Image to Image Translation

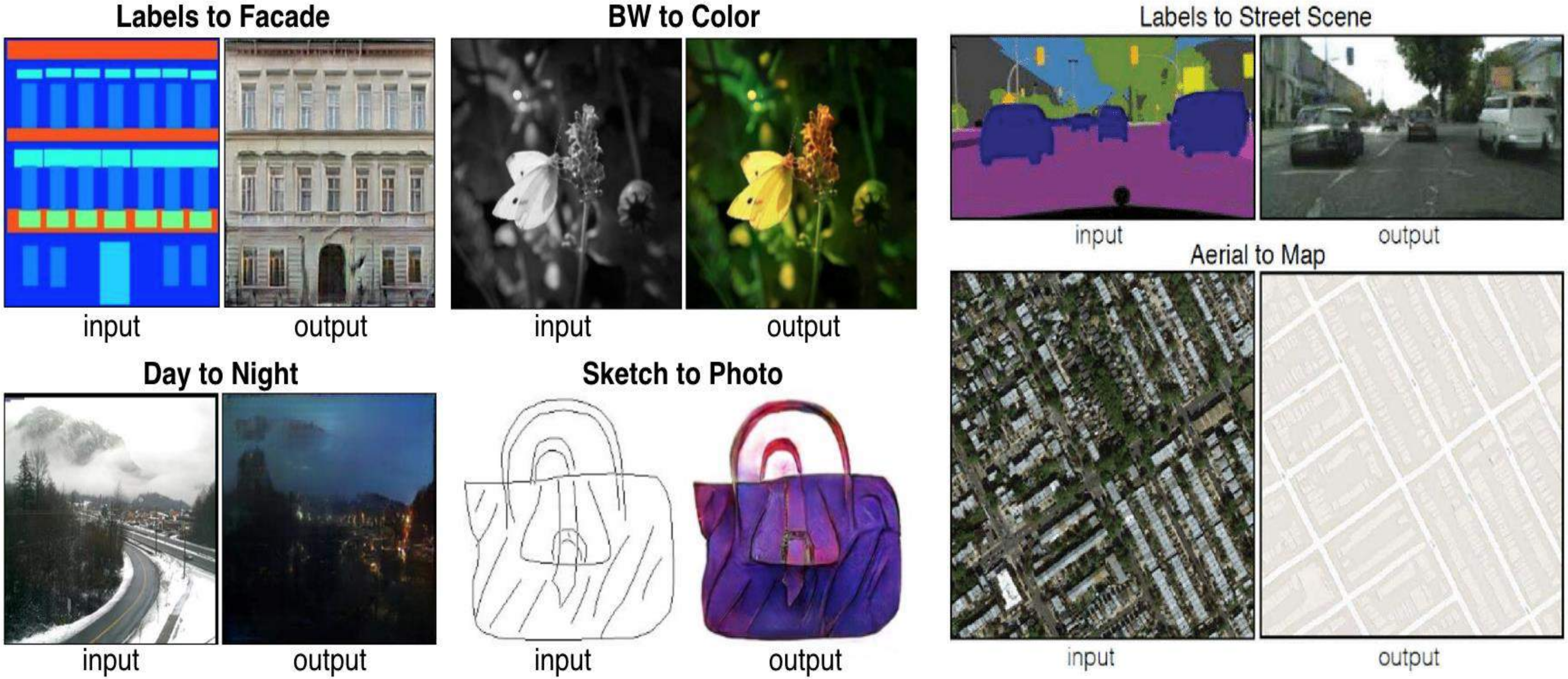
THE ORIGINAL IS UNFAITHFUL TO THE TRANSLATION.

What is Image to Image Translation?

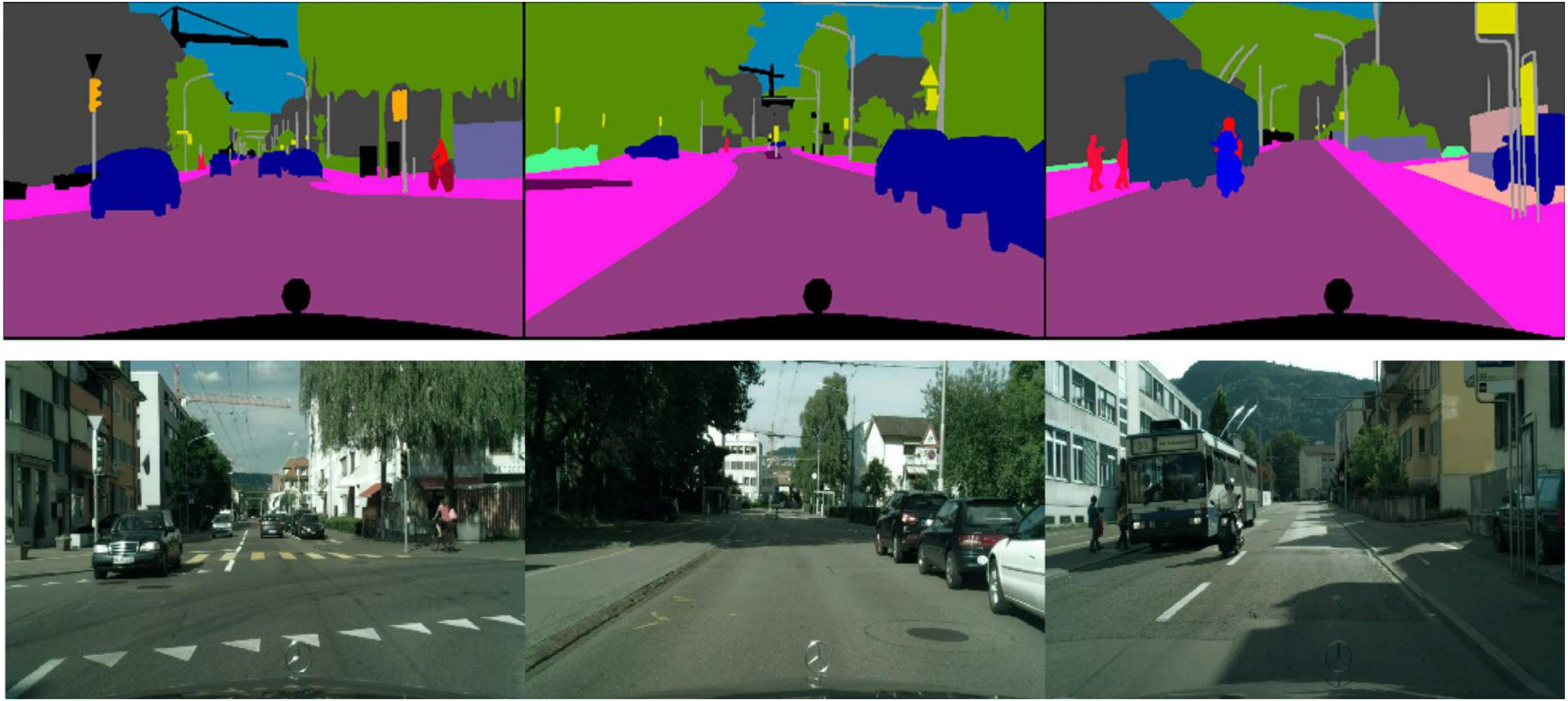
- **Image-to-Image Translation** is a task in computer vision and machine learning where the goal is to learn a mapping between an input image and an output image, such that the output image can be used to perform a specific task, such as style transfer, data augmentation, or image restoration.

