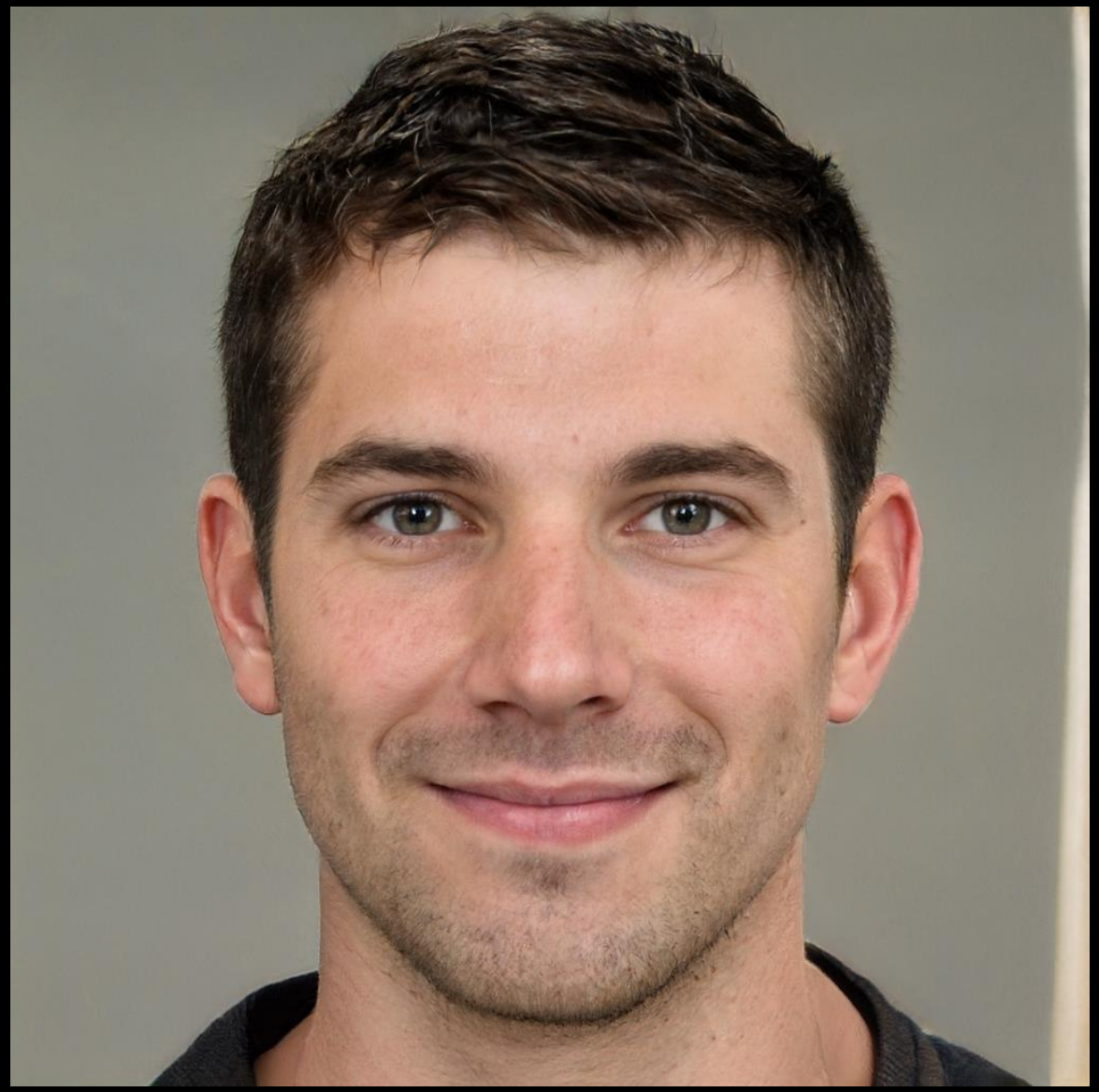
Generative Adversarial Networks (GANs)

Prof. Kuan-Ting Lai

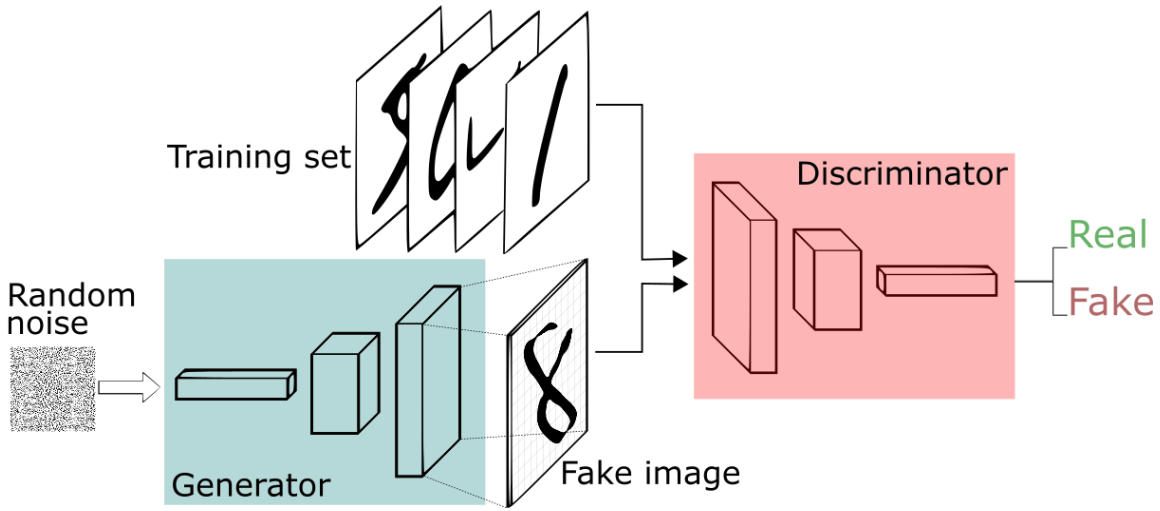
2022/6/6

DeepFake (Intro)

This Person does not Exist (thispersondoesnotexist.com)

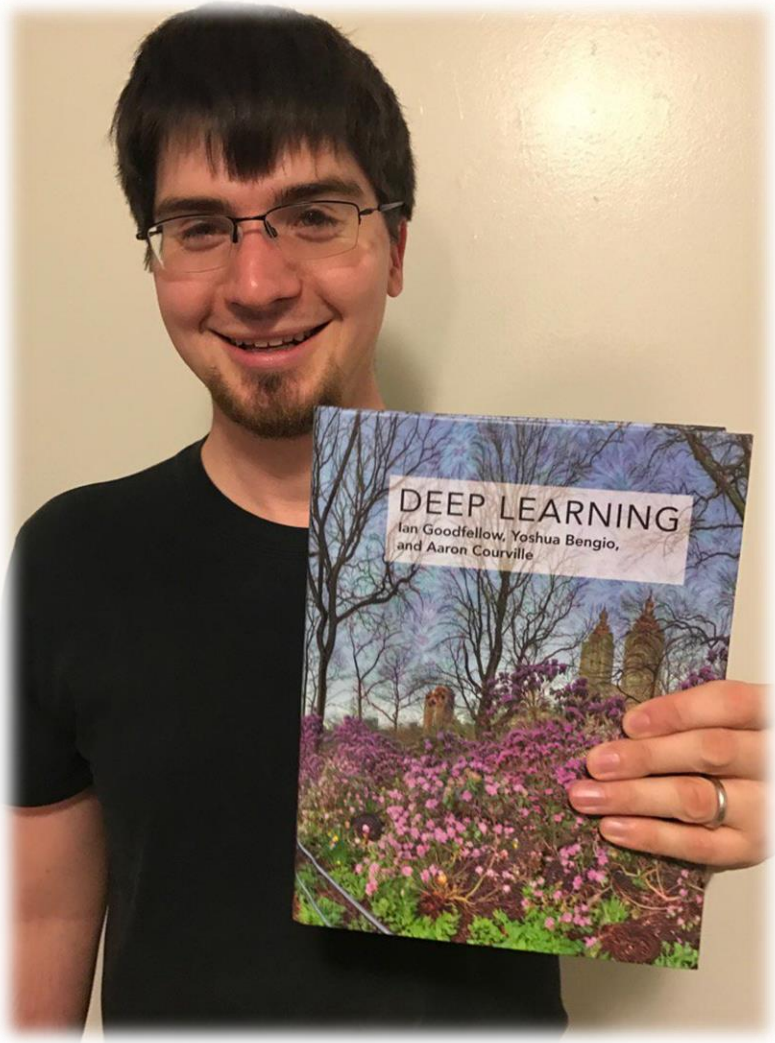


Father of GAN: Ian Goodfellow

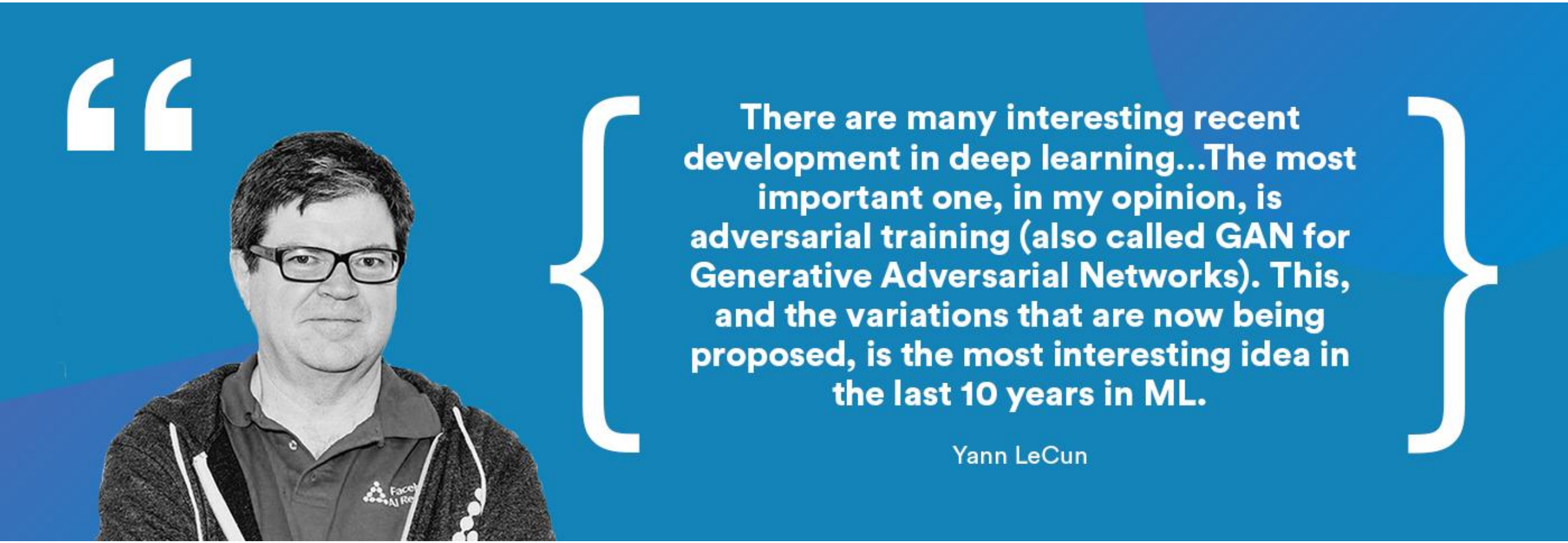


"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency.

Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles." [Goodfellow et. al.]



Generative Adversarial Networks (GAN)



- https://www.youtube.com/watch?v=9JpdAg6uMXs
- https://arxiv.org/abs/1701.00160

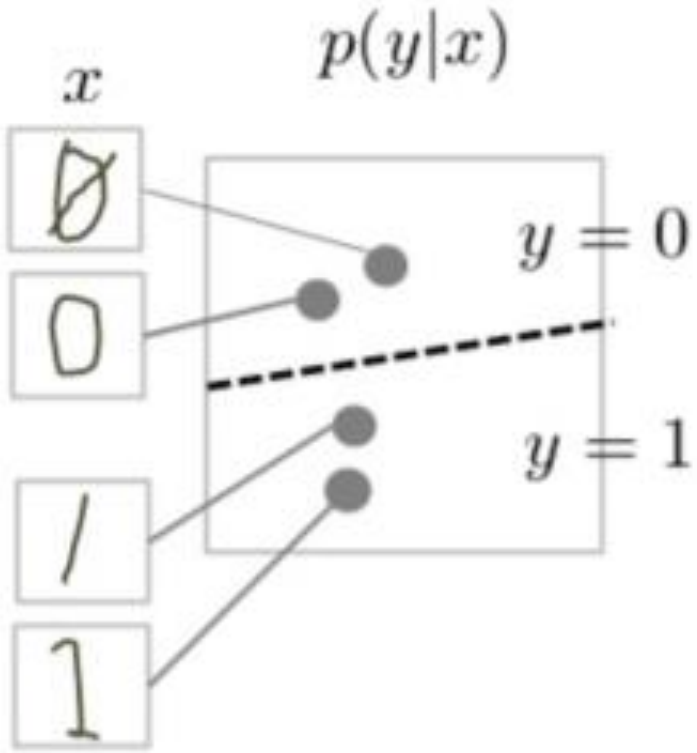
What is a Generative Model?

- Informally:
  - **Generative** models can generate new data instances.
  - **Discriminative** models discriminate between different kinds of data instances.
- Formally
  - **Generative** models capture the joint probability p(X, Y), or just p(X) if there are no labels.
  - **Discriminative** models capture the conditional probability p(Y | X).

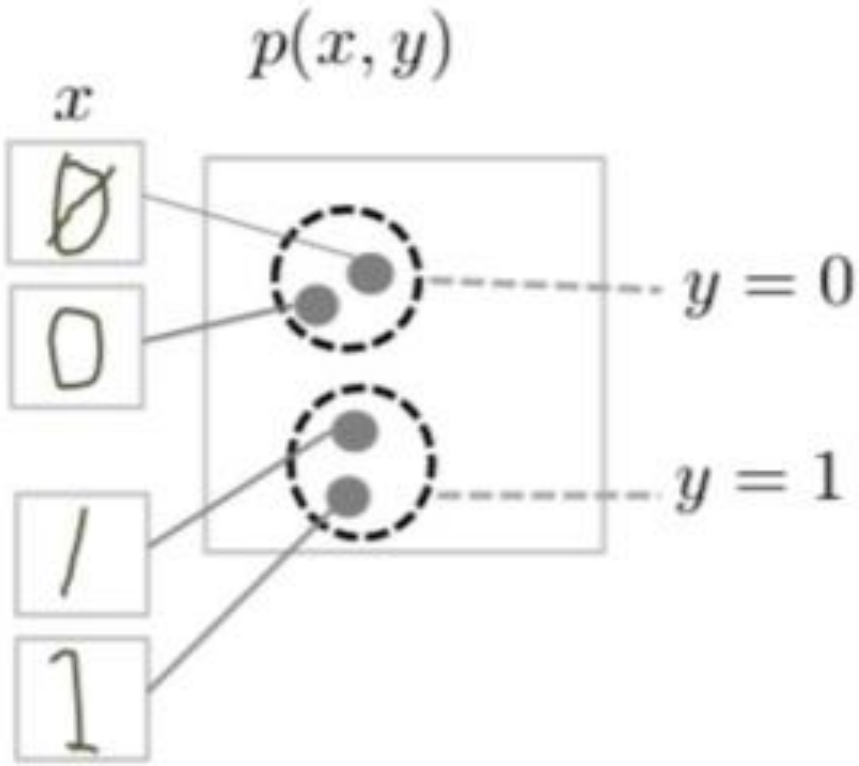
https://developers.google.com/machine-learning/gan

Training Generative Models is Hard!

