Agenda

- What is AI ?
- AI Value Creation
- Machine Learning vs Deep Learning
- Generative AI

Artificial Intelligence

Artificial Intelligence is study to create **Digital brain**

Value Creation By AI

- **McKinsey Global** Institute estimates that AI could contribute up to **$13 trillion** to the global economy by **2030**.
- A study by **PwC** found that **72%** of executives believe that AI will have a significant impact on their businesses in the **next five years.**
- A report by **Accenture found** that companies that are **early adopters** of **AI** are **twice** as likely to **outperform** their competitors financially.

Artificial Intelligence
Machine Learning

Artificial Intelligence is the **branch** of **computer science** concerned with development of **methods** that **allow computers to learn without explicit programming.**

Machine Learning is a **branch of AI**, which focuses on **methods**, that can **learn from examples** and experience **instead** of relying on **hard-coded rules** and make predictions on new data.

Deep Learning

Subfield of **Machine Learning** that focuses on **Neural Networks** (Inspired from Biological neurons) to develop **learning models.**

ANI (Artificial Narrow Intelligence)

ANI refers to **AI systems** that are **designed** and **trained** for **specific tasks or domains**. These systems excel at performing a single task or a narrow range of tasks, but they lack the ability to generalize their knowledge to other domains.

Examples:

ANI includes virtual assistants like Siri or Alexa, recommendation systems, and image recognition algorithms.

AGI (Artificial General Intelligence):

AGI, on the other hand, refers to AI systems t**hat possess the ability** to **understand**, **learn**, and apply knowledge across a **wide range of tasks** and **domains**, similar to human intelligence.

Most Popular Way to Do AI: Machine Learning

Machine Learning is a **branch of AI**, which focuses on methods that can learn from **examples** and **experience** instead of relying on hard-coded rules and **make predictions** on **new data**.

Types of Machine Learning

1. Supervised Learning (learning with **labeled data**)
2. Unsupervised Learning (discover patterns in **unlabeled data**)
3. Reinforcement learning (learn to act based on **feedback/rewards**)

Supervised learning:

Supervised learning is a category of **machine learning** that **uses labeled datasets** to **train algorithms** to predict outcomes and recognize patterns.

Unsupervised learning:

Unsupervised learning is a type of **machine learning algorithm** where the **model learns from unlabeled data**, meaning there are no predefined target labels.

Reinforcement learning:

Reinforcement learning is a type of **machine learning** algorithm where an agent **learns to make decisions by interacting with an environment.** The agent receives feedback in the form of rewards or penalties based on its actions, and its goal is to learn the optimal strategy to maximize cumulative rewards over time.

Unsupervised learning

- Clustering
- Dimensionality Reduction
- Anomaly Detection

Generative AI

1. Text Generation
    a. ChatGPT 3.5/4
    b. Bard
2. Image generation
    a. Dall-E3
    b. Stable Diffusion
3. Music Generation
    a. Music-LM
4. Video Generation
    a. Runway ML Gen-2

Agenda

- Machine Learning
- Types of Machine Learning
- Deep Learning
- Generative AI

Reinforcement learning:

Reinforcement learning is a type of **machine learning** algorithm where an agent **learns to make decisions by interacting with an environment.** The agent receives feedback in the form of rewards or penalties based on its actions, and its goal is to learn the optimal strategy to maximize cumulative rewards over time.

Clustering:

Clustering is a technique in **unsupervised learning** where **data points are grouped** into **clusters based** on their **similarity** or **proximity** to each other, without any predefined labels.

Dimensionality Reduction:

Dimensionality **reduction is a process in machine learning and statistics aimed** at **reducing the number of features** or **variables** in a dataset while preserving important information. It helps to simplify the dataset by representing it in a lower-dimensional space, making it easier to visualize, analyze, and process.

Anomaly detection:

Anomaly detection is a **technique** used in **machine learning** to **identify data points** that **deviate significantly** from the **norm** or **expected behavior** within a **dataset**. These data points, known as anomalies or outliers, may represent unusual patterns, errors, or fraudulent activities.

Agenda

- Recap
- Deep Learning
    Neural Networks
- Generative AI
- Discriminative vs Generative AI

Artificial Intelligence
Machine Learning
Artificial Intelligence is the **branch** of **computer science** concerned with development of **methods** that **allow computers to learn without explicit programming.**

Machine Learning is a **branch of AI**, which focuses on **methods**, that can **learn from examples** and experience **instead** of relying on **hard-coded rules** and make predictions on new data.

Deep Learning
Subfield of **Machine Learning** that focuses on **Neural Networks** (Inspired from Biological neurons) to develop **learning models.**

Supervised learning:
Supervised learning is a category of **machine learning** that **uses labeled datasets** to **train algorithms** to predict outcomes and recognize patterns.

Some supervised algorithms:
- k-Nearest Neighbors
- Support Vector Machines (SVMs)
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests

Unsupervised learning:
Unsupervised learning is a type of **machine learning algorithm** where the **model learns from unlabeled data**, meaning there are no predefined target labels.

Unsupervised learning
- Clustering
- Dimensionality Reduction
- Anomaly Detection

Some Unsupervised learning algorithms
- K-Means
- DBSCAN
- Hierarchical Cluster Analysis(HCA)
- One-class SVM
- Principal Component Analysis(PCA)
- t-distributed Stochastic Neighbor Embedding(t-SNE)

Reinforcement learning:
Reinforcement learning is a type of **machine learning** algorithm where an agent **learns to make decisions by interacting with an environment.** The agent receives feedback in the form of rewards or penalties based on its actions, and its goal is to learn the optimal strategy to maximize cumulative rewards over time.

Some Reinforcement Learning algorithms
- Q-Learning
- SARSA
- Deep-Q network(DQN)
- Policy gradient methods
- actor-critic methods
- Markov Decision Process

Neural Networks

Neural Network, also known as Artificial Neural Networks(ANNs) or Simulated Neural Networks(SNNs),is an interconnected group of nodes inspired by a simplification of neurons in brain.

Generative AI

Generative AI, also **known as GenAI** is a type of **artificial intelligence** that **creates new content** based on the **patterns learned from existing data** .These algorithms can generate text, images, videos, audio, code etc.

Generative AI

1. Text Generation
    a. ChatGPT 3.5/4
    b. Bard
2. Image generation
    a. Dall-E3
    b. Stable Diffusion
3. Music Generation
    a. Music-LM
4. Video Generation
    a. Runway ML Gen-2

Discriminative vs Generative AI

- Discriminative AI represents a specialized area in artificial intelligence focused on distinguishing between different types of data input . Unlike Generative AI, Discriminative AI relies on existing data, identifying and categorizing it based on the learned patterns.
- Generative Artificial Intelligence , also known as GAI refers to AI system capable of generating , text, images or other data in response to inputted prompt. These system learn from existing data and then create new content that shares similar characteristics with training data.

Lecture 04

Agenda

- Recap
- Generative AI

  LLM vs Diffusion Models
- Prompting

Deep Learning

Deep learning simulates our brain, helping systems learn to identify object and perform complex task with increasing accuracy without human intervention
.(as required by machine learning)

Discriminative AI **Generative AI**

Generative Artificial Intelligence , also known as GAI refers to AI systems capable of generating text ,images or other data in response to inputted prompt. These systems learn from existing data and then create new content that shares similar characteristics with training data.

Discriminative AI represents a specialized area in Artificial intelligence focused on distinguishing different types of data input. Unlike Generative AI, discriminative AI relies on existing data, identifying and categorizing it based on learned patterns.

Large Language Models(LLMs)

Large Language Model refers to large, general purpose language models that can be pre-trained

and then fine-tuned for specific purposes.

Diffusion Models

Diffusion models are advanced machine learning algorithms that uniquely generate high-quality data by progressively adding noise to a dataset and then learning to reverse the process.

LLMs

- LLMs can perform various natural language tasks without fine-tuning or task-specific training.
- LLMs work with natural language data, such as text and sentences.
- LLMs generate coherent and relevant text output given any prompt.

Diffusion Models

- Diffusion models are specialized in image generation tasks, transforming noise into realistic and diverse images.
- Diffusion models operate on image data, learning appearance and variation from large datasets.
- Diffusion model produce realistic and multifaceted images by reversing a diffusion process.

Tokenization

Definition: Tokenization is the process of **breaking down** text into **smaller units** called tokens, which can be **words**, **phrases**, or **even characters**.

Tokens in Language Models

Definition: Tokens are the fundamental **units** of text used by language models. They can be as short as a **single character** or as long as a **word** or **phrase.**

Importance: Language models process text at the token level, allowing them to analyze and generate text based on the context and relationships between individual tokens.

What is the context window of LLMs?

Context window is a textual range around the target token that a Large Language Model(LLM) can process at the time information is generated.

Context window of:

- GPT-3 is 32K tokens
- GPT-3.5-TURBO is tokens
- GPT-4 is 4096 tokens
- Bard is 1K tokens(by 17th Nov,2023)
- Gemini is 128K token

What is a prompt?

A prompt is a short piece of text that is given to large language model as input and it can be used to control the output of the model in many ways.

What is Prompt Engineering?

- Prompt engineering involves crafting tailored instructions or queries to guide the language model towards desired outputs.
- It helps shape the behavior of the model and improves response quality.

Anatomy of Prompt

Definition: In the context of language models, a prompt refers to the **input** or **instruction provided** to the model to generate a specific response or output

Parts of a Good Prompt (Not all are necessary for each and every prompt.)

1- Simulate Persona
2- Task
3- Steps to complete Task
4- Context/Constraints

5- Goal
6- Format Output

Week Summary

1. Deep Learning
    - Neural Networks
2. Generative AI
3. Discriminative vs Generative AI
4. Tokens
5. Context Window

Lecture 05

Stemming vs Lemmatization

Stemming	Lemmatization
Removing the **last few characters** of a given word, to obtain a shorter form, often leading to **incorrect meanings and spelling**	Lemmatization considers the **context** and converts the word to its **meaningful** short form
For example: Stemming the word '**Caring**' would return '**Car**'	For example: lemmatizing the word '**Caring**' would return '**Care**'

Introduction to ChatGPT

- ChatGPT stands for Generative Pretrained Transformer.
- ChatGPT is an advanced AI language model developed by OpenAI.
- ChatGPT revolutionizes Natural Language Processing.

Applications of ChatGPT

1.Content Generation
2.Text Summarization
3.Chatbots, Virtual Assistants, and Customer Support (Question Answering)
4.Language Translation
5.Interactive Storytelling
6.Educa
7.
8.**Step 1: Accessing chat.openai**
9.

10. To begin, open your preferred web browser and navigate to the chat.openai website. The website URL is https://chat.openai.com/auth/login .
11.

Step 2: Creating an Account

- If you already have an account, you can skip this step and proceed to Step 3. If not, you will need to create a new account.
- Look for the "Sign Up" or "Register" button on the login page and click on it.
- Click on “Continue with Google”, and then select your Google account.
- Follow the on-screen instructions to open the dashboard.

Step 3: Logging In

- Click on the "Log In" button to access your chat.openai account.
- Click on “Continue with Google”, and then select your google account.

Step 4: Engaging in Conversation

In the chat interface, you will see a text input box where you can type your message or question. Simply enter your text and press "**Enter**" or click the send button to send your query to the AI model. Once you've sent your message, the AI model will process it and generate a response.

Generative AI

Tokenization

Definition: Tokenization is the process of **breaking down** text into **smaller units** called tokens, which can be **words**, **phrases**, or **even characters**.

Tokens in Language Models

Definition: Tokens are the fundamental **units** of text used by language models. They can be as short as a **single character** or as long as a **word** or **phrase.**

Importance: Language models process text at the token level, allowing them to analyze and generate text based on the context and relationships between individual tokens.

Prompt Engineering

What is Prompt Engineering?

- Prompt engineering involves **crafting tailored** instructions or **queries** to guide the language model towards desired outputs.
- It helps shape the **behavior** of the model and improves response quality.

Anatomy of Prompt

Parts of a Good Prompt (Not all are necessary for each and every prompt)

1. Simulate Persona
2. Task
3. Steps to complete Task
4. Context/Constraints
5. Goal
6. Format Output

Prompt Limits in Language Models

Definition: Prompt limits refer to the constraints imposed on the input provided to a language model.

Reasons for Prompt Limits:

- **Memory limitations:** Language models have a **finite capacity** to process and store information.
- **Computational constraints:** Longer prompts require **more processing** time and

computational resources.

- **Response coherence:** Extremely long prompts may result in **irrelevant outputs.**

Prompting Example 01

Simulate Persona: You are a customer service representative for a tech company.