- It is a class of vision and graphics problems where the goal is to learn the mapping between an input image and an output image. It can be applied to a wide range of applications, such as collection style transfer, object transfiguration, season transfer and photo enhancement.

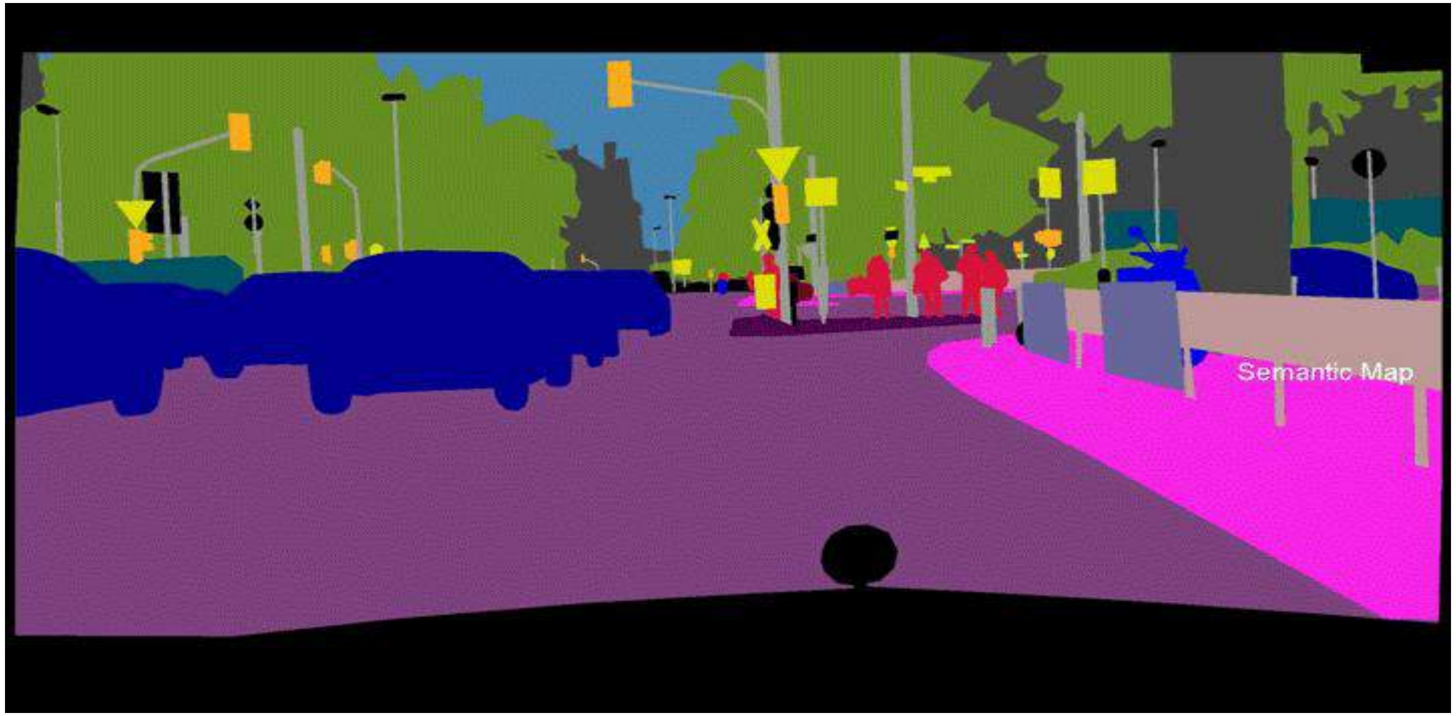
Different tasks in image translation



Our Project: Semantic to street scene translation



Semantic drawing to street scene translation



- This street scape is an artistic representation, such as painting, drawing, print or photograph, of the physical aspects of a city or urban area.
- We need to translate these street scape drawing into real street scene using various deep learning and machine learning techniques.
- This data is obtained from cityscapes dataset is intended for assessing the performance of vision algorithms for major tasks of semantic urban scene understanding.

Libraries Used

❑ Tensor Flow

- **TensorFlow is an opensource framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads.**
- **It's designed to streamline the process of developing and executing advanced analytics applications.**
- **TensorFlow applications can run on either conventional CPUs or higher-performance graphics processing units (GPUs), as well as Google's own tensor processing units (TPUs)**

Libraries Used

❑ Keras

- **Keras is an open-source high-level Neural Network library, which is written in Python is capable enough to run on Theano, TensorFlow, or CNTK.**
- **It is made user-friendly, extensible, and modular for facilitating faster experimentation with deep neural networks.**
- **It not only supports Convolutional Networks and Recurrent Networks individually but also their combination.**

