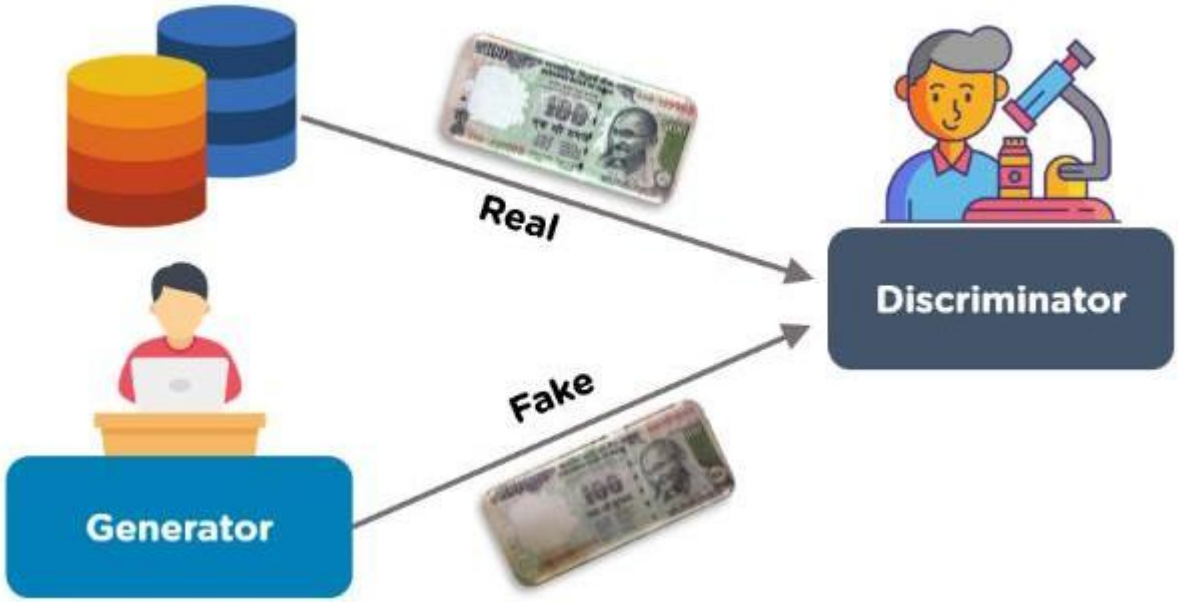
Module-6

Variational Autoencoders, Generative Adversarial Networks, Multi-task Deep Learning, Multi-view Deep Learning. Various Applications - speech, text, image and video

Introduction

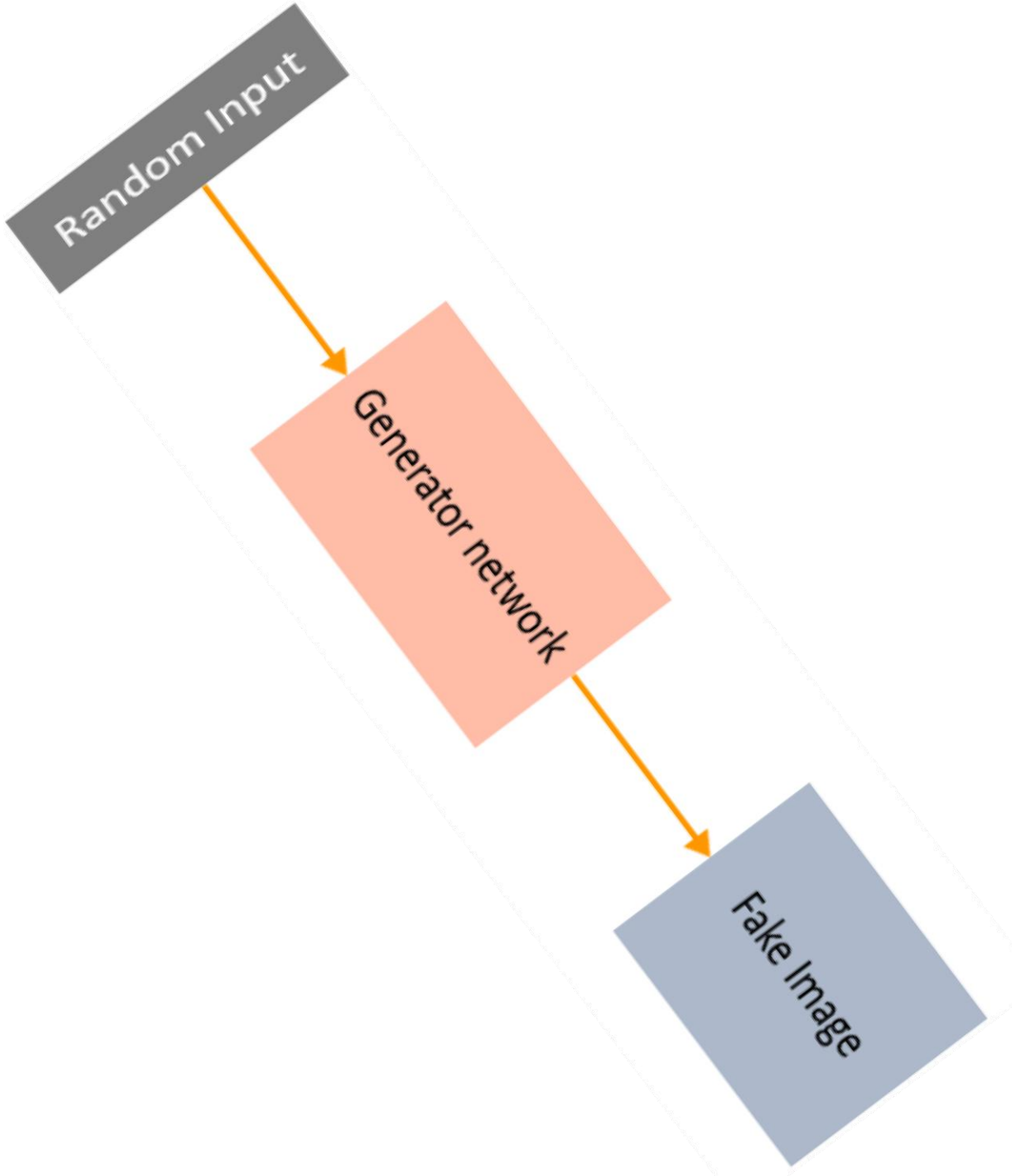
- Generative Adversarial Networks (GANs) were introduced in 2014 by Ian J. Goodfellow and co-authors.
- GANs perform unsupervised learning tasks in machine learning.
- GANs introduce the concept of **adversarial learning**, as they lie in the rivalry between two neural networks.
- These techniques have enabled researchers to create realistic-looking but entirely computer generated photos of people's faces.
- They have also allowed the creation of controversial "deepfake" videos.
- Actually, GANs can be used to imitate any data distribution (image, text, sound, etc.).

