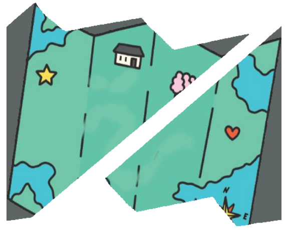
Workshop on Generative AI

Elan & nVision
2026

Plan of Action

Day 1	Session 1	Introduction to VAE, GANs, Diffusion models	Ambarish Parthasarathy
	Session 2	Introduction to Gen AI, LLMs, Prompt Engineering, LangChain	Athira Krishnan R
Day 2	Session 3	Introduction to RAG, Agents and Agentic AI	Athira Krishnan R Ambarish Parthasarathy



Session 1

Introduction to VAE, GANs, Diffusion models

Handled by,

Ambarish Parthasarathy
AI PhD scholar
IIT Hyderabad

Introduction

Gentle Recap to Convolution Neural Networks

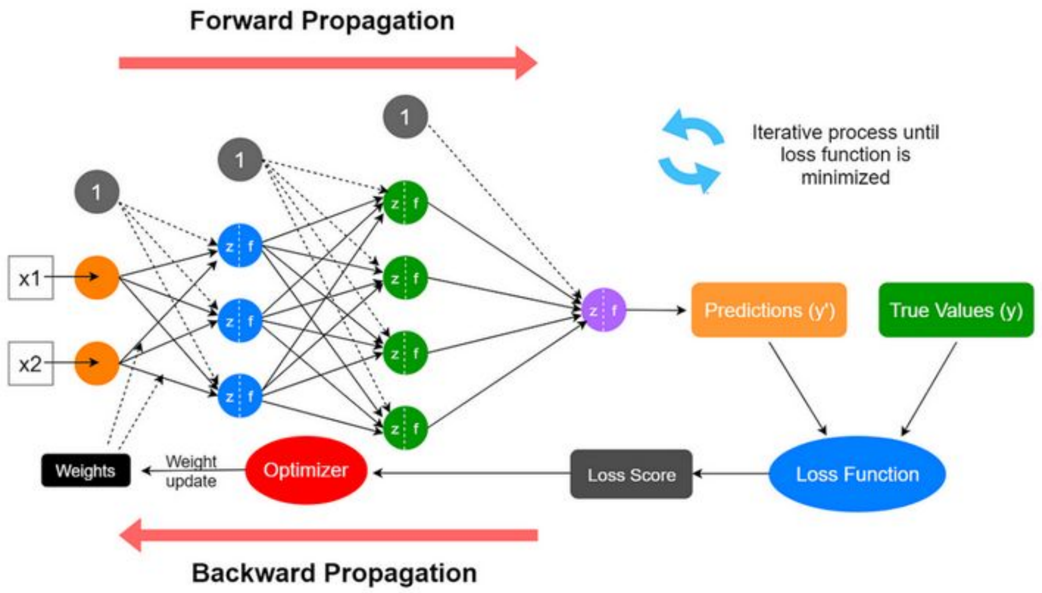
- Categories and the learning principles.
- Strategies of Drop Out and Augmentation.

AutoEncoders and the variants

Generative Adversarial Networks (GANs)

Diffusion Models

Recap to CNNs



A simple example of a neuron network.

The iterative flow of loss and gradient updates.

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

Convolution Neural Networks (CNNs)

What do you see in here ?!

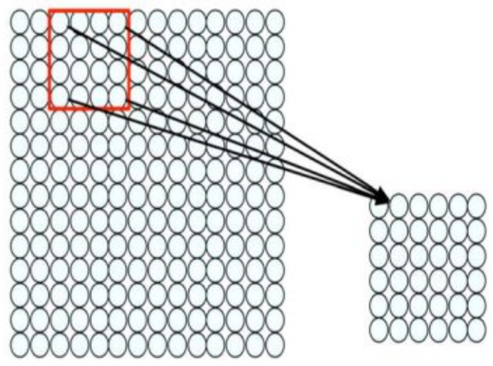
Convolution Neural Networks (CNNs)

Input is a 2D/3D image, array of pixel values.

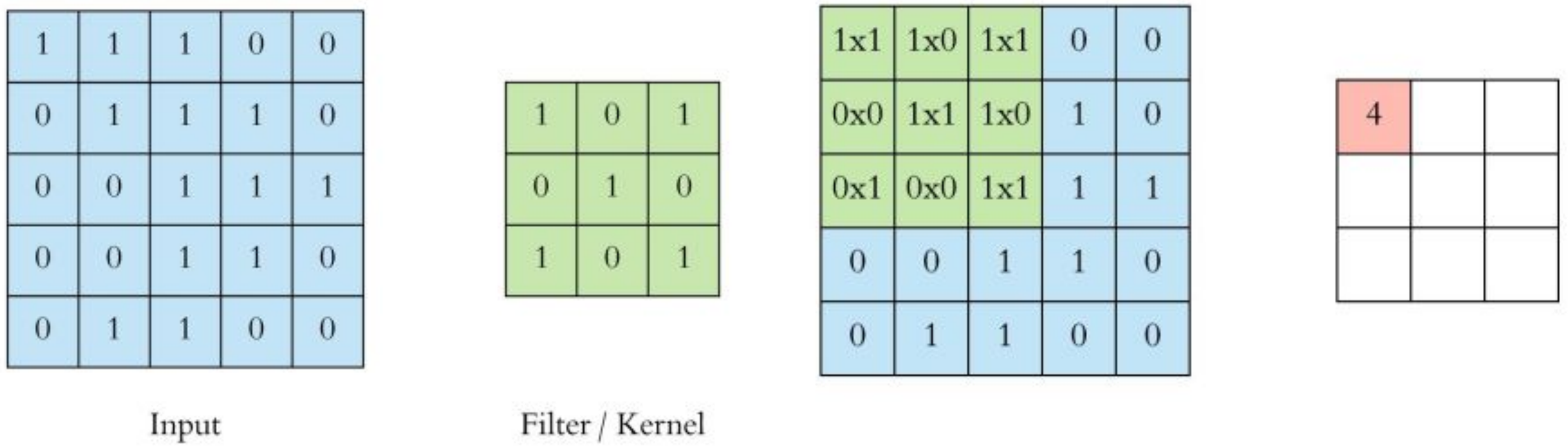
Approach: consider a patch, and connect it to neurons in a hidden layer.

Apply a filter of a particular size, shift it by a given unit; patchy operation of convolution.

Helps to extract the local features.



CNN; Convolution Kernel



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

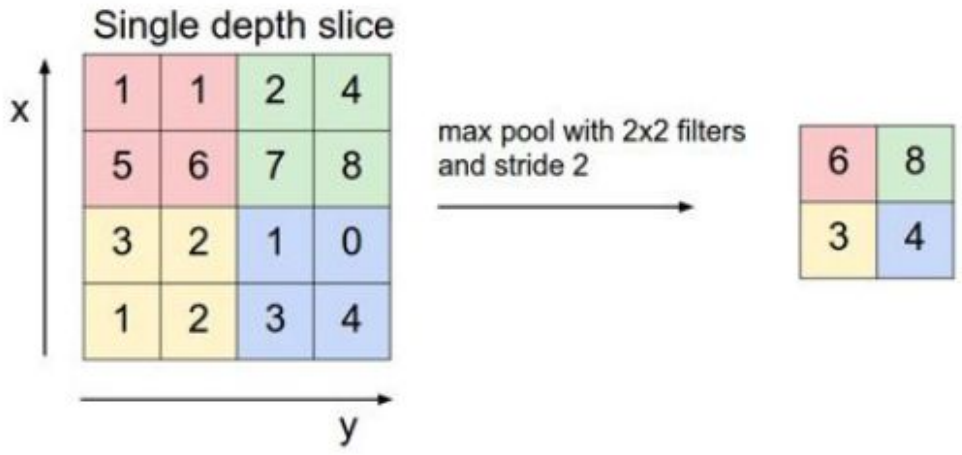
1	0	1
0	1	0
1	0	1

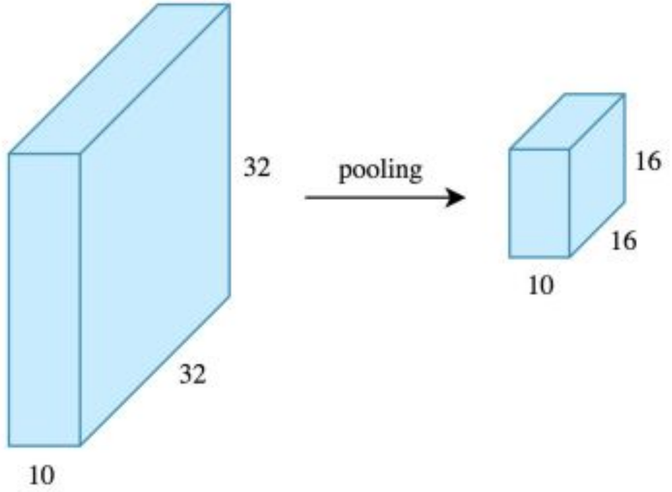
Filter / Kernel

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

CNNs; Max Pooling Stage

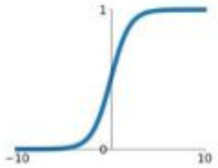




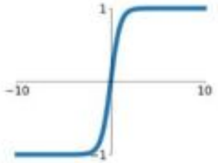
The dominant feature is selected in here.

