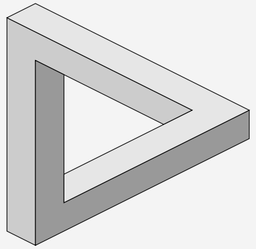
PClub Projects

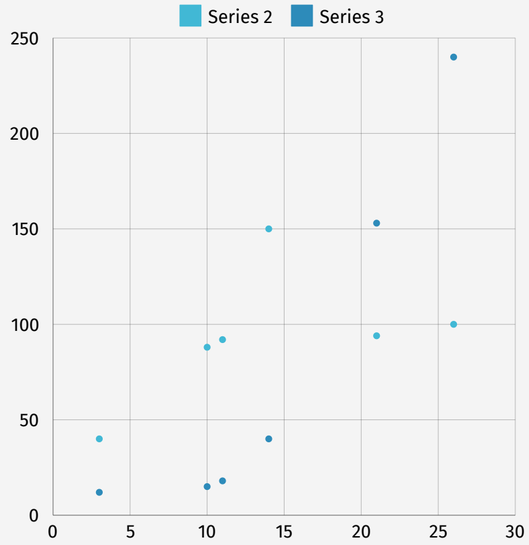
ILLUSIONCRAFT



Mid-eval Presentation

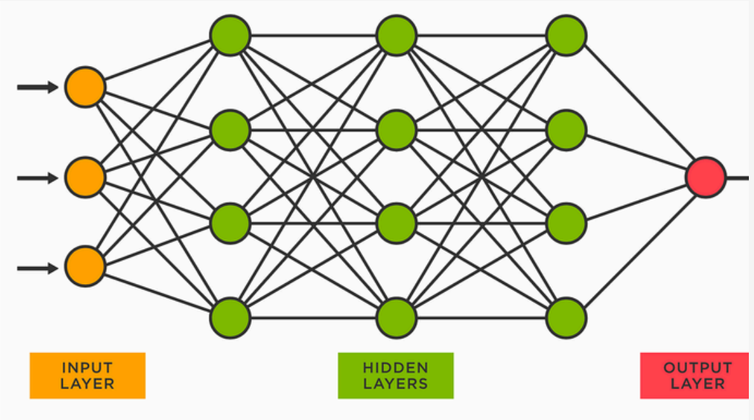
Contributors

Kshitij Gupta
Arihant Kumar
Durbasmriti Saha



Logistic Regression

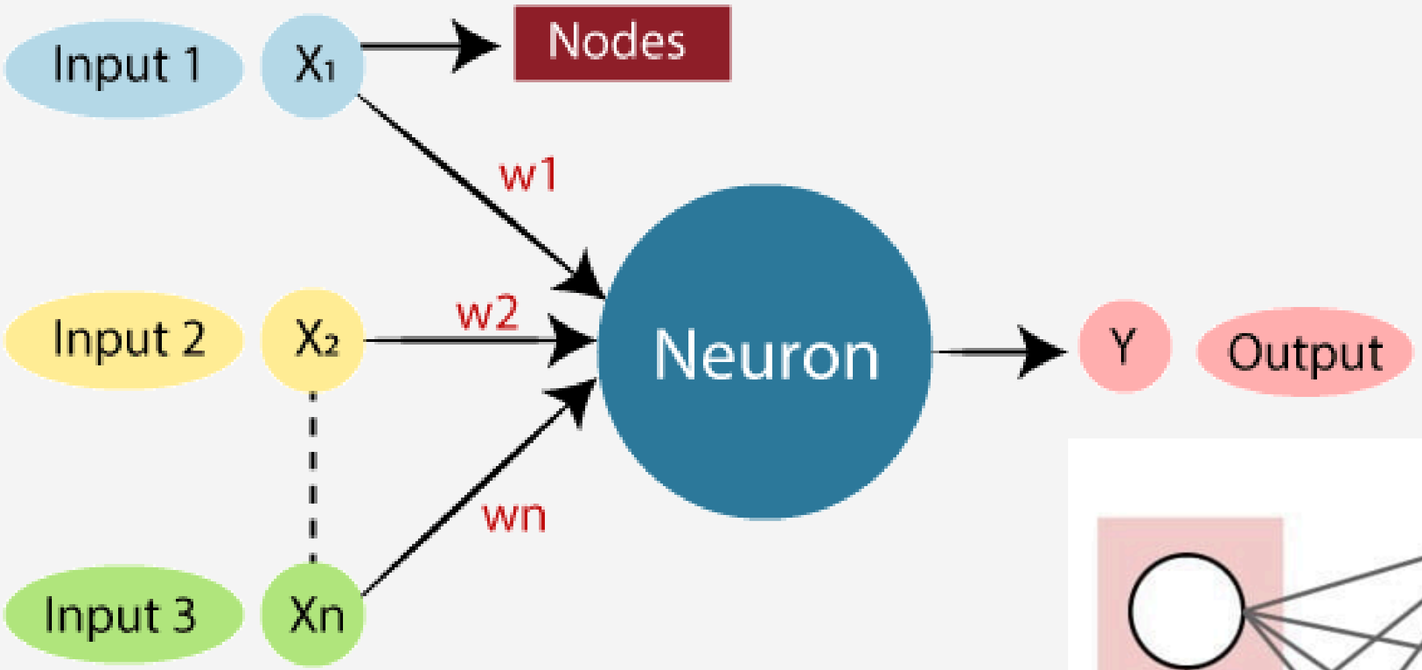
Logistic Regression is a **supervised machine learning algorithm** used for **classification** tasks where the goal is to predict the probability that an instance belongs to a given class or not. It uses **sigmoid and softmax function** that takes input as independent variables and produces a probability value between 0 and 1.



