Essential Terminology for GenAI and LLM

Natural Language Processing (NLP): It is a field of artificial intelligence focused on the interaction between computers and human language, aiming to enable machines to understand, interpret, and generate human language. It encompasses a wide range of tasks such as text analysis, translation, sentiment analysis, and speech recognition.

Natural Language Understanding (NLU): The capability of an AI system to understand and interpret human language in a meaningful way.

Natural Language Generation (NLG): The process of generating coherent and contextually relevant text from data.

Generative Adversarial Network (GAN): A framework involving two neural networks, a generator and a discriminator, competing against each other to generate realistic data.

Variational Autoencoder (VAE): A generative model that learns the latent representation of input data and can generate new samples by sampling from this latent space.

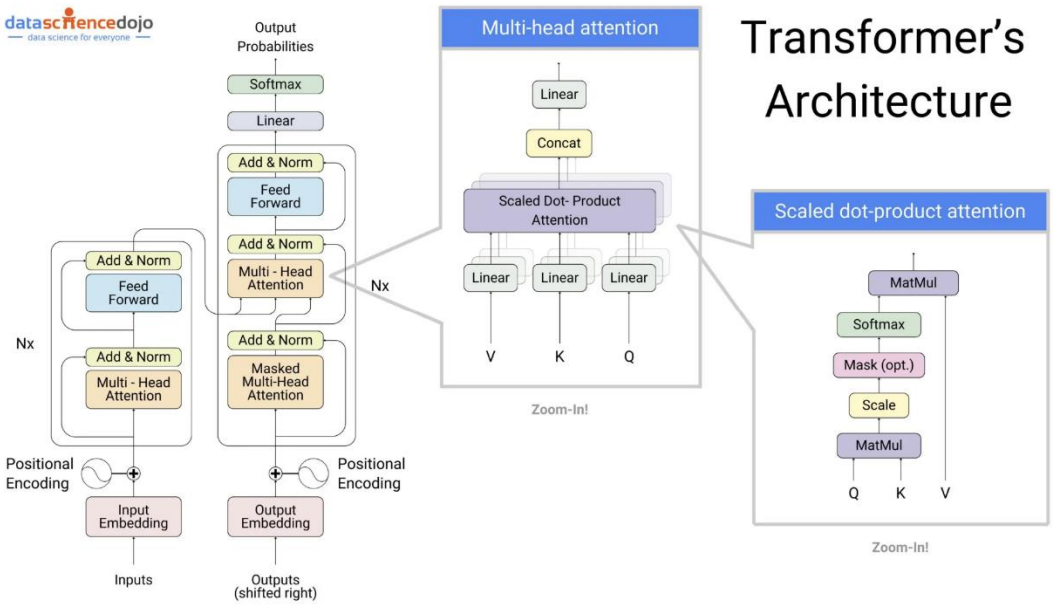
Diffusion Models: Generative models that start from noise and progressively generate structured data through a reverse diffusion process.

Text-to-Image Generation: The task of generating images from textual descriptions using models like DALL-E or CLIP, which combine language and vision understanding.

Recurrent Neural Networks (RNNs) are a type of neural network architecture designed to process sequences of data by maintaining a hidden state that captures temporal dependencies.

Long Short-Term Memory (LSTM) is a specific variant of RNNs that addresses the vanishing gradient problem and enables the model to capture long-term dependencies by introducing specialized memory cells and gating mechanisms.

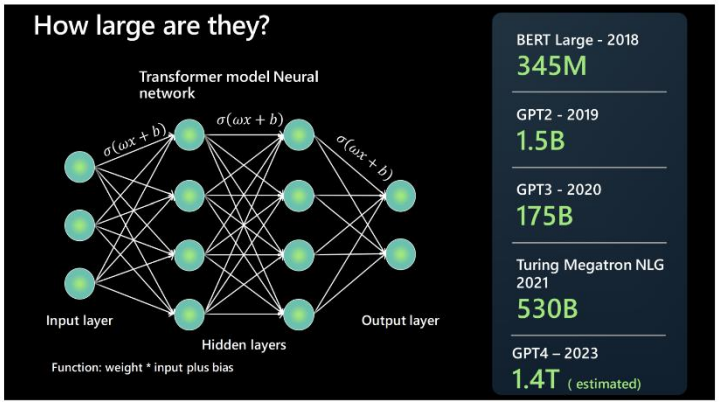
Transformer: A neural network architecture that uses self-attention mechanisms to process input data efficiently, foundational to modern LLMs.