CNNs; Activation Functions

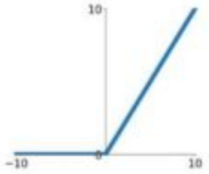
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



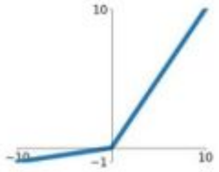
tanh
$\tanh(x)$



ReLU
$\max(0, x)$

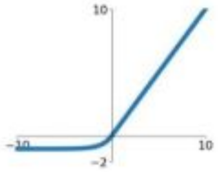


Leaky ReLU
$\max(0.1x, x)$

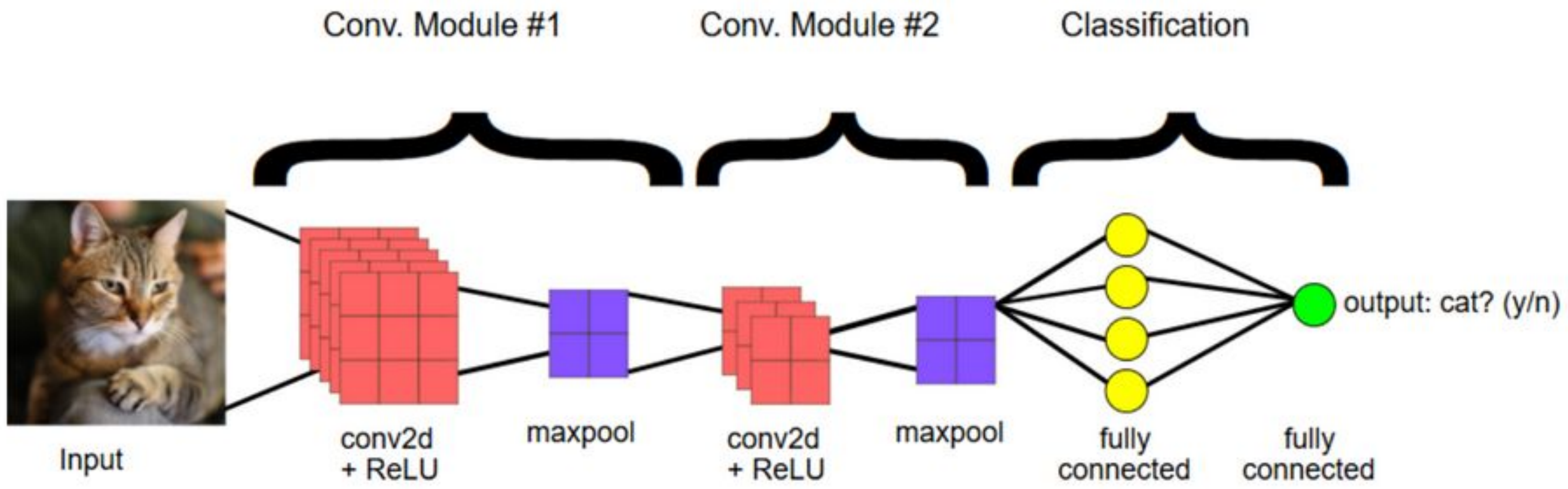


Maxout
$\max(w_1^T x + b_1, w_2^T x + b_2)$

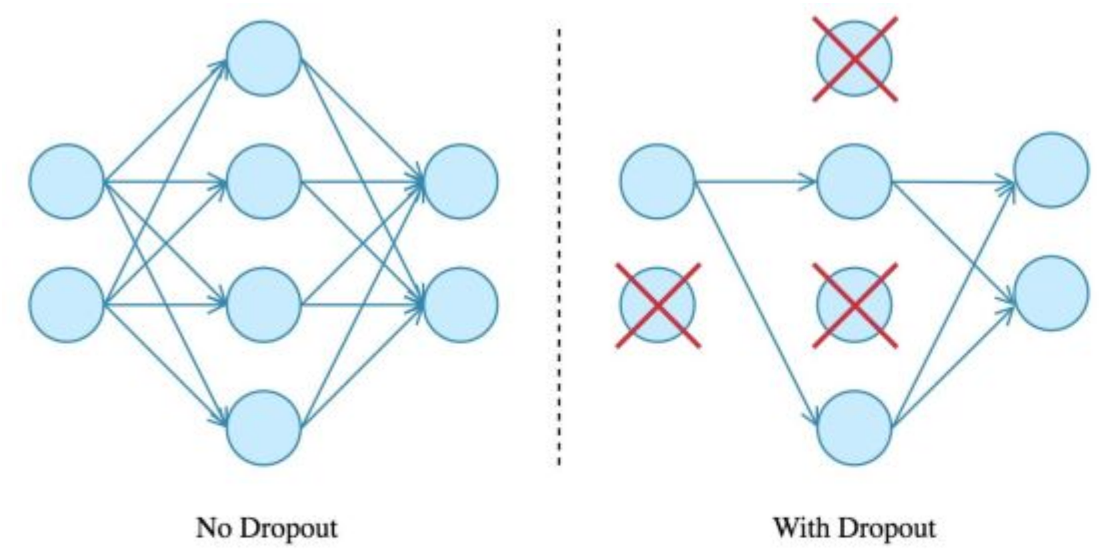
ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



CNN; Network


Conv. Module #1
Conv. Module #2
Classification
Input
conv2d + ReLU
maxpool
conv2d + ReLU
maxpool
fully connected
fully connected
output: cat? (y/n)

CNN; Strategies used

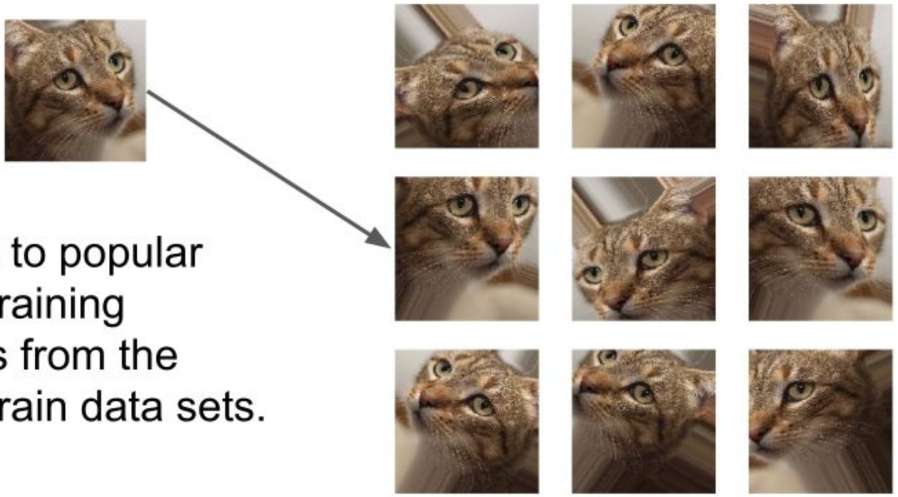


Strategy of Dropout

-Random removal of nodes.

-An indirect regularization technique reducing the overfitting phenomenon.

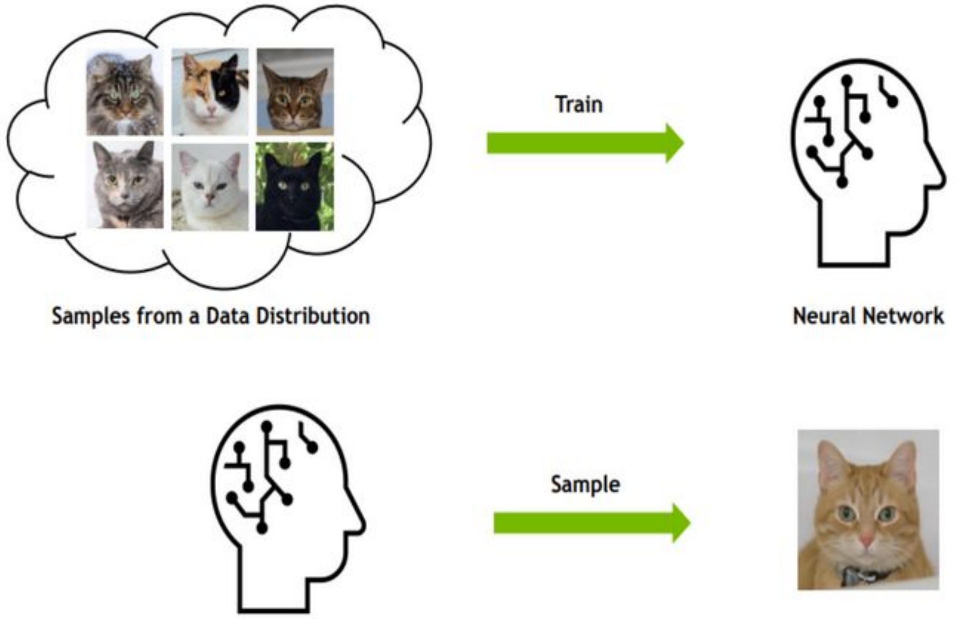
CNNs; Data Augmentation (DA)



DA helps to popular artificial training instances from the existing train data sets.

