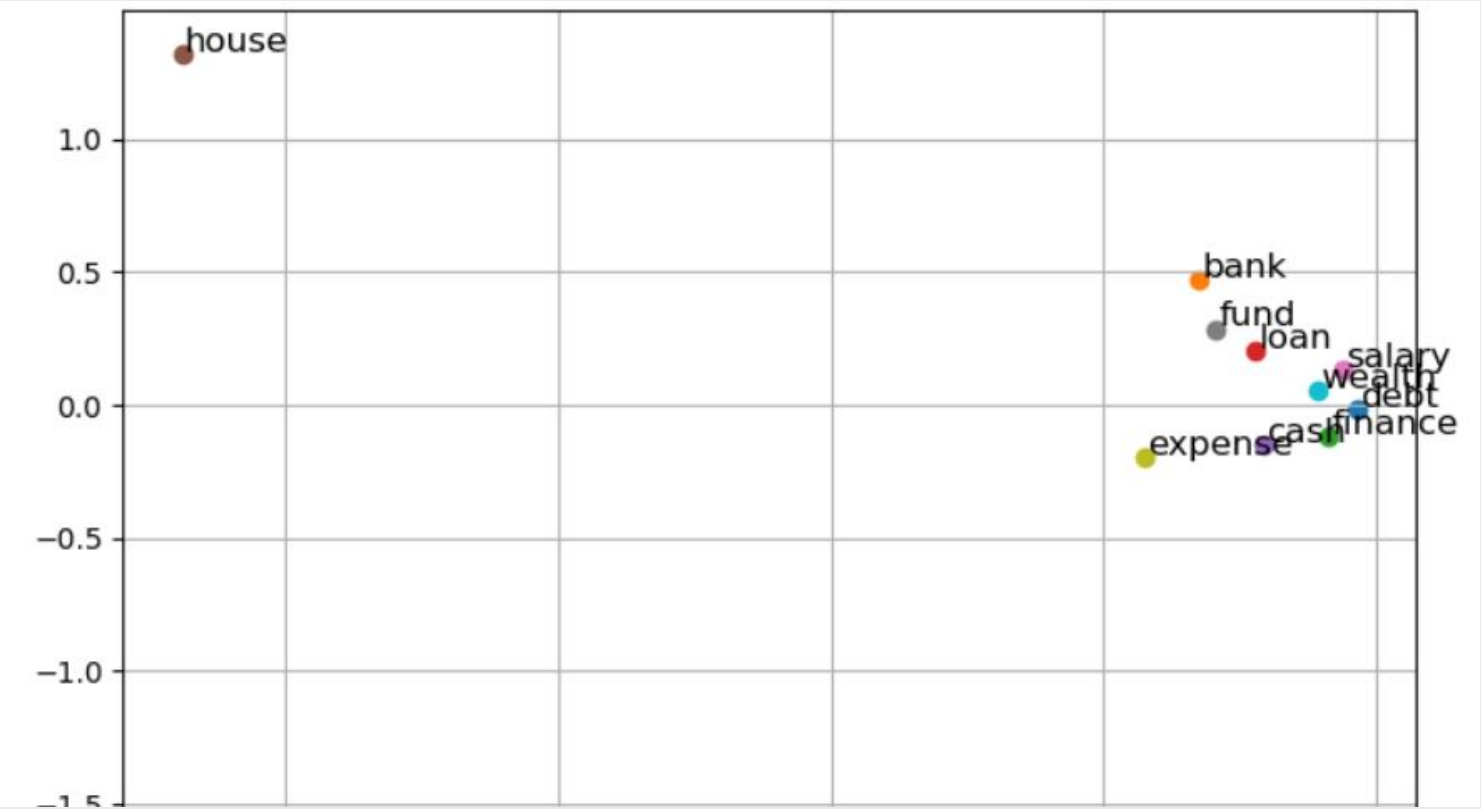
Explainable AI, Large Models, Neural Networks, and Beyond

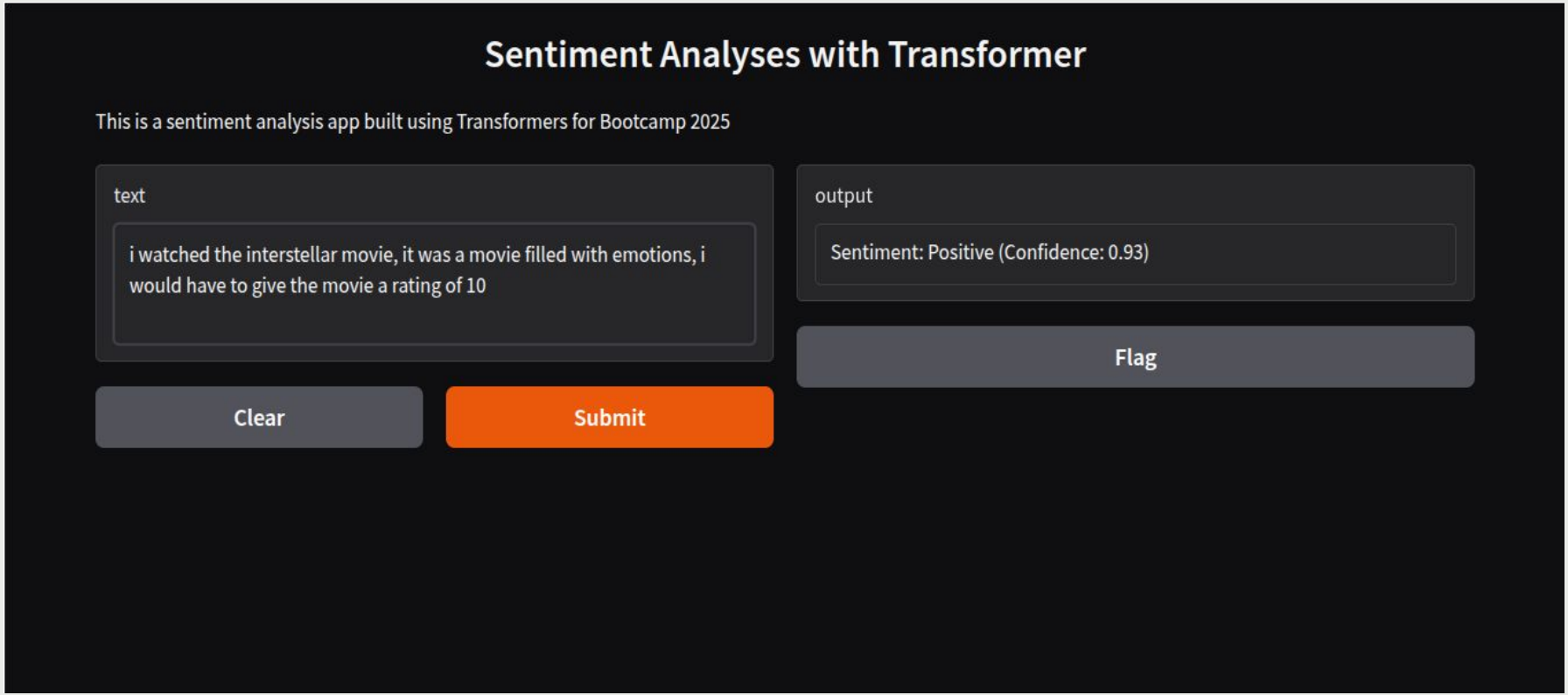
Davis Onyeoguzoro

Table of Contents

Apps built during lecture (1)