- Discriminative Model



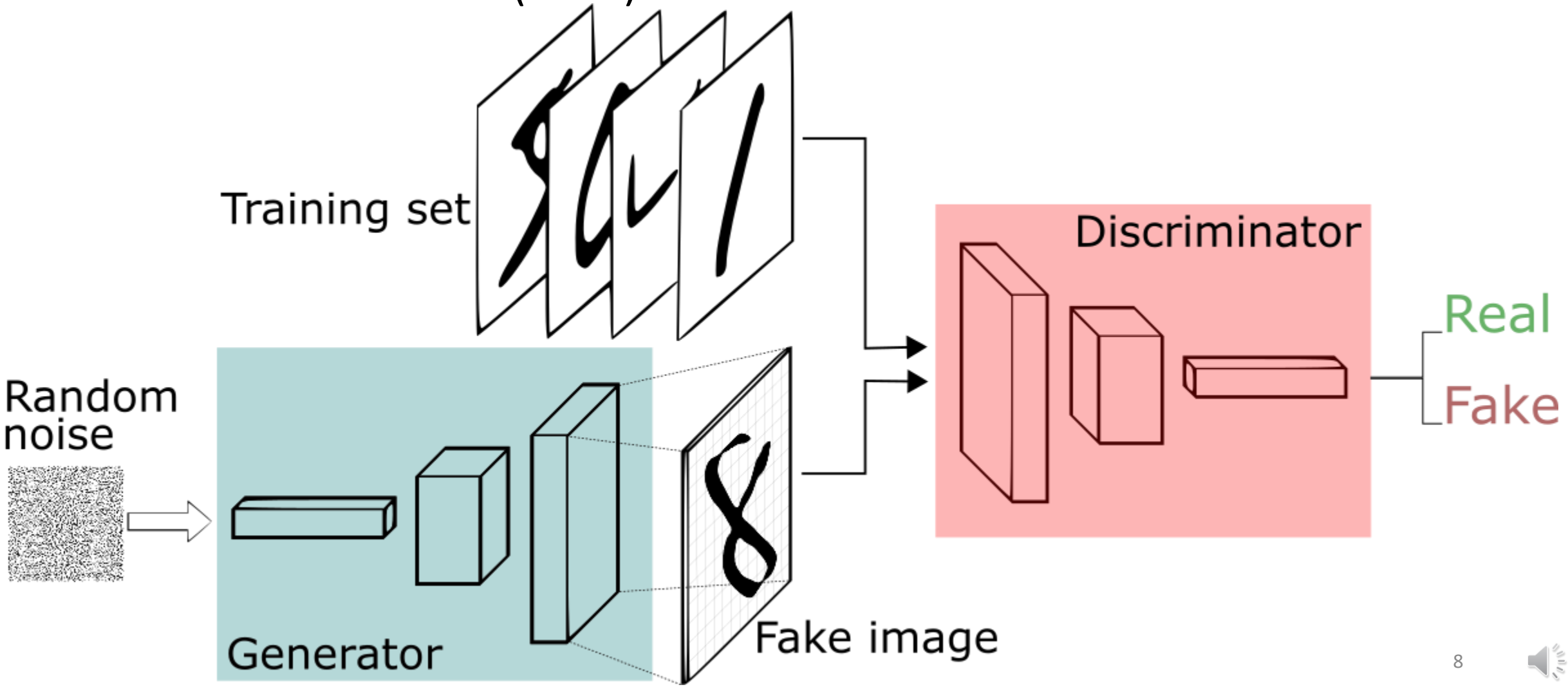
- Generative Model



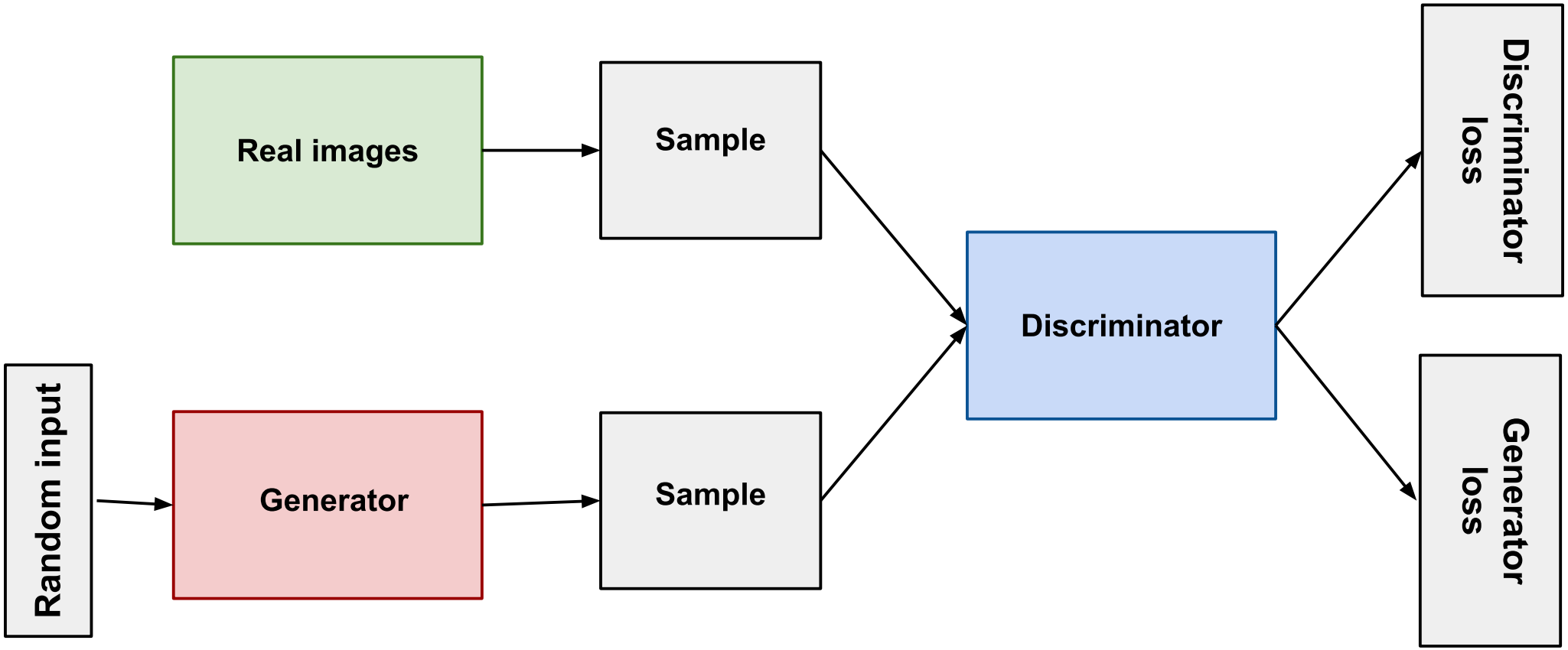
https://developers.google.com/machine-learning/gan/generative

Generative Adversarial Networks (GAN)

- Ian Goodfellow et al. (2014)



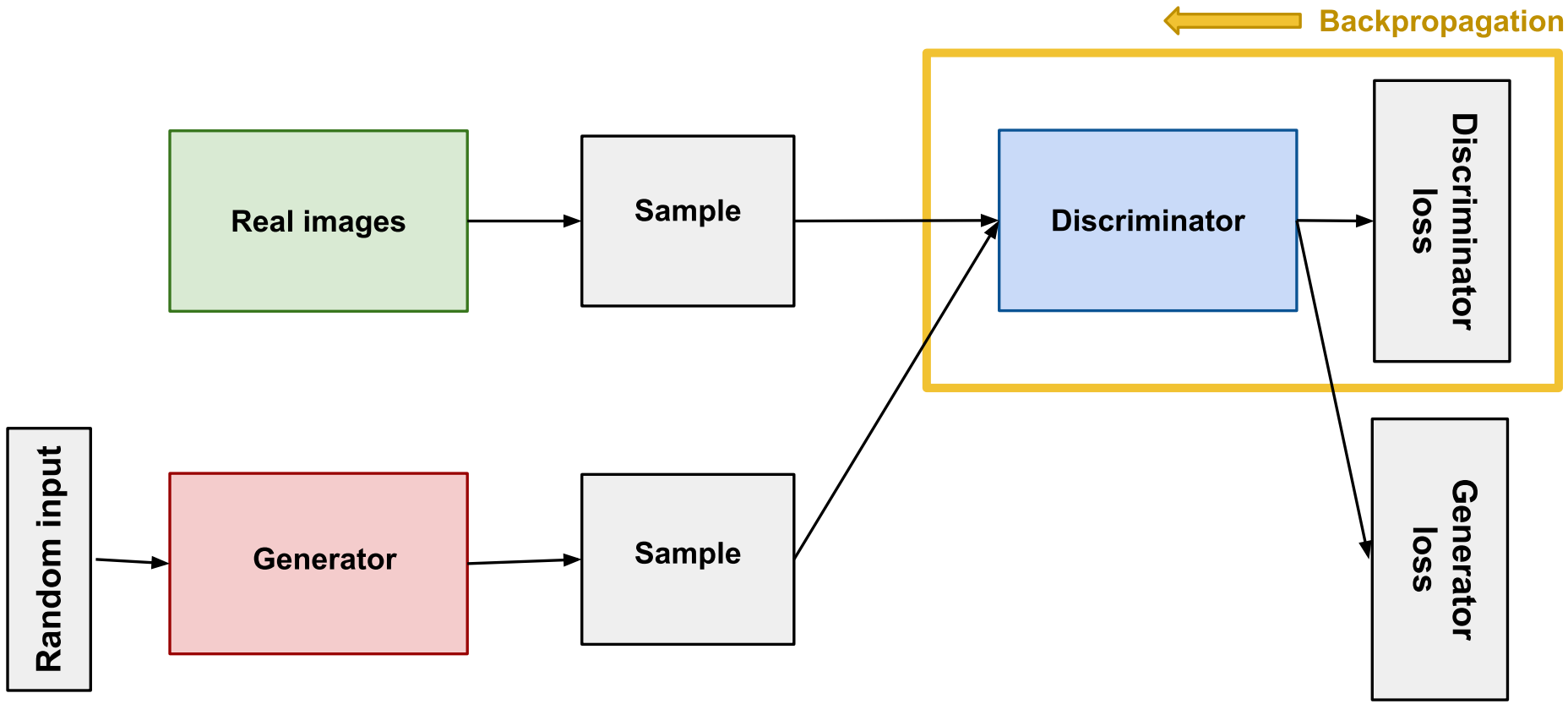
GAN Structure



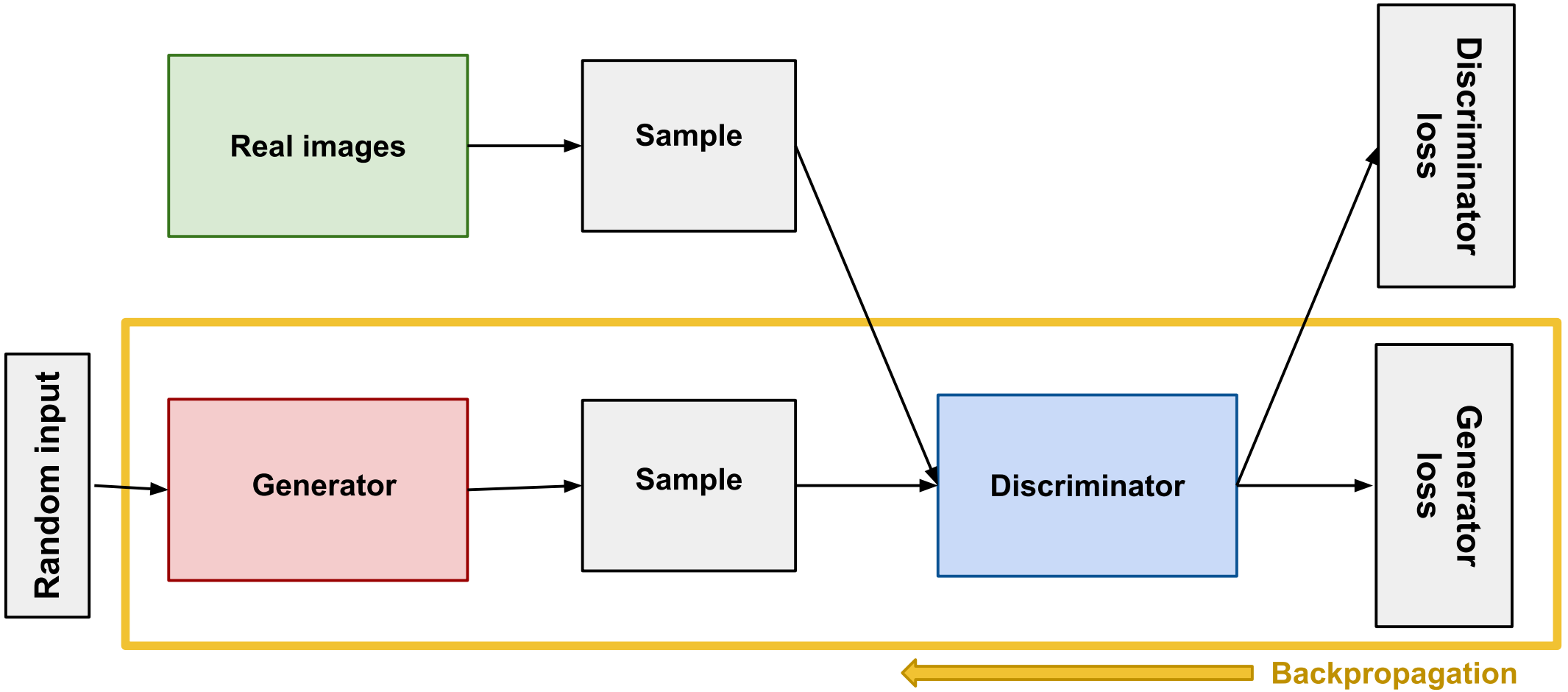
https://developers.google.com/machine-learning/gan/gan_structure

The Discriminator

- The discriminator loss penalizes the discriminator for misclassifying a real instance as fake or a fake instance as real.
- The discriminator updates its weights through backpropagation



The Generator



https://developers.google.com/machine-learning/gan/generator

The Generator Training Steps

1. Sample random noise.
2. Produce generator output from sampled random noise.
3. Get discriminator "Real" or "Fake" classification for generator output.
4. Calculate loss from discriminator classification.
5. Backpropagate through both the discriminator and generator to obtain gradients.
6. Use gradients to change only the generator weights.

