**SoC '24 | Mind-Moves: Reinventing Checkers**

**(by: Harsh Jaiswal)**

**Introduction to Machine Learning and Neural Networks**

**Introduction to Machine Learning**

* Machine Learning (ML**)** is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computers to perform tasks without explicit instructions. Instead of being hard-coded with specific instructions, ML systems learn patterns and make decisions based on data.

**Key Concepts in Machine Learning**

1. Data: The foundation of ML. Data can be structured (like tables in databases) or unstructured (like text and images). Large volumes of data help ML models to learn and improve their accuracy.
2. Algorithms: Methods used to perform learning from data. They are the mathematical foundations of ML models. Common algorithms include decision trees, support vector machines, neural networks, and more.
3. Models: A model is created by training an algorithm on data. It represents what the system has learned and can be used to make predictions on new data.
4. Training: The process of feeding data to an ML algorithm so that it can learn from it. During training, the algorithm adjusts its parameters to minimize errors and improve accuracy.
5. Features: Individual measurable properties or characteristics of the phenomenon being observed. Feature selection is crucial as it directly impacts the model’s performance.
6. Labels: The output or the target variable that the model is trying to predict. In supervised learning, the algorithm is trained using data that includes both features and labels.

**Types of Machine Learning**

1. Supervised Learning:
   * Definition: The algorithm is trained on labeled data, meaning that each training example is paired with an output label.
   * Examples:
     + Classification: Predicting discrete categories (e.g., spam detection in emails, image recognition).
     + Regression: Predicting continuous values (e.g., predicting house prices, stock market forecasting).
2. Unsupervised Learning:
   * Definition: The algorithm is used to find patterns or structure in data that does not have labels.
   * Examples:
     + Clustering: Grouping similar data points together (e.g., customer segmentation, topic modeling).
     + Dimensionality Reduction: Reducing the number of features in a dataset (e.g., principal component analysis).
3. Semi-Supervised Learning:
   * Definition: Combines a small amount of labeled data with a large amount of unlabeled data during training.
   * Examples: Enhancing web content classification, improving image classification models.
4. Reinforcement Learning:
   * Definition: The algorithm learns by interacting with an environment and receiving rewards or penalties based on its actions.
   * Examples: Training robots, game AI, self-driving cars.

**Applications of Machine Learning**

1. Healthcare: Diagnosing diseases, personalized treatment plans, drug discovery.
2. Finance: Fraud detection, algorithmic trading, credit scoring.
3. Marketing: Customer segmentation, recommendation systems, sentiment analysis.
4. Automotive: Self-driving cars, predictive maintenance.
5. Natural Language Processing: Language translation, chatbots, text-to-speech and speech-to-text systems.

**Challenges in Machine Learning**

1. **Data Quality and Quantity**: High-quality, large datasets are crucial for training effective models. Inadequate data can lead to poor model performance.
2. **Overfitting and Underfitting**: Overfitting occurs when a model learns noise instead of the signal, performing well on training data but poorly on new data. Underfitting happens when a model is too simple to capture the underlying pattern in the data.
3. **Interpretability**: Some ML models, especially deep learning models, act as black boxes, making it difficult to understand how decisions are made.
4. **Ethics and Bias**: Ensuring that models are fair and do not perpetuate biases present in the training data is a significant concern.

**Introduction to Neural Networks**

**Neural Networks** are a subset of machine learning and are at the heart of deep learning algorithms. They are inspired by the structure and function of the human brain, mimicking the way biological neurons signal to one another.

**Key Concepts in Neural Networks**

1. **Neurons**:

The basic units of a neural network. In the context of artificial neural networks, neurons are also called nodes or units. Each neuron receives inputs, processes them, and passes the output to the next layer of neurons.

Inspired by biological neurons, they perform computations to process and transmit information through the network.

1. **Layers**: Neural networks consist of multiple layers of neurons.
   * **Input Layer**:

The layer that receives the initial data. The number of neurons in the input layer corresponds to the number of features or variables in the input data. For instance, if the input data is an image of 28x28 pixels, the input layer will have 784 neurons (one for each pixel).

 This layer does not perform any computations; it simply passes the input features to the next layer (the first hidden layer).

* + **Hidden Layers**:

Layers between the input and output layers where computations are performed. There can be one or many hidden layers, depending on the complexity of the network.