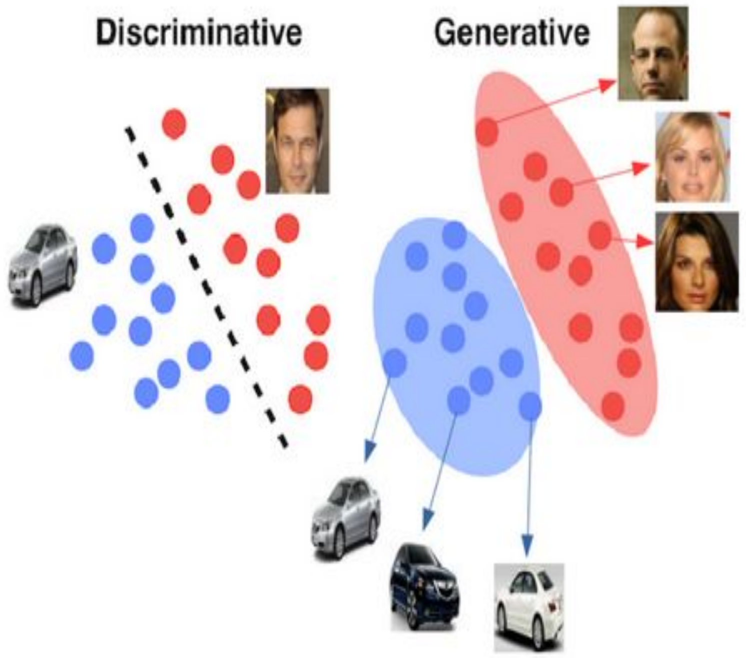
- Helps in duplicate of the <u>same</u> data.
- Solving the invariance issue for models.
- **INDUCES MORE BIAS INTO THE MODEL.**

Idea of data generation



- we have learnt a network with the given data.
- can the learnt network generate a data from the features that it is extracted and learnt ?

Generation Model against Discriminative Model



Discriminative Model; learns the boundary between the corresponding class; (SVMs, regressions)

Generative Modelling; model a distribution similar to the true distribution, from which the training samples were generated.

objectives and criterions to met

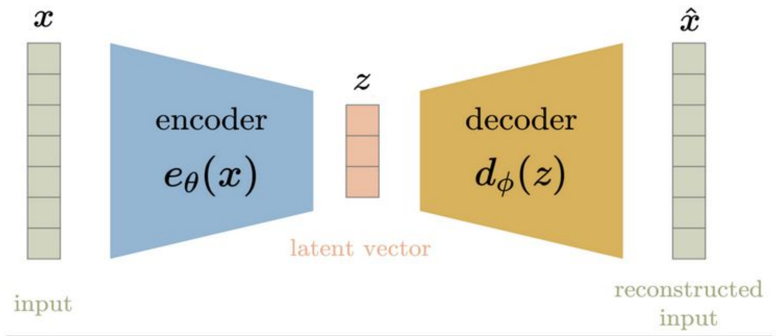
Diverse output

Realistic output

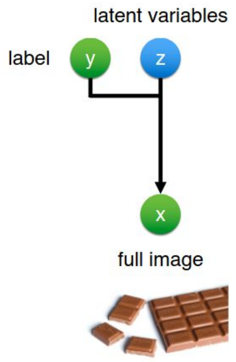
Effective method of learning

Good quality of generated data

AutoEncoders and Variational AutoEncoders



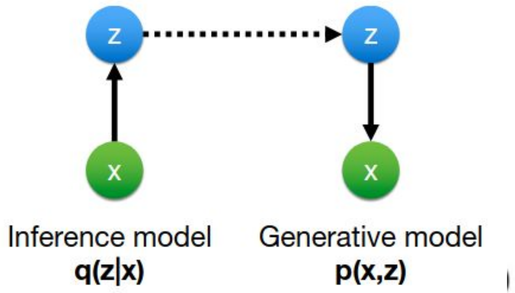
$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$



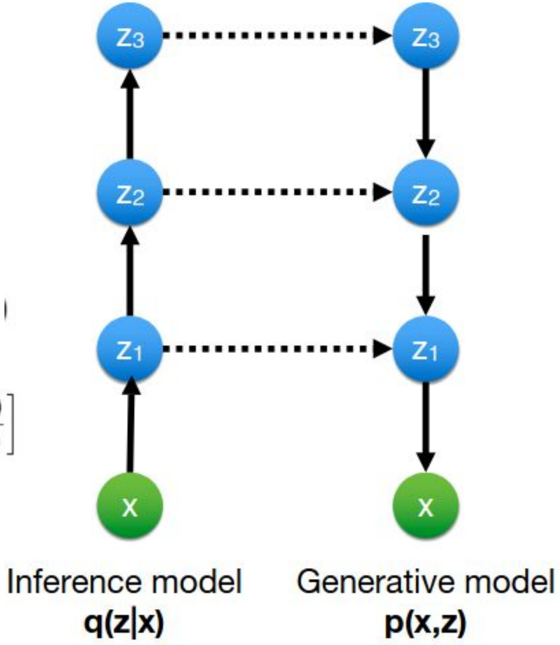
$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \\ p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z}) &= \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\theta(\mathbf{z},\mathbf{y}), \sigma^2\mathbf{I}) \\ p_\theta(\mathbf{x}|\mathbf{y}) &= \int_{\mathbf{z}} p(\mathbf{x}|\mathbf{y},\mathbf{z})p(\mathbf{z})\,d\mathbf{z} \\ &= \text{flexible distribution} \end{aligned}$$

- but the distribution of *p(x)* is not easy to compute.
- Thus an inference/intermediate modelling is done, to indirectly compute it.

VAEs; intermediate modelling

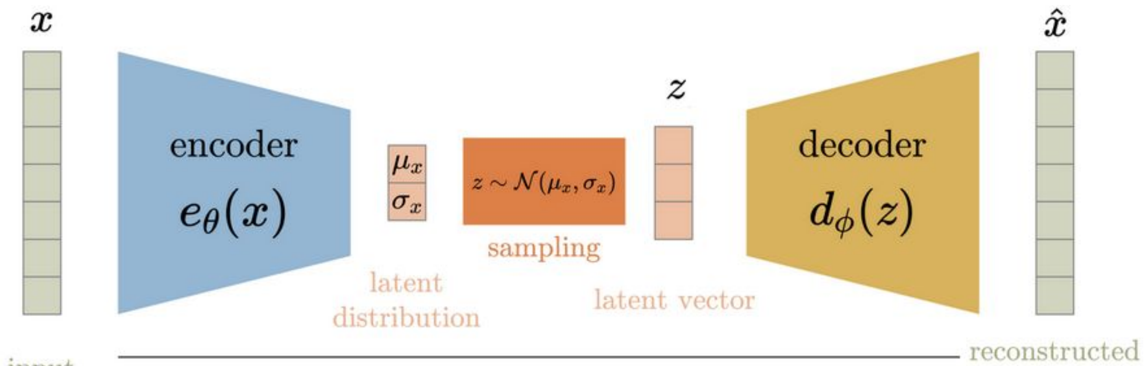


$$\log p_{\theta,\phi}(\mathbf{x}) \geq \text{ELBO}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]$$



- The optimization strategy of ELBO (Evidence Lower BOund) is used.
- A more hierarchical approach is used in here to improve generation ability.

VAEs; Objective



$$\text{reconstruction loss} = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$similarity\ loss = KL\ Divergence = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$loss = reconstruction\ loss + similarity\ loss$$

- an additional similarity loss gets added up in the case of variational autoencoders.

VAEs; cons

It is easy to optimize for smaller models; difficult if the model depth gets increased.

Sensitive to hyperparameter setups; leads to training instability.

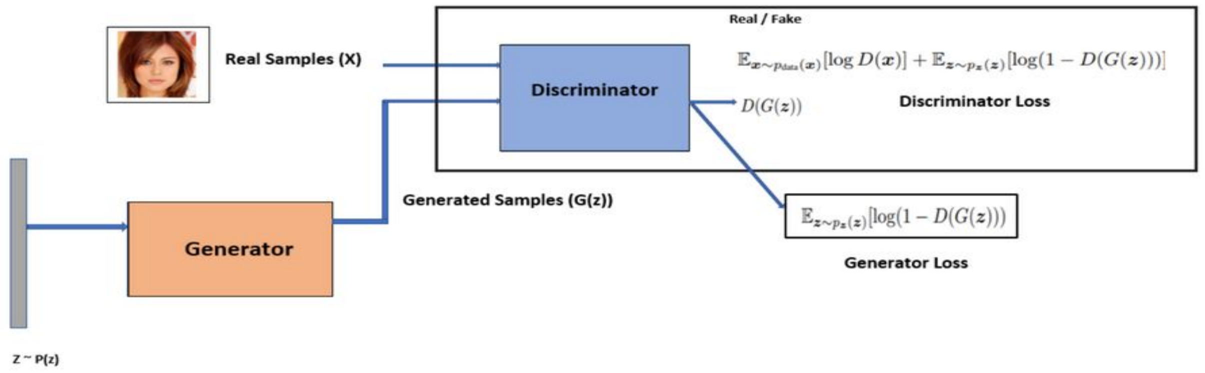
If the similarity score goes to zero, VAE turns into simple autoencoder, thus losing the ability of generation.