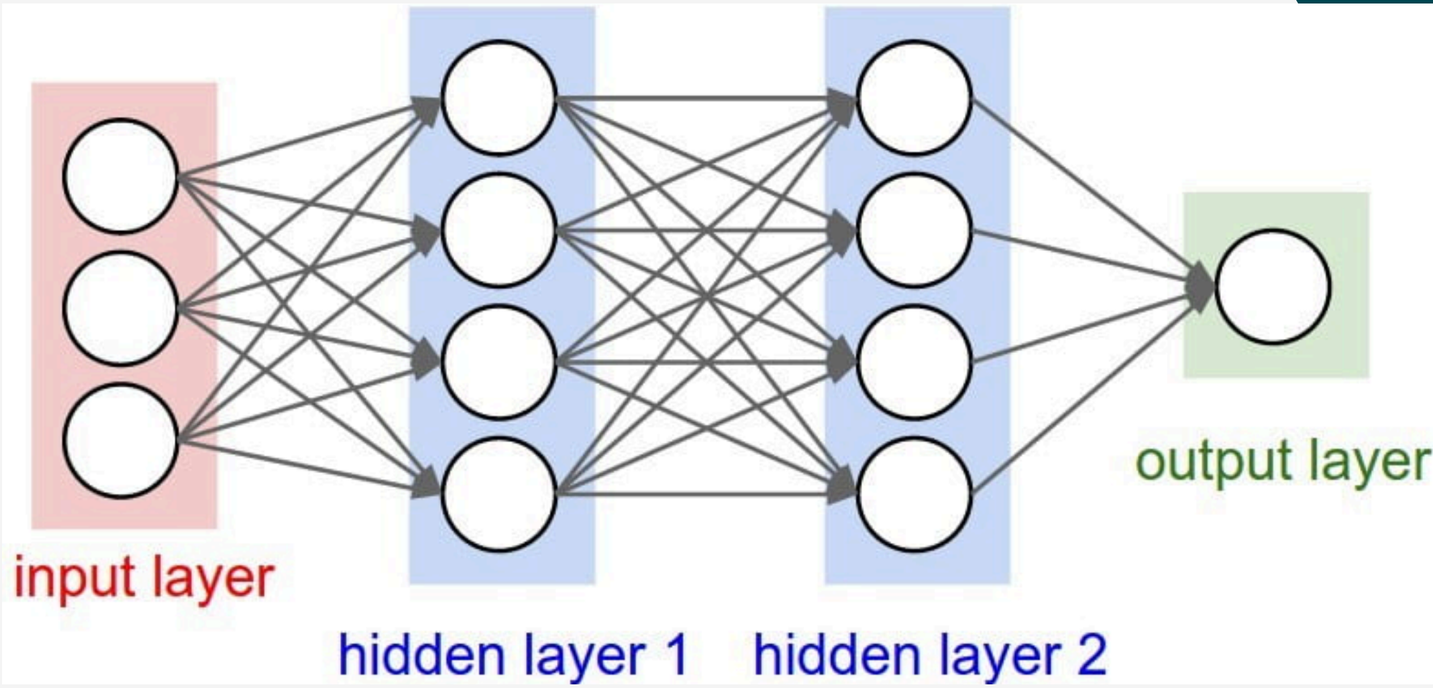
Neural Network

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

Architecture



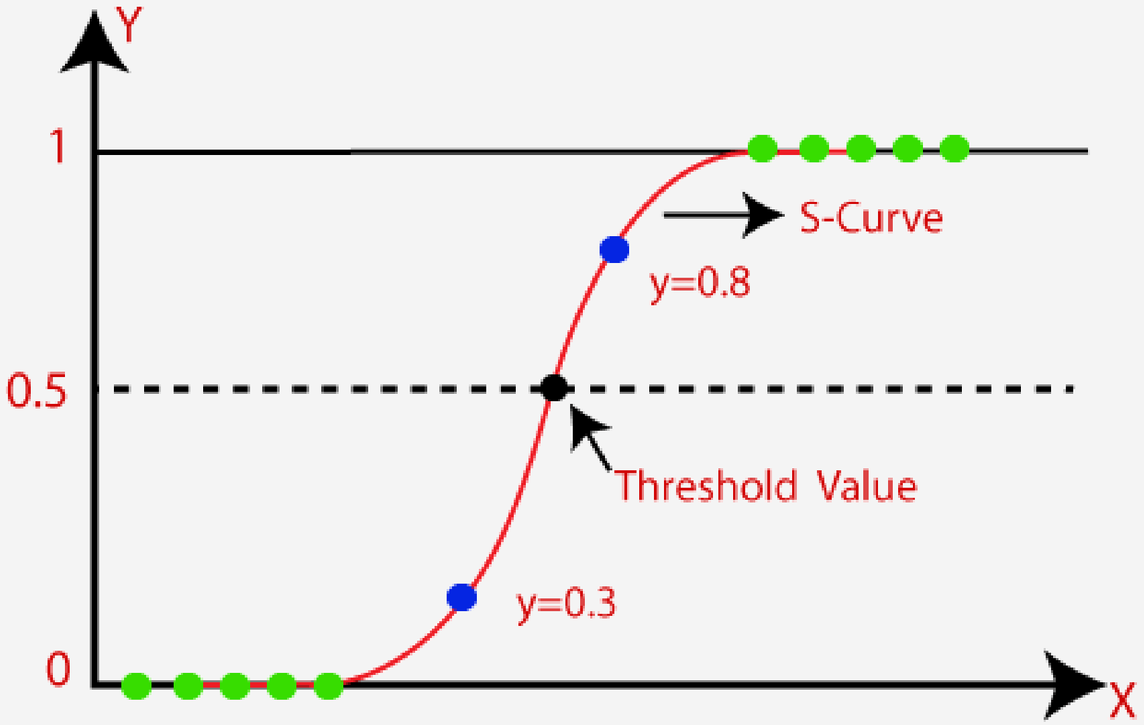
Neurons are building block of neural network



Activation Function

Sigmoid Function

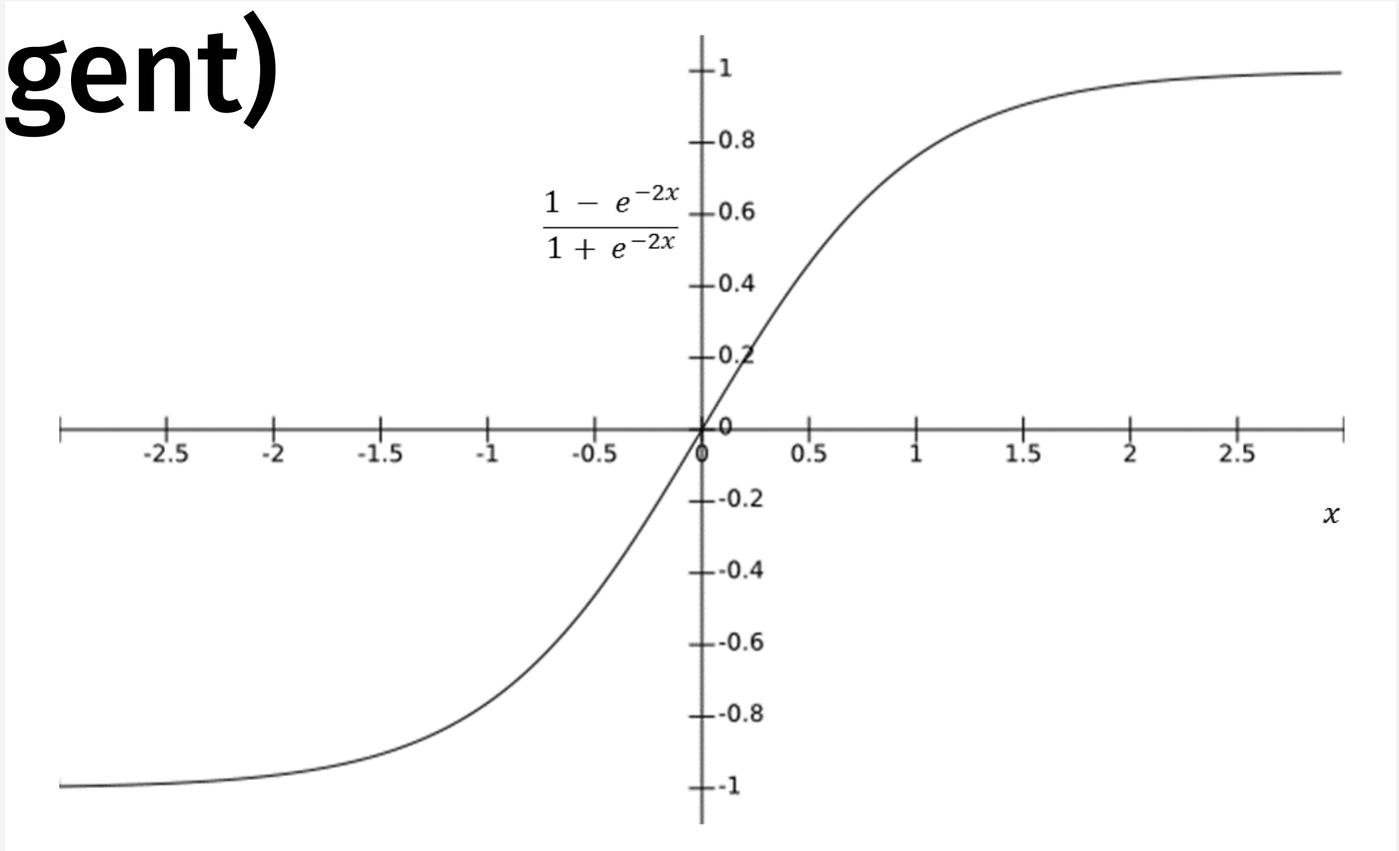
- Formula: $S(x) = \frac{1}{1 + e^{-x}}$
- Range : (0 , 1)



- **Characteristics**: 1) Used in binary classification problems
  2)Can suffer from vanishing gradients, making training deep networks difficult .

Tanh(Hyperbolic tangent)

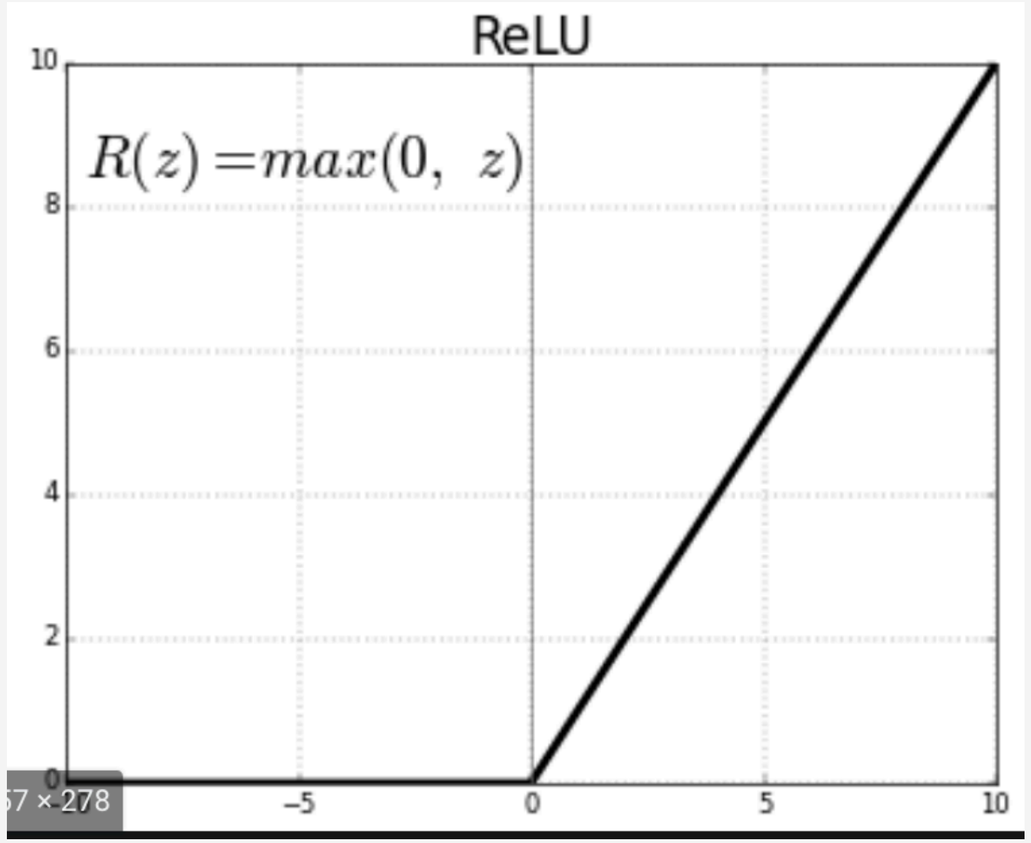
- Range: (-1,1)



Characteristics:

1) Zero-centered, which can make optimization easier
2) Like the sigmoid, can suffer from vanishing gradients