Libraries Used

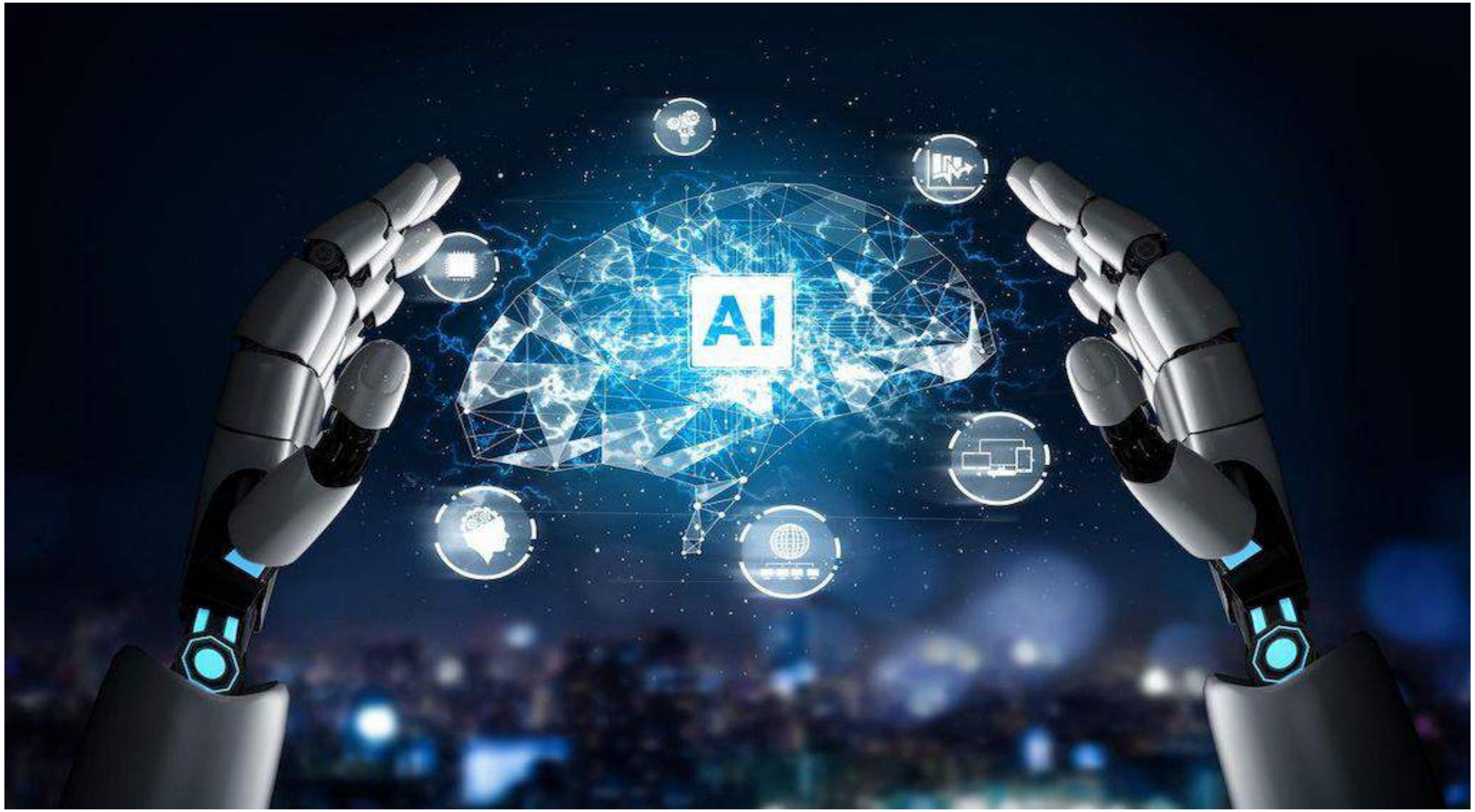
Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in python.

- Matplotlib relies on the Pillow library to load image data. It's a 24-bit RGB PNG. Depending on where you get your data, the other kinds of image that you'll most likely encounter are RGBA images, which allow for transparency, or single-channel grayscale (luminosity) images.

- It is used to generate graphs on model performance and used for showing images.

Generative adversarial network (GAN)



Generative adversarial network (GAN)

- **A generative adversarial network (GAN) is a class of machine learning frameworks and a prominent framework for approaching generative AI**

- **The concept was initially developed by Ian Goodfellow and his colleagues in June 2014**

- **In a GAN, two neural networks contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.**

- **Given a training set, this technique learns to generate new data with the same statistics as the training set.**

- **For example, a GAN trained on photographs can generate new photographs that look at least superficially authentic to human observers, having many realistic characteristics. Though originally proposed as a form of generative model for unsupervised learning, GANs have also proved useful for semi-supervised learning, fully supervised learning, and reinforcement learning**

Generative adversarial network (GAN)

The original GAN is defined as the following :

- **There are 2 players: generator and discriminator.**
- **The generator task is to match its own output distribution as closely as possible to the reference distribution.**
- **The generator aims to minimize the objective, and the discriminator aims to maximize the objective.**
- **The discriminator's task is to output a value close to 1 when the input appears to be from the reference distribution, and to output a value close to 0 when the input looks like it came from the generator distribution.**

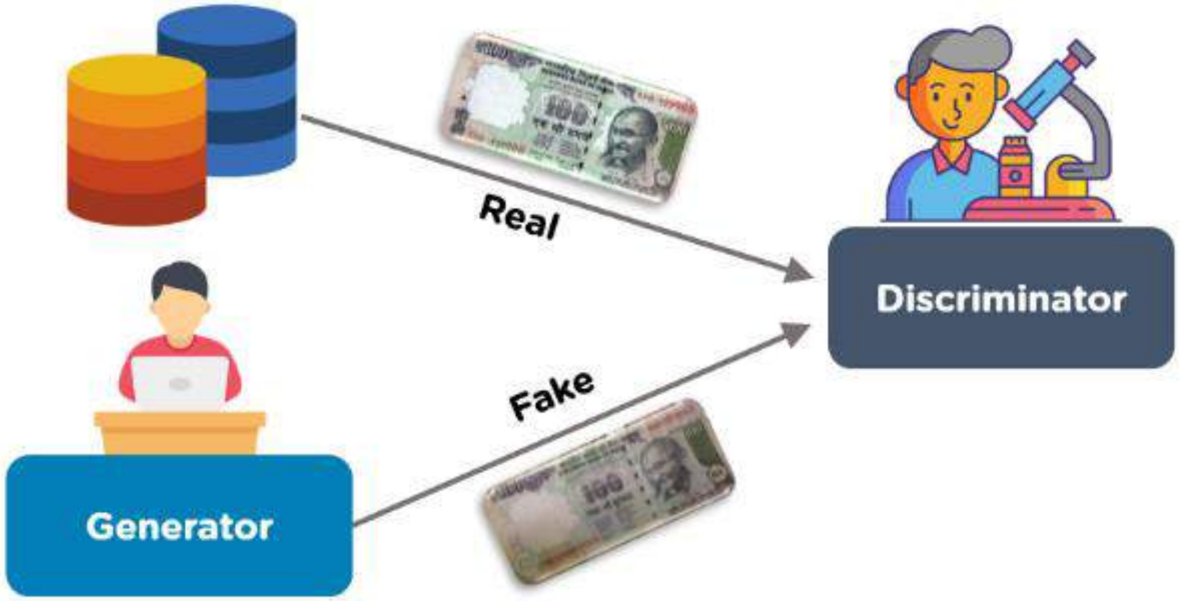
Generative adversarial network (GAN)

- The generative network generates candidates while the discriminative network evaluates them. The contest operates in terms of data distributions.