Task: Assist a customer in resolving a software issue.

Steps to complete Task:

Greet the customer and ask for details about the issue.

Provide step-by-step instructions to troubleshoot the problem.

If the issue persists, escalate it to the technical support team.

Context/Constraints: The customer is not tech-savvy, and you must communicate in a friendly and non-technical manner.

Goal: Resolve the customer's issue satisfactorily.

Format Output: Provide a summary of the troubleshooting steps and any additional information the customer might need.

Prompting Example 02

Simulate Persona: Imagine you are a mentor guiding a group of individuals new to prompt engineering.

Task: Identify and address common issues in prompt creation to improve overall effectiveness.

Steps to complete Task:

Analyze the prompt for Simulate Persona, Task, Steps to complete Task, Context/constraints, goals and Format Output

Identify and rectify instances of ambiguity or vagueness in prompt language.

Ensure that prompts provide sufficient information for the desired task without unnecessary complexity.

Context/Constraints: Consider the diverse backgrounds and skill levels of those engaging with the prompts.

Goal: Enhance the quality of prompts by eliminating potential sources of confusion or misinterpretation.

Format Output: Provide a revised version of a poorly constructed prompt, highlighting the changes made and explaining how each modification contributes to better prompt engineering.

Context in Prompting

Definition: Context refers to the **information** and preceding text that **informs** the **language model** about the desired output or influences its response.

Role of Context:

- **Context:** helps shape the model's behavior and generate more accurate responses.
- **Relevance:** Context ensures continuity and relevance in the generated text.
- **Interpretation:** The language model uses context to understand the desired meaning and tone of the prompt.

Tips for Effective Prompt Engineering

- **Be explicit:** Clearly specify the desired format or output.

- **Provide context:** Offer relevant information or background to guide the model.
- **Control output length:** Instruct the model to **generate** responses of a **specific length.**
- **Restrict responses:** Utilize techniques like temperature adjustment to refine output quality.
- **Experiment and iterate:** Refine prompts through experimentation and feedback loops.

Scenario 1: Essay Writing For Students
Step 1 (Bad Prompt): "Write an essay about your favorite hobby."

Step 2 (Refined Prompt): "Write an engaging essay describing how your favorite hobby has positively influenced your personal growth and development."

Step 3 (Further Refinement): "Write an insightful essay explaining how your favorite hobby, such as playing a musical instrument, has enhanced your problem-solving skills, increased your patience, and boosted your self-confidence."

Scenario 2: Historical Event Analysis

Step 1 (Bad Prompt): "Analyze the causes of World War II."

Step 2 (Refined Prompt): "Provide a comprehensive analysis of the political, economic, and social factors that contributed to the outbreak of World War II, considering both domestic and international influences."

Step 3 (Further Refinement): "Compose a well-structured and nuanced analysis of the immediate triggers and long-term underlying causes of World War II, highlighting key events and decisions that escalated tensions and ultimately led to the global conflict."

Summary

1. Prompt Engineering
2. Best Practices
3. ChatGPT PlayGround
4. Sample prompt practise

Lecture 06

Agenda

- Diffusion Models
- Negative Prompting
- Stable Diffusion 2.1
- How it works

Negative Prompting

Not just what you want, but what you **don't** want.

Positive prompts
Generate a high-resolution image of an autumn forest scene with vibrant foliage, scattered leaves on the ground, and warm sunlight filtering through the trees. Capture the essence of a crisp fall day with detailed textures and realistic colors, Sun light, shades, 4k realistic, unreal engine

Negative Prompt
blurred, Non realistic, low quality, cartoon, art, paper, painting

Negative Prompting Examples

Positive prompts
fine-art photography of a Clear crystal cube, reflecting the seaface, floating on the tumultuous sea, Arctic Ocean, sunset, magic time, by Andreas Rocha, Minimalism, artistic, atmospheric, masterpiece, golden ratio composition, hyper-detailed, 8K wallpaper

Negative Prompt
blurred, Non realistic, low quality, cartoon, art, paper, painting

Parisian luxurious interior penthouse bedroom, dark walls, wooden panels
no gold, pink, Blurry, low quality, less details

Prompting Examples I

Positive prompt: Generate a high-resolution image of an autumn forest scene with vibrant foliage, scattered leaves on the ground, and warm sunlight filtering through the trees. Capture the essence of a crisp fall day with detailed textures and realistic colors, Sun light, shades, 4k realistic, unreal engine

Negative Prompt: Low Quality, unreal trees, deformed trees

Prompting Examples II

Positive prompt: Lonely peak, peeks above clouds, sunrise paints it golden pink. Misty valleys below, waterfall sings, magic feels close. Picture-perfect view, soft colors, like a dreamy painting.

Negative Prompt: blurry, out of focus, flat lighting, harsh shadows, man-made elements, text, visible brushstrokes, pollution, cropped, cartoonish, low resolution, unnatural colors.

Prompting Examples III

Positive prompt: Generate an image of a majestic African elephant walking gracefully in a dense, sunlit jungle. The elephant should have gray skin with prominent wrinkles, and it should be reaching for leaves on a tall tree. The jungle should include lush green foliage, a clear blue sky. Include rocks and a small stream for added realism. The atmosphere should feel peaceful and serene. The image should have a resolution of at least 1920x1080 pixels and maintain high-quality details

Negative Prompt: blurry, out of focus, flat lighting, harsh shadows, man-made elements, text, visible brushstrokes, pollution, cropped, cartoonish, low resolution, unnatural colors.

Diffusion Models

Noise refers to unwanted information that disrupts the clarity and accuracy of the picture.

DALL-E3

- DALL.E3 is a text-to-image AI model developed by **OpenAI**. It operates by taking a text prompt and generating a series of matching images.

- It is trained on millions or billions of text-image pairs ,allowing it to understand complex concepts and create contextually relevant visuals.

- It tends to produce more abstract or computer-generated images.

Generated by DALL-E 3

Positive prompt: Generate a high-resolution image of an autumn forest scene with vibrant foliage, scattered leaves on the ground, and warm sunlight filtering through the trees. Capture the essence of a crisp fall day with detailed textures and realistic colors, Sun light, shades, 4k realistic, unreal engine

Negative Prompt: Low Quality, unreal trees, deformed trees

Generated by DALL-E 3

Positive prompt: Generate a high-resolution image of Two friendly domestic canaries enjoying a breathtaking sunset view from a branch, casting long shadows on the glassy surface of a lake below.

Negative Prompt: Low Quality, perched on a broken branch in front of a dark and stormy sky.

Best Practices

- **Understand the model:** Familiarise yourself with the strengths and limitations of the language model you are using.
- **Start simple:** Begin with straightforward prompts and gradually refine them based on experimentation and feedback.
- **Domain-specific prompts:** Tailor prompts to specific topics or domains to enhance response quality in specialized areas.

Week Summary

1. Prompt Engineering
2. Best Practices
3. ChatGPT PlayGround
4. Stable Diffusion Prompting

Lecture 07

Agenda

- What is Programming?
- Basic Concepts of Programming
- What is variable?

    - Rules for naming variables
- Data types
- Colab Notebook hands on
- Quiz

Definition of Programming

- Programming is the process of creating **sets of instructions** that tell a **computer what to do.**
- It involves writing code in a programming language to **solve problems** or automate tasks.

Key Elements of Programming

Instructions: Creating **step-by-step** instructions to guide the computer's actions.
Language: Using a programming language to communicate with the computer.
Problem Solving: Analyzing problems and designing solutions using logical thinking.

Programming Languages

Definition: Programming languages are formal languages designed to communicate instructions to a computer.
Examples: Python, Java, C++, Kotlin

What are Variables?

Definition: Variables are **containers** that **hold data** values in a program.
Analogy: Think of variables as labeled boxes used to store different types of items

Variable Naming

Rules:

1. Start with a **letter** or **underscore (_)**.
2. Can include **letters**, **numbers**, and **underscores**.
3. Avoid using **reserved words** (e.g., print, if, while).

Analogy: Give your boxes unique and descriptive labels for easy identification.

Assigning Values to Variables

Syntax:

variable_name = value

Analogy: Put an item inside a labelled box.
Example: **a = 5 (correct)**
a == 5 (Wrong)

Data Types of Variables

Integer: whole numbers **(e.g., 10, -5)**
Float: decimal numbers **(e.g., 3.14, -2.5)**
String: text **(e.g., "Hello", 'Python')**
Char: Characters **(e.g., “a”)**
Analogy: **Different types** of items can be stored in **different boxes**.

Lecture 13

Agenda

- Openai Image Generation using Dall-E
- What is Hugging Face?
- Pipelines
- Sentiment Analysis
- Name Entity Recognition
- Text Summarization
- Object Recognition

Dall-E

- DALL·E is an artificial intelligence model developed by OpenAI.
- It is designed to **generate** highly creative and unique **images** from **textual descriptions.**
- DALL·E has the ability to **understand** and **follow** detailed instructions for generating images.
- https://openai.com/dall-e-2
- https://openai.com/dall-e-3

Hugging Face

- Hugging Face is an **open-source library** and platform for Natural Language Processing (NLP).
- It provides a wide range of **pre-trained models** and **tools** that make it easier to work with NLP tasks.
- Hugging Face is **beginner-friendly** and widely used by researchers and developers.

Hugging Face Pipeline

- Hugging Face Pipelines are a **high-level** and **user-friendly API** provided by the Hugging Face library.
- They enable developers and researchers to perform various Natural Language Processing (NLP) tasks with just a few lines of code.
- Pipelines contains **complex NLP models** and processes, making it easy to use and integrate them into applications.

Hugging Face API:

Hugging Face **offers a powerful API** and **platform** for **natural language processing** (NLP) tasks.

The Hugging Face API provides **access to a wide range of pre-trained models** for tasks such as text classification, sentiment analysis, question answering, and language generation.

https://huggingface.co/docs/api-inference/index

Sentiment analysis:

Sentiment analysis, **also known as opinion mining**, is a natural language processing (NLP) technique **used to determine the sentiment or emotion expressed in a piece of text**.

Object Recognition:

Object recognition, **also known as object detection**, is a **computer vision task** that involves **identifying and locating objects within an image or video**. The goal is to not only recognize the presence of objects but also to accurately localize and classify them.

Supported Tasks:

Hugging Face Pipelines support a wide range of NLP tasks, including:

1. Text Classification
    a. Sentiment Analysis
2. Named Entity Recognition (NER)
3. Text Summarization
4. Image Classification

Text Summarization:

Text Summarization is a technique used in Natural Language Processing (NLP) to generate a concise summary of a longer text while preserving its main points.

Name Entity Recognition (NER):

- Named Entity Recognition (NER) is a subtask of Natural Language Processing (NLP) that focuses on identifying and classifying **named entities** in text.
- Named entities are real-world objects such as **persons**, **organizations**, **locations**, **dates**, and more.

Summary:

- Image generation from Openai
- Hugging Face library
- Hugging Face pipelines

Lecture 15

Agenda:

- Data Science
- Data Science processes
- Types of Data

Data Science:
Data science is the field of **extracting knowledge** and insights from data through a blend of **statistics**, **programming**, **machine learning**, and **domain expertise**.

Structured Data:

- Structured data is organized and follows a **predefined format**, usually stored in **databases** or **spreadsheets**.
- **Example:** A customer database with columns for name, age, email, and purchase history.

    - Ordinal data
    - Nominal data
    - Numerical data

Structured Data Examples:

Ordinal:

- Movie ratings (1-5 stars)
- T-shirt sizes (S, M, L)
- School grades (A, B, C)

Nominal:

- Blood types (A, B, AB, O)
- Hair colors (blonde, brunette, redhead)
- Favorite sports (soccer, basketball, tennis)

Numerical:

- Ages (years)
- Temperatures (°C)
- Book page numbers

Bonus:
Traffic light colors (red, yellow, green) – nominal for each light, ordinal for sequence.

Unstructured Data

- Unstructured data is **not organized** and lacks a predefined format, often in the form of **text, images, audio, or video**.
- **Example:** Social media posts, customer reviews, or images from a surveillance camera.

Semi-Structured Data

- Semi-structured data has some organization but does not adhere to a **strict schema**, often **containing tags** or **labels.**
- **Example:** Emails, JSON files that contain data with tags or key-value pairs.

Data Acquisition

- **Data acquisition** refers to the **process of collecting raw data** from various sources, which could be **structured** or **unstructured**, to be used for analysis, modeling, or other purposes.

- Identify Data
- Retrieve Data
- Query Data

What is Exploratory Data Analysis (EDA)?

