SBS4115 Fundamentals of AI & Data Analytics

Introduction to Advanced AI Techniques

Lecturer: Ir Dr Kelvin K. W. Siu
email: kelvinsiu@thei.edu.hk

高科院
Thei

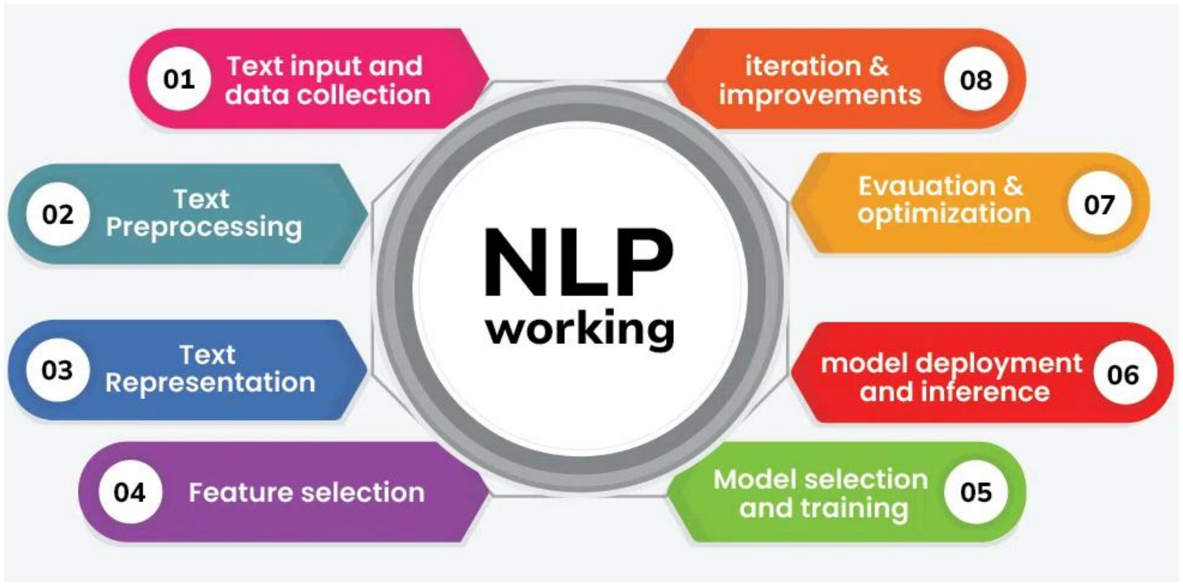
Department of Construction,
Environment and Engineering

Intended Learning Outcomes

- By the end of this lecture, you will be able to…
  - Describe NLP
  - Introduce generative AI
  - Apply OpenCV in image processing

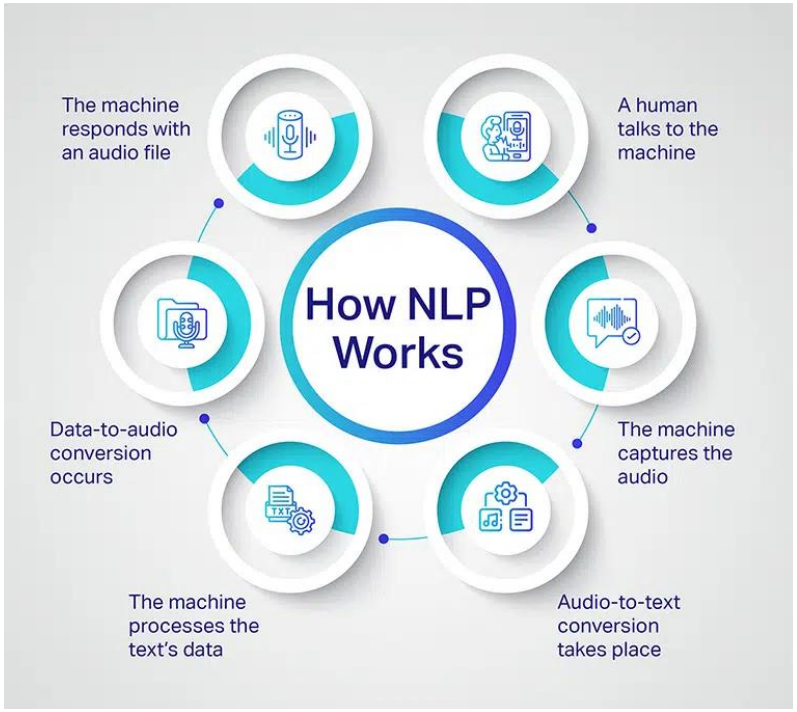
What is Natural Language Processing (NLP)?

- NLP enables computers to understand, interpret, and respond to human language.
- Computer Science, Linguistics, Machine Learning
- Language translation, sentiment analysis, information retrieval, and conversational AI


NLP working
01 Text input and data collection
02 Text Preprocessing
03 Text Representation
04 Feature selection
05 Model selection and training
06 model deployment and inference
07 Evauation & optimization
08 iteration & improvements

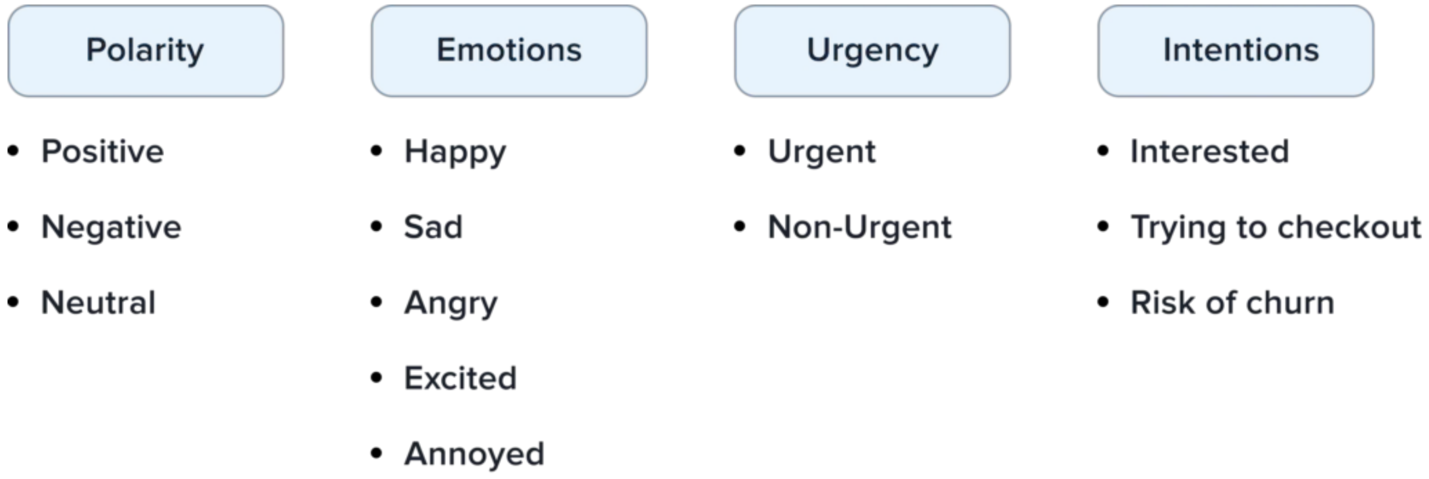
NLP in Everyday Life

- NLP transforms digital interactions to be more natural and seamless.
- Industries: Healthcare, Finance, Retail, Social Media
- Applications: Chatbots, Virtual Assistants (Siri, Alexa), and Content Moderation