ReLU (Rectified Linear Unit)



Range : [0, ∞)

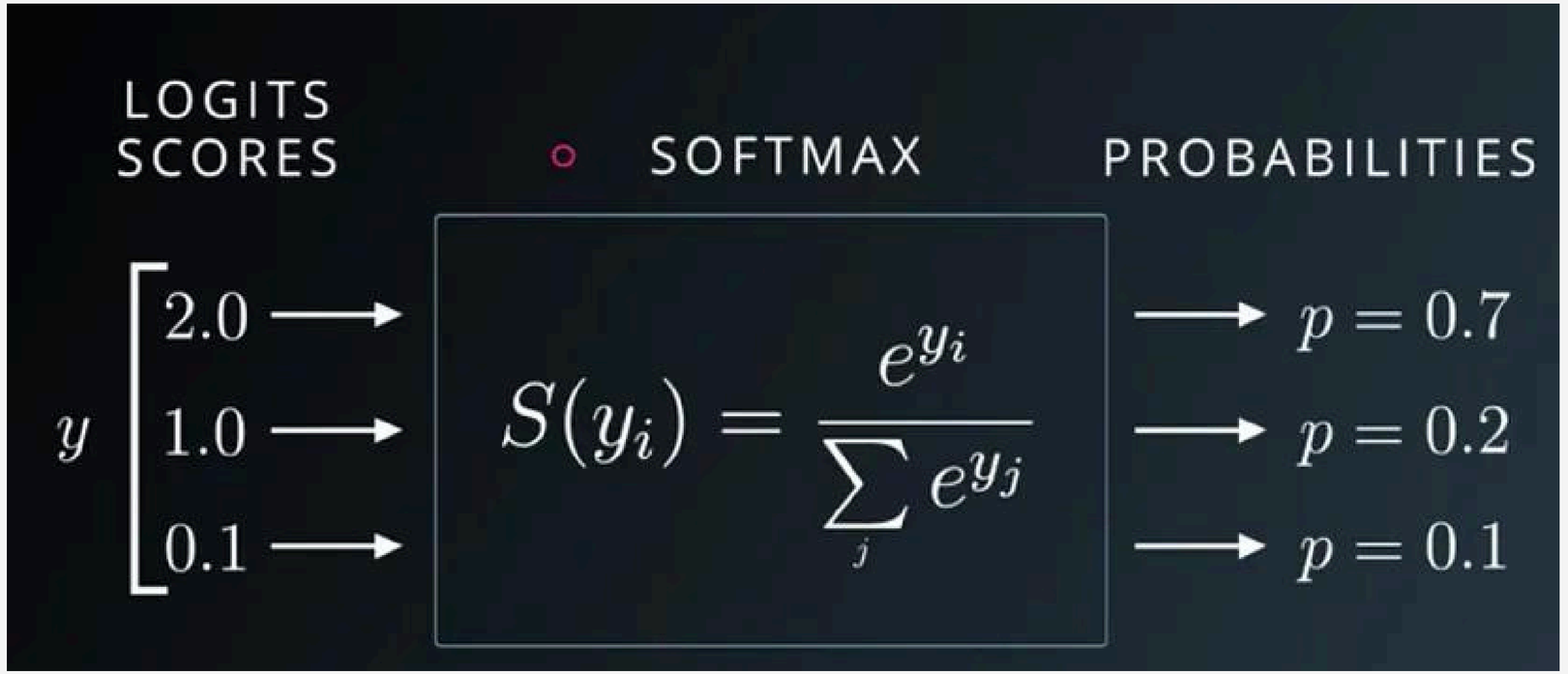
Characteristics:

1) Simple and efficient

2)Sparse activation (activates only a subset of neurons)

3)Can suffer from dying ReLUs, where neurons output zero for all inputs

Softmax Function

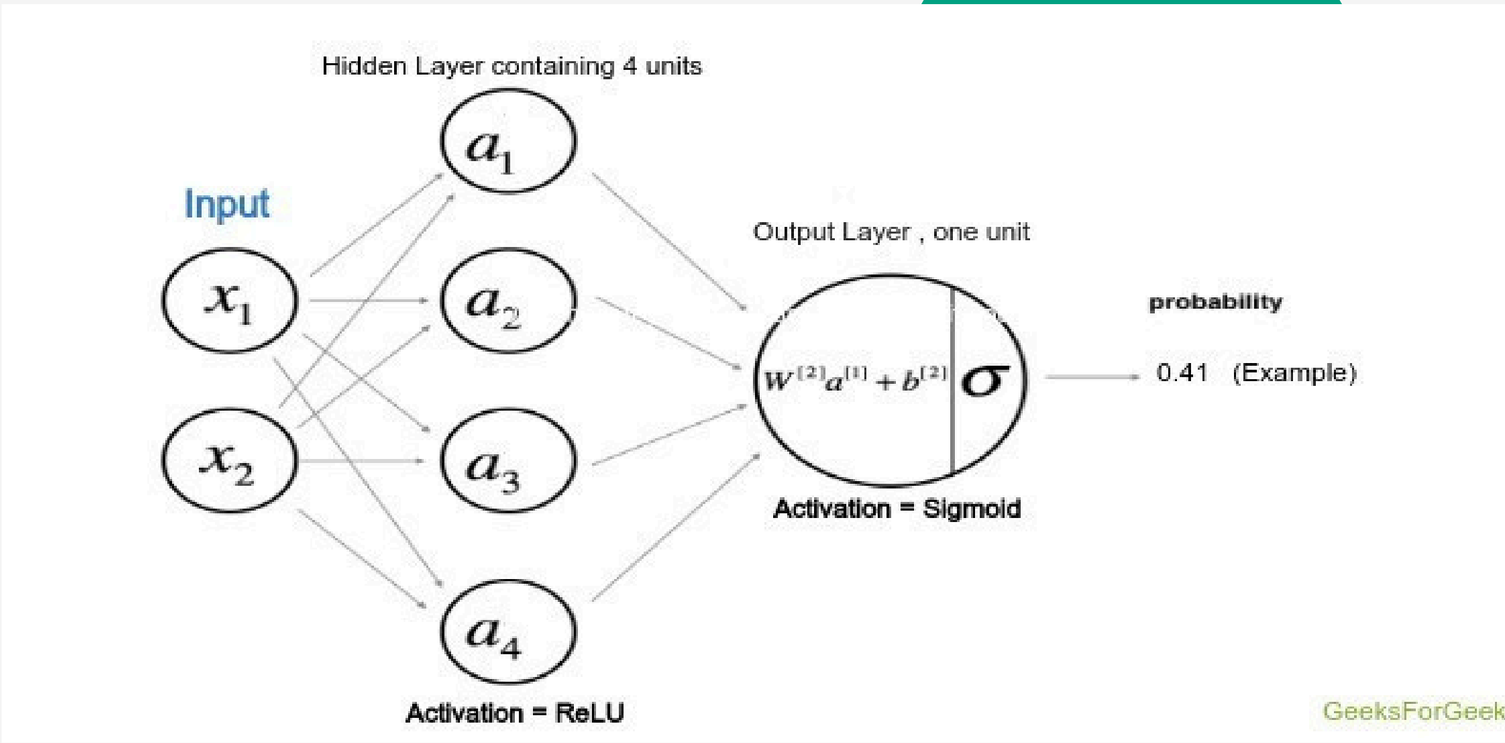


Characteristics

1) Used in the output layer for multi-class classification problems
2) Converts logits (raw prediction scores) to probabilities

Forward Propagation

1. For each neuron in the layers (hidden and output layers), the inputs are multiplied by corresponding weights, and a bias term is added. This results in a weighted sum , Z .



2. The weighted sum is then passed through an activation function to introduce non-linearity.

Lost and Cost function

: The loss function measures the error for a single training example.

: The cost function, also known as the objective function, is the average of the loss functions over the entire training dataset

Binary Cross Entropy
(For binary classification)

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

Categorical cross entropy
(For multiclass classification)

$$\text{Loss} = -\sum_{j=1}^{K} y_j \log(\hat{y}_j)$$

where k is number of classes in the data

Backword Propagation

Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error

It efficiently computes the gradient of the loss function with respect to each weight by using the chain rule of calculus, which allows the network to update its weights and biases to minimize the loss function

$$W_{new} = W_{old} - \alpha \underbrace{\frac{dJ}{dW}}_{\text{gradient}}$$

Here is an example of gradients calculated using backpropogation for a simple neural network

Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]} a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(\mathrm{z}^{[1]})$	$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(\mathrm{Z}^{[1]})$
$dW^{[1]} = dz^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

Andrew Ng

Optimisers

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses.

