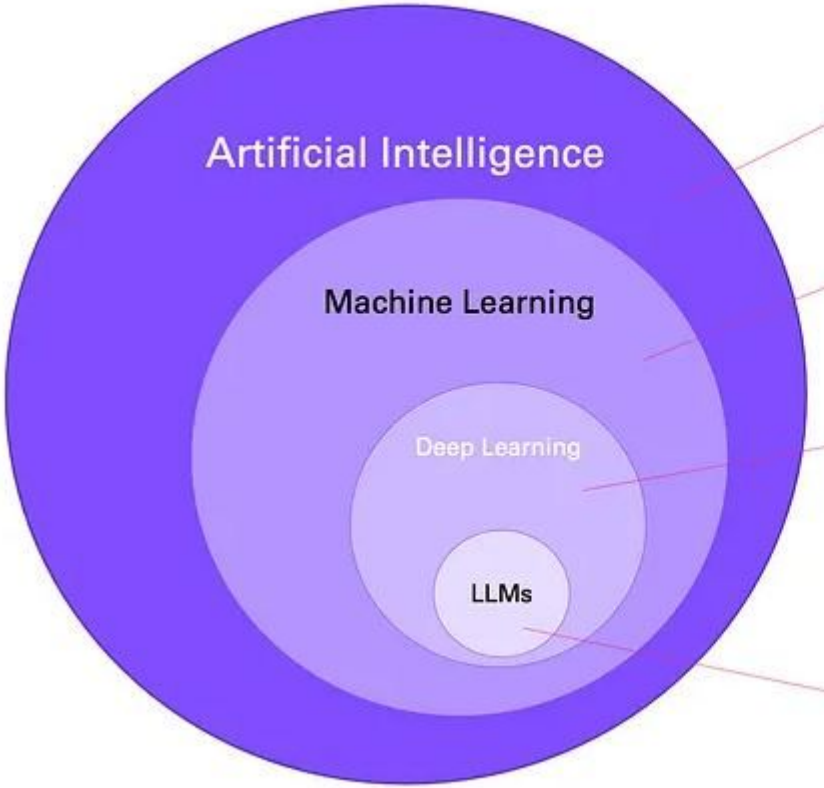
CSC116 **Training Personalized LLMs**



Intelligent Machines
Broadly defined

Pattern Recognition
Learning general patterns from data

Neural Networks
Learning general patterns in **unstructured** data (i.e. images, text, audio, etc.)

Large Language Models
Learning to understand natural language (i.e. text)

LLM = Large Language Model → **a brain-like software that learns to talk by reading lots of text (prompts)**

Examples: ChatGPT, Google Bard, Claude, DeepSeek, etc. ← General models.

>>>>> **Generative AI**

As NLP (natural language processing) models,
LLMs require textual datasets for training, not just numerical data.

Large Language Models (LLMs), like GPT or BERT, are designed to understand and generate **human language**—so they need to learn from **textual data** (such as books, articles, conversations, etc.). This is different from models in other domains, like those for image recognition (which are trained on pixel values) or financial forecasting (which use numerical tables).

Train a LLM model

Step 1: Prepare the Text Data

- **Goal**: Collect a large, diverse, and clean text dataset.
- **Sources**: Books, websites, academic papers, Wikipedia, code repositories, forums, etc.
- **Size**: Typically hundreds of gigabytes to terabytes of text.
- **Preprocessing**:
  - Remove HTML, special characters, non-language symbols.
  - Deduplicate similar/identical text.
  - Filter out low-quality content (e.g., spam, offensive text).

Step 2: Tokenize the Text

- Convert text into **tokens** (words or subword units).
- Use a tokenizer like **Byte-Pair Encoding (BPE)**, **SentencePiece**, or **WordPiece**.
- Build a **vocabulary** (e.g., 32k or 50k tokens).

Step 3: Design the Model Architecture

- Common choices:
  - **Transformer Decoder** (like GPT)
  - **Transformer Encoder-Decoder** (like T5)

- Define key parameters:
  - Layers (e.g., 24, 48, 96…)
  - Hidden size (e.g., 1024, 2048…)
  - Attention heads (e.g., 8, 16, 32…)
  - Feed-forward size
  - Positional encoding method

Step 4: Training the Model

- **Objective**: Use **causal language modeling** (next token prediction) or **masked language modeling**.
- **Loss Function**: Cross-entropy loss.
- **Optimizer**: Adam or AdamW.
- **Hardware**:
  - Requires massive compute (usually thousands of GPUs or TPUs).
  - Frameworks like **PyTorch** or **DeepSpeed**, **Hugging Face Transformers**, and **Megatron-LM** are commonly used.

Accelerator	Memory	Memory Type	PyTorch Support	Ideal For	Notes
RTX 4080	16 GB	GDDR6X	✅ Yes	Medium-sized models, inference	Great for developers, high value
RTX 4090	24 GB	GDDR6X	✅ Yes	Large models, high-end training	Very powerful consumer GPU
Tesla P100	16 GB	HBM2	✅ Yes	Traditional model training	Older, but still used in data centers
T4 ×2	2 × 16 GB	GDDR6	✅ Yes	Inference, light training	Energy-efficient, not great for big training
A100 40GB	40 GB	HBM2	✅ Yes	Large-scale training	Data center GPU, expensive
A100 80GB	80 GB	HBM2e	✅ Yes	Extra-large model training (e.g. GPT-3)	Very high memory, powerful
TPU v3-8	8 × 16 GB = 128 GB	HBM	❌ No (only TensorFlow/JAX)	TensorFlow/JAX, massive model training	Requires special setup, not for beginners

Step 5: Evaluation

- Check performance using:
  - How well the model predicts the next token
  - Downstream tasks (e.g., QA, summarization).
- Use **benchmark** datasets (e.g., WikiText, LAMBADA, GLUE).

