Drug Discovery

- What is Drug Discovery?

Drug Discovery is the process of identifying new candidate medications. It involves several stages aimed at discovering compounds that can interact with biological targets to treat diseases effectively.

-Problem Statement

The process of drug discovery involves finding new molecules that can interact with target proteins to treat diseases. Traditional methods are slow and expensive, often relying on extensive trial and error. This project aims to use machine learning, specifically Variational Autoencoders (VAEs), to generate new molecules in SMILES format. By training the VAE on molecular data, we hope to streamline the discovery of potential drugs by efficiently predicting and designing novel molecular structures.

-Abstract:-

Drug discovery is a costly and time-consuming process that seeks molecules capable of interacting with disease-related proteins. To improve this process, we propose using a Variational Autoencoder (VAE) to generate novel molecules represented in SMILES format. By training the VAE on the ZINC dataset/sample dataset , the model will learn to encode and decode molecular structures, facilitating the design of new drug candidates. This AI-driven approach aims to make drug discovery more efficient by quickly generating and evaluating potential therapeutic compounds.

Steps in Drug Discovery:

1. Target Identification and Validation
2. Hit Identification
3. Hit-to-Lead
4. Lead Optimization
5. Preclinical Testing
6. Clinical Trials
7. Regulatory Approval
8. Post-Market Surveillance

- Our Focus is mainly of the first 4 Steps (Technical Steps)

Key Terms in Drug Discovery:

1. **Target Molecule**: A biological molecule (often a protein) that is implicated in a disease and can be modulated by a drug to produce a therapeutic effect.
2. **Ligand**: A molecule that binds to a target molecule, such as a drug binding to a protein, to produce a biological response.
3. **Candidates**: Compounds that show promise in early testing stages and are selected for further development.
4. **Molecule**: A general term for any chemical compound, including drugs, that may interact with biological systems.
5. **Protein**: Large, complex molecules made up of amino acids that perform many critical functions in the body. Proteins are common targets in drug discovery.
6. **Polymer**: Large molecules made up of repeated subunits. In drug discovery, polymers can be used in drug delivery systems.
7. **Genome**: The complete set of DNA in an organism, containing all the genetic information. Genomic data is crucial for identifying drug targets and developing personalized medicines.

Role of AI and Computer Science in Drug Discovery:

- AI and computer science (CSE) have revolutionized drug discovery by accelerating various stages and increasing efficiency. Key applications include:

1. Predictive Modeling
2. Molecular Docking
3. Virtual Screening
4. De Novo Drug Design
5. Data Mining
6. Genomics and Personalized Medicine

Integrating AI and CSE in Drug Discovery:

1. **Target Identification**: AI analyzes genetic and proteomic data to identify potential targets.
2. **Hit Identification**: Virtual screening with AI to find potential hit compounds from large libraries.
3. **Lead Optimization**: Machine learning models predict how changes to lead compounds will affect their properties, speeding up optimization.
4. **Clinical Trials**: AI models predict patient responses and identify potential side effects, optimizing trial design and patient selection.

SMILES-(Simplified Molecular Input Line Entry System)

- It is a notation that allows a user to represent a chemical structure in a way that can be used by a computer.
- **Atoms**: Represented by their atomic symbols (‘**C**’)
- **Bonds**:
  - Single bonds are implicit ex:-Ethane: **CC**
  - double bonds are represented by **‘=‘** ex:-Ethylene: **C=C**
  - triple bonds by **‘#’** ex:-Acetylene: **C#C**
  - and aromatic bonds by **‘:’**
- **Branches**: Represented by parentheses. Ex:- isobutane is **CC(C)C**.
- **Rings**: Represented by numbers that indicate the start and end of a ring. Ex:- cyclohexane is **C1CCCCC1**.
- **Aromaticity**: Represented by lowercase letters. Ex:-benzene is **c1ccccc1**.
- **Charged atoms**: Represented by brackets with the charge indicated. Ex:- the ammonium ion is **[NH4+]**

Other Examples:-

Aspartic Acid: `NC(CCC(=O)O)C(=O)O`

Caffeine: `CN1C=NC2=C1C(=O)N(C(=O)N2C)C`

Variational Autoencoders (VAEs)