- It consists of 2 models that automatically discover and learn the patterns in input data.
- The two models are known as Generator and Discriminator.
- GANs can be used to generate new examples that plausibly could have been drawn from the original dataset.

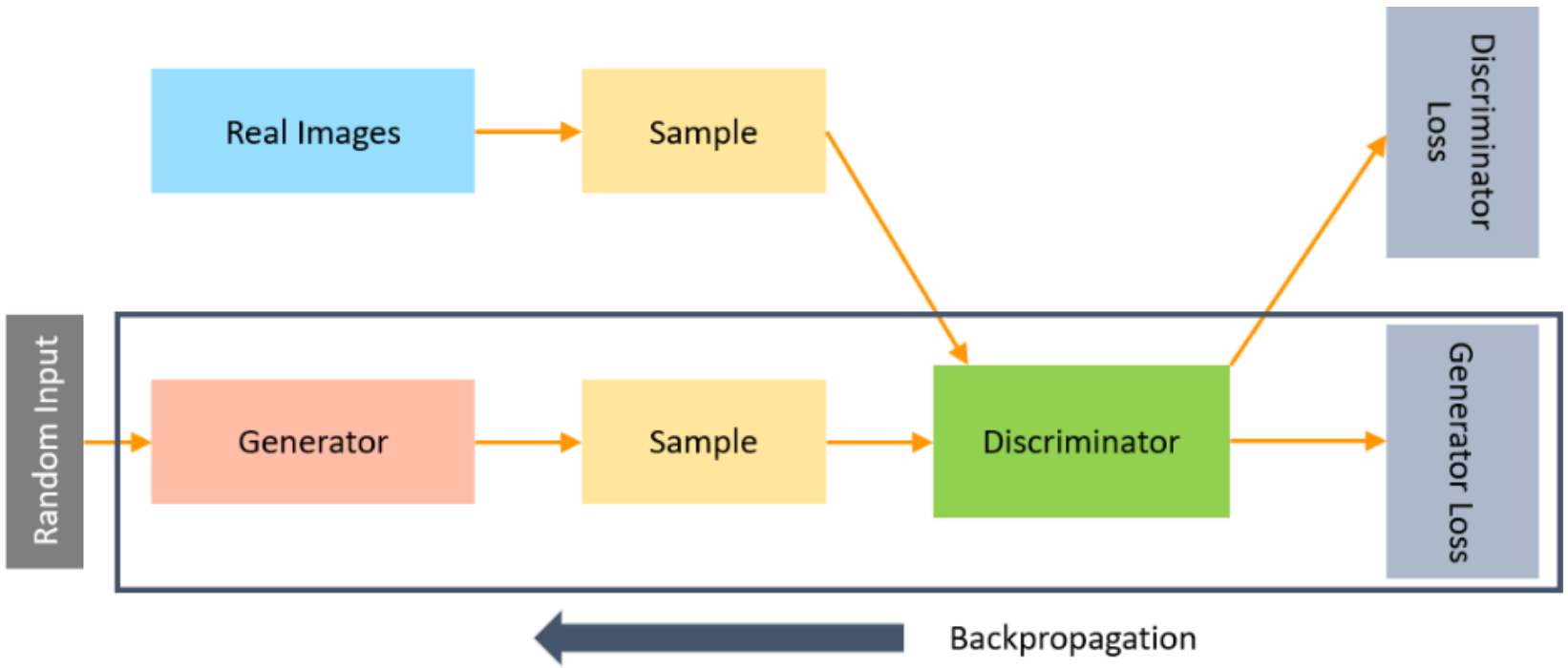


Generator

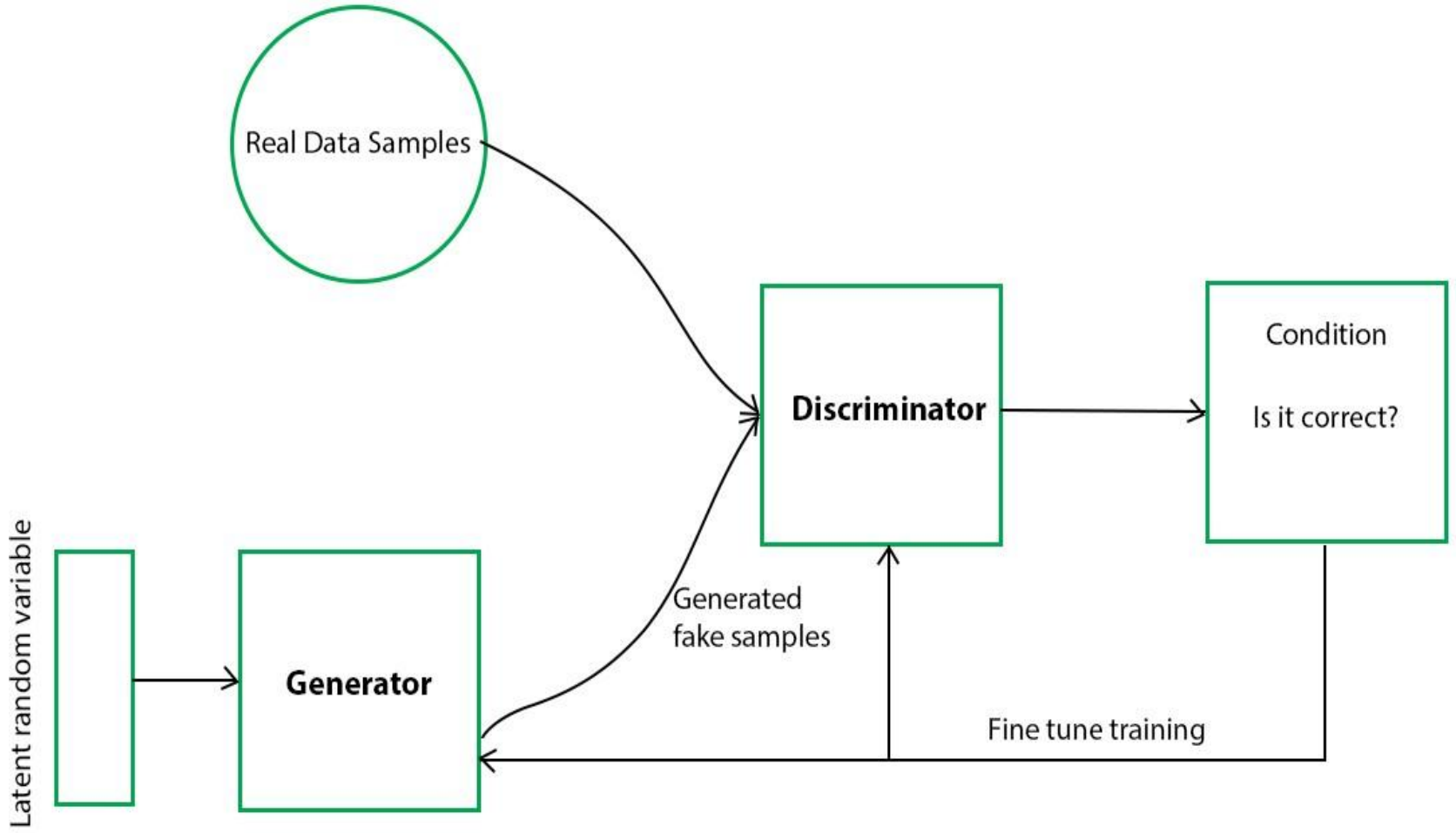
- A Generator in GANs is a neural network that creates fake data to be trained on the discriminator.
- It learns to generate plausible data.
- The generated examples/instances become negative training examples for the discriminator.
- It takes a fixed-length random vector carrying noise as input and generates a sample.



- The main aim of the Generator is to make the discriminator classify its output as real.
- The part of the GAN that trains the Generator includes:
- noisy input vector
- generator network, which transforms the random input into a data instance
- discriminator network, which classifies the generated data
- generator loss, which penalizes the Generator for failing to dolt the discriminator
- The backpropagation method is used to adjust each weight in the right direction by calculating the weight's impact on the output.
- It is also used to obtain gradients and these gradients can help change the generator weights.


Real Images
Sample
Discriminator Loss
Random Input
Generator
Sample
Discriminator
Generator Loss
Backpropagation

- Its aim is to generate the fake image based on feedback and make the discriminator fool that it cannot predict a fake image.
- And when the discriminator is made a fool by the generator, the training stops and we can say that a generalized GAN model is created.



