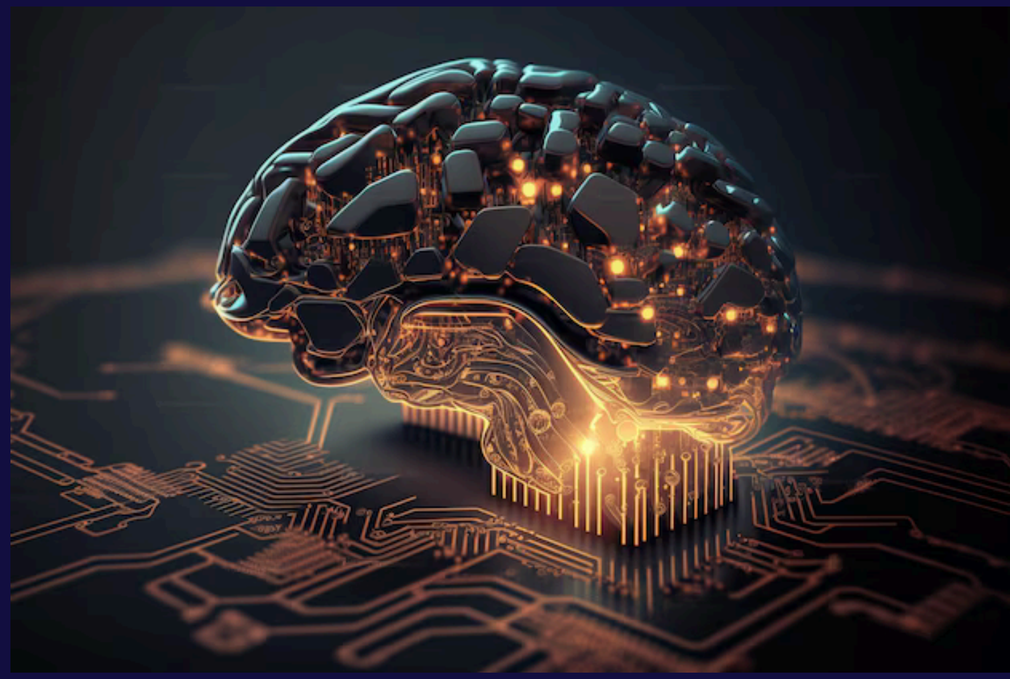
Large Language Models

Presented by: Zeynep Sude İleri

What is LLMs?

- Neural Networks
- Natural Language Processing

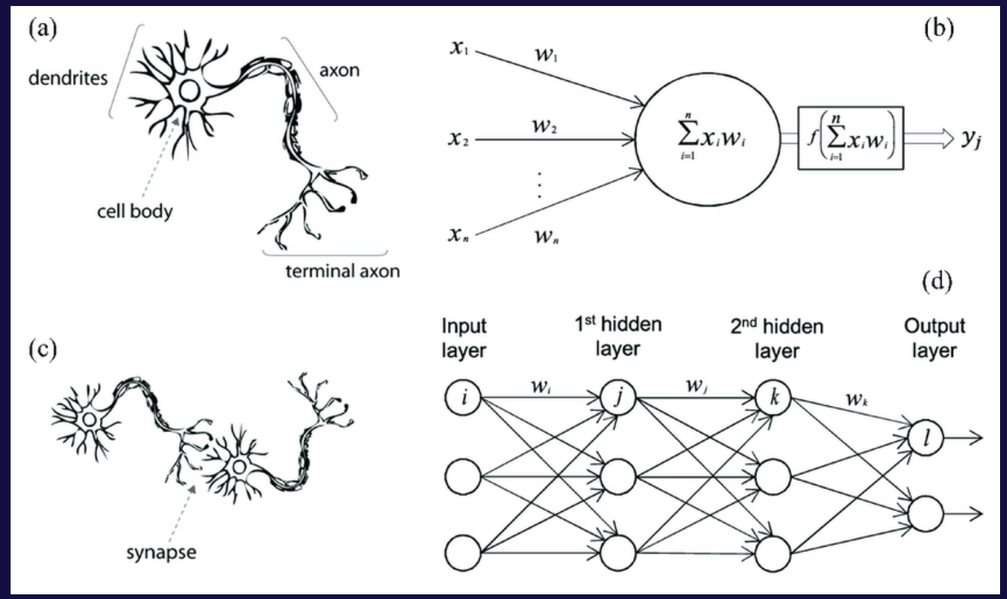


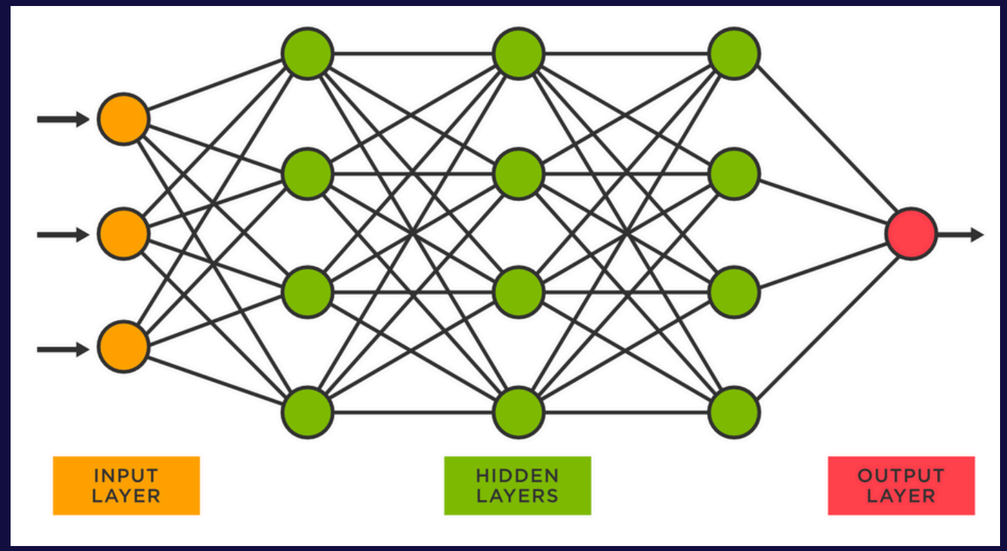
Neural Networks

- **Definition:**

Neural networks are computational models inspired by the human brain, designed to identify patterns and relationships in data.

