Advanced Topics in Deep Learning

Somesh Nandi

Attention Mechanisms

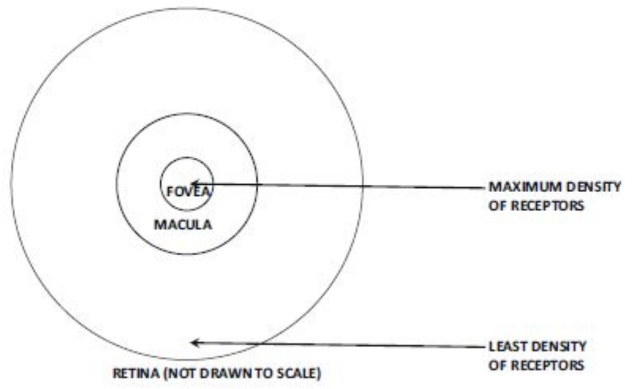
- Attention mechanisms in deep learning refer to techniques that allow a model to focus on specific parts of the input data when making predictions, rather than processing all parts of the input uniformly.

- This approach mimics how humans focus attention on relevant information while ignoring less relevant parts.

- Attention mechanisms have been particularly impactful in natural language processing (NLP), computer vision, and other sequence-based tasks

Attention Mechanisms

- **What is Attention?**
- In the context of neural networks, attention refers to the ability of the model to dynamically adjust the importance (or "weight") given to different parts of the input data while making a decision.
- Inspired by the Biological part of Eye


FOVEA
MACULA
MAXIMUM DENSITY OF RECEPTORS
LEAST DENSITY OF RECEPTORS
RETINA (NOT DRAWN TO SCALE)

Resolutions in different regions of the eye. Most of what we focus on is captured by the macula.

Attention Mechanisms for Machine Translation

- Translate the sentence: **"The cat sat on the mat."**
  to French: **"Le chat est assis sur le tapis.“**

- **Traditional Approach (Without Attention):**
- **Encoder-Decoder model:**
  - **Encoder** reads the entire sentence and converts it into a fixed-length vector (context vector).
  - **Decoder** uses this fixed-length vector to generate the translated sentence, word by word.
  - **Problem: The fixed-length vector may not capture all the details of a long sentence, leading to poor translations.**

Attention Mechanisms for Machine Translation

- With Attention Mechanism:
- Encoder (Initial Step):
- The encoder processes each word in the sentence and generates a sequence of hidden states, each representing a word's context (using an RNN or LSTM).

- The encoder output would look like a list of vectors corresponding to each word:
- Encoder outputs: [h_1, h_2, h_3, h_4, h_5]

Attention Mechanisms for Machine Translation

- **Attention Mechanism:** At each time step of decoding, the attention mechanism decides how much focus should be given to each word of the input.
- For instance, when the decoder generates the French word "Le" (which corresponds to "The" in English), it looks at the hidden states of the encoder (h_1 to h_5) and computes a set of attention weights indicating how much attention each input word should get.

- For the word "Le", the attention mechanism may give high weights to the first word "The" and low weights to the other words, because "Le" corresponds to "The" in French.
- Attention weights for "Le" = [0.7, 0.1, 0.05, 0.05, 0.1]

Attention Mechanisms for Machine Translation

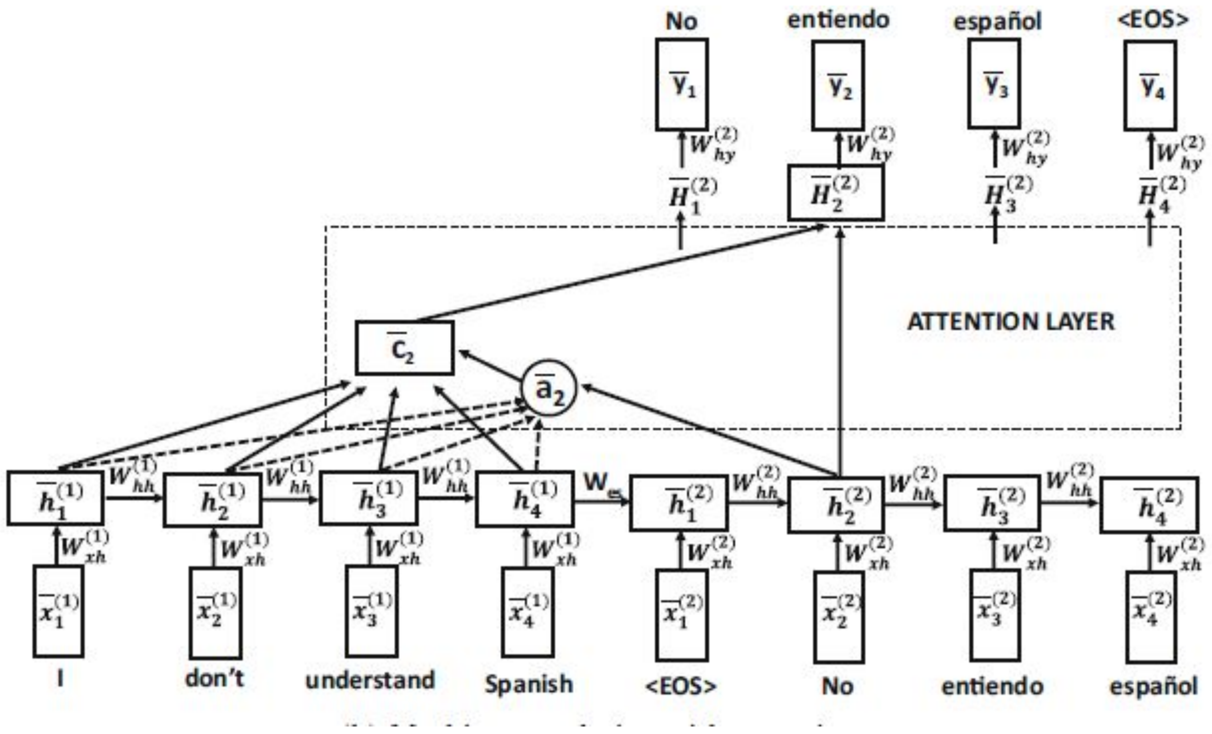
- Decoder: Using the context vector generated by the attention mechanism, the decoder produces the word "Le".

- Now, for the next word, "chat" (which is "cat" in French), the attention mechanism will compute a new set of attention weights.

- This time, the attention mechanism might give more weight to the second encoder state (corresponding to "cat") and less weight to the other states.

Attention weights for “cat" = [0.7, 0.9, 0.05, 0.05, 0.1]

Attention Mechanisms for Machine Translation



Machine translation with attention

Neural Turing Machines

- A **Neural Turing Machine (NTM)** is a type of neural network model introduced by **Google DeepMind** in 2014 that combines the power of neural networks with an external memory bank, mimicking some aspects of a traditional Turing machine.

- It can read from and write to this memory, allowing it to solve tasks that require learning to manipulate and store information over long durations.