Generative Adversarial Networks (GANs)

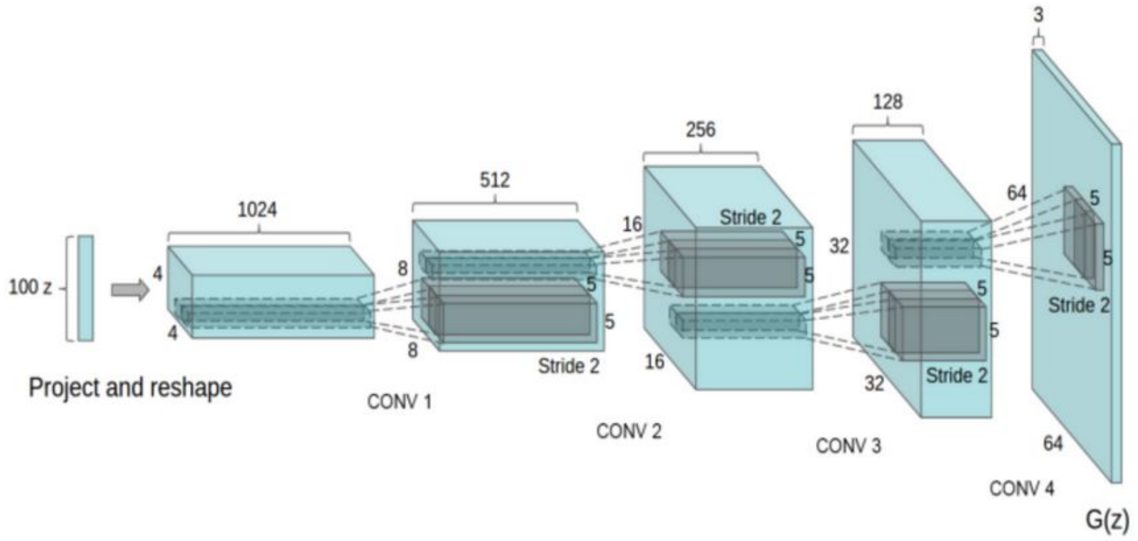
A min max game played by two components namely *generator* and *discriminator*.

Generator - generates an image from the given latent representation.

Discriminator - judges the generated image of the generator block.


Real / Fake
Real Samples (X)
Discriminator
$\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$
$D(G(z))$
Discriminator Loss
Generated Samples (G(z))
Generator
$\mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$
Generator Loss
Z ~ P(z)

GANs; Generator Component



The generator component of GANs

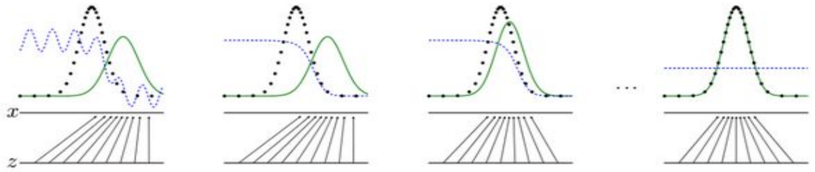
Objective of GANs

D and G play the following two player min-max game with value function $V(G; D)$:

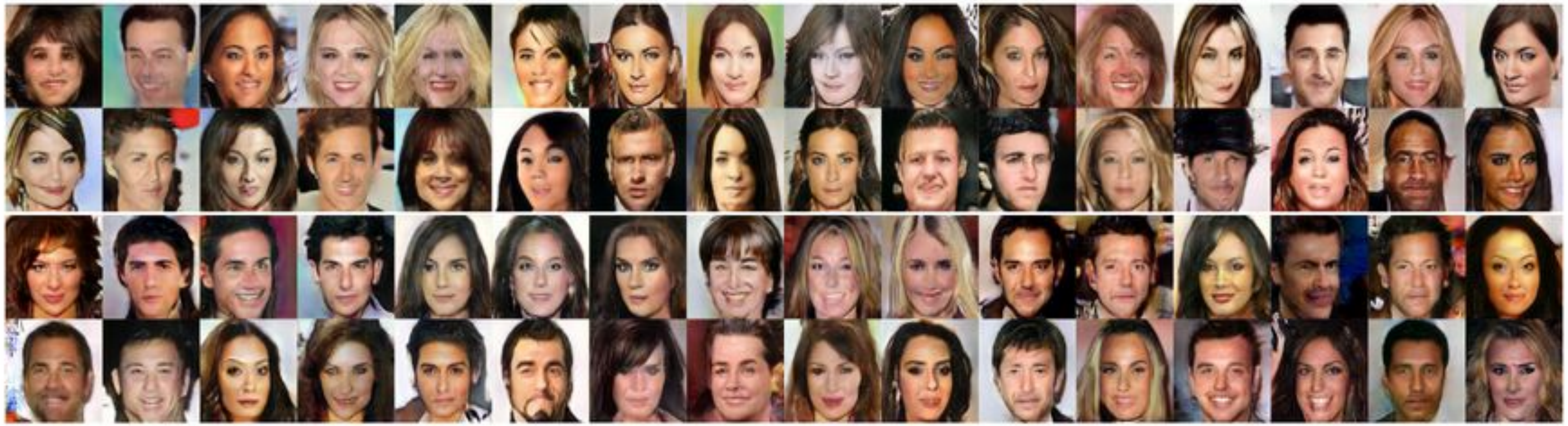
$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[log(D(x))] + E_{z \sim p_z(z)}[log(1 - D(G(z)))]$$

Discriminator D(x) is trained to discriminate the true and real samples and it will converge to $D^*(x)$.

$$D^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$



Qualitative Observations of GANs



Variants of GANs

Different variants of GANs, on grounds of optimization improvements such as WGANs (works on a wasserstein distance metric), improved WGANS-GP.

Despite the generating capabilities,

- Mode Collapse; underexplored regions of latent space, only concentrates and generates a particular sets only.

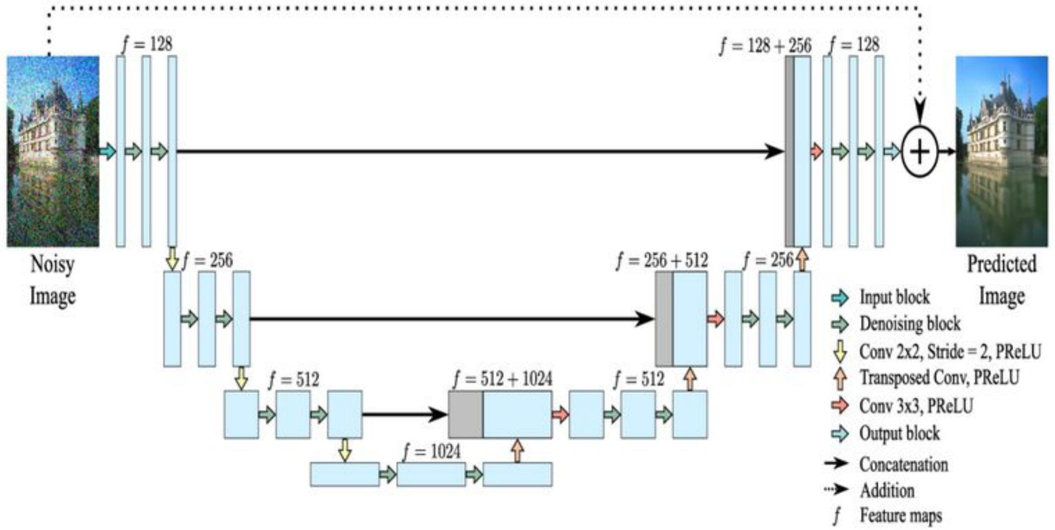
Introduction to Diffusion Models

A concept that is inspired from the topic of Constructive Imaging.

Learning to generate with the help of denoising.