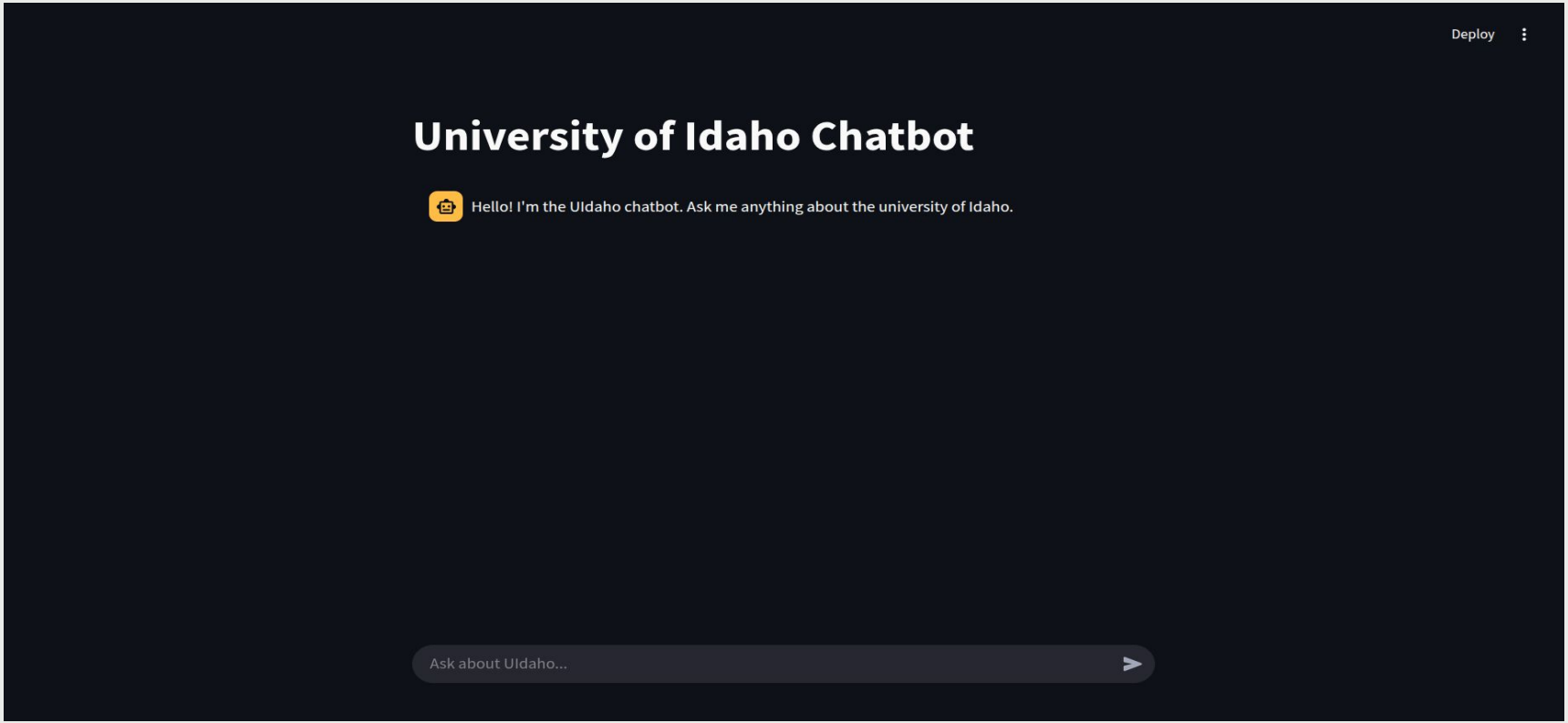


Apps built during lecture (2)

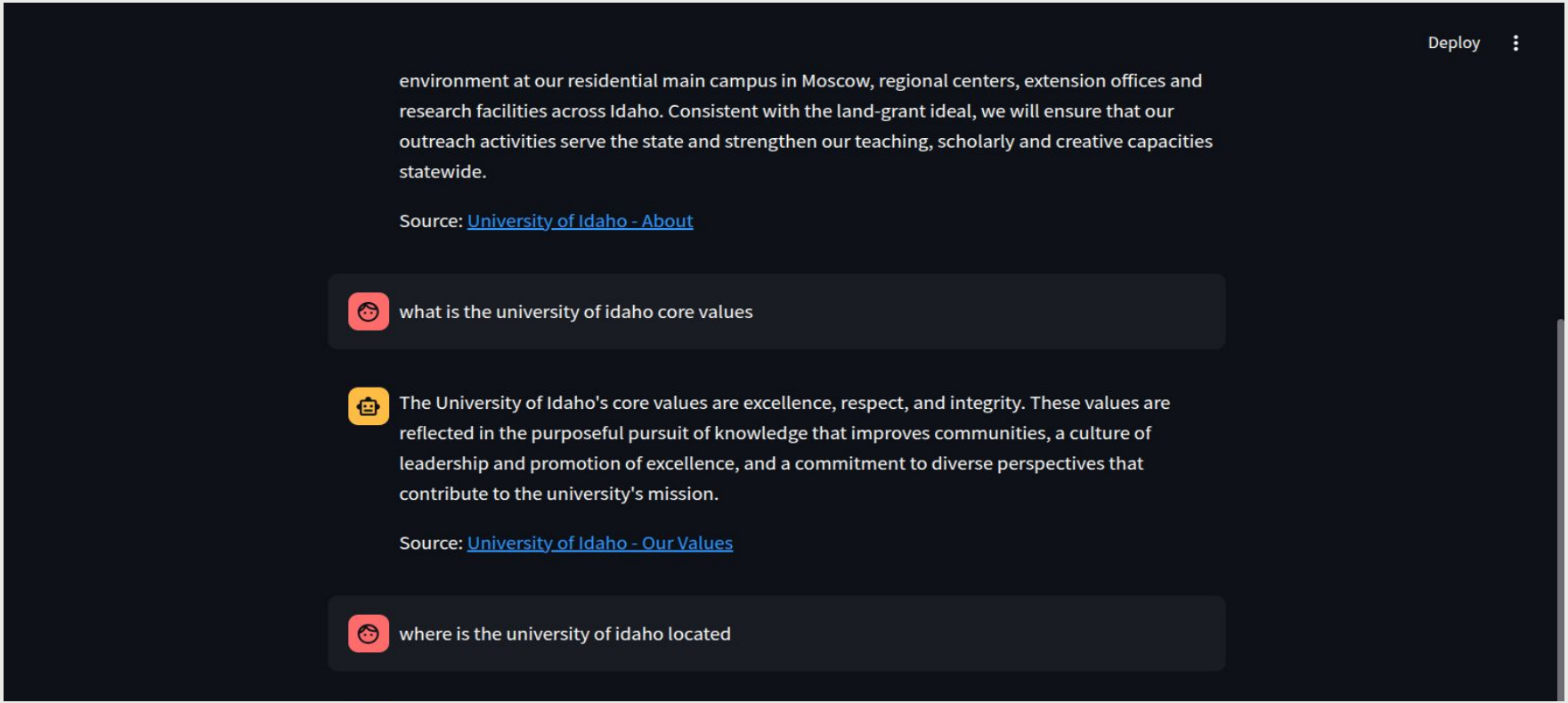


Apps built during lecture (3)

Apps built during lecture (3)



Apps built during lecture (3)



Fundamentals of AI

Understanding Artificial Intelligence

What is Artificial Intelligence?
Artificial Intelligence (AI) is a broad field of Computer Science that is focused on the creating of machines that can perform tasks that typically require human intelligence.

Basic tasks like

- Learning
- Reasoning
- Problem Solving
- Decision Making

AI enables computers to simulate human abilities.

