverse together

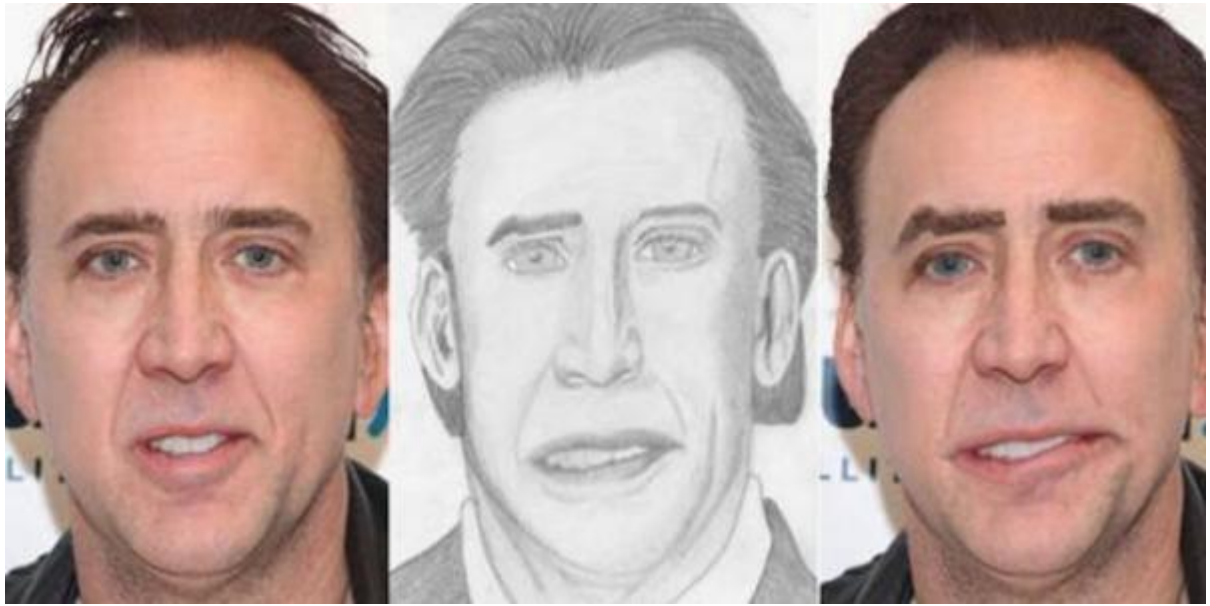
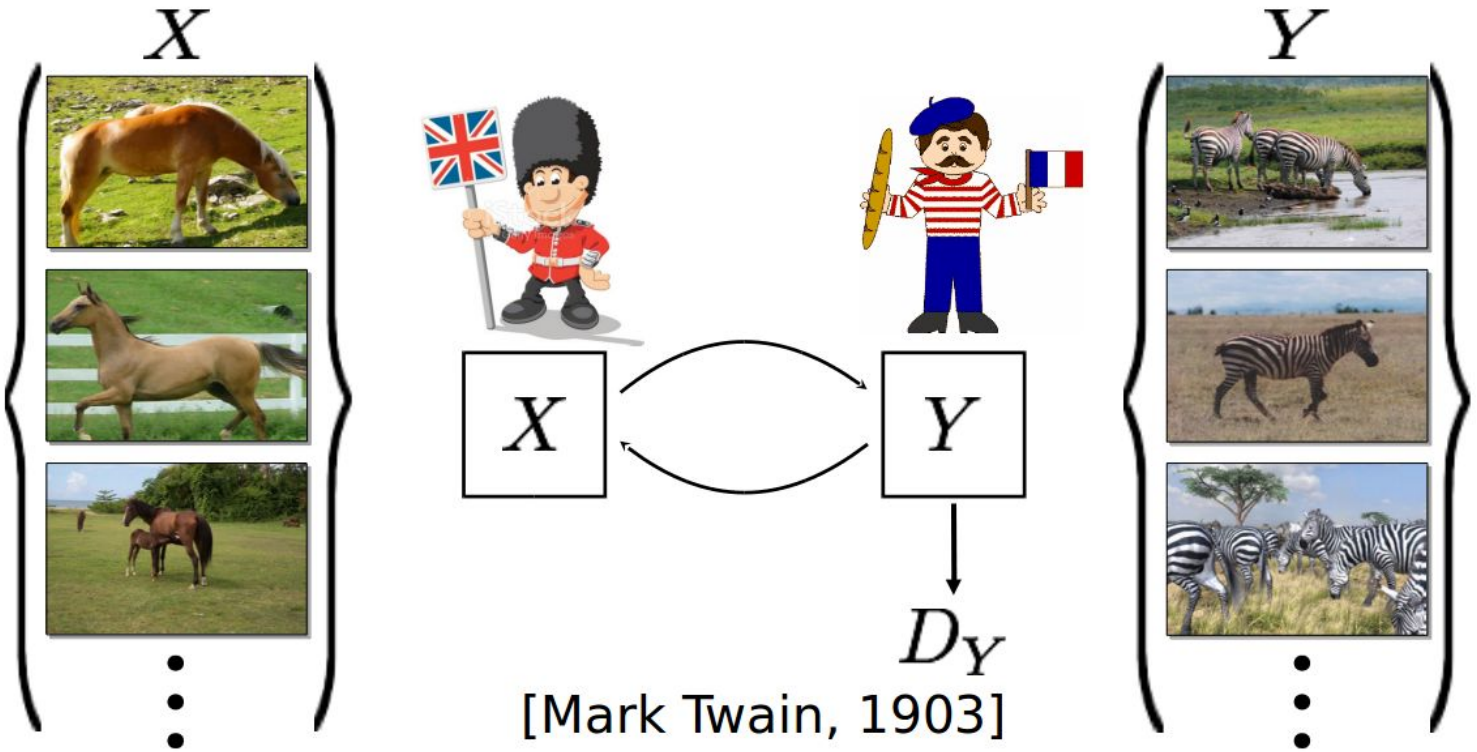


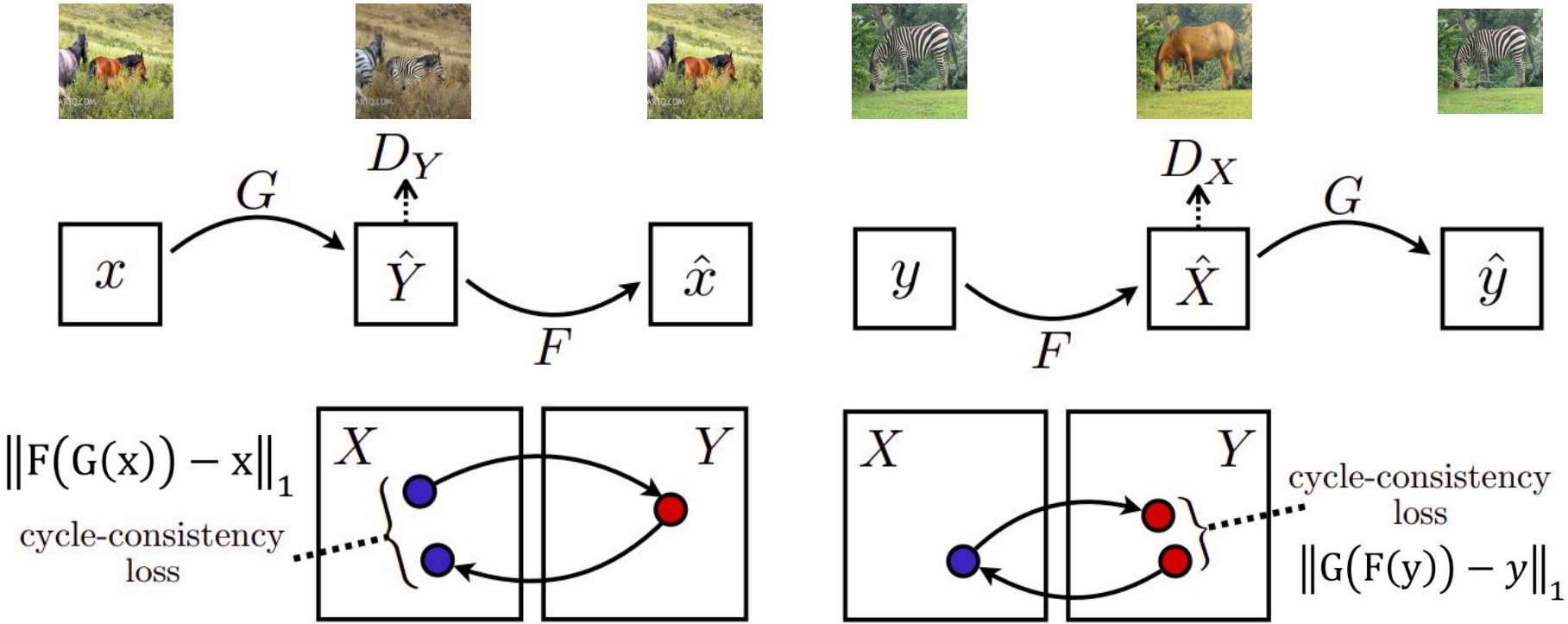
Image Source

Cycle Consistency



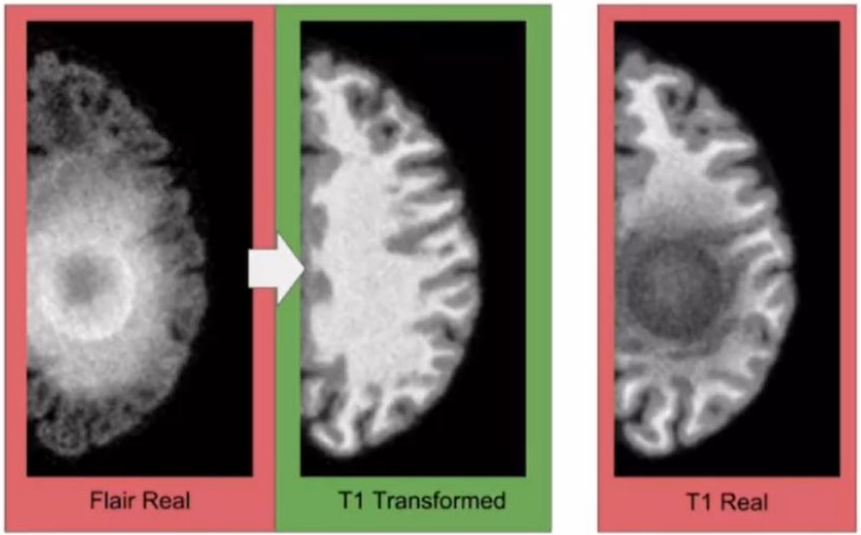
Credits: Jun-Yan Zhu

Cycle Consistency Loss

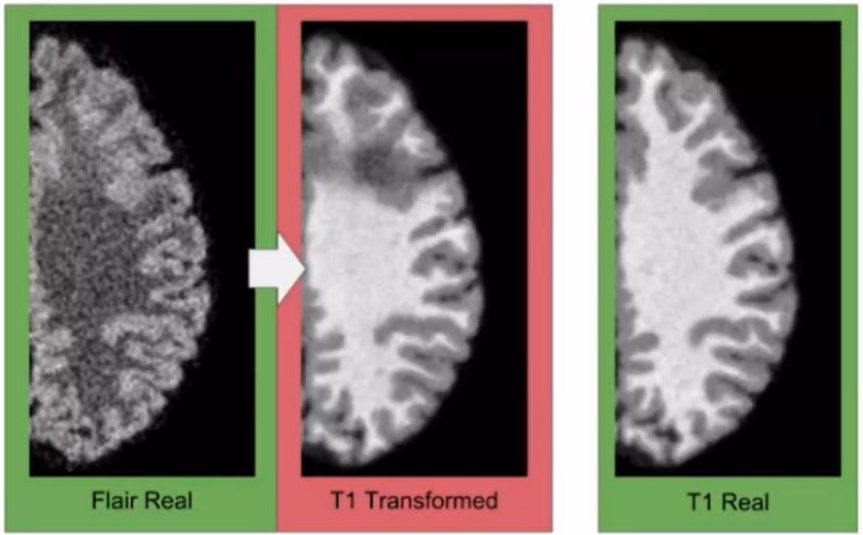


Zhu, J., Park, T., Isola, P., & Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *ICCV (2017)*