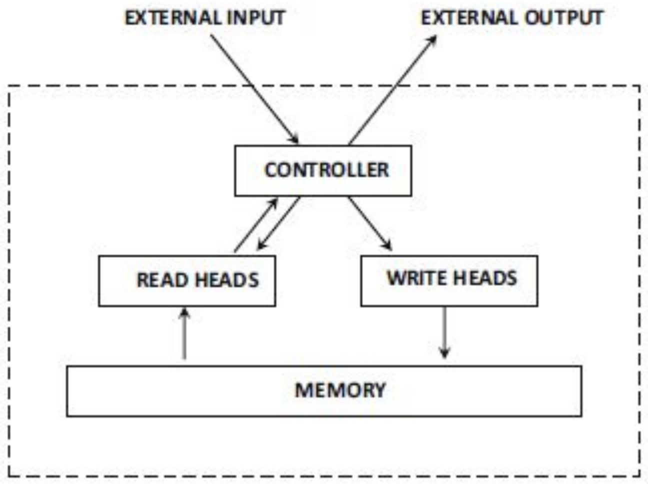
- It has following components : Controller, Read Heads, Write Heads and Memory

Neural Turing Machines

Controller (Neural Network):

The controller is typically a recurrent neural network (RNN) (such as an LSTM or GRU) that processes input and produces output.

The controller decides how to interact with the external memory based on the current input, its previous state, and the current memory content.



The neural Turing machine

Read/Write Heads:

Read head(s): These allow the controller to read from specific memory locations, using attention mechanisms to determine which parts of memory to access.
Write head(s): These are responsible for writing information to specific memory locations.

External Memory Matrix:

The memory is a large matrix where each row is a memory "slot" and the entire matrix holds information.
The network

Generative Adversarial Networks (GANs)

- **What are GANs?**
- GANs are like two **smart agents (models) competing with each other to create realistic things**,
- like fake images or music that look or sound real.

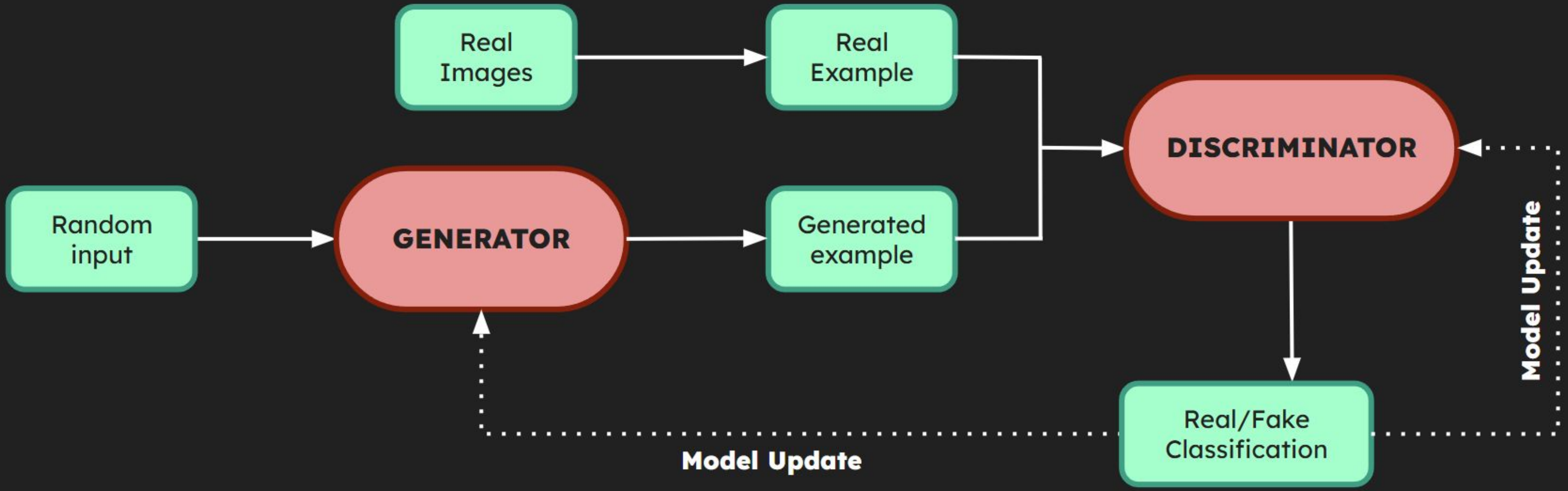
- **The Two Players:**
- **Generator:** Think of it as an artist who tries to create fake paintings that look real.
- **Discriminator:** Think of it as an art critic who tries to figure out if a painting is real or fake.

- **How They Work Together:**
- The **Generator** makes a fake image.
- The **Discriminator** looks at both real and fake images and guesses which ones are fake.
- The **Generator** tries to improve so it can fool the **Discriminator** better next time.

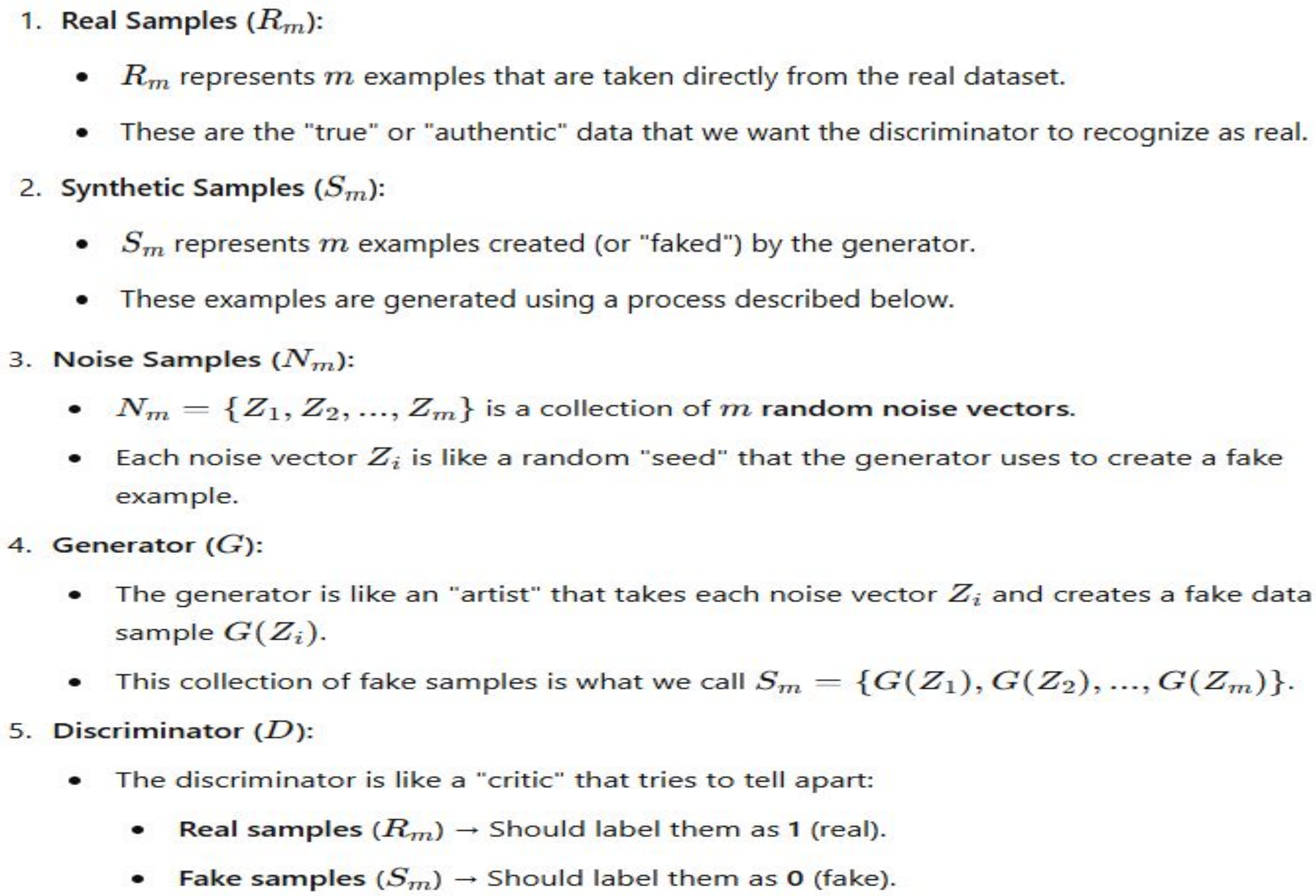
What are GANs?