The general architecture of transformer models

Source: https://datasciencedojo.com/blog/transformer-models-types-their-uses/

Neural Network in LLM: A neural network in the context of Large Language Models (LLMs) refers to a sophisticated machine learning architecture designed to process and generate human language. Here are key aspects:



Source: https://microsoft.github.io/Workshop-Interact-with-OpenAI-models/llms/

Useful Blog: https://www.labellerr.com/blog/evolution-of-neural-networks-to-large-language-models/

- **Architecture:** The neural network in LLMs is typically based on the Transformer architecture, which uses layers of self-attention and feed-forward neural networks to model complex language patterns.

- **Layers:** The Transformer architecture consists of multiple stacked layers, each comprising a self-attention mechanism and a feed-forward network. The

depth and complexity of these layers enable the model to capture intricate dependencies and relationships in text data.

- **Self-Attention Mechanism:** This mechanism allows the model to weigh the importance of different words in a sequence relative to each other, regardless of their position. It helps in understanding context and long-range dependencies.

- **Multi-Head Attention:** An extension of self-attention that allows the model to jointly attend to information from different representation subspaces.

- **Attention Mask:** A mechanism to prevent attention to certain parts of the input, such as padding tokens, ensuring the model focuses only on relevant tokens.

- **Feed-Forward Networks:** After the self-attention mechanism, data passes through fully connected layers that apply non-linear transformations, helping the model learn complex representations.

- **Embedding Layer:** This layer converts input tokens (words, subwords, or characters) into dense vectors of fixed size, capturing semantic information about the tokens.
- **Positional Encoding:** Transformers use positional encodings to inject information about the position of tokens in the sequence, as the architecture does not inherently understand sequence order.

- **Training:** Neural networks in LLMs are pre-trained on vast amounts of text data using unsupervised learning objectives like Masked Language Modeling (MLM) or Autoregressive Language Modeling (ALM). They are then fine-tuned on specific tasks using supervised or semi-supervised learning techniques.

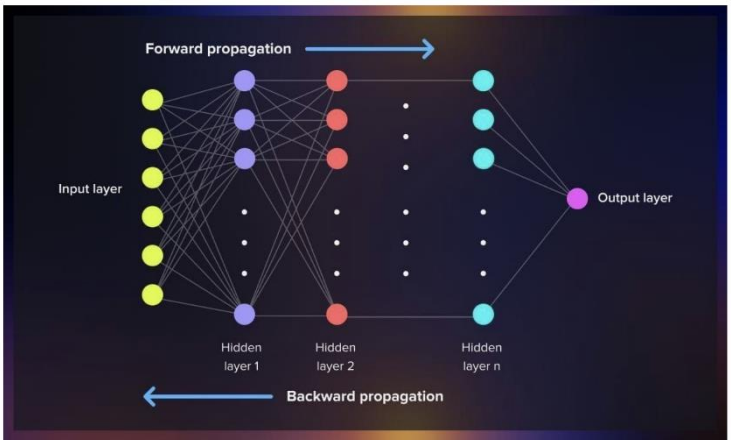
- **Attention Heads:** In multi-head attention, the model uses multiple attention mechanisms in parallel, allowing it to focus on different parts of the input sequence simultaneously, capturing various aspects of the data.

- **Optimization:** The training process involves optimizing the model's parameters using gradient descent-based algorithms, aiming to minimize a loss function that measures the difference between predicted and actual outcomes.

- **Generalization:** Well-designed neural networks in LLMs generalize from training data to perform well on unseen data, enabling them to generate