Applications



(a) A translation removing tumors



(b) A translation adding tumors

Source

Applications

- Deepfakes: Cadbury Ad Campaign
- Superresolution (improving video game quality, MRIs)
- Image Restoration
- Background Matting
- Image Compression
- JPEG Artifact Removal
- Fashion Models Generation

Applications

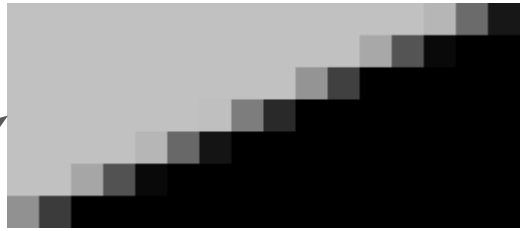
Super-resolution

Handcrafted Upsampling Algorithms

Trade-offs

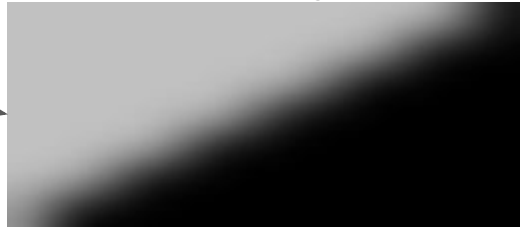
Test Image

Nearest Neighbour



Aliasing

Bilinear with Blur



Blurring

Bicubic

Edge Halos

Source

Super-resolution

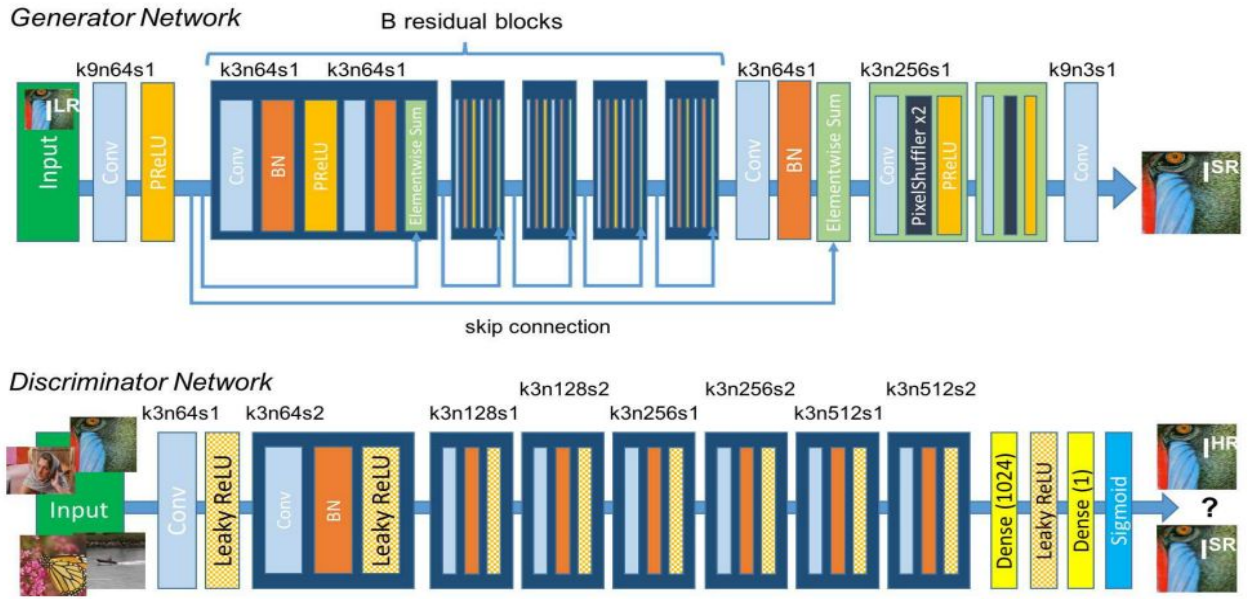


Figure 4: Architecture of Generator and Discriminator Network with corresponding kernel size (k), number of feature maps (n) and stride (s) indicated for each convolutional layer.

Ledig, Christian et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *CVPR* (2017)