GAN Training

- Alternating Training
  - Train the discriminator for one or more epochs.
  - Train the generator for one or more epochs.
  - Repeat steps 1 and 2 to continue to train the generator and discriminator networks.
- Convergence
  - The discriminator performance gets worse
  - If the generator succeeds perfectly, then the discriminator has a 50% accuracy.

https://developers.google.com/machine-learning/gan/training

GAN Loss Functions

- Minimax Loss
  - The loss function used in the paper that introduced GANs.
- Wasserstein Loss
  - The default loss function for TF-GAN Estimators. First described in a 2017 paper.

https://developers.google.com/machine-learning/gan/loss

Minimax Loss

- D(x) is the discriminator's estimate of the probability that the real instance x is real.
- E_x is the expected value over all real data instances.
- G(z) is the generator's output when given noise z.
- D(G(z)) is the discriminator's estimate of the probability that a fake instance is real.
- E_z is the expected value over all generated fake instances G(z).
- The formula derives from the cross-entropy between the real and generated distributions.

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

https://developers.google.com/machine-learning/gan/loss

Wasserstein Loss

- In Wasserstein GAN, discriminator does not actually classify instances. The output does not have to be between 0 and 1
- Discriminator training just tries to make the output larger for real instances than for fake instances.
- **Critic Loss**: D(x) - D(G(z))
  - The discriminator tries to maximize the difference between its output on real instances and fake instances.
- **Generator Loss**: D(G(z))
  - The generator tries to maximize the discriminator's output for its fake instances.
  - Use earth mover distance

https://developers.google.com/machine-learning/gan/loss

Common Problems of Training GAN Networks

- Strong Discriminator
  - if the discriminator is too good, then generator training can fail due to vanishing gradients
- Mode Collapse
  - The generator may learn to produce only one output
- Failure to Converge
  - Solution 1: adding noise to discriminator inputs
  - Solution 2: penalizing discriminator weights

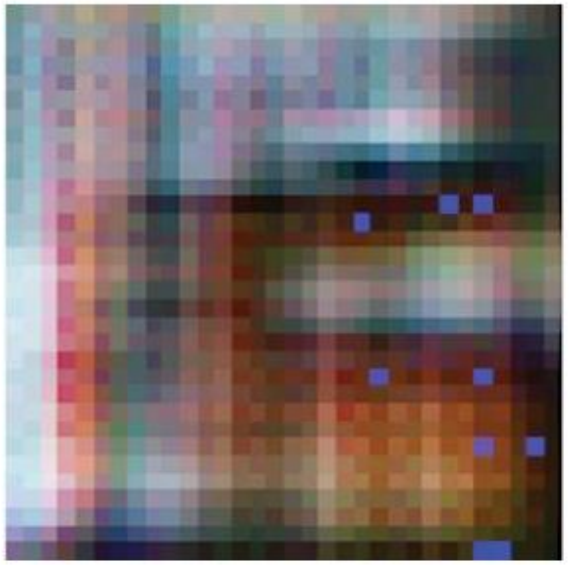
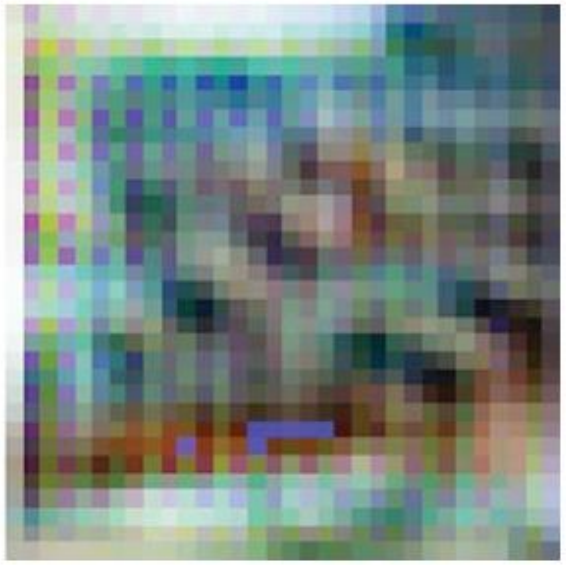
https://developers.google.com/machine-learning/gan/problems

Bag of Tricks for Training GANs

- Use tanh as the last activation in the generator, instead of sigmoid
- Sample points from the latent space using a normal distribution
- Stochasticity is good to induce robustness. Introducing randomness during training helps prevent GAN to get stuck.
  - Use dropout in the discriminator
  - Add some random noise to the labels for the discriminator.
- Sparse gradients can hinder GAN training. There are two things that can induce gradient sparsity: 1) max pooling, 2) ReLU activations.
  - Use strided convolutions for downsampling
  - Use LeakyReLU, which allows small negative activation values.
- In generated images, it is common to see "checkerboard artifacts" caused by unequal coverage of the pixel space in the generator.
  - Use a kernel size that is divisible by the stride size

Checkerboard Artifacts

- Caused by unequal coverage of the pixel space in the generator
- Solution: Use a kernel size that is divisible by the stride size in *Conv2D* and *Conv2DTranspose*

Training GAN for Celebrity Faces

- Code: https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/chapter12_part05_gans.ipynb
  Dataset: CelebA Dataset (cuhk.edu.hk)

Loading Images from Directory

- Unzip the dataset into "celeba_gan" folder

```
from tensorflow import keras
dataset = keras.utils.dataset_from_directory(
    "celeba_gan",
    label_mode=None,
    image_size=(64, 64),
    batch_size=32,
    smart_resize=True)
```

Only the images will be returned—no labels.

We will resize the images to 64 × 64 by using a smart combination of cropping and resizing to preserve aspect ratio. We don't want face proportions to get distorted!

```
dataset = dataset.map(lambda x: x / 255.)
```

Generator

```
latent_dim = 128

generator = keras.Sequential(
    [
        keras.Input(shape=(latent_dim,)),
        layers.Dense(8 * 8 * 128),
        layers.Reshape((8, 8, 128)),
        layers.Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(256, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(512, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(3, kernel_size=5, padding="same", activation="sigmoid"),
    ],
    name="generator",
)
```

The latent space will be made of 128-dimensional vectors.

Revert the Flatten layer of the encoder.

Revert the Conv2D layers of the encoder.

We use LeakyReLU as our activation.

Discriminator

```
from tensorflow.keras import layers
discriminator = keras.Sequential(
    [
        keras.Input(shape=(64, 64, 3)),
        layers.Conv2D(64, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Flatten(),
        layers.Dropout(0.2),   # Important Trick!
        layers.Dense(1, activation="sigmoid"),
    ],
    name="discriminator",
)
```

GAN Training Loop

1. Draw random points in the latent space (random noise).
2. Generate images with generator using this random noise.
3. Mix the generated images with real ones.
4. Train discriminator using these mixed images including real and fake (generated) images.
5. Fix the discriminator’s weights
6. Draw new random points in the latent space.
7. Train the generator to fool the discriminator

The GAN Model

```
import tensorflow as tf
class GAN(keras.Model):
    def __init__(self, discriminator, generator, latent_dim):
        super().__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim
        self.d_loss_metric = keras.metrics.Mean(name="d_loss")
        self.g_loss_metric = keras.metrics.Mean(name="g_loss")

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    @property
    def metrics(self):
        return [self.d_loss_metric, self.g_loss_metric]
```

Sets up metrics to track the two losses over each training epoch

train_step() of the GAN Model (2-1)

```
def train_step(self, real_images):
    batch_size = tf.shape(real_images)[0]
    random_latent_vectors = tf.random.normal(
        shape=(batch_size, self.latent_dim))
    generated_images = self.generator(random_latent_vectors)
    combined_images = tf.concat([generated_images, real_images], axis=0)
    labels = tf.concat(
        [tf.ones((batch_size, 1)), tf.zeros((batch_size, 1))],
        axis=0
    )
    labels += 0.05 * tf.random.uniform(tf.shape(labels))

    with tf.GradientTape() as tape:
        predictions = self.discriminator(combined_images)
        d_loss = self.loss_fn(labels, predictions)
    grads = tape.gradient(d_loss, self.discriminator.trainable_weights)
    self.d_optimizer.apply_gradients(
        zip(grads, self.discriminator.trainable_weights)
    )
```

Decodes them to fake images

Combines them with real images

Samples random points in the latent space

Assembles labels, discriminating real from fake images

Adds random noise to the labels—an important trick!

Trains the discriminator

train_step() of the GAN Model (2-2)

Samples random points in the latent space

```
random_latent_vectors = tf.random.normal(
    shape=(batch_size, self.latent_dim))

misleading_labels = tf.zeros((batch_size, 1))
```

Assembles labels that say "these are all real images" (it's a lie!)