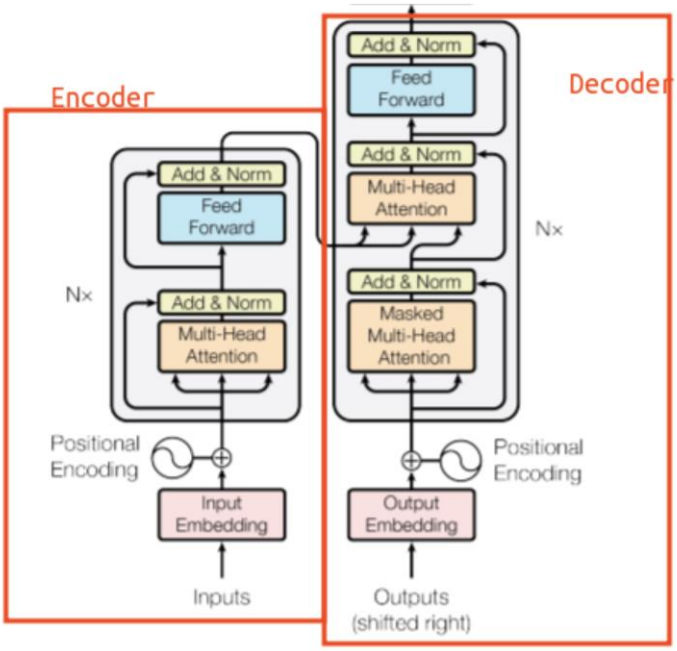
coherent and contextually relevant text, answer questions, translate languages, and more.

Backpropagation: The process of updating the weights of a neural network by propagating the error gradient backward through the network.



Source: https://serokell.io/blog/understanding-backpropagation

Parameter Sharing: Reusing parameters across different parts of a model, which reduces the total number of parameters and can improve generalization.



Transformer encoder-decoder model diagram (Attention is all you need)

Source: https://vaclavkosar.com/ml/Encoder-only-Decoder-only-vs-Encoder-Decoder-Transfomer

Encoder-Only Architecture: An LLM architecture where only the encoder part of the transformer is used, such as in BERT models.

Decoder-Only Architecture: An LLM architecture where only the decoder part of the transformer is used, such as in GPT models.

Encoder-Decoder Architecture: A neural network design where the encoder processes the input into a fixed representation, and the decoder generates the output from this representation.

Autoregressive Model (Usually Decoder only Model): A model that generates sequences one element at a time, predicting the next element based on previously generated elements, used in models like GPT.

Masked Language Modeling (MLM): A pre-training objective where some tokens in the input are masked, and the model learns to predict these masked tokens, used in models like BERT.

Activation Function: A mathematical function applied to a neural network's nodes to introduce non-linearity, allowing the network to learn complex patterns.

Zero-Shot Learning: The ability of an LLM to perform a task without having seen any specific examples of that task during training, relying instead on its general understanding.

Few-Shot Learning: The ability of an LLM to learn and perform a task after being provided with only a few examples.

Bidirectional Encoding: A technique where the context from both directions of a sequence is considered, enabling better understanding of the context, used in models like BERT.

Transfer Learning: Leveraging pre-trained models on one task and applying them to another related task, reducing the need for large task-specific datasets.

Inference: The process of using a trained model to make predictions on new data.

Tokenization: The process of converting text into a sequence of tokens, which are the smallest units (like words or subwords) that the model processes.

Subword Tokenization: A method of tokenization that splits words into smaller units like prefixes, suffixes, or characters, used to handle rare and out-of-vocabulary words.

Dropout: A regularization technique where randomly selected neurons are ignored during training, which helps prevent overfitting.

Beam Search: A search algorithm used in sequence generation tasks to find the most likely sequence of tokens by maintaining a fixed number of top candidates at each step.

Perplexity: A metric used to evaluate language models, measuring how well a model predicts a sample. Lower perplexity indicates better performance.

Gradient Clipping: A technique to prevent the gradients from exploding (becoming too large) by capping them at a maximum value during training.

Model Compression: Techniques like pruning, quantization, or distillation used to reduce the size of a model while maintaining its performance.

Model Pruning: The technique of reducing the size of a neural network by removing less important connections, which can improve efficiency and reduce overfitting.

Quantization: The process of reducing the precision of the model's weights and activations, which can lead to faster inference and lower memory usage.

LLM Parameters

Useful Blogs:

- https://cohere.com/blog/llm-parameters-best-outputs-language-ai
- https://www.promptingguide.ai/introduction/settings

Parameter: A model's internal variables that are adjusted during training, such as weights and biases in neural networks.

Sequence Length: The number of tokens in an input sequence that the model processes.

Temperature: Controls the randomness of predictions by scaling the logits before applying the softmax function. Lower values make the model's output more deterministic, while higher values increase diversity.