Generative Adversarial Networks (GANs) consist of two neural networks (Generator and Discriminator) trained in opposition to each other.

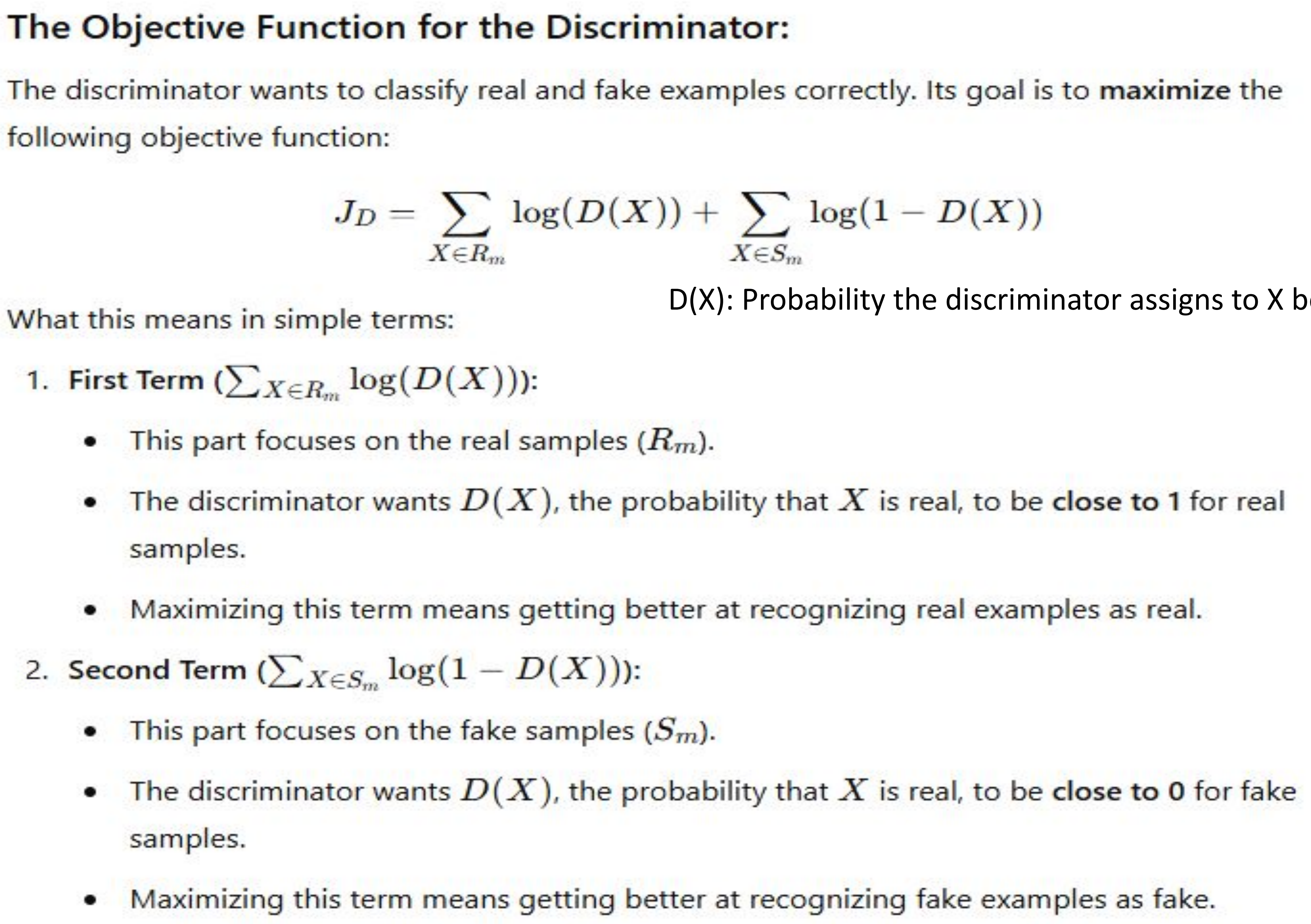
- **Generator:** Creates fake images.
- **Discriminator:** Differentiates between real and fake images.



Training a GAN



1. **Real Samples (R_m):**
   - R_m represents m examples that are taken directly from the real dataset.
   - These are the "true" or "authentic" data that we want the discriminator to recognize as real.
2. **Synthetic Samples (S_m):**
   - S_m represents m examples created (or "faked") by the generator.
   - These examples are generated using a process described below.
3. **Noise Samples (N_m):**
   - $N_m = \{Z_1, Z_2, ..., Z_m\}$ is a collection of m **random noise vectors**.
   - Each noise vector Z_i is like a random "seed" that the generator uses to create a fake example.
4. **Generator (G):**
   - The generator is like an "artist" that takes each noise vector Z_i and creates a fake data sample $G(Z_i)$.
   - This collection of fake samples is what we call $S_m = \{G(Z_1), G(Z_2), ..., G(Z_m)\}$.
5. **Discriminator (D):**
   - The discriminator is like a "critic" that tries to tell apart:
     - **Real samples** (R_m) → Should label them as **1** (real).
     - **Fake samples** (S_m) → Should label them as **0** (fake).



The Objective Function for the Discriminator:

The discriminator wants to classify real and fake examples correctly. Its goal is to **maximize** the following objective function:

$$J_D = \sum_{X \in R_m} \log(D(X)) + \sum_{X \in S_m} \log(1 - D(X))$$

D(X): Probability the discriminator assigns to X being real.

What this means in simple terms:

1. **First Term** ($\sum_{X \in R_m} \log(D(X))$):
   - This part focuses on the real samples (R_m).
   - The discriminator wants $D(X)$, the probability that X is real, to be **close to 1** for real samples.
   - Maximizing this term means getting better at recognizing real examples as real.
2. **Second Term** ($\sum_{X \in S_m} \log(1 - D(X))$):
   - This part focuses on the fake samples (S_m).
   - The discriminator wants $D(X)$, the probability that X is real, to be **close to 0** for fake samples.
   - Maximizing this term means getting better at recognizing fake examples as fake.

Objective Function for the Generator:

The generator's **objective function** J_G is:

$$J_G = \sum_{X \in S_m} \log(1 - D(X))$$

What this means:

1. $\log(1 - D(X))$:
   - For each fake sample $X \in S_m$, $D(X)$ is the discriminator's prediction that X is **real**.
   - $1 - D(X)$ is the discriminator's prediction that X is **fake**.
   - The generator wants $D(X)$ to be close to 1, which would make $1 - D(X)$ close to **0**. This would minimize the objective function.
2. **Minimizing** J_G:
   - The generator adjusts its parameters to make the discriminator "believe" that the fake samples are real.
   - This means $D(X)$, the discriminator's prediction for fake samples, gets closer to **1**.