- **Components:**
  - Input Layer: Receives raw data.
  - Hidden Layers: Perform feature extraction through weighted connections.
  - Output Layer: Produces the final prediction or classification.





Neural Networks

Key Concepts:

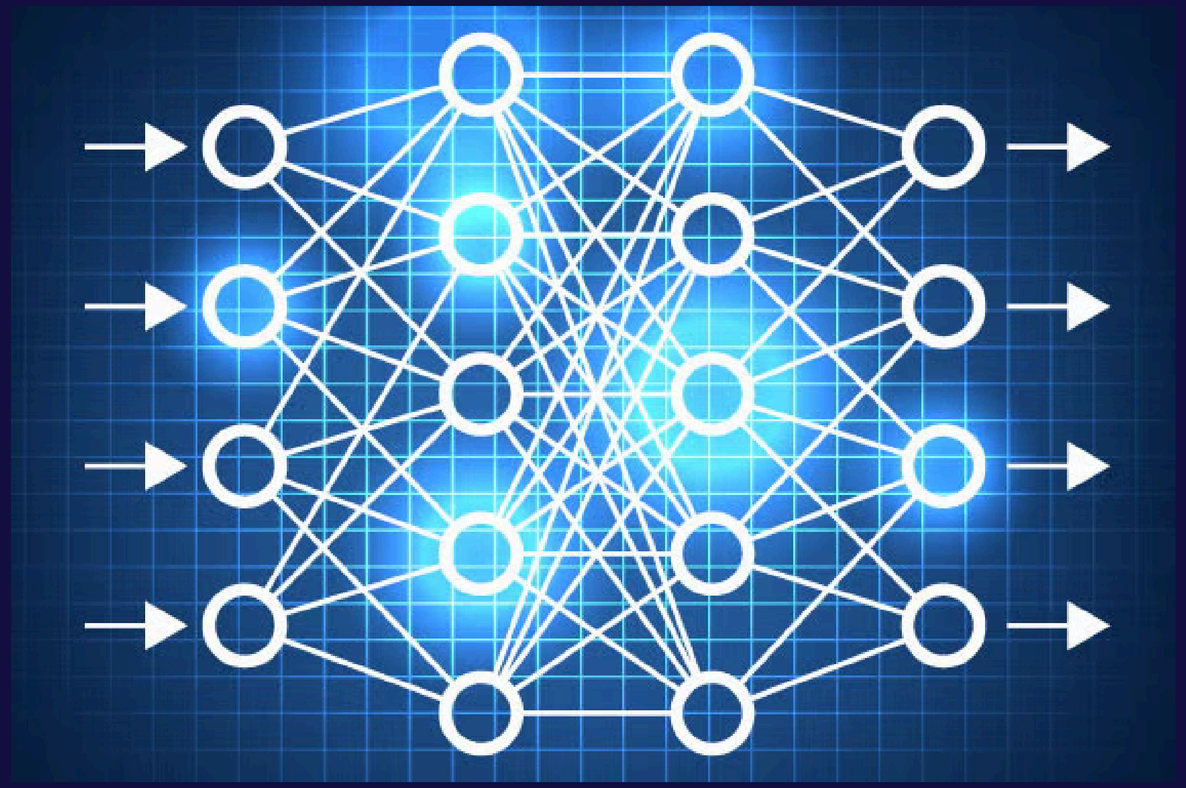
Activation Functions: Sigmoid, ReLU, and Softmax.

Backpropagation:

Optimizes weights using gradient descent.

Applications:

Image recognition, speech recognition, language modeling, and more.



Natural Language Processing

Definition:

A subfield of AI that focuses on enabling computers to understand, interpret, and generate human language.

Core Techniques:

Tokenization: Splitting text into meaningful units.

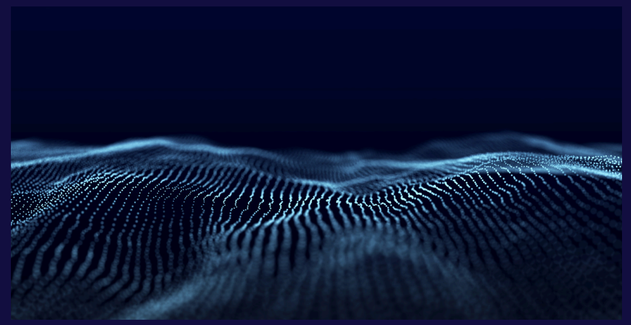
Embedding: Mapping words to numerical vectors for computational processing.

Parsing: Analyzing sentence structures.

Applications:

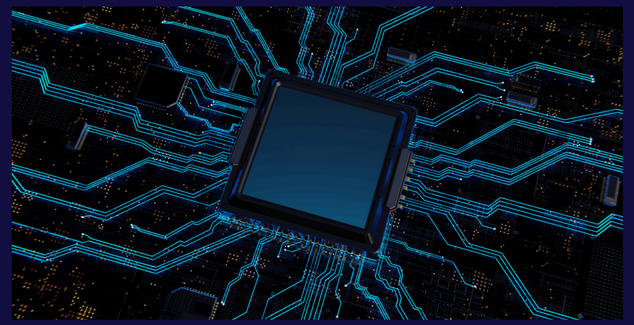
Machine Translation (Google Translate).
Chatbots (e.g., ChatGPT).
Sentiment Analysis (analyzing reviews or social media).
Information Retrieval (search engines).