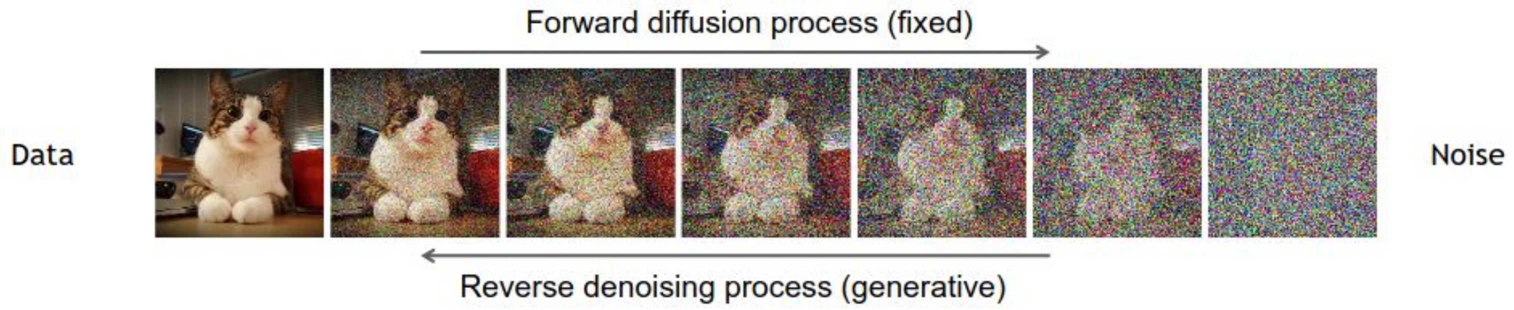
1. Forward phase of gradual addition of noise to the input
   a. This is a fixed phase.
2. Reverse phase of learning to generate the data from the noisy inputs.
   a. The learning phase of Diffusion models.

Diffusion Models



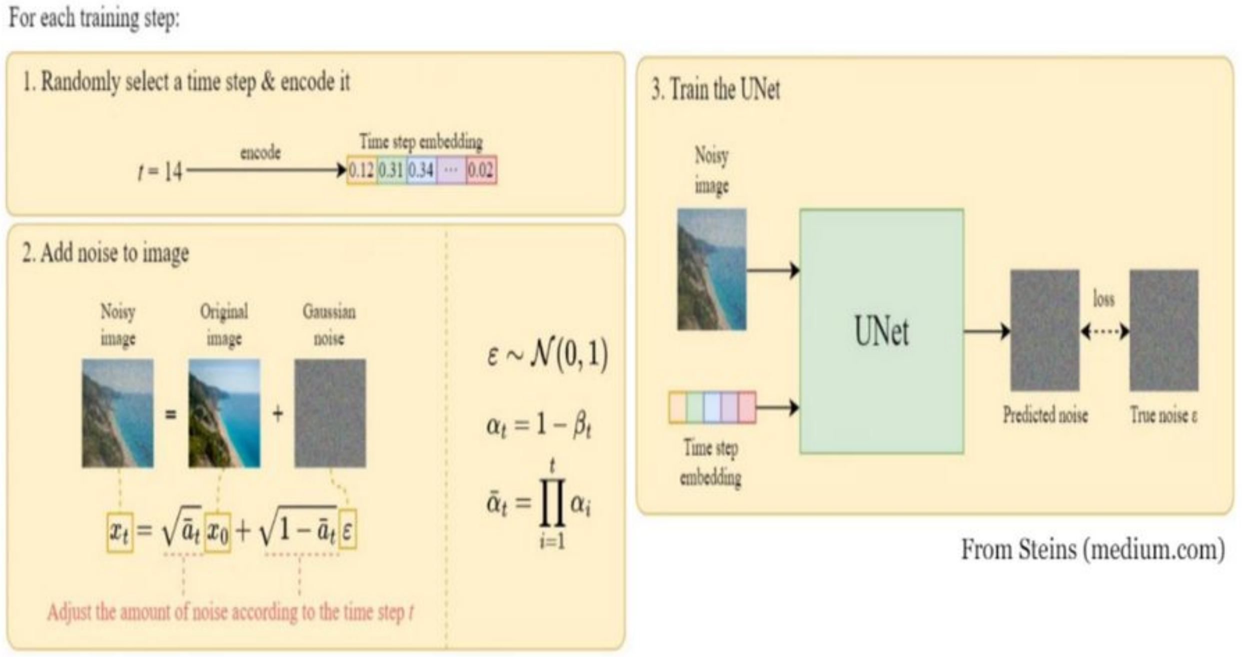
- A similar autoencoder approach where the noise is added by encoder and the learnt by the decoder.
- A skip connection is included in here.

Forward & Reverse process; Diffusion Models


Forward diffusion process (fixed)
Data
Noise
Reverse denoising process (generative)

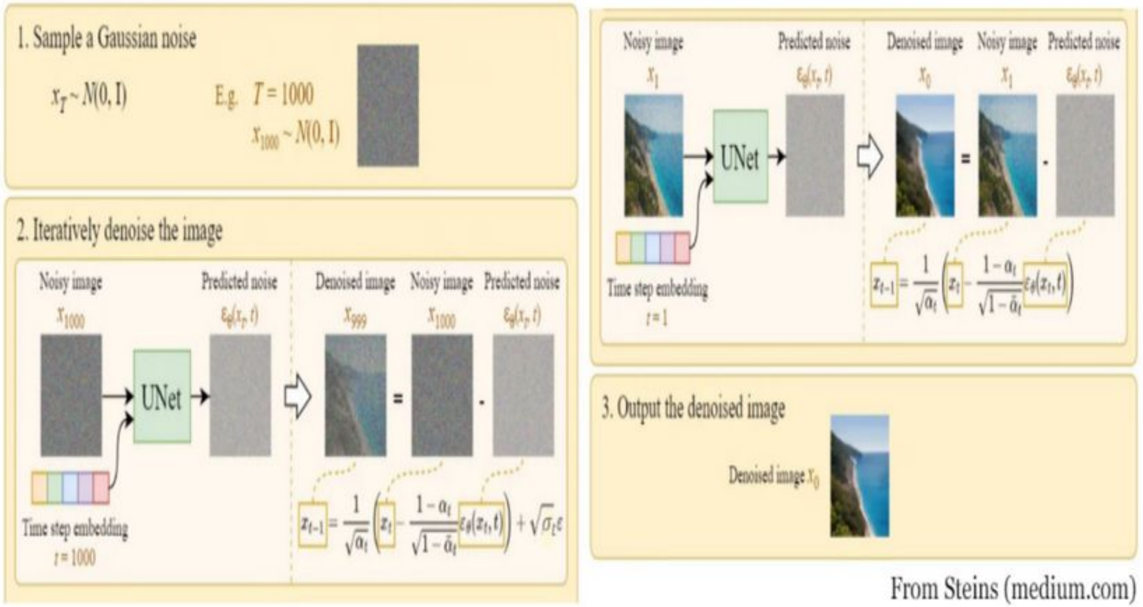
The forward (handled by encoder) and the backward (handled by decoder) processes.

Diffusion Models; Training mechanism

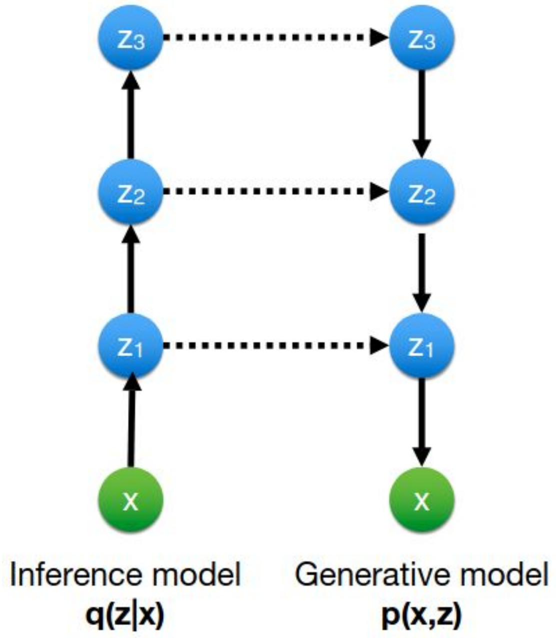

For each training step:
1. Randomly select a time step & encode it
$t = 14$
encode
Time step embedding
0.12 0.31 0.34 ··· 0.02
2. Add noise to image
Noisy image
Original image
Gaussian noise
$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon$
Adjust the amount of noise according to the time step t
$\varepsilon \sim \mathcal{N}(0,1)$
$\alpha_t = 1 - \beta_t$
$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$
3. Train the UNet
Noisy image
UNet
Time step embedding
Predicted noise
loss
True noise ε

From Steins (medium.com)

Generation Stage; Diffusion


1. Sample a Gaussian noise
$x_T \sim N(0, I)$
E.g. $T = 1000$
$x_{1000} \sim N(0, I)$
2. Iteratively denoise the image
Noisy image
x_{1000}
UNet
Predicted noise
$\epsilon_\theta(x_t, t)$
Time step embedding
$t = 1000$
Denoised image
x_{999}
Noisy image
x_{1000}
Predicted noise
$\epsilon_\theta(x_t, t)$
$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right) + \sqrt{\sigma_t}\epsilon$
Noisy image
x_1
UNet
Predicted noise
$\epsilon_\theta(x_t, t)$
Time step embedding
$t = 1$
Denoised image
x_0
Noisy image
x_1
Predicted noise
$\epsilon_\theta(x_t, t)$
$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$
3. Output the denoised image
Denoised image x_0
From Steins (medium.com)

Diffusion Model related with VAE



- Similar to the VAEs with some minor differences:
  - The inference model is fixed.
  - The ELBO algo can now be decomposed, thus can handle more deeper structures.
- No issues of mode collapse, can handle more deeper structures.

