**International Journal of Artificial Intelligence & Applications (IJAIAP)**
Volume 02, Issue 01, Jan-Dec 2023, pp. 70-81. Article ID: IJAIAP_02_01_004
Available https://iaeme.com/Home/issue/IJAIAP?Volume=2&Issue=1
Journal ID: 4867-9994

© **IAEME** Publication SCOPE DATABASE INDEXED

OPEN ACCESS

# GENERATIVE AI AND LLM OPTIMIZING TECHNIQUES FOR DEVELOPING COST EFFECTIVE ENTERPRISE APPLICATIONS

**Amreth Chandrasehar**
Informatica, CA, USA

## ABSTRACT

*Generative AI usage has increased exponentially since start of the year and has created tremendous opportunities from startups to large enterprises. As more and more LLMs are released for research and commercial use, it becomes complex for enterprises to adopt the LLMs either using a managed service offering or even hosting it in-house as the cost is extremely high. This paper will focus on helping companies to optimize LLM, provide examples of use cases and solutions on fine tuning, cost optimizations, hosting LLM models internally in Kubernetes to solve data privacy, security and governance risks.*

**Keywords**: Generative AI, LLM, Enterprise Applications, AI, Hosting LLMs, LLM in Kubernetes, quantization, LLM optimization, pruning, Llama, Cost Optimization

**Cite this Article:** Amreth Chandrasehar, Generative AI and LLM Optimizing Techniques for Developing Cost Effective Enterprise Applications, International Journal of Artificial Intelligence & Applications (IJAIAP), 2(1), 2023, pp. 70-81.
https://iaeme.com/Home/issue/IJAIAP?Volume=2&Issue=1

## 1. INTRODUCTION

Generative AI refers to algorithms that create new content from a variety of inputs, such as code, text, images, soumds, animation, 3D, etc. GPTs (Generative pre-trained transformers) are created based on foundational models which are successors to Transformer models, a type of deep learning model that are commonly used in NLP and other applications of generative AI. Examples of foundational models are GPT-3 and Stable diffusion.

As enterprises start to use LLMs, they often run into large bills from managed service providers, sometimes more than their cloud cost. There are various ways this can be avoided by using optimizing the models and running it in-house. This will provide the data security, flexibility to choose any model, optimize the model and use as needed.

Running LLMs in-house can be challenging, expensive without the right optimization techniques. Over the next few sections, using the quantization techniques, 6x compression of GPU requirements is achieved and able to run Llama-2 models in just 4 GPUs.

## 2. GENERATIVE AI USE CASES

It has been shown that LLMs perform well in NL applications such as summarization, information extraction, language translation, Natural language to SQL and entity disambiguation.

Most of the enterprises will be using above features to build an application for their customer base. Below are some of the ideas that can be used by enterprises to develop applications:

- Help assist - Train LLMs on internal or external domain knowledge base to accurately respond to customers queries.
- Summarization of operational incidents to provide quick context of incident based on correlation of alerts and events.
- Creation of documentations like technical manuals for users
- Multilingual translation to help customer care representatives
- Multilingual sentiment analysis to help customer care reps/agents
- Develop interactive learning materials, quizzes, and simulations.
- Create realistic human-like voices for virtual assistants, audiobooks, etc
- Automated report writing, generating summaries, and answering user queries in customer support.
- Generate custom visual content for branding and advertising
- A personalized product recommendation system based on a customer's preferences, purchase history and behavior

## 3. POPULAR OPEN SOURCE LLM MODELS

To build Generative AI applications, enterprises will need to use LLMs. There are many open source LLMs available to run them inhouse or use preoperatory LLMs like Open AI’s GPT3, GPT3.5 and GPT4 models.

Hugging face leaderboard is a good start to see current list of top performing LLMs based on Massive Multitask Language Understanding (MMLU), HellaSwag (a challenge dataset for evaluating commonsense NLI), Abstraction and Reasoning Corpus (ARC) and TruthfulQA (test measures how well models mimic human

falsehoods). Below are the current popular open-source models as of Aug 2023.

- Llama-2 7B, 13B and 70B - META AI
- Falcon 40B, TII
- StableBeluga2 - Stability AI derivative of Llama-2)
- Alpaca – Stanford (derivative of Llama)
- Lazarus 30 B - Caldera AI

The leaderboard changes rapidly as new LLM models are introduced, while evaluating LLM for use cases at enterprise, it is important to check the leaderboard and take informed decisions.