- Latent Space Representation.
- Optimization
- Diverse Molecular Generation
- Smooth Interpolation
- Property Prediction

Generative Adversarial Networks (GANs)

- High-Quality Molecule Generation
- Adversarial Training
- Customization
- Data Augmentation

Combined Benefits

- Accelerated Discovery Process
- Integration with Existing Pipelines
- Cost-Effective
- Scalability

How VAE's are integrated in DD:

- Variational Auto Encoders (VAEs) have shown significant promise in the field of drug discovery due to their ability to model complex data distributions and generate new molecular structures.
- **Overview of VAEs:-**
  - VAEs are a type of generative model that learn to encode input data into a latent space and then decode it back to the original space.
  - They consist of an encoder, which maps input data to a latent space (Bottle Neck) and a decoder, which reconstructs the data from the latent space.
  - VAEs are trained to replicate the given input at the output with high similarity. The output can be improvised if required.

- Reference: “Auto-Encoding Variational Bayes” by Kingma and Welling
- **Applications in Drug Discovery:**
    - VAEs can generate new molecular structures by learning the distribution of known molecules and sampling from the latent space.
    - They can help identify novel compounds with desired properties, optimizing drug-like molecules.
- **Tools and Libraries:**
  - **DeepChem:** Deep learning Library
  - **RDKit:** Useful for cheminformatics
  - **Jupyter Notebooks and Tutorials:** Jupyter notebooks demonstrate the application of VAEs in generating molecules.

➢ Practical Implementation:

- **Step-by-Step Guides:**
  - Follow guides that show how to preprocess molecular data, train a VAE, and generate new molecules.
  - Example guide: "Generating molecules using Variational Autoencoders" available on GitHub repositories.
- **Datasets:**

  ChEMBL: A large database of bioactive molecules with drug-like properties.

  ZINC Database: Contains commercially available compounds for virtual screening.

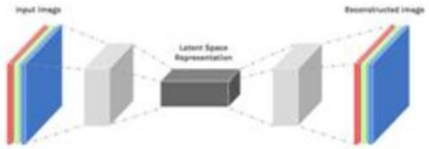
➢ Fundamentals of Drug Discovery

- **Drug Discovery Process:** Understanding the stages from target identification, validation, hit discovery, lead optimization, to preclinical and clinical testing.
- **Pharmacokinetics and Pharmacodynamics (PK/PD):** Basics of how drugs act in the body and their biological effects.
- **Molecular Biology and Chemistry:** Basic principles that underpin the interaction between drugs and their targets.

Autoencoders

What are Autoencoders

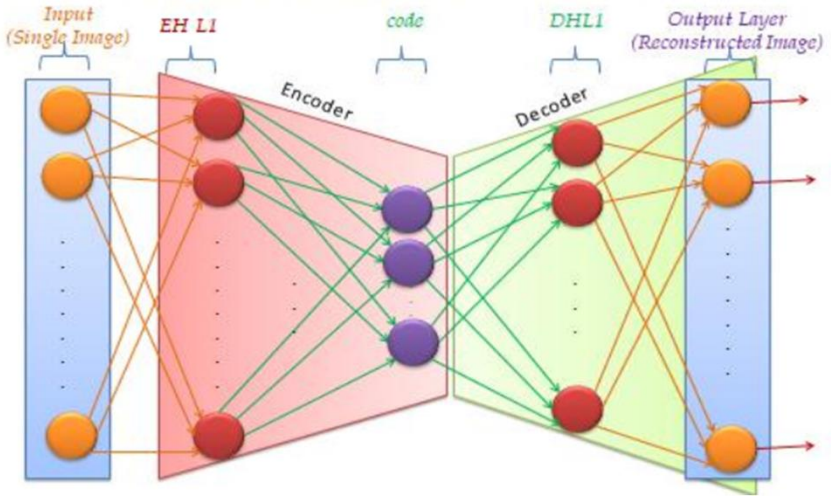
Unsupervised machine learning algorithm that applies back propagation, setting the target values to be equal to the inputs.



Used to reduce the size of our inputs into a smaller representation.

If anyone needs the original data, they can reconstruct it from the compressed data.