How NLP Works
A human talks to the machine
The machine captures the audio
Audio-to-text conversion takes place
The machine processes the text's data
Data-to-audio conversion occurs
The machine responds with an audio file

Advantages of NLP

- Automation: Customer support, data entry, document processing
- Enhanced Data Analysis: Identifies trends, sentiment analysis
- Improved Search Functionality: Better query intent understanding
- Content Generation: Articles, reports, and creative writing



Polarity	Emotions	Urgency	Intentions
• Positive	• Happy	• Urgent	• Interested
• Negative	• Sad	• Non-Urgent	• Trying to checkout
• Neutral	• Angry		• Risk of churn
	• Excited		
	• Annoyed		

Core NLP Components

- Syntax: Structure and grammar of sentence

  Example: Analyzing "The quick brown fox..."
- Semantics: Understanding word meanings and context

  Example: Interpreting "bank" as a financial institution or riverbank
- Pragmatics: Understanding implied meanings

  Example: “It’s cold in here” as a request to close a window
- Discourse: Maintaining context across sentences

Key Techniques in NLP

- Tokenization: Dividing text into smaller segments

  Example: "I enjoy reading books" becomes ["I", "enjoy", "reading", "books"]

- Stemming and Lemmatization: Reducing words to their roots

  Example: “Running” → “run” (stemming), “better” → “good” (lemmatization)

- Parsing: Understanding sentence structure

  Example: Identifying subject and predicate in “The cat sat on the mat”

Common NLP Tasks

- Coreference Resolution: Identifying when words refer to the same entity
- Named Entity Recognition (NER): Extracting names, dates, locations
- Part of Speech Tagging: Identifying nouns, verbs, adjectives
- Word Sense Disambiguation: Determining word meaning in context

Coreference Resolution

- Definition: Identifying when different words or phrases refer to the same entity in text.
- Example: “Tom went to the store. He bought milk.”
- Highlight: 'Tom' and 'He' refer to the same entity.

Named Entity Recognition (NER)

- Definition: Extracting specific names, dates, and locations from text.
- Example: "Apple Inc. was founded on April 1, 1976, in Cupertino, California.“
- Highlight: 'Apple Inc.' (Organization), 'April 1, 1976' (Date), 'Cupertino, California' (Location).

Part of Speech Tagging

- Definition: Assigning parts of speech to each word, such as nouns, verbs, adjectives.
- Example: "The quick brown fox jumps over the lazy dog.”
- Highlight: 'The' (Determiner), 'fox' (Noun), 'jumps' (Verb), 'lazy' (Adjective).

Word Sense Disambiguation

- Definition: Determining the correct meaning of a word based on context.
- Example: "I went to the bank to sit by the river." vs. "I went to the bank to deposit money.”
- Highlight: "Bank" as 'Riverbank' vs. 'Financial Institution.'

Approaches to NLP

- 1. Rules-Based: Uses specific if-then statements; limited flexibility
- 2. Statistical: Uses statistical methods for pattern recognition

  Examples: Spell check, predictive text
- 3. Deep Learning: Neural networks for complex tasks

  Models: Sequence-to-Sequence, Transformer (BERT, GPT)

Rules-Based Approach

- Definition: Uses a set of hard-coded if-then rules to process language. Effective for simple, predictable tasks but limited in handling complex language variations.
- Example: Text transformation for specific formats, such as converting dates ("January 1st" to "01/01").
- Application: Grammar checkers that follow predefined language rules.

Statistical Approach

- Definition: Employs statistical methods to identify patterns in large datasets, using probability-based algorithms.
- Examples: Spell Check: Analyzes common typing errors using frequency statistics.
- Predictive Text: Uses word frequency and probability to suggest the next word in a sentence.

Deep Learning Approach

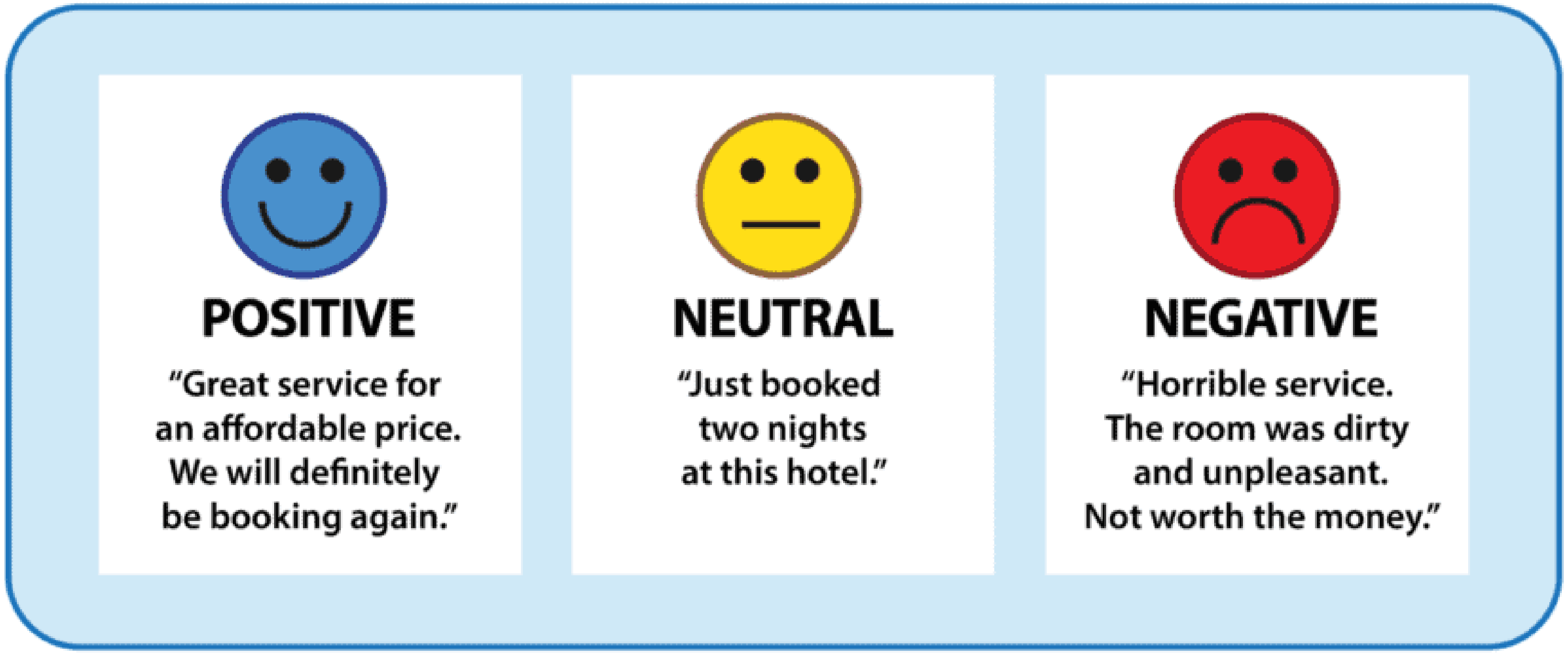
- Definition: Uses neural networks to process language, allowing for complex language understanding through layers of computation.
- Examples: Sequence-to-Sequence Models: Used in language translation where input sequences are converted to output sequences.
- Transformer Models: Advanced models like BERT and GPT understand context and generate coherent text.