GANs in Image Data

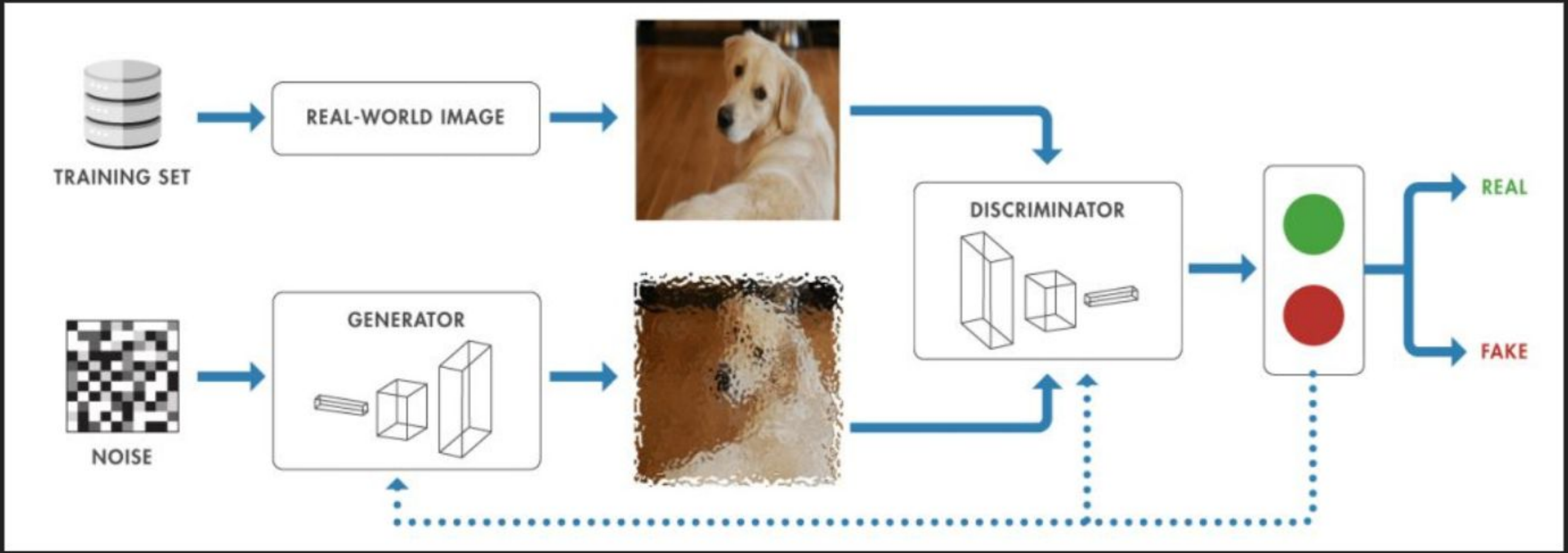
The image domain is one of the most common applications for GANs due to the richness and complexity of visual data.

Key Characteristics:

- Ability to capture and replicate intricate details and textures.
- Capability to create realistic images that resemble the training dataset.

Deconvolutional Networks (**DCGANs**) are used for generating images.

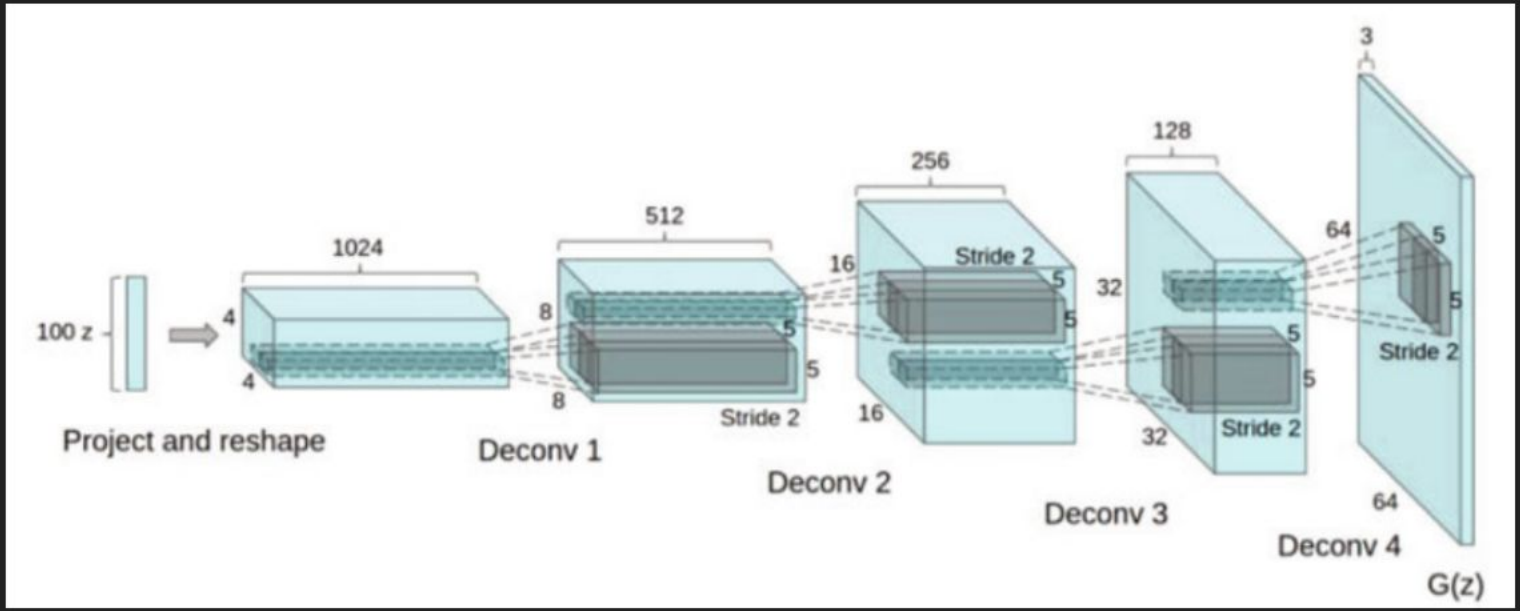
Impact: GANs are widely used for applications such as data augmentation, deepfake image creation, artistic content creation etc.



DCGAN Architecture

What is DCGAN?

Deep Convolutional GAN (DCGAN) is a specialized GAN architecture for image generation.



Input: Starts with 100-dimensional Gaussian noise.

Reshaping: Noise is reshaped into feature maps.

Layers:

- Feature map depth reduces gradually.
- Spatial dimensions (length & width) increase.