Different types of optimisers :

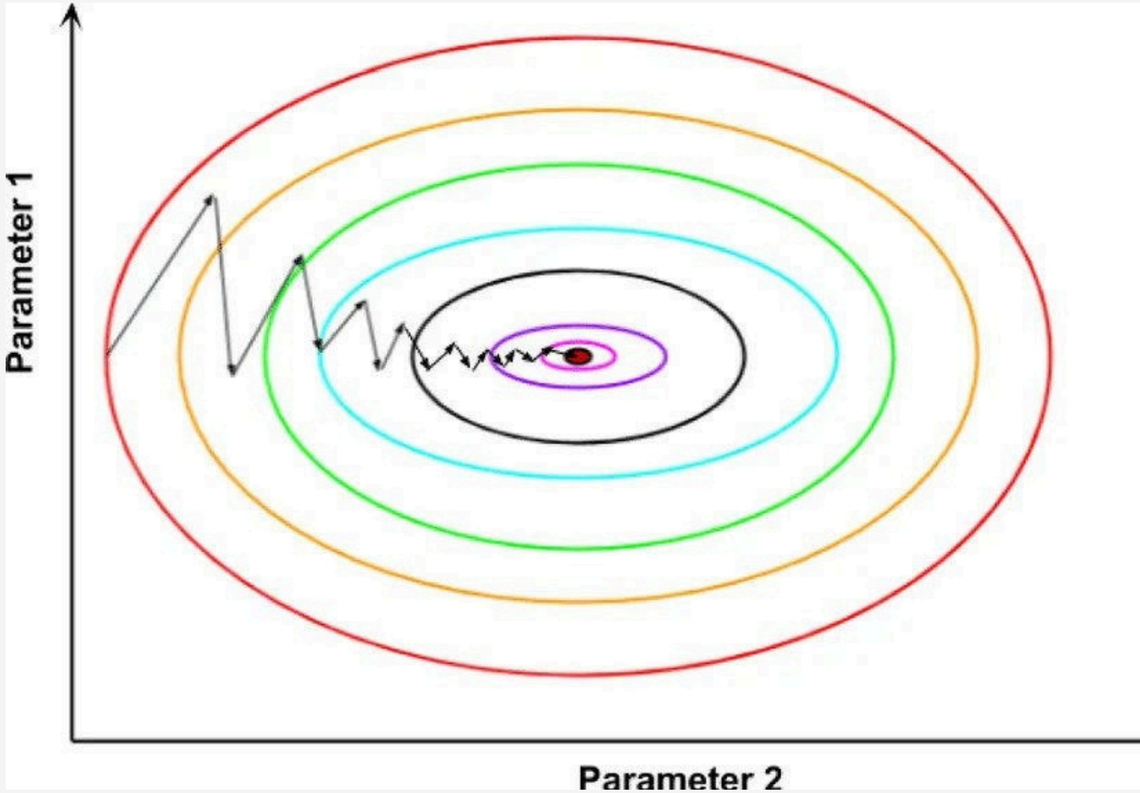
- Gradient Descent (GD) :

1)Batch Gradient Descent

2)Stochastic Gradient Descent (SGD)

3)Mini-batch Gradient Descent

- Momentum
- RMSprop (Root Mean Square Propagation)
- Adam (Adaptive Moment Estimation)



Convolutional Neural Netwok

What is CNN ?

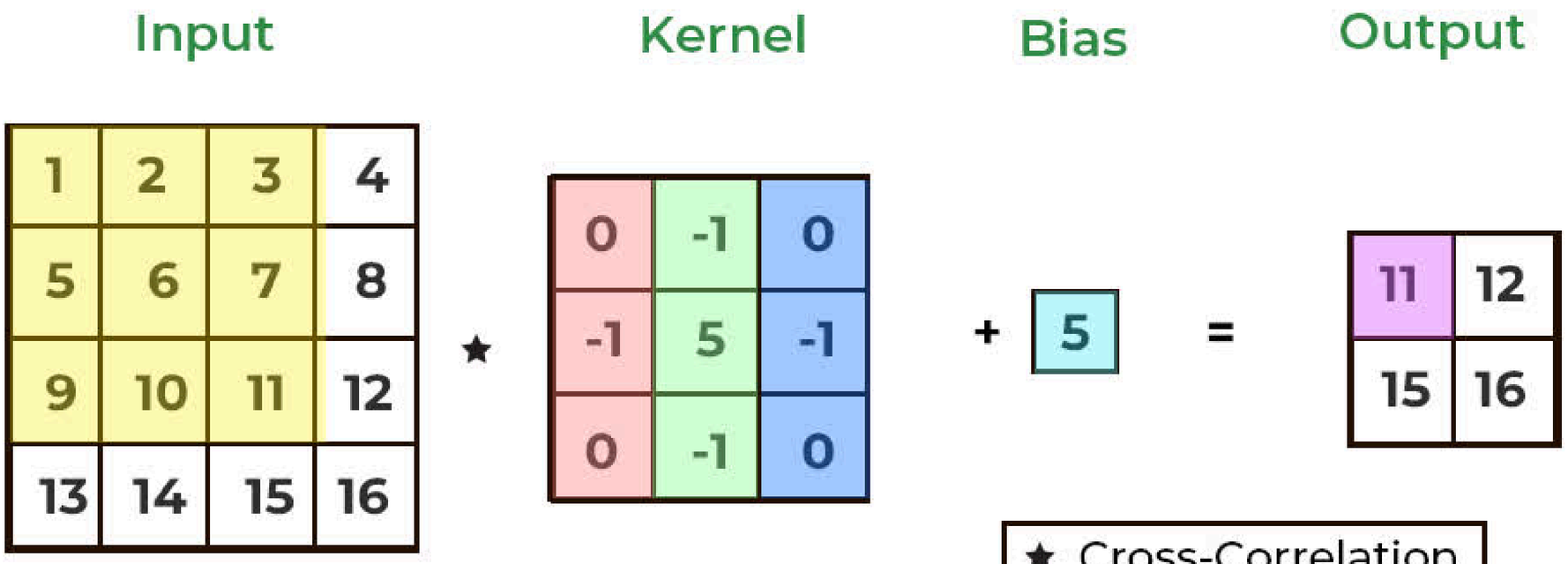
Convolutional Neural Networks (CNNs) are a class of deep learning algorithms specifically designed for processing structured grid data such as images. They are particularly well-suited for tasks involving spatial hierarchies in data, such as image recognition, object detection, and image segmentation. CNNs are inspired by the visual cortex and mimic the way humans see and process visual information.

Why CNN ?

By focusing on local regions and sharing parameters, CNNs require fewer computations and memory, making them more efficient for large-scale image processing.

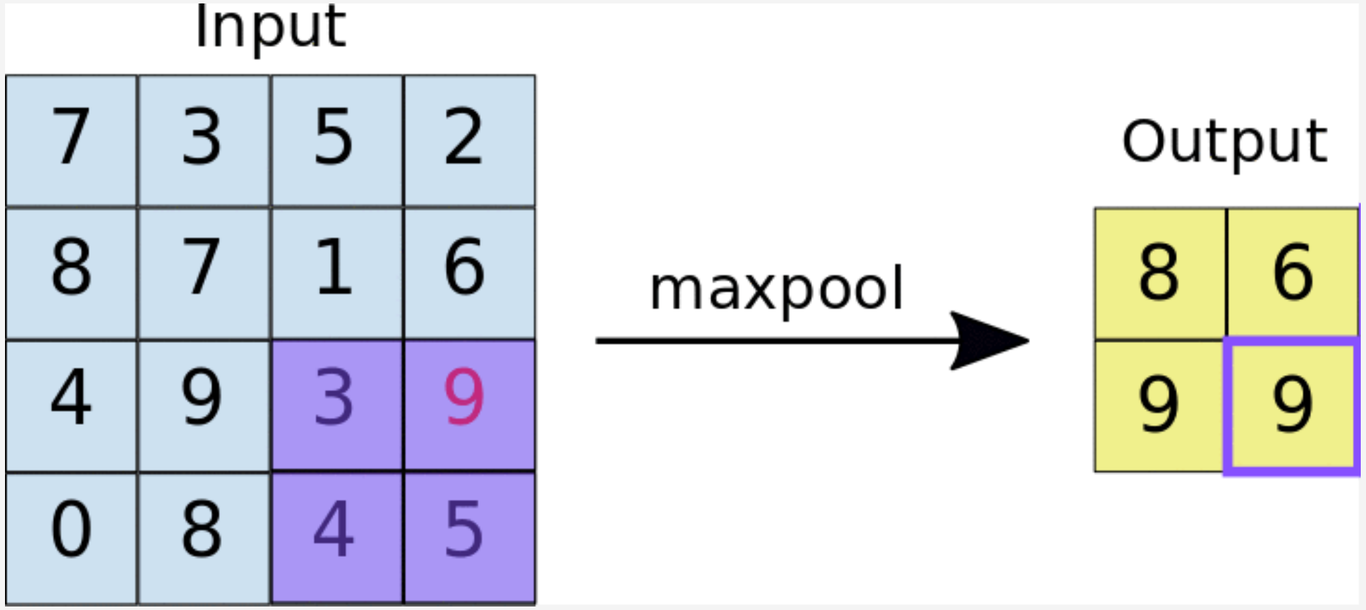
Convolution

It refers to the process of applying a filter (also called a kernel) to an input image to produce a feature map .