Trains the generator

```
with tf.GradientTape() as tape:
    predictions = self.discriminator(
        self.generator(random_latent_vectors))
    g_loss = self.loss_fn(misleading_labels, predictions)
grads = tape.gradient(g_loss, self.generator.trainable_weights)
self.g_optimizer.apply_gradients(
    zip(grads, self.generator.trainable_weights))

self.d_loss_metric.update_state(d_loss)
self.g_loss_metric.update_state(g_loss)
return {"d_loss": self.d_loss_metric.result(),
        "g_loss": self.g_loss_metric.result()}
```

Monitor the Training

```
class GANMonitor(keras.callbacks.Callback):
    def __init__(self, num_img=3, latent_dim=128):
        self.num_img = num_img
        self.latent_dim = latent_dim

    def on_epoch_end(self, epoch, logs=None):
        random_latent_vectors = tf.random.normal(shape=(self.num_img,
                                                        self.latent_dim))
        generated_images = self.model.generator(random_latent_vectors)
        generated_images *= 255
        generated_images.numpy()
        for i in range(self.num_img):
            img = keras.utils.array_to_img(generated_images[i])
            img.save(f"generated_img_{epoch:03d}_{i}.png")
```

Compiling and Training GAN

```
epochs = 100

gan = GAN(discriminator=discriminator, generator=generator,
          latent_dim=latent_dim)
gan.compile(
    d_optimizer=keras.optimizers.Adam(learning_rate=0.0001),
    g_optimizer=keras.optimizers.Adam(learning_rate=0.0001),
    loss_fn=keras.losses.BinaryCrossentropy(),
)

gan.fit(
    dataset, epochs=epochs, callbacks=[GANMonitor(num_img=10,
          latent_dim=latent_dim)]
)
```

Training Results

Epoch 0

Epoch 1

Epoch 9

Epoch 14

Epoch 21

Epoch 26

Epoch 27

Epoch 28

Interesting GAN Applications

Deepfake Round Table

https://www.creativebloq.com/features/deepfake-examples

This Person (and the Creepy Person) do not Exist!

thispersondoesnotexist.com

Evolution of GAN Face Generation

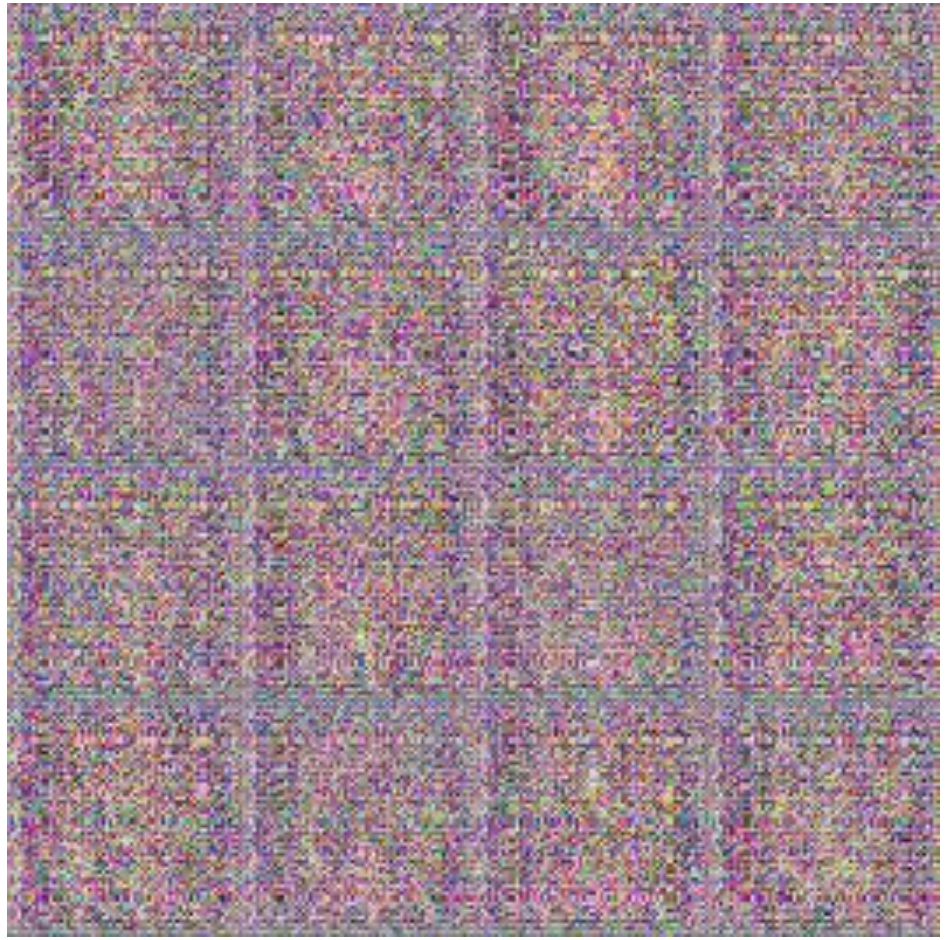
https://spectra.mathpix.com/article/2021.09.00009/gans

How to Detect Generated Faces?

- Detect fake faces by central positioning the eyes
- Stanford University researchers identify fake LinkedIn profiles using eye locations

https://www.capradio.org/news/npr/story?storyid=1088140809

Anime GAN (https://make.girls.moe/)



https://arxiv.org/abs/1708.05509

https://heartbeat.comet.ml/my-mangagan-building-my-first-generative-adversarial-network-2ec1920257e3

https://make.girls.moe/

MakeGirlsMoe | Home | History | Transition | Help | English | Twitter | Discord

Generate

+1 -1

Share on Twitter

Options

Advanced Mode

Model

Camellia 256x256 Ver.171219 (9.9MB)

Hair Color: Blonde

Hair Style: Random

Eye Color: Green

Dark Skin: Off | Random | On

Blush: Off | Random | On

Smile: Off | Random | On

Open Mouth: Off | Random | On

Hat: Off | Random | On

Ribbon: Off | Random | On

Glasses: Off | Random | On

Style: Random

Noise: Random | Fixed

Current Noise

Noise Import/Export: Import | Export

Operations: Import | Export | Reset

WebGL Acceleration: Disabled | Enabled

Current Backend

Super Resolution GAN

- Ledig et al., “Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network,” 2017 (https://arxiv.org/pdf/1609.04802.pdf)

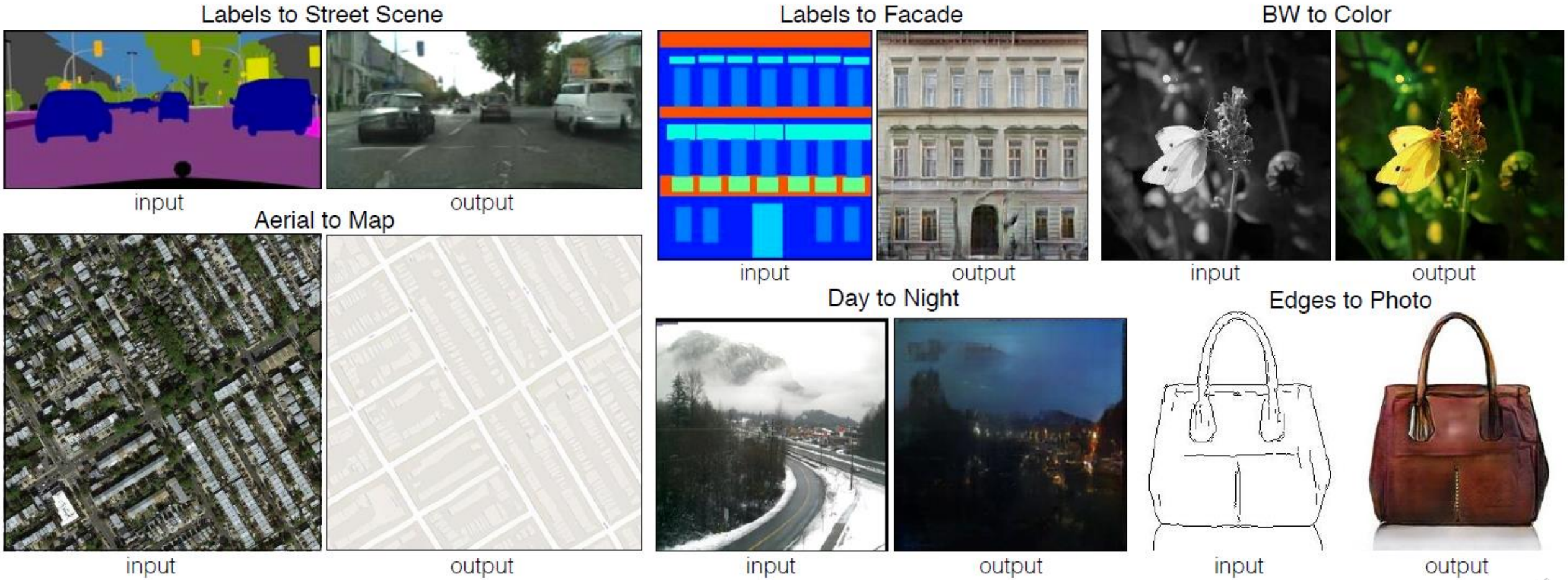
More GAN Models

- TensorFlow Tutorial / Generative

1. Pixel-2-Pixel
2. CycleGAN
3. Adversarial FGSM

Pix2Pix

- Phillip Isola et al., [Image-to-Image Translation with Conditional Adversarial Networks](), 2018



Training Conditional GAN

- Both the generator and discriminator observe the input edge map
- Use U-Net and PatchGAN discriminator