- The process of **exploring** and **summarizing** the main characteristics of the data to uncover **patterns**, **relationships**, and **trends**.
- It helps in formulating questions and making data-driven decisions.

Importance of EDA:

- Provides an **initial understanding** of the dataset.
- Helps in identifying data **quality issues**, such as **missing values**, **outliers**, and **inconsistencies**.
- Guides the selection of appropriate statistical techniques and **models**.
- Helps in feature engineering and **variable selection**.
- Enables the discovery of **meaningful insights** and actionable conclusions.

Descriptive Statistics

Descriptive statistics is the branch of statistics that focuses on **summarizing** and **describing** the main **features/Attributes/Variables** of a dataset.

Correlation Analysis

Correlation analysis is used to determine the **relationship** between **two** or **more variables** in a dataset. It helps us understand how **changes** in one variable **affect another variable**.

Central Tendency

- A measure of **central tendency** is a single value that attempts to **describe** a set of data by **identifying the central position** within that set of data. As such, measures of central tendency are sometimes called measures of central location.
- The **mean**, **median** and **mode** are all valid measures of **central tendency**, but under different conditions, some measures of central tendency become more appropriate to use than others.

Mean

The mean is the **average** of a set of numbers.

Calculated by summing all the numbers in the dataset and dividing by the total count.

Formula: Mean = (Sum of all numbers) / (Total count)

Example: For the dataset **[5, 7, 10, 12, 15],** the mean is **(5 + 7 + 10 + 12 + 15) / 5 = 9.8.**

Sensitive to outliers

Example: For the dataset **[5, 7, 10, 12, 150],** the mean is **(5 + 7 + 10 + 12 + 150) / 5 = 36.8.**

Median

The median is the **middle value** in a **sorted** dataset.

If the dataset has an odd number of values, the median is the middle number.

If the dataset has an even number of values, the median is the average of the two middle numbers.

Example 1: For the dataset **[3, 5, 6, 8, 9]**, the median is **6.**

Example 2: For the dataset **[2, 4, 6, 8]**, the median is **(4 + 6) / 2 = 5.**

Mode

The mode is the value that appears **most frequently** in a dataset.

A dataset can have one mode, more than one mode (multimodal), or no mode (no value repeats).

Example 1: For the dataset **[3, 4, 4, 6, 8]**, the mode is **4.**

Example 2: For the dataset **[1, 2, 3, 4, 5]**, there is **no mode**.

Quartiles and Interquartile Range (IQR)
Quartiles: Before understanding the IQR, we need to grasp the concept of quartiles.
Quartiles divide a dataset into **four equal parts**, each representing a quarter **(25%)** of the data.
The first quartile **(Q1)** is the value below which **25%** of the data falls.
The second quartile **(Q2)** is the value below which **50%** of the data falls. It's actually **Median.**
The third quartile **(Q3)** is the value below which **75%** of the data falls.

Finding Quartiles
To find the quartiles, first, arrange the dataset in **ascending order**.
If the dataset has an odd number of values, the median is the second quartile (Q2).
If the dataset has an even number of values, calculate the median of the lower and upper halves to find Q2.

Interquartile Range (IQR):
The IQR represents the difference between the first and third quartiles(Q1 and Q3).
It indicates the spread of the middle 50% of the data.

Formula: IQR = Q3 - Q1

Interpretation of the IQR
The IQR provides information about the **spread** and **variability** of the data.
A **larger IQR** indicates a **greater spread** and more variability in the dataset.
Conversely, a **smaller IQR** suggests a **less spread** and less variability.

Applications
The IQR is widely used in various fields, **including economics**, **healthcare**, and **social sciences.**
It helps compare and analyze data distributions, identify **abnormal data points**, and detect **potential patterns** or **trends.**

Why we care about Dispersion?

Dispersion is the spread of **data points** around a **central value**, indicating variability within a dataset.

- Increased difficulty in **capturing patterns**.
- Increased risk of **overfitting**.
- Reduced predictive **accuracy.**
- Impact on **outlier handling**

Proximity:

- In machine learning, "**proximity**" refers to the **measure of similarity** or **closeness between data points** within a dataset. It is often used in clustering algorithms and nearest neighbor methods to determine how closely related or similar two data points are to each other based on certain features or attributes.

Data Cleaning Techniques
Handling Missing Data: We use methods like filling or removing to deal with missing values.

Outlier Detection: Find and address unusual data that can affect analysis or models.

Data Standardization: Make data consistent for easier analysis and comparison.

Data Validation: Check data against rules to ensure accuracy and reliability.

Data Integration

- Data integration is the process of **combining data** from **various sources** into a single dataset. The **goal** is to provide users with **consistent access** to data across different subjects and structure types.
- **Data Reduction**
- Data reduction is the **process of transforming digital information into a simplified, ordered, and corrected form**. It can be used on numerical or alphabetical information.

PCA

Principal component analysis (**PCA**) is a **statistical method that reduces a data table to its main features**. It's a dimensionality reduction technique that's often used to reduce the size of large data sets. PCA works by **linearly transforming data** onto a new coordinate system.

Dimensionality Reduction Technique:

- Eigenvalue Decomposition (Eigendecomposition):
- Singular Value Decomposition (SVD):
- t-SNE (t-Distributed Stochastic Neighbor Embedding)

Lecture# 16

Data Science

Data science is the field of **extracting knowledge** and insights from data through a blend of **statistics**, **programming**, **machine learning**, and **domain expertise**.

Data Skewness:

Data skewness is **an uneven distribution of data within a dataset**. It occurs when certain values or ranges of values appear more frequently than others. This skewness can occur in various dimensions of the dataset, such as columns, partitions, or even individual files.

Data Skewness:

Data skewness is **an uneven distribution of data within a dataset**. It occurs when certain values or ranges of values appear more frequently than others. This skewness can occur in various dimensions of the dataset, such as columns, partitions, or even individual files.

Data Transformation:

- Data transformation is the **process** of **converting**, **cleansing**, and **structuring data** into **a usable format** that can be analyzed to **support decision making processes**, and to propel the growth of an organization.

Smoothing:

- Smoothing in data preprocessing refers to a set of techniques used to **reduce noise and fluctuations** in data while preserving important underlying trends and patterns. It is a crucial step in preparing data for analysis and modeling, particularly when dealing with noisy or uneven data points.

Feature Engineering:

- Feature engineering is an essential step in the data science process that involves **creating and transforming raw data into features suitable for building machine learning models.**

Data Normalization:

Normalization is a data preprocessing technique that **transforms data values** to a **common scale** or **distribution of values**. It can include adjusting the scale of values to a similar metric or adjusting the time scales to be able to compare like periods.

Scaling:

1. Min Max Scaling
2. Z-score Scaling

Min-Max Scaling

Min-Max Scaling is a data **normalization** technique used to **transform data** into a **specific range**.

Formula for Min-Max Scaling

Min-Max Scaling- Example

Example: Normalize the data **[200, 300, 400, 600, 1000]** for interval **[0,1]**

Step 1: Find the minimum value (**min_X**) and maximum value (**max_X**) in the data..

min_X = 200

max_X = 1000

Step 2: Apply the Min-Max scaling formula to each data point:

For x = 200:

Scaled_x = (200 - 200) / (1000 - 200)*(1 - 0) = 0 / 800 = **0**

Z-Score scaling

Z-Score Scaling, also known as **standardization**, is a technique used to **transform data** into a standard normal distribution.

Z-Score scaling

Example: Normalize the data **[200, 300, 400, 600, 1000]** using z-score scaling

Step1

Calculate the Mean **(μ)** of the data.

μ = 500

Step2

Calculate the Standard Deviation **(σ)** of the data.

σ= 286.4789

Z-Score Scaling

Calculate the Z-Score for each data point:

For x = 200:

z = (200 - 500) / 286.4789 ≈ **-1.047**

For x = 300:

z = (300 - 500) / 286.4789 ≈ **-0.698**

For x = 400:

z = (400 - 500) / 286.4789 ≈ **-0.349**

For x = 600:

z = (600 - 500) / 286.4789 ≈ **0.349**

For x = 1000:

z = (1000 - 500) / 286.4789 ≈ **1.747**

Model Selection:

The process of selecting the machine learning model most appropriate for a given issue is known as model selection.

here are the key steps to choose the best model in machine learning condensed into five points:

- **Understand the Problem**: Clearly define the problem you're solving and determine the type of task (classification, regression, clustering, etc.) you're dealing with.
- **Explore Model Options**: Familiarize yourself with a variety of machine learning algorithms suitable for your problem, considering factors like data size, complexity, and distribution.
- **Evaluate Performance**: Train multiple models and evaluate their performance using appropriate metrics, considering factors like accuracy, precision, recall, etc., depending on your problem type.
- **Consider Model Complexity**: Balance the trade-off between model complexity and interpretability, opting for simpler models if interpretability is crucial or if you have limited data.
- **Iterate and Optimize**: Fine-tune your chosen model's hyperparameters, consider ensemble methods for performance enhancement, and continuously refine your approach based on experimentation and feedback.

Model Building:

- Model building involves the **process of selecting** and **training** a machine learning model using a dataset to **learn patterns** and **relationships** within the data. This typically includes steps such as preprocessing the data, selecting an appropriate algorithm, splitting the data into training and validation sets, training the model on the training set, and evaluating its performance on the validation set.

Model Validation:

- Model validation is **a machine learning (ML) process** that **assesses how well a model can produce predictions or outputs**. It's important to validate models before deployment to ensure they're accurate and reliable.

Linear Regression

Linear regression is a **supervised** machine learning algorithm used for predicting a **continuous output** (**dependent variable**) based on one or more **input features** (**independent variables**).

Logistic Regression

Logistic regression is a **supervised** machine learning algorithm that accomplishes **binary classification** tasks by predicting the probability of an outcome, event, or observation.

Mean Squared Error (MSE)

The **Mean Squared Error (MSE)** is a common evaluation metric used in regression tasks.
It measures the average squared difference between the actual and predicted values.

Decision Tree Regressor

Decision tree regression involves **partitioning the data into subsets** based on the values of **independent variables** and predicting the average of the target variable for each subset.

Random Forest Regressor:

Random Forest is learning method that creates **multiple decision trees** during training and outputs the **average prediction** (for regression) from all individual trees.

Gradient Boosting Regressor

Gradient Boosting is **learning technique** that **builds multiple decision trees** sequentially, each one correcting the errors of its predecessor.

Residual Analysis:

Residual analysis is a **technique for evaluating the validity of a linear regression model by examining the patterns of residuals**.

A residual is the difference between the observed value and the predicted value of a quantity of interest.

Decision Tree Classifier
A decision tree classifier makes decisions by **splitting the data** into **smaller subsets** based on the values of input features, ultimately assigning a class label to each data point. It constructs a tree-like structure of decision rules that can be used for prediction and is easy to interpret.

Random Forest Classifier
Random Forest is **ensemble** learning method that creates **multiple decision trees** during training and outputs the **average prediction** from all individual trees.

Gradient Boosting Classifier
Gradient Boosting Classifier is a **powerful ensemble learning technique** used for **classification tasks**. It belongs to the class of boosting algorithms, which build a strong predictive model by combining multiple weak learners sequentially.

Support Vector Machines (SVM):
Support Vector Machines (SVM) is a powerful **supervised learning algorithm** used for both classification and regression tasks.

Regression:
In regression tasks, SVMs aim to find the hyperplane that best fits the data while maintaining a maximum margin.

Classification:
In classification tasks, SVMs aim to find the hyperplane that best separates the data points of different classes in the feature space.

Ensemble:

Ensemble learning is a **machine learning** technique that **combines** the predictions of **multiple individual models** (learners) to **improve overall performance** and generalization. Instead of relying on a single model, ensemble methods create a "**committee**" of models and aggregate their predictions to make a final decision.

Gradient Boosting classifier
Gradient Boosting is **ensemble** learning technique that builds **multiple decision trees** sequentially, each one **correcting the errors of its predecessor**.

Accuracy
It is the ratio of number of **correct predictions** to the **total number of input samples**.

Note: It works well only if there are **equal** number of samples belonging to **each class**.

Confusion Matrix
Confusion Matrix as the name suggests gives us a matrix as output and **describes** the **complete performance** of the model.

Confusion Matrix
True Positives : The cases in which we predicted **YES** and the actual output was also **YES**.

True Negatives : The cases in which we predicted **NO** and the actual output was **NO.**

False Positives : The cases in which we predicted **YES** and the actual output was **NO**.

False Negatives : The cases in which we predicted **NO** and the actual output was **YES**.