SRGAN Architecture

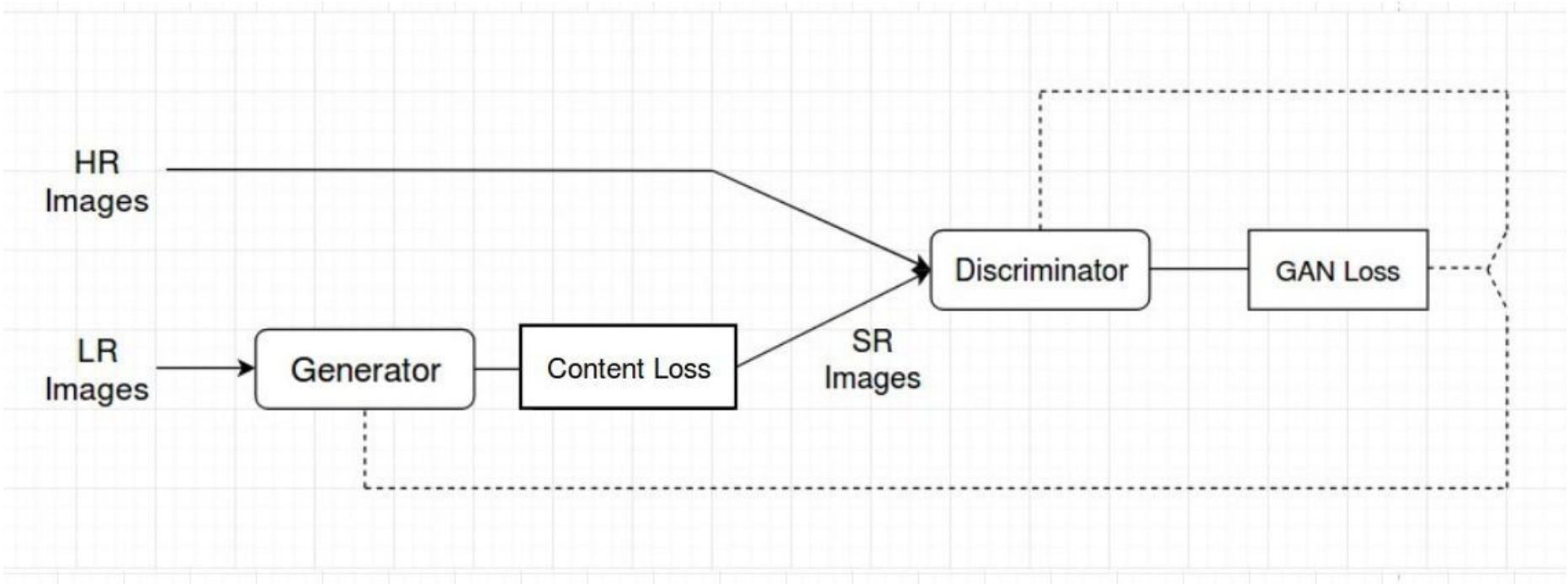


Image Source

SRGAN Architecture

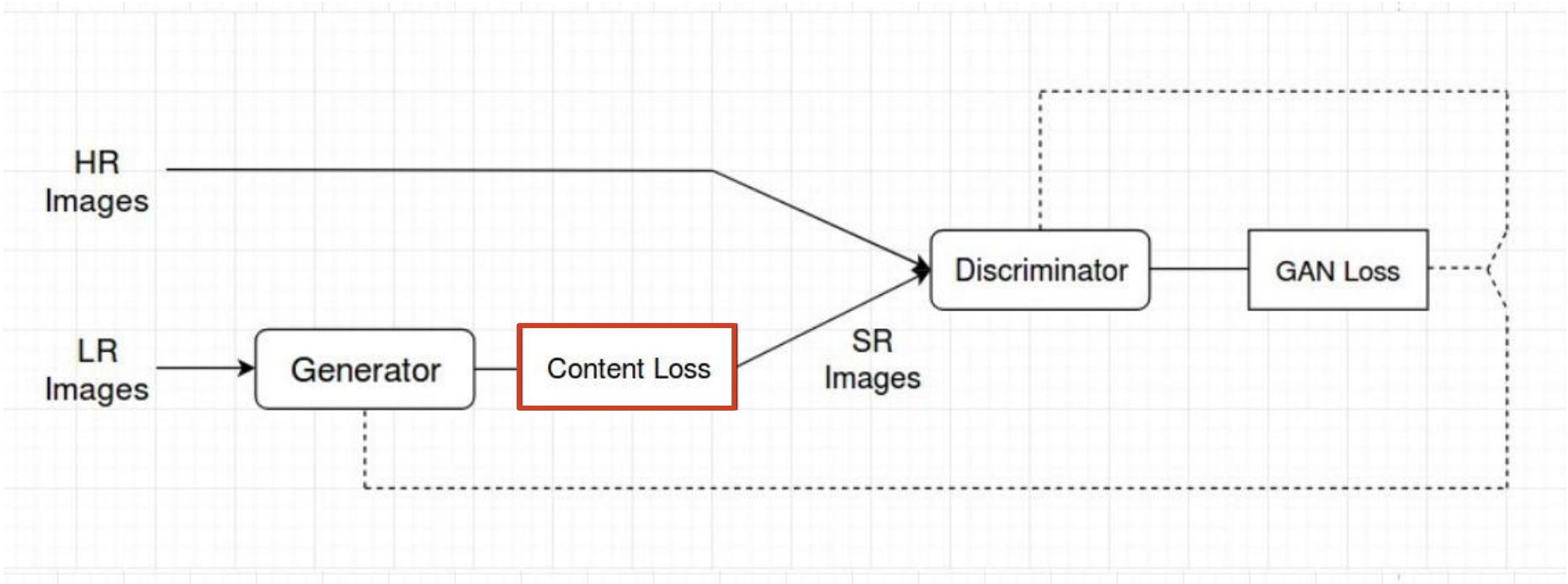


Image Source

Style Transfer

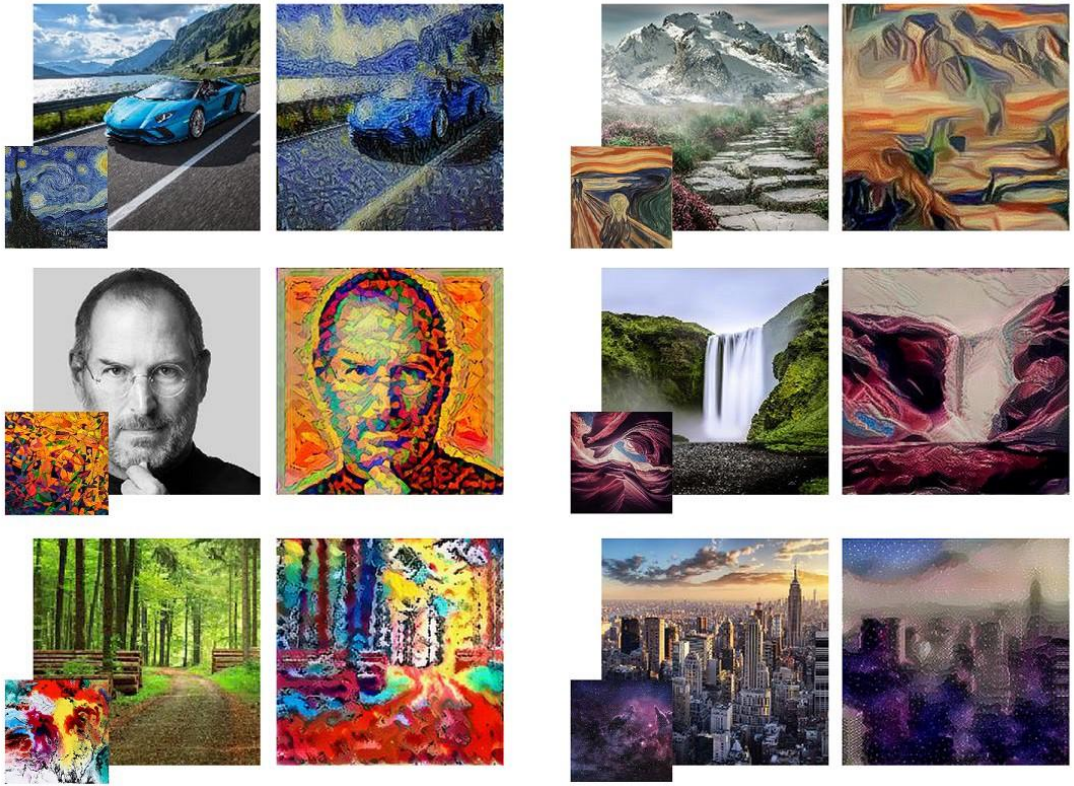
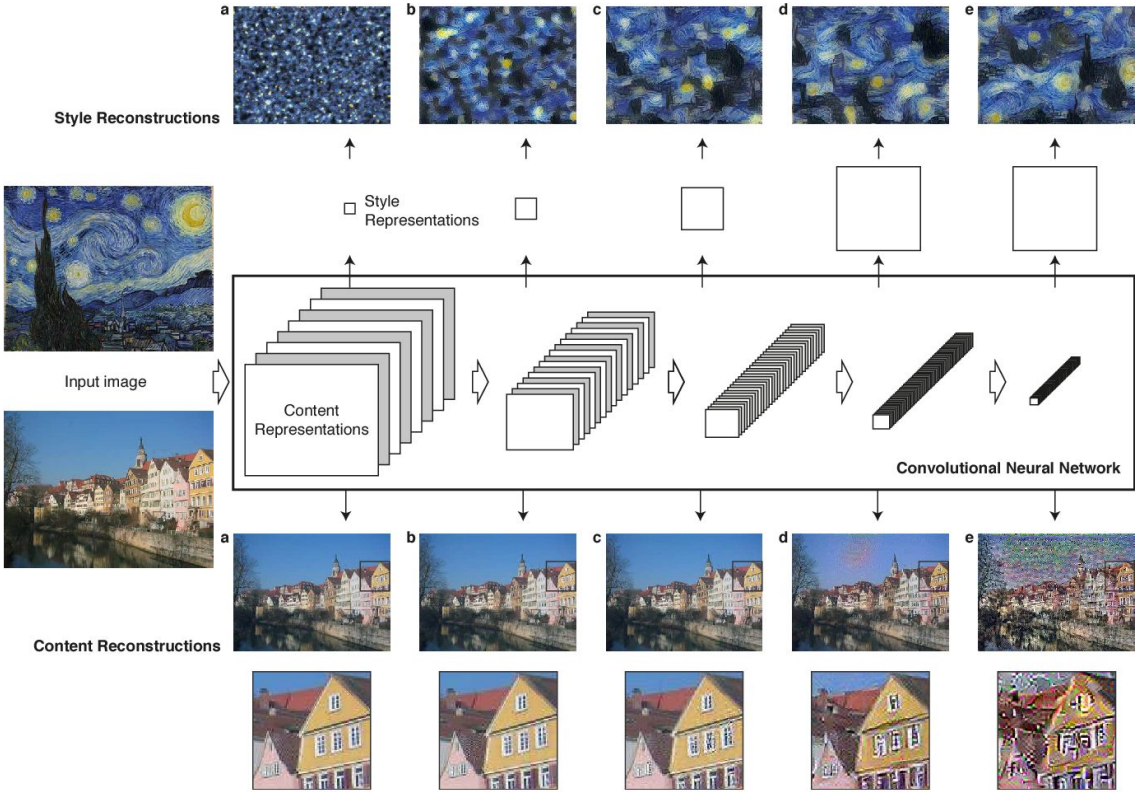


Image Source

Deep Feature Spaces



Gatys, L.A., Ecker, A.S., & Bethge, M. (2016). Image Style Transfer Using Convolutional Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2414-2423.

Deep Features as Perceptual Metric

Zhang, R., Isola, P., Efros, A.A., Shechtman, E., & Wang, O. (2018). The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 586-595.

Content Loss

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2}\sum_{i,j}(F_{ij}^{l} - P_{ij}^{l})^2$$

$\vec{p}$ The original image
$\vec{x}$ The generated image
l Layer
F_{ij}^{l} Activation of the i^{th} filter at position j in the feature representation of $\vec{x}$ in l
P_{ij}^{l} Activation of the i^{th} filter at position j in the feature representation of $\vec{p}$ in l

Source

SRGAN

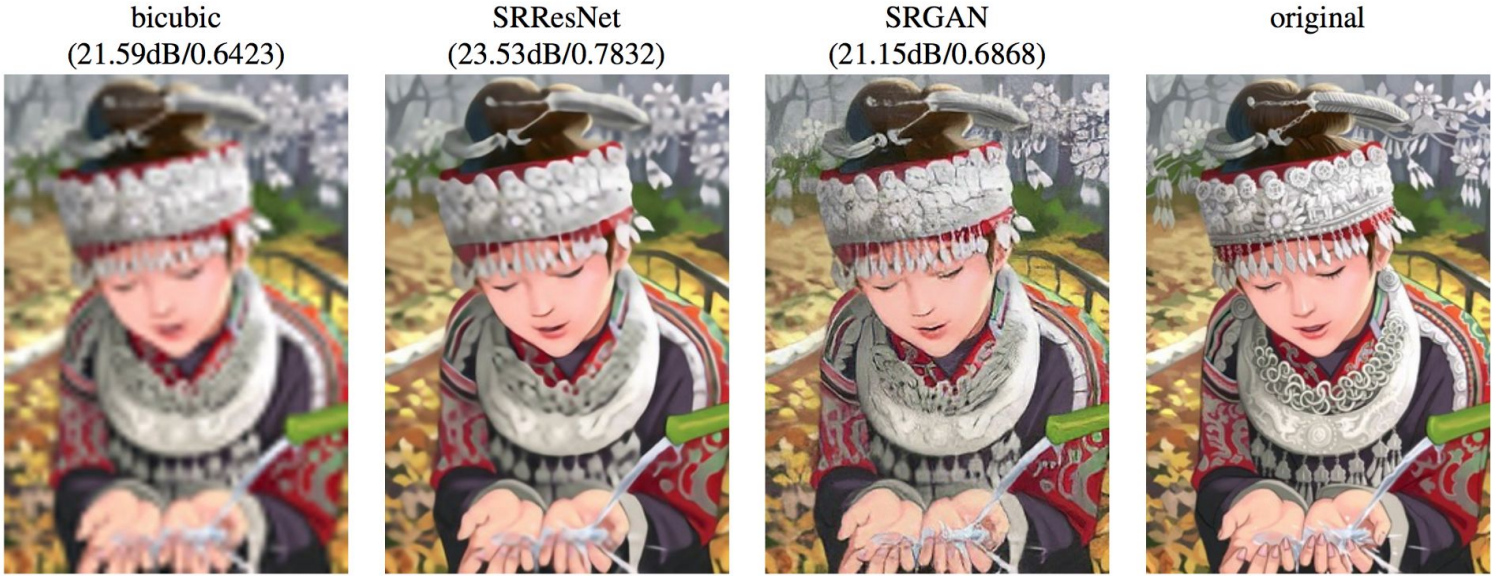
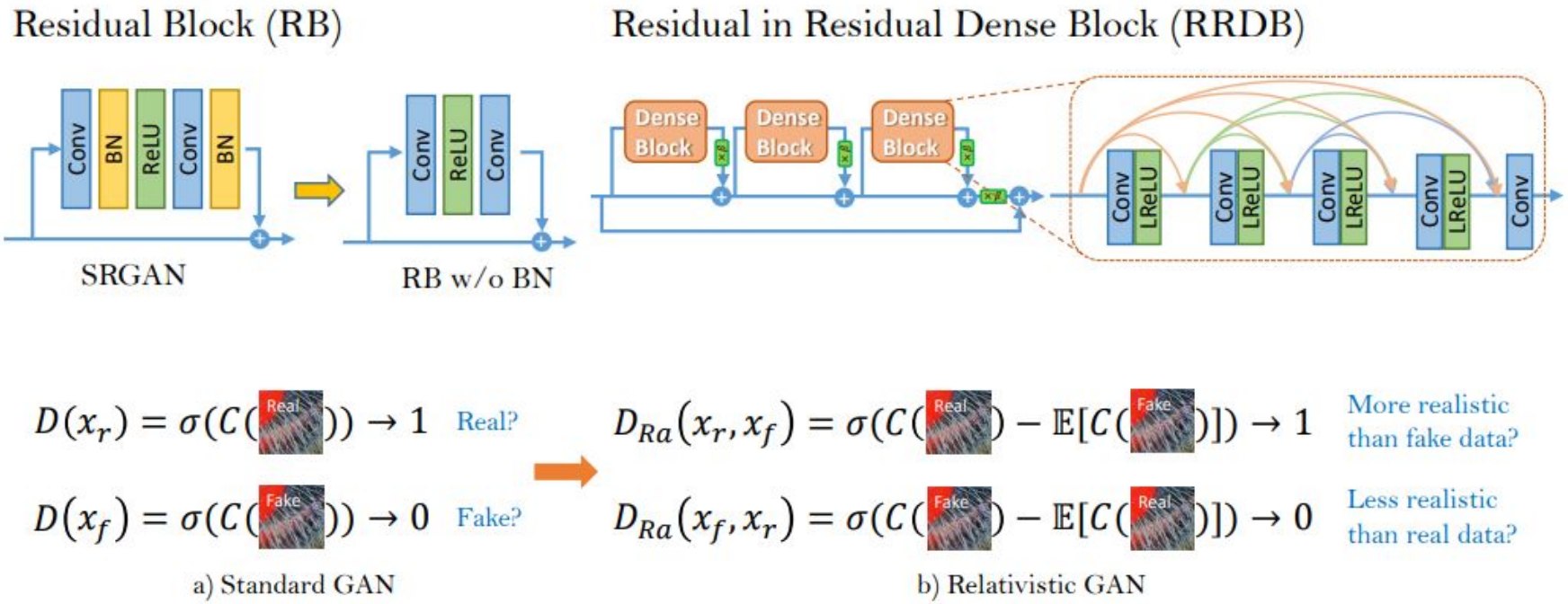


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Ledig, Christian et al. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *CVPR* (2017)

ESRGAN



Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C.C., Qiao, Y., & Tang, X. (2018). ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *ECCV Workshops*.