Deep Learning Models-BERT and GPT

- BERT (Bidirectional Encoder Representations from Transformers): Reads text bidirectionally for better context understanding.
- Application: Sentiment analysis, question-answering tasks.
- GPT (Generative Pre-trained Transformer): Generates text by predicting the next word in a sequence.
- Application: Chatbots, content creation.

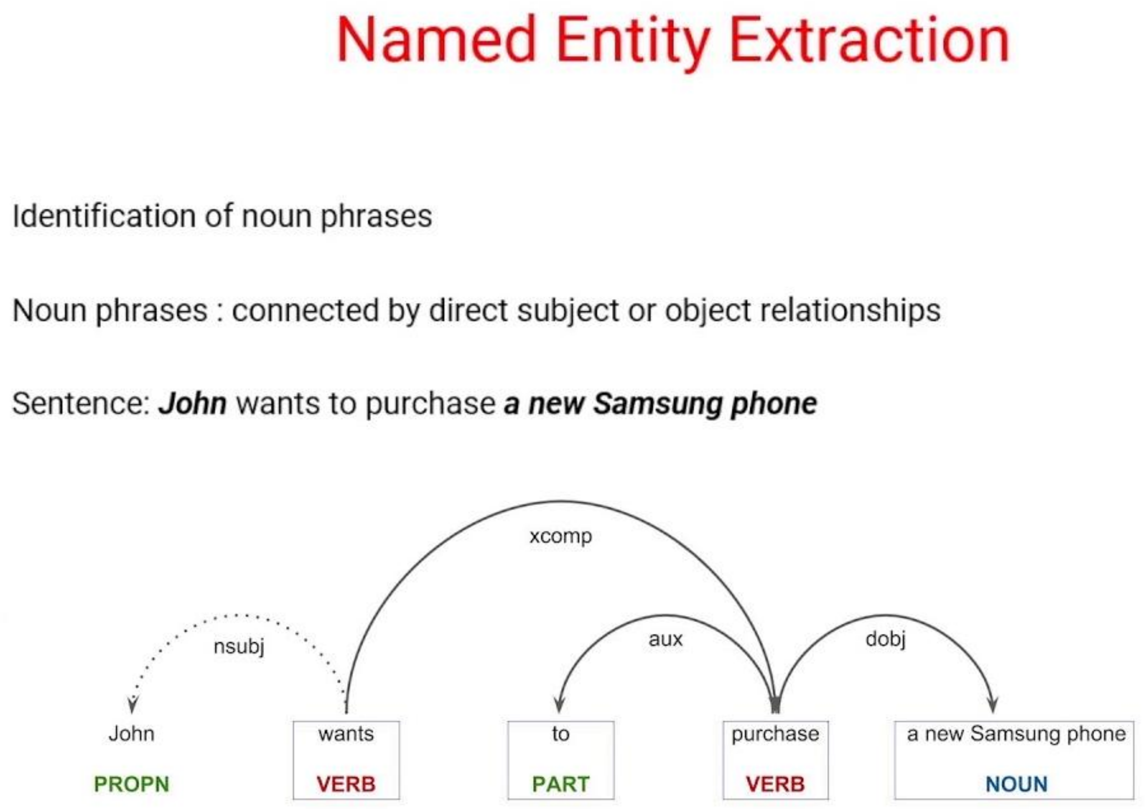
NLP Applications

- Sentiment Analysis: Classifies emotional intent in text
- Toxicity Classification: Identifies hostile language


POSITIVE
"Great service for an affordable price. We will definitely be booking again."
NEUTRAL
"Just booked two nights at this hotel."
NEGATIVE
"Horrible service. The room was dirty and unpleasant. Not worth the money."

NLP Applications

- Machine Translation: Automates language translation
- Named Entity Recognition (NER): Summarization, information extraction



Named Entity Extraction

Identification of noun phrases

Noun phrases : connected by direct subject or object relationships

Sentence: ***John*** wants to purchase ***a new Samsung phone***

NLP in Action

- Customer Service: Resolving common issues via chatbots
- Healthcare: Analyzing medical record
- Search Engines: Enhancing search relevance
- Social Media Moderation: Detecting harmful content

Challenges in NLP

- Handling Ambiguity: Language nuances are complex
- Bias: Models may reflect societal biases
- Contextual Understanding: Long-term conversation context is challenging

The Future of NLP

- Advancements: NLP is increasingly human-like, improving communication between humans and machines
- Opportunities: Enhanced interactions, inclusivity, and accessibility
- Challenges Ahead: Ongoing bias correction, better context handling

Tokenization with Python

```
import nltk

nltk.download('punkt')

from nltk.tokenize import word_tokenize, sent_tokenize


text = "Hello! NLP is fascinating. Let's explore it step by step."
```

Tokenization with Python

```
# Tokenize the text into sentences
sentences = sent_tokenize(text)
print("Sentences:", sentences)

# Tokenize the text into words
words = word_tokenize(text)
print("Words:", words)
```

Word Frequency Counter

```
from collections import Counter

import re



def word_frequency(text):

    # Remove punctuation and convert to lowercase

    text = re.sub(r'[^\w\s]', '', text).lower()

    words = text.split()

    return Counter(words)



text = "This is a simple example. A simple, simple example!"

print(word_frequency(text))
```

Remove Stop Words

```
from nltk.corpus import stopwords
nltk.download('stopwords')


def remove_stop_words(text):
    stop_words = set(stopwords.words("english"))
    words = word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return filtered_words
```

Remove Stop Words

```
text = "This is an example sentence, which we will use to remove stop words."

print("Filtered words:", remove_stop_words(text))
```

Basic Sentiment Analysis with TextBlob

```
!pip install textblob
```

Basic Sentiment Analysis with TextBlob

```
from textblob import TextBlob


def get_sentiment(text):
    blob = TextBlob(text)
    return blob.sentiment


text = "I love learning NLP, it's amazing!"
print("Sentiment:", get_sentiment(text))
```

Basic Sentiment Analysis with TextBlob

The result from TextBlob's sentiment analysis includes two values:

Polarity: This is a measure of how positive or negative the text is. It ranges from -1 to 1, where:

1 means the text is very positive.

0 means the text is neutral.

-1 means the text is very negative.

The polarity is 0.625, which is positive, indicating the sentence has an overall positive sentiment.

Basic Sentiment Analysis with TextBlob

Subjectivity: This measures the degree of personal opinion or factual information in the text.

It ranges from 0 to 1, where:

1 means the text is very subjective (based on personal opinion, emotion, or perspective).