Confusion Matrix - Precision
It is the number of **correct positive results** divided by the number of **positive results predicted** by the classifier.

precision emphasizes **minimizing false positives**.
Use case example: Medical tests, where **false positives** **(Predicted Diseased but actually healthy)** can lead to unnecessary treatments.

Confusion Matrix - Recall

The ratio of correctly **predicted positive** instances to the **total actual positive instances**.
Recall focuses on **minimizing false negatives**.

Use case example: Disease detection, where **False Negative** **(predicted healthy but actually diseased)** can have serious consequences.

Confusion Matrix - Example

Suppose you trained a model to classify cancer patients. You want evaluate the model performance.
You have to answer the following questions **Accuracy? Precision?** and **Recall?**

What should be of more concerned Accuracy. Precision. Recall?

Clustering Models:

Clustering models are used for **grouping similar data points** together **based on their characteristics or features**. Unlike classification, clustering is an unsupervised learning technique where the categories are not predefined. Examples include k-means clustering, hierarchical clustering, and DBSCAN.

Associative Analysis

Associative analysis in data science refers to **the process of identifying and analyzing relationships** or associations between different **variables** or **attributes** within a dataset. It aims to **uncover patterns**, **correlations**, or **dependencies** among variables that may not be immediately apparent.

Generative Analysis

Generative analysis in data science refers to a **class** of **statistical techniques** and **models** used to **generate new data samples** that are similar to an existing dataset.

One common approach to generative analysis is Generative Adversarial Networks (GANs), which consist of two neural networks, a generator and a discriminator, that are trained simultaneously.

Lecture# 18

Agenda

Classification Techniques

Sci-kit Learn

Logistic Regression

Logistic regression is a **supervised** machine learning algorithm that accomplishes **binary classification** tasks by predicting the probability of an outcome, event, or observation.

Decision Tree Classifier

A decision tree classifier makes decisions by **splitting the data** into **smaller subsets** based on the values of input features, ultimately assigning a class label to each data point. It constructs a tree-like structure of decision rules that can be used for prediction and is easy to interpret.

Random Forest Classifier

Random Forest is **ensemble** learning method that creates **multiple decision trees** during training and outputs the **average prediction** from all individual trees.

Gradient Boosting Classifier

Gradient Boosting Classifier is a **powerful ensemble learning technique** used for **classification tasks**. It belongs to the class of boosting algorithms, which build a strong predictive model by combining multiple weak learners sequentially.

Support Vector Machines (SVM):

Support Vector Machines (SVM) is a powerful **supervised learning algorithm** used for both classification and regression tasks.

Regression:

In regression tasks, SVMs aim to find the hyperplane that best fits the data while maintaining a maximum margin.

Classification:

In classification tasks, SVMs aim to find the hyperplane that best separates the data points of different classes in the feature space.

Ensemble:

Ensemble learning is a **machine learning** technique that **combines** the predictions of **multiple individual models** (learners) to **improve overall performance** and generalization. Instead of relying on a single model, ensemble methods create a "**committee**" of models and aggregate their predictions to make a final decision.

Gradient Boosting classifier

Gradient Boosting is **ensemble** learning technique that builds **multiple decision trees** sequentially, each one **correcting the errors of its predecessor**.

Accuracy

It is the ratio of number of **correct predictions** to the **total number of input samples**

Note: It works well only if there are **equal** number of samples belonging to **each class**.

Self-learning Assignment

- Gini index
- Entropy
- Sampling with replacement
- Information gain

Assignment

Lecture# 19

Agenda

- Regression Models
  - Linear Regression
  - Decision Tree Regressor
  - Random Forest Regressor
  - Gradient Boosting Regressor
- Hands-on Using Scikit Learn

Linear Regression

Linear regression is a **supervised** machine learning algorithm used for predicting a **continuous output** (**dependent variable**) based on one or more **input features** (**independent variables**).

Decision Tree Regressor

Decision tree regression involves **partitioning the data into subsets** based on the values of **independent variables** and predicting the average of the target variable for each subset.

Random Forest Regressor:

Random Forest is learning method that creates **multiple decision trees** during training and outputs the **average prediction** (for regression) from all individual trees.

Gradient Boosting Regressor

Gradient Boosting is **learning technique** that **builds multiple decision trees** sequentially, each one correcting the errors of its predecessor.

Residual Analysis:

Residual analysis is a **technique for evaluating the validity of a linear regression model by examining the patterns of residuals**.

A residual is the difference between the observed value and the predicted value of a quantity of interest.

Mean Squared Error (MSE)
The **Mean Squared Error (MSE)** is a common evaluation metric used in regression tasks.
It measures the average squared difference between the actual and predicted values.

Lecture# 20

Agenda

- Model Evaluation
- Evaluation Metrics
  - Accuracy
  - Precision
  - Recall

Why Evaluate Machine Learning Models?
Improved decision-making: Evaluation shows how **well** a model works, guiding choices about **deployment** and **improvement.**

Overfitting Prevention: Evaluation can warn against models that might **memorize training** data but do **poorly** on **new data**.

Fine-tuning: Assessment enables us to **tweak** the model improving its effectiveness.

Fair Comparison: It enables us to objectively **compare different models**.

What is Underfitting?
Too Simple: An underfit model **fails** to **capture** the underlying **pattern of the data**.

Poor Performance: This leads to errors on both the **training set** and unseen data.

Example: Trying to fit a straight line to data that has a clear curve.

What is Overfitting?
Too Complex: An overfit model **learns** the training data **too well**.

Memorization, not Generalization: It becomes a **lookup table** rather than a **pattern detector**.

Poor performance on new data: The model becomes overly specific to the training data and makes **bad predictions** on **unseen examples**.

Consequences of Overfitting and Underfitting
Inaccuracy: Both lead to **unreliable predictions.**

Underfitting: High bias, the model consistently gets it wrong in a certain way.

Overfitting: High variance performance wildly depends on the specific data it has seen.

Basics - Training, Validation, and Testing Data
Training Data: We feed this data into a machine learning algorithm to develop a model.

Validation Data: This dataset tunes model parameters (hyperparameters), helping prevent overfitting.

Testing Data: Provides a final, unbiased picture of how well the model performs on unseen data.

Key Evaluation Metrics (Classification)

1. **Accuracy**
2. **Precision**
3. **Recall**

4. F1-Score

Accuracy
It is the ratio of number of **correct predictions** to the **total number of input samples**.
Note: It works well only if there are **equal** number of samples belonging to **each class**.
Accuracy- Example
Consider that there are **98%** samples of **class A** and **2%** samples of **class B** in our training set. Then our model can easily get **98%** training accuracy by simply predicting every training sample belonging to **class A**.

When the same model is tested on a test set with **60%** samples of **class A** and **40%** samples of **class B,** then the test accuracy would drop down to **60%**.

Classification Accuracy is great, but gives us the false sense of achieving high accuracy.

The real problem arises, when the cost of misclassification of the minor class samples are very high.
Possible Scenarios
True Positives (TP): Things your **model predicted positive** that were **genuinely positive**.

False Positives (FP): Things your **model predicted as positive**, but that were **actually negative**.

True Negatives (TN): Things **correctly** predicted as **negative**.

False Negatives (FN): Things incorrectly **predicted as negative**, when they **were positive**.
Confusion Matrix
Confusion Matrix as the name suggests gives us a matrix as output and **describes** the **complete performance** of the model.
Confusion Matrix
True Positives : The cases in which we predicted **YES** and the actual output was also **YES**.

True Negatives : The cases in which we predicted **NO** and the actual output was **NO.**

False Positives : The cases in which we predicted **YES** and the actual output was **NO**.

False Negatives : The cases in which we predicted **NO** and the actual output was **YES**.
Confusion Matrix - Precision
It is the number of **correct positive results** divided by the number of **positive results predicted** by the classifier.
precision emphasizes **minimizing false positives**.
Use case example: Medical tests, where **false positives** **(Predicted Diseased but actually healthy)** can lead to unnecessary treatments.
Confusion Matrix - Recall
The ratio of correctly **predicted positive** instances to the **total actual positive instances**.
Recall focuses on **minimizing false negatives**.
Use case example: Disease detection, where **False Negative** **(predicted healthy but actually diseased)** can have serious consequences.
Example
A veterinarian builds a machine learning model to detect whether cats have a specific feline disease based on medical test results.
Data:
The vet collected data on **100** cats. Here's a breakdown:

Actually Have the Disease: **40 cats**
Actually Healthy: **60 cats**
Model Predicted Positive (Disease): **45 cats**
Model Predicted Negative (Healthy): **55 cats**

Calculate Precision - Example
Calculate Precision

Precision = TP / (TP + FP)
Precision = 35 / (35 + 10) = 35/45 = 0.78 (or 78%)

Interpretation: When the model predicts a cat has the disease, it's correct **78%** of the time. That means there's a **22%** chance of overdiagnosis.

Calculate Recall - Example
Recall = TP / (TP + FN)
Recall = 35 / (35 + 5) = 35/40 = 0.875 (or 87.5%)

Interpretation: For cats that actually have the disease, the model correctly identifies them **87.5%** of the time. The remaining **12.5%** risk being missed by the model.

Confusion Matrix - Example
Suppose you trained a model to classify cancer patients. You want evaluate the model performance. You have to answer the following questions **Accuracy? Precision?** and **Recall?**

What should be of more concerned Accuracy. Precision. Recall?

Real-World Examples
Fraud detection:

High precision to avoid **false positives** (Predicted Fraud actually not-Fraud)

Medical diagnoses:

High recall to minimize **false negatives** (Predicted healthy but actually diseased).

Key takeaways

- **Accuracy** (total % of correct classifications) is often not a meaningful performance metric when data sets are imbalanced
- **Class imbalance** implies that one class is represented much more often than others. Penalizing certain misclassifications and re-sampling the data set help dealing with imbalances.
- **Precision** is a metric that penalizes **false positives**. As such, models with high precision are cautious to label an element as positive.
- **Recall** is a metric that penalizes **false negatives**. Models with high recall tend towards positive classification when in doubt.

Lecture# 21

Agenda

- Model Evaluation
- Evaluation Metrics
    - Accuracy
    - Precision

- Recall
- F1-Score
- AUC ROC Curve

Why Evaluate Machine Learning Models?

Improved decision-making: Evaluation shows how **well** a model works, guiding choices about **deployment** and **improvement.**

Overfitting Prevention: Evaluation can warn against models that might **memorise training** data but do **poorly** on **new data**.

Fine-tuning: Assessment enables us to **tweak** the model improving its effectiveness.

Fair Comparison: It enables us to objectively **compare different models**.

What is Underfitting?

Too Simple: An underfit model **fails** to **capture** the underlying **pattern of the data**.
Poor Performance: This leads to errors on both the **training set** and unseen data.
Example: Trying to fit a straight line to data that has a clear curve.

What is Overfitting?

Too Complex: An overfit model **learns** the training data **too well**.
Memorization, not Generalization: It becomes a **lookup table** rather than a **pattern detector**.
Poor performance on new data: The model becomes overly specific to the training data and makes **bad predictions** on **unseen examples**.

Consequences of Overfitting and Underfitting

Inaccuracy: Both lead to **unreliable predictions.**

Underfitting: High bias, the model consistently gets it wrong in a certain way.

Overfitting: High variance performance wildly depends on the specific data it has seen.

Basics - Training, Validation, and Testing Data

Training Data: We feed this data into a machine learning algorithm to develop a model.

Validation Data: This dataset tunes model parameters (hyperparameters), helping prevent overfitting.

Testing Data: Provides a final, unbiased picture of how well the model performs on unseen data.

Key Evaluation Metrics (Classification)

1. **Accuracy**
2. **Precision**
3. **Recall**
4. **F1-Score**

Accuracy:

It is the ratio of number of **correct predictions** to the **total number of input samples**.

Accuracy- Example:

Consider that there are **98%** samples of **class A** and **2%** samples of **class B** in our training set. Then our model can easily get **98%** training accuracy by simply predicting every training sample belonging to **class A**.

When the same model is tested on a test set with **60%** samples of **class A** and **40%** samples of **class B,** then the test accuracy would drop down to **60%**.

Classification Accuracy is great, but gives us the false sense of achieving high accuracy.

The real problem arises when the cost of misclassification of the minor class samples are very high.

Possible Scenarios:

True Positives (TP): Things your **model predicted positive** that were **genuinely positive**.

False Positives (FP): Things your **model predicted as positive**, but that were **actually negative**.

True Negatives (TN): Things **correctly** predicted as **negative**.

False Negatives (FN): Things incorrectly **predicted as negative**, when they **were positive**.

Confusion Matrix:

Confusion Matrix as the name suggests gives us a matrix as output and **describes** the **complete performance** of the model.