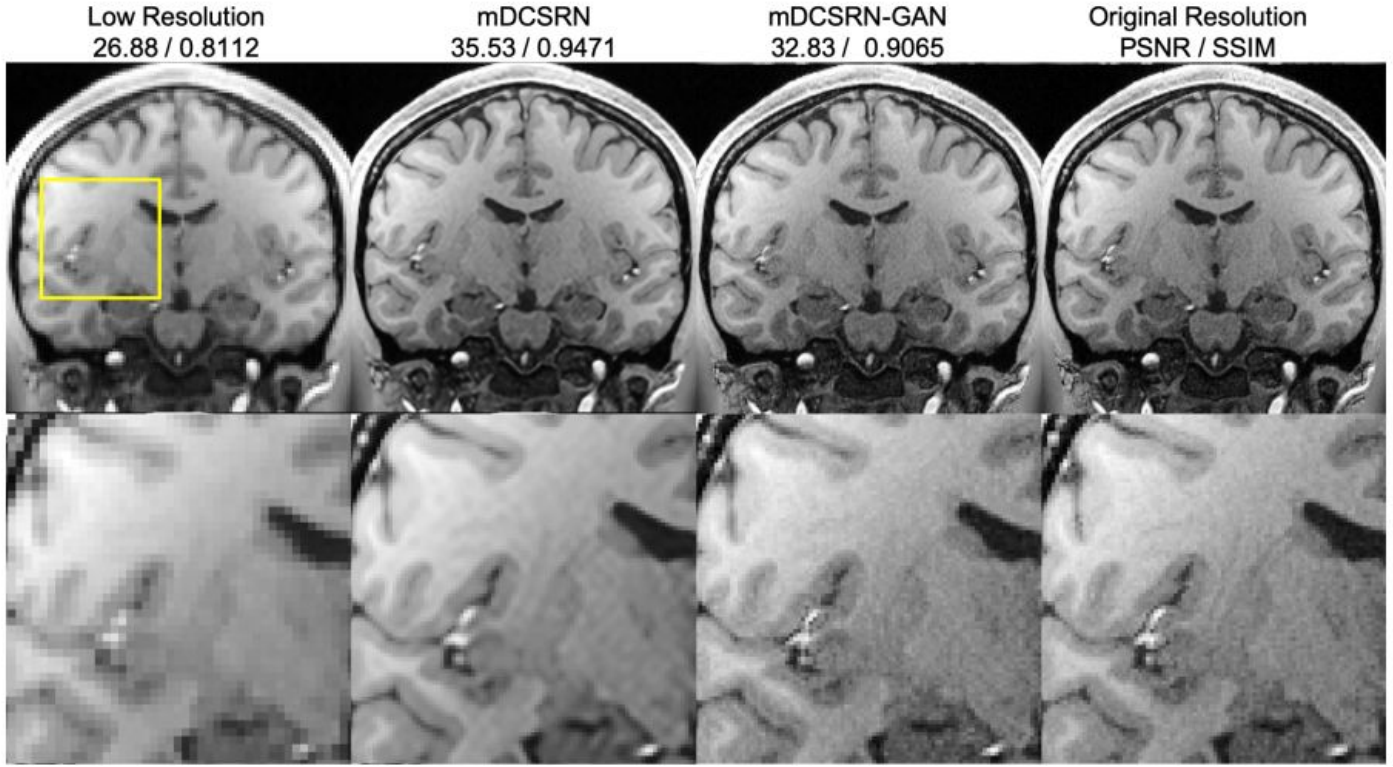
Remastering Old Video Games

Source

Remastering Old Video Games

Source

Improving MRI Quality



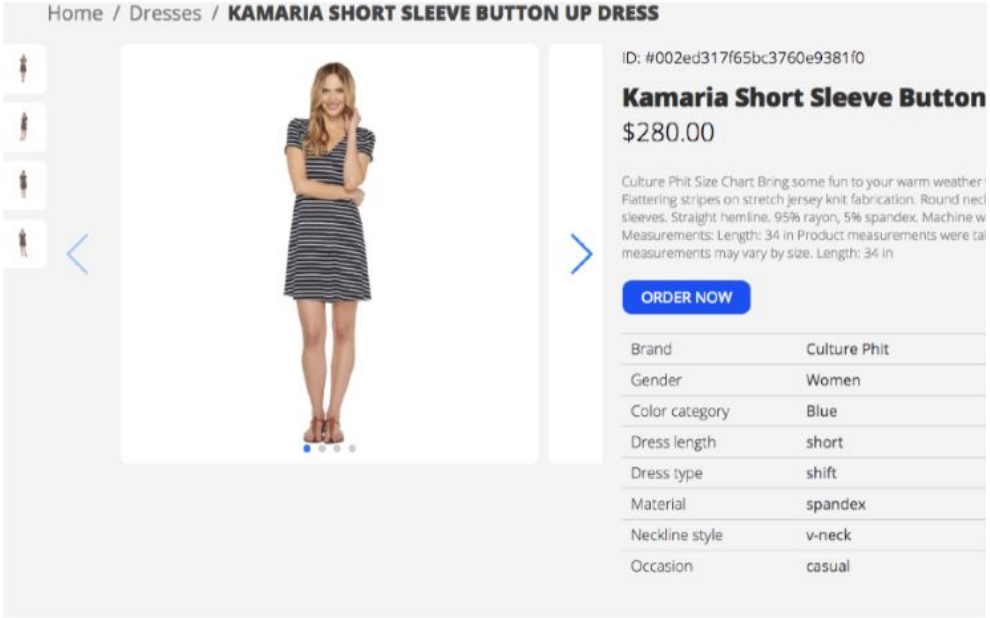
Chen, Y., Christodoulou, A.G., Zhou, Z., Shi, F., Xie, Y., & Li, D. (2020). MRI Super-Resolution with GAN and 3D Multi-Level DenseNet: Smaller, Faster, and Better. *ArXiv, abs/2003.01217*.