1 X 0 + 5 X -1 + 9 X 0 + 2 X -1 + 6 X 5 + 10 X -1 + 3 X 0 + 7 X -1 + 11 X 0 + 5 = 11

Pooling



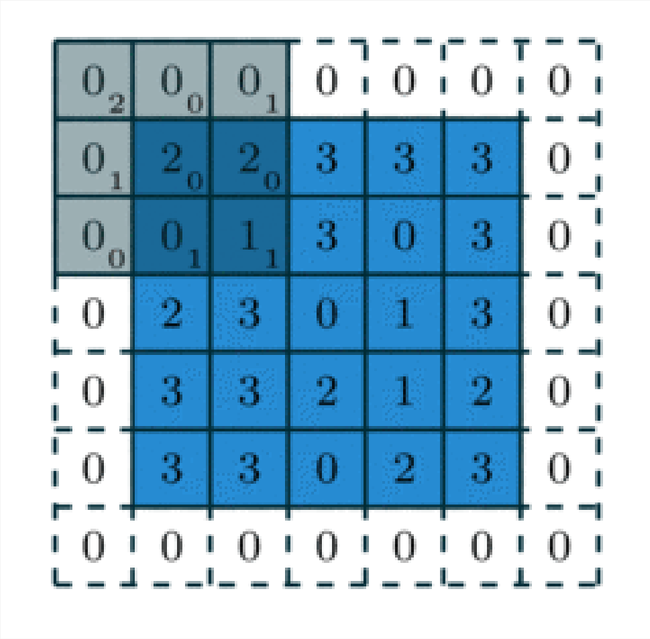
Pooling is nothing other than down sampling of an image. Role of pooling layer is to reduce the resolution of the feature map but retaining features of the map required for classification through translational and rotational invariants. In addition to spatial invariance robustness, pooling will reduce the computation cost by a great deal.

It is mainly of 2 types:

1. Max pooling:
2. Average poooling:

Padding

A technique used in Convolutional Neural Networks (CNNs) to adjust the spatial dimensions of the input volume to match the desired output dimensions. It involves adding additional border pixels (with zero values, for example) around the input image.

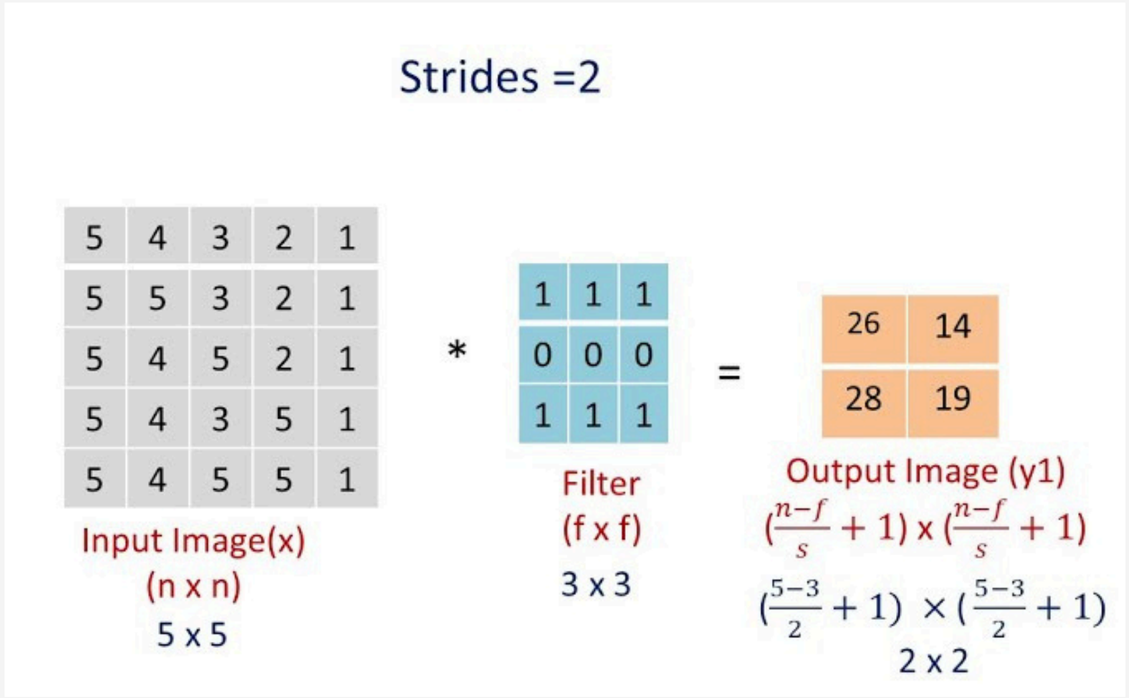


Why Padding ?

- Control Output Size
- Preserve Information at the Borders

Stride

The stride refers to the number of pixels by which the filter (kernel) is shifted over the input image. When performing convolutional operations, the filter slides over the input data in both the horizontal and vertical directions.

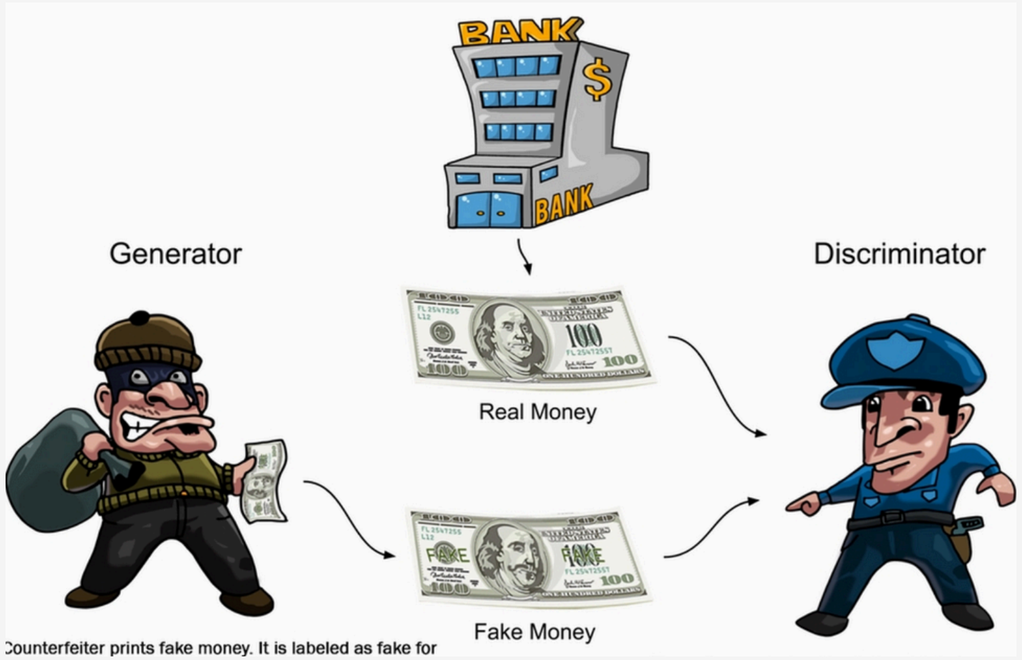


GANs

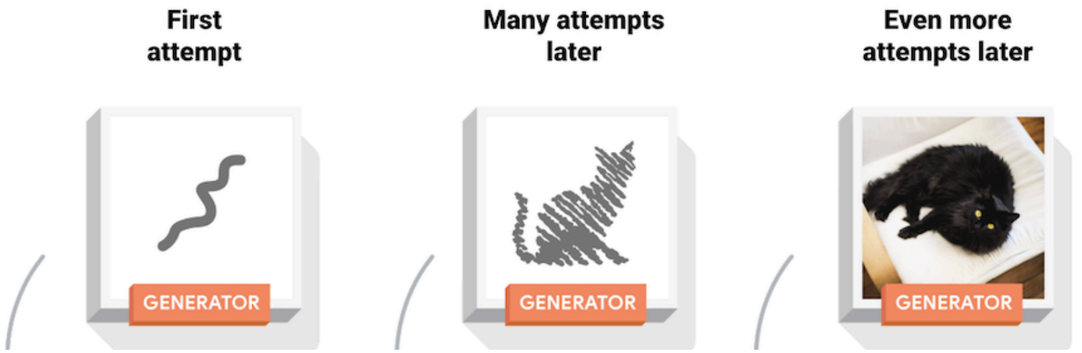
Generative Adversarial Networks (GANs) are a type of machine learning model that can create new data based on training data and is used to create realistic data.
It consists of two parts

1. Generator
2. Discriminator

They work together ,generator trying to fool discriminator and discriminator trying to catch him. Hence the word “Adverserial” comes.

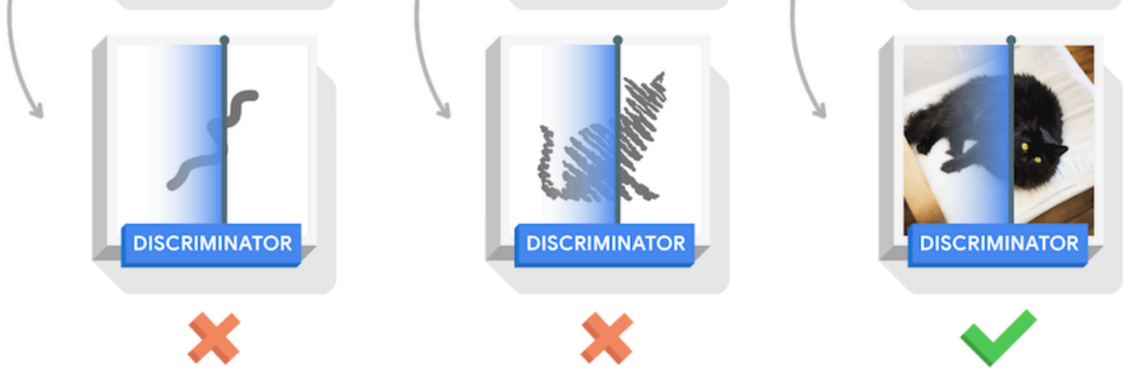


Generator



The Generator is a neural network designed to create **synthetic** data that resembles the real data it was trained on. It takes a **random noise vector** as input, which is typically sampled from a **standard normal distribution**, and transforms this noise through several layers, using linear transformations followed by non-linear activation functions, to produce an output that mimics the real data distribution. The generator's goal is to generate data that is indistinguishable from real data by the discriminator, which is another neural network in the GAN framework

Discriminator



The Discriminator is a neural network that serves as a **binary classifier**, distinguishing between real data from the training set and synthetic data generated by the generator. It receives input data, either **real or generated**, and processes it through several layers with linear transformations and non-linear activation functions to produce a single output value representing the probability that the input data is real. The discriminator's objective is to **maximize its accuracy** in classifying real versus fake data, thus providing **feedback** to the generator.