History of AI

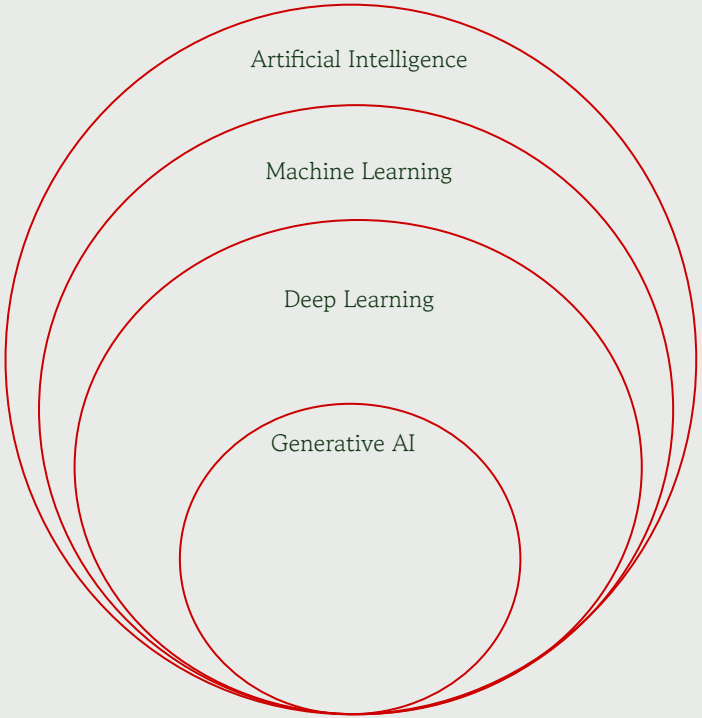
Artificial Intelligence has been around for a very long time, irrespective of the hype it has gained over the last decade.
Timeline of AI:

- 1950s: Alan Turing paper - Computing Machinery and Intelligence
- 1956: John McCarthy coined the term “AI” - Dartmouth College
- 1966: ELIZA - The first chatbot (simulating a therapist)
- 1980s: Development of Algorithms like Backpropagation
- 1997: IBM’s Deep Blue defeats chess champion
- 2006: Geoffrey Hinton popularized deep learning
- 2012: AlexNet revolutionizes image recognition
- 2016: Google DeepMind AlphaGo wins world champion in Go
- 2017: Transformers revolutionize NLP
- 2020s: AI becomes mainstream
- Future: Artificial General Intelligence (AGI) - AI with human-like reasoning

Subfields of AI

- Machine Learning - Algorithms learn patterns from data to make predictions without explicit programming
- Deep Learning - Using neural networks with multiple layers to model complex patterns. Secret sauce to rapid advancements in image recognition.
- Natural Language Processing - Enabling machines to understand, interpret, and generate human language. COmmon example is ChatGPT.
- Computer Vision - Extracting information from visual data (images/videos). Tasks like Object Detection, facial recognition.
- Robotics - Combines AI, Engineering, and Mechanical Engineering to design robots that perform physical tasks. Examples - Self driving cars, Industrial Robots
- Planning and Reasoning - Used for decision-making and problem-solving by simulating logic, search algorithms, and optimization.

Subfields of AI - Venn Diagram



Applications of AI

- Healthcare
  - Analyzing medical images (X-rays, MRIs) for early disease detection
  - Drug discovery, helps in accelerating research by predicting molecular interactions (e.g AlphaFold for protein folding)

- Agriculture
  - Crop Monitoring using Drones & AI to track crop health, pests, and soil conditions.
  - Precision Farming by optimizing irrigation, fertilization, and harvesting for higher yields.

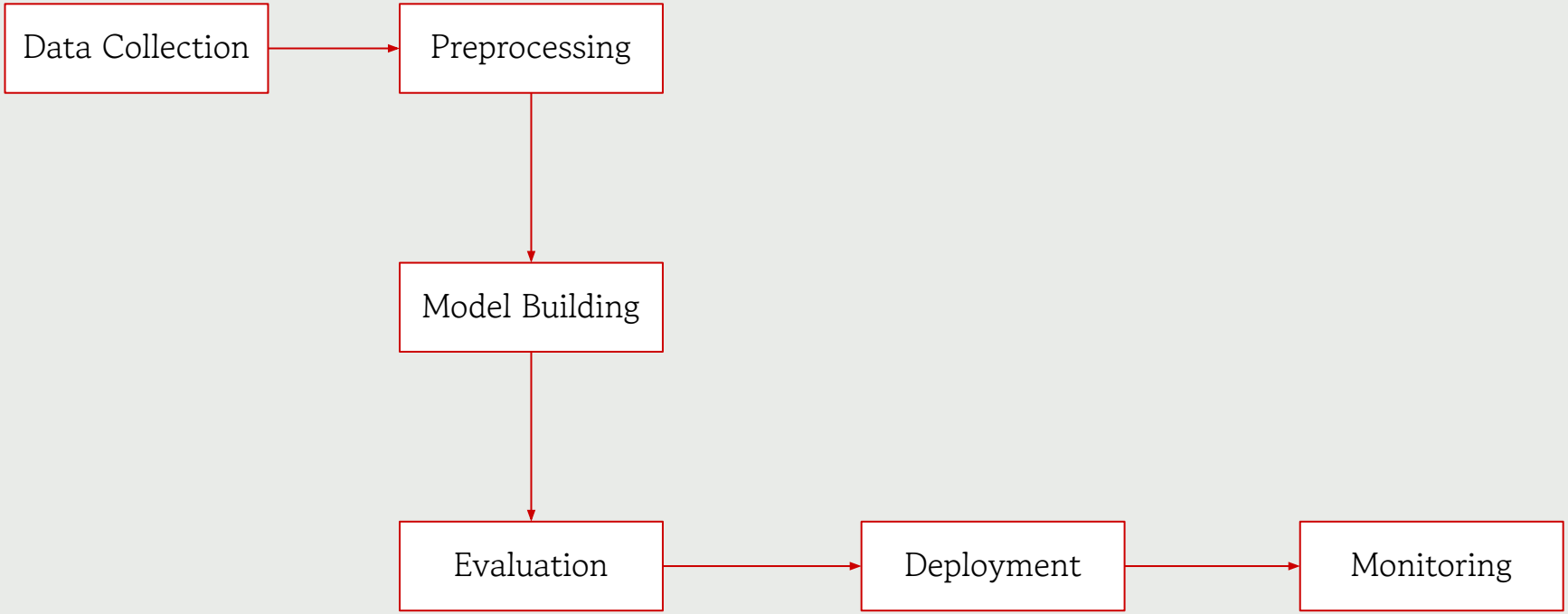
- Education
  - For tailoring lessons to individual student needs, AI studies individual user pattern and provides contents based on an individual level and not generalized content (Doulingo's personalized language learning).
  - Automated evaluation and grading of assignments with feedback (Turnitin's plagiarism detection).

Applications of AI (cont.)

- Finance
  - Fraud Detection for spotting unusual transatcions in real-time (Paypal's fraud detection).
  - Algorithmic Trading where AI-driven bots analyze markets and execute trades at high speed.

- Transportation
  - Self-Driving Cars - AI processes sensor data for autonomous navigation (Waymo - Google former self-driving car project).
  - Traffic Management - AI optimized traffic signals and reduces congestion (Smart city initiatives).

AI Pipelines



Neural Networks

What is a Neural Network?
These are machine learning models that is inspired by the human brain, and consist of interconnected nodes (neurons) that process data and learn processes.

Key Features of a Neural Network

1. Inspired by the Human Brain
    a. NNs mimic how our biological neurons process information
    b. Learn from daa by adjusting connections.
2. Structure: Layers of interconnected “Neurons”
    a. Input Layer: Receives the raw data
    b. Hidden Layers: Process data through weighted connections (where features are extracted from the data)
    c. Output Layer: Produces the final prediction (e.g Cat vs Dog, Translation)
3. How it Learns
    a. Forward Propagation: Data flows through the network to generate predictions.
    b. Backpropagation: Adjusts connection weights based on errors to improve accuracy, using optimization techniques like Gradient Descent.

Types of Neural Networks

1. Feedforward Neural Networks (FNNs)
    a. Data flows in one direction (Input -> Hidden -> Output)
    b. Use Case: Simple Classification Tasks (Spam Detection)

2. Convolutional Neural Networks (CNNs)
    a. Specialized for Image/Video Processing
    b. Layers
        i. Convolutional Layers: Detect patterns in images (edges) using filters.
        ii. Pooling Layers: Reduce dimensionality using Max Pooling.
    c. Use Case: Facial Recognition, Medical Imaging.

3. Recurrent Neural Networks (RNNs)
    a. Specialized for Sequential Data (Text, Time Series, Speech)
    b. Memory is located in the Hidden state that is used to retain past information.
    c. But it struggles with long term memory (analogy of old age)
    d. Solved by LSTM using 3 gates, input, forget, output gates.
    e. Use Case: Speech Recognition, Stock Price Prediction.

Anatomy of an Artificial Neuron

1. Inputs (X1, X2, ..., Xn)
   a. Raw data features gotten from the data, e.g Pixel values, sensor readings, word embeddings.