0 means the text is very objective (based on factual information).

In your example, the subjectivity is 0.75, indicating the text is fairly subjective, as it expresses personal enjoyment and positivity about learning NLP.

So, the sentence "I love learning NLP, it's amazing!" is interpreted as quite positive and mostly based on personal opinion rather than factual information.

What is Generative AI?

- Definition: AI that generates new, original content (text, images, audio).
- Core Difference: Unlike traditional AI, it creates rather than predicts or classifies.
- Examples: Text (ChatGPT), Images (DALLE), Synthetic Data

cloud. a cube
macro photogra
ssion dolores. a ph
in pop art style. a pa
ew of a strawberry. a cro
stack of chips sitting on a
a small red block on a
a photo of a museum in cuba. a ph
a photo of a computer from the 70s.
image of a blue pineapple. a neon si
large blue block.

A Brief History of Generative AI

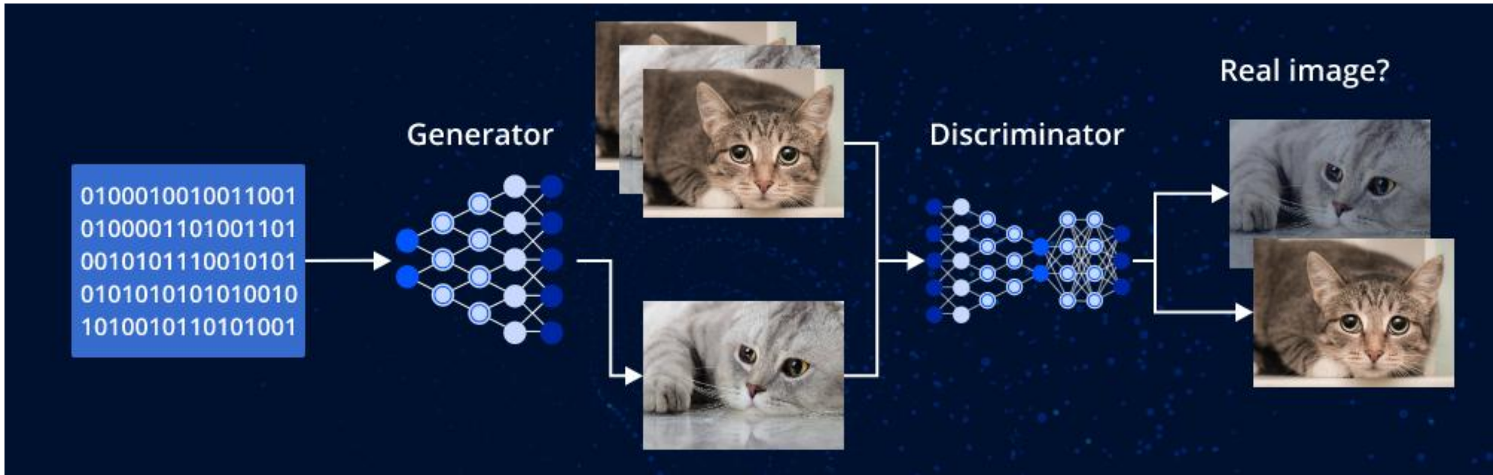
- 1950s: Text Analytics
- 1960s: Rule Based Systems
- 1980s: NLP Development
- 2000s: Rise of Machine Learning
- 2014: GANs by Ian Goodfellow
- 2017: Transformers by Google

Key Generative AI Models

- Large Language Models (LLMs): GPT, BERT – language generation and understanding
- GANs: Effective in realistic image/video generation
- Diffusion Models: Create high quality images (e.g., Stable Diffusion)
- Transformers: Foundation of LLMs for complex text generation

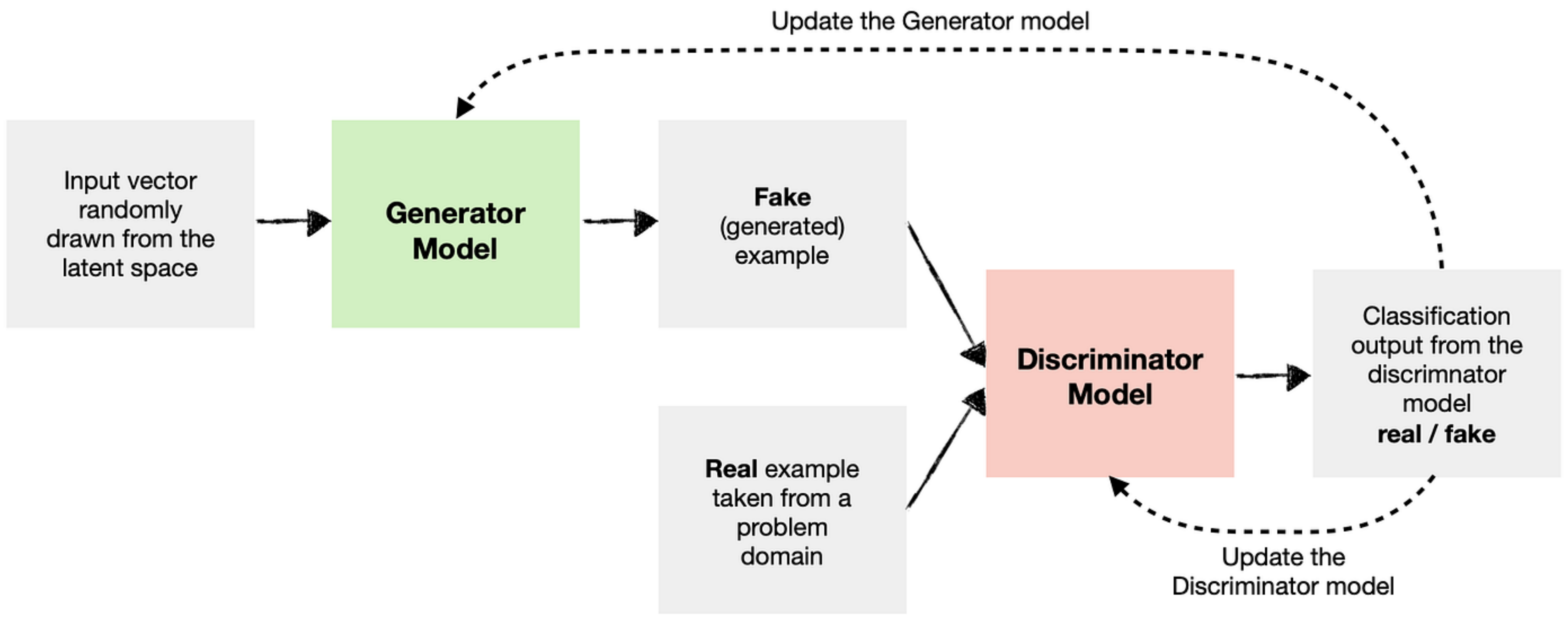
How Generative AI Works

- Data driven learning: Learns from data patterns to create humanlike outputs.
- Key techniques: GANs, Transformers, Diffusion Models
- GAN stands for Generative Adversarial Network.
- Consists of two neural networks: the Generator and the Discriminator.
- The Generator creates fake data, while the Discriminator tries to distinguish between real and fake data