Value Function The MiniMax game

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \qquad (1)$$

In Generative Adversarial Networks (GANs), the value function V(D,G) represents the objective of the minimax game played between the generator G and the discriminator D. The generator aims to create data that closely resembles real data, while the discriminator aims to accurately distinguish between real and generated data.consists of two parts: the **first term** represents the expected log probability that the discriminator correctly identifies real data **x**, and the **second term** represents the expected log probability that the discriminator correctly identifies the generated data **G(z)** as fake

Common Problems and Remedies

Vanishing Gradient

Modified Minimax:The original GAN paper notes that the above minimax loss function can cause the GAN to get stuck in the early stages of GAN training when the discriminator's job is very easy. The paper therefore suggests modifying the generator loss so that the generator tries to maximize log D(G(z)).

Mode Collapse

Wasserstein Loss: Instead of using a discriminator to classify or predict the probability of generated images as being real or fake, the WGAN changes or replaces the discriminator model with a critic that scores the realness or fakeness of a given image

Unrolled GANs: Unrolled GANs use a generator loss function that incorporates not only the current discriminator's classifications, but also the outputs of future discriminator versions. So the generator can't over-optimize for a single discriminator.

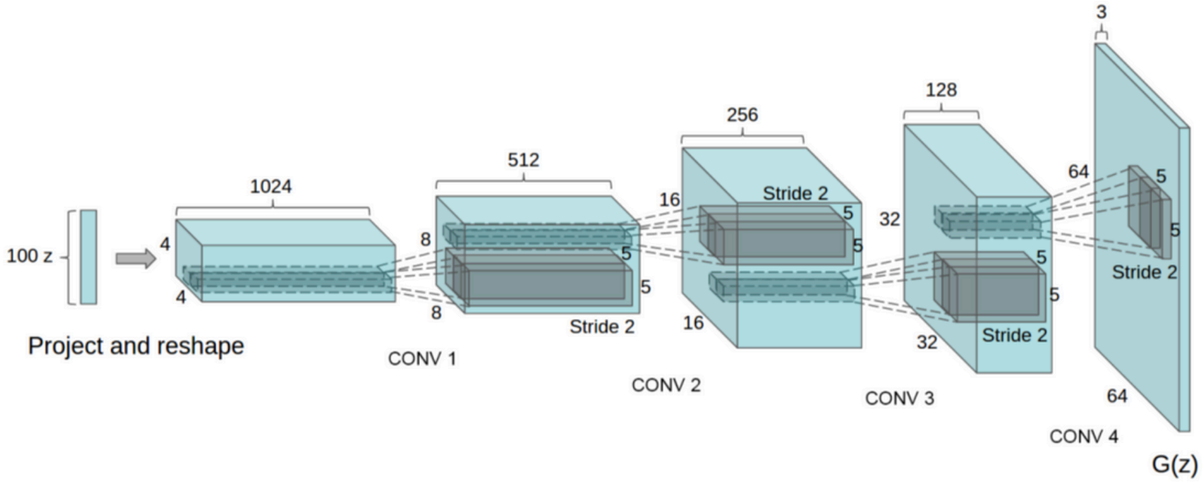
Types of GANs

There are many types of GANs, for example: Vanilla GAN, DCGAN, StackGAN, Conditional GAN etc.

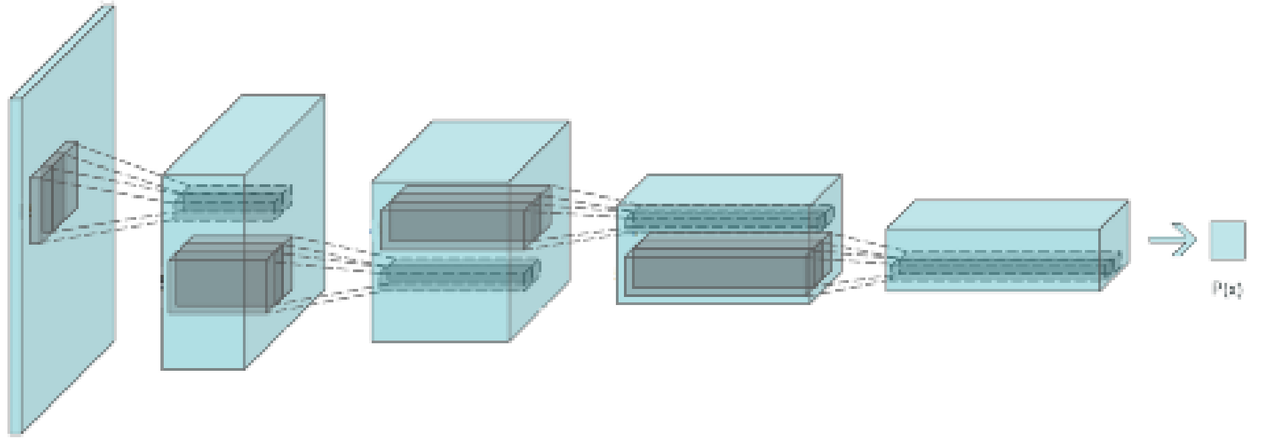
We are going to discuss only two of them in some detail in the upcoming slides. Those are:

1. DCGAN(Deep Convolutional GAN)
2. Stack GAN

DCGAN



Generator of DCGAN

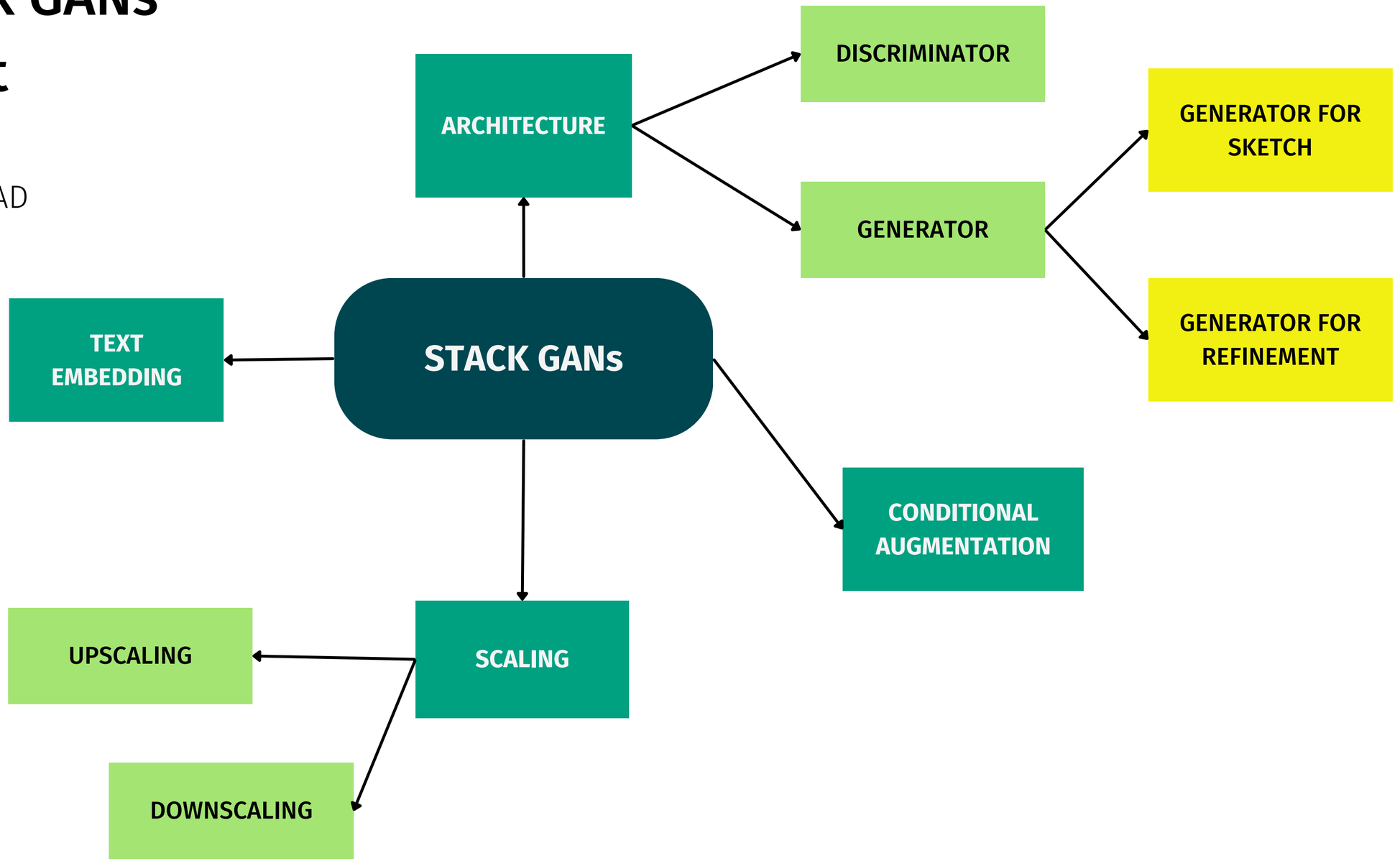


Discriminator of DCGAN

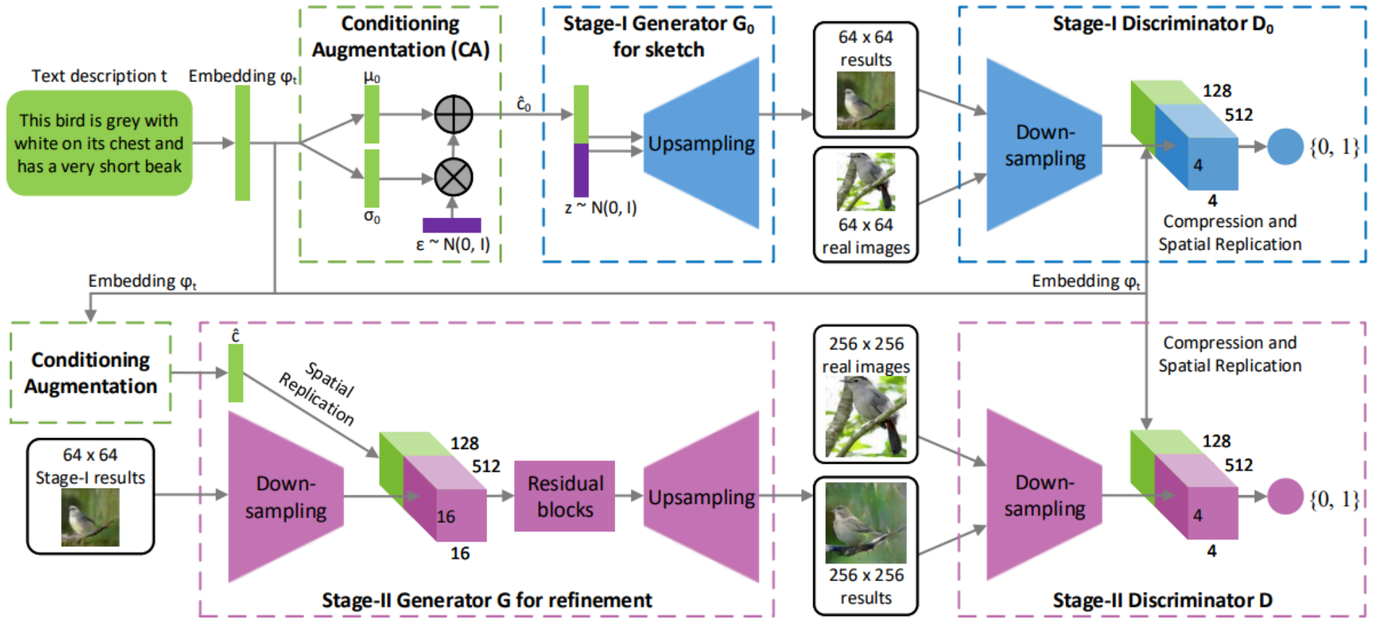
DCGAN is a type of GAN that uses deep convolutional networks to generate high-quality images. DCGANs use a series of convolutional and deconvolutional (transposed convolution) layers instead of fully connected layers, allowing them to capture spatial hierarchies in the data more effectively.

Stack GANs chart

EXPLAINED AHEAD

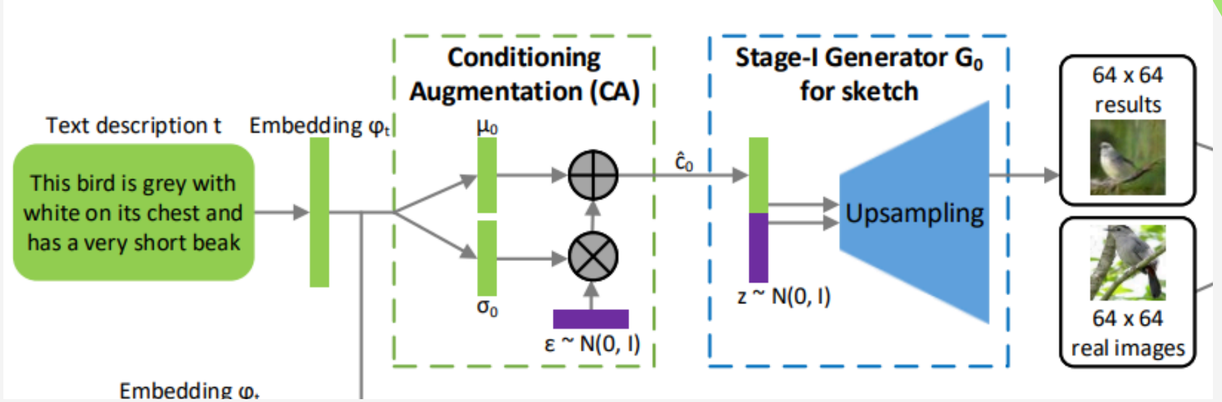


StackGAN



StackGANs address the challenge of generating fine details by dividing the generation process into two stages. In the first stage (Stage-I), a low-resolution image is generated from the text description, capturing the basic shape and colors. In the second stage (Stage-II), this low-resolution image is refined and upsampled to a higher resolution, adding finer details and improving visual quality

Steps



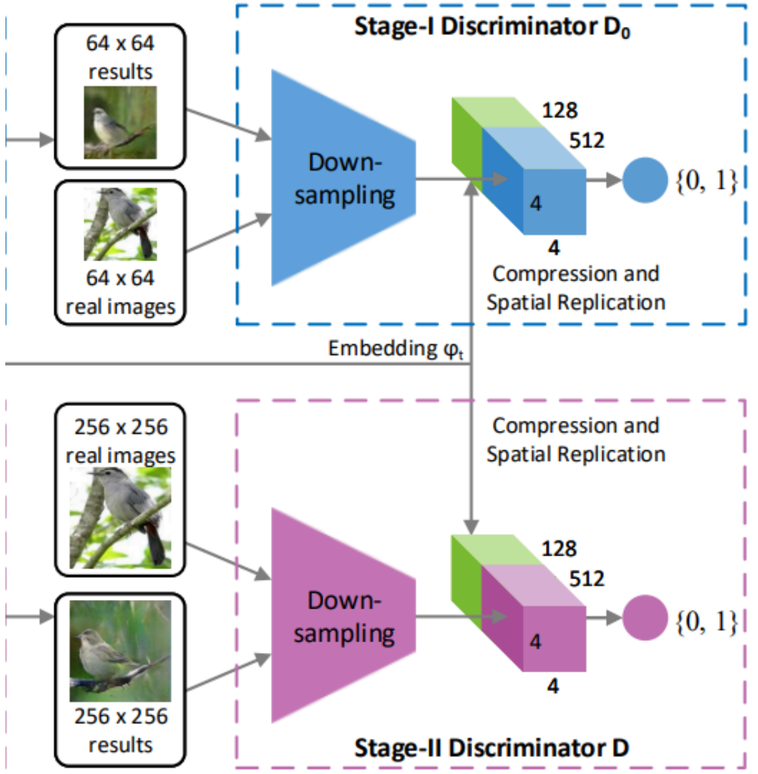
1. Text embedding and Generator G0

Text Embedding: Text embedding converts text into numerical vectors, enabling machine learning models to process textual data. Using this vector for spatial replication is called conditional augmentation.

StackGAN's First Generator (G0): G0 generates a coarse, low-resolution image from a text description. It transforms the text into an embedding, concatenates it with random noise, and feeds this input into a deep CNN to produce a low-resolution image that captures the basic structure, shape, and primary colors described by the text.

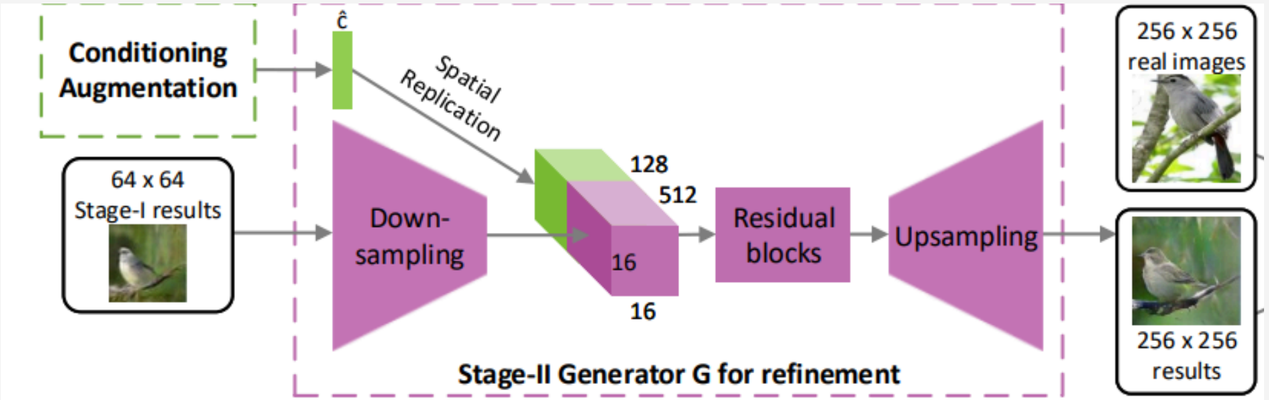
2. **Discriminator**

In Stage-I, the discriminator is a **CNN** that takes the low-resolution image produced by the Stage-I generator, along with the corresponding text embedding, as input. It evaluates whether the image is **real** (from the training data) or **fake** (generated by the generator) and whether it correctly matches the text description. This helps the generator to create images that are not only realistic but also **semantically consistent** with the text input. In Stage-II, the discriminator performs a similar role but operates on **higher-resolution** images refined by the Stage-II generator.