2. Weights (w1, w2, ..., wn)
   a. Learnable parameters that determine the importance of each input.
   b. This is not fixed, as it is constantly being adjusted during training to minimize errors (via Gradient Descent).

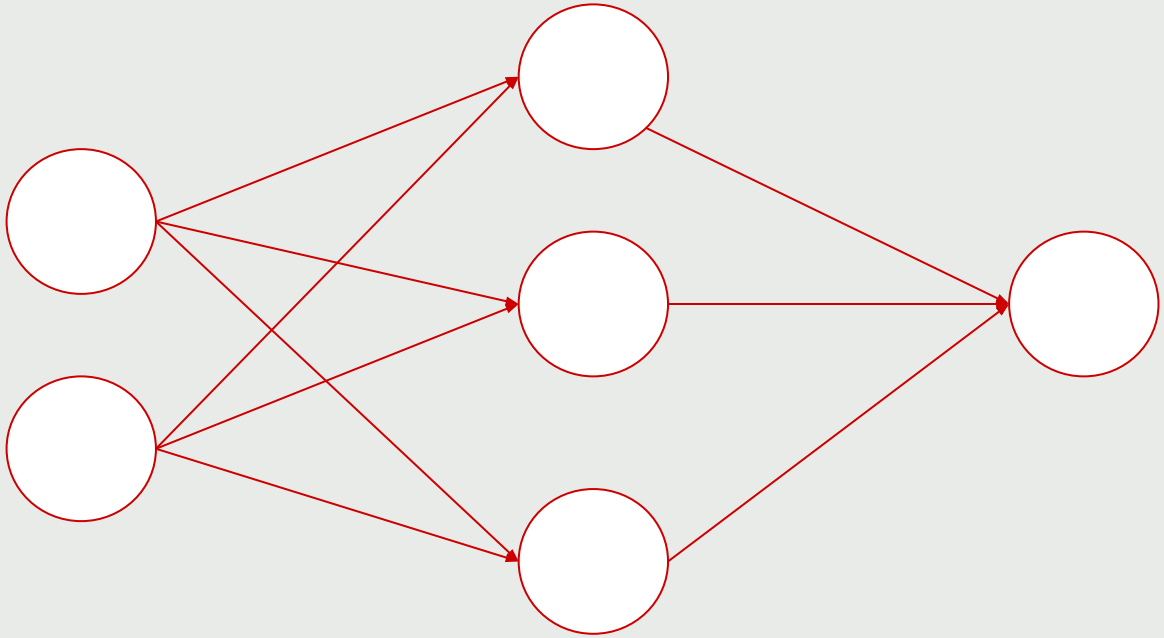
3. Bias (b)
   a. An offset term that shifts the output, it is usually a small value to prevent the neurons from being overly saturated.
   b. Introduces flexibility to the network.

4. Activation Function
   a. Introduces non-linearity, so that the network can learn complex patterns.
   b. Types - Sigmoid, ReLU, Softmax

Layers in a Neural Network

1. Input Layer:
    a. Receives raw data, just feeds the data into the network.
    b. One neuron per input feature
2. Hidden Layer:
    a. Receives raw data, just feeds the data into the network.
    b. One neuron per input feature
3. Output Layer:
    a. Receives raw data, just feeds the data into the network.
    b. One neuron per input feature

Layers in a Neural Network



Activation Functions

What is an Activation Function?
Activation Functions in neural networks are mathematical equations that determine the output of a neuron, makes decisions whether that particular neuron should be “activated” or not.

They introduce non-linearity in a network, which helps the model to learn complex patterns.

Why use Activation Functions?

1. Without activation functions, a neural network would just be a linear regression model, no matter how many layers it has.
2. Activation functions allow networks to learn non-linear relationships, like learning from curves and decision boundaries.

Types of Activation Functions

1. Sigmoid
    a. Range: [0,1]
    b. Use Case: Binary Classification where it outputs probability.
    c. Suffers from Vanishing Gradients, slow learning
2. ReLU (Rectified Linear Unit)
    a. Range: [0, ∞]
    b. Use Case: CNNs, Transformers
    c. Neurons can get stuck at zero.
3. Leaky ReLU
    a. Range: [-∞, ∞]
    b. Use Case: Allows some negative outputs to fix the ReLU problem.
4. Softmax
    a. Range: [0, 1]
    b. Use Case: Multi-class classification, M-NIST digit recognition.
5. Tanh
    a. Range: [-1, 1]
    b. Use Case: Stronger gradients than sigmoid.

Forward Propagation

Forward Propagation is all about prediction. It is the process of passing input data through a neural networks layers to generate a prediction (output).

Goal: Compute the neural network output from input data.

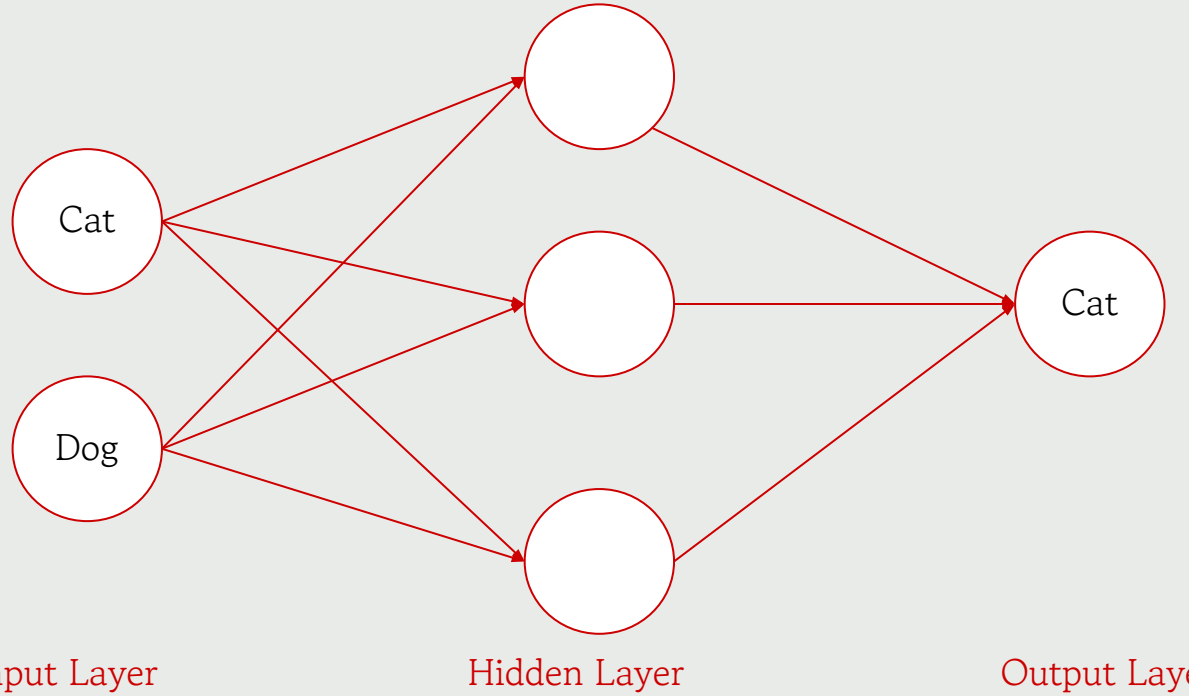
Process:

- Input data is passed through each layer.
- Each neuron calculates the weighted sum and the activation
- Final Output (Prediction) is compared to the True Label using Loss Function (more on this in future slides).

Example:

For an image classification of Cats vs Dogs, having gone through the entire AI Pipeline (as discussed earlier), we pass the data to the input layer, then the hidden layer, the forward pass generates probabilities like [0.1, 0.9] for dog vs cat (Softmax).