Context Window (Context Length): The maximum number of tokens the model can consider as context for generating the next token. Larger context windows allow the model to leverage more information from previous tokens.

Top-k Sampling: Limits the sampling pool to the top k most probable tokens at each step, ensuring only the most likely tokens are considered.

Top-p Sampling (Nucleus Sampling): Samples from the smallest set of tokens whose cumulative probability exceeds p, balancing the trade-off between diversity and coherence.

Repetition Penalty: Penalizes the model for generating repetitive sequences by modifying the probability of previously generated tokens.

LLM Fine Tuning

Useful Blogs:

- https://www.turing.com/resources/finetuning-large-language-models
- https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning

Fine-Tuning: The process of training a pre-trained LLM on a specific task or dataset to adapt it to particular requirements or applications.

Loss Function: A function that measures the difference between the predicted output and the actual target, guiding the optimization process.

Fine-Tuning: The process of taking a pre-trained language model and adapting it to a specific task or dataset by continuing training on task-specific data.

Pre-trained Model: A model that has been initially trained on a large, general-purpose dataset to learn general language features before being fine-tuned for specific tasks.

Transfer Learning: Utilizing a pre-trained model on one task and adapting it to another related task, which often improves performance and reduces training time.

Domain Adaptation: Adjusting a pre-trained model to work effectively in a specific domain (e.g. medical, legal, or financial text) by fine-tuning on domain-specific data.

Task-Specific Data: Data that is relevant to the particular task for which the model is being fine-tuned, such as labeled datasets for classification, translation, or summarization.

Supervised Fine-Tuning: Fine-tuning a model using labeled data where the correct output is provided, and the model learns to map inputs to the correct outputs.

Unsupervised Fine-Tuning: Fine-tuning a model using unlabeled data, often by employing self-supervised learning techniques where the model generates its own labels from the data.

Hyperparameter Tuning: The process of optimizing hyperparameters, such as learning rate and batch size, to improve the performance of the fine-tuned model.

Learning Rate: A hyperparameter that determines the step size at each iteration while moving toward a minimum of the loss function during training.

Learning Rate Schedule: A strategy to adjust the learning rate during fine-tuning, often decreasing it as training progresses to stabilize and fine-tune the model.

Batch Size: The number of training examples used in one iteration of model training, affecting the stability and convergence speed of the fine-tuning process.

Epoch: One complete pass through the entire training dataset during the fine-tuning process.

Gradient Descent: An optimization algorithm used to minimize the loss function by iteratively adjusting the model's parameters during fine-tuning.

Adam Optimizer: A popular optimization algorithm combining the advantages of two other extensions of stochastic gradient descent: AdaGrad and RMSProp.

Early Stopping: A technique to halt training when the model's performance on a validation set stops improving, which helps prevent overfitting.

Overfitting: A scenario where the fine-tuned model performs well on the training data but poorly on unseen data, capturing noise and details specific to the training set.

Underfitting: A scenario where the fine-tuned model is too simple to capture the underlying patterns in the data, resulting in poor performance on both training and validation data.

Validation Set: A subset of the data used to monitor the model's performance during fine-tuning, helping to tune hyperparameters and prevent overfitting.

Test Set: A separate subset of the data used to evaluate the final performance of the fine-tuned model on unseen data.

Dropout: A regularization technique used during fine-tuning where randomly selected neurons are ignored during training to prevent overfitting.

Regularization: Techniques used to prevent overfitting by adding constraints or penalties to the model's learning process, such as L2 regularization or dropout.

Fine-Tuning Objective: The specific loss function or metric that the model is optimizing during the fine-tuning process, aligned with the task requirements.

Data Augmentation: Techniques to artificially increase the size of the training dataset by creating modified versions of existing data, which can improve model generalization.

Model Generalization: The ability of the fine-tuned model to perform well on unseen data, not just the data it was trained on.

Checkpointing: Saving the state of the model at various points during fine-tuning, allowing for recovery in case of interruptions and enabling selection of the best-performing model.

Layer Freezing: A technique where certain layers of the pre-trained model are frozen (not updated) during fine-tuning, typically to retain learned features and reduce training time.

Feature Extraction: Using a pre-trained model to extract features from data, which can then be used as input to another model for a specific task.

