Table of Contents

[References 3](#_Toc179103080)

[7th Sep 4](#_Toc179103081)

[8th Sep 4](#_Toc179103082)

[Neural Network Architecture 4](#_Toc179103083)

[14th Sep 4](#_Toc179103084)

[Azure Account 4](#_Toc179103085)

[Service Models 4](#_Toc179103086)

[Infrastructure as a Service (IaaS) 4](#_Toc179103087)

[Platform as a Service (PaaS) 5](#_Toc179103088)

[Software as a Service (SaaS) 5](#_Toc179103089)

[15th Sep 5](#_Toc179103090)

[Sigmoid function 5](#_Toc179103091)

[Alternatives 6](#_Toc179103092)

[21st Sep 6](#_Toc179103093)

[22nd Sep 6](#_Toc179103094)

[Maximum (Singular) 6](#_Toc179103095)

[Maxima (Plural) 6](#_Toc179103096)

[Global Minima 6](#_Toc179103097)

[Local Minima 6](#_Toc179103098)

[28th September 6](#_Toc179103099)

[29th September 6](#_Toc179103100)

[BGD - Batch Gradient Descent 6](#_Toc179103101)

[SGD - Stochastic Gradient Descent 8](#_Toc179103102)

[BGD vs SGD 9](#_Toc179103103)

[BGD - Batch Gradient Descent 9](#_Toc179103104)

[5th October 10](#_Toc179103105)

[Perceptron or Single Layer Perceptron 10](#_Toc179103106)

# References

Lectured By

* [Dev](https://github.com/shikharkumar13)

GitHub Link

* [shikharkumar13 (Data Science With Shikhar) (github.com)](https://github.com/shikharkumar13)

Drive:

* [Data Science Files - Google Drive](https://drive.google.com/drive/folders/1jiKxJCI3Cnd_pgznxQ-690O8RyfmgiPV)
* <https://drive.google.com/drive/folders/16YcAr_oKbrtNjJkMhTSXVpldvy9m-p15>

# 29th Dec

## Long Short-Term Memory (LSTM)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) architecture designed to handle sequential data effectively by overcoming the limitations of traditional RNNs, particularly the vanishing gradient problem. LSTMs are extensively used in Generative AI for tasks that involve sequential or time-series data, such as text, audio, and video generation.

### Key Features of LSTM

**Memory Cells**:

LSTMs maintain a cell state that acts as a memory to store information over long sequences. This helps the network retain relevant context across long input sequences.

Gates: LSTMs use three types of gates to control the flow of information:

* Cell State
* Hidden State
* Forget Gate: Decides what information to discard from the cell state.
* Input Gate: Determines which information to add to the cell state.
* Output Gate: Controls what part of the cell state is output as the current hidden state.

### Role in Generative AI

LSTMs are widely used for generative tasks due to their ability to model sequential dependencies:

1. Text Generation:

* LSTMs can generate coherent and contextually relevant text by learning patterns and relationships in sequential data, such as sentences or paragraphs.
* Example: Given a seed text, an LSTM can predict and append the next word iteratively to generate a complete text.

1. Music and Audio Generation:

* In music generation, LSTMs can learn temporal patterns, melodies, and rhythms to create new music.
* In speech synthesis, they can generate realistic audio waveforms when combined with other models like WaveNet.

1. Image Captioning:

* LSTMs are used in conjunction with convolutional neural networks (CNNs) to generate descriptive captions for images.

1. Video Generation:

* They can model temporal dependencies in video frames to generate coherent video sequences.

### Advantages in Generative AI

* Handles Sequential Data: LSTMs excel in learning dependencies across long sequences.
* Temporal Memory: They retain relevant context over time, crucial for tasks where past inputs influence current outputs.
* Flexibility: They can adapt to various generative tasks with appropriate training and architecture modifications.

Although LSTMs are powerful, they have been increasingly replaced or supplemented by Transformer-based architectures (e.g., GPT, BERT) in many Generative AI applications due to the latter's scalability and ability to process sequences in parallel. However, LSTMs remain a foundational concept in understanding sequence modeling.

# 4th Jan 2025

## Transformers

Transformers are a foundational architecture in Generative AI (GenAI) and many modern machine learning models. They were introduced in the seminal paper **"Attention is All You Need"** by Vaswani et al. (2017) and have since revolutionized the field of Natural Language Processing (NLP) and other domains like computer vision and audio processing. Here's a breakdown of what transformers are and how they function in GenAI:

**What Are Transformers?**

Transformers are deep learning models designed to process sequences of data, such as text, images, or time series, by focusing on the relationships between different parts of the sequence. They rely on a mechanism called **self-attention** to determine which parts of the input sequence are most important for understanding a given context.

**Key Components of Transformers**

1. **Self-Attention Mechanism**
   * Determines the relevance of different words (or tokens) in a sequence relative to each other.
   * Computes **attention scores** to weigh the importance of each token when processing input data.
2. **Positional Encoding**
   * Adds information about the position of tokens in a sequence since transformers process the input data in parallel (not sequentially).
3. **Multi-Head Attention**
   * Allows the model to capture different types of relationships between tokens by computing multiple attention scores in parallel.
4. **Feedforward Neural Networks**
   * Processes the outputs of the attention mechanism, adding depth and non-linearity to the model.
5. **Encoder-Decoder Architecture**
   * **Encoder**: Processes the input sequence into a contextual representation.
   * **Decoder**: Uses the encoder's output and generates a new sequence (e.g., translating text or generating a response).
6. **Layer Normalization and Residual Connections**
   * Help stabilize training and improve gradient flow.