- Generative model captures the distribution of data and is trained in such a manner to generate the new sample that tries to maximize the probability of the discriminator to make a mistake(maximize discriminator loss).
- The discriminator on other hand is based on a model that estimates the probability that the sample it receives is from training data not from the generator and tries to classify it accurately and minimize the GAN accuracy.
- Hence the GAN network is formulated as a minimax game where the Discriminator is trying to minimize its reward **V(D, G)** and the generator is trying to maximize the Discriminator loss.
- Generator output is directly connected to the input of the discriminator.
- And discriminator predicts it and through backpropagation, the generator receives a feedback signal to update weights and improve performance.
- The discriminator is a feed-forward neural network.

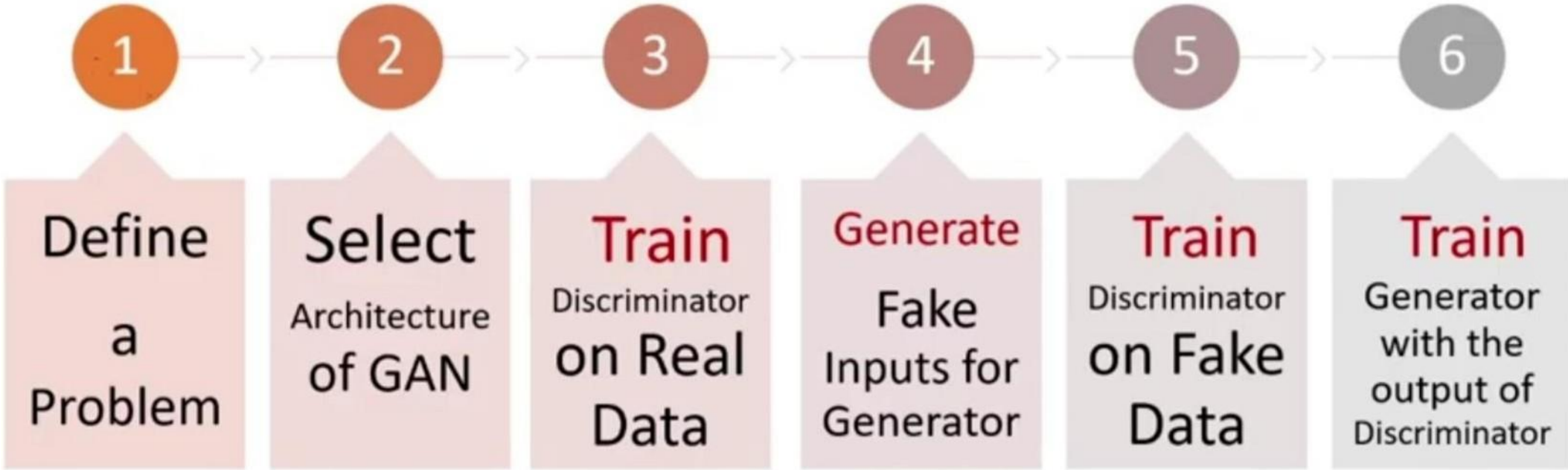
Training & Prediction of Generative Adversarial Networks (GANs)

- **Step-1) Define a Problem**
- **Step-2) Select Architecture of GAN**
- **Step-3) Train Discriminator on Real Dataset**
  - Discriminator is trained on a real dataset.
  - It is only having a forward path, no backpropagation is there in the training of the Discriminator in n epochs.
  - And the Data you are providing is without Noise and only contains real images, and for fake images, Discriminator uses instances created by the generator as negative output.
- It classifies both real and fake data.
- The discriminator loss helps improve its performance and penalize it when it misclassifies real as fake or vice-versa.
- weights of the discriminator are updated through discriminator loss.

- **Step-4) Train Generator**
  - Provide some Fake inputs for the generator(Noise) and It will use some random noise and generate some fake outputs.
  - When Generator is trained, Discriminator is Idle and when Discriminator is trained, Generator is Idle.
  - During generator training through any random noise as input, it tries to transform it into meaningful data. to get meaningful output from the generator takes time and runs under many epochs.
  - Steps to train a generator
    - get random noise and produce a generator output on noise sample
    - predict generator output from discriminator as original or fake.
    - we calculate discriminator loss.
    - perform backpropagation through discriminator, and generator both to calculate gradients.
    - Use gradients to update generator weights.

- **Step-5) Train Discriminator on Fake Data**
  - The samples which are generated by Generator will pass to Discriminator and It will predict the data passed to it is Fake or real and provide feedback to Generator again.
- **Step-6) Train Generator with the output of Discriminator**
  - Generator will be trained on the feedback given by Discriminator and try to improve performance.
  - This is an iterative process and continues running until the Generator is not successful in making the discriminator fool.

Training of GAN



Generative Adversarial Networks (GANs) Loss Function

- The generator tries to minimize the following loss function while the discriminator tries to maximize it. It is the same as a minimax game if you have ever played.

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

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

- D(x) is the discriminator’s estimate of the probability that real data 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 random inputs to the generator (in effect, the expected value over all generated fake instances G(z)).
- The formula is derived from cross-entropy between

Discriminator loss

- While the discriminator is trained, it classifies both the real data and the fake data from the generator.
- It penalizes itself for misclassifying a real instance as fake, or a fake instance (created by the generator) as real, by maximizing the below function.

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

- **log(D(x))** refers to the probability that the generator is rightly classifying the real image,
- maximizing **log(1-D(G(z)))** would help it to correctly label the fake image that comes from the generator.