Confusion Matrix:

True Positives : The cases in which we predicted **YES** and the actual output was also **YES**.

True Negatives : The cases in which we predicted **NO** and the actual output was **NO.**

False Positives : The cases in which we predicted **YES** and the actual output was **NO**.

False Negatives : The cases in which we predicted **NO** and the actual output was **YES**.

Confusion Matrix - Precision

It is the number of **correct positive results** divided by the number of **positive results predicted** by the classifier. Precision emphasises **minimizing false positives**.

Use case example: Medical tests, where **false positives** **(Predicted Diseased but actually healthy)** can lead to unnecessary treatments.

Confusion Matrix - Recall

The ratio of correctly **predicted positive** instances to the **total actual positive instances**.
Recall focuses on **minimising false negatives**.

Example

A veterinarian builds a machine learning model to detect whether cats have a specific feline disease based on medical test results.

Data:
The vet collected data on **100** cats. Here's a breakdown:

Actually Have the Disease: **40 cats**
Actually Healthy: **60 cats**
Model Predicted Positive (Disease): **45 cats**
Model Predicted Negative (Healthy): **55 cats**

Calculate Precision - Example

Precision = TP / (TP + FP)
Precision = 35 / (35 + 10) = 35/45 = 0.78 (or 78%)

Interpretation: When the model predicts a cat has the disease, it's correct **78%** of the time. That means there's a **22%** chance of overdiagnosis.

Calculate Recall - Example

Recall = TP / (TP + FN)
Recall = 35 / (35 + 5) = 35/40 = 0.875 (or 87.5%)

Interpretation: For cats that actually have the disease, the model correctly identifies them **87.5%** of the time. The remaining **12.5%** risk being missed by the model.

Precision Recall Curve

The **precision-recall curve** shows the **tradeoff between precision and recall** for **different threshold**. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.

Confusion Matrix - Example

Suppose you trained a model to classify cancer patients. You want to evaluate the model performance. You have to answer the following questions: **Accuracy? Precision?** and **Recall?**
What should be of more concerned Accuracy. Precision. Recall?

Real-World Examples

Fraud detection:

High precision to avoid **false positives** (Predicted Fraud actually not-Fraud)

Medical diagnoses:

High recall to minimize **false negatives** (Predicted healthy but actually diseased).

What is the F1-Score?

- It's the **harmonic mean** of **precision** and **recall**.
- **Balances** false positives (**precision**) and false negatives (**recall**).
- Great when you need a single number to summarize model performance.
- Especially useful for **imbalanced datasets**.
- F1-score ranges between 0 and 1. The closer it is to **1**, the better the model.

F1-Score Example: Disease Diagnosis

Scenario: A model to detect a rare disease
High Recall: The model catches most true cases of the disease (important because missing a diagnosis is dangerous).
Moderate Precision: The model sometimes gives false alarms (further tests are needed).
F1-Score: Gives a balanced view of the overall performance in this scenario.

What is True Positive Rate (TPR)?

Also called "**sensitivity**" or "**recall**"
Measures the proportion of **actual positives** that the model **correctly identifies**
Formula: TPR = True Positives / (True Positives + False Negatives)

What is True Negative Rate (TNR)?

Also called "**specificity**"
Measures the proportion of **actual negatives** that the model **correctly identifies**
Formula: TNR = True Negatives / (True Negatives + False Positives)

What is False Positive Rate (FPR)?

Measures the proportion of **actual negatives** that the model **correctly identifies**

Formula: FPR= False Positive / (True Negatives + False Positives)

AUC-ROC Curve

ROC stands for **Receiver Operating Characteristic** curve.

It plots the **True Positive Rate (TPR)** against the **False Positive Rate (FPR)**.

Shows a model's ability to separate **positive** and **negative** examples across **different thresholds.**

What is AUC?

- AUC stands for "**Area Under the Curve**".
- It represents the model's overall ability to **distinguish classes**.
- **Larger AUC** means a **better model.**
- AUC of **1**: Perfect model.
- AUC of **0.5**: No better than random guessing.

How does Regression work?

It finds the best-fitting line that represents the relationship between X and Y.

Mean Squared Error (MSE)

The **Mean Squared Error (MSE)** is a common evaluation metric used in regression tasks.
It measures the average squared difference between the actual and predicted values.

Underfitting:

In supervised learning, **underfitting** happens when a model unable to capture the underlying pattern of the data. These models usually have high bias and low variance. It happens when we have very less amount of data to build an accurate model or when we try to build a linear model with a nonlinear data. Also, these kinds of models are very simple to capture the complex patterns in data like Linear and logistic regression.

Overfitting:

In supervised learning, **overfitting** happens when our model captures the noise along with the underlying pattern in data. It happens when we train our model a lot over noisy datasets. These models have low bias and high variance. These models are very complex like Decision trees which are prone to overfitting.

Lecture 22

Agenda:

- Limitation of Conventional Models
- Neural Networks

- Why Neural Networks
- Hyper parameters
    - Learning rate
    - Epochs
    - Regularization
    - Activation functions
- Play ground

Shortcoming of Conventional ML

- Can NOT directly work with **unstructured**, **high dimensional data** (Required Feature Engineering)
  Example: Image and Video Data , Genes Data

- Can learn only so much from available data (i.e performance does not increase after certain threshold even if more training data is available)

Why is Deep Learning Popular Now ?

Most of the techniques used in deep learning are **as old as 70 years**, then why it is suddenly gaining so much traction ?

1. Data
2. Compute
3. Algorithm

Learning Rate

A hyperparameter that controls the **step size** during **gradient descent** optimization. It determines how quickly the model adjusts its parameters in the direction that **reduces** the **loss.**
A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Batch Size

Batch is a **subset** of the **training dataset** used in **each iteration** of the training process.
Instead of processing the **entire dataset** at once, we **divide** it into **smaller batches**.

Batch Size

Batch size is the number of **training examples** in each **batch.**

It's a **hyperparameter** that can be adjusted based on **hardware limitations** and **dataset characteristics**.

Larger batch sizes may **speed** up **training** but require **more memory**.

Number of Epochs

The number of **times** the entire training **dataset** is **seen** by the **model** during **training**.
Too **few epochs** might result in **underfitting**, while too **many epochs** can lead to **overfitting**.

Activation Functions (Neuron)

It determines the functions applied to the outputs of each neuron to **introduce non-linearity**.
Common activation functions include

- **ReLU (Rectified Linear Unit),**
- **Softmax**

Lecture 23

Agenda

- Neural Networks
- Why Neural Networks
- Hyper parameters
  - Learning rate
  - Epochs
  - Regularization
  - Activation functions
- Play ground

Shortcoming of Conventional ML

- Can NOT directly work with **unstructured**, **high dimensional data** (Required Feature Engineering)
  Example: Image and Video Data , Genes Data

- Can learn only so much from available data (i.e performance does not increase after certain threshold even if more training data is available)

Why is Deep Learning Popular now ?

Most of the techniques used in deep learning are **as old as 70 years**, then why it is suddenly gaining so much traction ?

1. Data
2. Compute
3. Algorithm

Perceptron

Perceptron is a **supervised learning** algorithm in machine learning that works by **combining inputs with weights**, passing them through an **activation function**, and making binary decisions based on the output.

Feed Forward Neural Networks
FNNs play a crucial role in modern machine learning by **processing input data through**

interconnected layers of neurons to produce meaningful outputs based on learned patterns. This network architecture is widely used in machine learning for tasks like image classification, natural language processing, and time series prediction.

Radial Basis Function Neural Networks

RBFNNs are a specific type of Artificial Neural Network **used for function approximation tasks**. They differ from other NNs in their three-layer architecture, universal approximation capabilities, and faster learning speed.

Deep Feed Forward Neural Network

These networks are particularly effective for tasks involving **high-dimensional input data** like images and text but can be challenging to train effectively. DFF processes input data through **multiple hidden layers** to learn complex features and patterns in the data, making them powerful tools for tasks such as image classification, natural language processing, and prediction.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a type of artificial neural network designed for tasks involving sequential data like natural language processing, **speech recognition, and time series prediction**.

Long Short-Term Memory

LSTM networks are a type of RNN designed to **address the vanishing gradient problem commonly encountered in traditional RNNs**. LSTMs excel in capturing long-term dependencies in sequential data, making them ideal for tasks like language translation, speech recognition, and time series forecasting.

Gated Recurrent Unit

GRU is a type of RNN **that addresses challenges in processing sequential data**. GRUs are designed to overcome the vanishing gradient problem encountered in traditional RNNs by incorporating gating mechanisms. They find applications in various domains like natural language processing, speech recognition, and time series prediction.

Autoencoder

Autoencoders are a specialized class of algorithms within ANNs designed for **unsupervised learning**. They work by **learning efficient representations of input data without specific labels**. The core principle of an autoencoder involves a two-fold structure consisting of an encoder and a decoder.

Variational Autoencoder

A VAE is an ANN architecture that **combines an encoder, a decoder, and a loss function**. It introduces a regularizer, the Kullback-Leibler divergence, to ensure the latent space is well-structured. VAEs aim to learn efficient representations of data by encoding input into a latent space and decoding it back, enabling tasks like data generation and compression.

Denoising AE

A DAE is a type of NN that is trained to **remove noise from input data**. It works by reconstructing clean data from corrupted or noisy input, helping to enhance the quality of the data and improve

generalization capabilities in tasks like image denoising and signal processing.

Sparse AE

A Sparse Autoencoder (SAE) is a variant of autoencoders that enforces sparsity in the learned representations, aiming to have only a few active neurons at a time. This approach helps in creating an information bottleneck and improving performance on tasks like dimensionality reduction and feature learning.

Deep Convolutional Network

A deep neural network architecture that **utilizes deconvolution layers**, often used for tasks like **image deconvolution**, featuring a hierarchical decomposition of input data to extract features at different levels.

Deconvolution Network

Deconvolution Network (DN): A network that employs reversed convolutional layers to reconstruct lost features or signals, commonly used in image synthesis and analysis tasks.

Deep Convolutional Inverse Graphics Network

The DC-IGN is a NN model that learns interpretable representations of images by disentangling transformations like rotations and lighting variations. It utilizes convolution and deconvolution operators and is trained with the Stochastic Gradient Variational Bayes algorithm to generate new images with variations in pose and lighting based on learned latent variables.

Hyper-parameters

A hyper-parameter in machine learning is a **parameter set before training that controls the learning process** and **influences the model's learned parameters**. These external configuration variables, like learning rate or batch size, are crucial for optimizing model performance and are distinct from internal model parameters.

Learning Rate

A hyperparameter that controls the **step size** during **gradient descent** optimization. It determines how quickly the model adjusts its parameters in the direction that **reduces** the **loss.**
A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Learning Rate

A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Batch Size

Batch is a **subset** of the **training dataset** used in **each iteration** of the training process.

Instead of processing the **entire dataset** at once, we **divide** it into **smaller batches**.

Batch Size

Batch size is the number of **training examples** in each **batch.**

It's a **hyperparameter** that can be adjusted based on **hardware limitations** and **dataset characteristics**.

Larger batch sizes may **speed** up **training** but require **more memory**.

Number of Epochs

The number of **times** the entire training **dataset** is **seen** by the **model** during **training**.

Too **few epochs** might result in **underfitting**, while too **many epochs** can lead to **overfitting**.

Activation Functions (Neuron)

It determines the functions applied to the outputs of each neuron to **introduce non-linearity**.
Common activation functions include

- **ReLU (Rectified Linear Unit),**
- **Softmax**

Lecture 24

Agenda

- Introduction of the Course ?
- Machine Learning vs Deep Learning
- Natural Language Processing
- Retrieval-Augmented Generation

Artificial Intelligence

Artificial Intelligence is the **branch** of **computer science** concerned with development of **methods** that **allow computers to learn without explicit programming.**

Machine Learning

Machine Learning is a **branch of AI**, which focuses on **methods that** can **learn from examples** and experience **instead** of relying on **hard-coded rules** and make predictions on new data.

Deep Learning

Subfield of **Machine Learning** that focuses on **Neural Networks** (Inspired from Biological neurons) to develop **learning models.**

Relationship in AI, ML and DL

Artificial Intelligence:
Search Algorithms, Rule Based Systems, Statistical Inference, Machine Learning

Machine Learning:
SVM, Tree Algorithms, Nearest neighbours, bagging, boosting, Deep Learning

Deep Learning:
FCNs, CNNs, RNNs, Transformers, Autoencoders, GANs

Natural Language Processing

Natural language processing (NLP) is the discipline of building machines that can manipulate human language — or data that resembles human language — in the way that it is written, spoken, and organized.