Implementing Autoencoder



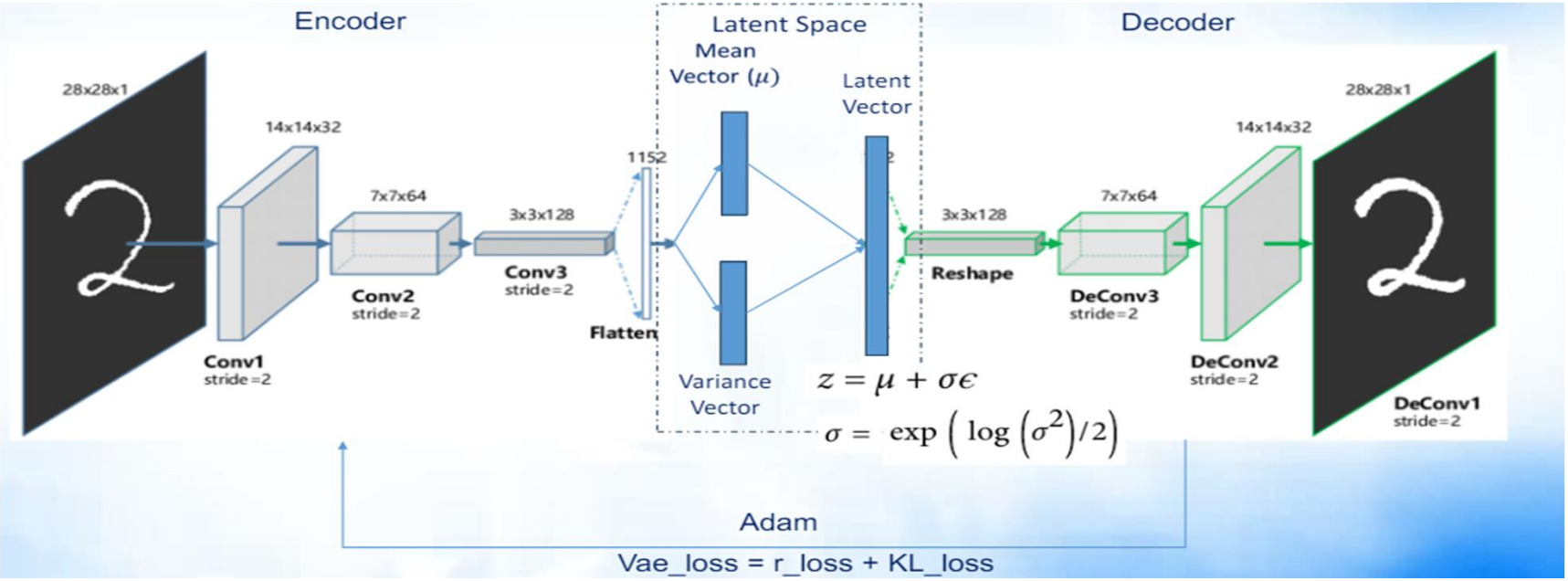
Types of Autoencoders

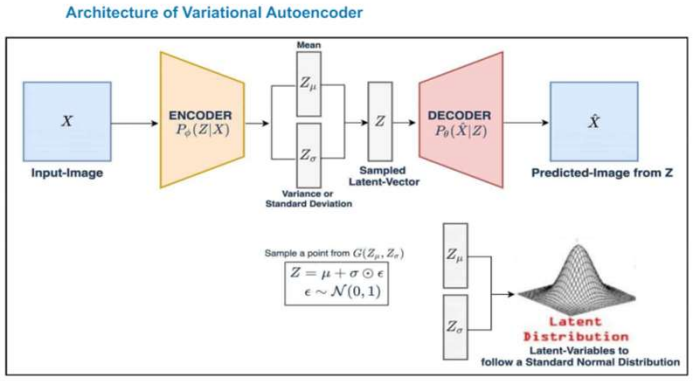
- Vanilla/Base Autoencoder
- Convolutional Autoencoder (CAE)
- Denoising Autoencoder
- Seq2Seq or Recurrent Autoencoder
- Variational Autoencoder (VAE)

Limitations of Autoencoder

- Deterministic Encoding
- No Probabilistic Framework
- Difficulty in Interpolation
- Limited use in Generative Modelling