Generator loss

- While the generator is trained, it samples random noise and produces an output from that noise. The output then goes through the discriminator and gets classified as either “Real” or “Fake” based on the ability of the discriminator to tell one from the other.
- The generator loss is then calculated from the discriminator’s classification – it gets rewarded if it successfully fools the discriminator, and gets penalized otherwise.
- The following equation is minimized to training the generator:

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

Challenges of GAN

- **Mode Collapse**
- **we would want our GAN to produce a range of outputs**. We would expect, for example, another face for every random input to the face generator that we design.
- Instead, through subsequent training, the network learns to model a particular distribution of data, which gives us a **monotonous output**

- In the process of training, the **generator is always trying to find the one output** that seems most plausible to the discriminator.
- Because of that, the **discriminator’s best strategy is always to reject the output** of the generator.
- But if the next generation of discriminator gets stuck in a local minimum and doesn’t find its way out by getting its weights even more optimized, it’d get easy for the next generator iteration to find the most plausible output for the current discriminator.

Vanishing Gradients

- This phenomenon happens when the discriminator performs significantly better than the generator. Either the updates to the discriminator are inaccurate, or they disappear.
- One of the proposed reasons for this is that the generator gets heavily penalized, which leads to saturation in the value post-activation function, and the eventual gradient vanishing.

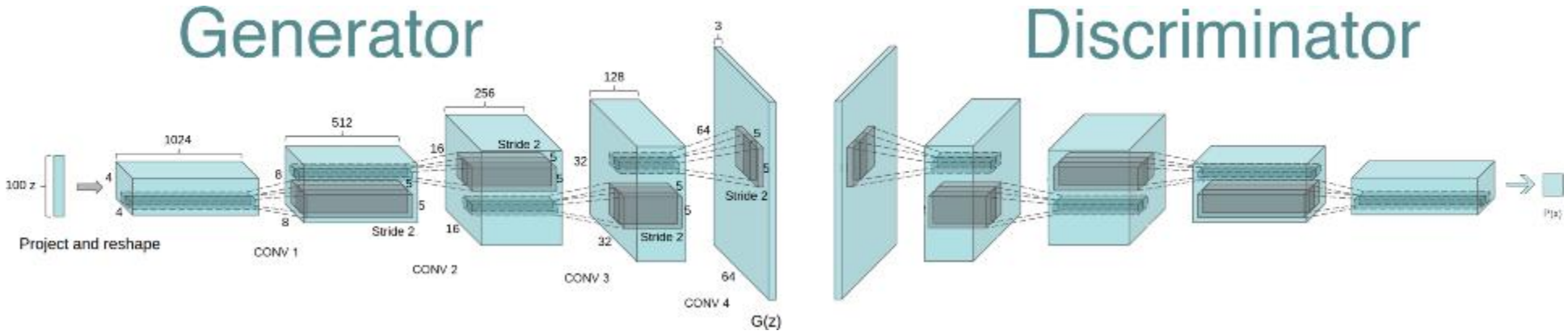
Wasserstein Generative Adversarial Network (WGAN)

- This is one of the most powerful alternatives to the original GAN loss. It **tackles the problem of Mode Collapse and Vanishing Gradient.**
- In this implementation, the **activation of the output layer of the discriminator is changed from sigmoid to a linear one**. This simple change influences the discriminator to give out a score instead of a probability associated with data distribution, so the output does not have to be in the range of 0 to 1.
- Here, the discriminator is called critique instead, because it doesn't actually classify the data strictly as real or fake, it simply gives them a rating.
- Following loss functions are used to **train the critique and the generator**, respectively.
- The output of the critique and the generator is not in probabilistic terms (between 0 and 1), so the absolute difference between critique and generator outputs is maximized while training the critique network.
- Similarly, the absolute value of the generator function is maximized while training the generator network.

- CycleGAN
- StyleGAN
- pixelRNN
- text-2-image
- DiscoGAN
- lsGAN
- SRGAN
- InfoGAN

Deep Convolutional GAN

- DCGANs are an improvement of GANs that use convolutional neural nets.
- CNNs are good for extracting important features and representation from the data, making them more stable and enabling them to generate higher quality images.



Conditional GAN

- Conditional GANs train on a labeled data set and let you specify the label for each generated instance. For example, an unconditional MNIST GAN would produce random digits, while a conditional MNIST GAN would let you specify which digit the GAN should generate.
- Instead of modeling the joint probability P(X, Y), conditional GANs model the conditional probability P(X | Y).
- **Application:**
- Image-to-image translation
- Text-to-image synthesis
- Video generation

Image-to-Image Translation(Pix2Pix GAN)

- Image-to-Image translation GANs take an image as input and map it to a generated output image with different properties. For example, we can take a mask image with blob of color in the shape of a car, and the GAN can fill in the shape with photorealistic car details.
- Similarly, you can train an image-to-image GAN to take sketches of handbags and turn them into photorealistic images of handbags.
- loss is a weighted combination of the usual discriminator-based loss and a pixel-wise loss that penalizes the generator for departing from the source image.