Catastrophic Forgetting: A problem where a fine-tuned model loses information learned during pre-training, often mitigated by techniques like gradual unfreezing.

Gradient Accumulation: A technique where gradients are accumulated over several batches before updating the model parameters, allowing for effective training with larger effective batch sizes.

Model Distillation: A process where a smaller model (student) is trained to replicate the behaviour of a larger, fine-tuned model (teacher), often used for model compression.

Fine-Tuning Schedule: A plan or strategy for how fine-tuning will proceed, including decisions on learning rates, batch sizes, and epochs.

Evaluation Metrics: Metrics used to assess the performance of the fine-tuned model, such as accuracy, F1 score, precision, recall, and, ROUGH score for Summarization and BLEU score for translation tasks.

Multi-Task Learning: Training a model on multiple tasks simultaneously, which can improve performance on individual tasks by leveraging shared representations.

Cross-Validation: A technique to evaluate the model's performance by dividing the data into multiple subsets and training/testing the model on different combinations of these subsets.

Ensemble Methods: Combining predictions from multiple fine-tuned models to improve overall performance and robustness.

Contextual Adaptation: Adjusting the fine-tuned model to better handle the specific context in which it will be deployed, ensuring relevance and accuracy.

Inference Speed: The time it takes for the fine-tuned model to make predictions on new data, an important factor for real-time applications.

Parameter Efficiency: The effectiveness of the model in utilizing its parameters during fine-tuning, which can impact memory usage and inference speed.

Scalability: The ability to efficiently scale the fine-tuning process across multiple GPUs or machines, enabling the handling of large models and datasets.

Continual Learning: The process of continually fine-tuning a model as new data becomes available, helping the model to adapt to changing patterns and trends.

RLHF (Reinforcement Learning from Human Feedback): It is a technique used to fine-tune AI models by leveraging feedback from human evaluators to guide the learning process. This approach combines reinforcement learning with human judgment to align model outputs with desired behaviors and improve the overall quality of the generated responses.

LoRA (Low-Rank Adaptation): LoRA is a technique used to reduce the number of trainable parameters during the fine-tuning of large language models. It achieves this by decomposing weight matrices into lower-rank matrices, significantly reducing computational requirements while maintaining performance.

QLoRA (Quantized Low-Rank Adaptation): QLoRA combines quantization with LoRA to further reduce the memory and computational footprint of large language models. By applying both low-rank adaptations and quantization techniques, QLoRA enhances the efficiency of fine-tuning and inference.

bfloat16 (Brain Floating Point): bfloat16 is a 16-bit floating-point representation designed by Google. It retains the same exponent size as 32-bit floating-point (float32) but reduces the precision of the significand, allowing for faster computations and reduced memory usage without significantly impacting model accuracy.

nf4 (Normalized Float 4): nf4 is a 4-bit floating-point format that normalizes values to reduce the range and precision while still enabling efficient computations. It's often used in scenarios where extreme compression is required, balancing performance with resource constraints.

Related to LoRA and QLoRA

Useful Blogs:

- https://www.mercity.ai/blog-post/guide-to-fine-tuning-llms-with-lora-and-qlora
- https://cloud.google.com/vertex-ai/generative-ai/docs/model-garden/lora-qlora

Matrix Factorization: A technique that decomposes a matrix into a product of two or more smaller matrices, used in LoRA to reduce the number of trainable parameters.

Rank: The number of linearly independent rows or columns in a matrix; in the context of LoRA, lower rank implies fewer parameters and simpler models.

Adapter Layers: Additional layers added to a pre-trained model to facilitate fine-tuning on new tasks without altering the original model parameters extensively.