To understand how the most popular Llama-2 model is built, below image provides information on pretraining, fine-tuning and human feedback to build the model.

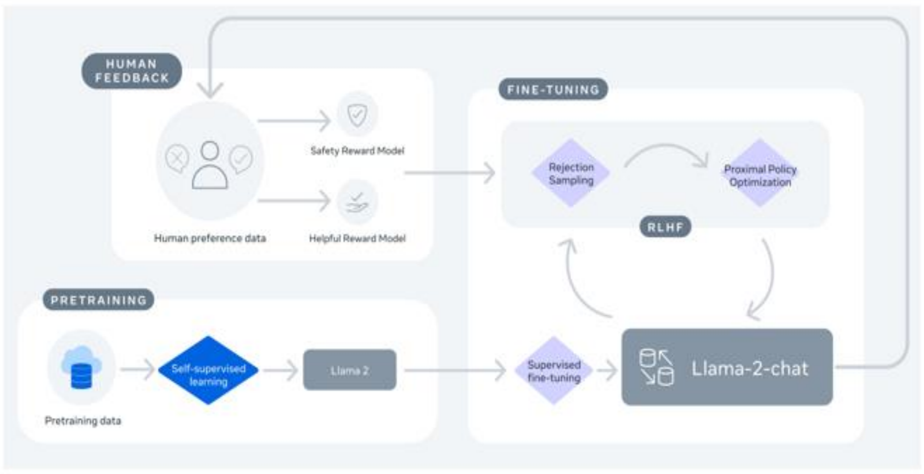


**Figure [1]** Training Llama-2 Chat [12]

Training Llama 2-Chat process begins with pretraining the model using publicly available online sources, supervised fine-tuning, RLHF method is used to iteratively refine the model, specifically through rejection sampling and PPO. Throughout the RLHF stage.

## 4. WHY USE IN-HOUSE LLM MODELS

Organizations can have limitations to use managed LLMs such as Open AI, Google Generative AI studio due to cost or security reasons. Below are some of the benefits of Hosting LLM models internally:

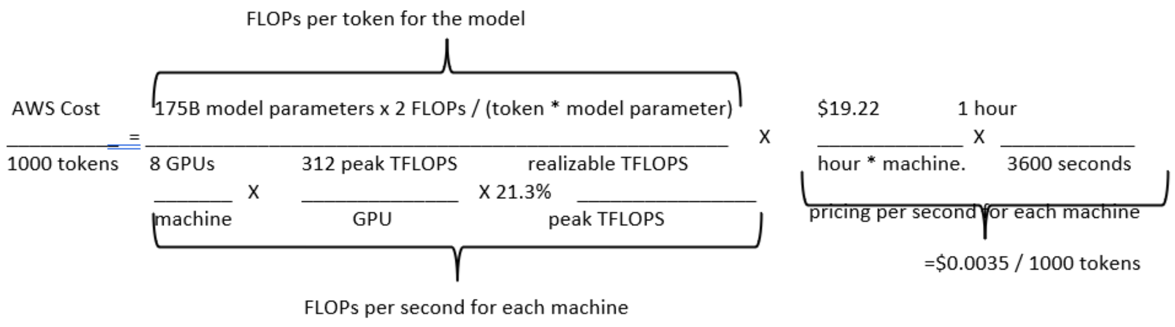
- Security
- Data Privacy
- Lower Cost
- Ability to deploy any region, any cloud
- Scalable, no constraint due to capacity with Cloud vendors

### Cost calculation of Open AI vs running LLMs inhouse:

### OpenAI Pricing (Aug 2023)

| Model | Context | Input/ 1K tokens | Output/ 1K tokens |
|---|---|---|---|
| GPT-4 | 8K | $0.03 | $0.06 |
| GPT-4 | 32K | $0.06 | $0.12 |
| GPT-3.5 | 4K | $0.0015 | $0.002 |
| GPT-3.5 | 16K | $0.003 | $0.004 |

OpenAI’s Davinci API has same parameter count as GPT 3.5 model. Cost of using this model costs ~$0.02 per 750 words ($0.02 per 1000 tokens/ ~750 words. With information available publicly, the cost comes to $0.010/query.  But usually, in application development 2-Stage Search Summarizer is what will be used, and this cost comes to around $0.066/query [14]


FLOPs per token for the model
AWS Cost
1000 tokens
175B model parameters x 2 FLOPs / (token * model parameter)
8 GPUs
machine
312 peak TFLOPS
GPU
X 21.3%
realizable TFLOPS
peak TFLOPS
FLOPs per second for each machine
$19.22
hour * machine.
1 hour
3600 seconds
pricing per second for each machine
=$0.0035 / 1000 tokens

The estimated cost of $0.0035 per 1000 tokens are ~20% of Open Ai’s API pricing of $0.02 per 1000 tokens, implying ~80% gross margins.