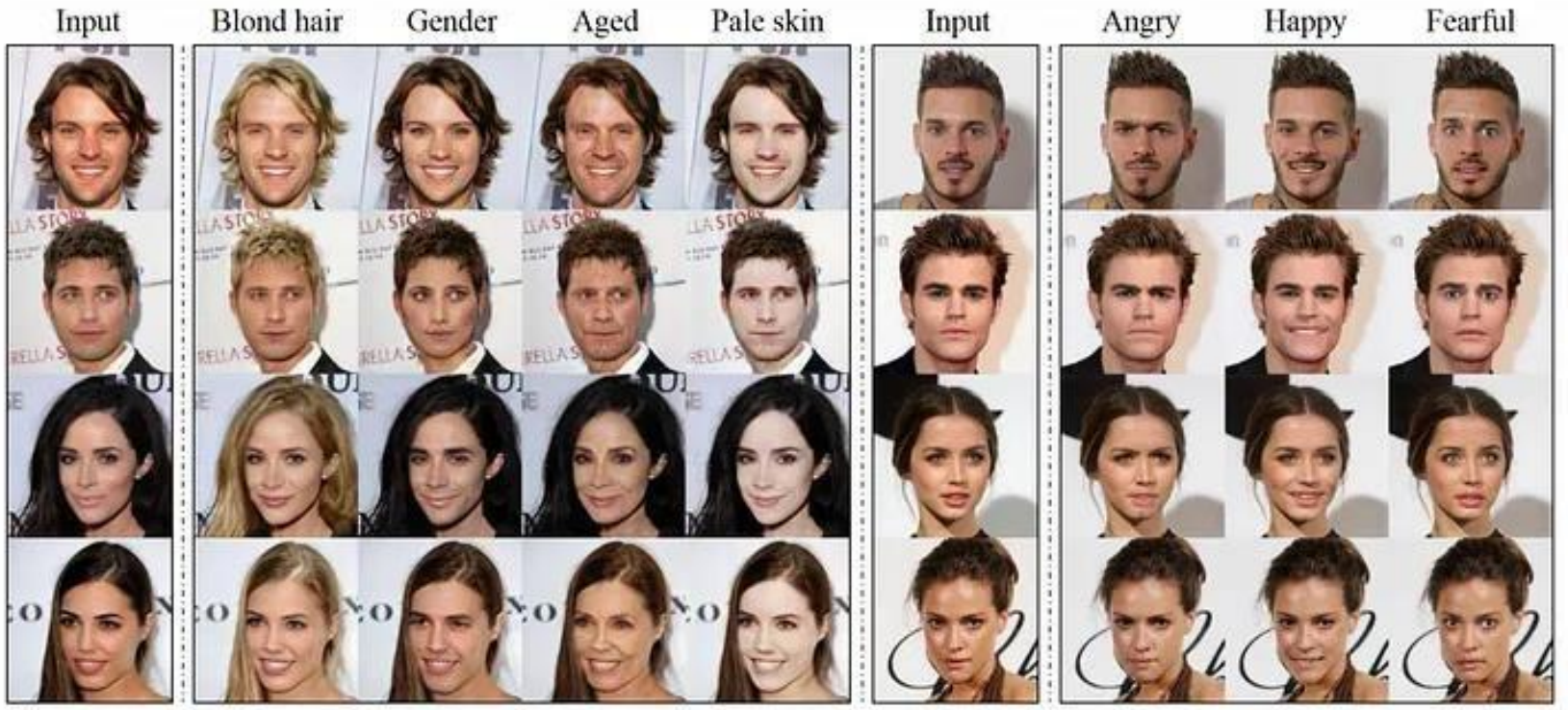
CycleGAN

- CycleGANs learn to transform images from one set into images that could plausibly belong to another set. For example, a CycleGAN produced the righthand image below when given the lefthand image as input. It took an image of a horse and turned it into an image of a zebra.
- The training data for the CycleGAN is simply two sets of images (in this case, a set of horse images and a set of zebra images). The system requires no labels or pairwise correspondences between images.

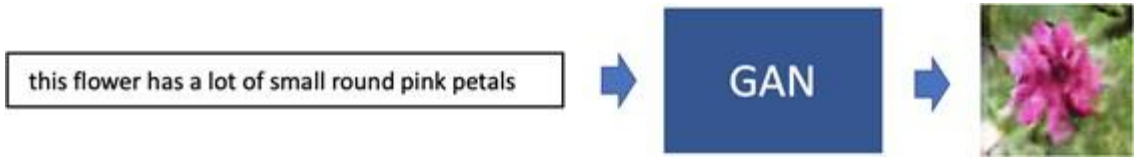
StyleGAN

- ***The idea is to build a stack of layers where initial layers are capable of generating low-resolution images (starting from 2*2) and further layers gradually increase the resolution.***
- The easiest way for GAN to generate high-resolution images is to remember images from the training dataset and while generating new images it can add random noise to an existing image.
- In reality, StyleGAN doesn’t do that rather it learn features regarding human face and generates a new image of the human face that doesn’t exist in reality.