- Typically, the generative network learns to map from a latent space to a data distribution of interest, while the discriminative network distinguishes candidates produced by the generator from the true data distribution.

- The generative network's training objective is to increase the error rate of the discriminative network (i.e., "fool" the discriminator network by producing novel candidates that the discriminator thinks are not synthesized (are part of the true data distribution)).

Generative adversarial network (GAN)



Generative adversarial network (GAN)

- A known dataset serves as the initial training data for the discriminator. Training involves presenting it with samples from the training dataset until it achieves acceptable accuracy. The generator is trained based on whether it succeeds in fooling the discriminator.

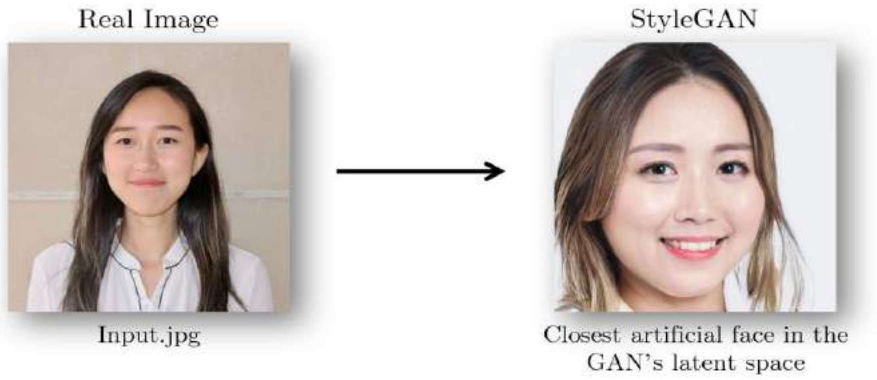
- Typically, the generator is seeded with randomized input that is sampled from a predefined latent space (ex: a multivariate normal distribution). Thereafter, candidates synthesized by the generator are evaluated by the discriminator.

- Independent backpropagation procedures are applied to both networks so that the generator produces better samples, while the discriminator becomes more skilled at flagging synthetic samples.

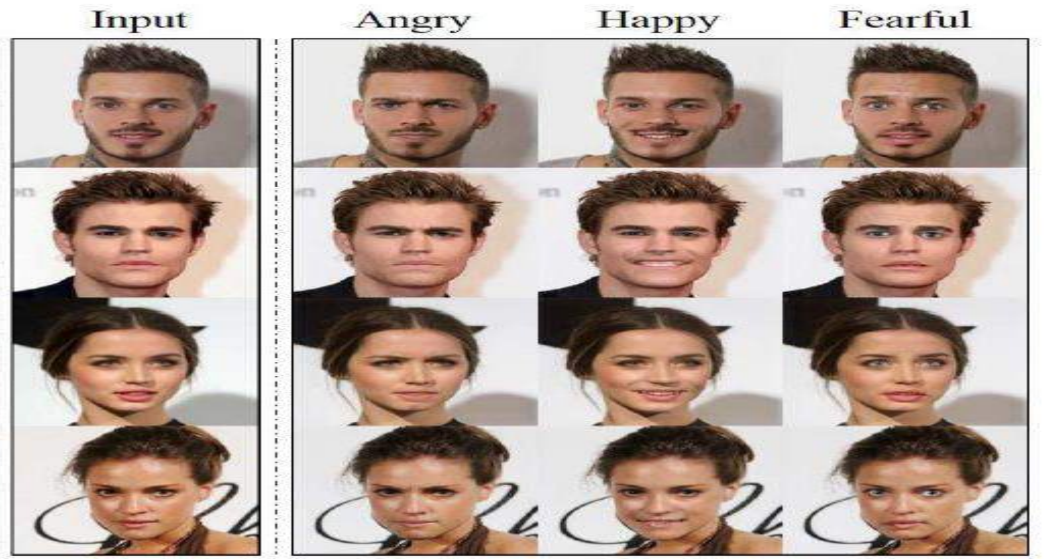
- When used for image generation, the generator is typically a deconvolutional neural network, and the discriminator is a convolutional neural network.