As you can see without fine tuning or optimizations, using LLM managed service offerings can be very expensive. Companies spend millions of dollars, sometimes more than the public cloud providers bill as many teams in the organizations start to use the models without any optimizations.

| **Model** | **Training**/ 1K tokens | **Usage**/ 1K tokens |
|---|---|---|
| Ada | $0.0004 | $0.0016 |
| Babbage | $0.0006 | $0.0024 |
| Curie | $0.0030 | $0.0120 |
| Davinci | $0.0300 | $0.1200 |

## Anthropic pricing for Claude and Claude 2 LLMs

| **Model** | **Context window** | **Prompt cost** | **Completion cost** |
|---|---|---|---|
| Claude Instant | 100,000 tokens | $1.63/ million tokens | $5.51/ million tokens |
| Claude 2 | 100,000 tokens | $11.02/ million tokens | $32.68/ million tokens |

With cloud provisioned instances AWS A100 (P4 Instance) or GCP TPU v4, provides 312 TFLOPS (teraFLOPs/second), the cost comes to $0.0035/1000 tokens. This is based on 21% model utilization (FLOPS), which is in-line with GPT-3.5’s during training. Thus, for a 175B parameter model like GPT-3: [14]

LLMs use a consumption pricing model based on Figure [2] AWS Cost calculation for hosting LLMs tokens (amount of text characters). Each model has a fixed “token window” for the context length used by the model for a given task. For example, Llama-2 can use 4k tokens to store conversation history of a chat session. Due to high cost due to unpredictable prompts, new cost-optimization techniques are being used by developers working with Large Language Models. Some of these cost-optimization techniques include, Fine tuning, summarization of chat history, prompt engineering, use of vector stores and chains. In coming sections of this paper, these techniques will be discussed in detail.

## 5. DEPLOYING LLM MODELS IN-HOUSE USING KUBERNETES

To demonstrate deploying LLM models in-house, Llama-2 7B and Llama-2 70B modes used. The Llama-2 is pretrained and fine-tuned LLMs consists of 7B, 13B, 70B models. The pretrained models are trained on 40% more tokens compared to Llama-1 and has a context length of 4k tokens.

Llama-2 models requires GPUs, the Kubernetes worker nodes should be provisioned with GPU AWS Cost/1000 token [13] nodes. In below diagram, AWS instance type p3.8xlarge is used to run the 70B model. This instance types provides 4 GPUs and using GGML quantization technique  this can be run with lesser GPUs than 46 GPUs. But, going further than just using 4-bits quantization, we were able to deploy with 16-bits. This helped to run with much less resources, 1 GPU using AWS p3. xlarge instance type, than using p3.8xlarge instance type with 8 GPUs

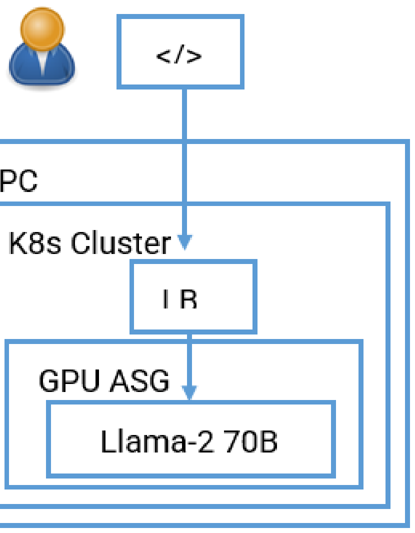


**Figure [3]** High level architecture of how Open-Source models can be deployed within internal K8s cluster

To gain access to Llama-2 modes, visit this site to request access and get the token. Once token is available, replace “ACCESSTOKENVALUEHERE” with the token provided. Then, use the code in this link to run the Llama-2 70 B model in Kubernetes cluster.