Session 2

Introduction to Gen AI, LLMs, Prompt Engineering, LangChain

Handled by,

Athira Krishnan R
AI PhD scholar,
IIT Hyderabad
GSAC Advisor & Treasurer,
IEEE RAS Kerala Chapter

Timeline

2000	**MIS Reporting**
2005	**Predictive modelling**
2014	**ML**
2018	**DL**
2023	**Gen AI**
2025	**Agentic AI**

Applications

GenAI

- Content creation
- Code snippet generation
- Language translation
- Email and assistants

Agentic AI

- AI customer support
- Research agents
- Business process automation
- Coder agents

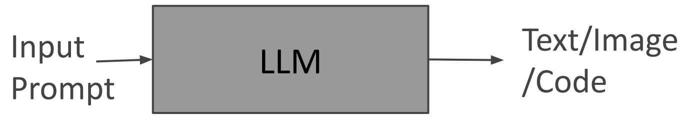
Gen AI

Generates output from input prompt.

ChatGPT, DALLE, Gemini

Low Complexity

Limitation: Limited knowledge



LLM

100 Experts in your pocket

Imagine you have an expert to,

- Correct your **English grammar**.
- Think like an **expert philosopher** to offer deep insights.
- Expert **gym trainer** to motivate your fitness goals.
- Expert **interview panel** to give expert advice.
- A **lawyer** to explain the legal concepts.
- A **maths teacher** to simplify the concepts.

What is an LLM?

Large Language Model

Model is anything that takes input from a user and gives prediction as output.

Language models takes language as input like sentiment analysis model that takes user reviews as input, next best word prediction in whatsapp chat. These are small Language models which does a specific job.

Large language model takes a sequence of words as input and predicts the next word in the sequence.

Example: Llama by Facebook, ChatGPT by Open AI, Gemini by Google, Grok by X, Co pilot by Microsoft.

Why “Large”?

- Large language models are capable of understanding and generating more complex and human like language
- 3 reasons why they are called “Large”
    - Large amount of data.
        - Books, literature database, scientific journals, websites, blogs, social media forums, news, dictionaries,open source datasets and repositories.
    - Large amount of parameters.
        - Billions/Trillions of parameters
    - Large scale computations.

Parameters count in LLMs

Model	Year	Parameter	Notable features
GPT	2018	110M	First large scale transformer model
GPT 2	2019	1.5B	Improved transformer model
GPT 3	2020	175B	Capable of few shot learning
DALL.E 2	2022	22B	Improved DALL.E

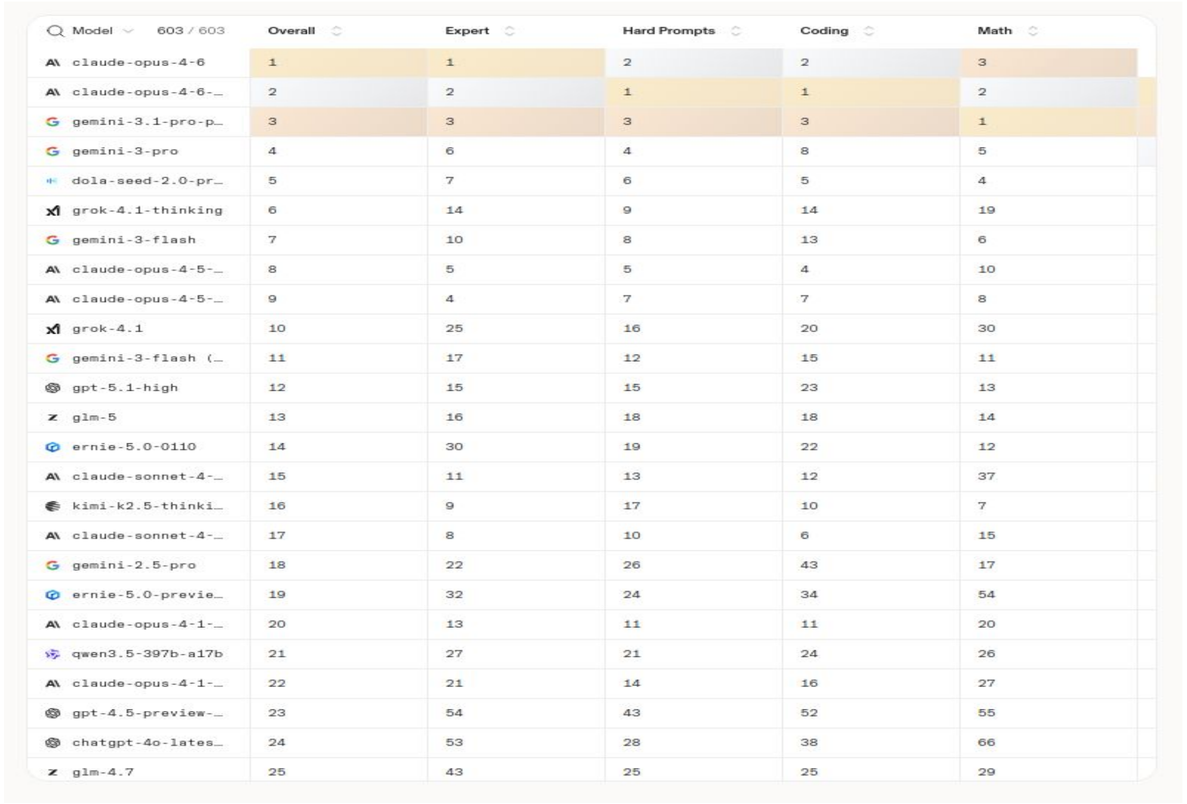
Results of LLMs

LLMs can understand, interpret and generate human like text.