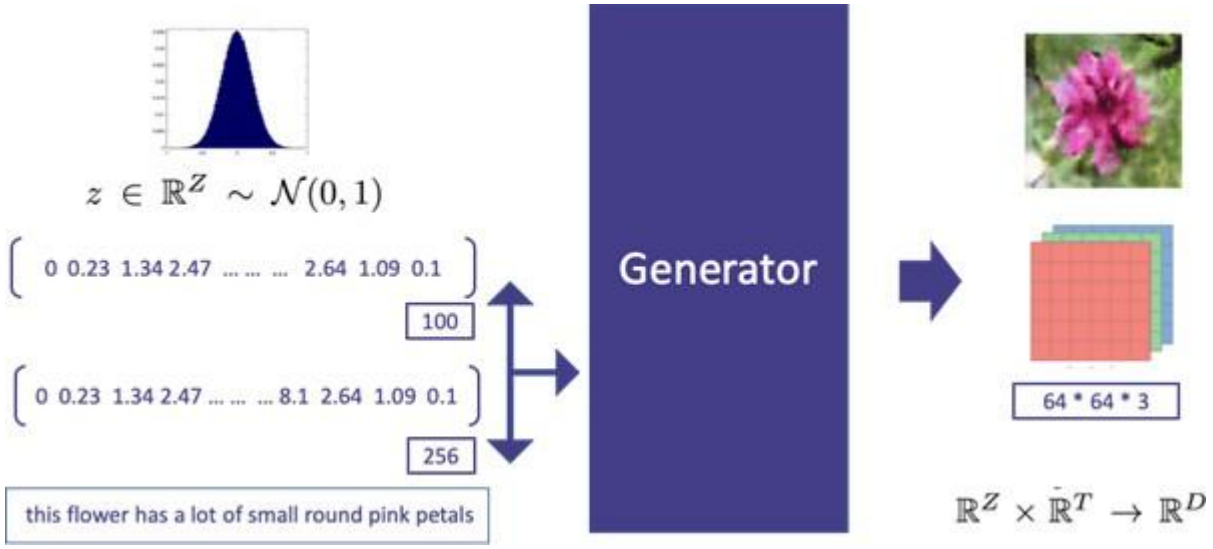
StarGAN


Input
Blond hair
Gender
Aged
Pale skin
Input
Angry
Happy
Fearful

text-2-image



- *Instead of giving only noise as input to the Generator, the textual description is first transformed into a text embedding, concatenated with noise vector and then given as input to the Generator.*



- For the Discriminator, instead of having the only image as input, a pair of image and text embedding are sent as input.
- Output signals are either 0 or 1. Earlier the Discriminator's responsibility was just to predict whether a given image is real or fake.
- ***Now, the Discriminator has one more additional responsibility.***
- ***Along with identifying the given image is read or fake, it also predicts the likelihood of whether the given image and text aligned with each other.***
- This formulation force the Generator to not only generate images that look real but also to generate images that are aligned with the input textual description.
-

- To fulfill the purpose of the 2-fold responsibility of the Discriminator, during training time, a series of different (image, text) pairs are given as input to the model which are as follows:
  - Pair of (Real Image, Real Caption) as input and target variable is set to 1
  - Pair of (Wrong Image, Real Caption) as input and target variable is set to 0
  - Pair of (Fake Image, Real Caption) as input and target variable is set to 0
- The pair of Real Image and Real Caption are given so that the model learns whether a given image and text pair are aligned with each other.
- The wrong Image, Read Caption means the image is not as described in the caption.
- In this case, the target variable is set to 0 so that the model learns that the given image and caption are not aligned.
- Here Fake Image means an image generated by the Generator, in this case, the target variable is set to 0 so that the Discriminator model can distinguish between real and fake images.

- The training dataset used for the training has image along with 10 different textual description that describes properties of the image.

1. **the petals are long and purple, and oval shaped on the tips.**
2. this flower is pink in color, with petals that are oval shaped.
3. long pink slender petals with a light and dark purple and yellow pistil.
4. the flower has purple petals surrounding its yellow anther and blue filament
5. the flower has long elongated petals, with yellow anthers in the center.
6. a light velvet large flower with a purple golden center.
7. this flower has a single layer of small oblong purple petals.
8. this flower has petals that are pink and has yellow stigma
9. **the bright pink petals of this flower have a pointed edge, which is rounded, and surround a group of short stamen that are topped with bright pink antenna.**
10. the flat, pink petals form a disk around stamen with golden colored anther.