Once the Kubernetes pod is running, it can be made accessible outside the cluster using a proxy (nginx) or adding load balancer to the service or ssh into the pod to run below curl command to test the output

## Request:

curl --location 'http://<url?.elb.us-west-2.amazonaws.com:8080/v1/models/model:predict' --header 'Content-Type: application/json' --data '{"prompt": "What is mechanics?"}'

## Response:

{"data": {"generated_text": "What is mechanics?\nMechanics is a branch of physics which deals with motion of objects and the forces that act upon them. It is a fundamental subject that is essential for understanding the behavior of the physical world around us.\nMechanics is divided into two main branches: statics and dynamics. Statics deals with objects that are at rest or moving at a constant velocity, while dynamics deals with objects that are accelerating or changing their velocity.\nSome of the key concepts in mechanics include:\n* Forces: A force is a push or pull that acts on an object. Forces can be"}}

Now, we have successfully run Llama-70B model in Kubernetes using a single GPU node. This can be repeated to run for 70B, 13B models or even Falcon 40B model. The response time for the model is under 5 seconds, but this can further be improved with techniques discussed in below sections.

## 6. LLM MODEL COMPRESSION TECHNIQUES

Below are model compression techniques that can be used to reduce the cost of training and hosting LLMs.

- Pruning – Given an optimized model f, you must produce a compressed version f that maximizes the performance of the previous model. Pruning helps to remove redundant weights from pretrained models.

There are many types of pruning approaches available for LLMs such as BERT-Large, Wanda (Pruning by Weights and activations), LLM-Pruner and SparseGPT. Below we will be discussing 2 types of pruners - LLM-Pruner and SparseGPT. Each of these techniques have produced promising results.

LLM-Pruner: It accomplishes by iteratively examining each neuron within the model as a trigger for identifying dependency groups, thereby constructing the LLM's dependency graph. Subsequently, LLM-Pruner assesses the importance of these groups using both parameter-wise and weight-wise estimation. Utilize LoRA for fast recovery and adjustment of the pruned model. The experimental results indicate that LLM-Pruner successfully prunes the model, reducing computational burden while retaining its zero-shot capabilities. [4]

SparseGPT: SparseGPT is a post-training pruning method for compressing LLMs such as GPTs. It works by first identifying a set of important weights in the LLM by their contribution to the model's loss function. The unimportant weights are pruned, which helps to reduce the size of the model. SparseGPT has been shown to be effective in compressing LLMs while maintaining or even improving their accuracy. For example, SparseGPT was able to compress GPT-3 by 50% without any loss in accuracy.

- Fewer computations: LoRa stands for Low-Rank Adaptation of Large Language Models, which is a method to reduce the model size and computational requirements by approximating large matrices using low-rank decomposition. Faster fine-tuning, and you can share the LoRa weights only (orders of magnitude smaller than a fine-tuned model). Used often in Stable Diffusion. [11]

Low-Rank Adaptation (LoRA) is a technique for fine-tuning LLMs, which reduces the size and complexity of the model while maintaining or even improving its accuracy. LoRA works by first decomposing the LLM into a low-rank matrix and a sparse matrix. The sparse matrix captures less important features, but the low-rank matrix captures the most important features, then fine-tuned on a downstream task. This is done by adding a small number of parameters to the low-rank matrix. The sparse matrix is not fine-tuned. LoRA is effective in various tasks, such as text classification, question answering, and summarization. LoRA has been able to reduce the size of LLMs by up to 90% without any loss in accuracy.

- Quantization - reducing the precision of weights

    - Quantization is the process of converting a continuous signal or variable into a discrete set of values. This can be done for a few reasons, such as to reduce the amount of data that needs to be stored or transmitted, or to make the signal easier to process. In machine learning, quantization is often used to reduce the size (lower precision like 8-bit integers) and complexity of neural networks.

This can make the networks faster to train and deploy, and it can also make them more energy efficient.

Benefits of quantization includes reduced size, complexity, improved performance, increased portability and energy efficiency of neural networks, without impacting model accuracy when successful.