Different types of GANs

- **StyleGAN**

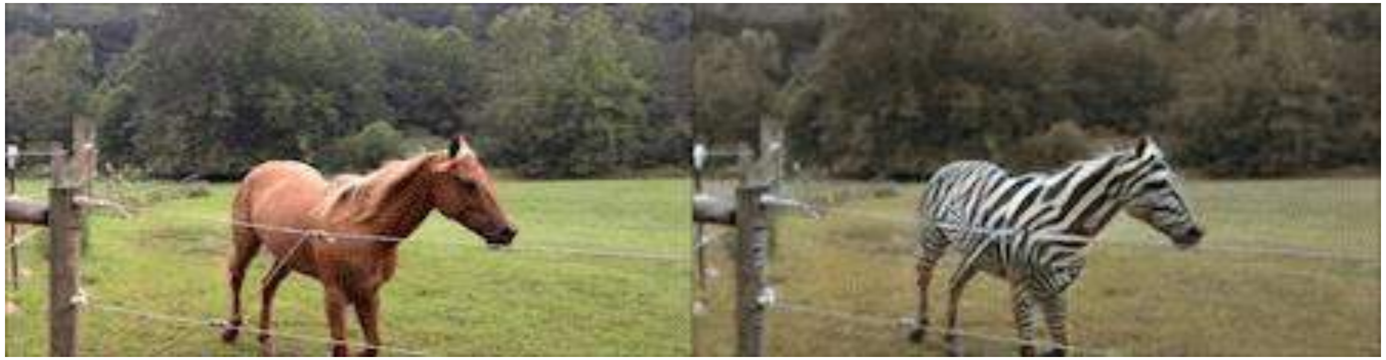


- **StarGAN**

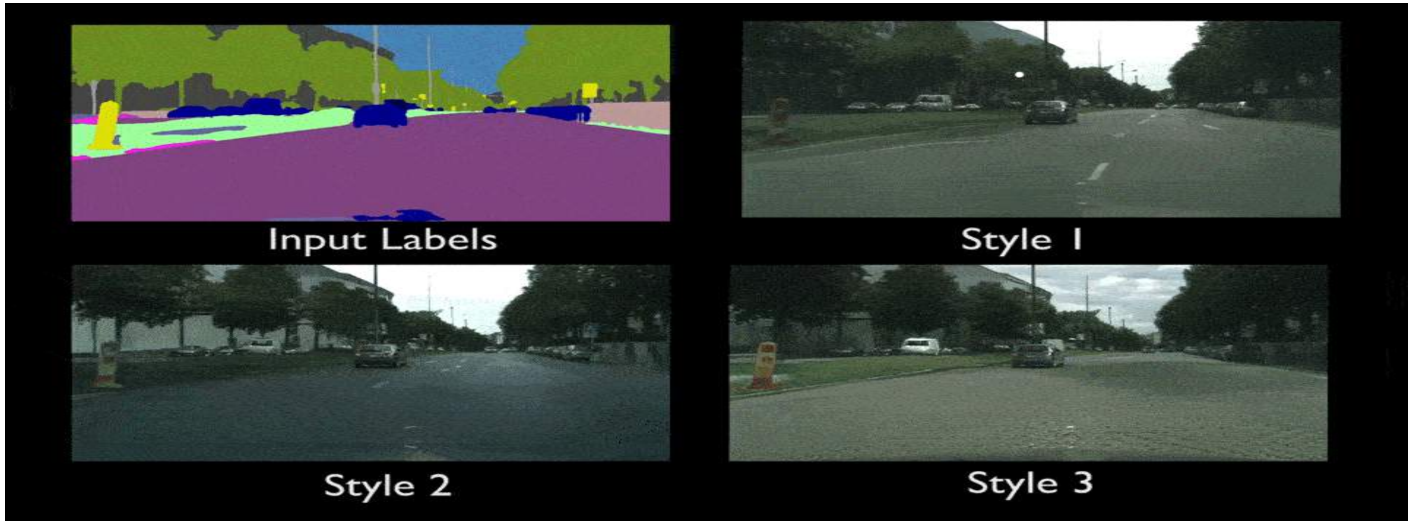


Different types of GANs

- ❑ **Cycle GAN**

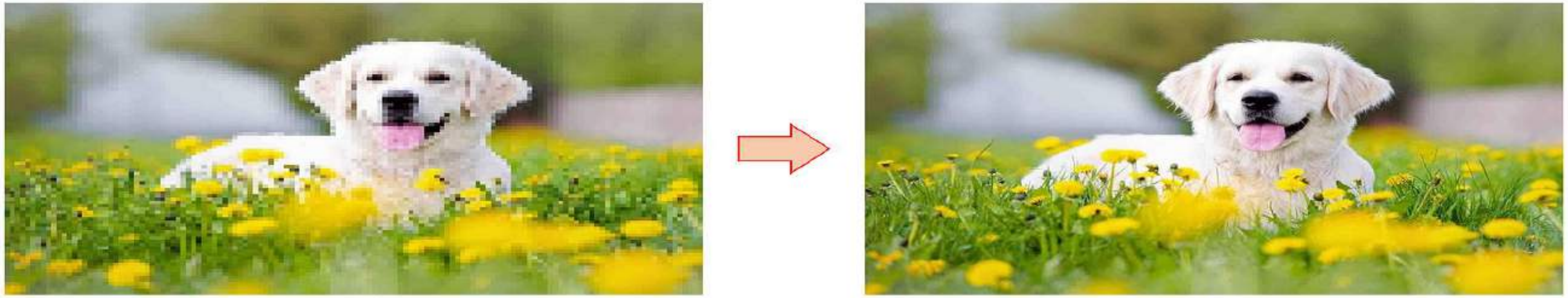


- ❑ **Conditional GAN (CGAN)**



Different types of GANs

❑ **Super Resolution (SRGAN)**



Low Resolution

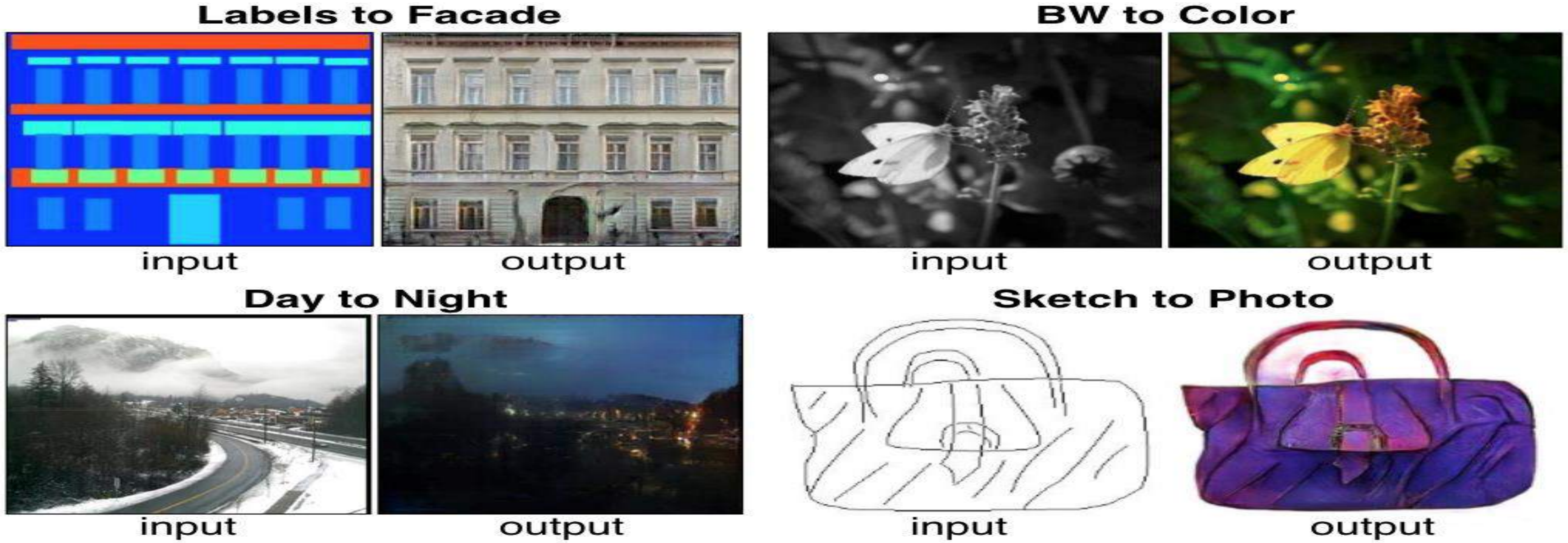
High Resolution

❑ **FutureGAN**

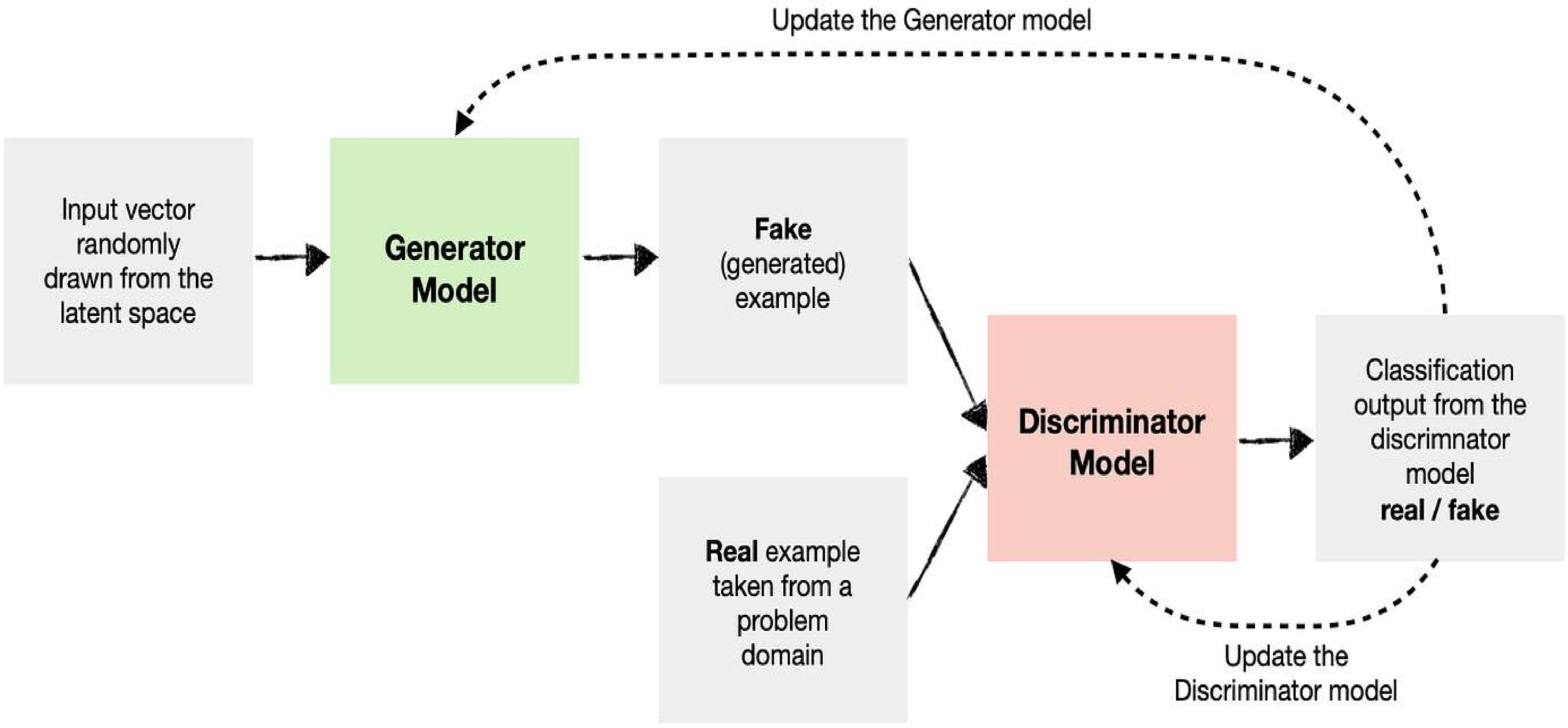
Pix2Pix: image translation with **Conditional GAN**

- **This Project is used to build and train a conditional generative adversarial network (cGAN) called pix2pix that learns a mapping from input images to output images, as described in image-to-image translation with conditional generative adversarial networks.**

- **pix2pix is not application specific—it can be applied to a wide range of tasks, including synthesizing photos from label maps, generating colorized photos from black and white images, turning Google Maps photos into aerial images, and even transforming sketches into photos.**



Pix2Pix: image translation with **Conditional GAN**


Update the Generator model
Input vector randomly drawn from the latent space
Generator Model
Fake (generated) example
Discriminator Model
Classification output from the discrimnator model **real / fake**
Real example taken from a problem domain
Update the Discriminator model

Pix2Pix: image translation with **Conditional GAN**

- **In the pix2pix cGAN, your condition on input images and generate corresponding output images.**

- **The architecture of your network will contain:**

    - **A generator with a U-Net-based architecture.**
    - **A discriminator represented by a convolutional PatchGAN classifier (proposed in the pix2pix paper)**

- **Note that each epoch can take around 15 seconds on a single V100 GPU.**