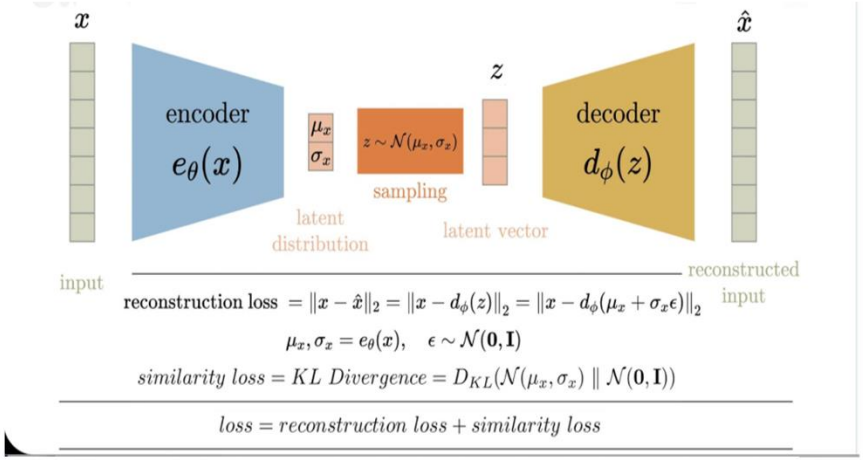
Variational Autoencoders (VAEs)



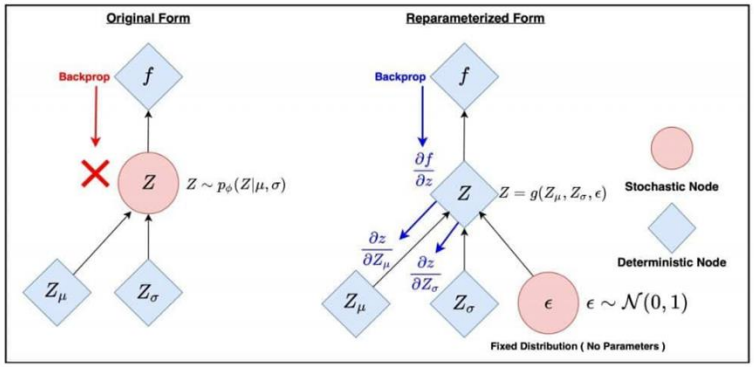


Latent Space Representation.
Optimization
Diverse Molecular Generation
Smooth Interpolation
Property Prediction

VAE Architecture



Reparameterization Trick




```
[8]  # Define the Encoder
     latent_dim = 2
     input_shape = (max_len, len(char_to_index))

     encoder_inputs = layers.Input(shape=input_shape)
     x = layers.Conv1D(128, 5, activation='relu', padding='same')(encoder_inputs)
     x = layers.Conv1D(128, 5, activation='relu', padding='same')(x)
     x = layers.Flatten()(x)
     x = layers.Dense(256, activation='relu')(x)
     z_mean = layers.Dense(latent_dim, name='z_mean')(x)
     z_log_var = layers.Dense(latent_dim, name='z_log_var')(x)

     class Sampling(layers.Layer):
         def call(self, inputs):
             z_mean, z_log_var = inputs
             batch = tf.shape(z_mean)[0]
             dim = tf.shape(z_mean)[1]
             epsilon = tf.random.normal(shape=(batch, dim))
             return z_mean + tf.exp(0.5 * z_log_var) * epsilon

     z = Sampling()([z_mean, z_log_var])
     encoder = models.Model(encoder_inputs, [z_mean, z_log_var, z], name='encoder')
```