Challenges of quantization include Loss of accuracy, Increased complexity and limited support in ML frameworks

- GPTQ: GPTQ stands for Gaussian Process Quantized Transformers. It is a quantization technique for generative pre-trained transformers also known as GPTs. GPTQ uses a Gaussian process to approximate the distribution of the weights and then uses the approximation to quantize the weights. GPTQ has achieved good accuracy and performance on a various task, while also being significantly smaller and faster than its floating-point counterparts. GPTQ Uses a Gaussian process to approximate the distribution of the weights, uses a one-shot quantization method and can be used to quantize GPTs of all sizes
- GGML: It is a quantization technique focuses on CPU optimization and supports 4-bit and 8-bit quantization with different tradeoffs between efficiency and performance. It uses Gaussian techniques to approximate the gradient of the model and uses the approximation to quantize the weights of the model. Despite being significantly smaller, it faster, more accurate and performant than its floating-point counterparts. Few differences between GPTQ and GGML:
  - GPTQ uses a Gaussian process to approximate the distribution of the weights, while GGML approximates the gradient of the model. This makes GPTQ more accurate for tasks with small changes in the weights, while GGML is more accurate for large changes in the weights.
  - GPTQ uses a one-shot quantization method, whereas GGML uses either a one-shot or a multi-shot quantization method (more accurate, but slower than one-shot)
  - GPTQ can be used to quantize GPTs of all sizes, while GGML is primarily designed for GPTs of small to medium size, since GGML is more computationally expensive than GPTQ.

- Mixed precision — Using a combination of lower (e.g., float16) and higher (e.g., float32) precision arithmetic to balance performance and accuracy.
- Model ensembles — Combining the outputs of multiple smaller models, each specialized in a sub-task, to improve overall performance. Might use a similarity search on embeddings or some other heuristic to figure out what models to call.
- Knowledge distillation is a ML technique that transfers knowledge from a large, complex model to a smaller, simpler model. The larger model is trained on a large dataset and has learned to perform a task well, but the smaller model is trained on a smaller dataset and is not as accurate as the larger model. KD process involves training the smaller model to replicate the predictions of the larger model. This can be achieved by feeding the smaller model the outputs of the larger mode. It gradually becomes more accurate by feeding outputs of the larger model.

Knowledge distillation has been shown to be effective in a lot of machine learning tasks, like image classification, NL processing, and speech recognition. It can be used to reduce the size and complexity of machine learning models, while maintaining or even improving their accuracy. The benefits are reduced size and complexity, improved accuracy and transfer of knowledge.

## 7. FINE-TUNING LLMS

LLMs are pre-trained with extensive data gathered on various sources available publicly. The accuracy of the results will significantly drop if the requests are specific to a domain or a company's internal knowledge. Fine-tuning or embeddings of LLMs becomes important to ensure accuracy of the models are good to ensure customer requests are rightly responded. Below is a flow chart that can help to determine what strategy can be used while using LLMs in applications.