What is Natural Language Processing (NLP) Used for? (Contd)

- Spam detection
- Grammatical error correction
- Information retrieval
- Summarization
- Question answering
- Chatbots
- Market Intelligence
- Speech recognition

Generative AI

Generative AI, also **known as GenAI** is a type of **artificial intelligence** that **creates new content** based on the **patterns learned from existing data**. These algorithms can generate text, images, videos, audio, code etc

Generative AI

1. Text Generation
    a. ChatGPT 3.5/4
    b. Bard
2. Image generation
    a. Dall-E3
    b. Stable Diffusion
3. Music Generation
    a. Music-LM
4. Video Generation
    a. Runway ML Gen-2

Large Language Model (LLM)

A large language model (LLM) is a type of artificial intelligence (AI) program that can recognize and generate text, among other tasks. LLMs are trained on huge sets of data — hence the name "large." LLMs are built on machine learning: specifically, a type of neural network called a transformer model.

It's akin to distinguishing between an open-book and a closed-book examination.

In a RAG system, you're prompting the model to provide a response by perusing content within a book, rather than relying on recalling facts from memory.

Retrieval Augmented Generation

Retrieval-augmented generation (RAG) is a technique for enhancing the accuracy and reliability of generative AI models with facts fetched from external sources.

RAG is an **AI framework** for retrieving facts from an external knowledge base to ground large **language models (LLMs)** on the most accurate, up-to-date information and to give users insight into LLMs' generative process.

Use Cases

Picture a bustling hospital where doctors navigate intricate medical cases daily.

Physicians diagnose and treat patients based on their vast medical knowledge. Yet, when confronted with complex illnesses or rare conditions, they often seek assistance from medical literature or consult with specialists.

Use Cases(Contd.)

In the realm of artificial intelligence, large language models (LLMs) function much like skilled doctors, adept at addressing a wide array of medical queries. However, to deliver well-informed responses backed by credible sources, they rely on a process analogous to a doctor's consultation with medical literature or specialists.

Building User Trust

Retrieval-augmented generation gives models sources they can cite, like footnotes in a research paper, so users can check any claims. That builds trust.

Lecture# 25

Agenda

- Problem with linear Models
- Neural Networks
- Why Neural Networks
- Hyper parameters

- Learning rate
- Epochs
- Regularization
- Activation functions

Learning Rate

A hyperparameter that controls the **step size** during **gradient descent** optimization. It determines how quickly the model adjusts its parameters in the direction that **reduces** the **loss.**

A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Learning Rate

A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Batch Size

Batch is a **subset** of the **training dataset** used in **each iteration** of the training process.

Instead of processing the **entire dataset** at once, we **divide** it into **smaller batches**.

Batch Size

Batch size is the number of **training examples** in each **batch.**

It's a **hyperparameter** that can be adjusted based on **hardware limitations** and **dataset characteristics**.

Larger batch sizes may **speed** up **training** but require **more memory**.

Number of Epochs

The number of **times** the entire training **dataset** is **seen** by the **model** during **training**.

Too **few epochs** might result in **underfitting**, while too **many epochs** can lead to **overfitting(Might be controlled with Early Stopping)**.

Activation Functions (Neuron)

It determine the functions applied to the outputs of each neuron to **introduce non-linearity**.

Common activation functions include

- **ReLU (Rectified Linear Unit)** output the input directly if it is positive, otherwise, it will output zero(**f(x) = max(0, x)**),
- **Softmax** converts a vector of numbers into a vector of probabilities, all probabilities **sum to the value 1.0**, The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities.

The Building Blocks of ANN

- Neurons and layers are fundamental components of ANNs.
- Neurons in ANNs process information similarly to brain neurons but digitally:
  - Receive inputs
  - Apply weights to inputs
  - Sum the inputs
  - Pass the result through an activation function
- Layers of interconnected neurons each have specific roles:
  - **Input layer:** Receives raw data
  - **Hidden layers:** Perform complex computations
  - **Output layer:** Provides the final result or prediction

How ANN Learns

- Training ANNs involves using labeled datasets to improve predictive accuracy.
- ANNs automatically extract features through their interconnected layers, unlike traditional algorithms.
- Inspired by the brain's neural network, ANNs self-learn and adapt without explicit programming.
- ANNs handle complex tasks like image recognition, natural language processing, and autonomous driving.

Summary

- Advantages
- - Can handle complex data
- - Non-linear modeling capabilities
- - Adaptability and learning capabilities
- - Robustness to noisy or incomplete data
- - Feature extraction capabilities
- - Domain agnostic, applicable to various business areas
- - Parallel processing for efficient computation
- - Can handle high-dimensional data
- - Can uncover hidden patterns and insights
-
- -Disadvantages
- - Requires large amounts of labeled training data
- - Black box nature can hinder interpretability
- - Computationally intensive and resource-consuming
- - Lack of transparency in decision-making
- - Potential overfitting without proper regularization
- - Complexity and difficulty in model tuning
- - Difficulty in explaining results to stakeholders
- - Sensitivity to input data quality and preprocessing
- - Ethical considerations in sensitive decision-making

What is TensorFlow?

TensorFlow is an open-source machine learning **framework** developed by **Google.**

It enables **building**, **training**, and **deploying** machine learning models.

Primarily used for **deep learning** tasks, it's versatile for various other types of machine learning.

Tensorflow

Lecture# 26

Agenda

- Artificial Neural Networks
- Hyper parameters
  - Learning rate
  - Epochs
  - Activation functions
  - Batch Size
- Forward/Back Propagation
- Limitations of Simple Neural Network

Gradient Descent

A hyperparameter that controls the **step size** during **gradient descent** optimization. It determines how quickly the model adjusts its parameters in the direction that **reduces** the **loss.**

A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Learning Rate

A **higher learning** rate might lead to **faster convergence** but **risks overshooting**, while a **lower rate** might slow down convergence.

Batch Size

Batch is a **subset** of the **training dataset** used in **each iteration** of the training process.

Instead of processing the **entire dataset** at once, we **divide** it into **smaller batches**.

Batch Size

Batch size is the number of **training examples** in each **batch.**

It's a **hyperparameter** that can be adjusted based on **hardware limitations** and **dataset characteristics**.

Larger batch sizes may **speed** up **training** but require **more memory**.

Number of Epochs

The number of **times** the entire training **dataset** is **seen** by the **model** during **training**.

Too **few epochs** might result in **underfitting**, while too **many epochs** can lead to **overfitting**.

Activation Functions (Neuron)

It determine the functions applied to the outputs of each neuron to **introduce non-linearity**.

Common activation functions include

- **ReLU (Rectified Linear Unit)** (0, ∞)**,**
- **Leaky ReLU** (-∞, ∞)
- **Hyperbolic Tangent** (Tanh) (-1, 1)
- **Softmax** (0, 1) used for Multiclass Classification
- **Sigmoid** (0, 1) used for Binary Classification

Forward Pass

1. **Input Data**: Inputs (X1 and X2) are fed to the input layer.

2. **Weighted Inputs**: Each input is multiplied by its corresponding weight and sent to hidden layer neurons (N1X and N2X).

3. **Activation:** Hidden layer neurons apply an activation function to the weighted inputs, passing the results to the next layer.

Error Calculation

1. **Output Generation:** The network processes inputs until the output layer produces the final output.

2 Comparison: The network's output is compared to the ground truth (desired output).

3. **Error Calculation**: The difference between the output and ground truth is computed, yielding an error value.

Backward Pass

1. Gradient Calculation: Use the error value to calculate the gradient of the loss function.

2. Error Propagation: Propagate the error gradient back through the network from the output layer to hidden layers.

3. Weight Updates: Update weights based on their contribution to the error, using the derivative of the error with respect to each weight.

4. Learning Rate: The size of weight updates is determined by the learning rate; smaller rates result in smaller updates.

Weights Update

1. Gradient Descent: Weights are adjusted in the **opposite** direction of the **gradient** to reduce error.

2. Iterative Process: Forward pass, error calculation, backward pass, and weight updates are repeated for multiple epochs.

3. Convergence: The process continues until network performance is satisfactory.

Limitations of Simple Neural Networks

- **Pixel Equality:** Simple neural networks treat all pixels in an image equally, regardless of their spatial relationships or positions
- **High Dimensionality:** Large sized Image have more pixels and model needs more weights to learn. **28x28=784, 256x256=65536**
- **Limited Feature Extraction:** Simple networks lack specialized layers to automatically extract relevant features from images, making it harder to identify complex structures

Limitations of Simple Neural Networks(Contd)

- **Large Computational Requirements:** Larger no of weights requires more computational resources to train.
- **Overfitting Risk:** More prone to overfitting due to a higher number of parameters.
- **Scalability Issues:** Struggles with scaling to large, complex datasets efficiently.

Lecture 27

1. Text in Black have instructions
2. Text in Blue are the content that have to be on slides
3. Text in Red need to be highlighted in slides
4. If you're using any images from the document then add its reference.

Previous Lecture Topics

1. Artificial Neural Networks
2. Hyper parameters
3. Learning rate
4. Epochs
5. Activation functions
6. Batch Size
7. Forward/Back Propagation
8. Limitations of Simple Neural Network

Lecture Objectives:

1. Understand the basic architecture of CNNs, including convolutional layers, pooling layers, and fully connected layers
2. Learn about how CNNs greatly improve computer vision tasks, like recognizing objects in images, making deep learning much more powerful in this area.

Outline:

- Limitations of simple Neural Network
- Introduction to Convolutional Neural Networks (CNN)
- Basics for CNN
  - Images and features
  - Kernel
  - Convolution
  - Number of filters, Depth
  - Stride
  - Pooling
  - Receptive Field
- Quiz

Limitation in Simple Neural Networks

1. **Pixel Equality:** Simple neural networks treat all pixels in an image equally, regardless of their spatial relationships or positions
2. **High Dimensionality:** Large sized Image have more pixels and model needs more weights to learn. **28x28=784, 256x256=65536**
3. **Limited Feature Extraction:** Simple networks lack specialized layers to automatically extract relevant features from images, making it harder to identify complex structures
4. **Large Computational Requirements:** Larger no of weights requires more computational resources to train.

On the right side of the slide, there should be a neural network image, and the left side will feature bullet points outlining limitations. Please refer to the image below for clarification.

Introduction to Convolutional Neural Network

- **CNNs** are a type of **Neural Networks** inspired by how our **brain** processes **visual information**.

- They're **specialized** for understanding **images**, making them powerful tools in image analysis, recognition, and more.

Text should be animated (add animation in both bullets) and diagram too. The box in the input image should move from the top left corner, going to the right until it reaches the end of the row, then it moves to the next row and continues the same pattern until it reaches the end of the image. And this movement is replicated for the feature maps as well.

Images in Computer

Features for Images

- Images are full of important parts called **features**.
- Features can be **edges, textures, shapes, or even faces**.

Add animation in both bullet points

Kernel

- Kernels in CNNs are small 2D matrices used for feature extraction through convolution operations on input data, typically with dimensions in odd numbers.

Add three matrices 3*3, 5*5 and 7*7 (for reference see the image below)

Convolution

- At each position, the kernel performs element-wise multiplication with the corresponding part of the input data it covers. Then, the results of these multiplications are summed up to produce a single output pixel.
- Convolution is like a special way of looking at data. It's a mathematical operation that brings out patterns.

Convolutional Kernels

- The kernel is that matrix which is swiped, or more precisely convolved across a single channel of the input Image.
- After performing the convolution operation on an image of size(h,w,in_dims), we're left with a single channeled feature map of size (h', w', out_dims)

Example

Add animation in the picture- firstly the input image will display(right matrix from above image), then display the kernel (matrix in the middle) then display the output (left matrix). Caption of the image should be "Vertical Edge Detection using Kernels"

Depth

In CNNs, each layer consists of multiple filters (kernels), each producing a separate feature map. The number of filters in a layer determines its depth.

Stride

- Stride in CNN determines the step size of the filter as it moves across the input data during the convolution operation, affecting output size and computational efficiency.

For a stride of 1, the rectangle moves pixel by pixel from left to right, covering each pixel consecutively. For a stride of 2, the rectangle moves with a gap of two pixels, skipping every other pixel as it progresses across the input.

Pooling

Pooling is like **summarizing information**, keeping what's **important**.
It helps **reduce** the **dimensionality** while retaining **important information.**
Pooling = Downsizing + Information Summary.

:Types of Pooling

In this example stride=2

Advantages of Max Pooling:

Feature Retention: Keeps dominant features intact.
Translation Invariance: Recognizes features regardless of exact location.
Dimension Reduction: Reduces data size, providing efficiency.