Few notable applications of LLMs

- Generating code
- Creative content generation
- Sophisticated Q&A

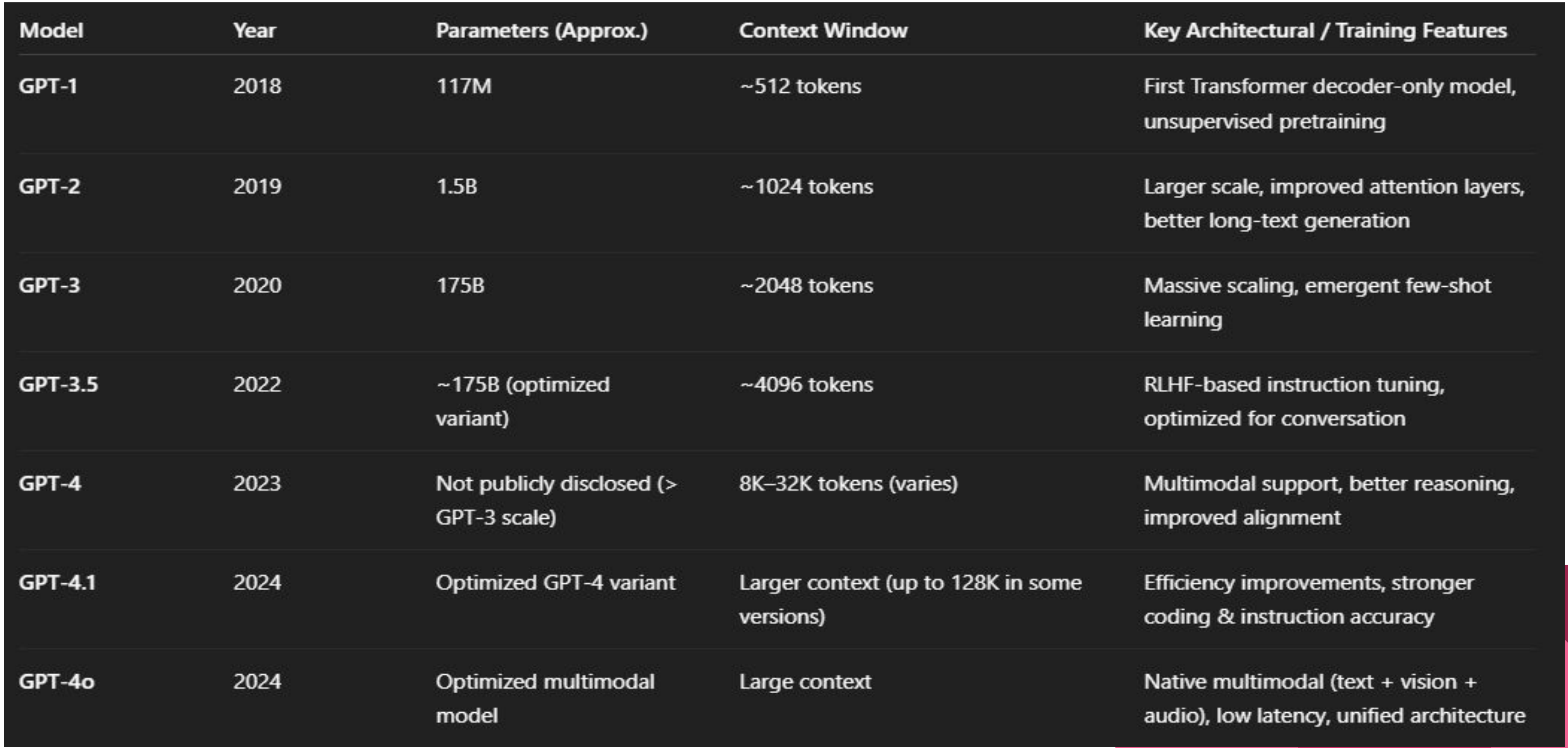
Comparing Top LLMs



Model 603 / 603	Overall	Expert	Hard Prompts	Coding	Math
claude-opus-4-6	1	1	2	2	3
claude-opus-4-6-…	2	2	1	1	2
gemini-3.1-pro-p…	3	3	3	3	1
gemini-3-pro	4	6	4	8	5
dola-seed-2.0-pr…	5	7	6	5	4
grok-4.1-thinking	6	14	9	14	19
gemini-3-flash	7	10	8	13	6
claude-opus-4-5-…	8	5	5	4	10
claude-opus-4-5-…	9	4	7	7	8
grok-4.1	10	25	16	20	30
gemini-3-flash (…	11	17	12	15	11
gpt-5.1-high	12	15	15	23	13
glm-5	13	16	18	18	14
ernie-5.0-0110	14	30	19	22	12
claude-sonnet-4-…	15	11	13	12	37
kimi-k2.5-thinki…	16	9	17	10	7
claude-sonnet-4-…	17	8	10	6	15
gemini-2.5-pro	18	22	26	43	17
ernie-5.0-previe…	19	32	24	34	54
claude-opus-4-1-…	20	13	11	11	20
qwen3.5-397b-a17b	21	27	21	24	26
claude-opus-4-1-…	22	21	14	16	27
gpt-4.5-preview-…	23	54	43	52	55
chatgpt-4o-lates…	24	53	28	38	66
glm-4.7	25	43	25	25	29

Image Credits: https://arena.ai/leaderboard

ChatGPT



Model	Year	Parameters (Approx.)	Context Window	Key Architectural / Training Features
GPT-1	2018	117M	~512 tokens	First Transformer decoder-only model, unsupervised pretraining
GPT-2	2019	1.5B	~1024 tokens	Larger scale, improved attention layers, better long-text generation
GPT-3	2020	175B	~2048 tokens	Massive scaling, emergent few-shot learning
GPT-3.5	2022	~175B (optimized variant)	~4096 tokens	RLHF-based instruction tuning, optimized for conversation
GPT-4	2023	Not publicly disclosed (> GPT-3 scale)	8K–32K tokens (varies)	Multimodal support, better reasoning, improved alignment
GPT-4.1	2024	Optimized GPT-4 variant	Larger context (up to 128K in some versions)	Efficiency improvements, stronger coding & instruction accuracy
GPT-4o	2024	Optimized multimodal model	Large context	Native multimodal (text + vision + audio), low latency, unified architecture

2 ways to interact with LLMs

1. Use the web interface / app provided by them-> User
2. **Using an API -> Developer**

Prompt Engineering

Input text to LLM’s is known as prompt.

Quality of the results you get is directly proportional to the quality of the prompt you use.

Chat GPT prompt is not same as a Google search string

Avoid brief small queries or phrases typically used in google searches.

You need to structure your prompt.

How to prompt

Avoid Zero shot prompt (1 line prompts)

Elements in a structured prompt

1. **Roleplaying**: Character insight
2. **Example**: Exceptions/ Inclusions
3. **Domain and context**: Background/setup
4. **Display of output**: Structure and format
5. **Iterations**: Refinements/ Regenerations

Try these

1. Write a short blog post about why time management is important
2. Write a mail to your boss about the salary increment.

LangChain

LLM Framework

It facilitates migrating from one LLM to other seamlessly, combining LLMs with various data and functionalities.

Gives more power than using a simple API.

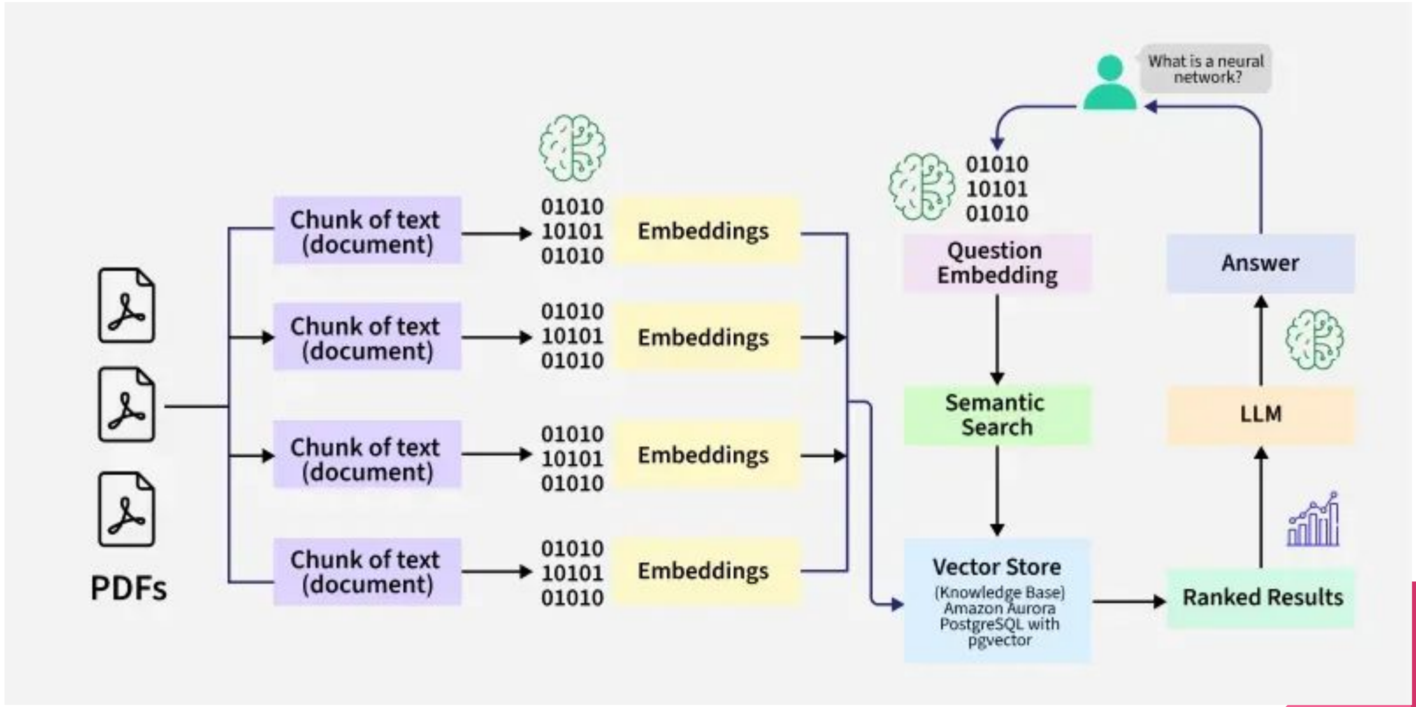
LlamaIndex is the next best alternative.

What is a framework?

In software development, it is toolbox that provides a structured way for developers to build and manage their software projects.

Framework includes libraries of prewritten code, software tools and guidelines that help in the rapid development of applications. For example react, angular, flutter.

LangChain Pipeline



LangChain - Model i/o

Input: Csv file, PDF, website, image, etc

Ouput: text in different format

Important Steps:

- Initialize the model
- Set the template
- Set the prompt
- Get the results by invoking the model.

https://github.com/Athirakr94/GenAI_Notes

Session 3

Introduction to RAG, Agents and Agentic AI

Intelligent Document Processing (IDP)

1. Invoice processing
2. Customer Onboarding
3. Claims Processing
4. Email and attachment processing
5. Medical records management

RAG

RETREIVAL AUGMENTED GENERATION

Problem statement:

I have some private documents that I need to ask questions about.

I want the LLM to focus only on these documents for answers, rather than relying on the entire dataset.

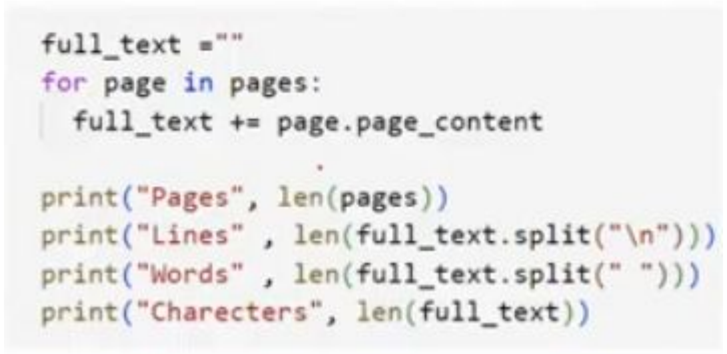
These documents weren’t available during the model’s training**- LLM may hallucinate**

Why RAG is needed?