What is LLMs?



Transformer Architecture

-Encoder-Decoder Structure
-Self-Attention Mechanism
-Multi-Head Attention
-Positional Coding
-Layer Norm and Feed Forward Networks



Educational Process and Features

-Pre-training Process
-Fine-tuning Process
-Model Features
-Training Data and Dimension
-Performance Measures

Model Optimization Techniques

-Scalability Challenges
-Regularization Techniques
-Hyperparameter Tuning
-Efficiency in Training

TRANSFORMER ARCHITECTURE

Encoder-Decoder Structure

Overview:

The Transformer architecture eliminates the need for sequential computation (used in RNNs), enabling parallelism and efficient handling of long-range dependencies.

Encoder:

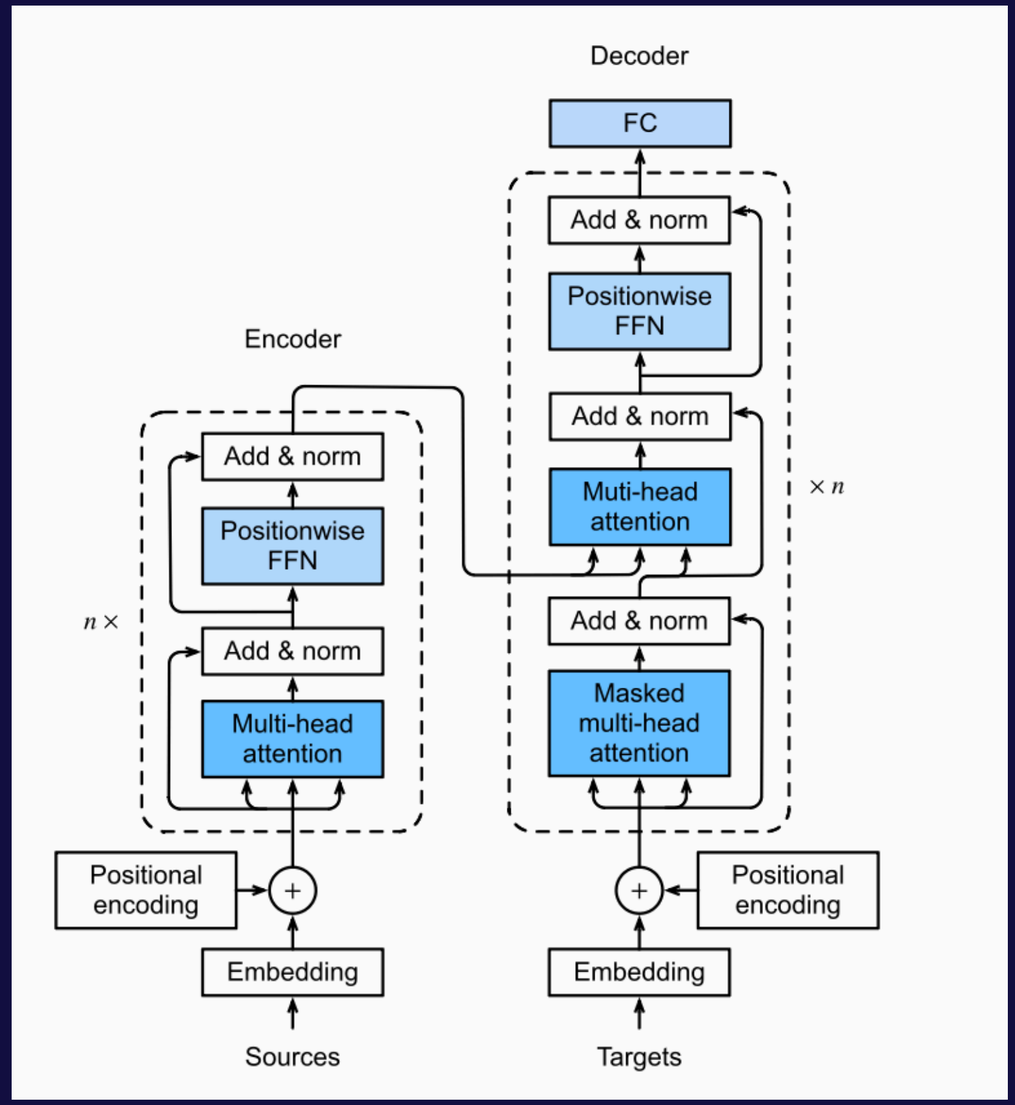
Transforms input sequences into contextualized embeddings.
Composed of self-attention and feed-forward layers.

Decoder:

Generates outputs based on the encoder's embeddings and previously generated tokens.

Benefits:

Scalability and high efficiency.
Handles context over long sequences.



TRANSFORMER ARCHITECTURE

Self-Attention Mechanism

What is Self-Attention?
A mechanism that determines the relevance of each word in a sequence to every other word, enabling the model to understand context effectively.

Query, Key, and Value:

Query (Q): What this word is looking for in the sequence.

Key (K): Encodes the importance of words for the current query.

Value (V): Contains the representation to pass forward.