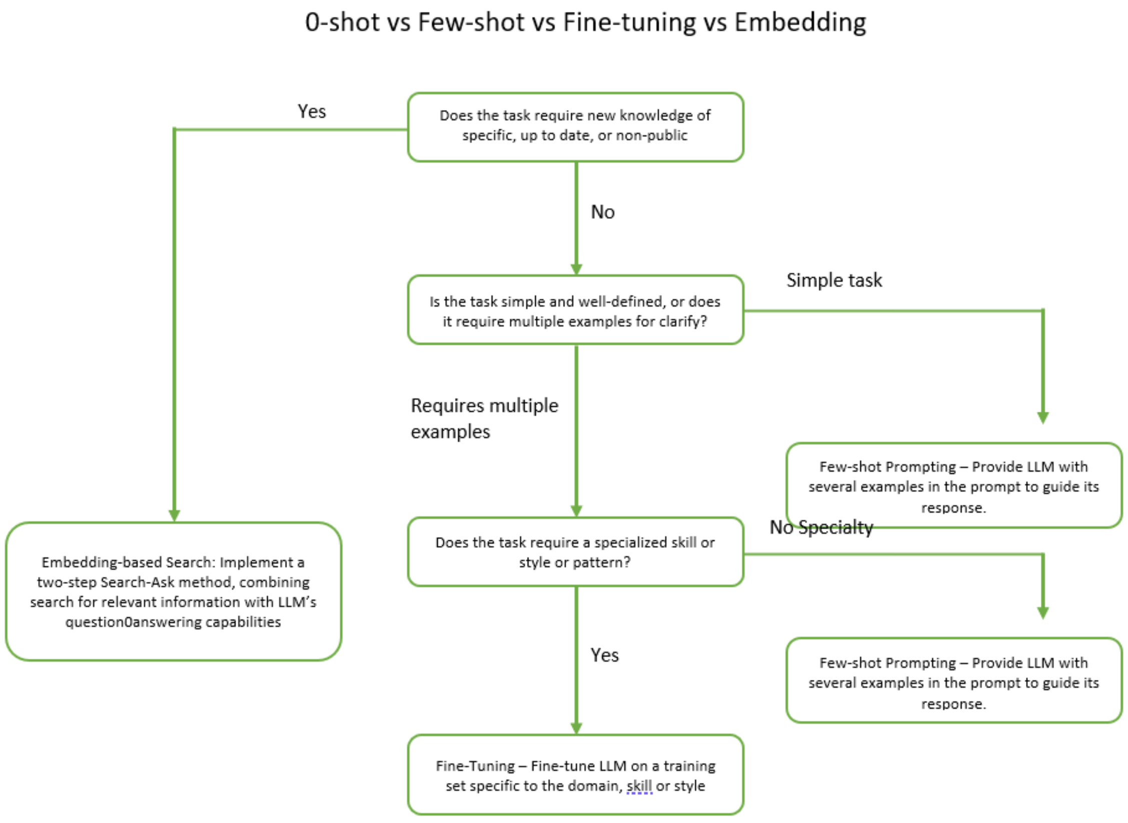

0-shot vs Few-shot vs Fine-tuning vs Embedding
Yes
Does the task require new knowledge of specific, up to date, or non-public
No
Is the task simple and well-defined, or does it require multiple examples for clarify?
Simple task
Requires multiple examples
Few-shot Prompting – Provide LLM with several examples in the prompt to guide its response.
No Specialty
Embedding-based Search: Implement a two-step Search-Ask method, combining search for relevant information with LLM's question0answering capabilities
Does the task require a specialized skill or style or pattern?
Yes
Few-shot Prompting – Provide LLM with several examples in the prompt to guide its response.
Fine-Tuning – Fine-tune LLM on a training set specific to the domain, skill or style

**Figure [4]** Choosing the right LLM Strategy [13]

Below are some of various fine-tuning methods that can be used to improve accuracy of LLM responses:

- Fine-tuning with PEFT - Parameter-Efficient Fine-Tuning (PEFT) is a library for efficiently fine-tuning LLMs without touching all of the LLM's parameters. PEFT supports the QLoRa method to fine-tune a small fraction of the LLM parameters with 4-bit quantization.

- QLoRA stands for Quantized Low Rank Adapters. It is a new approach to fine-tuning large language models that uses quantization and knowledge distillation

- techniques. QLoRA achieves good accuracy and performance on a variety of tasks, while being significantly smaller and faster than traditional fine-tuning methods. QLoRA is a promising new approach to fine-tuning LLMs that can be used to improve the performance, portability, and energy efficiency of LLMs. QLoRA Uses quantization, knowledge distillation and low rank adapters to reduce size and complexity, improve accuracy, robustness and transfer learning to LLMs. QLoRA is still under development, but it has the potential to revolutionize the way that LLMs are fine-tuned.

- Supervised fine-tuning (SFT) vs Unsupervised fine-tuning Unsupervised fine-tuning and Supervised fine-tuning (SFT) are two approaches to fine-tuning large language models (LLMs).

Supervised fine-tuning is the process of fine-tuning an LLM using a labeled dataset. The dataset contains pairs of inputs and outputs, and the output is the desired output for the input. The LLM is then fine-tuned to predict the outputs for the inputs in the datasets.

Unsupervised fine-tuning is a process of fine-tuning an LLM without using a labeled dataset. In this case, the dataset does not contain pairs of inputs and outputs compared to SFT. The LLM is then fine-tuned to predict the next word in a sequence, or to generate text that is like the text in the dataset.

Benefits of SFT is higher accuracy and can be used for transfer learning, whereas unsupervised fine-tuning benefits from robustness and efficiency.

- RLHF stands for Reinforcement learning from human feedback - It is a machine learning technique that combines reinforcement learning (RL) and human feedback. Popular example of RLHF is ChatGPT. In RL, an agent learns to perform a task by trial and error, receiving rewards for taking actions that lead to desired outcomes and punishments for taking actions that lead to undesired outcomes. In RLHF, the agent also receives feedback from humans, which can be used to improve the agent's learning process. Human reviewers are recruited to rate the output of the model on various prompts. The human feedback can be in the form of ratings, comments, or demonstrations. The agent uses the feedback to learn which actions are more likely to be rewarded by reviewers, and which actions are more likely to be punished. This will help the agent to learn more quickly and efficiently and avoid making mistakes leading to negative feedback. RLHF is effective in a variety of tasks, such as playing games, controlling robots, and generating text. It is a promising technique that can be used to train agents to perform a wide range of tasks.