Forward Propagation (fig)



Backpropagation

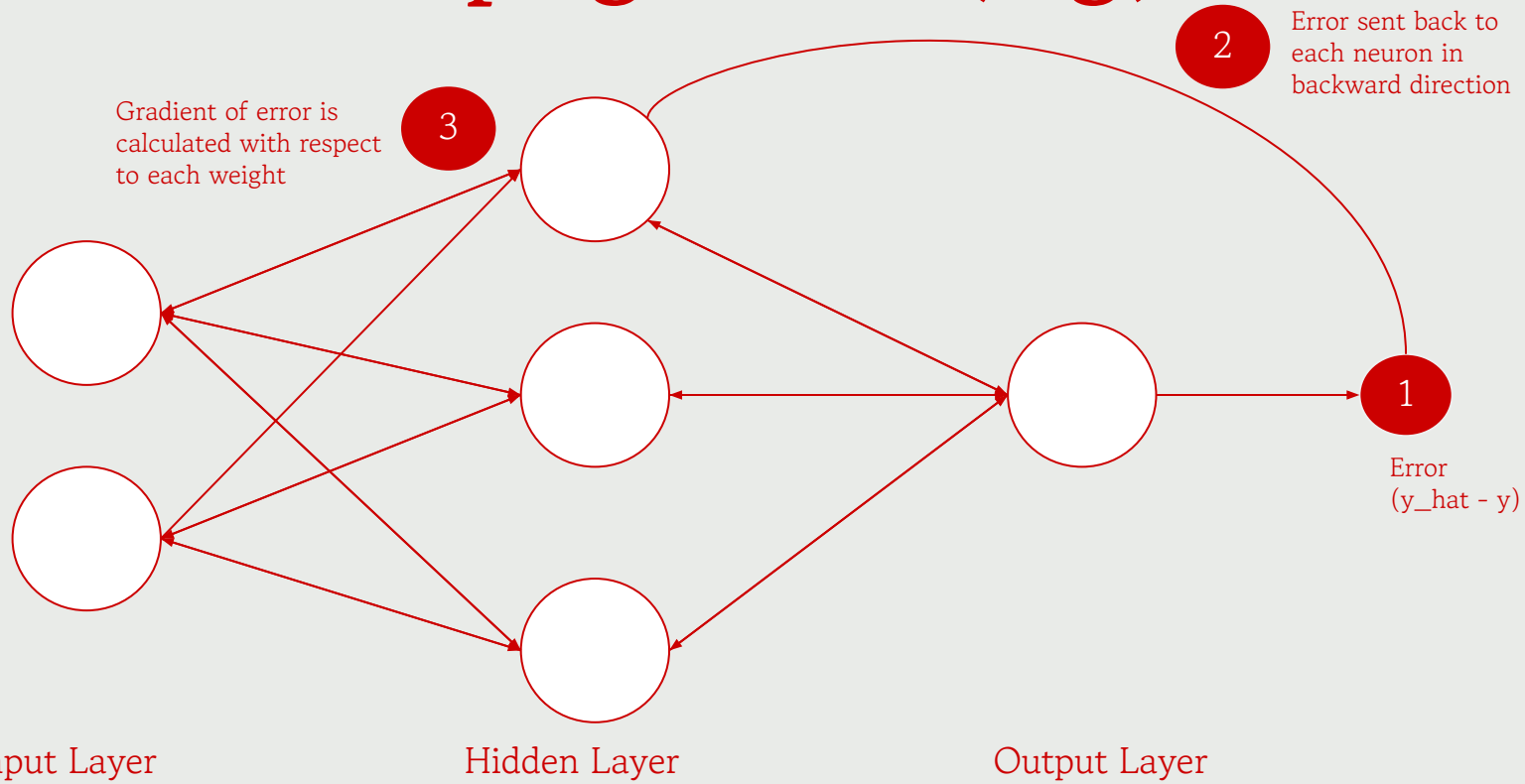
Backpropagation is all about learning from experience. It is the process of optimizing the weights within the network by minimizing the difference between the predicted outputs and actual outputs.

Goal: Compute gradients of the loss with respect to each weight/bias in other to minimize error.

Process:

- Calculates the gradient loss with respect to each output.
- Recursively compute gradients for each layer, moving backward.
- Updates the weights to reduce the loss.

Backward Propagation (fig)



Loss Functions

It quantifies the difference between a model's predictions and the actual values, serving as a measure of the model's error.
It basically measures the error in prediction.

Examples of Loss Functions

1. Mean Squared Error (MSE)
    - Purpose: Used for regression tasks (predicting continuous values).
    - Sensitive to outliers.
2. Cross-Entropy Loss
    - Purpose: Used for classification tasks (discrete categories).
    - Works with probability outputs (Sigmoid/Softmax).
3. Hinge Loss
    - Purpose: Used for SVMs and some neural networks.
    - Robust to outliers.

Overfitting & Underfitting

Overfitting

- This occurs when a model learns the training data too well, including irrelevant details and noise, which results to poor performance on new, unseen data.
- High accuracy on training data, poor performance on test data.
- Solution: Regularization (L1/L2) which adds penalty to large weights, ensuring sparsity and weight distribution, Dropout which randomly disables neurons during training.

Underfitting

- This occurs when a model fails to capture the underlying patterns in a training data, leading to poor performance on both the training and test data.
- Poor performance on both training and test data.
- Solution: Add more layers and/or neurons, train for longer by increasing epochs.

Hyperparameters

These are parameters that are set before the training process of the model begins and it control's the models learning process based on the configuration of this hyperparameters.

Some examples of hyperparameters are:

- Learning Rate: Is a step size for the weight updates. It controls how much the model weights should be adjusted during each iteration.
- Epochs: This is the one complete pass through the entire training dataset during one cycle of model training
- Optimizers: These are algorithms that are used to minimize the loss function during model training by adjusting parameters like the weights and biases.
- Batch size: This is the number of training samples a model processes before updating its internal parameters like the weights and biases.

Advance Neural Architectures

Convolutional Neural Network: Is a type of artificial neural network that is used to process image/video data.
Key Features:

- Convolutional Layers: These layers are used to extract features from the input data by applying a small filter on the image and performing element-wise multiplication.
- Pooling Layers: These layers are used to reduce the spatial dimensions of the feature maps, while preserving the most important information.
- Use Cases: Image Classification, Object Detection, Image Segmentation.

Recurrent Neural Network: Is a type of neural network that is used to process sequential data, that means the order of elements is important.
Key Features:

- Memory: It has a hidden state (h_t) that retains information from previous steps.

Long Short Term Memory: Is a type of RNN that is used to address the vanishing gradient probem of RNN through the use of forget gate, input gate, and output gate.

Transformers

What is Transformers?
Transformers is a type of neural network architecture that excels at understanding and processing sequential data, including text, audio, and images.
They achieve this by leveraging a mechanism called “Self Attention”, which allows the model weigh the importance of different parts of the input sequence in relation to each other.
Transformers was introduced in 2017 in the “Attention is All You Need” research paper

Self-Attention Mechanism: The self attention mechanism allows models to focus on specific parts of an input data.
Each word in a sequence is mappred to three vectors:
Query (Q): What am I looking for?
Key (K): What can i offer?
Value (V): Actual information to propagate

Transformers (cont.)

Multi-Head Attention: This layer extends the self-attention by allowing the model to attend to multiple representation in parallel.
It works by splitting the query, key, and value vectors into multiple smaller vectors, or heads, and applying the key feature of the self-attention mechanism to each head separately.

Feed-Forward Neural Network: It processes the output sequence length out of the attention vectors and generates the final output sequence.

Positional Encoding: They provide the model with information about the order of words in a sequence.

A common advantage of using transformers is that it processes in sequence, that way you can use multiple gpus for your training, but the issue is that if the order of the data is important, this parallelism would result in the data being scattered all over the place, thereby losing crucial information. That is why positional encoders are important.

She gave the book to the child ≠ The child gave the book to her
Same words, different structure, different meaning

Transformers (simplified)

Self Attention basically allows the mode to pay attention to different parts of the input when producing each word in the output. It answers:
“What other word(s) in this sentence are important for understanding this current word?

Example:
“The animal didn’t cross the road because it was too tired.”

To understand what “it” refers to, the model must pay attention to “animal” - not road.
Self-attention helps it to do that.
How it works:

- Each word is turned into 3 vectors:
  - Query (Q): What am I looking for?
  - Key (K): What do I offer?
  - Value (V): What is my meaning?
- After that, the model then compares each word’s query to all other keys, calculates attention scores, and uses those to blend the values.

Transformers (simplified)

Self Attention example 2

Sentence:
“The cat sat on the mat.”
If the model is processing the word “sat”.