**Transformers in Generative AI**

In GenAI, transformers are used to create models capable of generating new content, such as text, images, or code. These models often rely on large-scale training and fine-tuning to achieve state-of-the-art performance. Examples include:

1. **Language Models**
   * **GPT (Generative Pre-trained Transformer)**: Generates coherent and contextually relevant text (e.g., ChatGPT, GPT-4).
   * **BERT (Bidirectional Encoder Representations from Transformers)**: Excels at understanding the context but is typically not generative.
2. **Text-to-Image Models**
   * Use transformers to generate images from text prompts (e.g., DALL-E, Stable Diffusion).
3. **Audio Generation**
   * Models like Whisper or AudioLM leverage transformers for tasks like speech-to-text and music generation.
4. **Code Generation**
   * Models like Codex (used in GitHub Copilot) use transformers to assist in writing and debugging code.

**Advantages of Transformers**

* **Scalability**: Can handle very large datasets and sequences.
* **Parallelization**: Processes data in parallel, enabling faster training compared to recurrent models like RNNs or LSTMs.
* **Contextual Understanding**: Captures complex relationships and dependencies in data sequences.

**Why Are Transformers Important in GenAI?**

* They form the backbone of state-of-the-art AI systems in various domains.
* Their ability to model context and relationships at scale has enabled breakthroughs in generating high-quality and creative outputs.

If you'd like, I can dive deeper into specific aspects like attention mechanisms, training processes, or applications in GenAI. Let me know!

## Embeddings

In natural language processing and machine learning, embeddings are a way to represent words, phrases, or even entire documents as vectors in a continuous vector space. This makes it easier for machine learning models to understand and work with textual data. Here are some common types of embeddings:

1. **Word Embeddings**: These are vectors representing individual words. Examples include:
   * **Word2Vec**: Developed by Google, it represents words based on their context in a sentence using a neural network model.
   * **GloVe (Global Vectors for Word Representation)**: Created by Stanford, it captures the global statistical information of a corpus by aggregating word co-occurrence statistics.
2. **Character Embeddings**: Instead of representing entire words, character embeddings represent individual characters within words. This is useful for handling misspellings, rare words, or morphological variations.
3. **Sentence Embeddings**: These vectors represent entire sentences rather than individual words. Examples include:
   * **InferSent**: Developed by Facebook, it uses supervised learning to capture the meaning of sentences.
   * **Universal Sentence Encoder**: Created by Google, it encodes sentences into high-dimensional vectors using a transformer architecture.
4. **Document Embeddings**: These represent entire documents or paragraphs. Examples include:
   * **Doc2Vec**: An extension of Word2Vec that can generate embeddings for paragraphs or entire documents.
5. **Contextual Embeddings**: These embeddings capture the context in which a word is used. Examples include:
   * **ELMo (Embeddings from Language Models)**: Developed by Allen Institute, it generates word representations that are functions of the entire input sentence.
   * **BERT (Bidirectional Encoder Representations from Transformers)**: Created by Google, it provides deep contextualized word representations by considering both the left and right context in all layers.
   * **GPT (Generative Pre-trained Transformer)**: Also from OpenAI, it focuses on generating text based on the given input context.

Each type of embedding has its strengths and is used based on the specific requirements of the task at hand. The field continues to evolve, with new and more sophisticated embeddings being developed regularly.

# 11th Jan

## Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is an advanced technique in generative AI that enhances the capabilities of large language models (LLMs) by incorporating an information retrieval system Here's a breakdown of how it works:

1. **Combining LLMs with Retrieval Systems**: RAG integrates an information retrieval system with an LLM. When a query is made, the retrieval system fetches relevant data from a database or other sources.
2. **Providing Up-to-date and Contextual Information**: Traditional LLMs are trained on static datasets and might not have the most current information. RAG helps by providing real-time, contextually relevant data, making the responses more accurate and timelier.
3. **Improving Domain-Specific Knowledge**: RAG allows LLMs to access domain-specific data that they might not have been trained on, such as company-specific information or recent events.
4. **Enhancing Transparency and Auditability**: By using RAG, generative AI systems can cite their sources, making it easier to understand where the information came from and improving the transparency of the responses .

In essence, RAG helps generative AI applications provide more accurate, relevant, and up-to-date responses by augmenting the capabilities of LLMs with additional data sources

## Lang Chain

LangChain is an open-source framework designed to help developers build, deploy, and manage applications powered by large language models (LLMs). It provides a flexible and composable set of tools and libraries that simplify every stage of the LLM application lifecycle. Here are some key features of LangChain:

1. **Development**: LangChain offers open-source components and third-party integrations to build applications using LLMs. It includes tools for creating stateful agents with support for streaming and human-in-the-loop interactions.
2. **Productionization**: LangChain includes LangSmith, a tool for inspecting, monitoring, and evaluating LLM applications to optimize and deploy them with confidence.
3. **Deployment**: LangChain's LangGraph Platform allows developers to turn their applications into production-ready APIs and assistants.
4. **Integration**: LangChain integrates with various providers and technologies, such as embedding models and vector stores, making it easy to incorporate different data sources and methods.

LangChain is designed to be flexible and vendor-agnostic, allowing developers to future-proof their applications by making vendor optionality part of their LLM infrastructure design.

# 12th Jan

One hands on session on building a ChatGPT project