Formulas:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

- X: Input embeddings
- W^Q, W^K, W^V: Trainable weight matrices

Self-Attention Calculation:

1. **Score:**

$$\text{Score}_{ij} = \frac{Q_i \cdot K_j^T}{\sqrt{d_k}}$$

2. **Softmax Normalization:**

$$\alpha_{ij} = \text{softmax}(\text{Score}_{ij})$$

3. **Weighted Sum:**

$$\text{Output}_i = \sum_j \alpha_{ij} V_j$$

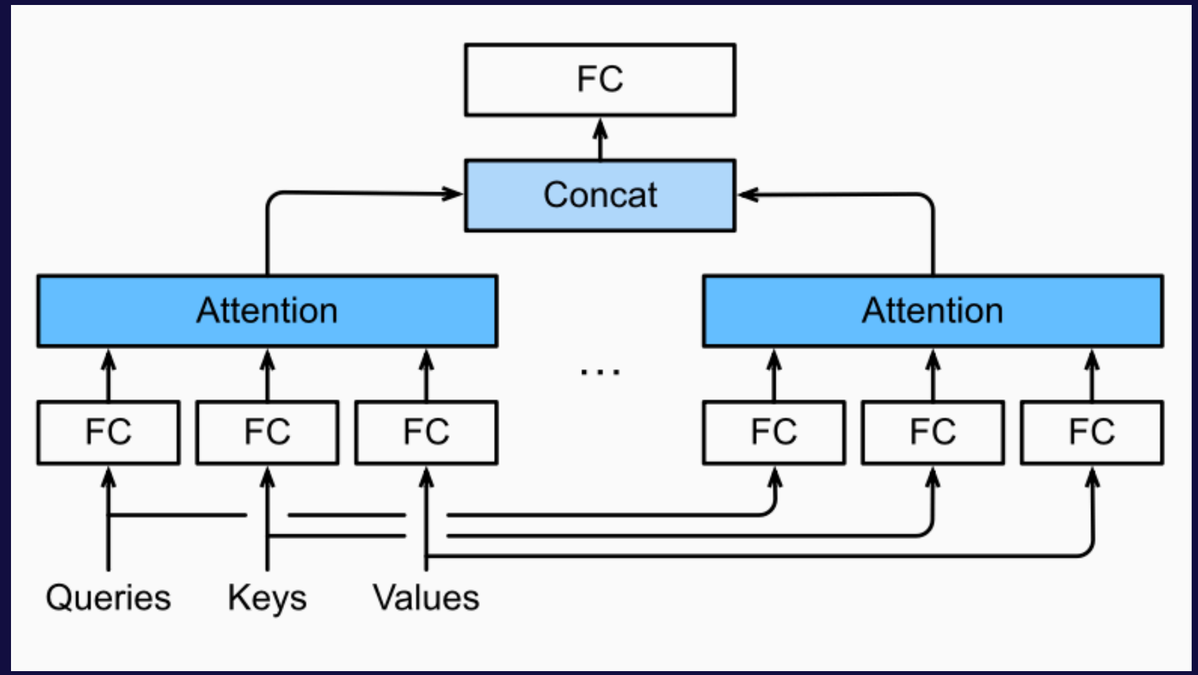
TRANSFORMER ARCHITECTURE

Multi-Head Attention

Definition:
Extends the self-attention mechanism by using multiple attention heads.

Process:
Input vectors are linearly transformed into multiple subspaces.
Self-attention is computed independently in each subspace.
Outputs are concatenated and passed through a final linear layer.

Importance:
Captures diverse linguistic features.
Balances global and local dependencies.



TRANSFORMER ARCHITECTURE

Positional Coding

Necessity:

Transformers lack inherent sequence ordering; positional encoding provides the sequence information.

Key Points:

Encodes position information using trigonometricfunctions.
Facilitates handling sequences of varying lengths.

Formula:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

TRANSFORMER ARCHITECTURE

Layer Norm and Feed Forward Networks

Layer Normalization:

Stabilizes training by normalizing activations across the feature dimension.

Feed-Forward Networks(FFN):

Independently processes each position using two linear layers with a ReLU activation.