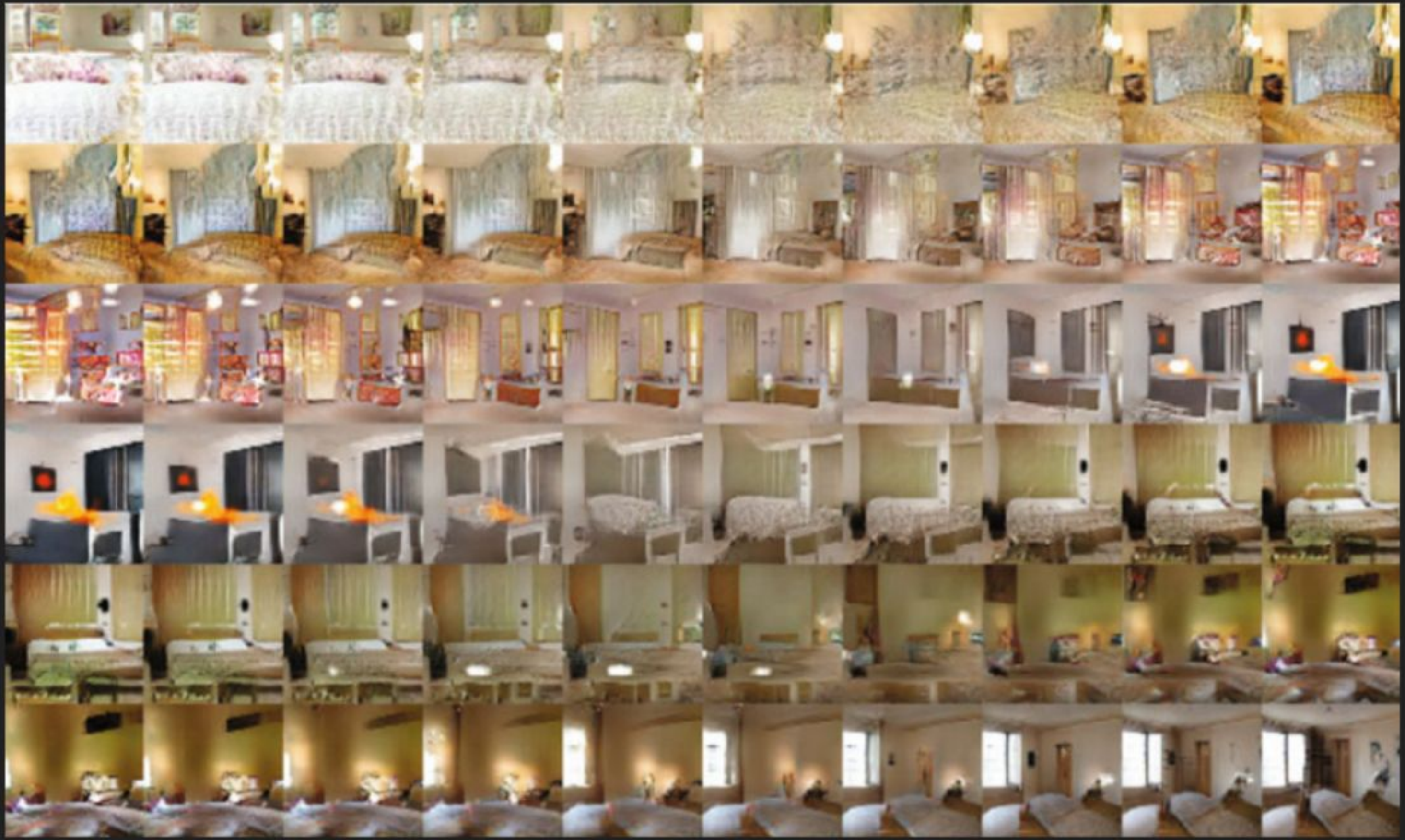
Upscaling:

- Achieved using **fractionally strided convolutions.**
- No need for pooling layers.

Image Generation Examples

The generated images are sensitive to noise samples.

Example 1:
A room without a window is gradually transformed into one with a large window

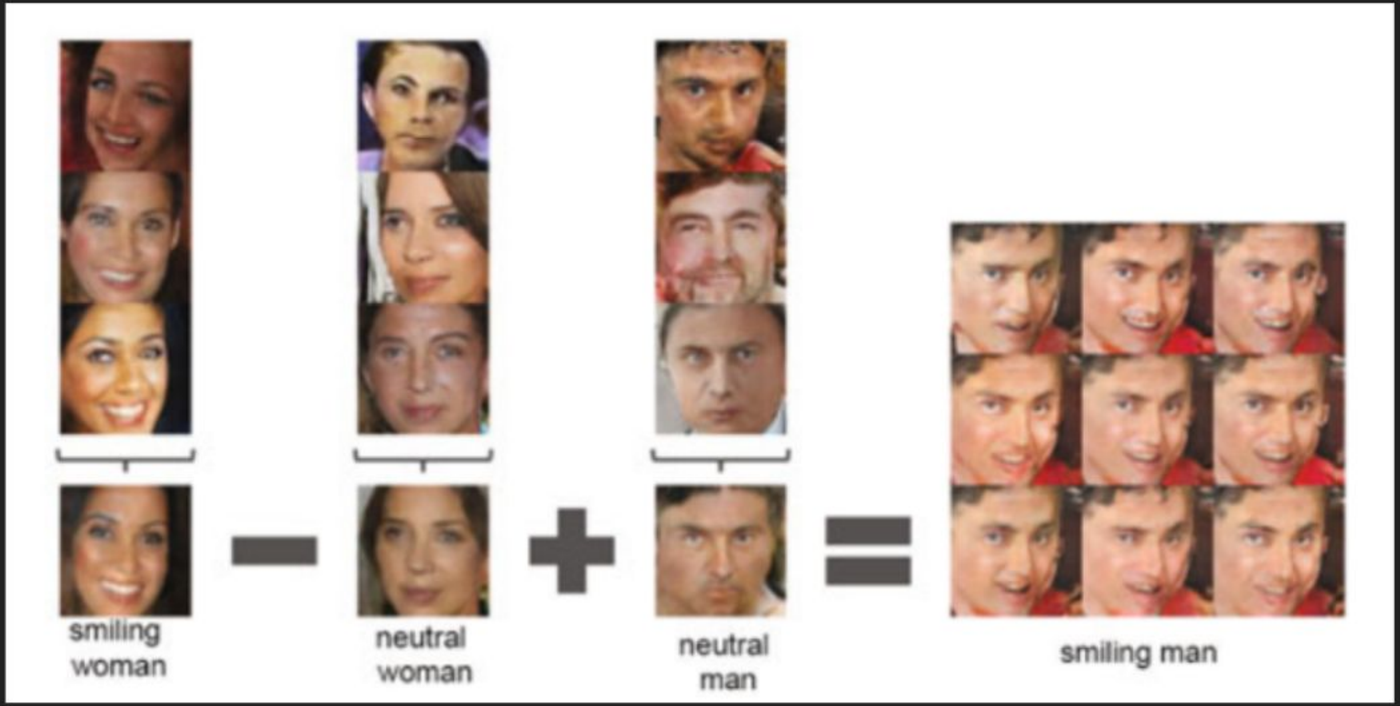


Smooth image transitions caused by changing input noise are shown in each row

Image Generation Examples

Operations on noise vectors yield meaningful outcomes.

Example 2:
(Smiling Woman - Neutral Woman) + Neutral Man = Smiling Man



Arithmetic operations on input noise have semantic significance

Discriminator Architecture

- Uses a CNN architecture
- **Leaky ReLU** is used instead of ReLU for distinguishing real from fake images.
- The final convolutional layer is flattened and fed into a **single sigmoid output for binary classification.**
- Fully connected layers are not used.
- **Batch normalization** is done to mitigate any issues related to the vanishing and exploding gradient problems.

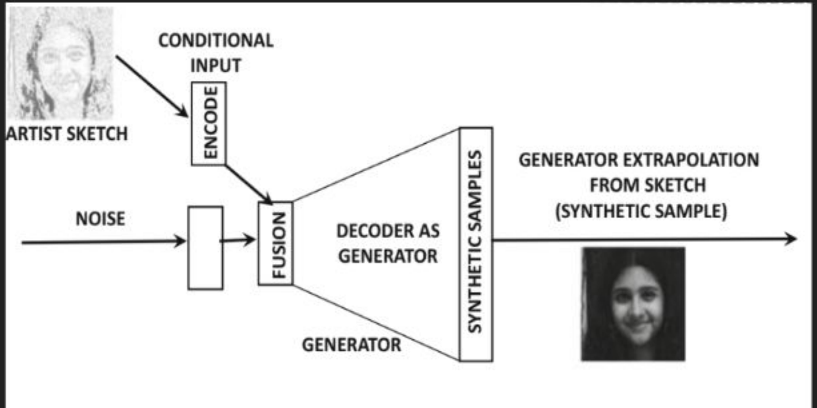
Conditional GANs

In a cGAN, both the generator and discriminator are conditioned on **external information**, such as labels or text.