## 8. WHEN NOT TO FINE-TUNE LLMS

Several challenges exist while fine-tuning a large language model such as GPTs affecting its efficiency, scalability, and effectiveness. Below are the main challenges associated with fine-tuning LLMs:

- Computational Costs - Fine-tuning can become very expensive with limited budgets
- Training Data Quality - Sourcing can be time consuming and expensive
- Overfitting, resulting in poor generalization to new examples
- Confabulation and Hallucination, leading to untrustworthy and unreliable responses
- Adaptability, when updated knowledge or new information becomes available
- Ethical Considerations, such as bias, misinformation or stereotypes

Below we discuss retrieval augmentation and embeddings as alternatives to fine tuning.

- In-context learning or retrieval augmentation – When data in the application might change frequently or not enough data is not available for finetuning or creating a personalized chatbot for each user, it can be used, when the model is provided with context during inference time. This is a better option than fine tuning LLMs which can produce inaccurate responses.

This can be achieved by using vector database which stores embeddings of company documents. When a user enters a prompt, relevant documents is retrieved from vector DB and the output is added as context to the model. A hybrid approaches can also be used, where fine-tuning the model on a specific dataset and respond with user-specific context as output. Below is the diagram depicting how to use vector databases with LLMs.

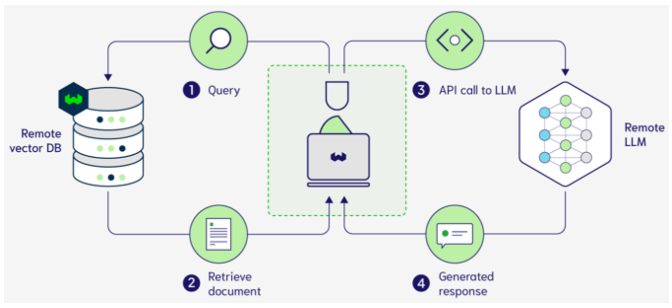


**Figure [5]** Retrieval Augmentation using Vector DB [8]

Step 1&2: Query remotely deployed vector database storing proprietary data to retrieve the documents relevant to the prompt.

Step3&4: Combine the returned documents with the prompt to the remote LLM; which it will then use to generate a custom response.

- Semantic Embeddings: Are a type of vector representation of words or phrases that captures the meaning. Semantic embeddings are used in neural network models to predict the surrounding words in a sentence, predicting co-occurrence of words in a text, improve the accuracy of translation systems, using embeddings to create a knowledge base, search efficient retrieval of relevant information and as question answering systems. Benefits of using Semantic embeddings are improved accuracy, improved interpretability and reduced dimensionality. The knowledge-based method is also a cost-effective approach as it removes the need for fine-tuning, cutting down on expenses and making AI adaptation more economically viable.

## 9. FUTURE WORK

There are more techniques using retrieval augmentation methods such as LangChain library, LlamaIndex (GPT Index) that can be used to improve the accuracy of the results. More work in this area needs to be done to evaluate these options to improve accuracy of model response.

Running transformer models on non-GPU VMs needs more research, as an example AWS offers Inferentia accelerator [24] as an alternate to GPUs, which can save infrastructure cost by 75% needs to be done once transformer models are supported in these instances. Additional optimization of running with 32-bits quantization can also be done to save more resources but need to evaluate on accuracy of the models.

## 10. CONCLUSION

Large language models (LLMs) are promising in various tasks, including NLP understanding, generation, and translation. However, important challenges such as bias present in the data that is being trained on and computationally expensive training cycles needs to be optimized before it is released to the real-world. Hence, it is important for the engineers, pm and architects to understand the use case, determine the right optimization techniques, fine-tune or not to fine tune. Building your own LLM is another option, but pre-training the model can be extremely costly, but it can be cost effective if domain based LLMs are created, like Salesforce's XGen-7B model.

LLMs have the potential to revolutionize, become even more powerful and versatile. LLMs can be used to create more natural and engaging user interfaces, and to provide us with access to information and services that were previously unavailable. As LLMs continue to develop and will play an increasingly important role in our lives and have the potential to make a positive contribution to society.