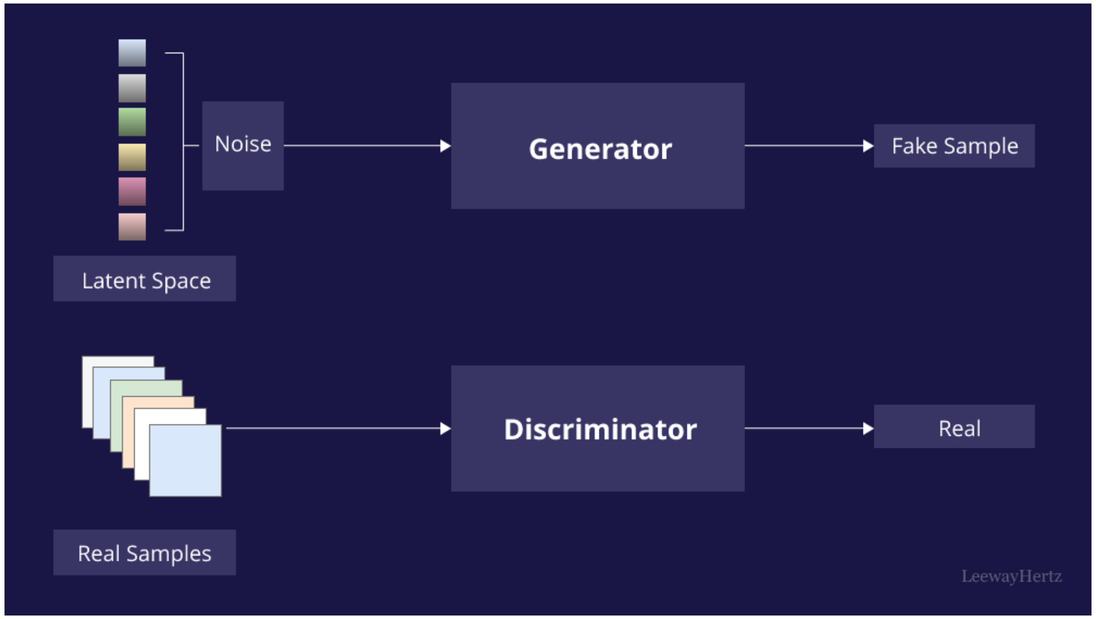
Generative AI Techniques Overview

- Generative Adversarial Networks (GANs): Dual network structure (Generator vs. Discriminator)
- Transformers: Uses attention for context in language, basis for LLMs
- Diffusion Models: Gradual noise manipulation for high quality image synthesis



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

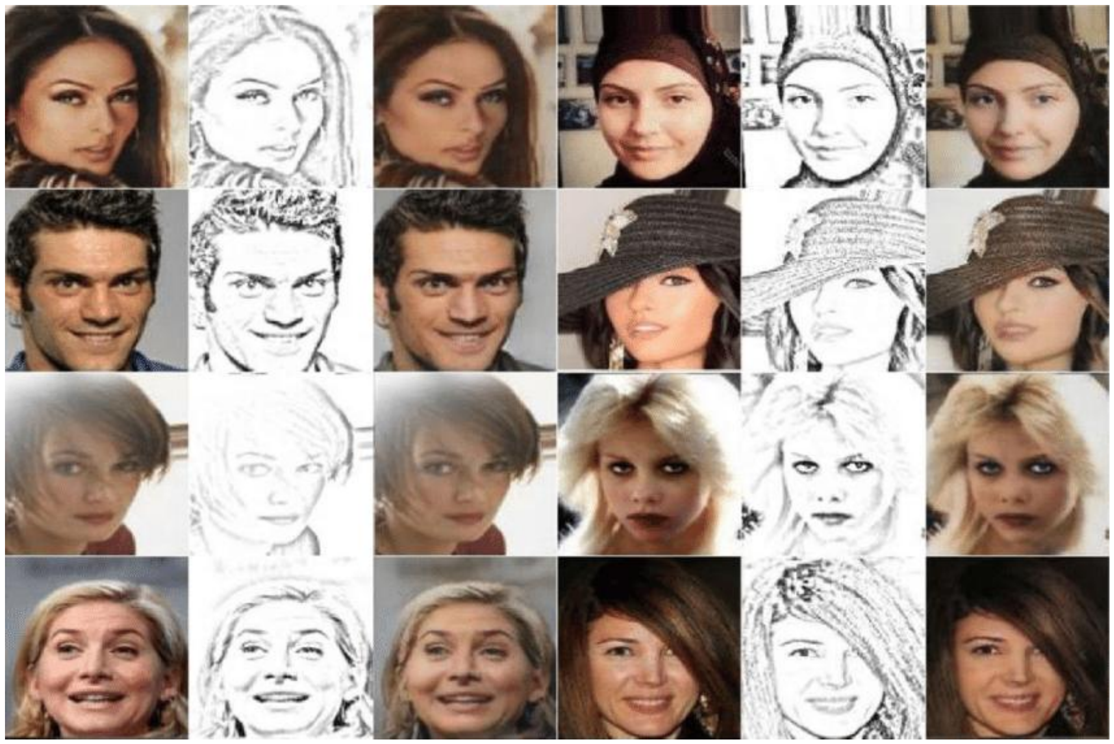
How GANs Work?