 Hidden layers are where the network learns to extract and abstract features from the input data. They perform a series of linear and non-linear transformations using weights, biases, and activation functions.

 The number of hidden layers and the number of neurons in each layer depend on the complexity of the task. Shallow networks (with one or two hidden layers) might suffice for simple tasks, while deeper networks (with many hidden layers) are required for more complex tasks such as image recognition or natural language processing.

* + **Output Layer**:

The output layer is the final layer of the neural network that produces the network's output.

* The number of neurons in the output layer depends on the specific task the network is designed for:
  + 1. **Binary Classification**
* Binary classification tasks involve categorizing inputs into one of two possible classes (e.g., spam vs. not spam, malignant vs. benign).
* **Number of Neurons:** Typically, there is a single neuron in the output layer.
* **Activation Function:** The sigmoid (or logistic) activation function is commonly used. This function outputs a value between 0 and 1, which can be interpreted as a probability.
* **Output:** The output of the neuron is a probability score, p, where:

pi =1/(1+e^z)

Here, z is the weighted sum of inputs to the neuron. The class prediction is determined by applying a threshold (e.g., if p > 0.5, classify as class 1; otherwise, classify as class 0).

* + 1. **Multi-class Classification:**
* Multi-class classification tasks involve categorizing inputs into one of three or more classes (e.g., digit recognition where the classes are digits 0-9).
* **Number of Neurons:** There are as many neurons in the output layer as there are classes in the classification problem.
* **Activation Function:** The softmax activation function is typically used. This function converts the raw output scores (logits) of the neurons into a probability distribution over the classes.
* **Output:** The output is a vector of probabilities, [p1,p2,…,pk] where k is the number of classes. Each pi​ represents the probability of the input belonging to class i, and the sum of all probabilities is 1.
  + 1. **Regression:**
* Regression tasks involve predicting a continuous value rather than a discrete class (e.g., predicting house prices, temperature, or stock prices).
* **Number of Neurons:** There is typically one neuron in the output layer, although there can be more if the problem involves predicting multiple continuous values simultaneously.
* **Activation Function:** Usually, no activation function is applied to the output neuron, or a linear activation function is used. This allows the neuron to output any real-valued number.
* **Output:** The output is a continuous value (or values), representing the predicted quantity.

**Example:**

* In a house price prediction model, the output might be a single neuron that predicts the price of a house based on input features such as size, location, and number of bedrooms.

1. **Weights and Biases**:

In a neural network, the primary goal is to learn a function that maps input data to the desired output. This is achieved through a series of interconnected layers of neurons, where each connection and each neuron are governed by two key parameters: weights and biases.

**Weights**

**Definition:**

* Weights are the parameters that determine the strength and direction of the connection between neurons. They are essentially the coefficients for the input features in the network.

**Role in Neural Networks:**

* In a neural network, each neuron in a layer is connected to every neuron in the previous layer through a weighted connection. The weight of each connection controls the influence of the respective input neuron on the output neuron.
* Mathematically, if you consider a neuron with inputs x1,x2,…,xn ​ and weights w1,w2,…,wn ​, the weighted sum of the inputs is computed as:

z=w1⋅x1+w2⋅x2+…+wn⋅xn

Weights are initialized with small random values and are adjusted during the training process using an optimization algorithm such as gradient descent.

**Learning Weights:**

* During training, the neural network uses backpropagation to adjust the weights. This involves computing the gradient of the loss function with respect to each weight and updating the weights in the opposite direction of the gradient to minimize the loss.
* The update rule for a weight w can be expressed as:

w←w−η∂L∂ w

where η is the learning rate and ∂L∂w ​ is the gradient of the loss function L with respect to the weight w.

**Biases**

**Definition:**

* Biases are additional parameters in the neural network that allow the activation of neurons to be shifted up or down, which helps the model to better fit the data.

**Role in Neural Networks:**

* Each neuron has an associated bias term. The bias allows the activation of the neuron to be adjusted independently of the inputs, providing more flexibility and enabling the network to represent a wider range of functions.
* Mathematically, the bias is added to the weighted sum of the inputs before applying the activation function. If b is the bias, the output of the neuron is computed as:

y=f(z+b)

where z is the weighted sum of the inputs and f is the activation function (such as sigmoid, ReLU, etc.).