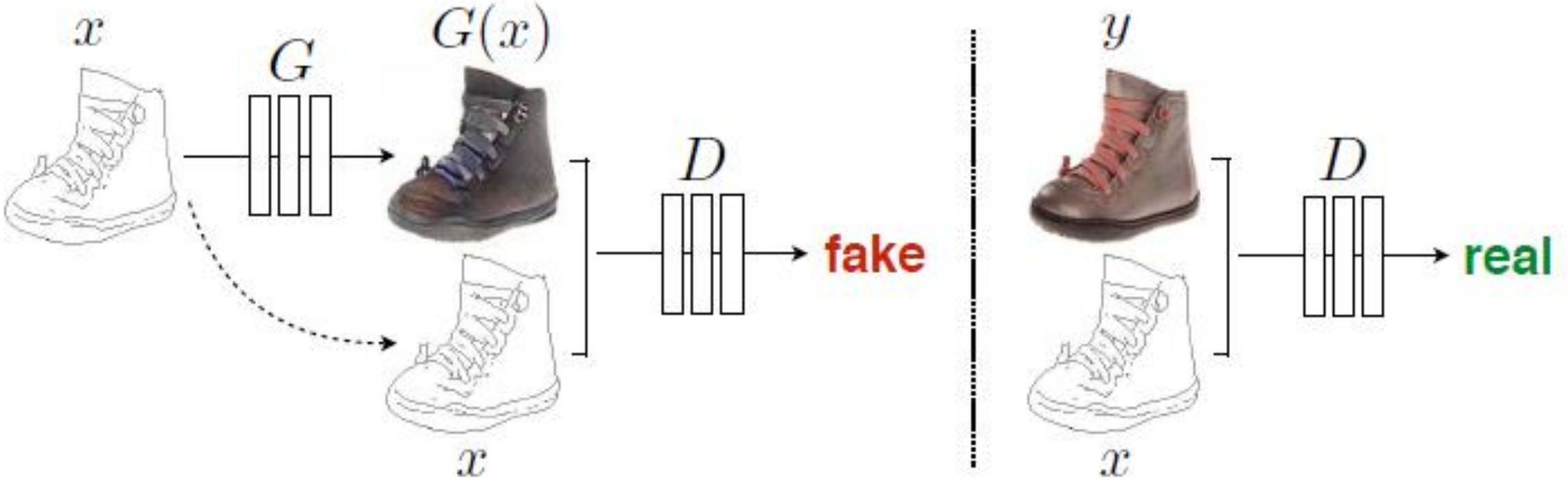
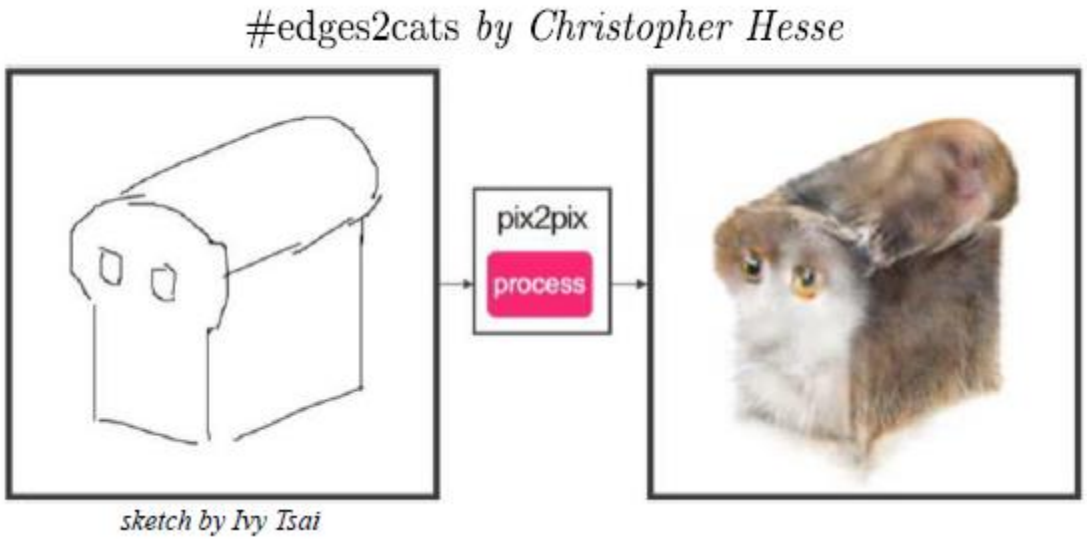
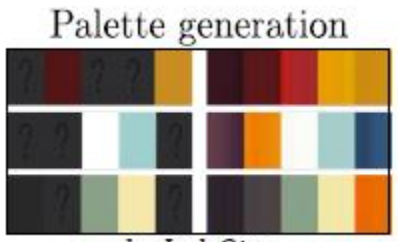
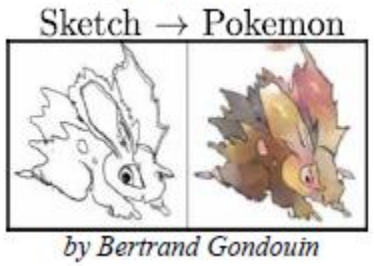