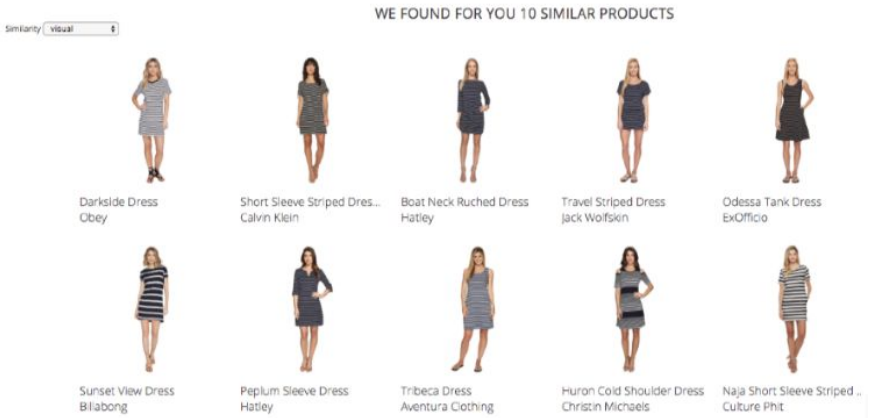
Making Satellite Images Clearer

Source

Detour: Content Loss Application

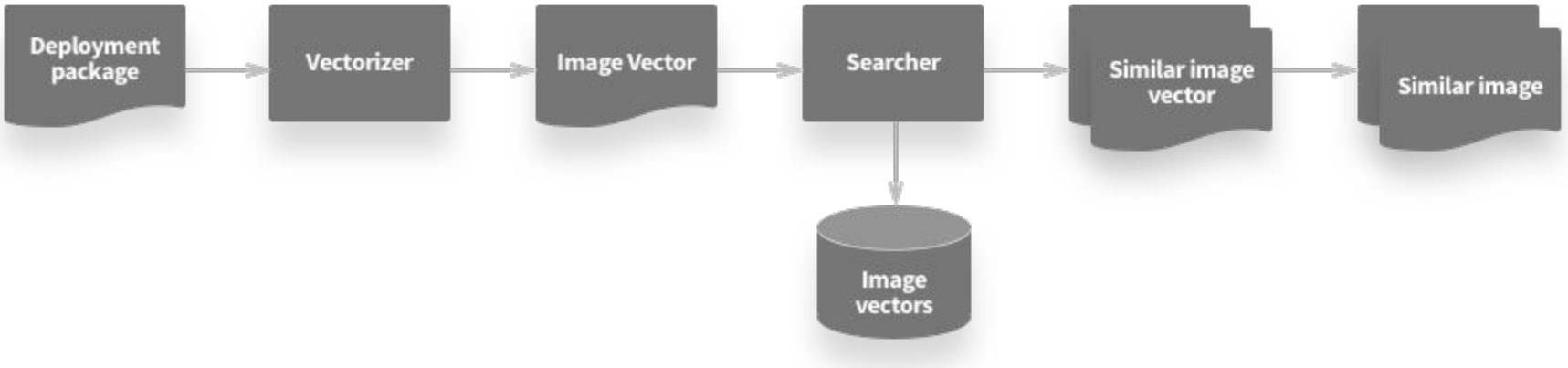
Search Visually Similar Products





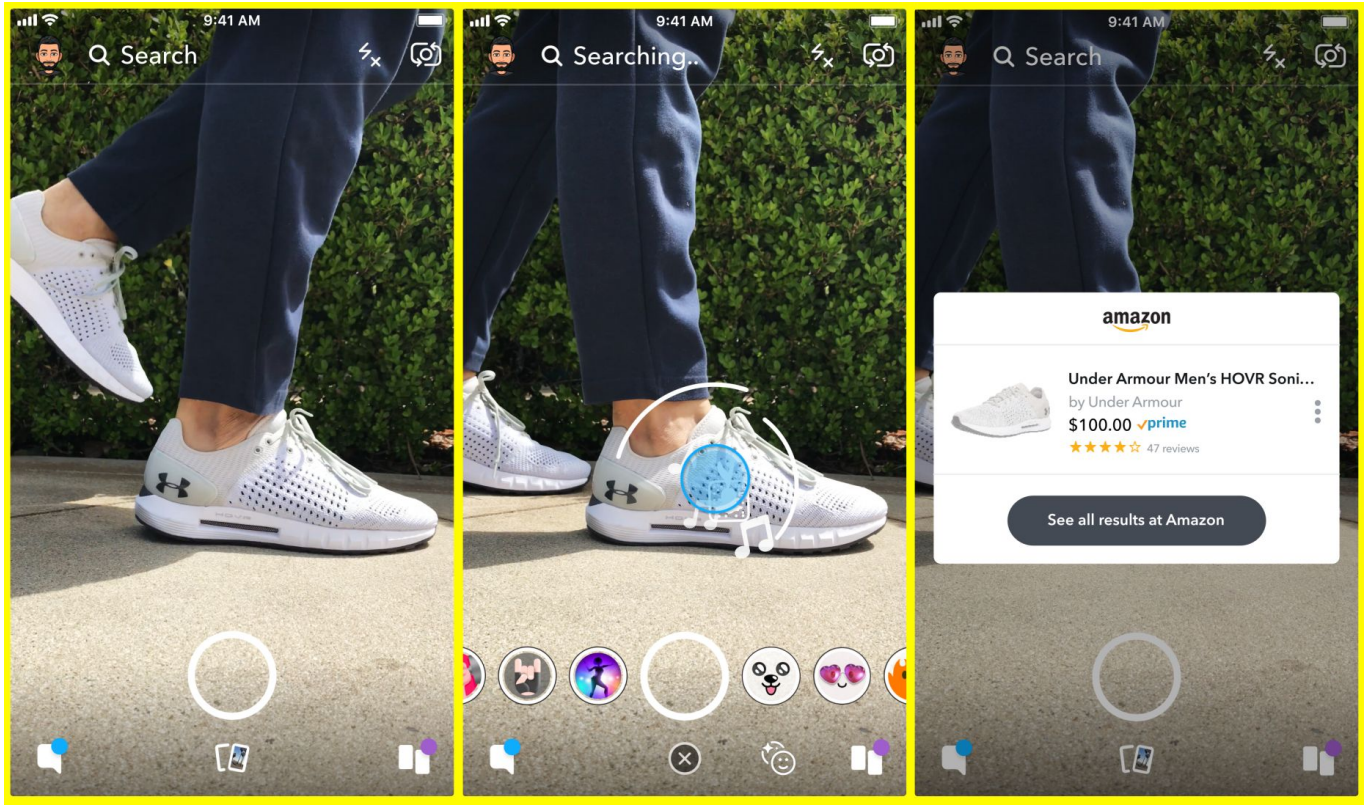
Source

Under the Hood



Source

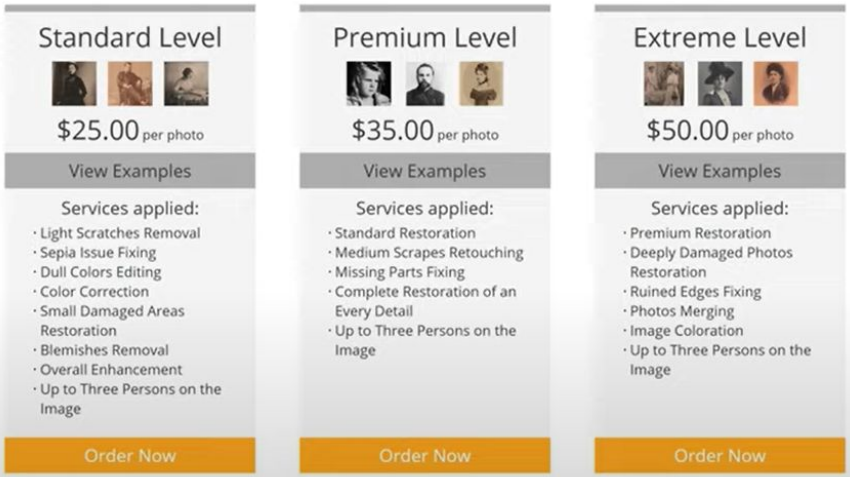
Product Search via Visual Query



Source

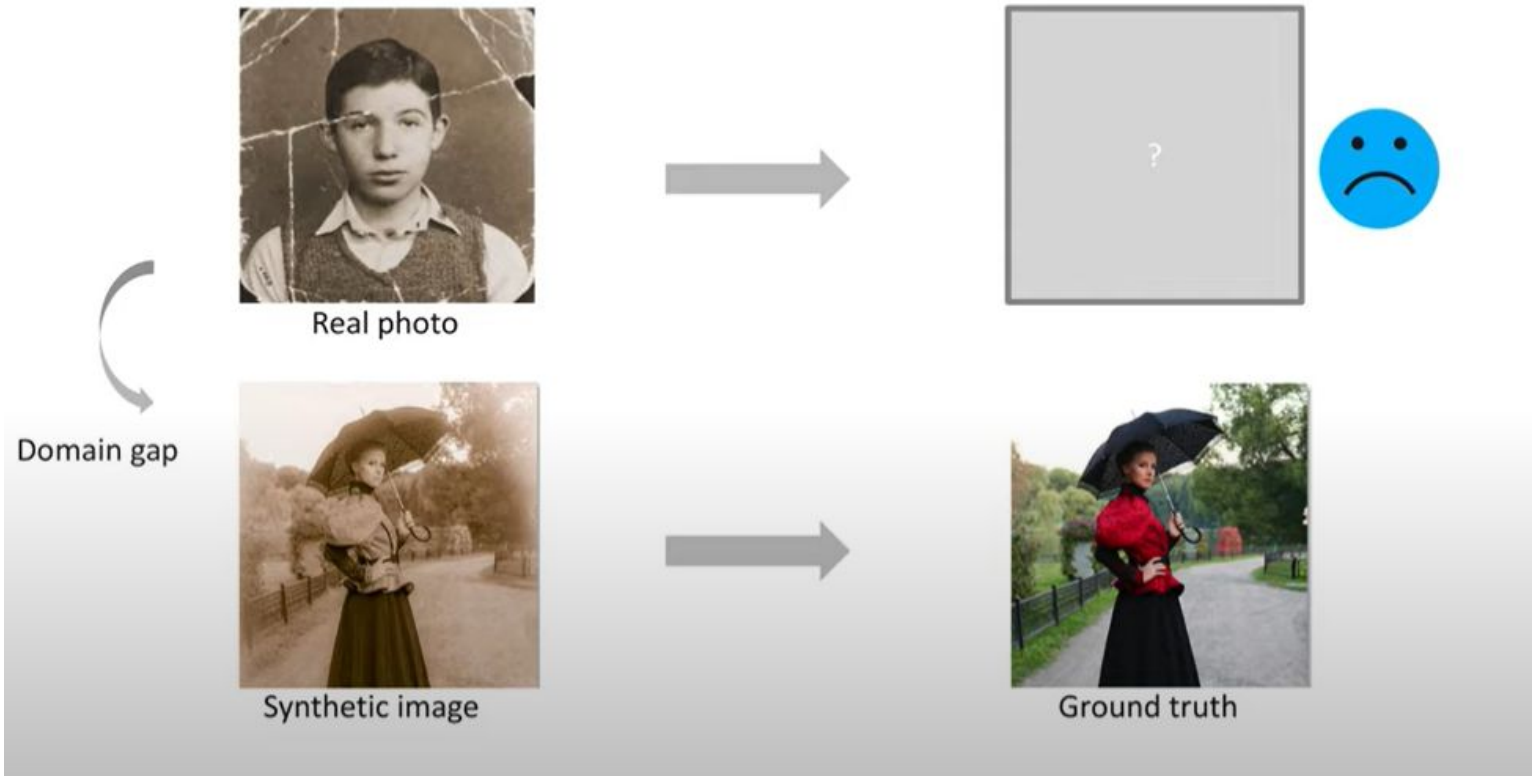
Image Restoration

Bringing Old Photos Back to Life

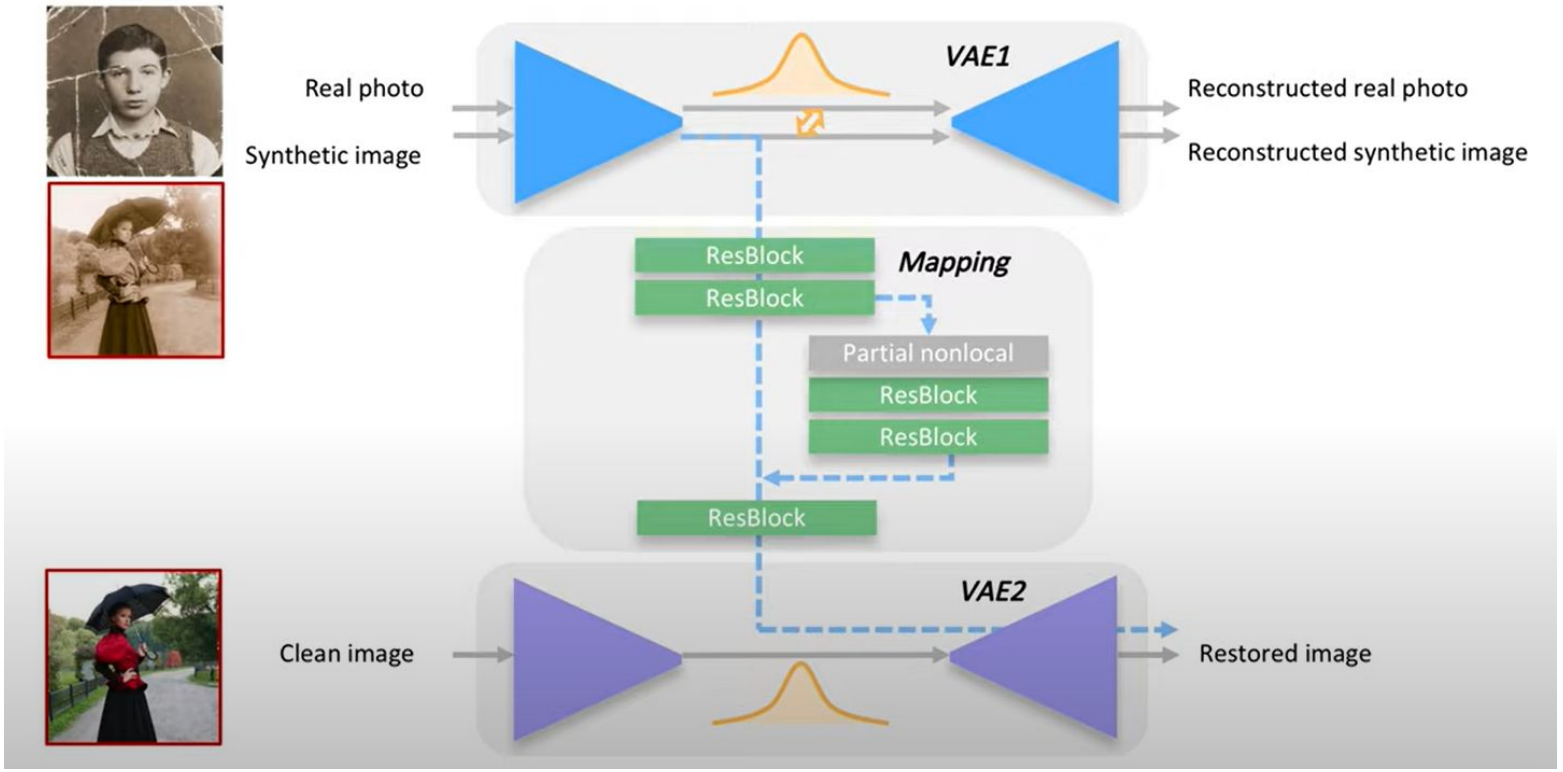