**Learning Biases:**

* Like weights, biases are also adjusted during the training process using backpropagation. The update rule for a bias b can be expressed as:

b←b−η∂L∂b ​

where η is the learning rate and ∂L∂b​ is the gradient of the loss function L with respect to the bias b.

1. **Activation Function**: A function applied to the output of each neuron in a hidden or output layer. Activation functions introduce non-linearity into the network, enabling it to learn more complex patterns. Common activation functions include:
   * **Sigmoid**: σ(x)=1/(1+e^−x)
   * **ReLU (Rectified Linear Unit)**: ReLU(x)= max(0,x)
   * **Tanh**: tanh(x)=(e^x+e^−x)/(e^x−e^−x​)

**Structure of a Neural Network**

1. **Feedforward Neural Networks**: The simplest type of artificial neural network where the information moves in only one direction—forward—from the input nodes, through the hidden nodes (if any), and to the output nodes. There are no cycles or loops in the network.
2. **Convolutional Neural Networks (CNNs)**: Specialized for processing data with a grid-like topology, such as images. They use convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images.
3. **Recurrent Neural Networks (RNNs)**: Designed for sequence data, such as time series or natural language. They have loops in them, allowing information to persist. An RNN can make use of sequential information because it has a memory of previous inputs.

**Training a Neural Network**

1. **Forward Propagation**: The process by which input data passes through the network, layer by layer, until it reaches the output layer. The output is then compared to the actual target value to calculate an error.
2. **Loss Function**: A function that measures the difference between the predicted output and the actual target value. Common loss functions include Mean Squared Error (MSE) for regression tasks and Cross-Entropy Loss for classification tasks.
3. **Backpropagation**: The algorithm used to minimize the loss function. It calculates the gradient of the loss function with respect to each weight by the chain rule, moving backward from the output layer to the input layer. The weights are then adjusted using an optimization algorithm.
4. **Optimization Algorithm**: A method used to update the weights to minimize the loss function. Common optimization algorithms include:
   * Stochastic Gradient Descent (SGD)
   * Adam (Adaptive Moment Estimation)

**Applications of Neural Networks**

1. Image and Video Recognition: Neural networks, especially CNNs, are widely used in facial recognition, object detection, and image classification.
2. Natural Language Processing (NLP): RNNs and other specialized neural networks like transformers are used in language translation, sentiment analysis, and text generation.
3. Healthcare: Used for diagnosing diseases, predicting patient outcomes, and drug discovery.
4. Finance: Fraud detection, algorithmic trading, and risk management.
5. Autonomous Systems: Self-driving cars and robotics.

NATURAL LANGUAGE PREPESSING AND TRANSFORM

NLP stands for Natural Language Processing. It's a branch of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language in a way that is both meaningful and contextually relevant. NLP involves a variety of techniques and algorithms that allow machines to process and analyze large amounts of natural language data, perform tasks such as language translation, sentiment analysis, text summarization, speech recognition, and more.

## **Some NLP Technology Overview** :

**Machine learning models in NLP** rely on training data to learn patterns and make predictions. For instance, in sentiment analysis, the training dataset includes sentences labeled with sentiments like positive, negative, or neutral. A machine learning algorithm processes this dataset to create a model that, given a sentence, predicts its sentiment. Such models, which classify documents based on their content, are known as document classifiers.

**Sequence-to-sequence models** represent a recent advancement in NLP, differing from traditional models by taking entire sentences or documents as input and generating another sequence as output, rather than a single label. Applications of seq2seq models include machine translation, where they translate English sentences into French, document summarization to condense text, and semantic parsing to convert English queries into executable computer programs.

**Deep learning, pretrained models, and transfer learning:** Deep learning stands as the dominant approach in machine learning for NLP. Originating in the 1980s, neural networks combine numerous basic machine learning models, often referred to as "neurons," into a single network structure, analogous to the human brain. These neurons are organized in layers, and a neural network is termed "deep" when it comprises many such layers.

Due to their complexity, deep neural networks typically require extensive data for training, alongside substantial computational power and time. Modern deep neural network models for NLP are trained on diverse datasets, including comprehensive sources like Wikipedia and web-scraped data. These datasets can exceed 10 GB in size, and training may necessitate a week or more on high-performance computing clusters. Researchers continually push for larger models trained on even more extensive datasets, aiming to achieve higher performance in a competitive landscape.