Formula:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma + \epsilon}$$

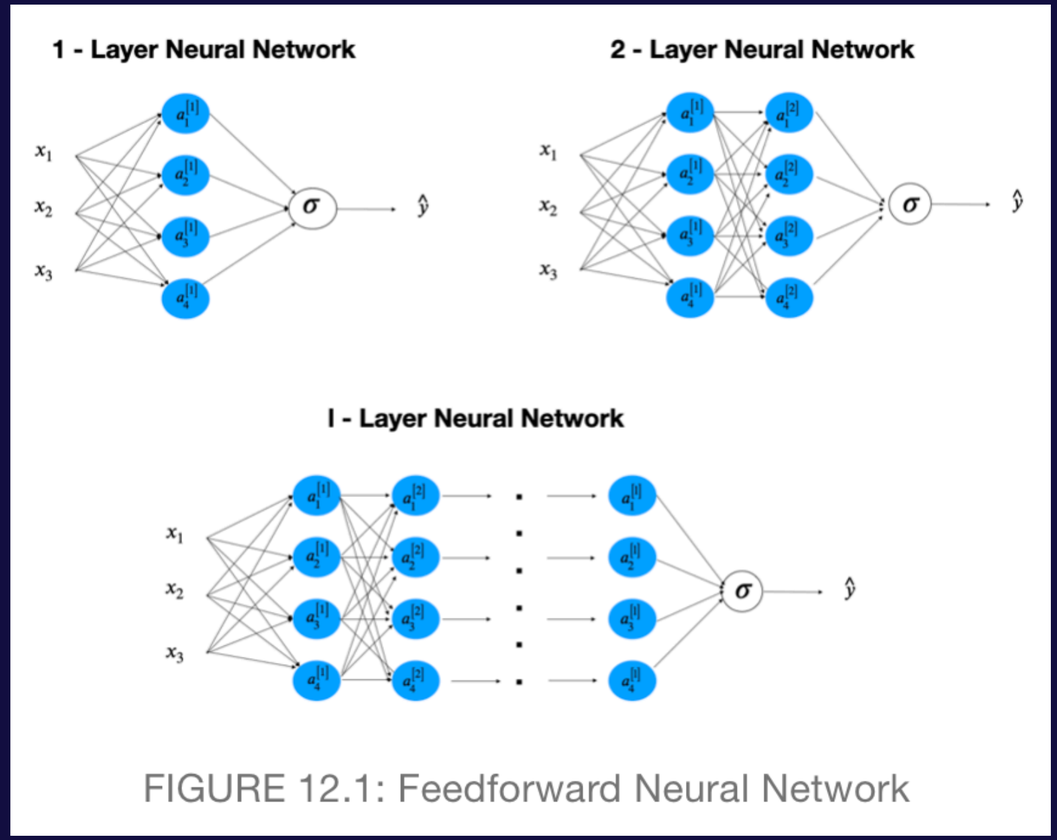


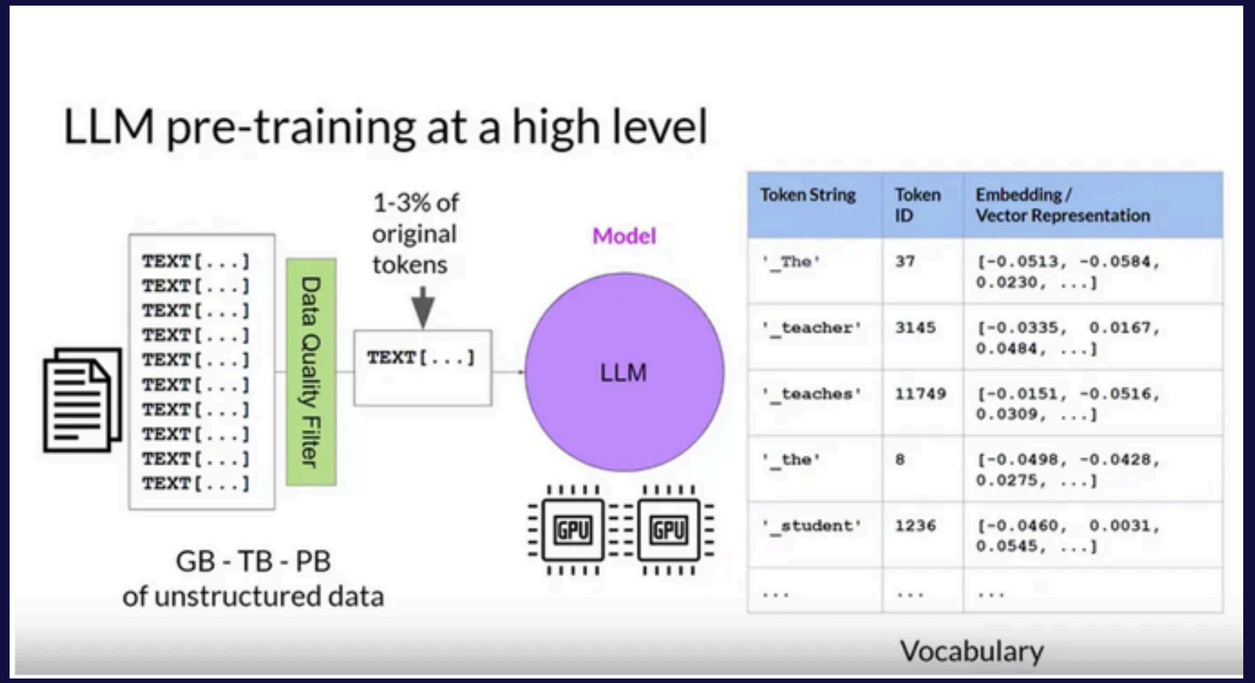
FIGURE 12.1: Feedforward Neural Network

Educational Process and Features

Pre-training Process

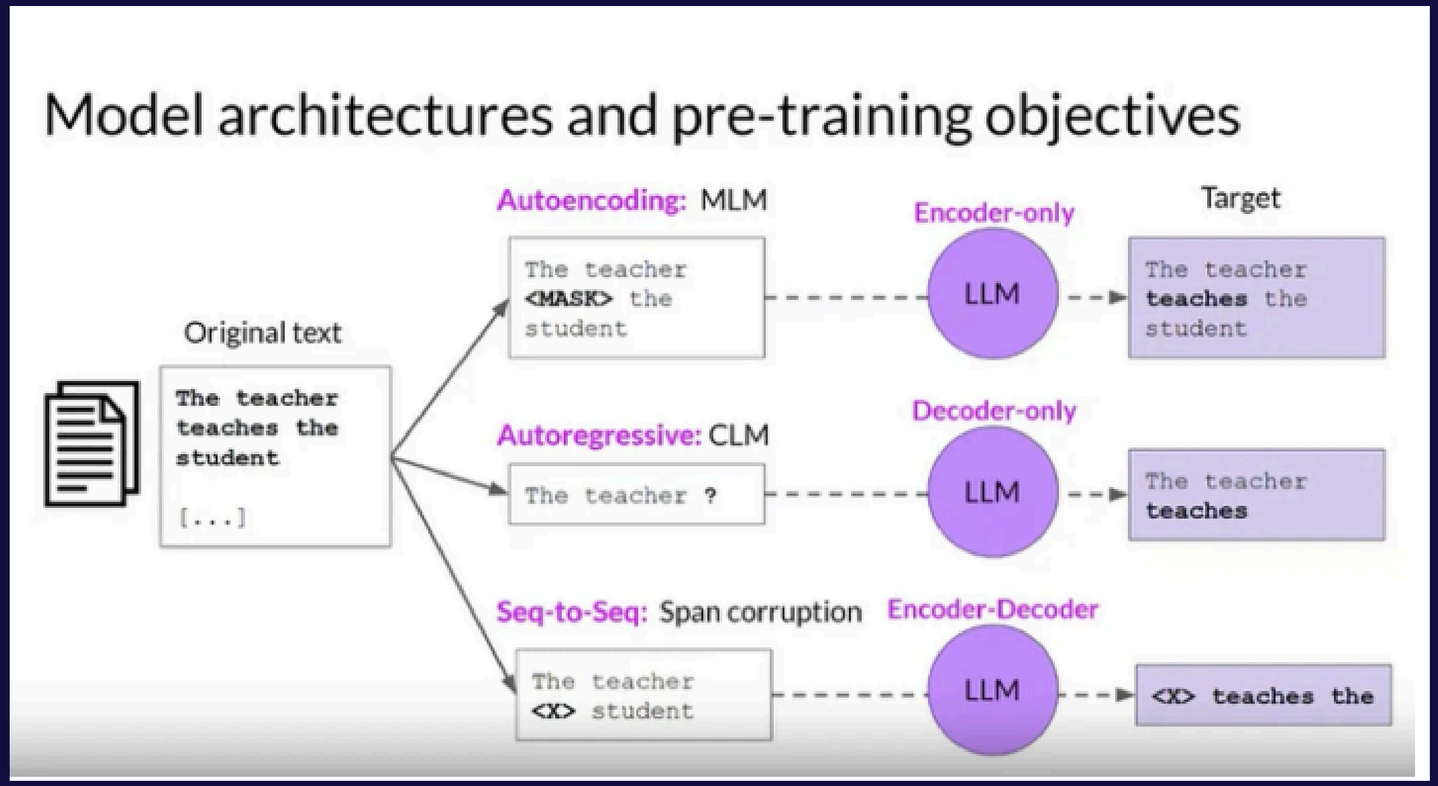
- **Objective:**
- **Learn general language representations using large, unsupervised datasets.**

- **Techniques:**
- **Masked Language Modeling (MLM): Predicting masked tokens in sentences (e.g., BERT).**