This flower has purple petal

I want a flower having red petal

Yellow is the color of petal

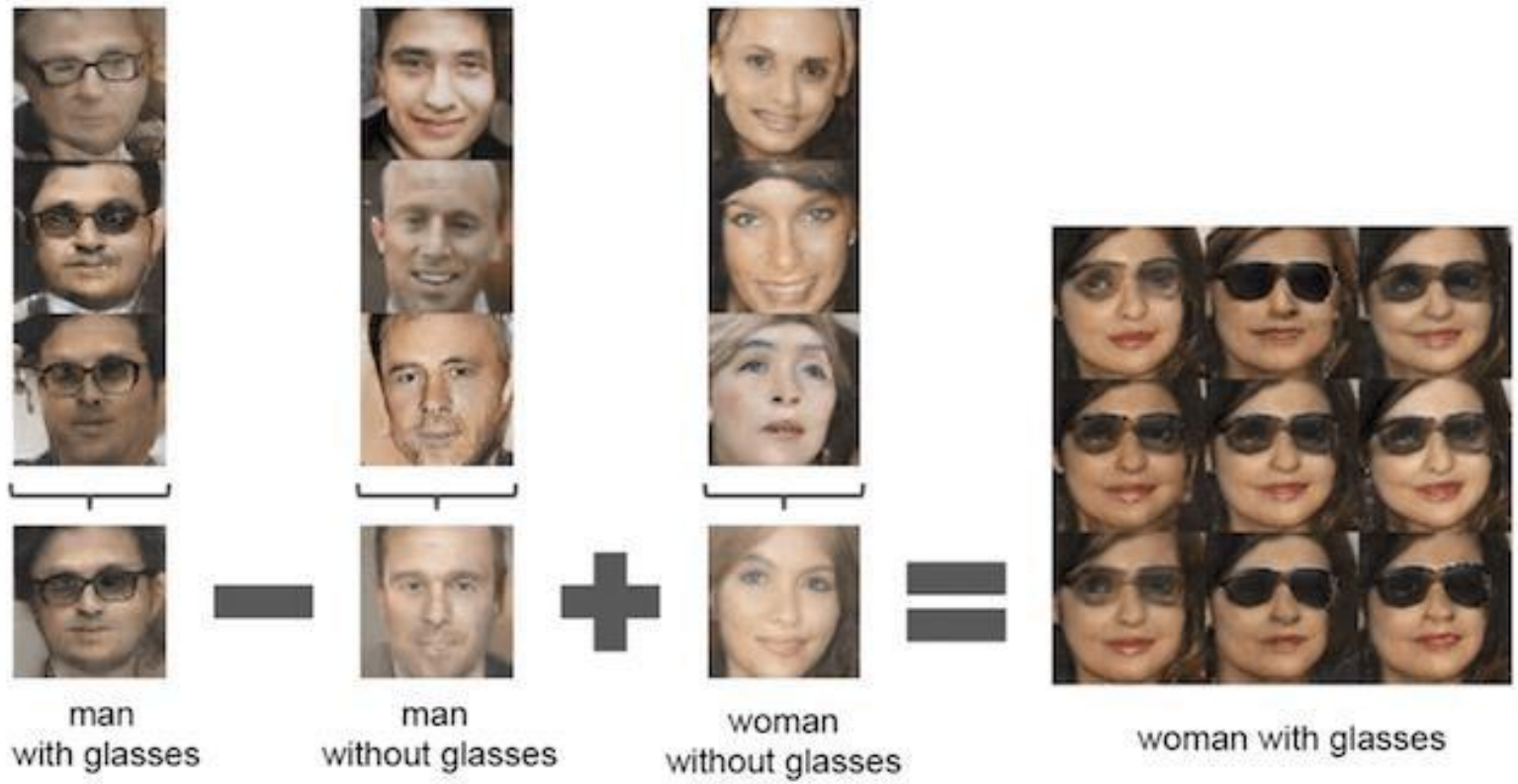
This flower is pink in color and petals are rounded

DiscoGAN

- given images of two different domains, a human can figure out how they are related to each other.
- As an example, in the following figure, we have images from 2 different domains and just by one glance at these images, we can figure out very easily that they are related by the nature of their exterior color.
- ***The primary difference between DiscoGAN and CycleGAN is that DiscoGAN uses two reconstruction loss, one for both the domain whereas CycleGAN uses single cycle-consistency loss.***

(b) Handbag images (input) & **Generated** shoe images (output)

- Generate New Human Poses
- GANs can generate images of people in new poses, such as difficult or impossible for humans to achieve. It can be used to create new content or to augment existing image datasets.
- Photos to Emojis
- GANs can be used to convert photographs of people into emojis, creating a more personalized and expressive form of communication.
- Photograph Editing
- GANs can be used to edit photographs in various ways, such as changing the background, adding or removing objects, or altering the appearance of people or animals in the image.
- Face Aging
- GANs can be used to generate images of people at different ages, allowing users to visualize how they might look in the future or to see what they might have looked like in the past.


man
with glasses
man
without glasses
woman
without glasses
woman with glasses

CGAN

StackedGAN

- Stacked Generative Adversarial Networks (StackGAN) can generate images conditioned on text descriptions.

- The architecture comprises a stacked series of text and image GAN models.
- It’s again a class of conditional GANs. It has two GANs, also known as stage-I GAN and stage-II GAN.
- StackGAN uses a sketch-refinement process where the first level generator, Stage-I GAN, is conditioned on text and generates a low-resolution image, i.e. the primitive shape and colors of the descriptive text.
- The second level generator, Stage-II GAN, is conditioned both on the text and on the low-resolution image, which takes the stage-I results and adds compelling detail.

LIMITATIONS

- One issue is that GANs can be notoriously difficult to train. This is because the two networks in a GAN (the generator and the discriminator) are constantly competing against others, which can make training unstable and slow.
- Additionally, GANs often require a large amount of training data in order to produce good results. This can be a problem if the dataset is not readily available or if it is too small.
- Finally, GANs can be vulnerable to mode collapse, which is when the generator only produces a limited number of outputs instead of the variety that is desired.

- **Density Estimation**, we still cannot predict the accuracy of the density of the evaluated model and state that this image is denser enough to move forward with. These metrices of the data generated are still deciding manually.
- GANs are the elegant mechanism of Data Generation but due to **Unstable Training** and unsupervised leanring method it becomes harder to train and generate output.