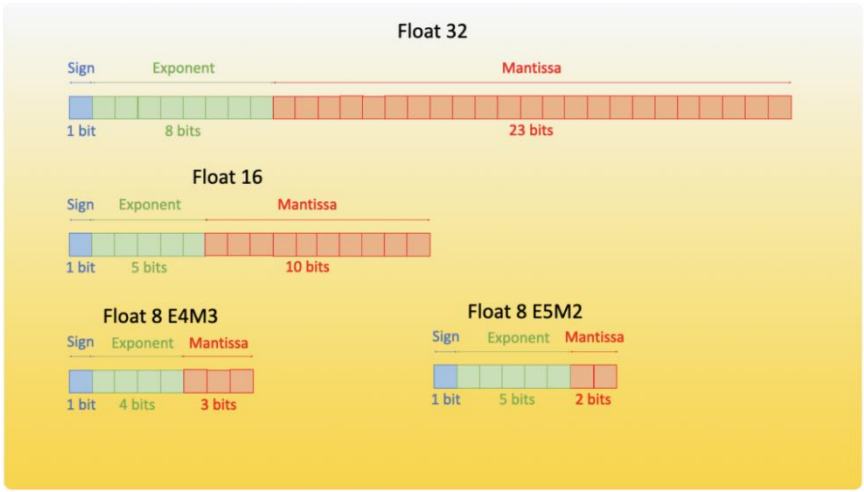
Parameter Efficiency: The effectiveness of a model in using its parameters to achieve high performance with fewer resources, a key goal of LoRA and QLoRA.

Model Compression: Techniques aimed at reducing the size of a model, which can include pruning, quantization, and low-rank adaptations like LoRA.

Quantization: The process of reducing the precision of the numbers used to represent a model's parameters, often used in QLoRA to enhance efficiency.

Mixed-Precision Training: A training approach that uses different numerical precisions (e.g., float32 and bfloat16) for different parts of the model to balance computational efficiency and model accuracy.

Related to bfloat16 and nf4



Overview of Floating Point 8 (FP8) format. Source: Original content from sgugger

Source: https://huggingface.co/blog/4bit-transformers-bitsandbytes

Floating Point Precision: The accuracy of a floating-point number representation, which impacts the performance and memory usage of neural networks.

Exponent and Mantissa: Components of floating-point representation; the exponent dictates the range, while the mantissa (or significand) dictates the precision.

Numerical Stability: The extent to which algorithms produce consistent results with varying numerical precision; bfloat16 is designed to maintain stability while reducing precision.

Half-Precision (float16): A 16-bit floating-point format with lower precision than float32, commonly used to reduce memory usage and increase computation speed.

Fixed-Point Arithmetic: An alternative to floating-point representation where numbers are represented by a fixed number of digits before and after the decimal point, used in some quantization schemes.

Quantization Aware Training (QAT): A training technique where quantization is simulated during the training process to prepare the model for post-training quantization.

Post-Training Quantization (PTQ): The process of applying quantization to a pre-trained model without retraining, useful for deploying models on resource-constrained devices.

General Optimization Techniques

Useful Blog: https://deepgram.com/learn/model-pruning-distillation-and-quantization-part-1

Distillation: A process where a smaller model (student) is trained to replicate the behaviour of a larger model (teacher), effectively compressing the model.

Pruning: Removing less significant weights from a neural network to reduce its size and improve computational efficiency.

Knowledge Transfer: The process of transferring learned knowledge from one model (often a larger, pre-trained model) to another (often smaller) model.

Sparse Representation: Using sparsity in neural networks to reduce the number of active parameters, enhancing efficiency without significantly impacting performance.

Weight Sharing: Reusing the same parameters across different parts of a neural network to reduce the number of unique parameters.

Activation Quantization: The process of reducing the precision of the activations (intermediate outputs) in a neural network, often combined with weight quantization.

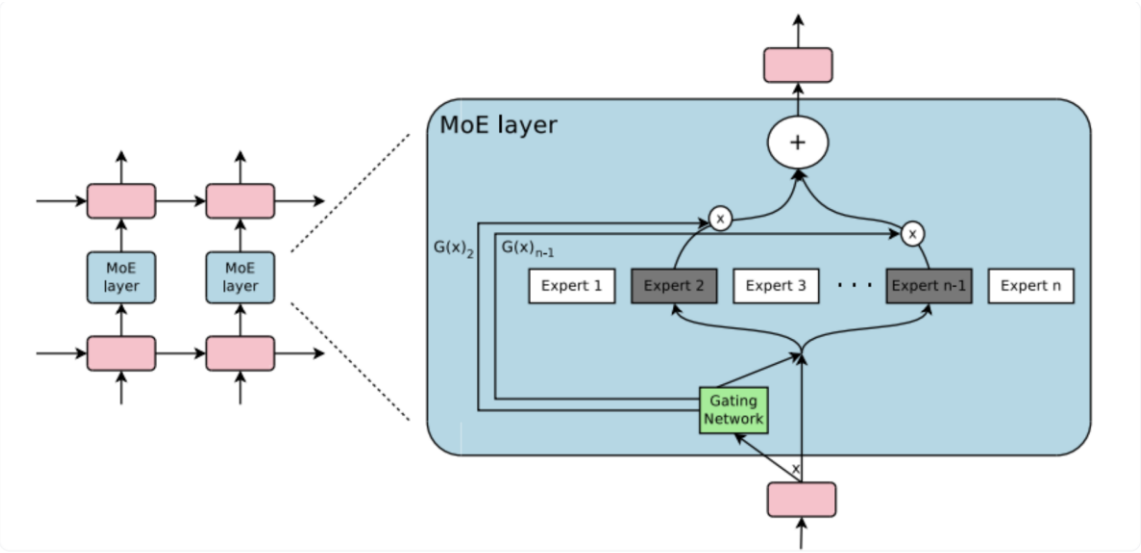
Retrieval Augmented Generation (RAG)