1. Private document repository
2. Reduced hallucinations
3. Length prompt issue

Steps in RAG

Load the document



```
full_text =""
for page in pages:
    full_text += page.page_content

print("Pages", len(pages))
print("Lines" , len(full_text.split("\n")))
print("Words" , len(full_text.split(" ")))
print("Charecters", len(full_text))
```

Split the data into chunks of tokens/ words/lines, etc.

Creating embedding

Storing in a vector DB

Retrieval

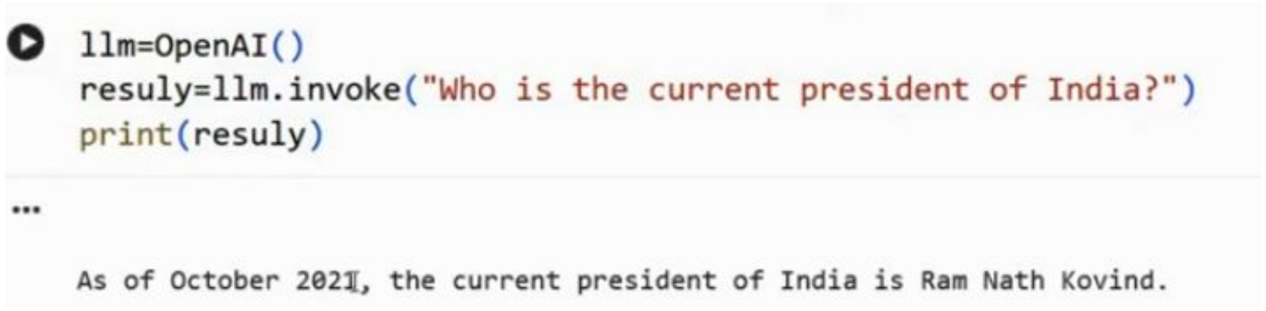
Limitations

Couldn’t solve math

Couldn’t access to internet

Couldn’t give answers related to recent facts.

Need for an alternate option!!!!



```
llm=OpenAI()
resuly=llm.invoke("Who is the current president of India?")
print(resuly)
```

```
As of October 2021, the current president of India is Ram Nath Kovind.
```

AI agents

Agents are software program designed to imitate how humans perform task.

Agents go through the cycle of Think Act and Observe

Think: You start thinking on what needs to be done

Act : Search online to gather information

Observe : After reviewing the results decide if the content is useful.

This cycle is repeated until the goal is achieved

Agents interact with real world, external events, current data.

Agents = LLM Reasoning + Tools

Agents receive natural language inout from the user.

Employ LLM to process the inout and create an action plan.

Execute the action plan.

Delivers the output from the executed plan.

Tools:

Google search, Wikipedia, Calculator, Python REPL

Agents

1. Agents are designed to imitate humans.
2. AI agents can simulate human thinking, decision making and communication.
3. They canhandle task like analysis, customer service and coordination faster without fatigue.
4. As they improve many routine and creative roles may shift from humans to AI agents.

Be careful with agents

Agents are quick to build but, they show some issues like

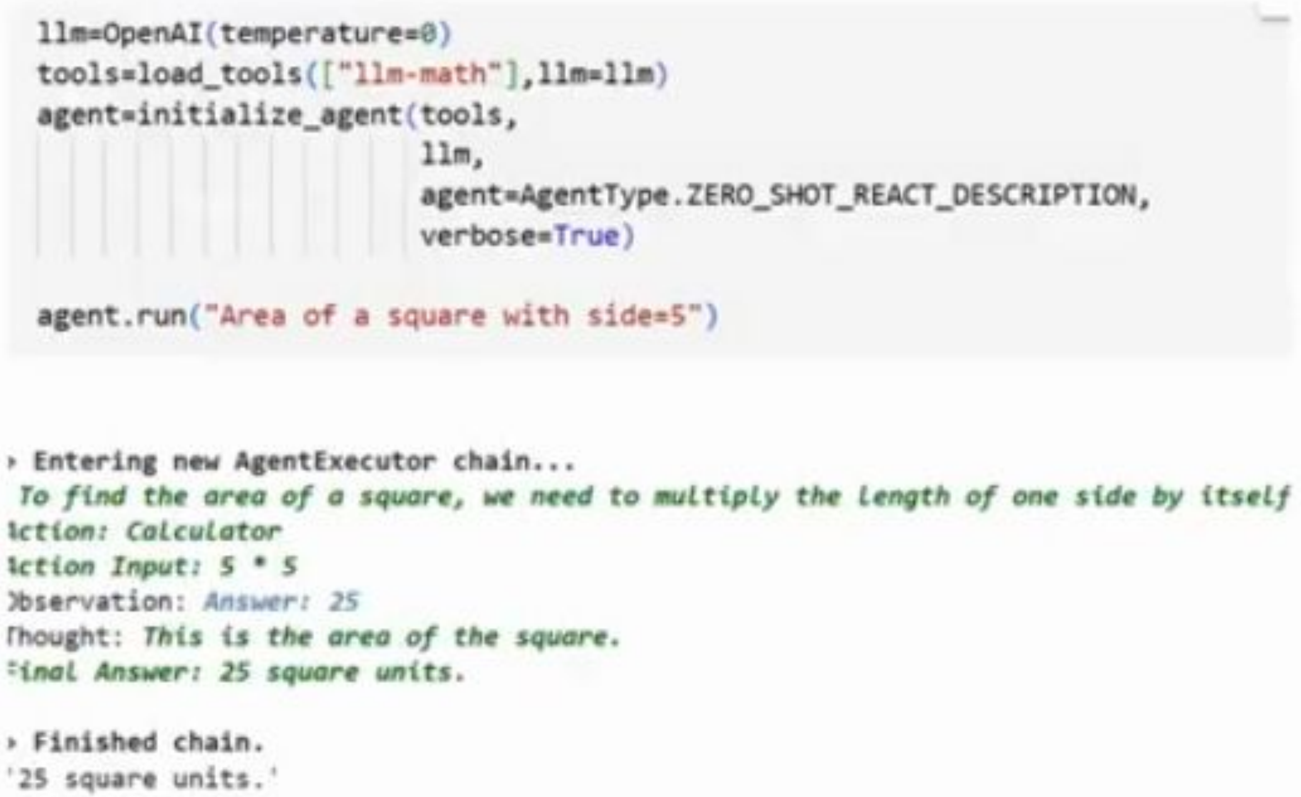
1. In consistent results
2. Max token related errors
3. Timed out errors
4. Less power to customize an existing agent

Agents are still evolving

Building a agent

1. Define the tools.
2. Define the internal thinking prompt.
3. Define the agent
4. Work with LLM agent.

Calculator tool



```
llm=OpenAI(temperature=0)
tools=load_tools(["llm-math"],llm=llm)
agent=initialize_agent(tools,
                       llm,
                       agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
                       verbose=True)

agent.run("Area of a square with side=5")
```

```
> Entering new AgentExecutor chain...
 To find the area of a square, we need to multiply the length of one side by itself
ction: Calculator
ction Input: 5 * 5
bservation: Answer: 25
hought: This is the area of the square.
inal Answer: 25 square units.

> Finished chain.
'25 square units.'
```

What is an Agentic AI system?

A single AI agent performs a specific goal-oriented task using tools, memory or reasoning.

Example: Access internet, get contents and context then generate a message.

Agentic AI involves multiple specialized AI agents working together collaboratively, to achieve a goal with planning, decision making, tool use and self reflection.

Agentic AI system

It has,

1. Autonomy- to decide what actions to take without step by step human prompting.
2. Tools/APIs- to use external functions like search, code execution, database, etc.
3. Reasoning- to break down task into smaller steps and sequence them.
4. Memory/ Context- to keep it short term/ long term state for multi-turn tasks.

Vibe Coding

- Refers to using natural language prompts to guide AI systems to generate working code.
- Developers shift from hand coding to directing, reviewing and refining via conversation.
- Most of vibe coding tools like Lovable, Cursor IDE, Copilots, and Devin are agentic AI systems.

Examples of complex agentic AI systems

1. Devin (by Cognition AI)
2. ChatGPT with advanced tools (OpenAI)
3. Copilot for Microsoft 365
4. Replit’s Ghostwriter/ Cursor IDE agents.

Headlines are Dramatic

AI emergence resembles that of computers.

AI makes engineers more powerful not useless.

New job categories are emerging like AI agent architect, AI governance officer, AI workflow designer, LLM application engineer, Prompt engineer, AI product manager, etc.

AI lowers the entry barrier for entrepreneurship.

Thank you

ai22resch01001@iith.ac.in
ai23resch01001@iith.ac.in