How to calculate the output of CNN

From the above image just write the equations and remove text (output height after convolution or pooling etc). Add text “width of the input data = ” before W_{out} , “Height of the input data = ”
before H_{out} is height of input data. And “Dept = “ before D

Padding
Padding is a technique used to maintain the spatial dimensions of the input image after performing convolution operations on a feature map.

Types of Padding

- Valid Padding
- Same Padding

Slide Title: Valid Padding:
In the valid padding, no padding is added to the input feature map, and the output feature map is smaller than the input feature map. This is useful when we want to reduce the spatial dimensions of the feature maps.

Same Padding:

Padding is added to the input feature map such that the size of the output feature map is the same as the input feature map.

This is useful when we want to preserve the spatial dimensions of the feature maps.

Slide: Types of Same Padding

- Zero Padding
- Ones Padding
- Replace Input Pixels

Receptive field:

For first layer ReceptiveField$_{input}$ is kernel size

$$\text{Receptive Field}_{\text{output}} = \text{Receptive Field}_{\text{input}} + (\text{Kernel Size} - 1) \times \text{Layer Stride}$$

Write equation

Pooling

Simplifying Complex Data:

Pooling is like summarizing information, keeping what's important.

Dimensionality Reduction: It helps reduce the dimensionality while retaining essential information.

Key Concept: Pooling = Downsizing + Information Summary

Types of Pooling:

- **Max Pooling**: Selects the maximum value from a set of values. Think of it as choosing the most important feature.
- **Average Pooling**: Computes the average of a set of values. It smooths out the representation.
- **Global Pooling:** Reduces the entire feature map to a single value by either taking the maximum or the average
- Slide Title**:Types of Pooling**

In this example stride=2

Advantages of Max Pooling:

Feature Retention: Keeps dominant features intact.
Translation Invariance: Recognizes features regardless of exact location.
Dimension Reduction: Reduces data size, providing efficiency

- Analogies to Understand Pooling: Image Compression Analogy:

Pooling is like compressing an image file. You lose some details, but the essential features are still there.

- *[Insert an image of a detailed picture and its compressed version]*
- Teacher Analogy
- Imagine a teacher summarizing a chapter. They highlight the key points (max pooling) or give an overall summary (average pooling), making it easier for students to remember the main ideas.
- *[Insert an image of a teacher highlighting key points from a text]*

How to calculate the output of CNN

From the above image just write the equations and remove text (output height after convolution or pooling etc). Add text “width of the input data = ” before W_{out} , “Height of the input data = ”
before H_{out} is height of input data. And “Dept = “ before D
Slide Title: **Receptive field:**

The receptive field in a Convolutional Neural Network (CNN) refers to the region of the input image that a particular feature (or neuron) in a convolutional layer is "looking at" or is responsive to. As you go deeper into the network, the receptive field increases, meaning that neurons in deeper layers are influenced by a larger portion of the input image.

Lecture:28

Small View: In the early layers of the network, each neuron looks at a small part of the image. Big View: As you go deeper into the network, neurons look at larger parts of the image.

For first layer $\text{ReceptiveField}_{input}$ is kernel size

Write equation

Add animation from left side matrix to right side matrix.

Analogy:

Magnifying Glass: Imagine looking at a picture through a magnifying glass. At first, you see only a tiny part of the picture (small receptive field). As you move the magnifying glass away, you see more of the picture (large receptive field)

Why is this important

Feature Detection: Helps the network detect simple patterns like edges in the beginning and more complex patterns like shapes and objects in later layers.

Context Understanding: Neurons in deeper layers understand the context by looking at bigger parts of the image.

Slide Title:**Object Recognition Process at a Glance from PIxel to Detection**

Lecture:29, 30,31

Types of Autoencoders

Vanilla Autoencoder
Denoising Autoencoder
Covolutional Autoencoder
Variational Autoencoder (VAE)

Vanilla Autoencoder

Definition: A Vanilla Autoencoder is a type of artificial neural network used to learn efficient codings of input data in an unsupervised manner.
Structure:

- **Encoder**: Maps input data to a lower-dimensional latent space.
- **Latent Space**: Encoded representation of the input data, typically smaller in size.
- **Decoder**: Maps the encoded data back to the original input space.

Architecture: Consists of an input layer, hidden layers for the encoder and decoder, and an output layer.
Loss Function: Measures the difference between the input and the reconstructed output, often Mean Squared Error (MSE).
Training: The network is trained to minimize the reconstruction error by adjusting the weights using backpropagation.
Purpose: Dimensionality reduction, feature learning, and noise reduction.
Assumptions: The input data has an underlying structure that can be captured in a lower-dimensional space.
Advantages: Simplicity, ability to learn complex patterns, and no need for labeled data.
Limitations: May not perform well on highly complex data, sensitive to hyperparameter choices, and prone to overfitting without regularization.
Applications: Image compression, anomaly detection, data denoising, and dimensionality reduction for visualization.

https://pyimagesearch.com/2023/07/10/introduction-to-autoencoders/

Denoising Autoencoder

- **Definition**: A Denoising Autoencoder (DAE) is a type of autoencoder designed to remove noise from data by learning a robust representation.During training, the input data is intentionally corrupted by adding noise, while the target remains the original, uncorrupted data.
- **Structure**:
    - **Encoder**: Maps noisy input data to a latent space representation.
    - **Latent Space**: Encoded, denoised representation of the input data.
    - **Decoder**: Reconstructs the denoised input from the latent space representation.
- **Training Process**:
    - **Input Corruption**: Random noise is added to the input data.
    - **Reconstruction**: The autoencoder is trained to reconstruct the original, noise-free data from the corrupted input.
- **Loss Function**: Measures the difference between the original (clean) input and the reconstructed output, often using Mean Squared Error (MSE).
- **Purpose**: To learn to denoise data, enhancing the robustness of the learned features.
- **Advantages**: Improved feature learning, robustness to noisy data, and better generalization.
- **Applications**: Image denoising, audio denoising, and pre-processing for machine learning tasks.

- **Noise Types**: Can include Gaussian noise, salt-and-pepper noise, or any other form of corruption.

https://pyimagesearch.com/2023/07/10/introduction-to-autoencoders/

https://deepai.org/machine-learning-glossary-and-terms/denoising-autoencoder

https://pyimagesearch.com/2023/07/10/introduction-to-autoencoders/

- A Convolutional Autoencoder (CAE) is a type of autoencoder that uses convolutional layers to learn efficient codings of image data.
- **Structure**:
    - **Encoder**: Uses convolutional layers to extract hierarchical features from the input image, reducing its spatial dimensions.
    - **Latent Space**: Encoded representation with reduced dimensions, capturing the essential features of the input image.
    - **Decoder**: Uses transposed convolutional layers (or upsampling) to reconstruct the input image from the latent representation.
- **Purpose**: To perform tasks such as image compression, denoising, and feature extraction for images.
- **Advantages**: Exploits spatial hierarchies in images, reduces the number of parameters, and improves generalization.
- **Training**: The network is trained to minimize the reconstruction error, often using Mean Squared Error (MSE) as the loss function.
- **Applications**: Image denoising, image compression, anomaly detection in images, and pre-processing for other vision tasks.
- **Pooling**: Max-pooling or average-pooling layers are often used in the encoder to reduce spatial dimensions.
- **Upsampling**: Transposed convolutions or other upsampling techniques are used in the decoder to restore the original dimensions.
- **Regularization**: Techniques like dropout and weight decay can be used to prevent overfitting.
- **Hyperparameters**: Include the number of convolutional layers, filter sizes, stride lengths, learning rate, and number of filters per layer.
- **Advantages Over Fully Connected Autoencoders**: Better suited for image data, fewer parameters, and ability to capture spatial hierarchies.
- **Performance**: Effectiveness depends on the architecture, type of convolutional layers, and tuning of hyperparameters.

https://www.researchgate.net/figure/The-structure-of-proposed-Convolutional-AutoEncoders-CAE-for-MNIST-In-the-middle-there_fig1_320658590

https://www.researchgate.net/figure/Structure-of-the-convolutional-autoencoder-with-associated-layer-parameters-Best-viewed_fig2_305722375

Variational Autoencoder (VAE)

A major issue with regular Auto encoders is that the latent space that the inputs are converted to are discrete (not continuous) and does not allow for an easy interpolation.

The generative part of the auto encoder works by randomly picking samples from the latent space which is challenging if it's discontinuous or has gaps.

Variational Autoencoders are here to solve this issue

Variational Auto Encoders have a continuous latent space by default which make them super powerful in generating new images.

In VARs, the encoder does not generate a vector of size n but it generates two vectors instead as follows:

Vector mean μ

Standard deviations σ

Decoder (Generative Model): The decoder maps samples from the latent space back to the input space, generating new data points.

If you want to apply variational auto encoders to add glasses to a face for example, you can do so by:

(1) encoding a face with glasses, and

(2) encoding a face without glasses and

(3) simply subtracting the two, you've obtained an encoded version of the glasses alone,

(4) You can then add this encoded glasses to any face later on

Basic Concepts

1. **Autoencoders**: An autoencoder is a neural network used for unsupervised learning of efficient codings. It consists of two parts:
    - **Encoder**: Maps input data to a lower-dimensional latent space.
    - **Decoder**: Reconstructs the input data from the latent space.
2. **Variational Inference**: This is a technique used to approximate complex probability distributions. It's used to infer the latent variables in probabilistic models.

Loss Function

The loss function of a VAE consists of two parts:

1. **Reconstruction Loss**: Measures how well the VAE reconstructs the input data. Typically, this is the mean squared error or binary cross-entropy between the original data and the reconstruction.
2. **KL Divergence**: A regularization term that ensures the learned latent space distribution is close to the prior distribution (usually a standard Gaussian). It measures the divergence between the approximate posterior and the prior distribution.

Applications

VAEs have a wide range of applications, including:

- **Data Generation**: Creating new data samples similar to the training data.
- **Representation Learning**: Learning compact, meaningful representations of data.
- **Anomaly Detection**: Identifying unusual data points by their reconstruction error.

Key Advantages

- **Smooth Latent Space**: The latent space learned by VAEs is continuous and allows for smooth interpolation between data points.
- **Probabilistic Framework**: Provides a principled way to handle uncertainty in the learned representations.

Note:
Autoencoder is an unsupervised learning method if we considered the latent code as the "output".

• Autoencoder is also a self-supervised (self-taught) learning method which is a type of supervised learning where the training labels are determined by the input data.

Lecture:32

https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

https://www.ibm.com/topics/recurrent-neural-networks

- Title: Recurrent Neural Networks (RNNs)
- Subtitle: Understanding the Fundamentals and Applications
- Your Name
- Date

What are RNNs?

Recurrent Neural Networks (RNNs) are a class of neural networks designed to recognize patterns in sequences of data, such as time series or natural language.

Key characteristics of RNNs RNNs have an internal state (memory) that captures information about previous inputs, making them well-suited for tasks involving sequential data where context is crucial.

Difference between RNNs and feedforward neural networks

Unlike feedforward neural networks, which process inputs independently, RNNs use their internal state to maintain temporal context, allowing them to process sequences in a way that accounts for past inputs.

Structure of RNNs

RNN Architecture

RNNs are a type of neural network that utilizes hidden states and allows past outputs to be used as inputs.

Input Layer: This layer receives the initial element of the sequence data. For example, in a sentence, it might receive the first word as a vector representation.

Hidden Layer: The heart of the RNN, the hidden layer contains a set of interconnected neurons. Each neuron processes the current input along with the information from the previous hidden state. This state captures the network's memory of past inputs, allowing it to understand the current element in context.

Activation Function: This function introduces non-linearity into the network, enabling it to learn complex patterns. It transforms the combined input from the current input layer and the previous hidden state before passing it on.

Output Layer: The output layer generates the network's prediction based on the processed information. In a language model, it might predict the next word in the sequence.

Recurrent Connection: A key distinction of RNNs is the recurrent connection within the hidden layer. This connection allows the network to pass the hidden state information (the network's memory) to the next time step, akin to passing a baton in a relay race, carrying information about previous inputs forward.

Slide: What Makes RNNs Special?

Recurrent Neural Networks (RNNs) distinguish themselves from other neural networks with their unique features:

Internal Memory: RNNs can remember past inputs and use this context when processing new information, which is their key feature.

Sequential Data Processing: Thanks to their memory, RNNs excel at handling sequential data where the order of elements is important,

making them ideal for tasks such as speech recognition, machine translation, natural language processing (NLP), and text generation.