Q = “sat”. What context do I need to understand this word?

It attends to:
“Cat” (subject)
“Mat” (location)

The representation of “sat” is influenced most by “cat” and “mat”, and less by “the” or “on”.

Transformers (simplified)

Multi-Head Attention
Instead of using just one attention mechanism, transformers use multiple attention heads at the same time. Each head focuses on a different kind of relationship in the input sentence. This lets the model capture more context from multiple angles.

Exercise
“The boy kicked the ball.”

- What is the subject-action head? (What is the person/thing doing the action?)
  - What 2 words would you connect?
- What is the object head? (What is the action being done?)
  - What do you focus on?
- What is the modifier head? (What is receiving the action?)
  - What might you attend to?

Transformers (simplified)

Multi-Head Attention
Instead of using just one attention mechanism, transformers use multiple attention heads at the same time. Each head focuses on a different kind of relationship in the input sentence. This lets the model capture more context from multiple angles.

Exercise
“The boy kicked the ball.”

- What is the subject-action head? (What is the person/thing doing the action?)
  - boy
- What is the object head? (What is the action being done?)
  - kicked
- What is the modifier head? (What is receiving the action?)
  - the ball
- Modifiers - the, red

Transformers (simplified)

Positional Encoding
Transformers do not have a built-in sense of order like RNNs, so we add positional information to the input embeddings to help the model know the order of words.

Example: “I love AI”
Without Positional Encoding, the model sees:
“I” - [0.8, 0.1]
“Love” - [0.4, 0.9]
“AI” - [0.6, 0.2]
But it does not know if “I” comes first or last.

Word	Embedding	Positional Vector	Final Output
I	[0.8, 0.1]	[0.1, 0.0]	[0.9, 0.1]
love	[0.4, 0.9]	[0.2, 0.0]	[0.6, 0.9]
AI	[0.6, 0.2]	[0.3, 0.0]	[0.9, 0.2]

Transformers (simplified)

The Transformer Architecture

Transformer = Encoder + Decoder

It is like a 2 part machine:
Encoder: Understands the input
Decoder: Generates the output

It works together like this:
“You speak to it -> It understands -> Then it speaks back”

Transformers (simplified)

ENCODER: Understanding the input

1. Input words:
   - “The cat sat.”
2. Word embeddings:
   - “cat” - [0.8, 0.1, 0.5]
3. Add Positional Encodings
4. Self-Attention Layers:
   - “sat” attends to “cat” to know who sat
5. Feed-Forward Network: Extract deeper meaning.
6. Output: Contextualized word vectors:
   - “Cat” = animal + subject
   - “Sat” = action + who did it

Transformers (simplified)

DECODER: Generating the Output

1. Starts with a special token
    - <START>

2. Looks at Encoder's Output
    - What did the encoder understand?

3. Uses Self-Attention to look at the words it has generated so far

4. Generates Next Word
    - Predicts one word at a time:
        i. The
        ii. The dog
        iii. The dog barked

5. Repeats until End Token
    - <END>

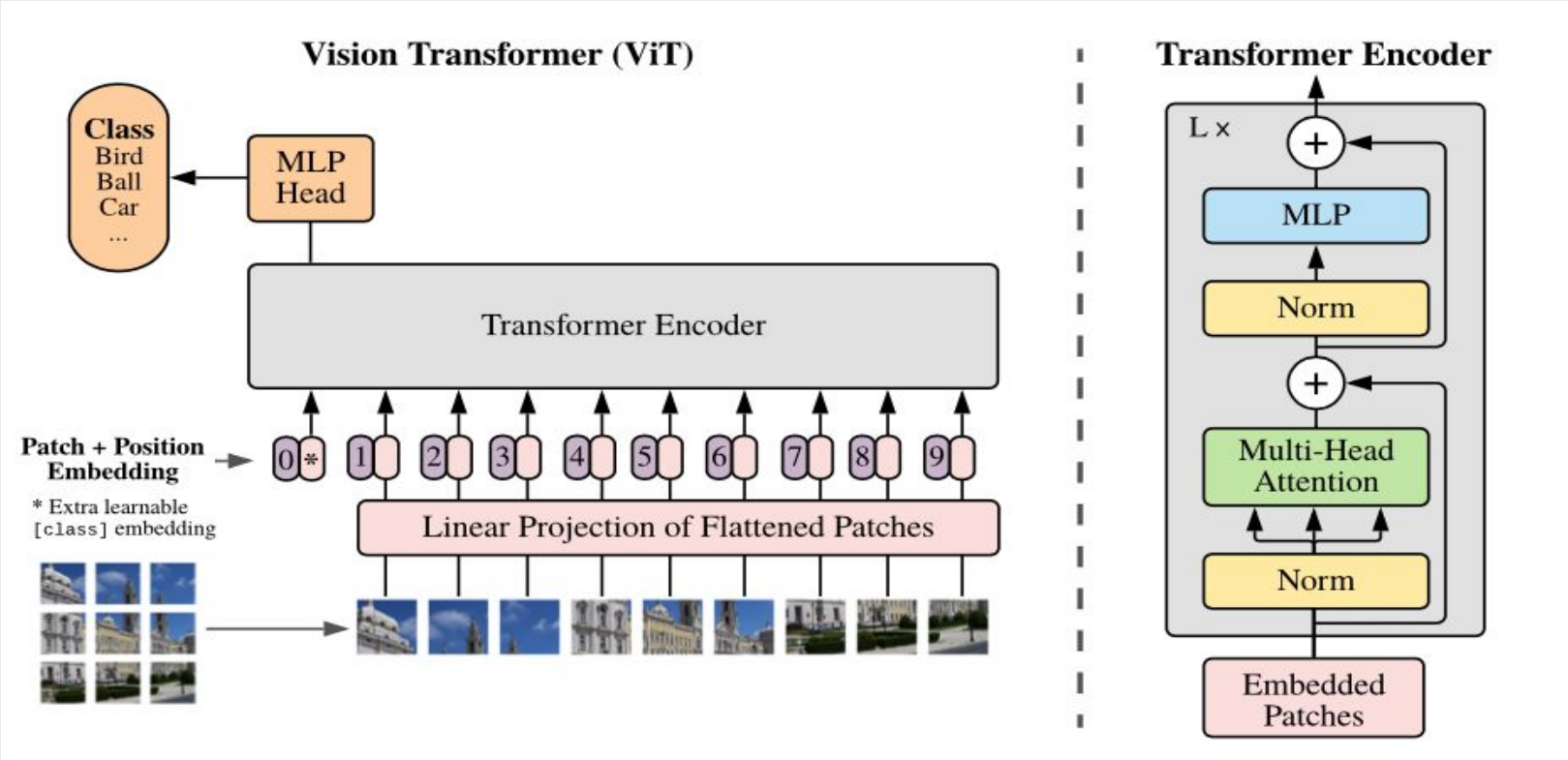
Transformers in Vision (History)

Date	Model	Description	Vision Transformer
2017 June	Transformer	Model solely based on Attention-mechanism. Excellent performance in NLP	❌
2020 May	DETR	Makes a prediction on an image on a single pass	✅
2020 Oct	ViT	Pure transformer architecture for visual recognition	✅
2021 - Today	ViT Variants	Several ViT variants DeiT, TNT, Swin, CSwin (2022)	✅

Transformers in Vision

Source: Google Research

Transformers in Vision



Source: Google Research

Transformers to LLMs

Large Language Models (LLMs)

LLMs are basically huge transformers trained on massive amounts of data. They can understand, complete, and generate human-like text.

LLM Type	Transformer Part Used	Example of Models	What it does
Encoder-Only	Encoder	BERT, RoBERTa	Understands text Good for classification and QnA
Decoder-only	Decoder	GPT, GPT-2, GPT-4	Generates text Good for chatbots and story writing
Encoder-Decoder	Both	T5, BART, mT5	Translates, Summarizes, Rephrases

LLMs

What makes LLMs “Large”?

