PROJECT REPORT

INTEL UNNATI INDUSTRIAL TRAINING PROGRAM – 2024

Submitted by

COMPILER CREW

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
PATIA, BHUBANESWAR,
ODISHA 751024

PROJECT INFORMATION

Problem Statement No.	:	**PS-4**
Problem Statement	:	Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot
Github Repo	:	https://github.com/KushagraIsTaken/Finetuning_Dolly-v2-3b_on_Alpaca_Dataset.git

COMPILER CREW TEAM

Team Leader	:	**Kushagra Agrawal**
Team Members	:	**Pracheeta Gupta** **Anushka Arya** **Amritanshi Acharya** **Yashaswini Sarangi**
Institution	:	**KIIT Deemed to be University**

MENTORS

Faculty Mentor	:	**Mr. Ajit Kumar Pasayat**
Industry Mentor	:	Ms. Vasudha Kumari
External Mentor	:	Mr. Abhishek Nandy

Project Report

PS-4: Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot

Abstract

This report explores the development and optimization journey of a Large Language Model (LLM) focused on creative text generation. The primary objective is to empower the model, named Dolly-v2-3b, to generate engaging and coherent narratives in response to diverse prompts. By fine-tuning on a specialized dataset comprising 15,000 instruction/response pairs, curated across various domains, the model excels in tasks such as text-generation. Its efficiency, with 3 billion parameters, ensures high-quality responses while maintaining computational lightness, crucial for practical applications prioritizing responsiveness and cost-effectiveness.

Integration with the Intel Extension for Transformers plays a pivotal role in enhancing the model's performance. This collaboration optimizes hardware utilization, resulting in faster inference times and improved efficiency during text generation tasks. Evaluation metrics like *eval_loss* and *eval_ppl* underscore the model's accuracy and predictive capability, showcasing its ability to deliver precise and contextually appropriate responses.

Benchmarking exercises highlight the model's robustness, with metrics indicating low latency and high throughput during inference. For instance, the model processes 100 samples in approximately 14.16 seconds, achieving an average throughput of 7.061 samples per second and demonstrating its suitability for real-time applications requiring rapid response capabilities. Furthermore, this report discusses the impact of fine-tuning methodologies, utilizing a systematic approach to ensure the model's outputs uphold ethical standards and inclusivity. By embedding prompts that encourage socially conscious storytelling, the training process mitigates bias and promotes the creation of engaging, unbiased narratives.

Keywords: *Large Language Models, Fine-Tuning, Alpaca Dataset, CPU Inference*

Problem Statement Analysis

This project focuses on the implementation and fine-tuning of Large Language Models (LLMs) to create a custom chatbot, with an emphasis on performing fine-tuning and inference on a CPU. First and foremost, it is essential to grasp the foundational concepts of Generative AI (GenAI) and its diverse applications. This involves exploring how GenAI can be leveraged to develop sophisticated models capable of generating detailed, relevant, and contextually accurate information.

Over the past few years, Large Language Models (LLMs) have become highly prominent due to their impressive abilities in natural language understanding and generation [1]. Despite their strengths, LLMs are known for being memory-inefficient and demanding substantial computational resources. For example, BERT (Bidirectional Encoder Representations from Transformers) has 110 million parameters in its base model, while GPT-3 includes 175 billion parameters, necessitating a minimum of 320 gigabytes of storage in half-precision (16-bit floating point) [2].

A significant aspect of this project is conducting simple LLM inference on a CPU. This step ensures that the solution remains accessible even with limited computational resources. Performing inference on a CPU is typically slower and more computationally intensive compared to using GPUs or TPUs. However, by utilizing the Intel Extension for Transformers, we can optimize model performance and efficiency specifically for Intel hardware. This allows us to ensure that the model runs efficiently, making the system more accessible[3].

The project also involves fine-tuning a pre-trained LLM, such as Dolly 3B, on specific datasets like Alpaca for a text-generation task [4], [5]. This process is of great importance as it allows us to tailor the model to provide detailed, relevant, and contextually accurate information, going beyond the capabilities of a mere chatbot. Fine-tuning involves addressing several challenges, such as the computational expense and the difficulty in evaluating fine-tuned model's performance [6]. To mitigate the lack of domain-specific data, techniques like data augmentation, transfer learning, and active learning are employed to selectively fine-tune the model with the most informative samples [7].