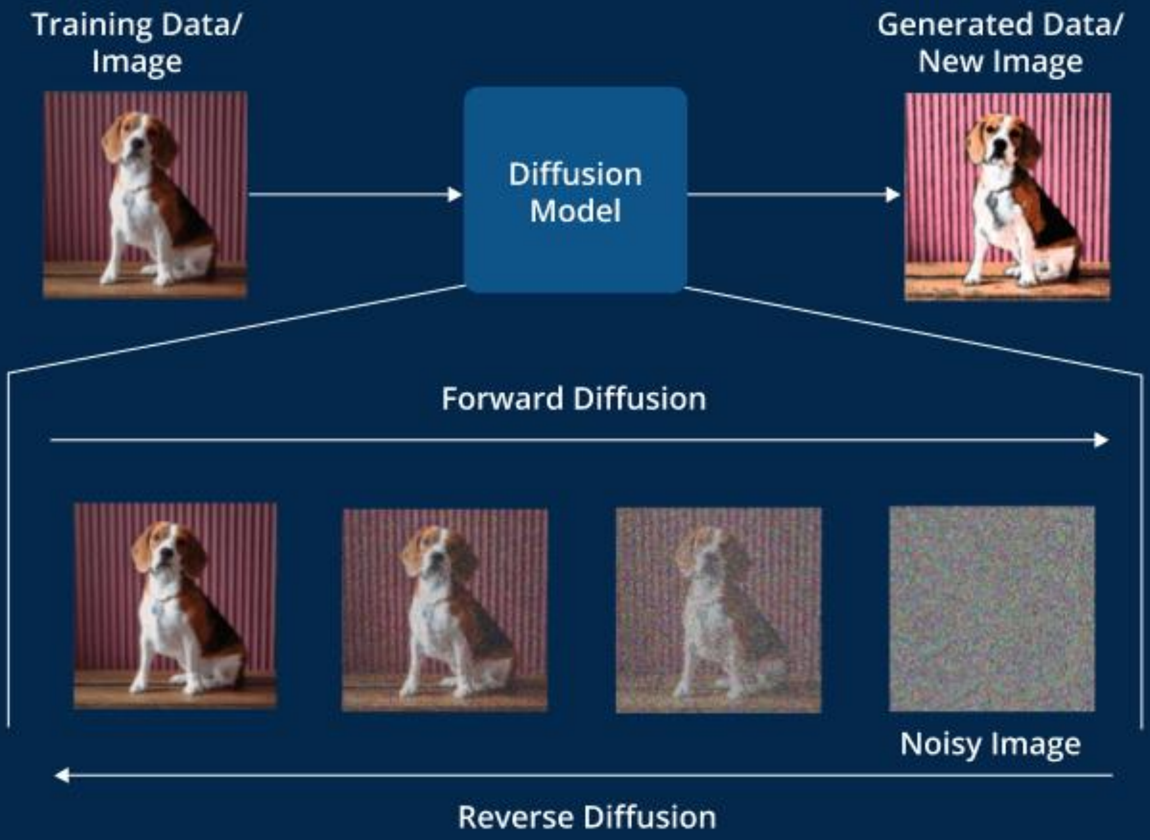


- The Generator and Discriminator are trained together in a zero-sum game.
- The Generator aims to produce data that can fool the Discriminator.
- The Discriminator aims to correctly identify real vs. fake data.
- Training continues until the Discriminator cannot reliably tell the difference

Applications of GANs

- Image generation (e.g., creating realistic photos of non-existent people).
- Image-to-image translation (e.g., converting sketches to photos).
- Data augmentation for training other models



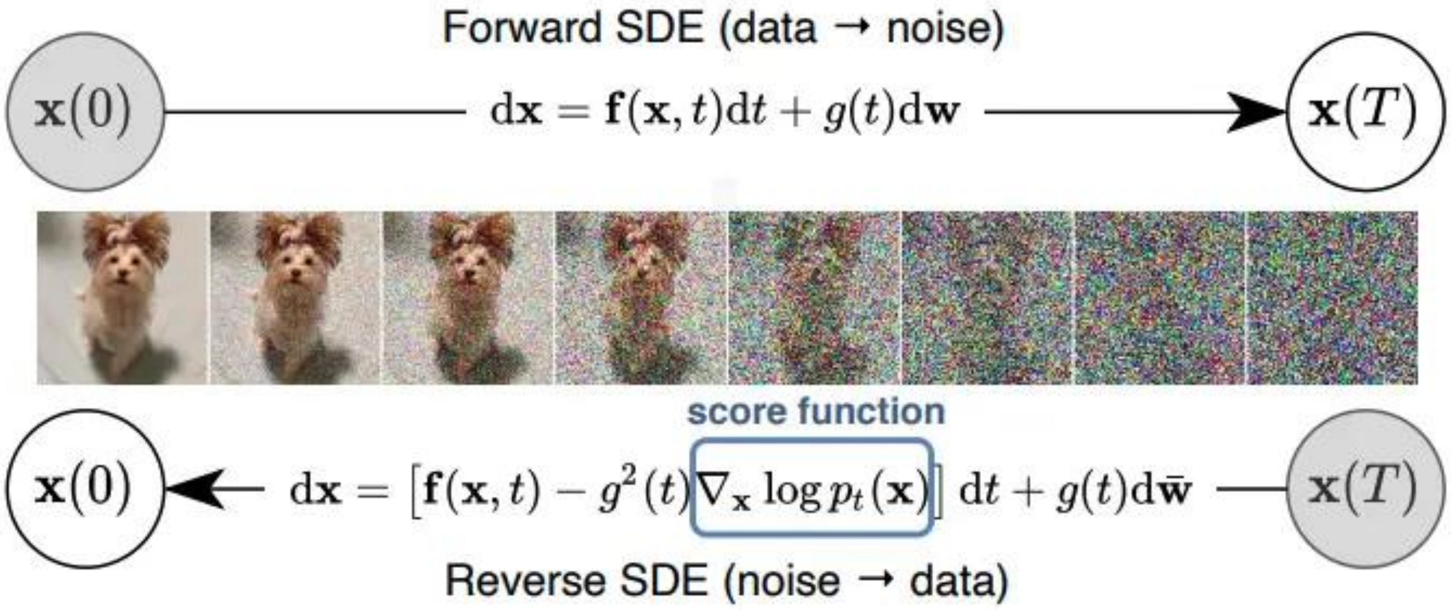


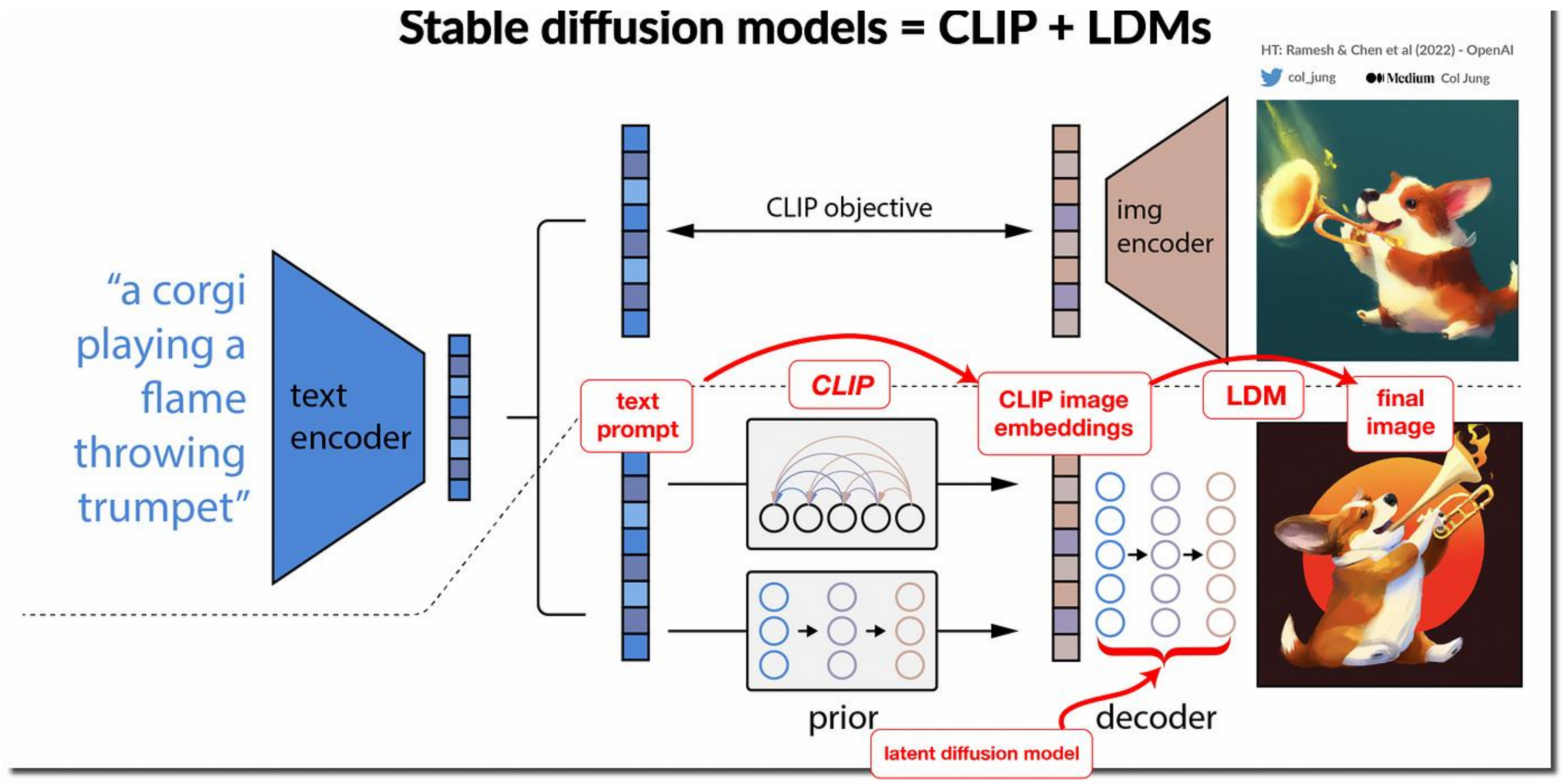
What are Diffusion Models?