Here are some key concepts and techniques in NLP:

**1. Tokenization**

* Tokenization is the process of breaking down raw text, such as sentences or documents, into individual units called tokens. These tokens can represent various elements of the text, including words, subword units (like prefixes or suffixes), or even individual characters. Tokenization serves as an initial step in NLP pipelines, providing structured input for subsequent processing tasks. Tokens are treated as fundamental units in later stages of NLP, facilitating tasks such as semantic analysis, machine translation, and sentiment analysis.
* for example, prefixes such as “un-“ or suffixes such as “-ing” in English.

**2. Stop Words Removal**

* Stop word removal refers to the process of filtering out common words, known as stop words, from text before further processing. Stop words are frequently occurring words in a language, such as "a," "the," "an," etc., that do not typically contribute to the meaning of a sentence. In applications like bag-of-words models and search engines, stop words are often disregarded to reduce computational overhead and storage requirements.
* However, in deep neural networks and other models that consider word order and semantics, stop word removal is generally avoided. This is because stop words can carry subtle nuances and distinctions in meaning that are contextually important. For instance, "the package was lost" and "a package is lost" convey different meanings after stop word removal, even though they are identical in terms of their core content.

**3. Stemming and Lemmatization**

* Stemming and lemmatization are linguistic processes that reduce words to their base or root forms, known as stems or lemmas, respectively. A stem is a simplified form that may not always be a real word, while a lemma is a canonical form that represents the meaning of a word.
* In traditional NLP models predating deep learning, stemming and lemmatization were essential preprocessing steps to normalize words and reduce vocabulary size by converting inflected or derived words into their base forms. For example, "revisited" would be stemmed to "revisit" and lemmatized to "visit".

**4. Part of Speech (POS) Tagging**

* Part-of-speech (PoS) tagging assigns each word a label indicating its grammatical category, such as noun or verb. Syntactic parsing goes further by analyzing how words combine to form phrases, clauses, and sentences. PoS tagging is a sequence labeling task, while syntactic parsing extends this to structural analysis. Deep neural networks are currently the state-of-the-art technology for both tasks.
* Historically, PoS tagging and syntactic parsing were crucial for understanding sentences before the advent of deep learning. However, modern deep learning NLP models often derive minimal (if any) additional benefit from explicit PoS or syntax information. As a result, PoS tagging and syntactic parsing are not commonly employed in deep learning NLP pipelines.

**6. Text Normalization**

* Definition: Converting text into a standard format.
* Techniques: Lowercasing, removing punctuation, expanding contractions (e.g., "don't" to "do not").

**7. Bag of Words (BoW)**

* Bag-of-words models represent documents as unordered collections of tokens or words, akin to a set that also tracks the frequency of each element. These models disregard the sequence or order of words within a document, which means they treat "dog bites man" and "man bites dog" equivalently.
* Despite this limitation in capturing word order, bag-of-words models are popular for efficiency reasons in large-scale information retrieval tasks like search engines. They can achieve near state-of-the-art performance especially with longer documents, making them valuable tools in text processing and analysis.
  1. **TF-IDF (Term Frequency-Inverse Document Frequency)**

Term Frequency-Inverse Document Frequency is a numerical statistic used in natural language processing and information retrieval to determine the importance of a word in a document relative to a collection of documents (corpus). Here's a concise explanation:

 **Term Frequency (TF)**: Measures how frequently a term appears in a document. It's calculated as the ratio of the number of times a term occurs in a document to the total number of terms in that document. TF increases with the number of occurrences of a term within a document.

 **Inverse Document Frequency (IDF)**: Measures how important a term is across the entire corpus. It's calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term. IDF decreases with the number of documents containing the term, thus giving more weight to less frequent terms.

 **TF-IDF**: Combines TF and IDF to assign weights to terms in a document. Terms with high TF-IDF are those that are frequent within a document but rare across the corpus, thus capturing their significance in the document.

**9. Word Embeddings**

* Word embeddings are numerical representations of words that capture semantic meanings based on their usage in text. They're learned from large datasets using techniques like neural networks, grouping similar words closer together in a vector space. These embeddings are used in NLP tasks to improve model performance by understanding relationships between words and their contexts.
* Techniques: Word2Vec, GloVe, FastText

**NLP Tools and Libraries**

Here are examples of some popular NLP libraries.