3. **Generator 2**

In Stage-I, the discriminator is a **CNN** that takes the low-resolution image produced by the Stage-I generator, along with the corresponding text embedding, as input. It evaluates whether the image is **real** (from the training data) or **fake** (generated by the generator) and whether it correctly matches the text description. This helps the generator to create images that are not only realistic but also **semantically consistent** with the text input. In Stage-II, the discriminator performs a similar role but operates on **higher-resolution** images refined by the Stage-II generator.



Stage 1 Generation

Stage 2 Generation

Some Important terms told

1. **Vectorisation**: It is a technique that converts data into numbers that represent a point or sequence of points in a vector space. This process is a step in feature extraction, where the goal is to extract distinct features from raw data, such as text, for the model to train on
2. **Upscaling**: It is increasing dimensions. To put it simply it is decoding.
3. **Downscaling**: It is decreasing dimensions. To put simply it is encoding.
4. **One Hot Encoding**: It is a type of vectorisation. It is a technique in machine learning that turns categorical data, like colors (red, green, blue), into numerical data for machines to understand. It creates new binary columns for each category, with a 1 marking the presence of that category and 0s elsewhere.

Text Preprocessing

Tokenization: In NLP, tokenization splits text into words or subwords for tasks like text classification, sentiment analysis, and machine translation.

Stemming: Stemming reduces words to their root form to normalize variations, treating different inflections or suffixes as the same word.

Lemmatization: Lemmatization, akin to stemming, reduces words to their base form, or lemma. Unlike stemming, it uses dictionary lookup or morphological analysis to find the precise base form, ensuring more accurate normalization.

DATA AUGMENTATION

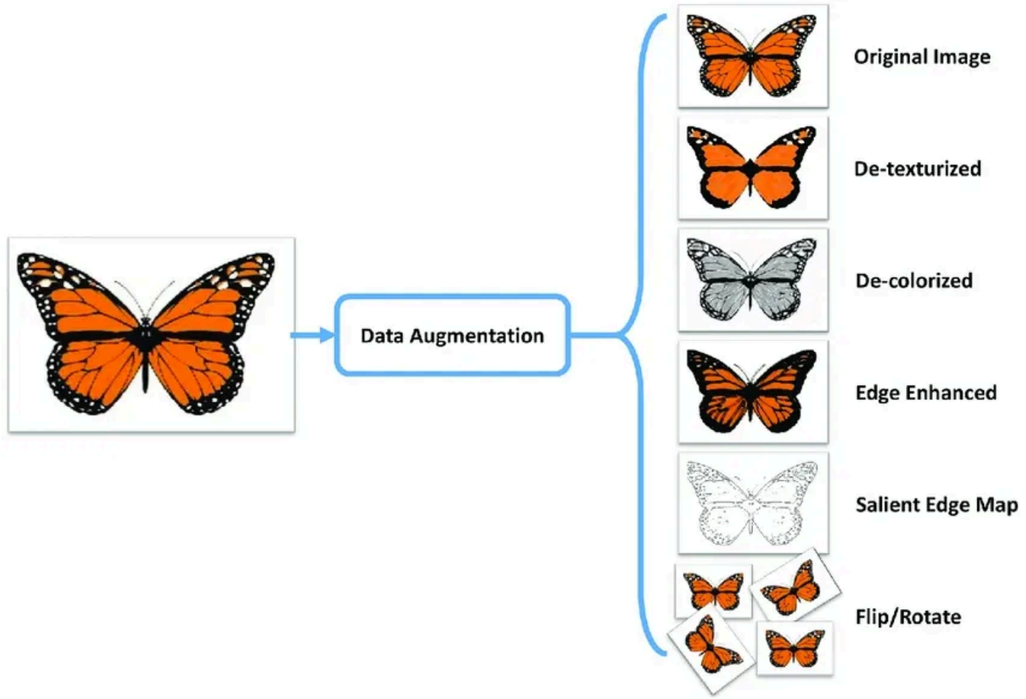
Data augmentation is a statistical technique that increases the size and diversity of a dataset by making small changes to existing data or generating new data points.

Benefits of data augmentation:

- Improved generalization
- Reduced overfitting
- Increased accuracy
- Reduced cost

Types of Data Augmentation

1. Classical
2. Advanced



CLASSICAL

This includes:

1. Flipping and Rotation
2. Cropping and resizing
3. Colour Jittering
4. Adding Noise
5. Image Warping
6. Random Erasing

These have been used for a long time and still helps. But there are many advanced techniques developed now that are better then these. Those are mentioned ahead.

ADVANCED

This includes:

1. Cutout

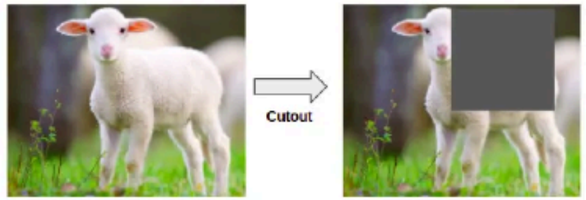


Figure 1: Cutout Illustration

2. Mixup

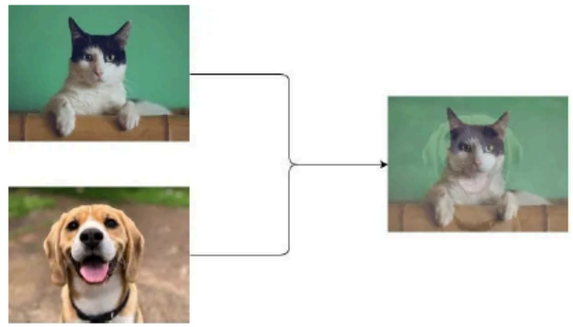


Figure 2: Mixup Illustration

3. CutMix

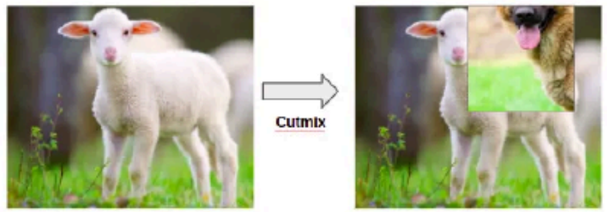


Figure 3: Cutmix Illustration

ADVANCED

4. AugMix

Augmix uses three separate chains of one to three randomly chosen augmentation operations, combined with the original image using different weights.It creates multiple sources of randomness, including the selection
of operations, their intensity, the length of the chains, and the mixing weights.
Combined with a consistency loss to ensure the augmented images are semantically
meaningful.

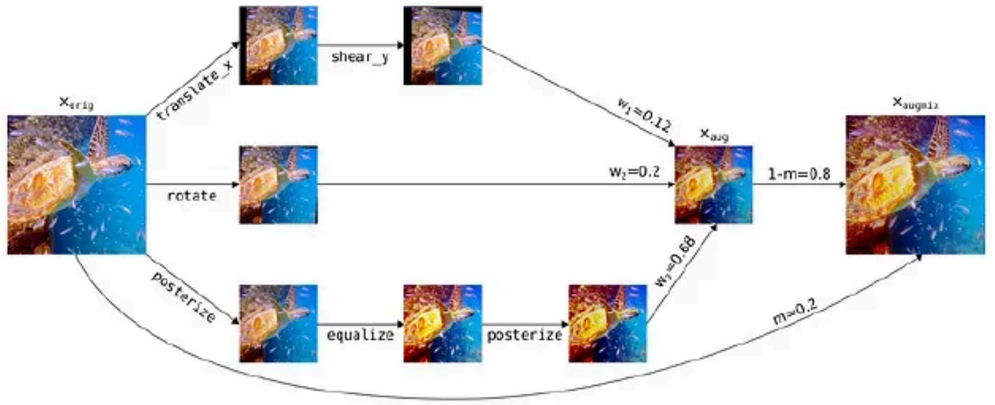


Figure 4: Augmix Illustration

Jensen-Shannon Consistency Loss

$$JS(p,q) = \frac{1}{2}\left(KL(p\|m) + KL(q\|m)\right)$$

$$L_{JS} = \lambda \cdot JS(p,q)$$

where $m = \frac{1}{2}(p+q)$ and λ is the weight of JSC in the loss.

Jensen-Shannon is an asymmetric metric that measures the relative entropy or difference in information represented by two distributions. Closely related to KL Divergence, it can be thought of as measuring the distance between two data distributions showing how different the two distributions are from each other.