- Diffusion Models are generative models inspired by non-equilibrium thermodynamics.
- They add noise to data in a forward process and learn to reverse this process to generate new data

How Diffusion Models Work?

- Forward diffusion process: Gradually adds Gaussian noise to data.
- Reverse diffusion process: Learns to remove the noise and reconstruct the data.
- Uses a Markov chain to model the noise addition and removal


Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

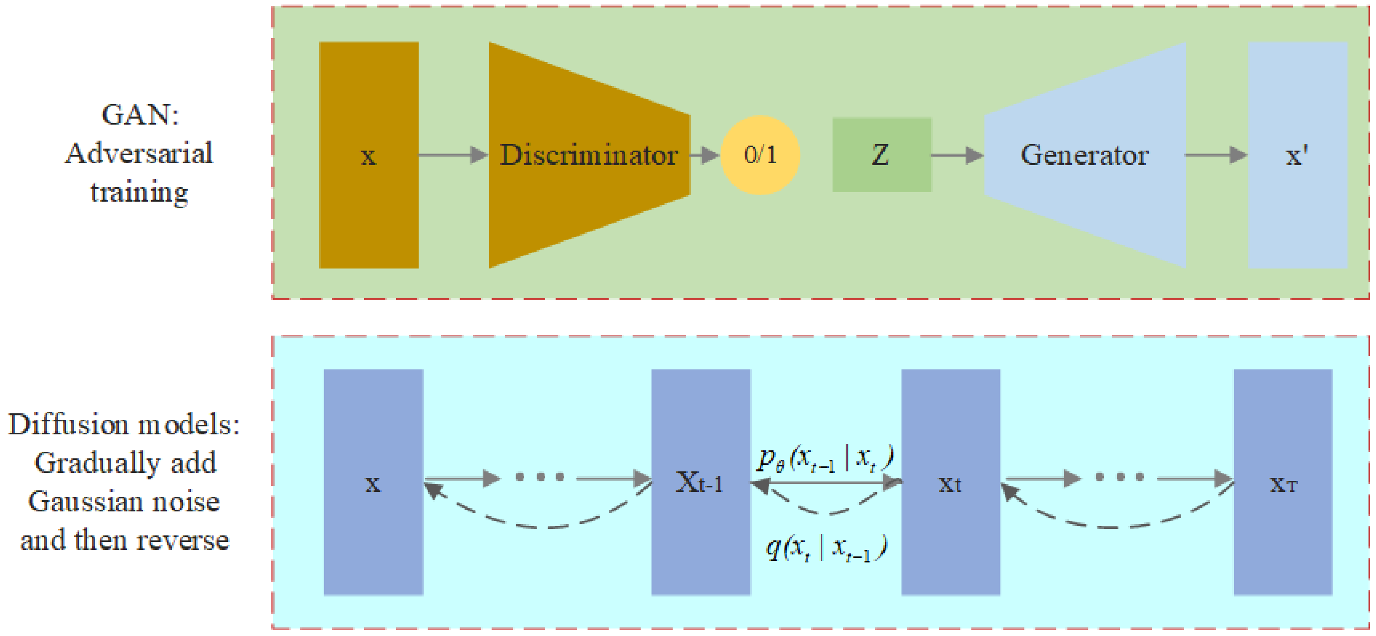


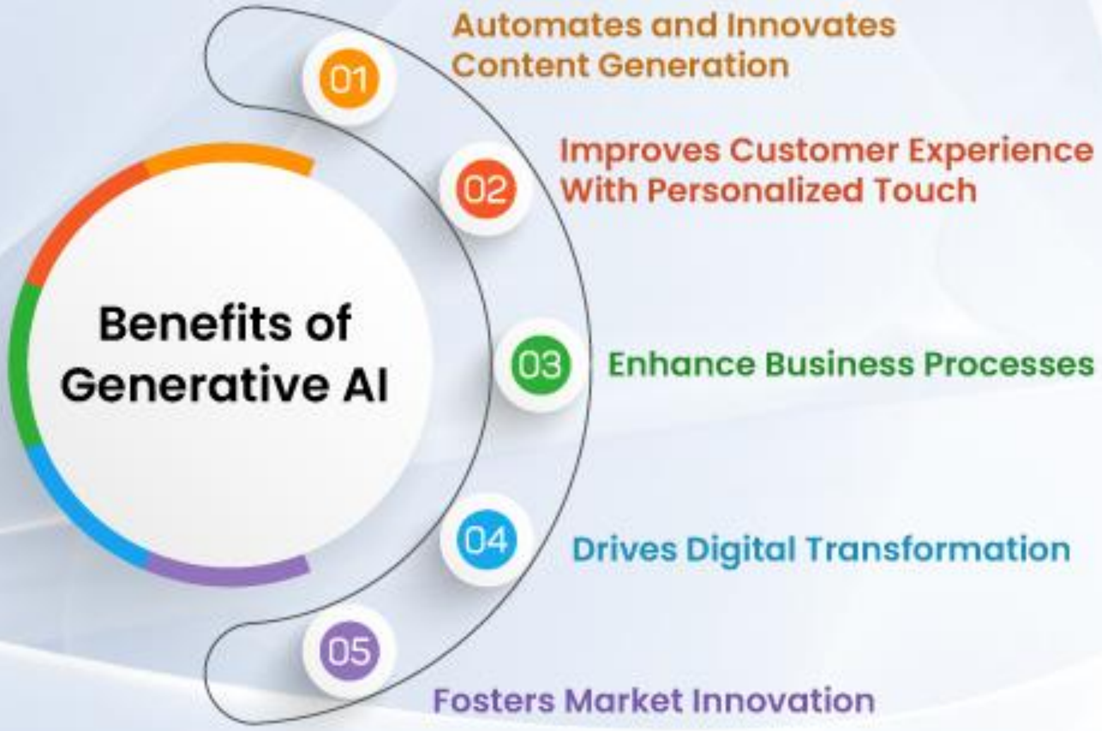
Applications of Diffusion Models

- High-quality image synthesis (e.g., DALL-E 2).
- Text-to-image generation.
- Potentially more stable and diverse than GANs

Comparison of GANs and Diffusion Models

- GANs: Adversarial training, potential instability, high-quality image generation.
- Diffusion Models: Noise-based training, stable, high-quality image generation.