- **Causal Language Modeling: Predicts the next token in a sequence (e.g., GPT).**



Educational Process and Features

Pre-training Process



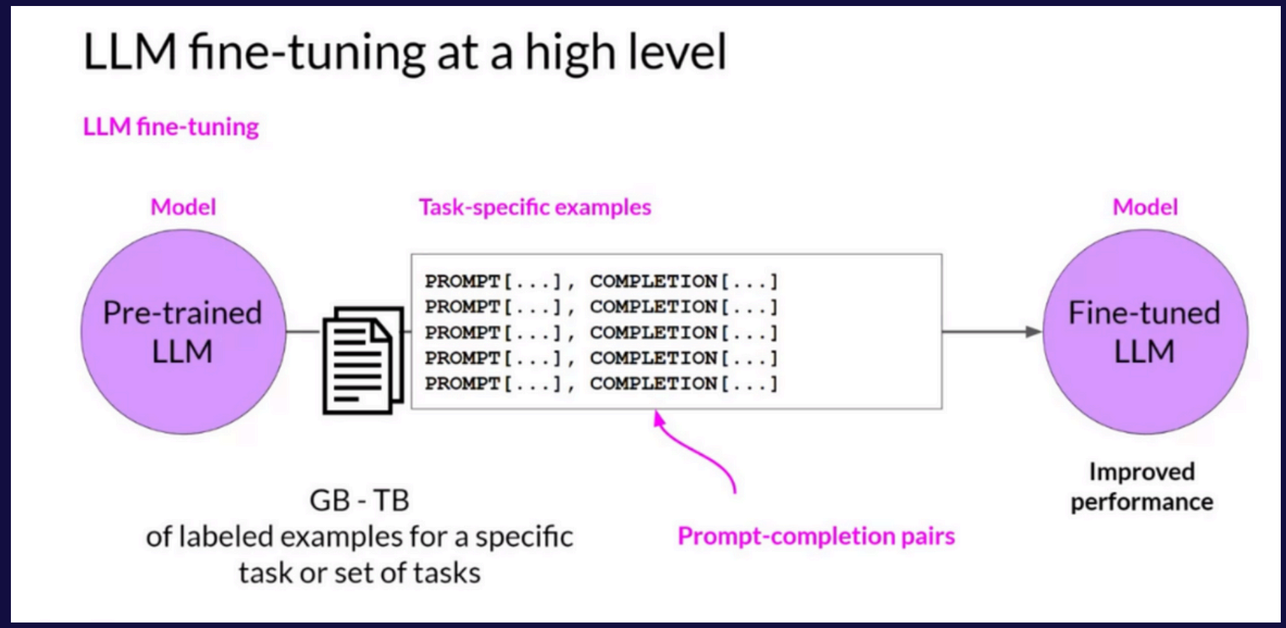
Educational Process and Features

Fine-tuning Process

Objective:

- Adapt pre-trained models to specific tasks using supervised learning.
-

- **Approach:**
- Retain pre-trained weights while training on labeled datasets.
- Common tasks: Classification, translation, summarization.