The “L” in LLM stands for Large, because:

- They are trained on billions or trillions of words
- They have millions to billions of parameters
- They learn world knowledge, grammar, facts and reasoning

Example:

- GPT-4 by OpenAI has over 1 trillion parameters
- BERT by Google has about 340 million parameters
- T5 by Google has about 11 billion
- LLAMA 2 by Meta has 3 variations, 7 billion, 13 billion, 70 billion
- Claude by Anthropic did not disclose, but various sources say 175 billion (not verified).
- Google Gemini Ultra by Google uses 1.5 trillion parameters

LLMs

What are these millions - trillions of Parameters?
Parameters are like the knobs or switches inside a neural network that the model learns and adjusts during training.

They are like our brain's memory
The way our brain strengthens certain connections when we learn something new (like you are doing now), AI models strengthen or weaken parameters to "learn" from data.

Each parameters is usually a number (weight or bias) in the model. Together they form matrices that:

- Connect layers
- Transform word embeddings
- Compute attention scores
- Decide outputs

The more parameters usually equate to the more knowledge the model can store.

Should we trust LLMs?

Many of us has at one point interacted with an LLM, perhaps ChatGPT or others, and we have realized how powerful they are, they can be a companion, write code, answer questions, generate text, and many more.

But, Do we really understand how these LLMs make these decisions?

If an LLM (like ChatGPT for instance) gives you legal advice, or diagnoses a medical condition based on the symptoms you gave it, and maybe for some reason, the answers were pretty accurate.

Would you not want to know why it gave that answer?

For instance, I played a game with several LLMs, asking them about their opinions on the Heinz Dilemma.

The Black Box Problem

LLMs and Deep Learning models are often black boxes. They give us an answer, but we can't easily see how or why they arrive at such conclusion.

For instance;

- Why did the LLM give that particular response to the Heinz Dilemma
- Why did it diagnose or predict this illness?
- Why did it classify this resume as "not qualified" and the other as "qualified"?
- Why is one user seeing harmful content and another isn't?

The only way to have a better understanding of the responses we receive from the various LLMs we interact with is by opening the Black Box.

What is XAI?

Opening the Black Box - XAI

Explainable AI (XAI)

Explainable AI is about making AI models more transparent, helping humans understand, trust, and sometimes challenge what a model is doing.

Explainability is important especially for large language models like GPT, BERT, and others, because it is crucial for ethics, fairness, debugging, and accountability.

Explainable AI seeks to make the responses of this LLMs interpretable and trustworthy, especially in high-stakes fields like medicine, law, or finance.

- Trust and Transparency: Users must trust model decisions, especially in critical applications like cancer diagnosis or credit approval.
- Debugging and Development: Understanding why a model made a mistake, can help us improve it.
- Legal and Ethical Compliance: Regulations like GDPR require explanations of automated decisions.

Explainability Techniques

Several techniques help demystify complex AI models. These range from visualizations to approximating model behavior with simpler models.

- LIME (Local Interpretable Model-agnostic Explanations: This approximates predictions locally using a simple model like a decision tree to explain individual predictions.
- SHAP (SHapley Additive exPlanations): This uses a game theory to assign contribution scores to features for a prediction. It works on any model.
- Saliency Maps (for CNNs): Visual highlights of which parts of an image most influences the output.
- Attention Maps (for Transformers): Shows what tokens the model focused on the most when making predictions.

Bias & Fairness in AI

AI systems can reflect and even amplify societal biases if trained on biased data. Fairness in AI is essential to prevent discrimination and ensure inclusivity.

Sources of Bias:

- Data Bias: Historical inequalities in datasets.
- Labeling bias: Annotators personal perspectives.
- Representation bias: Underrepresentation of minority groups.

Example:
A hiring algorithm rejecting resumes from certain demographics due to biased in training data.

How do we solve this?

Tackling AI Bias

Combating bias in AI requires at every stage: data collection, training, and deployment.

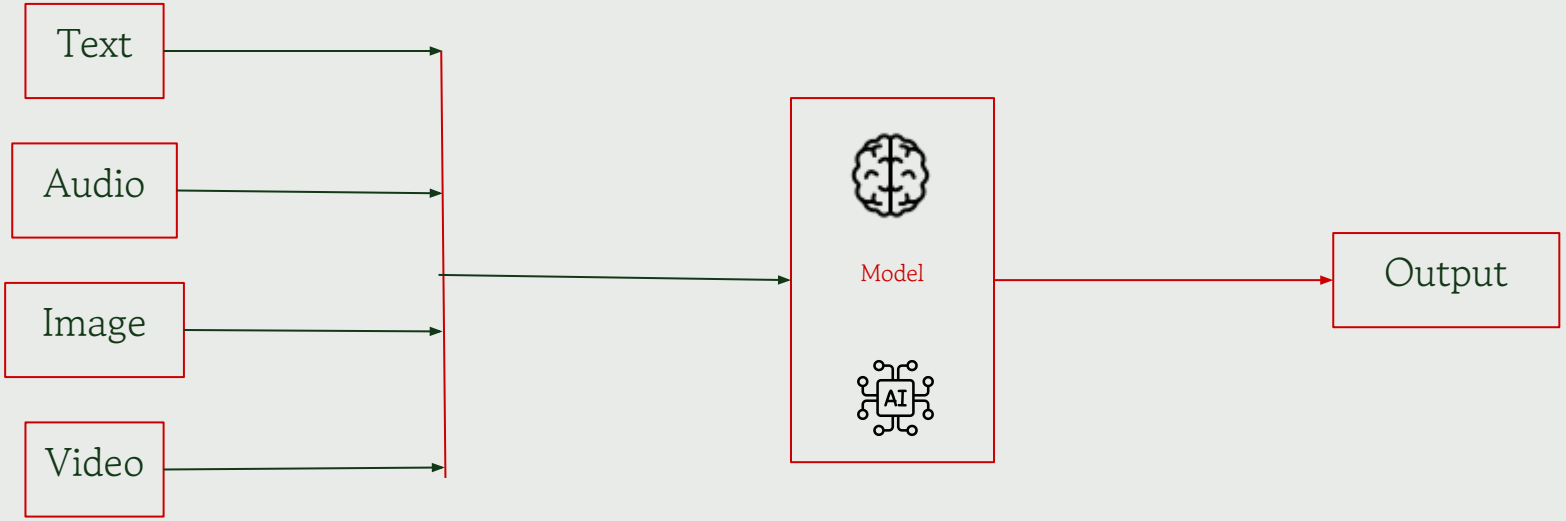
- Preprocessing: Balance in datasets, remove offensive terms, augment underrepresented classes.
- In-processing: Add fairness constraints during training (e.g equal opportunity loss).
- Post-processing: Adjust predictions ro remove discriminatory patterns, like calibrating thresholds for different groups.
- Regular Audits: Routinely evaluate model performance across diverse subpopulations.

Multimodal? Unimodal

What is Multimodal AI?
To understand Multimodal AI, we need to first understand Unimodal AI