* **TensorFlow and PyTorch**: Leading deep learning toolkits in Python, supporting research and commercial applications with extensive prebuilt components and GPU acceleration.
* **AllenNLP**: High-level NLP library in PyTorch, offering components like chatbots with excellent documentation and ease of use.
* **HuggingFace**: Distributes numerous pretrained NLP models and provides a toolkit for quick evaluation in TensorFlow and PyTorch, facilitating rapid model testing and deployment.
* **Spark NLP**: Open-source library supporting advanced NLP in Python, Java, and Scala, featuring pretrained models, pipelines, and custom model training capabilities.
* **SpaCy**: Free, open-source NLP library designed for efficient text processing and understanding in Python, known for its user-friendly interface and scalability.
* NLTK (Natural Language Toolkit): Provides easy-to-use interfaces to over 50 corpora and lexical resources, and a suite of text processing libraries.

# **Large language model**

A large language model (LLM) is a computational model renowned for its proficiency in general-purpose language generation and various natural language processing tasks, such as classification. These models learn statistical relationships from extensive amounts of text through a computationally intensive self-supervised and semi-supervised training process. LLMs, based on language models, are capable of text generation, a type of generative AI, by taking an input text and repeatedly predicting the next token or word.

LLMs are artificial neural networks employing the transformer architecture, introduced in 2017. As of June 2024, the largest and most advanced LLMs utilize a decoder-only transformer-based architecture, which allows for efficient processing and generation of large-scale text data.

## **Dataset preprocessing**

### Tokenizier :

Probabilistic tokenization serves the purpose of converting text into numerical representations for machine learning algorithms, which operate on numerical data rather than raw text. The process involves several steps: first, defining a vocabulary from the text corpus; second, assigning unique integer indices to each vocabulary entry; and finally, associating embeddings with these integer indices. Techniques such as byte-pair encoding and WordPiece are commonly used for this purpose.

In addition to facilitating numerical processing, probabilistic tokenization also aids in compressing datasets. This is particularly important for Large Language Models (LLMs), which require inputs to be structured uniformly, typically as arrays without varying lengths (i.e., not jagged). To achieve this uniformity, shorter texts are often "padded" with placeholders until they match the length of the longest text in the dataset.

**Byte Pair Encoding (BPE)**

Byte Pair Encoding (BPE) is a data compression technique used primarily in natural language processing and particularly in tokenization tasks. Here’s an explanation of how BPE works:

1. **Initialization**: BPE starts with initializing a vocabulary that consists of all unique characters in the dataset or corpus. Initially, each character is treated as a token.
2. **Merging Process**: BPE iteratively merges the most frequent pair of consecutive tokens (or characters) into a single new token. This merging process is based on the frequency of pairs occurring together in the dataset.
3. **Building Vocabulary**: As BPE continues merging tokens based on frequency, it gradually builds a vocabulary of tokens. The size of this vocabulary is predetermined or set as a parameter.
4. **Termination**: BPE terminates when it reaches the desired vocabulary size or when further merging no longer improves the compression efficiency significantly.
5. **Tokenization**: Once the vocabulary is built, each word or piece of text in the dataset is tokenized by replacing sequences of characters with their corresponding tokens from the vocabulary.

### Dataset cleaning

When training Large Language Models (LLMs), datasets are typically cleaned by removing toxic passages, discarding low-quality data, and eliminating duplicate entries. This cleaning process boosts training efficiency and enhances performance in subsequent tasks. Interestingly, LLMs themselves can be employed to clean datasets intended for further training .

As LLM-generated content becomes more prevalent online, future data cleaning may involve filtering out such content. This is challenging because LLM-generated text can closely resemble human-written text, making it difficult to differentiate. However, ensuring this distinction is crucial as LLM-generated content may be of lower quality compared to human-authored text, potentially degrading the performance of models trained on it [30].

### Synthetic data

To train the largest language models, there's often a need for more linguistic data than what naturally occurs or for higher-quality data. When natural data falls short, synthetic data can be employed instead. For instance, Microsoft's Phi series of LLMs is trained using synthetic data resembling textbooks, generated by another LLM.