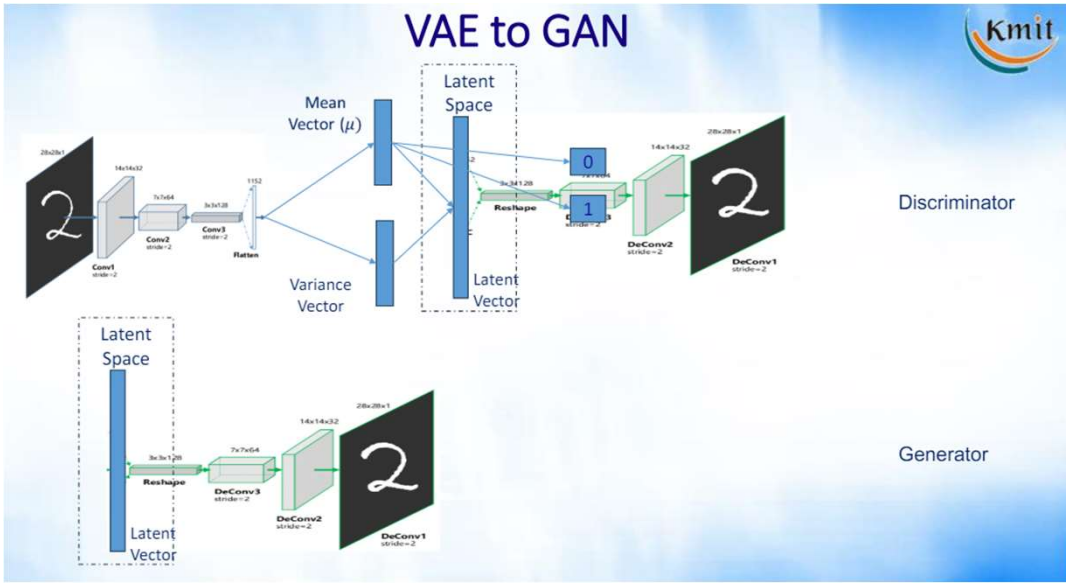
```
[9]  # Define the Decoder
     latent_inputs = layers.Input(shape=(latent_dim,))
     x = layers.Dense(256, activation='relu')(latent_inputs)
     x = layers.Dense(max_len * 128, activation='relu')(x)
     x = layers.Reshape((max_len, 128))(x)
     x = layers.Conv1DTranspose(128, 5, activation='relu', padding='same')(x)
     x = layers.Conv1DTranspose(128, 5, activation='relu', padding='same')(x)
     decoder_outputs = layers.Conv1DTranspose(len(char_to_index), 5, activation='softmax', padding='same')(x)

     decoder = models.Model(latent_inputs, decoder_outputs, name='decoder')
```

Generative Adversarial Networks (GANs)

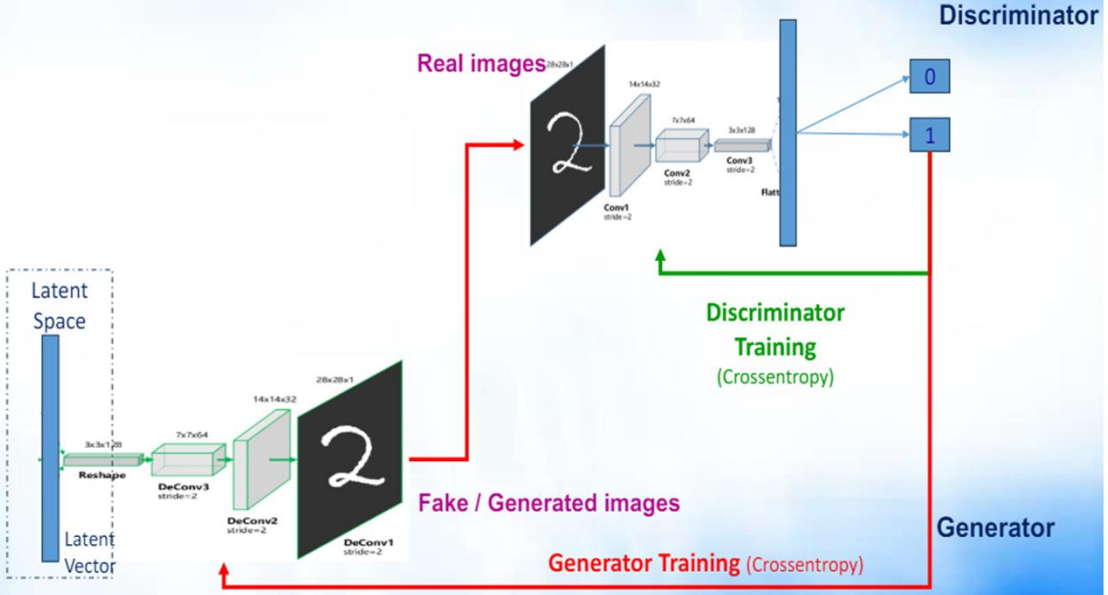
Generative Adversarial Networks (GANs) are a way of training a generative model by framing the problem as a supervised learning problem with two sub-network models:

- The first network generates data
- Second network tries to find the difference between the real data and the fake data generated by the first network.