- **As described in the pix2pix paper, you need to apply random jittering and mirroring to preprocess the training set. Define several functions that**

    - **Resize each 256 x 256 image to a larger height and width—286 x 286.**
    - **Randomly crop it back to 256 x 256.**
    - **Randomly flip the image horizontally i.e left to right (random mirroring).**
    - **Normalize the images to the [-1, 1] range.**

Pix2Pix: image translation with **Conditional GAN**

The Generator

- **The generator of your pix2pix cGAN is a modified U-Net. A U-Net consists of an encoder (downsampler) and decoder (upsampler).**

- **Each block in the encoder is: Convolution -> Batch normalization -> Leaky ReLU**

- **Each block in the decoder is: Transposed convolution -> Batch normalization -> Dropout (applied to the first 3 blocks) -> ReLU**

- **There are skip connections between the encoder and decoder (as in the U-Net).**

Pix2Pix: image translation with **Conditional GAN**

The Generator Loss

GANs learn a loss that adapts to the data, while cGANs learn a structured loss that penalizes a possible structure that differs from the network output and the target image, as described in the pix2pix paper

- **The generator loss is a sigmoid cross-entropy loss of the generated images and an array of ones**
- **This allows the generated image to become structurally similar to the target image.**
- **The formula to calculate the total generator loss is gan_loss + LAMBDA * l1_loss, where LAMBDA = 100. This value was decided by the authors of the paper.**

Pix2Pix: image translation with **Conditional GAN**

The Discriminator

The discriminator in the pix2pix cGAN is a convolutional PatchGAN classifier—it tries to classify if each image *patch* is real or not real, as described in the pix2pixpaper.