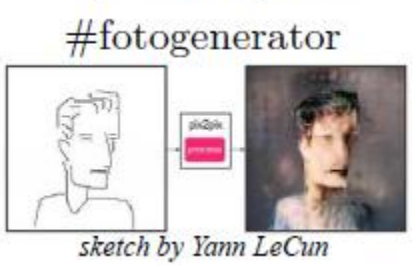
Image-to-Image Demo

- https://affinelayer.com/pixsrv/

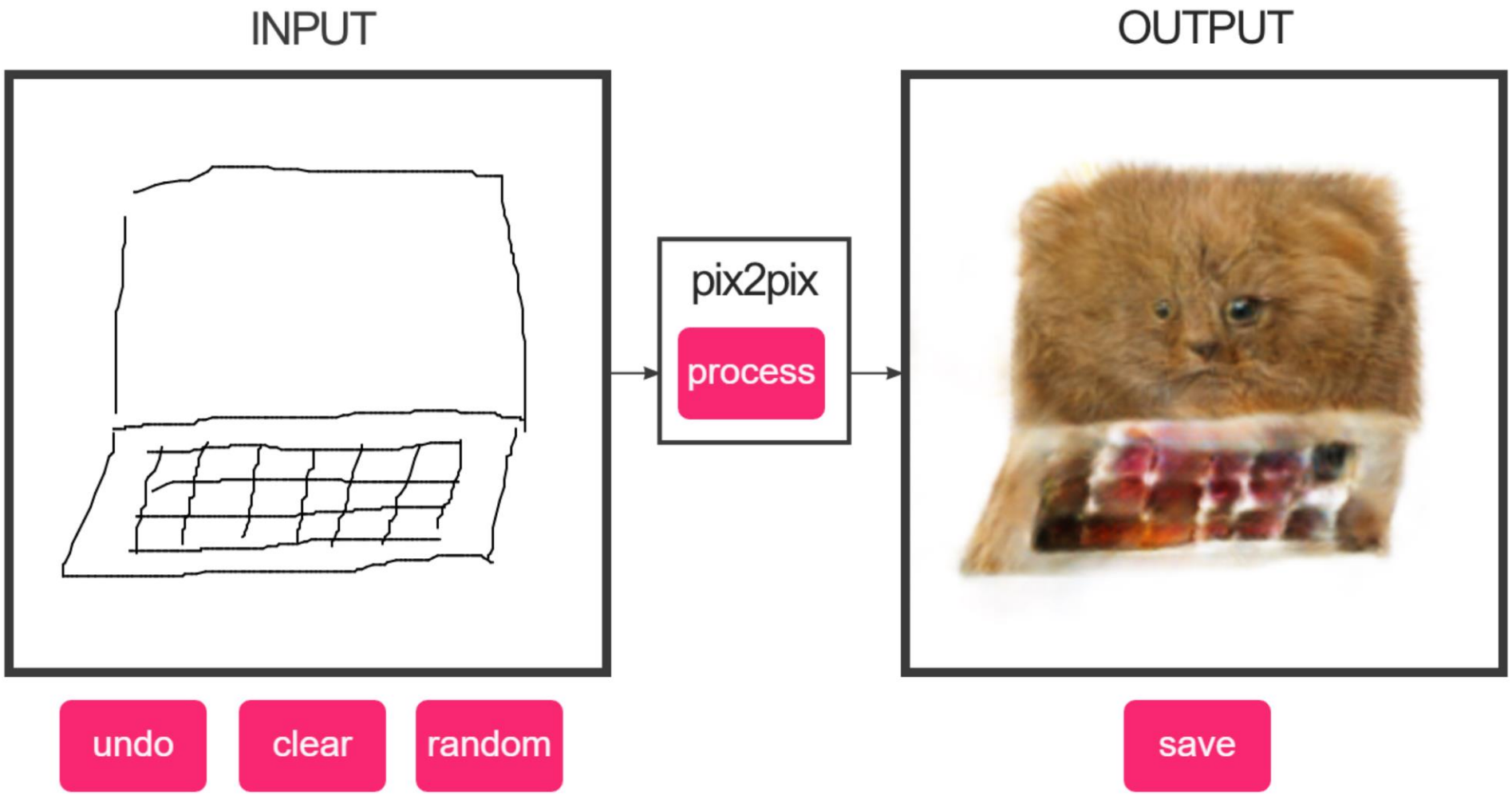






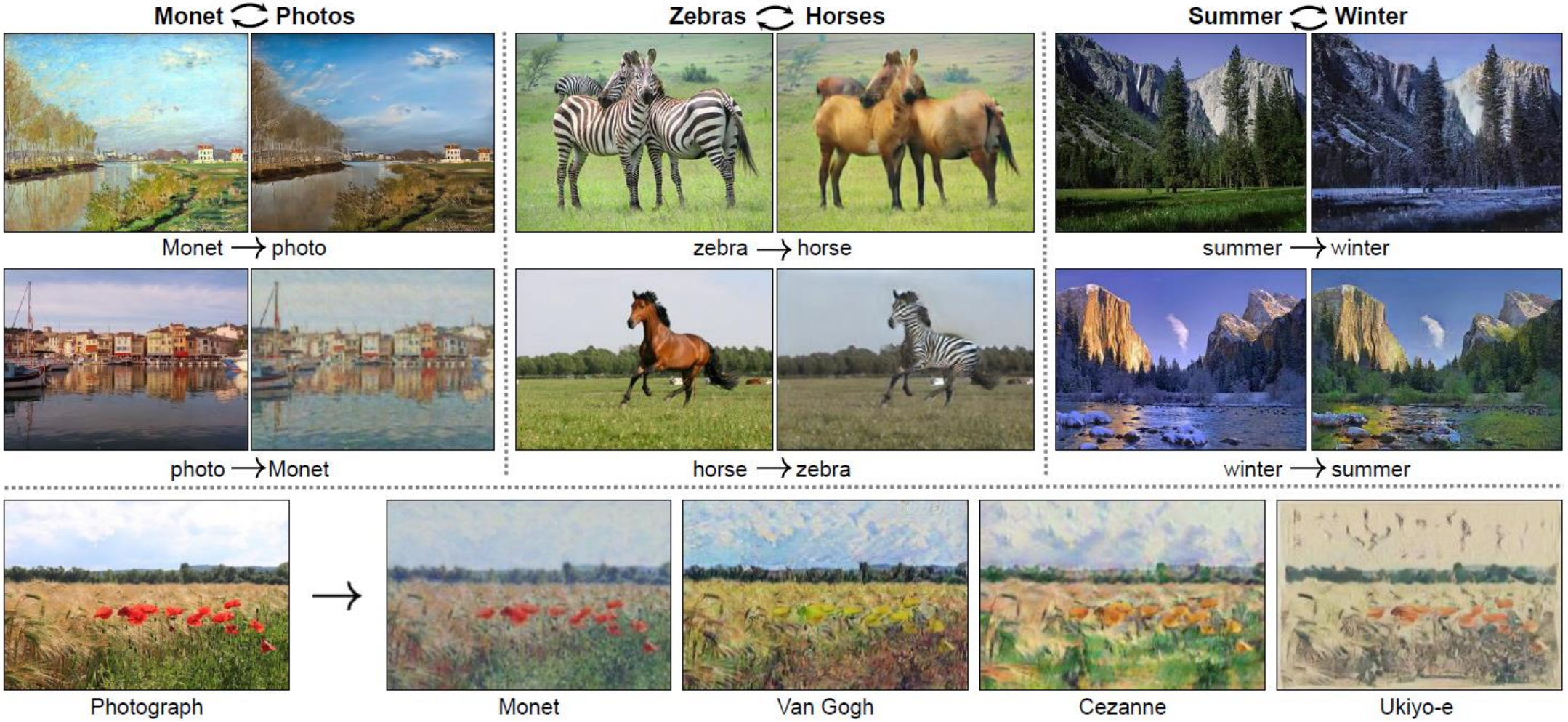


My Work (Cat Computer)



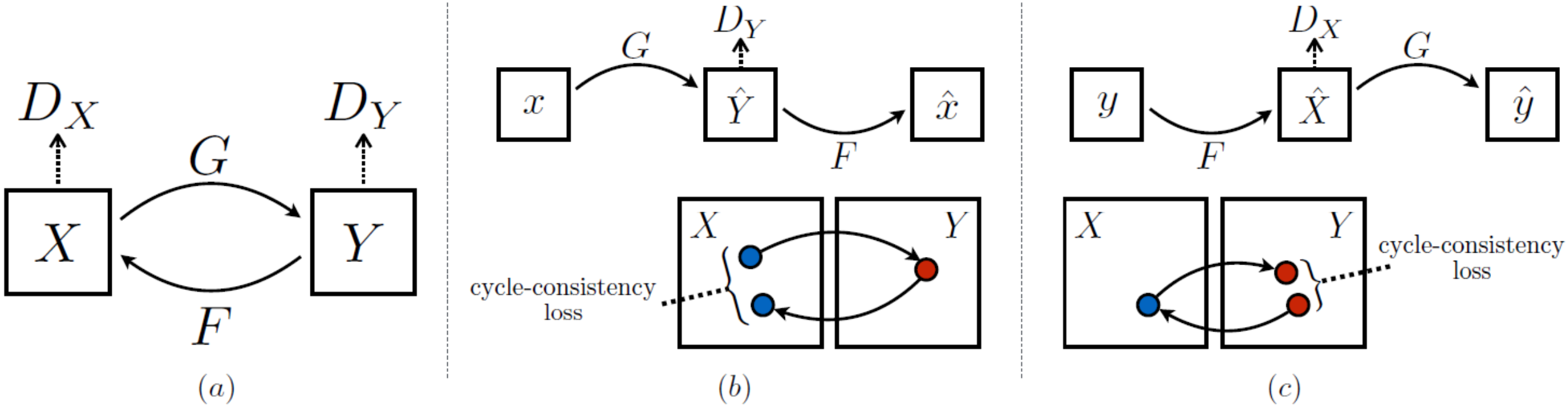
CycleGAN

- Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2018
- Learn to automatically “translate” an image from one into the other and vice versa



Model of CycleGAN

- Two mapping functions G : X → Y and F : Y → X
- Two cycle consistency losses:
  - Forward cycle-consistency loss: x → G(x) → F(G(x)) ≈ x
  - Backward cycle-consistency loss: y → F(y) → G(F(y)) ≈ y



TensorFlow CycleGAN Results

Input Image

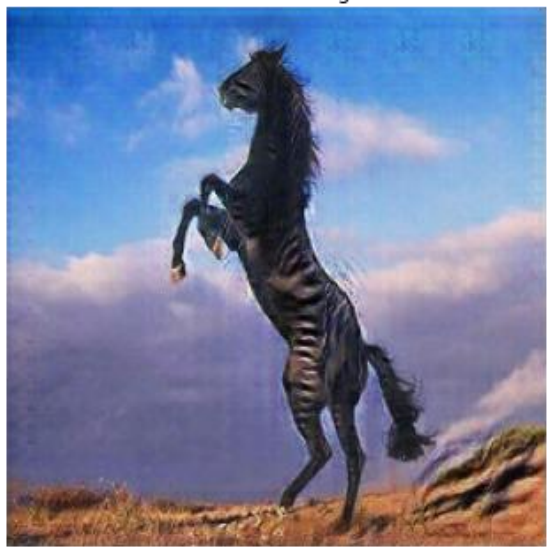
Predicted Image

Input Image

Predicted Image

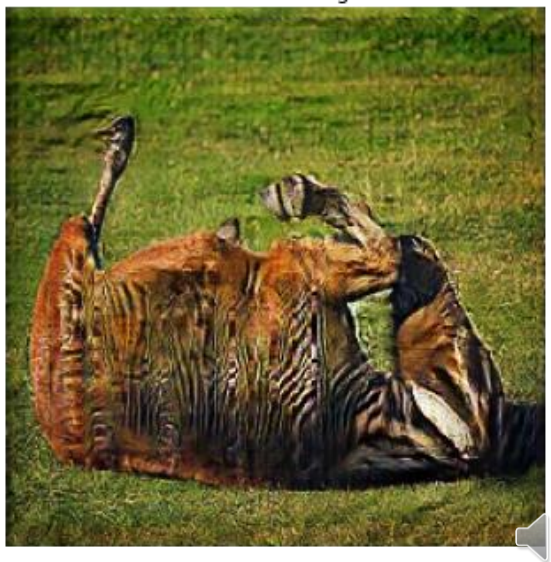
Input Image

Predicted Image



Input Image

Predicted Image



Segmentation | Sketch | Image | All

Stroke Shape

Fuzzy th: 20

Visualize | Visualize | Visualize

Stroke Size: 10

Enabled | Enabled | Enabled

Blending:

Enabled

ocean

GauGAN (NVIDIA)

ext-to-Image

NVIDIA AI Playground

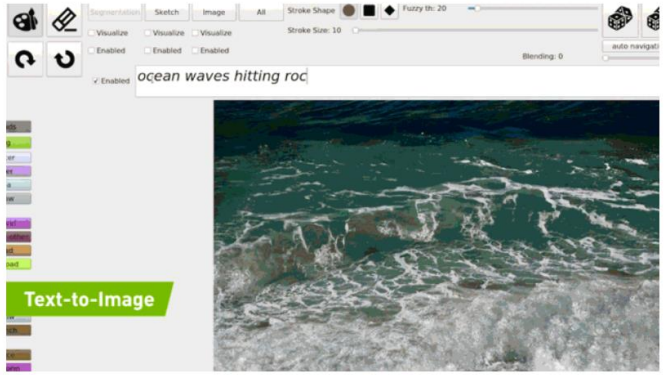
RESEARCH RESEARCH AREAS PUBLICATIONS AI PLAYGROUND TEAM CAREERS LICENSING

AI PLAYGROUND WHAT'S NEW COLLECTIONS SUBSCRIBE

WELCOME TO THE

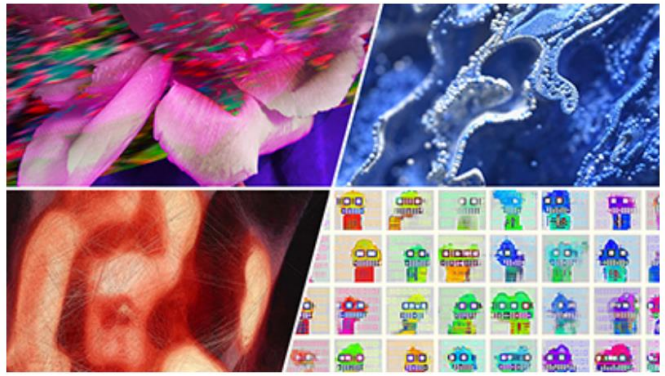
AI PLAYGROUND

The Intersection of AI, Art, and Science.