Useful Blog: https://www.promptingguide.ai/research/rag

Mixture of Experts (MoE)

Useful Blogs:

- https://huggingface.co/blog/moe
- https://developer.nvidia.com/blog/applying-mixture-of-experts-in-llm-architectures/



MoE layer from the Outrageously Large Neural Network paper

Source: https://huggingface.co/blog/moe

Mixture of Experts (MoE): It is a machine learning paradigm that involves multiple models (experts) and a gating mechanism to dynamically select which experts to use for each input.

- **Expert:** An individual model or neural network in the MoE framework that specializes in a particular subset of the data or task. Each expert processes the input independently and provides its output.

- **Gating Network**: A network that determines the weights or probabilities for each expert based on the input. It decides which experts to activate and how much influence each expert's output should have on the final decision.

- **Soft Gating:** A technique where the gating network assigns a soft probability distribution over the experts, allowing multiple experts to contribute to the final output in a weighted manner.

- **Hard Gating:** A technique where the gating network selects one or a few experts deterministically, typically based on the highest scores or probabilities, to contribute to the final output.

- **Sparse Gating:** A variation of MoE where only a small subset of experts is activated for any given input, reducing computational cost and improving efficiency.

- **Capacity Factor:** A hyperparameter that controls the number of experts selected or the proportion of the model capacity used during inference, influencing the trade-off between accuracy and computational efficiency.

- **Load Balancing:** Techniques to ensure that the workload is evenly distributed among experts, preventing some experts from being overutilized while others remain underutilized.

- **Dynamic Routing**: The process of dynamically selecting and routing inputs to different experts based on the gating network's decisions.

- **Conditional Computation:** A broader concept encompassing MoE, where parts of a model are conditionally activated based on the input, allowing for more efficient use of computational resources.

- **Ensemble Learning:** A related concept where multiple models (not necessarily in an MoE framework) are combined to improve performance, though MoE specifically uses a gating mechanism to select among experts.

- **Hierarchical MoE:** A structure where experts themselves can be MoE models, creating a multi-level hierarchy of experts for more complex decision-making processes.

- **Expert Training:** The phase during which individual experts are trained on different subsets or aspects of the data, often with the gating network being trained simultaneously or subsequently.

- **Expert Specialization:** The phenomenon where each expert in an MoE learns to specialize in particular types of data or aspects of the task, contributing to the overall model's performance.

- **Inference:** The phase during which the gating network selects the relevant experts based on new inputs and the model generates outputs using the chosen experts.

- **Scalability:** The ability of the MoE framework to handle increasing amounts of data or larger models by efficiently distributing the workload among multiple experts.

- **Generalization:** The capacity of the MoE framework to perform well on unseen data by leveraging the diverse specializations of different experts.

- **Computation Graph:** The representation of the operations performed by the MoE model, including the gating decisions and the contributions of selected experts.

- **Overfitting:** A risk in MoE models where individual experts might become too specialized to their training data, potentially reducing generalization to new data.

- **Regularization:** Techniques to prevent overfitting in MoE, such as dropout or weight decay, applied either to individual experts or the gating network.

- **Performance Metrics:** Criteria used to evaluate the effectiveness of an MoE model, including accuracy, computational efficiency, and load balancing across experts.