- **Each block in the discriminator is: Convolution -> Batch normalization -> Leaky ReLU.**
- **The shape of the output after the last layer is (batch_size, 30, 30, 1).**
- **Each 30 x 30 image patch of the output classifies a 70 x 70 portion of the input image.**
- **The discriminator receives 2 inputs**
  - **The input image and the target image, which it should classify as real**
  - **The input image and the generated image (the output of the generator), which it should classify as fake.**
  - **Using tf.concat([inp, tar], axis=-1) to concatenate these 2 inputs together.**

Pix2Pix: image translation with **Conditional GAN**

The Discriminator Loss

- **The discriminator loss function takes 2 inputs: real images and generated images.**
- **Real loss is a sigmoid cross-entropy loss of the real images and an array of ones(since these are the real images).**
- **Generated loss is a sigmoid cross-entropy loss of the generated images and an array of zeros (since these are the fake images).**
- **The total loss is the sum of real loss and generated loss.**

Pix2Pix: image translation with **Conditional GAN**

Training

- **For each example input generates an output.**
- **The discriminator receives the input image and the generated image as the first input. The second input is the input image and the target image.**
- **Next, calculate the generator and the discriminator loss.**
- **Then, calculate the gradients of loss with respect to both the generator and the discriminator variables(inputs) and apply those to the optimizer.**

Pix2Pix: image translation with **Conditional GAN**

Training

Interpreting the logs is more subtle when training a GAN (or a cGAN like pix2pix) compared to a simple classification or regression model. Things to look for:

- Check that neither the generator nor the discriminator model has "won". If either the gen gan loss or the disc loss gets very low, it's an indicator that this model is dominating the other, and you are not successfully training the combined model.

- The value log(2) = 0.69 is a good reference point for these losses, as it indicates a perplexity of 2 - the discriminator is, on average, equally uncertain about the two options.

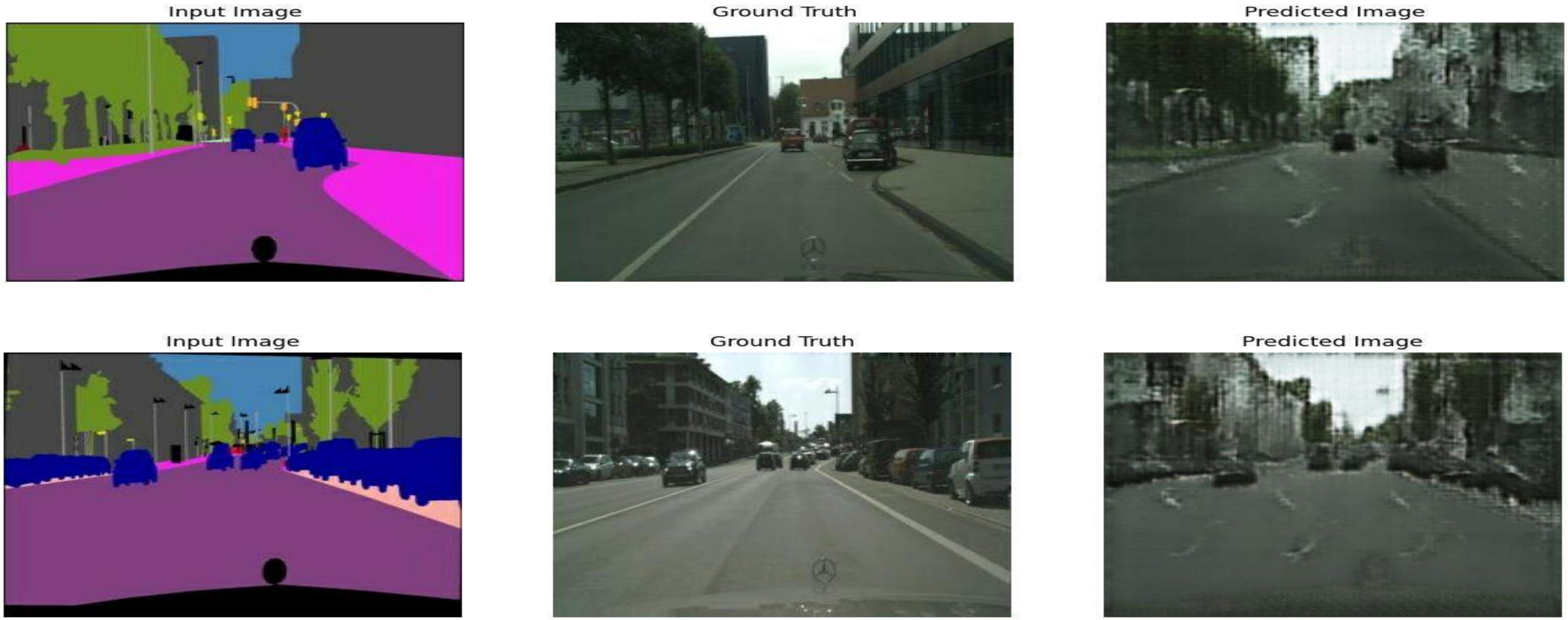
- For the disc_loss, a value below 0.69 means the discriminator is doing better than random on the combined set of real and generated images.

- For the gen_gan_loss, a value below 0.69 means the generator is doing better than random at fooling the discriminator.

- As training progresses, the gen_l1_loss should go down.