Source

Using Synthetic Data



Solution Via Latent Space Mapping



Wan, Z., Zhang, B., Chen, D., Zhang, P., Chen, D., Liao, J., & Wen, F. (2020). Bringing Old Photos Back to Life. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2744-2754.

Results

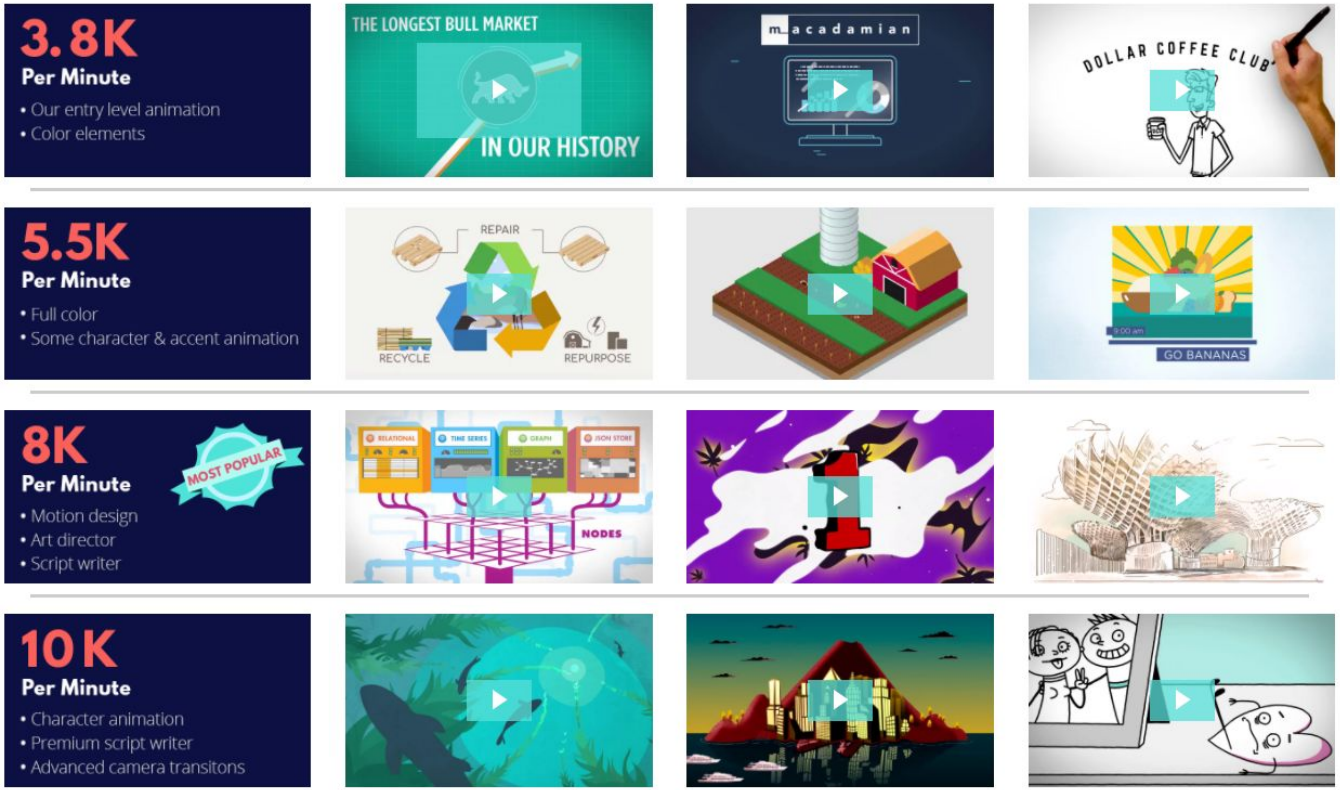
Wan, Z., Zhang, B., Chen, D., Zhang, P., Chen, D., Liao, J., & Wen, F. (2020). Bringing Old Photos Back to Life. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2744-2754.