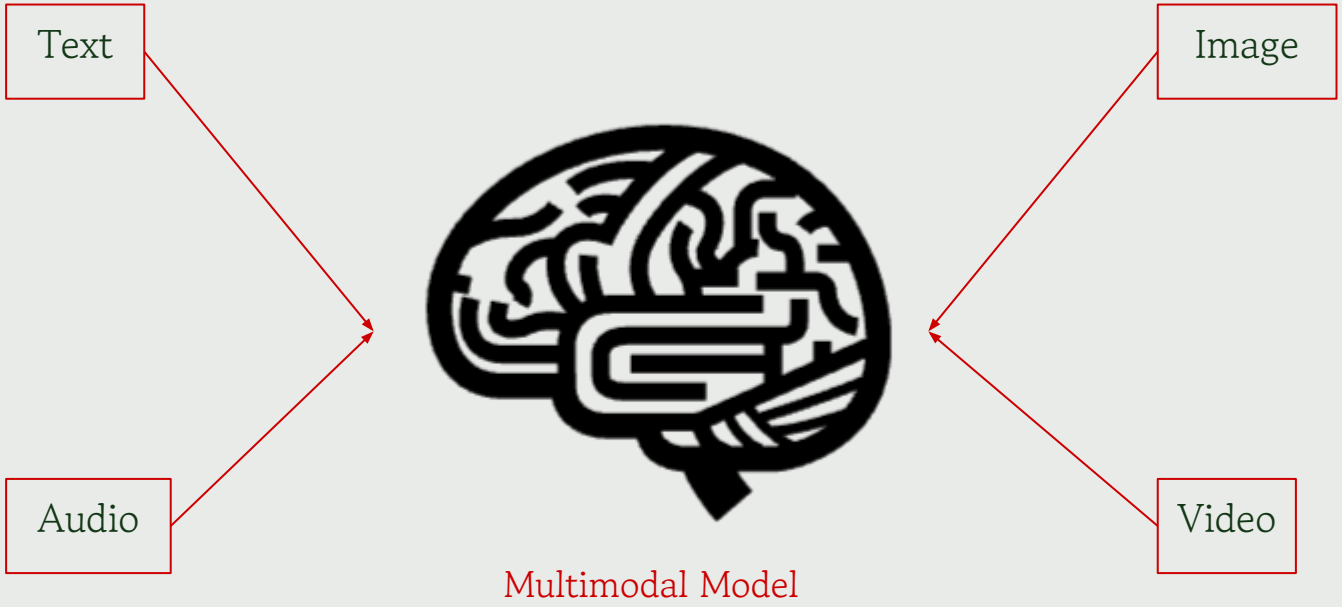
Unimodal AI

This refers to AI systems that are designed to analyze and process one type of data. The single type of source is either a text, image, or audio.

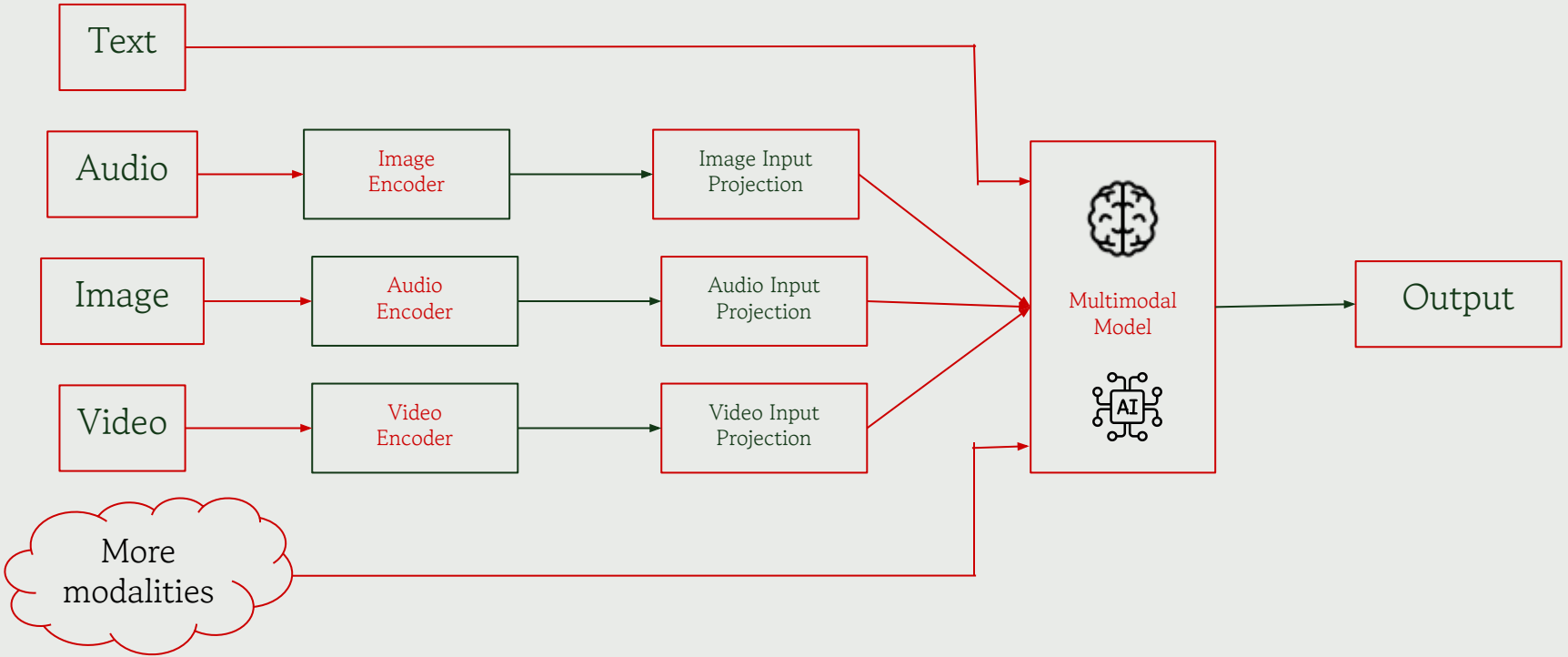


Multimodal AI

This refers to AI systems that can process and understand multiple types of data simultaneously, such as text, images, audio, and video. These systems mimic how humans learn from our various senses.



Multimodal AI (fig)



Applications of Multimodal AI

- Text: Text Generation, Text Summarization, Search Engines, Text Editing, Support (Chatbots, SMA), Note-taking, Translation, Plagiarism Detection, Auto-Correction, Sentiment Analysis, Named Entity Recognition, Text Simplification.
- Code: Code generation, Debugging, Documentation, Automated Code Completion, Code Review, Code Optimization, Refactoring.
- Image: Image Creation, Image Classification, Object Detection, Image Segmentation, Image Restoration, Image Forensics, Artistic Style Transfer.
- Speech: Text to Speech, Speech to Text, Voice Recognition, Voice Search, Natural Language Interaction, Voice Assistant, Speech Emotion Recognition.
- Video: Video Generation, Editing, Text to Video, Video Classification, Object Tracking, Video Retrieval, Video Quality Enhancement, Video Captioning.
- 3D: 3D Modelling, 3D Object Detection, 3D Animation, 3D Reconstruction, 3D Model Optimization, 3D Physics Simulation.
- Other: Robotic Process Automation, Gaming, Data Analysis, Material Science, Recommendation Systems, Creative Content Generation, Music Composition.

Benefits of Multimodal AI

- Contextual Comprehension: Multimodal models can process and understand data from multiple sources, which provides a more comprehensive and contextually relevant understanding of the information.
- Natural Interaction: They allow for more natural and intuitive interactions. They can understand and generate responses in various formats, such as text, images, audio, and video, which makes them more user-friendly.
- Accuracy Enhancement: The ability to process information from multiple sources, helps increase the accuracy of predictions and outputs. For instance, in medical diagnoses application, a multimodal model could analyze both medical images as well as patient notes to make a more accurate diagnoses.
- Capability Enhancement: They can perform a wider range of tasks compared to unimodal models. They can generate different types of data such as text, images, and code, which makes them more versatile.

Embodied AI

Embodied AI refers to intelligent agents that interact physically with the world, like household robots or autonomous drones.
Unlike the Multimodal AI (like we discussed previously), these systems bridge the digital and physical worlds through sensors, actuators, and adaptive decision-making.
These systems combine perception, action, and planning.

- Perception - Seeing & Sensing
  - Sensors: Cameras, LIDAR, radar, tactile sensors, microphones
  - Tasks: Object recognition, Spatial mapping, Force freedback.
- Planning - Thinking & Deciding
  - Algorithms: Reinforcement learning, symbolic reasoning.
  - Challenges: Real-time adaptation to dynamic environments (e.g crowded rooms).
- Action - Moving & Interacting
  - Actuators: Motors, Grippers, Propellers
  - Precision: From delicate surgery robots (like the Da Vinci 5 Robotic Surgical System) to heavy-duty industrial arms.

AGI - The Future???

Artificial General Intelligence (AGI)
AGI refers to a machine with broad, human-like cognitive abilities that is capable of learning, reasoning, and adapting to novel tasks across diverse domains without specialized training. Unlike narrow AI (like ChatGPT for texts, and others), AGI would possess general problem-solving intelligences.

Key Capabilities of AGI

- Transfer Learning Across Domains
- Abstract Reasoning & Logic
- Self-Awareness & Goal Setting

Demo & Live Coding

Link to Code: https://github.com/Davisonyeas/AI_Presentation

https://tinyurl.com/mp5zbun7