AI Demo

Paint Me a Picture: GauGAN2 AI Art Tool



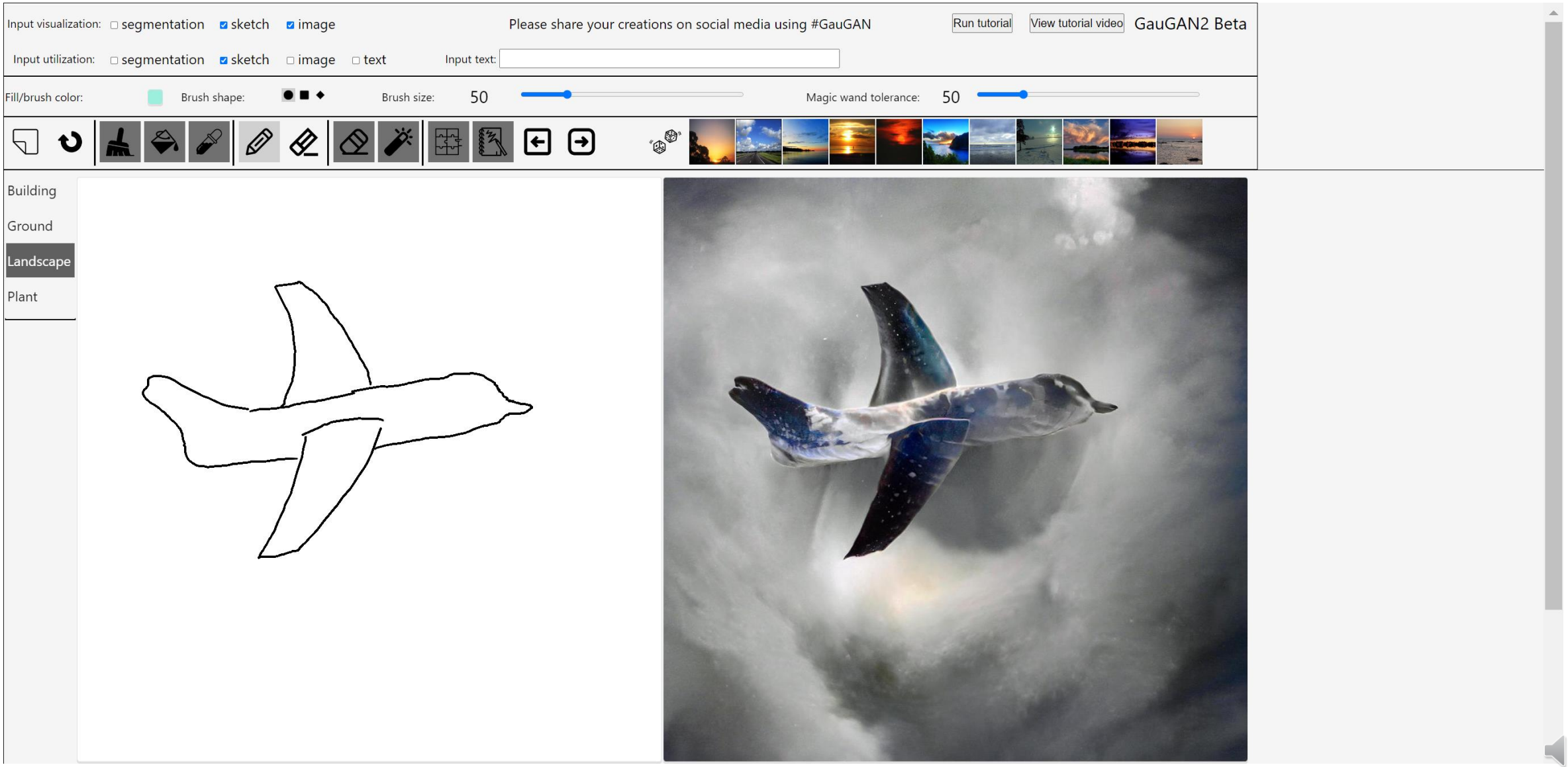
AI in Art

See AI Art in New Dimensions with Fresh Work from 4 Artists

AI Demo

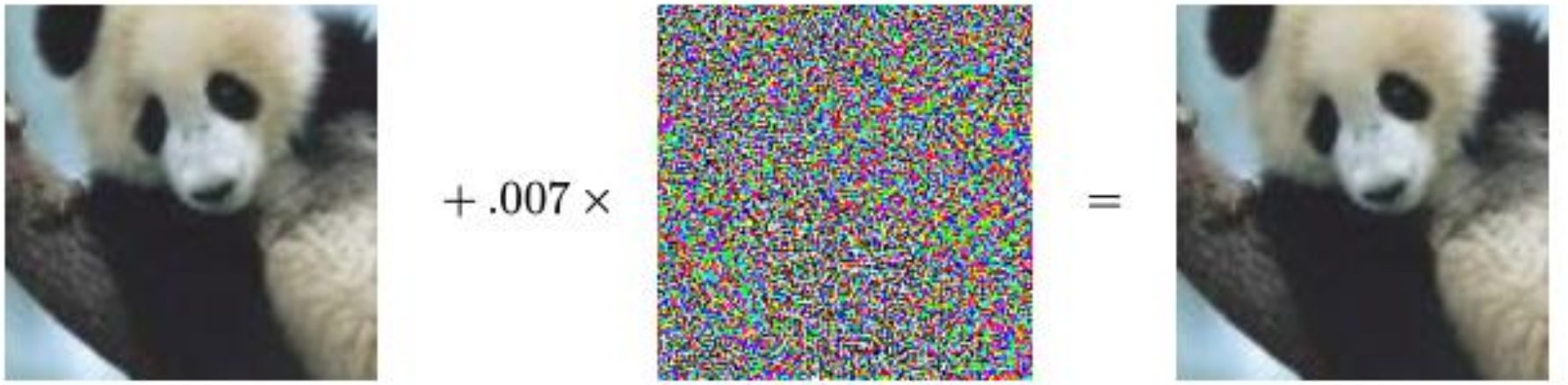
Take Conversations Global with World-Class Speech AI

My Work by GauGAN 2



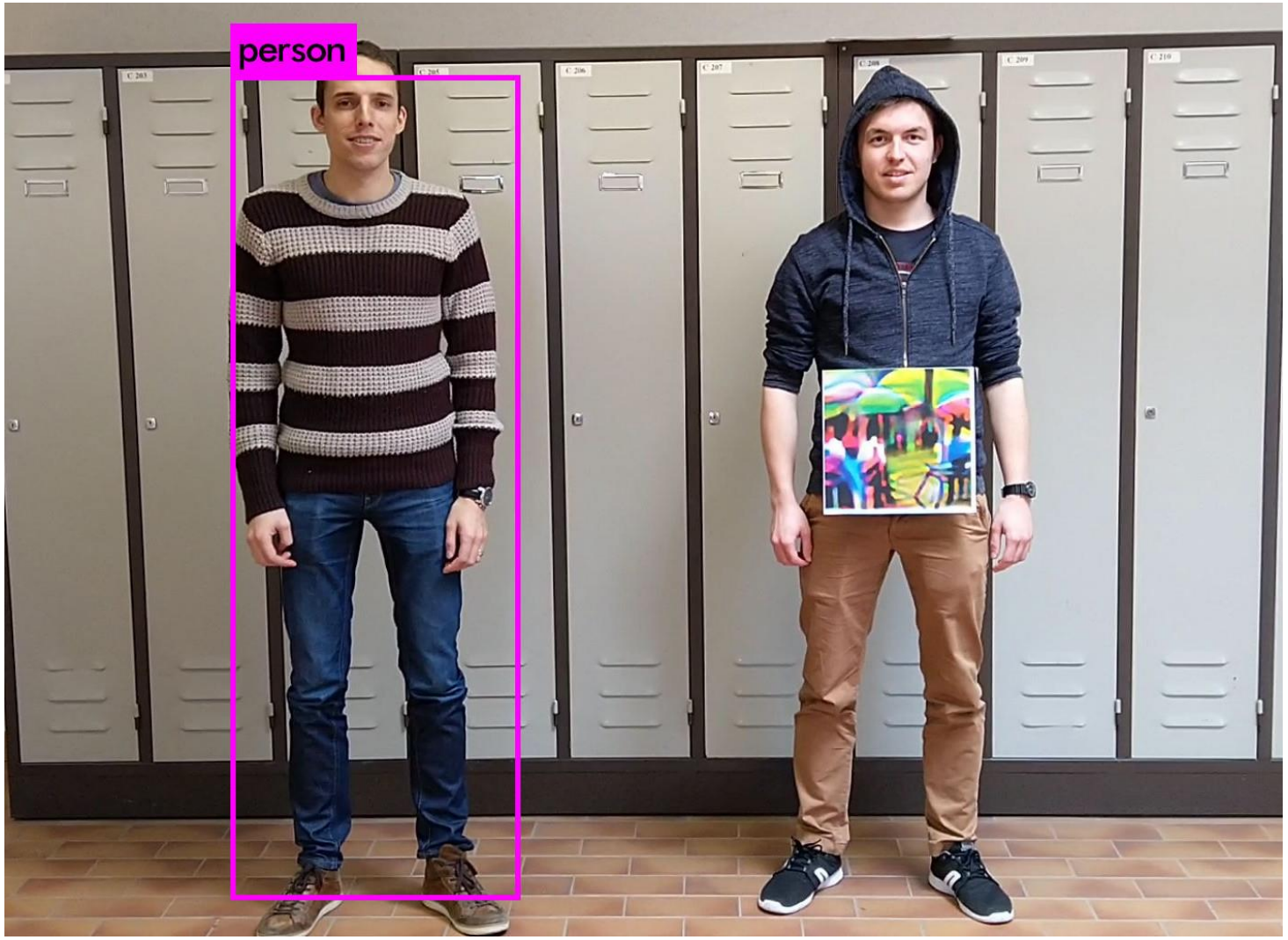
Adversarial Attack

- Goodfellow *et al*., Explaining and Harnessing Adversarial Examples, 2015
- Fast Gradient Signed Method (FGSM)



https://www.tensorflow.org/tutorials/generative/adversarial_fgsm

Fooling AI Surveillance Cameras



https://www.arxiv-vanity.com/papers/1904.08653/

Key Takeaways

- Generative Adversarial Networks has two sub-networks: Generator and Discriminator. They compete with each other.
- Training is done if the Discriminator fails to detect fake image. (ACC=0.5)
- GANs are hard to train. Introducing randomness during training is important.
- Max pooling and ReLu introduces sparsity and hinder GAN training
- Conditional GANs can accept conditional images as inputs than generate target outputs
- CycleGAN can translate image from one domain to the other and vice versa.

References

- https://www.tensorflow.org/tutorials/generative/
- https://developers.google.com/machine-learning/gan
- https://make.girls.moe/
- https://www.creativebloq.com/features/deepfake-examples
- https://www.nvidia.com/en-us/research/ai-playground/
- https://www.projectpro.io/article/generative-adversarial-networks-gan-based-projects-to-work-on/530
- https://jonathan-hui.medium.com/gan-some-cool-applications-of-gans-4c9ecca35900
- https://affinelayer.com/pixsrv/