Educational Process and Features

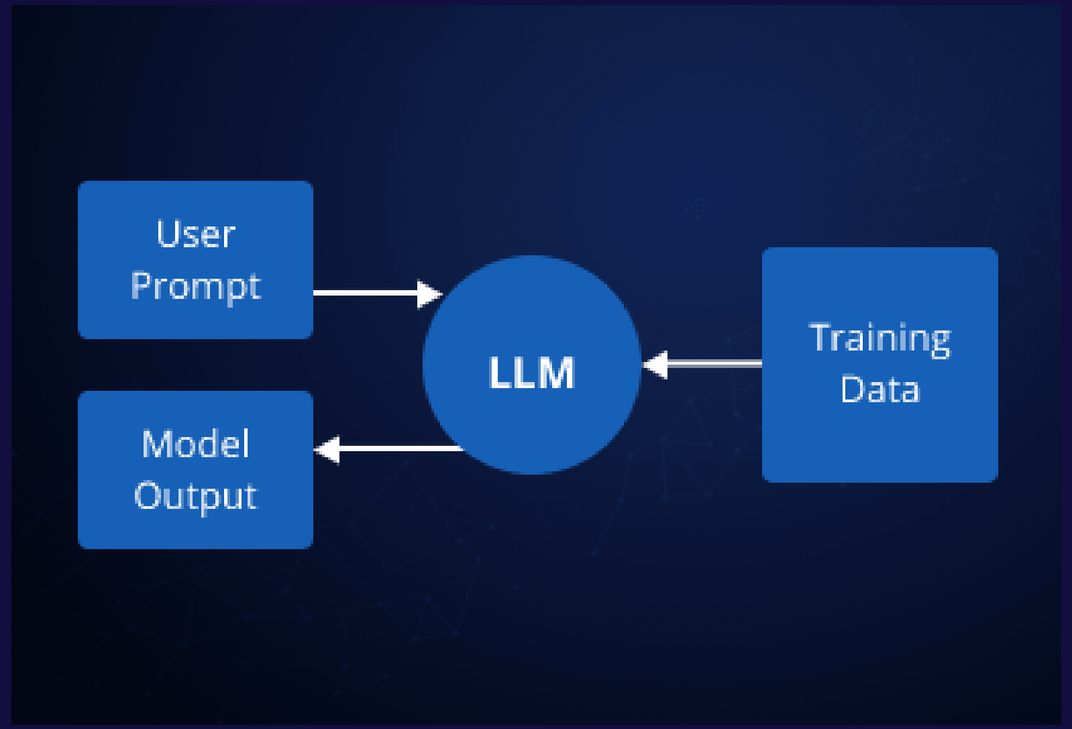
Model Features

- **Key Aspects:**
  - Contextual embeddings that adapt based on surrounding tokens.
  - High parameter scalability (e.g., billions of parameters).
  - Flexible architecture for diverse NLP tasks.

Educational Process and Features

Training Data and Dimension

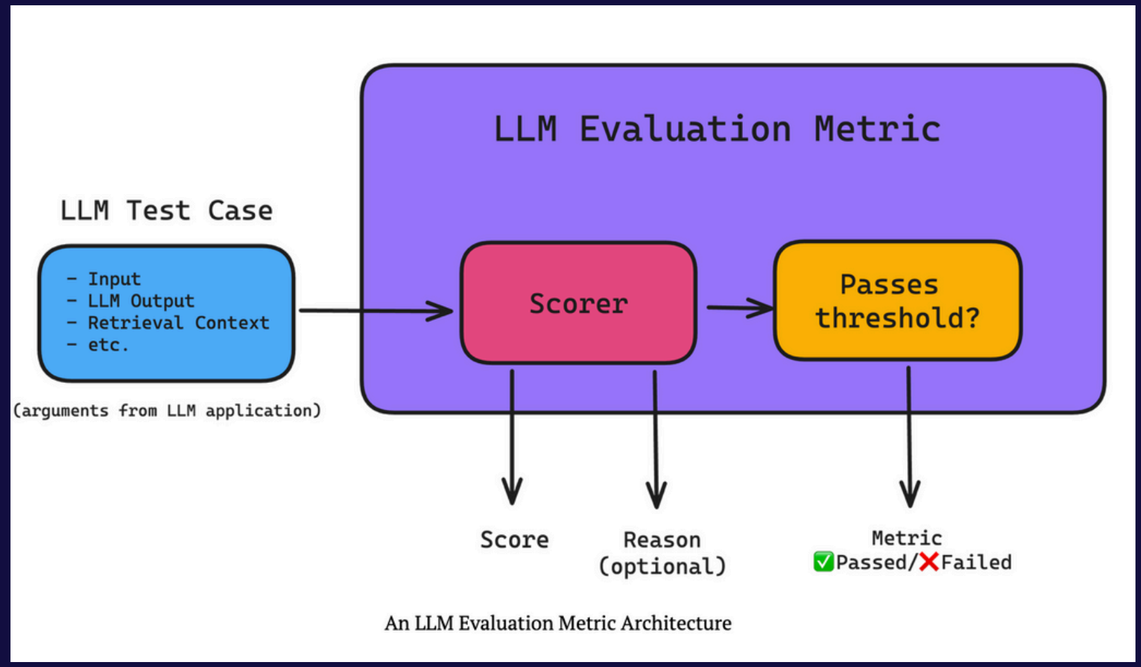
- **Data Characteristics:**
- **Large-scale corpora (books, Wikipedia, web text).**
- **Tokenization strategies (e.g., byte-pair encoding).**
- **Model Scaling:**
- **Token count: Up to trillions.**
- **Parameters: Models like GPT-4 exceed 175 billion parameters.**



Educational Process and Features

Performance Measures

- **Metrics:**
  - **Perplexity:** Measures uncertainty in predictions.
  - **BLEU/ROUGE:** Used for evaluating translation and summarization tasks.
- **Efficiency Metrics:**
  - **Latency and throughput during inference.**



An LLM Evaluation Metric Architecture

Model Optimization Techniques

Scalability Challenges

Issues:
GPU/TPU memory limits.
Long training times.