## REFERENCES

[1] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, Ji-Rong Wen, A Survey of Large Language Modelshttps://arxiv.org/abs/2303.18223

[2] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, QLoRA: Efficient Finetuning of Quantized LLMs, https://arxiv.org/abs/2305.14314, https://github.com/artidoro/qlora

[3] Suresh Bhojwani, Supercharging Language Models: Strategies for Optimizing LLM and GPT, https://medium.com/@sureshbhojwani001/supercharging-language-models-strategies-for-optimizing-llm-and-gpt-f5cd59e706ca

[4] Xinyin Ma, Gongfan Fang, Xinchao Wang, LLM-Pruner: On the Structural Pruning of Large Language Models, https://arxiv.org/abs/2305.11627

[5] Montana Low, Smaller is Better: Q8-Chat LLM is an Efficient Generative AI Experience on Intel® Xeon® Processors, https://www.intel.com/content/www/us/en/developer/articles/case-study/q8-chat-efficient-generative-ai-experience-xeon.html

[6] Announcing GPTQ & GGML Quantized LLM support for Huggingface Transformers, https://postgresml.org/blog/announcing-gptq-and-ggml-quantized-llm-support-for-huggingface-transformers

[7] Ben Dickson, The complete guide to LLM fine-tuning, https://bdtechtalks.com/2023/07/10/llm-fine-tuning/

[8] Zain Hasan, Running Large Language Models Privately - privateGPT and Beyond, https://weaviate.io/blog/private-llm

[9] Tomaz Bratanic, Knowledge Graphs & LLMs: Fine-Tuning vs. Retrieval-Augmented Generation, https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/

[10] Vantage Team, Optimizing Large Language Models for Cost Efficiency, https://www.vantage.sh/blog/optimize-large-language-model-costs

[11] Matt Rickard, A Hacker's Guide to LLM Optimization, https://matt-rickard.com/a-hackers-guide-to-llm-optimization

[12] Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, Yue Zhang , PandaLM: An Automatic Evaluation Benchmark for LLM Instruction Tuning Optimization https://arxiv.org/abs/2306.05087

[13] Sunil Ramlochan, Master Prompt Engineering: LLM Embedding and Fine-tuning, https://www.promptengineering.org/master-prompt-engineering-llm-embedding-and-fine-tuning/

[14] Sunyan, The Economics of Large Language Models, https://sunyan.substack.com/p/the-economics-of-large-language-models

[15] Sunal Swarkar, Understanding LLaMA-2 Architecture & its Ginormous Impact on GenAI, https://medium.com/towards-generative-ai/understanding-llama-2-architecture-its-ginormous-impact-on-genai-e278cb81bd5c

[16] Llama 2: Open Foundation and Fine-Tuned Chat Models https://arxiv.org/pdf/2307.09288.pdf

[17] Yuxian Gu, Li Dong, Furu Wei, Minlie Huang, Knowledge Distillation of Large Language Models, https://arxiv.org/abs/2306.08543

[18] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, Dan Alistarh, GPTQ: Accurate Post-Training Quantization for Generative Pre-Trained Transformers, https://arxiv.org/pdf/2210.17323.pdf

[19] Elias Frantar, Dan Alistarh, SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot, https://arxiv.org/abs/2301.00774

[20] Saverio Proto, Is possible to run Llama2 with 70B parameters on Azure Kubernetes Service with LangChain agents and tools ?, https://medium.com/microsoftazure/is-possible-to-run-llama2-with-70b-parameters-on-azure-kubernetes-service-with-langchain-agents-and-e6664ea52723

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, LoRA: Low-Rank Adaptation of Large Language Models, https://arxiv.org/abs/2106.09685

[22] https://aws.amazon.com/ec2/instance-types/inf2/

**Citation:** Amreth Chandrasehar, Generative AI and LLM Optimizing Techniques for Developing Cost Effective Enterprise Applications, International Journal of Artificial Intelligence & Applications (IJAIAP), 2(1), 2023, pp. 70-81

**Article Link:**
https://iaeme.com/MasterAdmin/Journal_uploads/IJAIAP/VOLUME_2_ISSUE_1/IJAIAP_02_01_004.pdf

**Abstract:**
https://iaeme.com/Home/article_id/IJAIAP_02_01_004