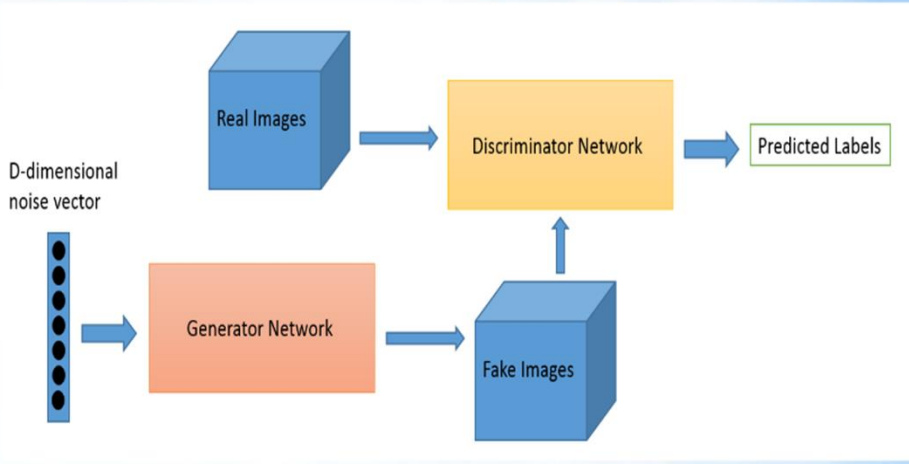


VAE to GAN

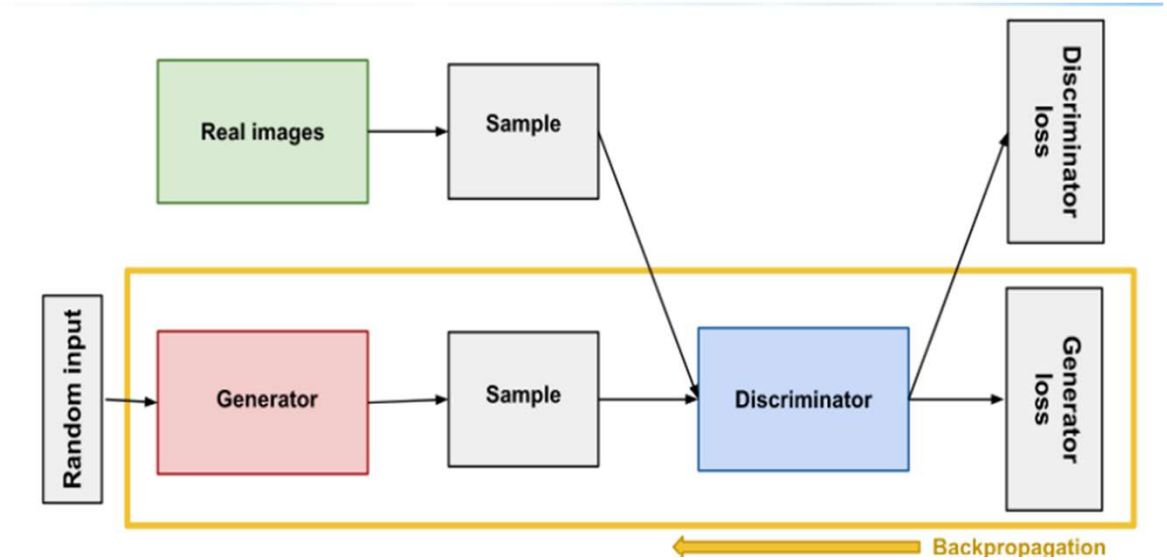
Generative Advarsarial Network



GAN Architecture



Generator

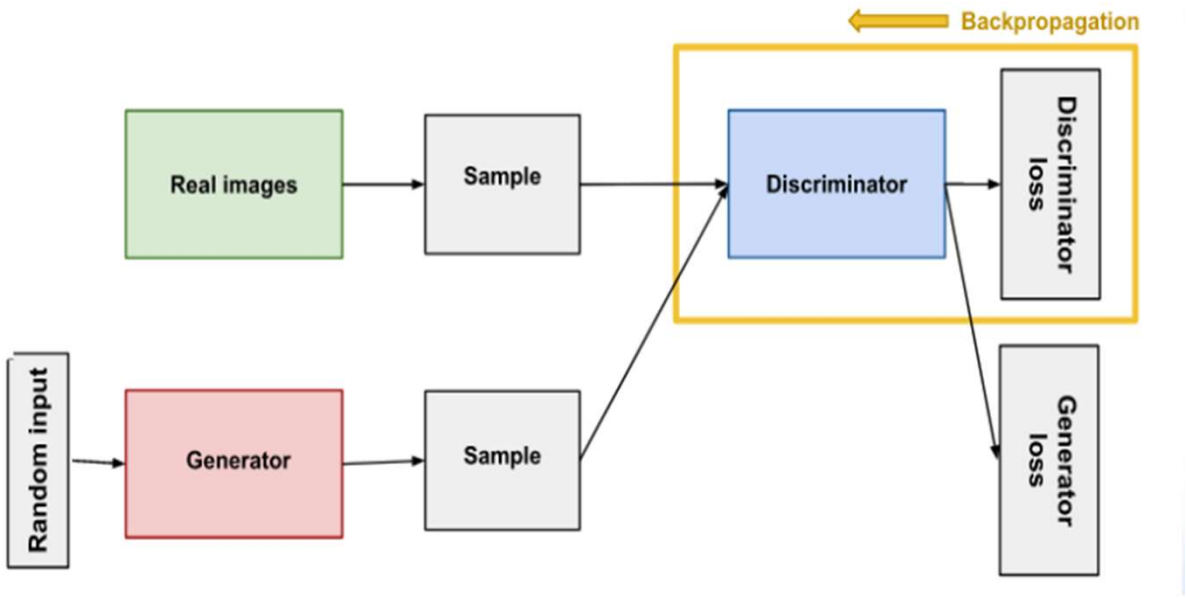



```
def gen_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(4*4*512, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization()) #batch normalization to do pre-processing at every layer of the network.
    model.add(layers.ReLU()) #ReLU is a linear (identity) for all positive values, and zero for all negative values.
    model.add(layers.Reshape((4, 4, 512)))
    model.add(layers.Conv2DTranspose(256, (4,4), strides=(2, 2), padding='same', use_bias=False))
    #(4,4) specifies the height and width of the 2D convolution window.
    #Strides = (2,2) specifies the strides of the convolution along the height and width.
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())

    model.add(layers.Conv2DTranspose(128, (4,4), strides=(2, 2), padding='same', use_bias=False))
    #(4,4) specifies the height and width of the 2D convolution window.
    #Strides = (2,2) specifies the strides of the convolution along the height and width.
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())

    model.add(layers.Conv2DTranspose(64, (4, 4), strides=(2, 2), padding='same', use_bias=False))