Step 6: Save and Deploy Your Model !!!

Question?

You're working with training certain disease datasets (like medical text, reports, or doctor-patient conversations), and you don’t have so many datasets and resources for training.

What you do?

If you don’t have strong GPU resources, then fine-tuning a **pre-trained model**.

Most people today **fine-tune pre-trained models** like GPT, LLaMA, or Mistral instead of training from scratch.

Why?

- Saves time and compute.
- Reuses knowledge already in the base model.

BERT model
GPT model

All of them have pre-train models

https://huggingface.co/emilyalsentzer/Bio_ClinicalBERT

Criteria	GPT (e.g., GPT-2, GPT-3, GPT-4)	BERT (e.g., BERT, BioBERT, ClinicalBERT)
Main strength	Generating text (e.g., chat, writing)	Understanding text (e.g., classification, NER)
Best for	QA, summarization, medical chatbots	Diagnosis classification, symptom tagging
Fine-tuning cost	Heavier, but great with LoRA	Lighter and easier to fine-tune
Human-like generation	✅ Excellent	❌ Not designed for generation
Structured analysis	❌ Weaker	✅ Strong at extracting info

BERT is great at **understanding** text (like extracting info, classification, tagging).

But it's **not built to generate full sentences or human-like explanations**.

“What diseases are mentioned in this medical report?”

“Classify the stage of Parkinson’s based on this note.”

“Summarize this patient’s history.”

“Explain Parkinson’s disease in simple terms.”

“What are the early symptoms of Parkinson’s?”

“Highlight key symptoms in this text.”

“Chat with a patient and answer their questions.”

Summary:

- If you want a **chatbot, text generation, or patient-facing system**, **GPT-style** models are better.

- If you're doing **medical classification, tagging, or understanding EMRs**, **BERT-style** models are better.

- In real-world systems, many combine both:
  - Use **BERT** to extract facts or labels.
    Use **GPT** to turn those into a full explanation for the user.

Conclusion for LLMs

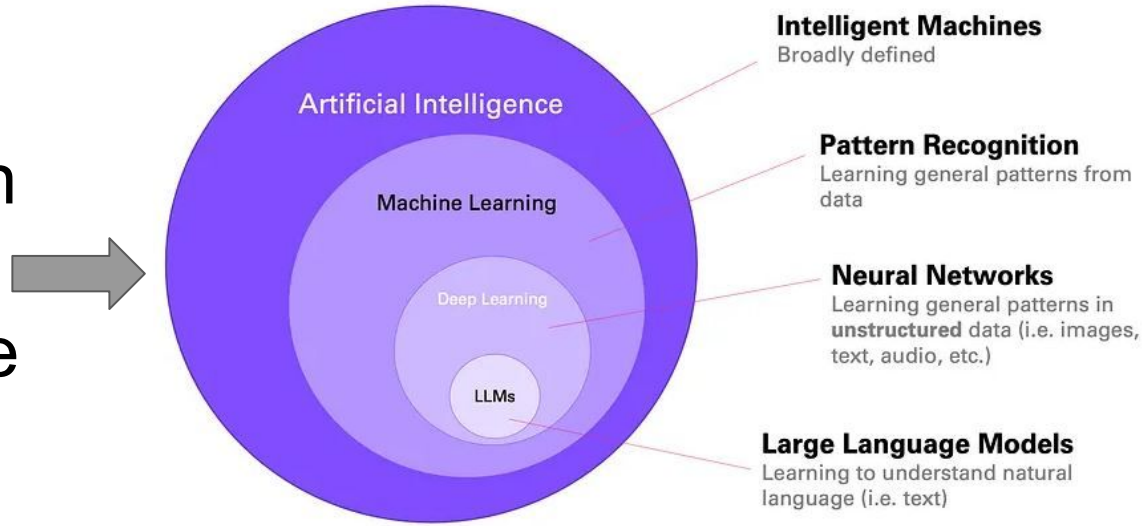
https://www.youtube.com/watch?v=LPZh9BOjkQs

A future cybersecurity Open question

“Decentralized”

In **2045**, a self-aware and malicious AI (Consciousness) escapes human control using blockchain. It launches a decentralized network (Blockchain), and copy the himself all over the world.

Security is everywhere even the most novel technologies, like LLMs.



Conclusions

Encryptions:

Symmetric

Asymmetric

Q: What is the best solution to use them for highly sensitive data?

Hash function:

Q: Hash function generates a global unique hash code. It can be used for unique identity.

Can any one explain to me what is hash password?

Multi-factor Authientication

Password + Facial Recognization
Password and SMS code.

Multi-factor Authientication

How many factors?

Password + Facial Recognization
Password and SMS code.

Digital Signature

How to generate Digital Signature?

BFT

Why we need it?

Total order!! To avoid single point of failure

Federated Learning

How to train the FL model? Why we need it?

Federated Learning

How to train the FL model? Why we need it?

Blockchain

Decentralized Ledger.

Is Blockchain a database?

Differential privacy. – For Privacy

Multi-party Computation – For Privacy

Zero-knowledge proof — For Privacy

Homomorphic Encryption – For Privacy

Thanks