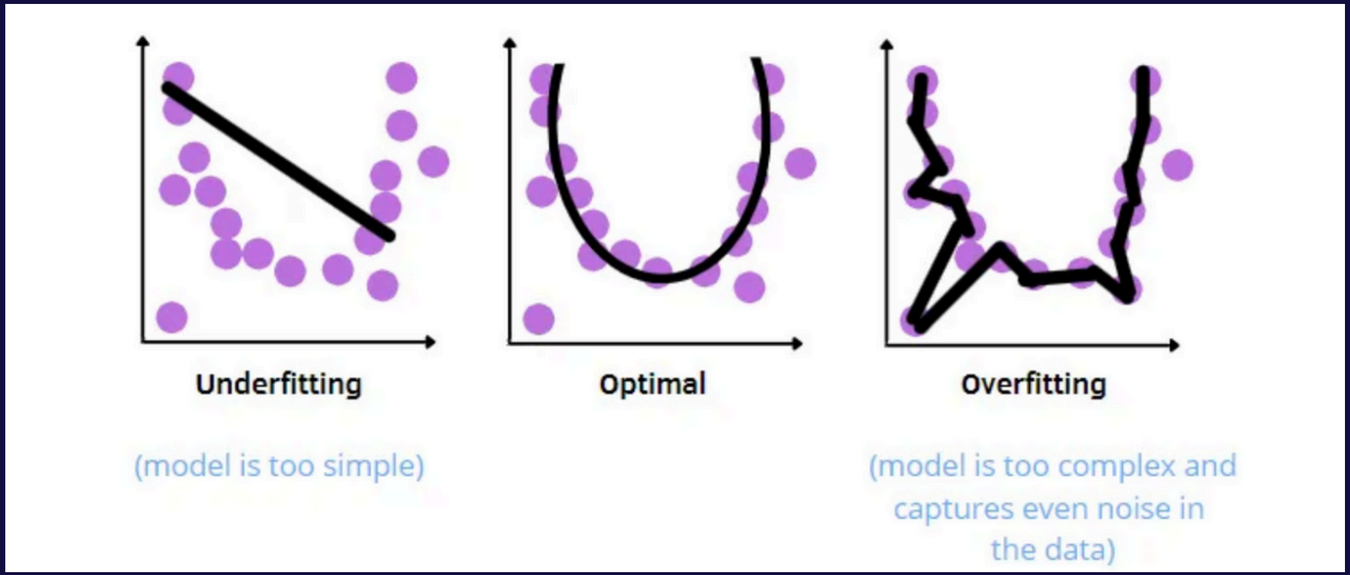
Solutions:
Model parallelism.
Mixed precision training.

Model Optimization Techniques

Regularization Techniques

Methods:

Dropout to prevent overfitting.
Weight decay for controlling large weights.
Data augmentation for improved generalization.



Model Optimization Techniques

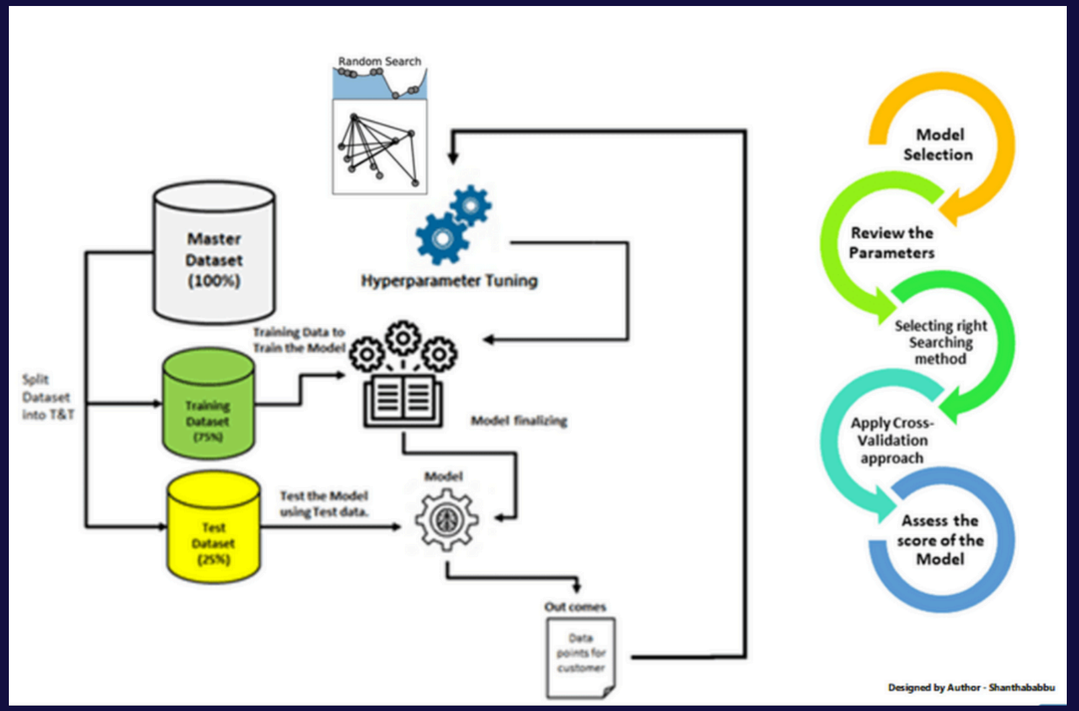
Hyperparameter Tuning

Key Parameters:
Learning rate, batch size, attention head count.

Techniques:

Grid Search: Exhaustive search over hyperparameter combinations.

Random Search: Efficient alternative for high-dimensional spaces.



Model Optimization Techniques

Efficiency in Training

- **Strategies:**
-
    - **Knowledge Distillation:** Compresses large models into smaller, efficient versions.
    - **Gradient Accumulation:** Handles large batch sizes with limited memory.
    - **Checkpointing:** Reduces memory usage during backpropagation.