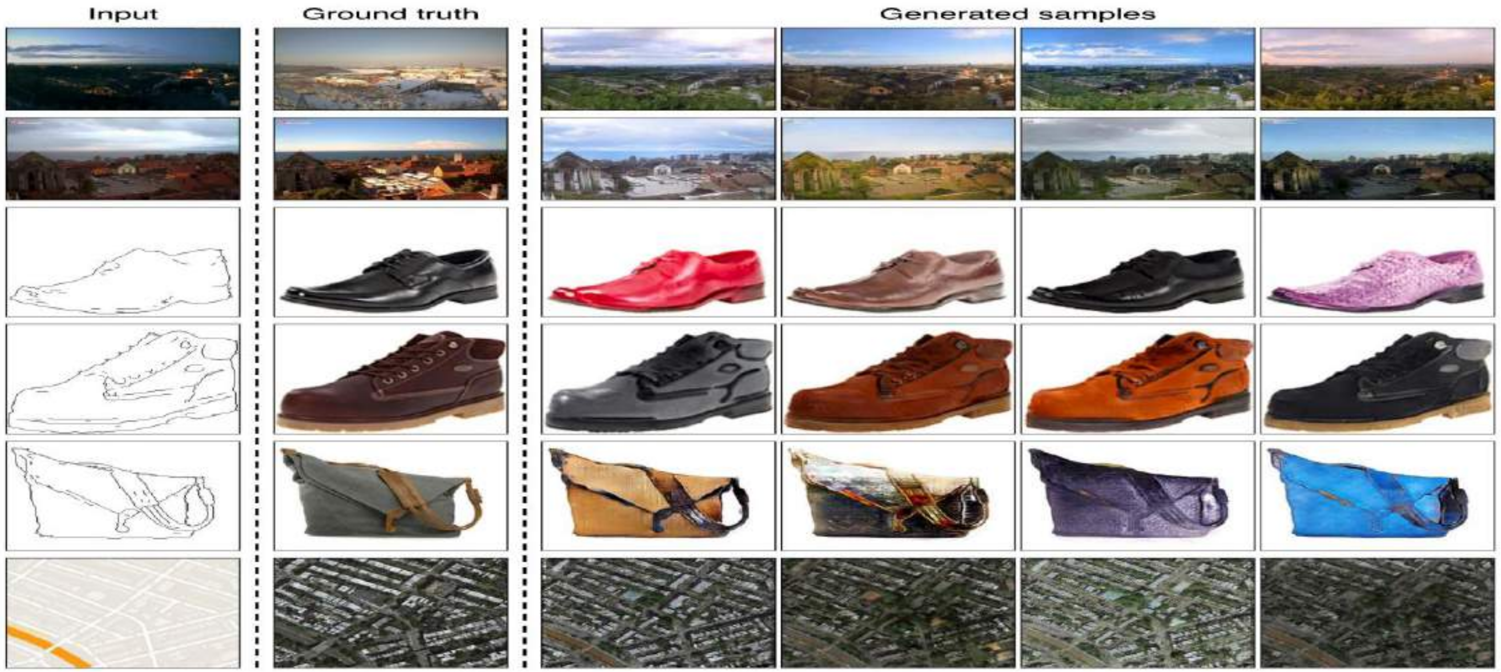
Pix2Pix: image translation with **Conditional GAN**

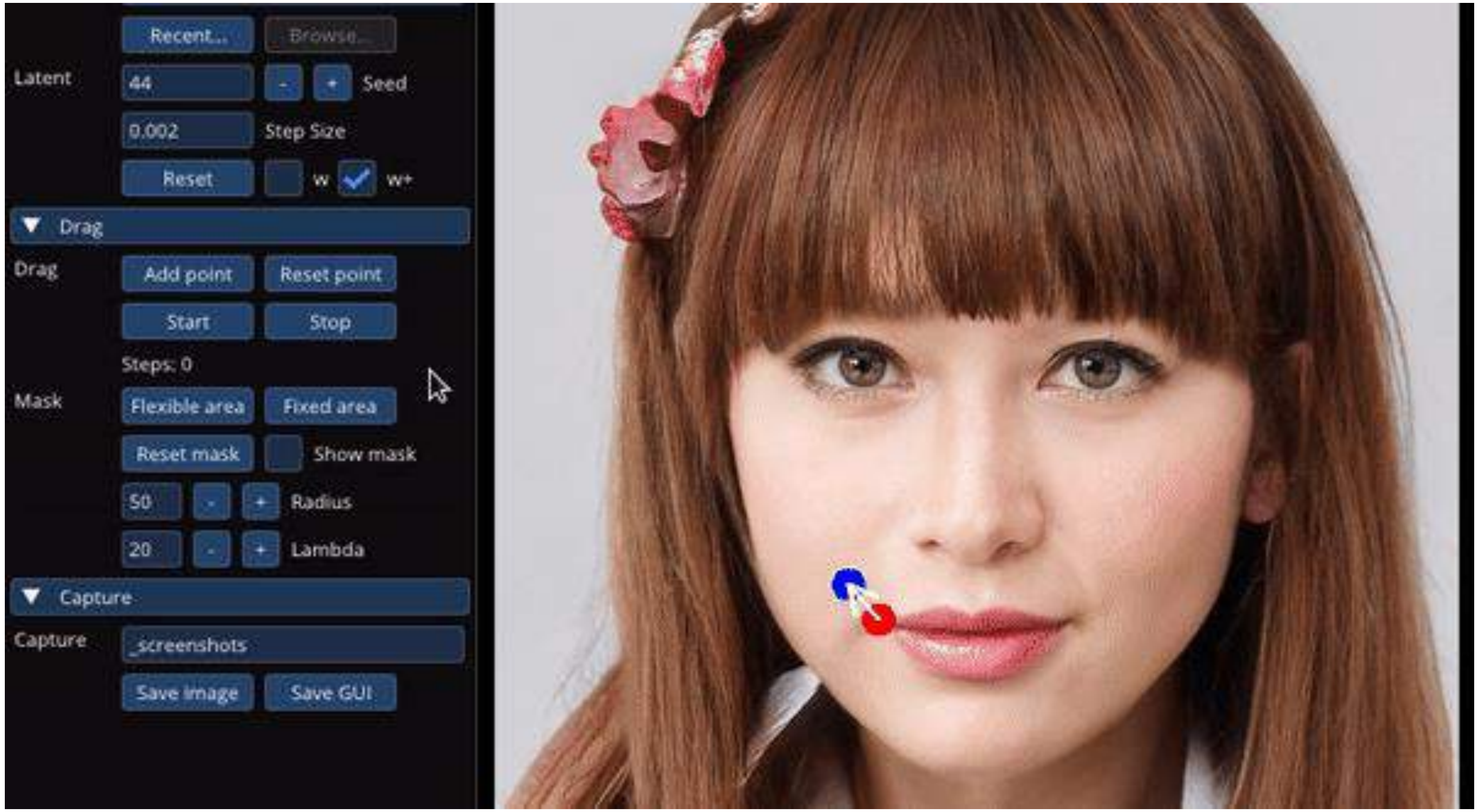
Generating some images using test data



Pix2Pix: Image to Image translation Uses

- It can be applied to a wide range of tasks, including synthesizing photos from label maps, generating colorized photos from black and white images, turning Google Maps photos into aerial images, and even transforming sketches into photos.





Latent 44 Seed

0.002 Step Size

□ w ☑ w+

Steps: 0

□ Show mask

50 Radius

20 Lambda

Capture _screenshots

Thank you!