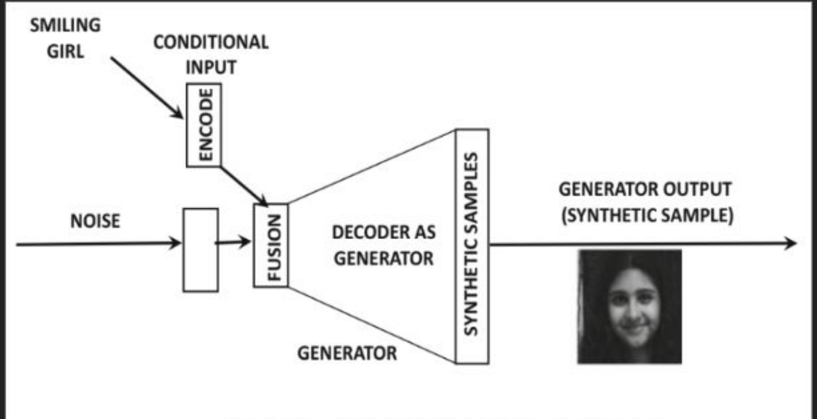
They enable creativity in image synthesis, allowing for more control over the generated images or data.

Applications:

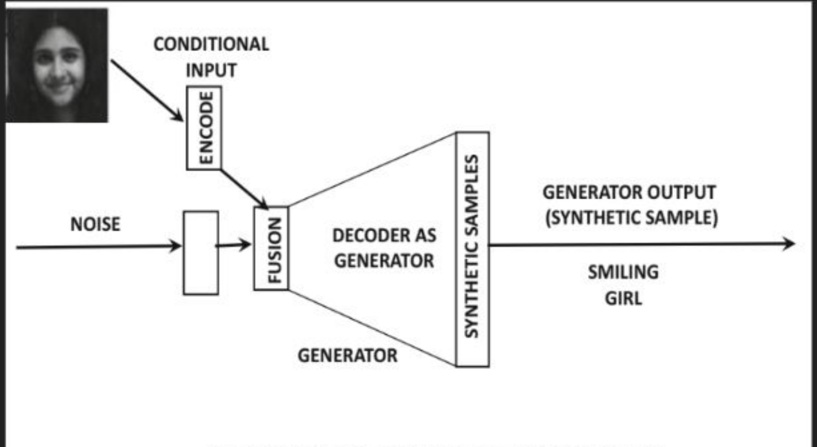
- Image-to-image translation (e.g., sketches to photorealistic images).
- Text-to-image generation (e.g., "smiling girl" description to image).
- Image-to-text conversion.



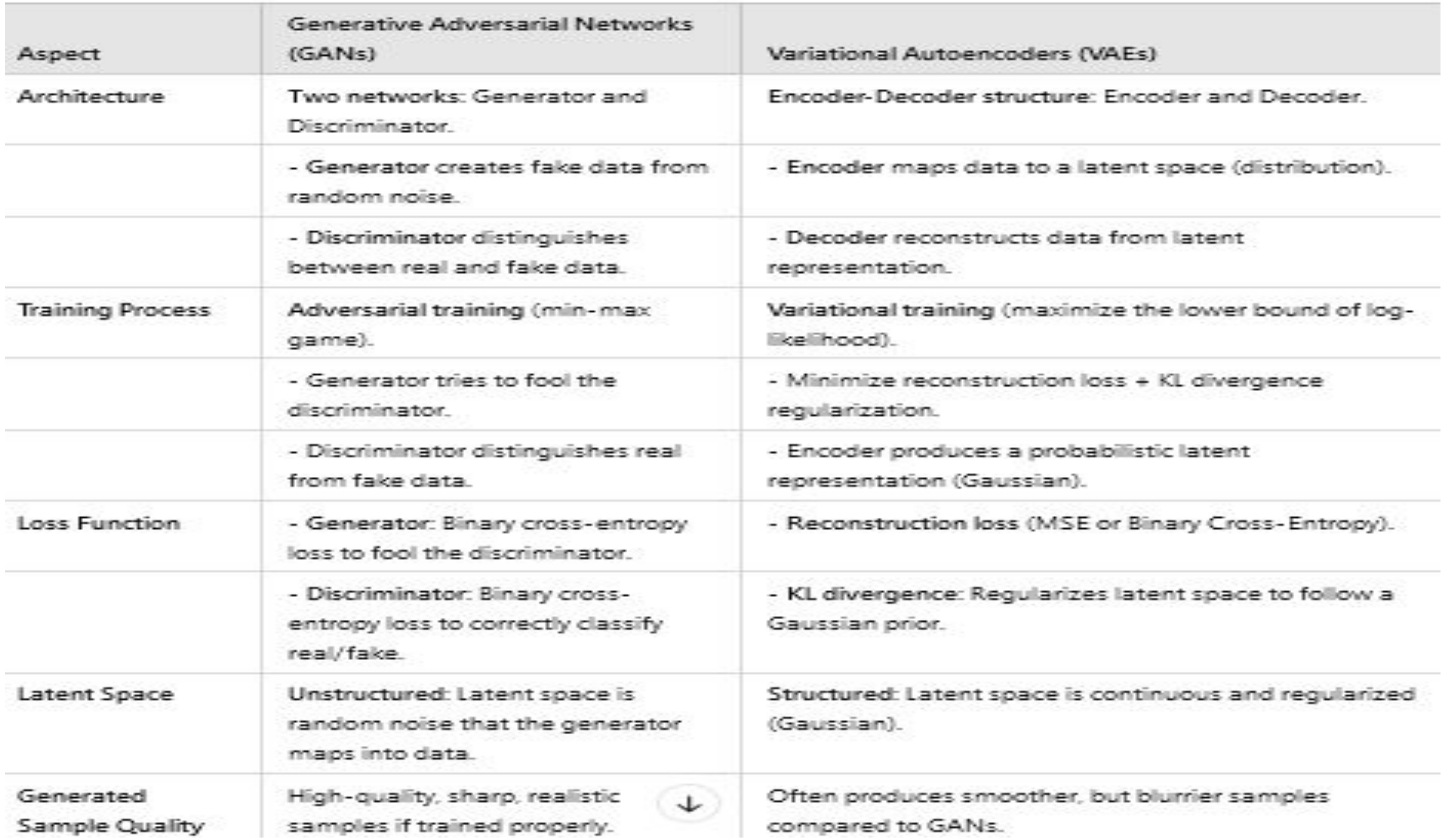
CONDITIONAL GANs FOR IMAGE-TO-IMAGE TRANSLATION



CONDITIONAL GANs FOR TEXT-TO-IMAGE TRANSLATION

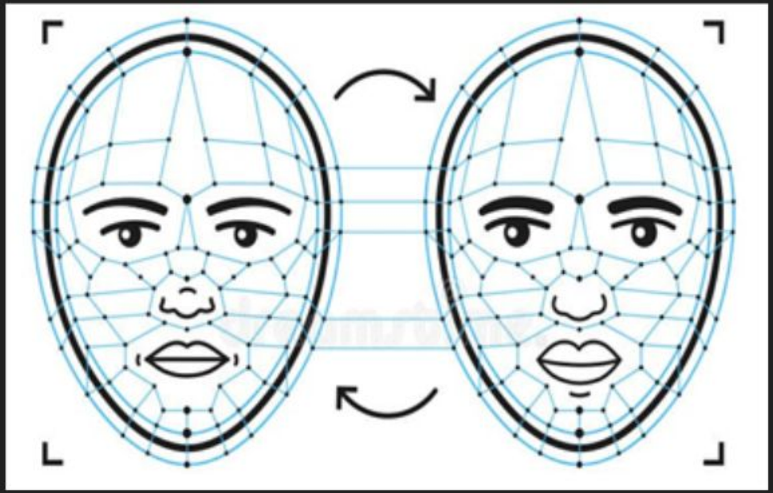


CONDITIONAL GANs FOR IMAGE-TO-TEXT TRANSLATION conditional GANs.