Frame Interpolation

How are 2D Animations Made

Source

Costs

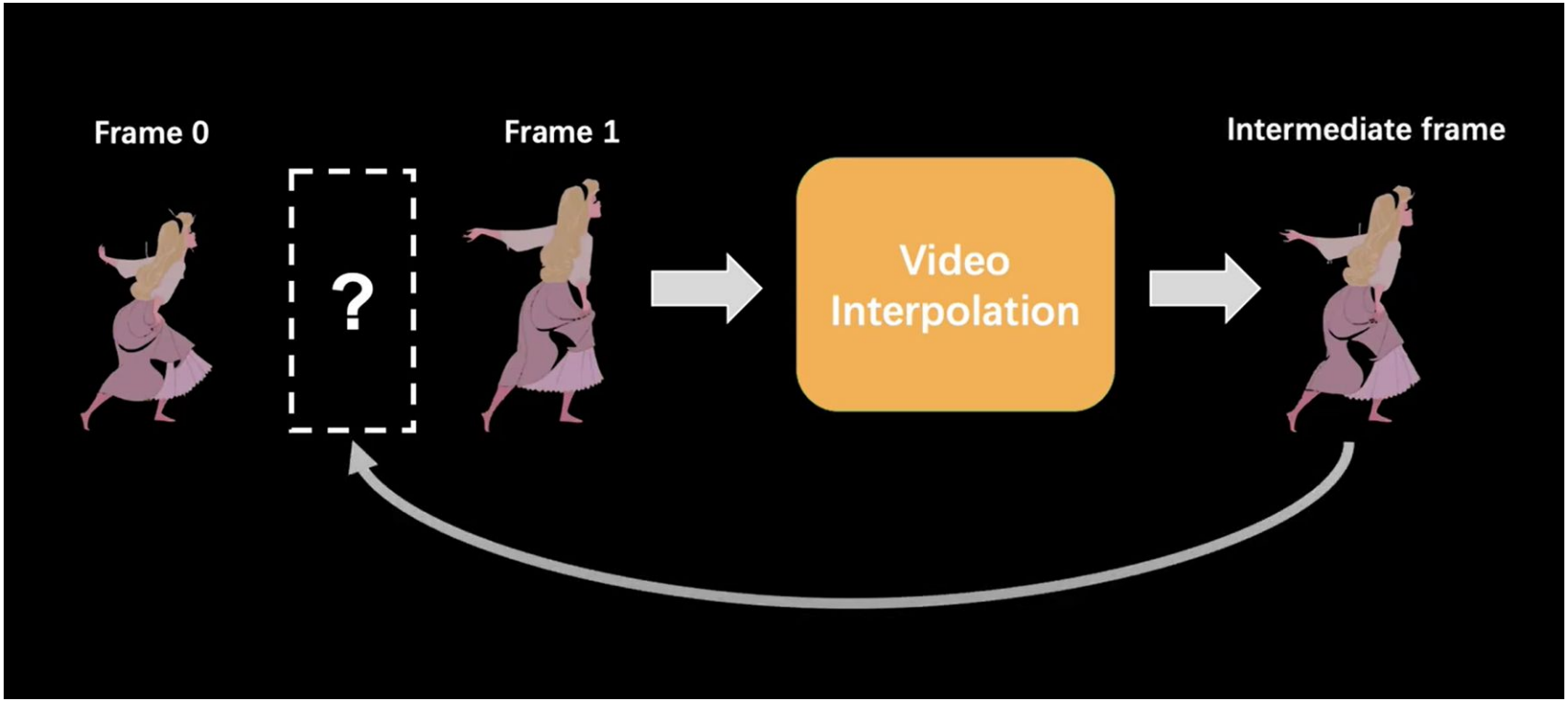


Source

Saving Cost and Time

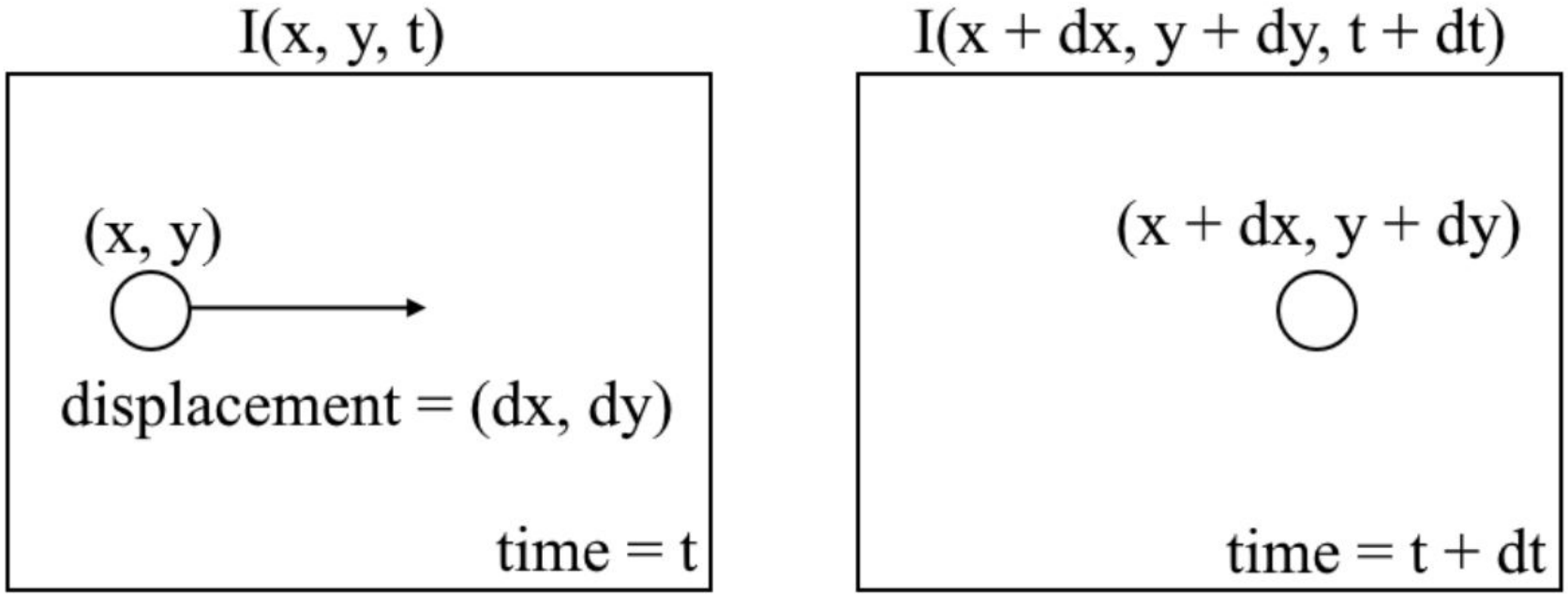
Source

Frame Interpolation



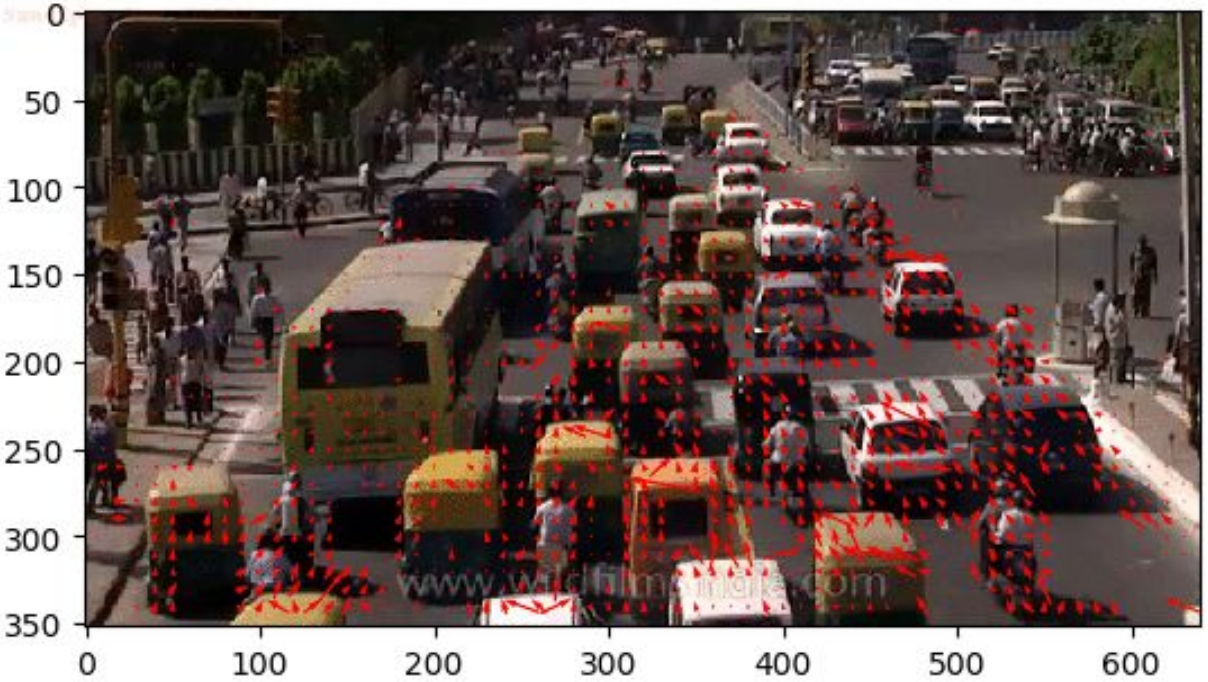
Source

Optical Flow



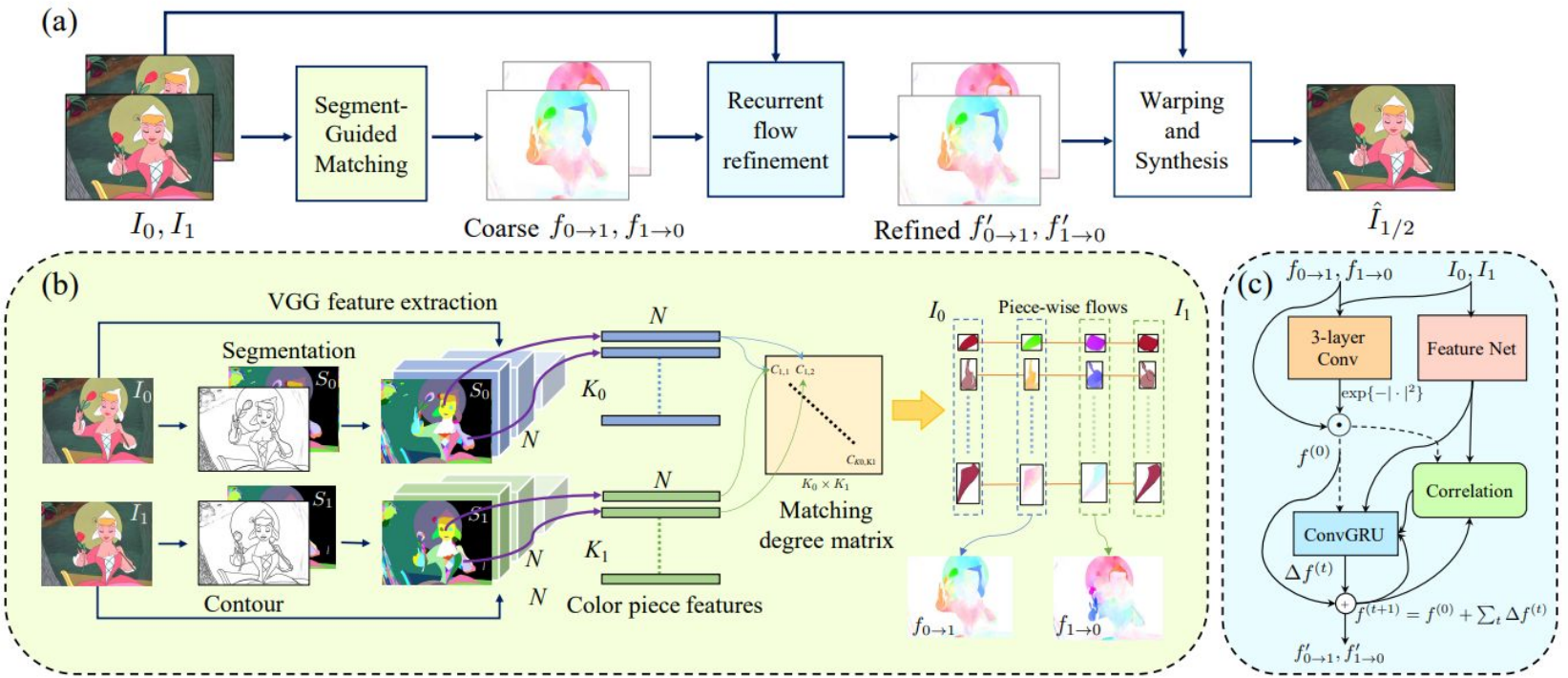
Source

Optical Flow



Source

Anime Interp



Li, S., Zhao, S., Yu, W., Sun, W., Metaxas, D.N., Loy, C.C., & Liu, Z. (2021). Deep Animation Video Interpolation in the Wild. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6583-6591.

Image Matting

Image Matting

- Separating an image into foreground and background
- Matting equation

$$C = F * \alpha + B * (1 - \alpha)$$

- C is the scene, F are the foreground pixels, B are background pixels and alpha is transparency

Semantic Segmentation

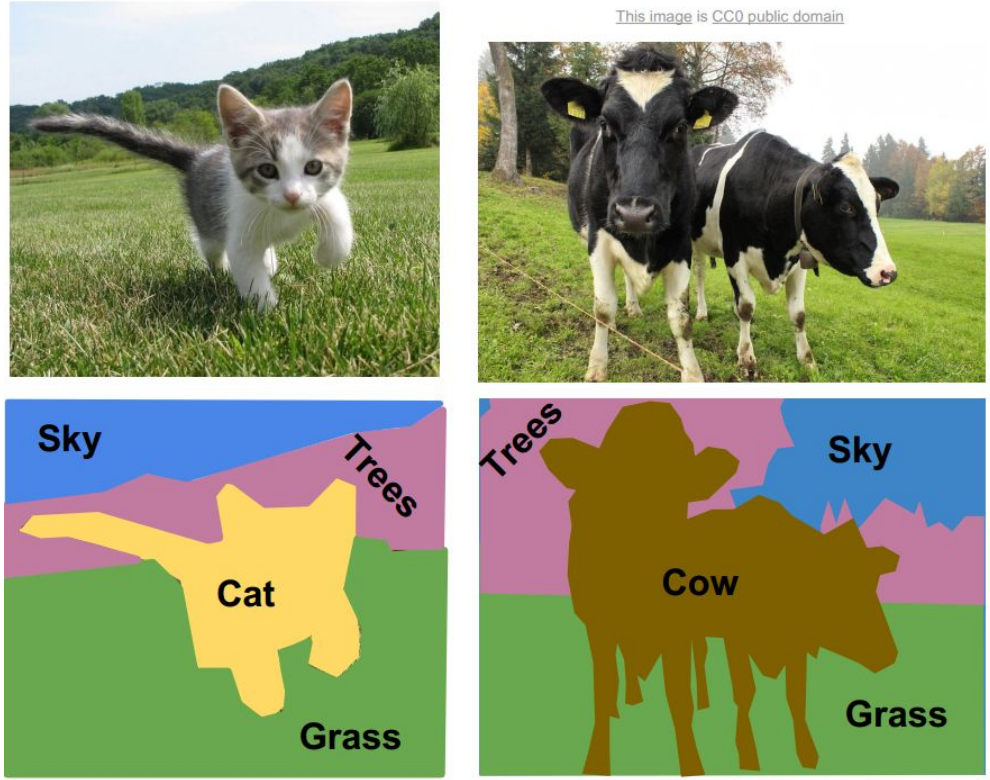
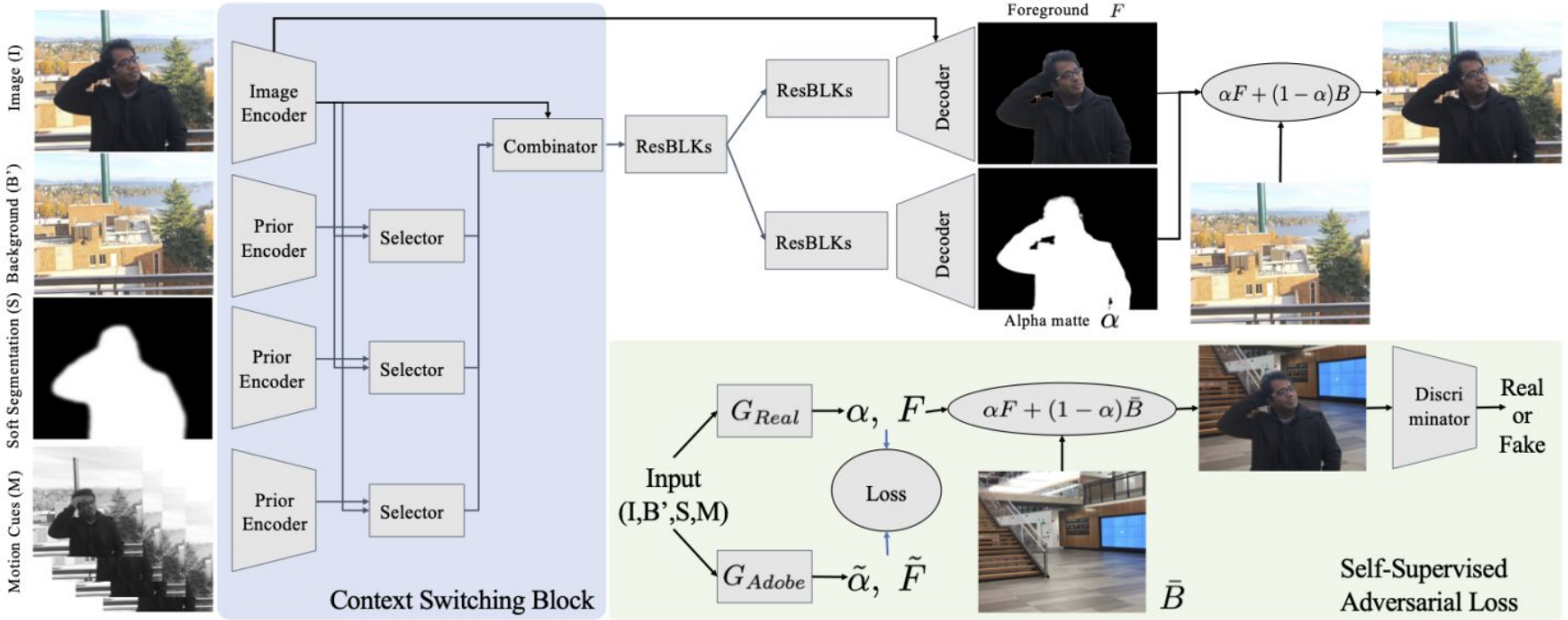


Image Source

Problem With Segmentation

Image Source

Alpha Matte Learning



Sengupta, S., Jayaram, V., Curless, B., Seitz, S.M., & Kemelmacher-Shlizerman, I. (2020). Background Matting: The World Is Your Green Screen. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2288-2297.

Steps

- Supervised Learning Step
- Finetuning Using a GAN
- Using the Supervised Network as a Teacher to prevent trivial solutions

Results

Sengupta, S., Jayaram, V., Curless, B., Seitz, S.M., & Kemelmacher-Shlizerman, I. (2020). Background Matting: The World Is Your Green Screen. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2288-2297.