GAN: Adversarial training
x
Discriminator
0/1
Z
Generator
x'
Diffusion models: Gradually add Gaussian noise and then reverse
x
Xt-1
$p_\theta(x_{t-1} | x_t)$
xt
$q(x_t | x_{t-1})$
xT


Benefits of Generative AI
01 Automates and Innovates Content Generation
02 Improves Customer Experience With Personalized Touch
03 Enhance Business Processes
04 Drives Digital Transformation
05 Fosters Market Innovation

Applications of Generative AI

- Content Creation: Marketing content, social media, text/image generation
- Synthetic Data Generation: Supports model training, safe data for privacy
- Medical Research: Drug discovery, protein structure prediction (AlphaFold)
- Art & Design: Artwork, music, and visual effects generation

Business Applications of Generative AI

- Business Process Automation: Intelligent customer service chatbots
- Efficiency Boost: Automates repetitive tasks
- Real world example: ChatGPT, DALLE in digital marketing

Challenges in Generative AI

- Bias and Fairness: Potential for biased content
- Intellectual Property Concerns: Copyright and plagiarism risks
- Misuse (Deepfakes): Risks in disinformation and identity theft
- Job Displacement: Automation’s impact on the workforce

The Future of Generative AI

- Advanced Applications: Automated design, multimodal AI, ethical AI
- Potential Fields: Fabrication, Autonomous Agents, Transparent AI
- Emerging Trends: Combined use of text, audio, and visual data for immersive experiences

Some basics of OpenCV

- OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

OpenCV

Some basics of OpenCV

First you need to install OpenCV:

```
!pip install opencv-python
```

The try:

```
import cv2

cv2.namedWindow("Image")

cv2.destroyWindow("Image")
```

Open an image

```
import cv2


img = cv2.imread(r'C:\Users\User user\Desktop\image1.jpg', 0)

cv2.imshow("image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

How to show the image in color?

Save an image

Convert and save a colored image to grayscale using OpenCV

```
import cv2

img = cv2.imread(r'C:\Users\User user\Desktop\image1.jpg')

gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imwrite(r'C:\Users\User user\Desktop\image1_gray.jpg', gray_img)
```

Syntax of OpenCV - line

```
cv2.line(img, pt1, pt2, color, thickness)
```

img: The image on which to draw.

pt1: Starting point of the line (x1, y1).

pt2: Ending point of the line (x2, y2).

color: Line color in BGR (e.g., (255, 0, 0) for blue).

thickness: Line thickness.

Syntax of OpenCV – rectangle

```
cv2.rectangle(img, pt1, pt2, color, thickness)
```

img: The image on which to draw.

pt1: Top-left corner of the rectangle (x1, y1).

pt2: Bottom-right corner of the rectangle (x2, y2).

color: Rectangle color in BGR.

thickness: Rectangle border thickness. Use -1 to fill the rectangle.

Syntax of OpenCV – circle

```
cv2.circle(img, center, radius, color, thickness)
```

img: The image on which to draw.

center: Center of the circle (x, y).

radius: Radius of the circle.

color: Circle color in BGR.

thickness: Circle border thickness. Use -1 to fill the circle.

Syntax of OpenCV – polyline

```
cv2.polylines(img, pts, isClosed, color, thickness)
```

img: The image on which to draw.

pts: Array of points defining the polyline.

isClosed: If True, the polyline is closed.

color: Polyline color in BGR.

thickness: Polyline thickness.

Example – draw a line and polyline

```
import cv2

import numpy as np


# Create a blank image

img = np.zeros((512, 512, 3), np.uint8)


# Draw a line

cv2.line(img, (0, 0), (511, 511), (255, 0, 0), 5)
```

Example – draw a line and polyline

```
# Draw a polyline

pts = np.array([[10, 5], [20, 30], [70, 20], [50, 10]], np.int32)

pts = pts.reshape((-1, 1, 2))

cv2.polylines(img, [pts], True, (0, 255, 255), 2)


# Display the image

cv2.imshow('image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Class exercise 1 – draw a rectangle and a circle

Use what you have just learned, draw a rectangle and a circle. You can decide their sizes and positions.

Syntax of OpenCV – write text

cv2.putText(image, text, position, font, font_scale, color, thickness)

image: The image on which you want to draw the text.

text: The string of text you want to write.

org: The bottom-left corner of the text string in the image.

font: The font type (e.g., cv2.FONT_HERSHEY_SIMPLEX).

fontScale: The scale factor that is multiplied by the font-specific base size.

color: The color of the text in BGR (Blue, Green, Red) format.

thickness: The thickness of the lines used to draw the text.

lineType: The type of line (e.g., cv2.LINE_AA for anti-aliased line).

Example – write text

cv2.putText(image, text, position, font, font_scale, color, thickness)

image: The image on which you want to draw the text.

text: The string of text you want to write.

org: The bottom-left corner of the text string in the image.

font: The font type (e.g., cv2.FONT_HERSHEY_SIMPLEX).

fontScale: The scale factor that is multiplied by the font-specific base size.

color: The color of the text in BGR (Blue, Green, Red) format.

thickness: The thickness of the lines used to draw the text.

lineType: The type of line (e.g., cv2.LINE_AA for anti-aliased line).

Example – write text

```
import cv2

image = cv2.imread(r'C:\Users\User user\Desktop\image1.jpg’)

text = "sunrise photo“

position = (250, 50)

font = cv2.FONT_HERSHEY_SIMPLEX

font_scale = 1

color = (0, 255, 0)  # green color in BGR

thickness = 2
```

Example – write text

```
line_type = cv2.LINE_AA

cv2.putText(image, text, position, font, font_scale, color, thickness, line_type)

cv2.imshow('Image with Text', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Face detection

CascadeClassifier in OpenCV is used for object detection, such as detecting faces in an image.

The syntax for using **CascadeClassifier** with the **Haar cascade file** for frontal face detection is as follows:

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml’)
```

Face detection

```
import cv2

# Load the pre-trained Haar cascade for frontal face detection

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')


# Read the image

image = cv2.imread(r'C:\Users\User user\Desktop\Obama.jpg')


# Convert the image to grayscale (Haar cascades work better on grayscale images)

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Face detection

```
# Detect faces in the image

faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))


# Draw rectangles around the detected faces

for (x, y, w, h) in faces:

        cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)


# Display the image with detected faces

cv2.imshow('Image with Detected Faces’, image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

Checklist

- Can you:
  1. Describe NLP?
  2. Introduce generative AI?
  3. Apply OpenCV in image processing?