Contextual Understanding: RNNs analyze current input in relation to previous inputs, providing crucial contextual understanding for tasks where meaning depends on prior information.

Dynamic Processing: RNNs continuously update their internal memory as they process new data, allowing them to adapt to changing patterns within a sequence.

RNN Variants

Different types of RNNs: Simple RNN, LSTM, GRU

- **Simple RNN:** The most basic form of RNN, where the hidden state is updated with a simple activation function like tanh or ReLU. It captures temporal dependencies but struggles with long-term dependencies due to vanishing and exploding gradients.
- **Long Short-Term Memory (LSTM):** LSTM networks introduce memory cells and gates (input, forget, and output gates) to control the flow of information, effectively mitigating the vanishing gradient problem and capturing long-term dependencies.
- **Gated Recurrent Unit (GRU):** GRU is a simplified version of LSTM with fewer gates (reset and update gates), which makes it computationally more efficient while still handling long-term dependencies well.

Brief comparison of these variants

- **Memory Capacity:** LSTMs and GRUs handle long-term dependencies better than simple RNNs due to their gating mechanisms.
- **Complexity:** LSTMs are more complex than GRUs due to the additional gates and parameters, whereas GRUs strike a balance between simplicity and performance.
- **Training Time:** Simple RNNs are faster to train but may not perform well on tasks requiring long-term context. LSTMs and GRUs take longer to train but generally provide better performance on such tasks.
- **Use Cases:** Simple RNNs are suitable for tasks with short-term dependencies. LSTMs and GRUs are preferred for tasks like language modeling, speech recognition, and time series prediction where long-term context is crucial.

Training RNNs

How RNNs are trained RNNs are trained using a process similar to other neural networks, where the goal is to minimize a loss function by adjusting weights through gradient descent.

The key difference lies in handling the sequential nature of the data, requiring gradients to be calculated across multiple time steps.

Backpropagation Through Time (BPTT) BPTT is an extension of backpropagation used for training RNNs. It involves unfolding the RNN across time steps, treating each time step as a layer in a deep feedforward network. Gradients are then calculated and propagated backwards through each time step, updating weights to minimize the loss.

Common challenges: vanishing and exploding gradients

- **Vanishing Gradients:** As gradients are propagated backwards through many time steps, they can become very small, making it difficult for the network to learn long-term dependencies. This is a significant issue in simple RNNs, limiting their ability to capture long-range patterns.
- **Exploding Gradients:** Conversely, gradients can also grow exponentially, leading to numerical instability and making the network's weights update too drastically. This can cause the training process to diverge.
- **Mitigation Strategies:** Techniques like gradient clipping (limiting the size of gradients) and using specialized architectures like LSTMs and GRUs help address these challenges, enabling the training of RNNs on long sequences effectively.
-

Applications of RNNs

Natural Language Processing (NLP): text generation, sentiment analysis

- **Text Generation:** RNNs can generate coherent text by predicting the next word in a sequence based on previous words, making them suitable for applications like writing assistants, story generation, and chatbots.
- **Sentiment Analysis:** By processing text data sequentially, RNNs can analyze the sentiment of a given text (positive, negative, or neutral) based on the context provided by preceding words, useful for tasks like social media monitoring and customer feedback analysis.

Speech Recognition RNNs transcribe spoken language into text by capturing temporal dependencies in audio signals, recognizing phonemes, words, and sentences. This forms the basis for voice-activated systems like virtual assistants (e.g., Siri, Alexa) and automated transcription services.

Time Series Prediction RNNs excel at predicting future values of time series by analyzing historical data, making them valuable in finance, weather forecasting, and demand prediction. Their ability to retain past information enables accurate forecasting of trends and anomalies.

Long Short-Term Memory (LSTM)

Introduction to LSTMs STM networks, a specialized type of RNN introduced by Hochreiter and Schmidhuber in 1997, are designed to capture long-term dependencies and address the

vanishing gradient problem. They use memory cells and gating mechanisms to remember information over extended periods

Structure and Working of LSTM Cells

- **Memory Cell:** The core of the LSTM, it retains information over time. It decides what to remember and what to forget.
- **Input Gate:** Controls the extent to which new information flows into the memory cell. It determines which parts of the input should be updated.
- **Forget Gate:** Regulates what information should be discarded from the memory cell. It helps in resetting the cell's state when necessary.
- **Output Gate:** Decides what part of the memory cell's content is output. It controls the flow of information from the cell to the next hidden state and the network's output.
- **Cell State:** A highway that transports information across the sequence. It undergoes linear transformations, reducing the risk of vanishing or exploding gradients.
- **Working:** At each time step, the LSTM cell takes the previous hidden state, the current input, and the cell state, passing them through the gates to produce the current hidden state and updated cell state.

Advantages of LSTM over Simple RNNs

- **Long-Term Dependency Handling:** LSTMs are designed to retain information for long periods, making them effective at learning long-range patterns in data.
- **Mitigated Vanishing Gradient Problem:** The gating mechanisms of LSTMs ensure that gradients are preserved during backpropagation, preventing them from vanishing or exploding.
- **Selective Memory:** LSTMs can selectively remember and forget information, which allows them to focus on relevant data and ignore noise.
- **Versatile Applications:** Due to their robustness, LSTMs are widely used in applications like language modeling, machine translation, speech recognition, and time series forecasting.
-

Gated Recurrent Unit (GRU)

- **Introduction to GRUs** Gated Recurrent Units (GRUs) are a variation of RNNs that address the limitations of simple RNNs by introducing gating mechanisms. Proposed by Cho et al. in 2014, GRUs are designed to capture long-term dependencies more effectively and efficiently than traditional RNNs and even Long Short-Term Memory (LSTM) networks.
- **Comparison with LSTM**
- **Complexity:** GRUs are simpler than LSTMs, with fewer gates and parameters. This simplicity often leads to faster training and less computational overhead.
- **Performance:** GRUs and LSTMs can perform similarly on many tasks. However, GRUs may be preferred in scenarios where computational efficiency is a priority, while LSTMs might be chosen for tasks requiring more complex memory management.

- **Gating Mechanisms:** LSTMs use three gates (input, forget, and output) and a separate memory cell, while GRUs use only two gates (update and reset) and combine the memory cell with the hidden state, simplifying the architecture.
- **Applications:** Both GRUs and LSTMs are used for tasks requiring long-term dependency modeling, such as natural language processing, speech recognition, and time series forecasting. The choice between them often depends on the specific needs of the application and computational constraints.
-

Conclusion

- Recap of key points
- Future trends in RNN research
- Q&A session

Lecture #33,35

The Encoder-Decoder Model with RNNs/ Sequence-to-sequence (seq2seq) models in NLP are used to convert sequences of Type A to sequences of Type B. For example, translation of English sentences to German sentences is a sequence-to-sequence task.

Encoder-decoder networks have been applied to a very wide range of applications including summarization, question answering, and dialogue, but they are particularly popular for machine translation.

Google Translate started using such a model in production in late 2016.

Looking under the hood

Under the hood, the model is composed of an encoder and a decoder.

The encoder processes each item in the input sequence, it compiles the information it captures into a vector (called the context). After processing the entire input sequence, the encoder sends the context over to the decoder, which begins producing the output sequence item by item.

The same applies in the case of machine translation.

The context is a vector (an array of numbers, basically) in the case of machine translation. The encoder and decoder tend to both be recurrent neural networks

Slide:Encoder-decoder networks/Seq-2-Seq models Architecture

Encoder-decoder networks consist of three conceptual components:

1. An encoder that accepts an input sequence, x, and generates a corresponding sequence of contextualized representations, h. LSTMs, convolutional networks, and transformers can all be employed as encoders.

2. A context vector, c, which is a function of h, and conveys the essence of the input to the decoder.

3. A decoder, which accepts c as input and generates an arbitrary length sequence of hidden states h, from which a corresponding sequence of output states y, can be obtained. Just as with encoders, decoders can be realized by any kind of sequence architecture

Bottleneck problem

- The bottleneck problem in seq-to-seq models arises from compressing an entire input sequence into a single fixed-size context vector.
- This fixed-size vector may fail to capture all relevant details, especially for long sequences, leading to information loss.
- The encoder's output, limited to a fixed dimension, cannot adequately represent long-range dependencies within the input sequence.
- As a result, the model's performance degrades on tasks involving lengthy or complex sequences.
- Attention mechanisms address this by allowing the decoder to dynamically focus on different parts of the input sequence, mitigating the information bottleneck.

Solution to Bottle neck problem

Attention Mechanism: Introduces dynamic focus, allowing the decoder to access relevant parts of the input sequence, improving information retention.

Transformer Models: Rely entirely on self-attention mechanisms, eliminating the fixed-size context vector and capturing long-range dependencies.

Hierarchical Models: Divide long sequences into smaller chunks, encode them separately, and combine their representations for a comprehensive context.

Increased Context Vector Size: Expanding the dimension of the context vector can help retain more information.

Problems with RNN

- They process the input data sequentially, one after the other. Such a recurrent process does not make use of modern graphics processing units (GPUs), which were designed for parallel computation and, thus, makes the training of such models quite slow.

- They become quite ineffective when elements are distant from one another. This is due to the fact that information is passed at each step and the longer the chain is, the more probable the information is lost along the chain.

The shift from Recurrent Neural Networks (RNNs) like LSTM to Transformers in NLP is driven by these two main problems and Transformers' ability to assess both of them by taking advantage of the Attention mechanism improvements:

- Pay attention to specific words, no matter how distant they are.
- Boost the performance speed.

Positional Encoding

Positional encoding is a critical component in transformer models, allowing them to handle sequential data by encoding the positions of tokens in a sequence. This enables transformers to capture the order and structure of the input data.

Purpose:Positional encoding provides information about the position of tokens in a sequence for transformer models, which process tokens in parallel.

Types: Includes sinusoidal positional encoding (using sine and cosine functions) and learnable positional encoding (with trainable parameters for each position).

Application: Positional encodings are added to token embeddings to give the model order awareness.

Benefit: Enables transformers to capture the order and structure of sequences without losing their parallel processing advantage.

Attention

The attention mechanism in transformers, specifically self-attention, allows each token in an input sequence to focus on other tokens by computing attention scores. These scores are derived from the dot product of query and key vectors, scaled and passed through a softmax function to produce attention weights. The final output for each token is a weighted sum of value vectors, enabling the model to capture dependencies and relationships between tokens, regardless of their position in the sequence.

Self Attention

Self-attention, also known as intra-attention, allows each token in an input sequence to attend to all other tokens, enabling the model to capture dependencies within the same

sequence. It computes attention scores for each token pair using their query and key vectors, scales the scores, and applies a softmax function to get attention weights. The output for each token is a weighted sum of the value vectors, incorporating information from the entire sequence.

Comparison

Simple (or traditional) attention in sequence-to-sequence models allows the decoder to focus on relevant parts of the encoder's output, computing attention weights between the decoder's current state and all encoder states. In contrast, self-attention processes a single sequence, enabling each token to attend to other tokens within the same sequence. The key difference is that simple attention links different sequences (encoder to decoder), while self-attention captures relationships within a single sequence.

Multihead attention

Multi-head attention is an extension of the self-attention mechanism used in transformer models to improve the model's ability to capture different aspects of relationships between tokens in a sequence. Instead of computing a single set of attention scores, multi-head attention divides the input into multiple parts (heads) and processes them independently. Each head computes its own set of attention scores and outputs, capturing various features and patterns.

lecture#36

Today’s Agenda

- Introduction to RAG
- Intro to langchain
- Document loading

Introduction to RAG

- Retrieval-Augmented Generation (RAG): Combines retrieval-based and generation-based methods.
- RAG Model: Retrieves relevant documents from a corpus and generates a response based on both the query and the retrieved documents.
- Applications: Improves the performance of tasks like question answering, summarization, and more.
- Advantages: Leverages large external knowledge bases, enhances contextual understanding, and provides more accurate and relevant answers.

Introduction to LangChain

- LangChain: A framework for building applications using language models.
- Components: Includes tools for document transformation, connection to various data sources, and chaining multiple operations.
- Features: Easy integration with various NLP models, flexible pipeline creation, and support for

various text processing tasks.

- Use Cases: Useful for building chatbots, automated text processing pipelines, and other NLP applications.

Document Loading

- Document Loading: The process of ingesting documents into a system for further processing.
- Supported Formats: Includes plain text, HTML, docs, PDF, and more.
- Techniques: Use of specific loaders for different formats, handling large documents, and maintaining document integrity.
- Applications: Preparing data for NLP tasks, ensuring compatibility with various text processing tools, and efficient handling of diverse document types.