Applications

Image Source

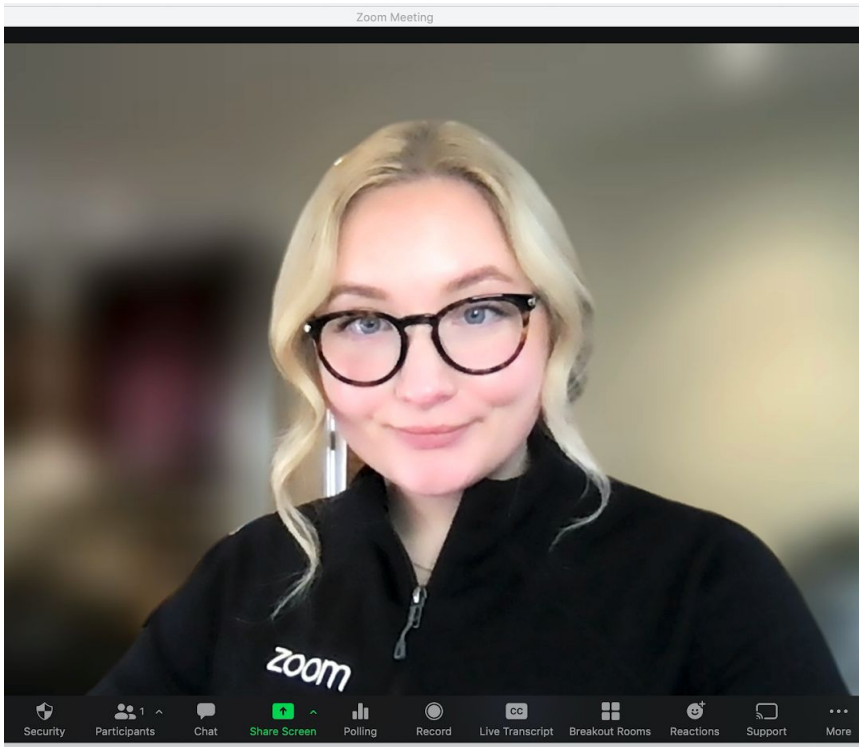


Image Source

Applications

Image Source

Applications

Image Source

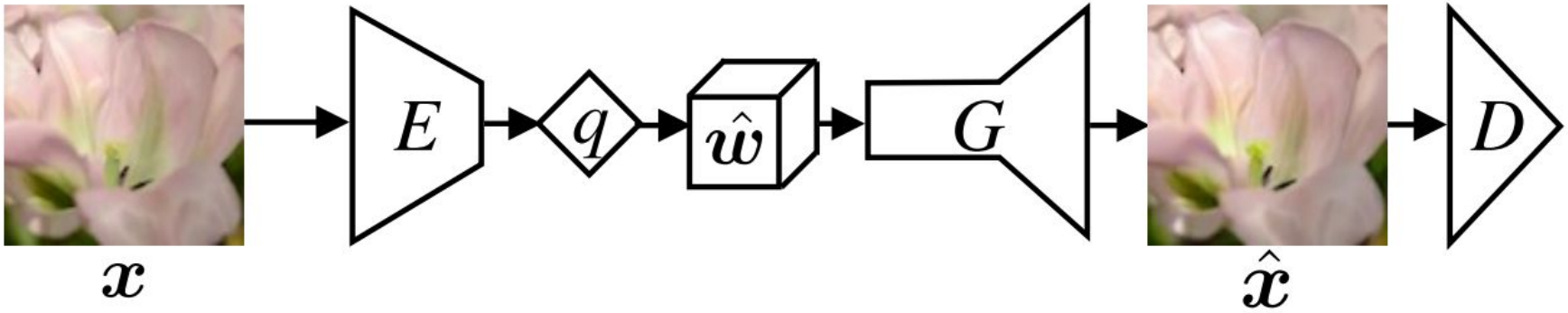
Data Compression

Motivation

- Handcrafted algorithms lack the “understanding” of the content they are compressing
- All pixels are treated equally thus causing distortions and artifacts

JPG Artifacts

Data Compression



Agustsson, E., Tschannen, M., Mentzer, F., Timofte, R., & Gool, L.V. (2019). Generative Adversarial Networks for Extreme Learned Image Compression. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 221-231.

Loss for Compression

$$\min_{E,G} \quad \mathcal{L}_{\text{GAN}} + \lambda \mathbb{E}[d(\boldsymbol{x}, G(\boldsymbol{z}))] + \beta H(\hat{\boldsymbol{w}})$$

d: L1/L2 distance

H: Entropy

HiFiC vs JPG

HifiC | JPG | Original






Dynamic Ad Creation

Facial Reenactment

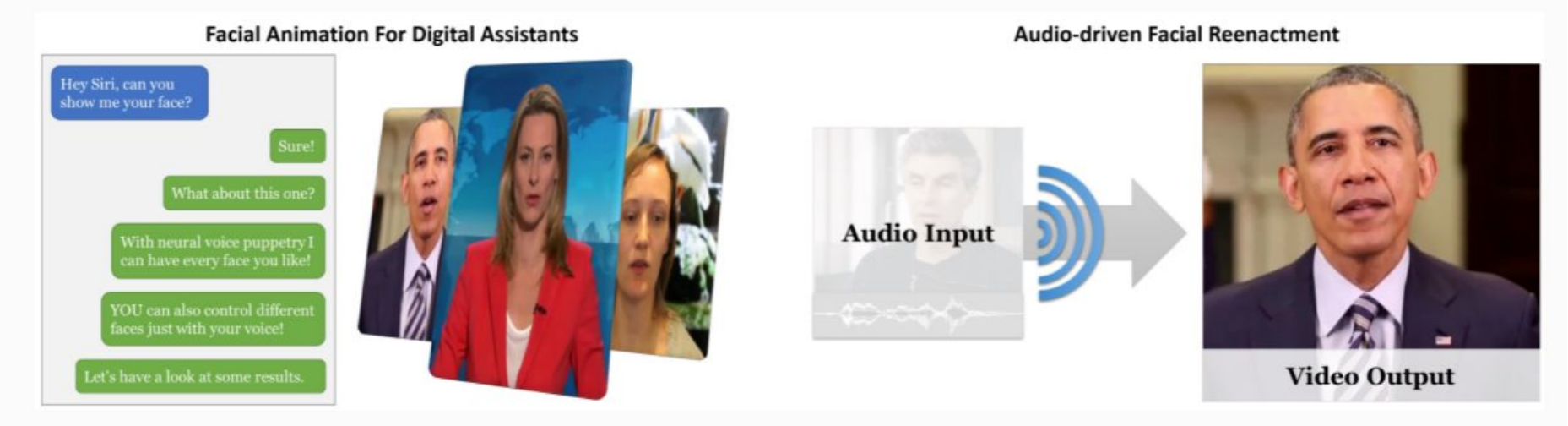
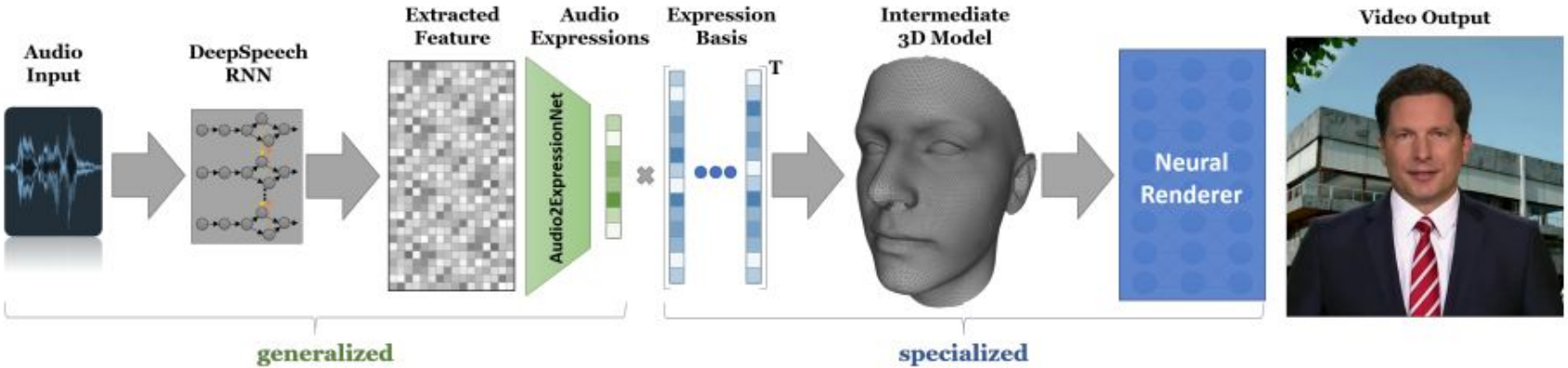


Image Source

Neural Voice Puppetry



Thies, J., Elgharib, M.A., Tewari, A., Theobalt, C., & Nießner, M. (2020). Neural Voice Puppetry: Audio-driven Facial Reenactment. *ECCV*.

Intermediate 3D Model

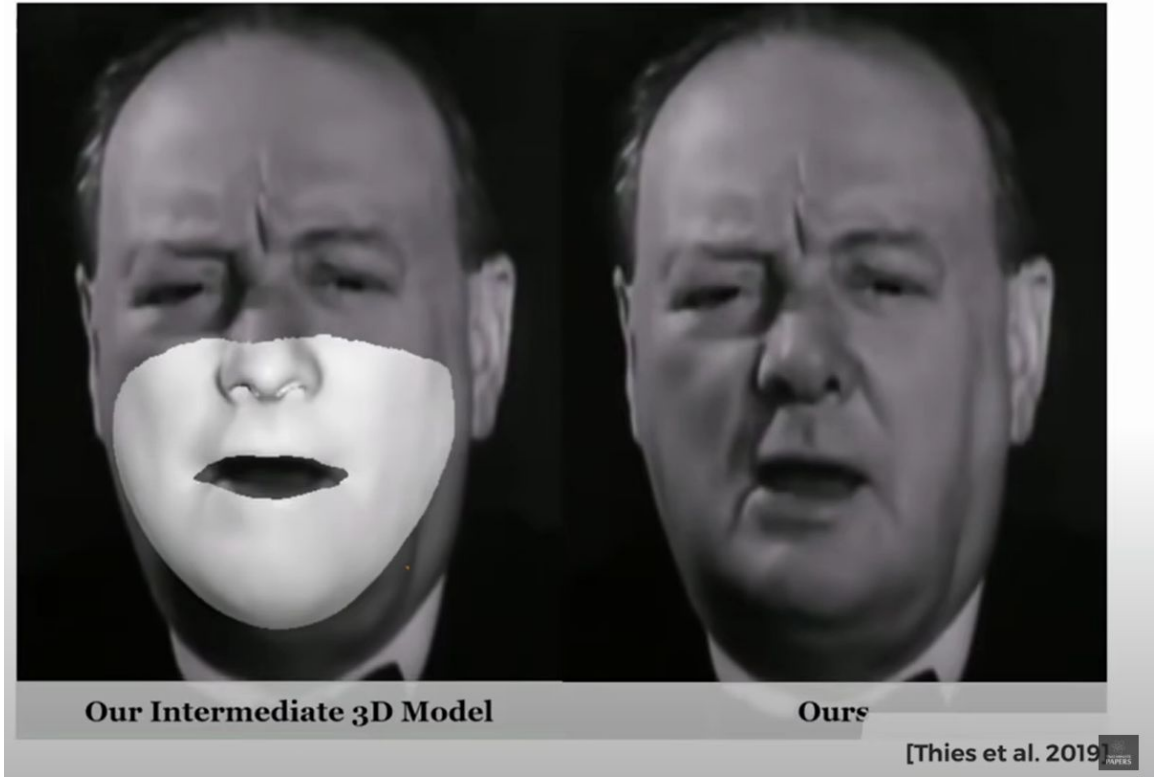
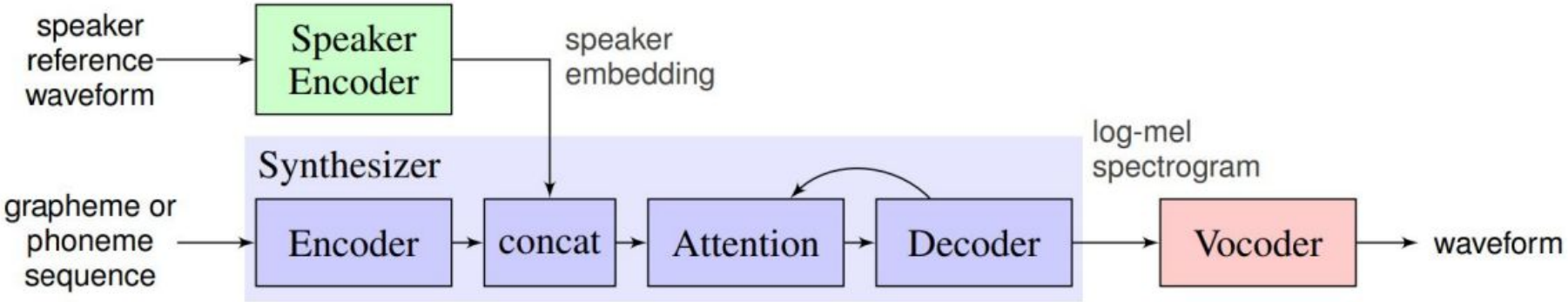


Image Source

Voice Cloning



Louppe, G. (2019). Master thesis : Automatic Multispeaker Voice Cloning.


Celebrations
Empire Footwear, you've obviously heard of them? Make sure that you buy your shoes from there.

Thank you