    #(4,4) specifies the height and width of the 2D convolution window.
    #Strides = (2,2) specifies the strides of the convolution along the height and width.
    model.add(layers.BatchNormalization())
    model.add(layers.ReLU())
    model.add(layers.Conv2DTranspose(3, (4, 4), strides=(2, 2), padding='same', use_bias=False, activation='sigmoid'))
    return model
```

Discriminator




```
def disc_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (4, 4), strides=(2, 2), padding='same',
                                    input_shape=[64, 64, 3]))
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.3)) #dropout consists in randomly setting

    model.add(layers.Conv2D(128, (4, 4), strides=(2, 2), padding='same'))
    model.add(layers.ReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model
```

Types of GANs

1. Standard GAN (SGAN)
2. Conditional GAN (cGAN)
3. Deep Convolutional GAN (DCGAN)
4. Wasserstein GAN (WGAN)
5. Variational Autoencoder GAN (VAE-GAN)
6. CycleGAN
7. DiscoGAN
8. Pix2Pix (Image-to-Image GAN)
9. StackGAN
10. BigGAN
11. Progressive GAN
12. Self-Attention GAN (SAGAN)
13. StyleGAN and StyleGAN2

Applications:

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Inpainting
- Clothing Translation
- Video Prediction

Thank You

Bhargav Manchala
KMEC CSE-A
245521733303
FS-A1