Aspect	Generative Adversarial Networks (GANs)	Variational Autoencoders (VAEs)
Architecture	Two networks: Generator and Discriminator.	Encoder-Decoder structure: Encoder and Decoder.
	- Generator creates fake data from random noise.	- Encoder maps data to a latent space (distribution).
	- Discriminator distinguishes between real and fake data.	- Decoder reconstructs data from latent representation.
Training Process	**Adversarial training** (min-max game).	**Variational training** (maximize the lower bound of log-likelihood).
	- Generator tries to fool the discriminator.	- Minimize reconstruction loss + KL divergence regularization.
	- Discriminator distinguishes real from fake data.	- Encoder produces a probabilistic latent representation (Gaussian).
Loss Function	- **Generator**: Binary cross-entropy loss to fool the discriminator.	- **Reconstruction loss** (MSE or Binary Cross-Entropy).
	- **Discriminator**: Binary cross-entropy loss to correctly classify real/fake.	- **KL divergence**: Regularizes latent space to follow a Gaussian prior.
Latent Space	**Unstructured**: Latent space is random noise that the generator maps into data.	**Structured**: Latent space is continuous and regularized (Gaussian).
Generated Sample Quality	High-quality, sharp, realistic samples if trained properly.	Often produces smoother, but blurrier samples compared to GANs.

Case Study: GANs and Deepfake Image Creation



Deepfake Original

What are Deepfakes?

Synthetic media created using GANs to generate realistic images or videos that mimic real people, often by replacing a person in a picture or video with someone else in a highly convincing manner.

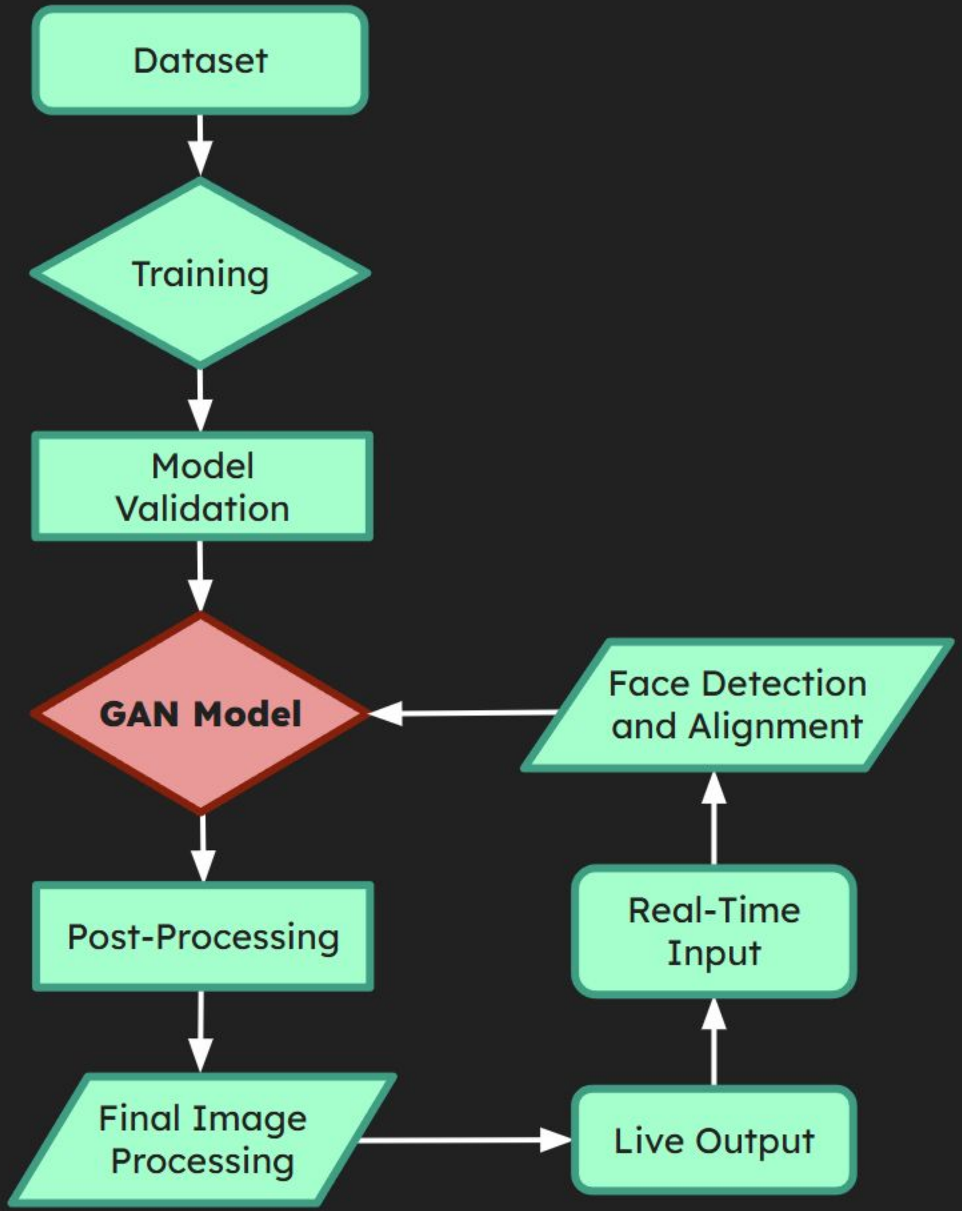
Applications:

- Entertainment
- Movie effects
- Creating digital avatars

Disadvantages:

- Ethical concerns due to misinformation and misuse in fraud
- Impersonation
- False evidence

Case Study: GANs and Deepfake Image Creation



Usage of GANs in deepfake technology:

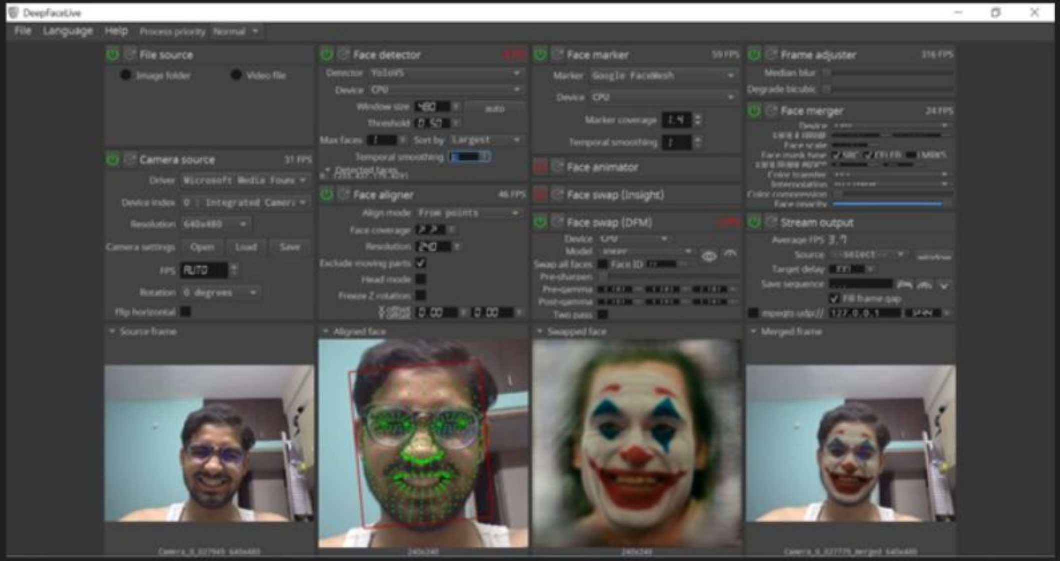
- DCGANs: Provide high-quality image generation by refining visual details.
- CGANs: Allow for conditioning on specific inputs (facial expressions, voices) to create contextually accurate outputs.

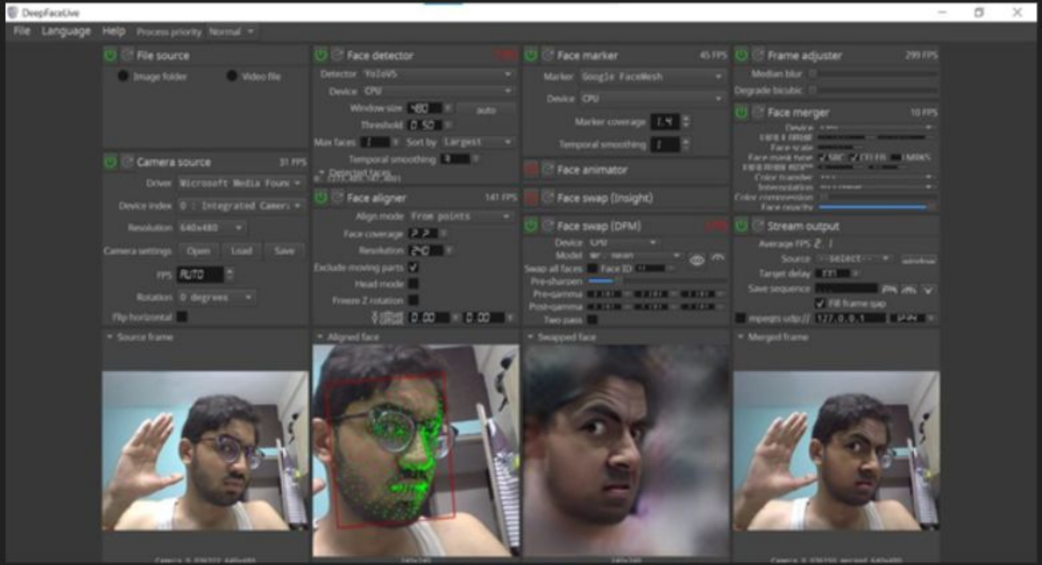
Autoencoders:

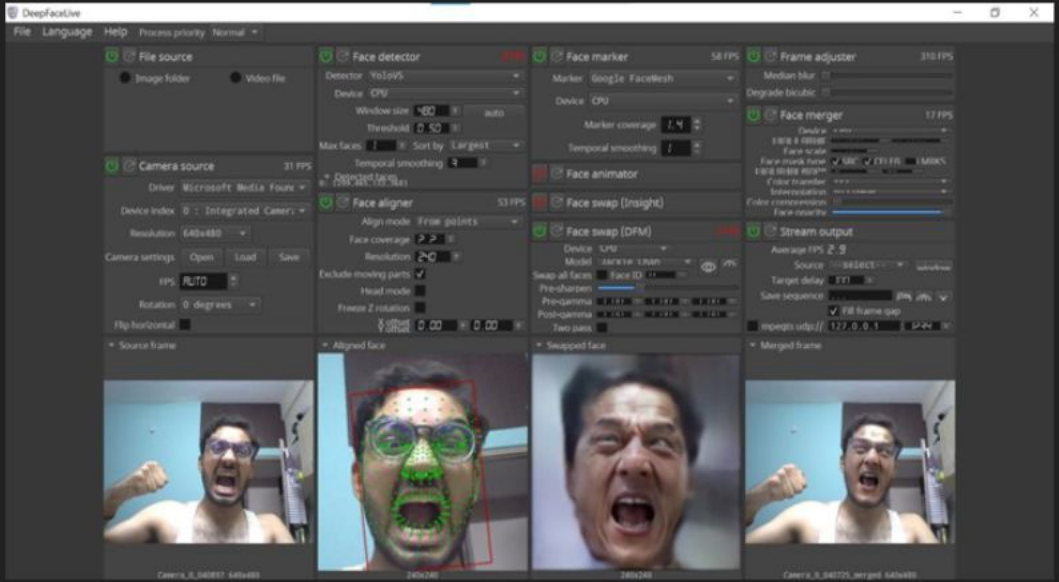
A type of neural network architecture

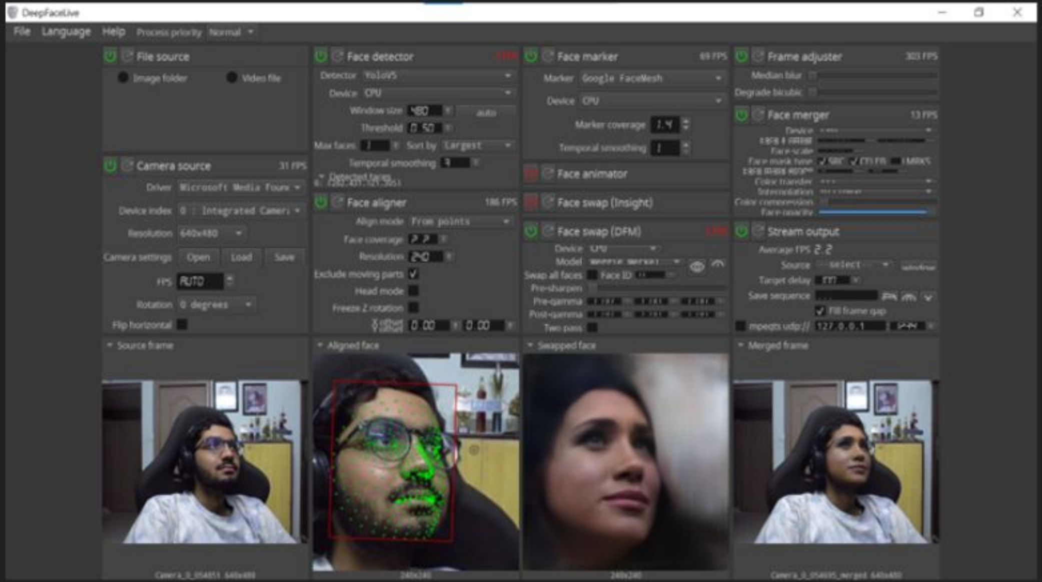
- Encoder: Maps input data to a lower-dimensional latent space
- Decoder: Reconstructs the input data from the latent space.

Sample Deepfake Results Using DeepFaceLive Software


Example 1: Joker


Example 2: Mr Bean


Example 3: Jackie Chan


Example 4: Meghan Markle

THANK YOU