In summary, this project encompasses the foundational concepts of generative AI and practical steps to leverage large language models for building a custom chatbot. By fine-tuning a pre-trained model on specific datasets and optimizing its performance on CPU using Intel Extension for Transformers, we aim to create a sophisticated text-generation system suitable for various applications such as customer support and information retrieval.

Generative AI

Generative AI, sometimes referred to as generative modeling, is a subset of artificial intelligence that generates totally new content via machine learning algorithms. This content may consist of audio, video, text, code, or graphics. They learn to understand underlying patterns in content by identifying patterns and relationships in data. They then use this knowledge to create new products that are similar but not copies of the teaching materials [1], [7]. There are different types of Generative AI applications, some of which are listed in Table 1.

Table 1: Different Applications of Generative AI

Type of Model	Description	References
Text Generation Model	These models generate various written texts, such as stories, emails, and conversations.	[8]
Image Generation Models	GANs like StyleGAN and BigGAN create realistic and diverse visuals by competing neural networks.	[9]
Music Generative Models	Algorithms like MuseNet and Jukebox compose, combine, and complete songs by understanding musical patterns.	[10]
Video Creation Model	Models such as Video Transformers and DVD-GAN produce short, high-quality videos with minimal hardware.	[11]

Speech Generation Models	Models like Tacotron and WaveNet simulate human voice complexity for text-to-speech synthesis.	[12]
3D Model Creation	Algorithms like StructureNet generates 3D images from text, descriptions, or 2D drawings.	[13]
Conversation Creation Model	Custom models like those used by Siri and Alexa generate responses based on context and past interactions.	[14]
Image to Image Conversion	Models like pix2pix and CycleGAN transform input images into different forms, showcasing their versatility.	[15]
Text-to-image synthesis	Models like DALL-E 2 and Imagen create realistic images from textual descriptions by translating content.	[16]
Text-to-speech synthesis	Text-to-speech models like Tacotron and WaveNet transform text into natural-sounding audio through a two-step process.	[17]

Large Language Models (LLMs)

A sort of artificial intelligence known as an LLMs, or large language model, is made to comprehend, produce, and work with human language. At the forefront of natural language processing (NLP), this model is capable of a wide range of language-related tasks, including summarization, translation, text recognition, text synthesis, and even complicated question answering.

LLMs are built on the foundation of machine learning, particularly deep learning. Large datasets containing a diverse range of text sources, including books, papers, webpages, and more, are used to train them. The extensive training allows them to grasp the variation of human languages, including grammar, syntax, and semantics [18], [19].

The OpenAI-developed GPT-3 (Generative Pre-trained Transformer 3) is one example of an LLM. It is one of the biggest and most potent language models available today, with over 175 billion parameters. The term "parameter" describes the aspects of the model that are taught by training data; the more parameters a model has, the more sophisticated and powerful it will be [18]. The ability of GPT-3 and related models to do a variety of tasks with a high degree of fluency and relevance is what makes them successful and useful tools in a variety of applications, from customer support to content creation [19].

Nevertheless, despite their remarkable powers, LLMs have several limitations. One significant issue is the tendency of these models to produce "hallucinations," generating responses that sound plausible but are factually incorrect or nonsensical. This is because LLMs do not possess true understanding or consciousness; they generate responses based on patterns learned from their training data, without any grounding in the real world. Fine-tuning can help reduce hallucinations in Large Language Models (LLMs) to some extent, but it may not eliminate them [18].

Our model, dolly-v2-3b, is a Databricks-developed causal language model with 2.8 billion parameters. It is based on EleutherAI's Pythia-2.8b and has been refined using a corpus of roughly 15,000 records that Databricks staff members provided as instructions. Several competence areas from the InstructGPT paper are covered by this corpus: generation, information extraction, brainstorming, categorization, closed quality assurance, open quality assurance, and summarization. [20].

Although dolly-v2-3b is not a state-of-the-art model, it demonstrates surprisingly high-quality instruction-following behavior, which is not typical of the foundation model it is based on.

LLMs use a specific type of neural network architecture known as **Transformers**. Figure 1 illustrates the architecture of transformers, including its simplified form.They are particularly effective for NLP tasks because they can handle long-range dependencies in text, meaning that it can understand the context and relationship between words over long passages of text. This is achieved through mechanisms like self-attention, which allows the model to weigh the importance of different words in a sentence relative to each other.

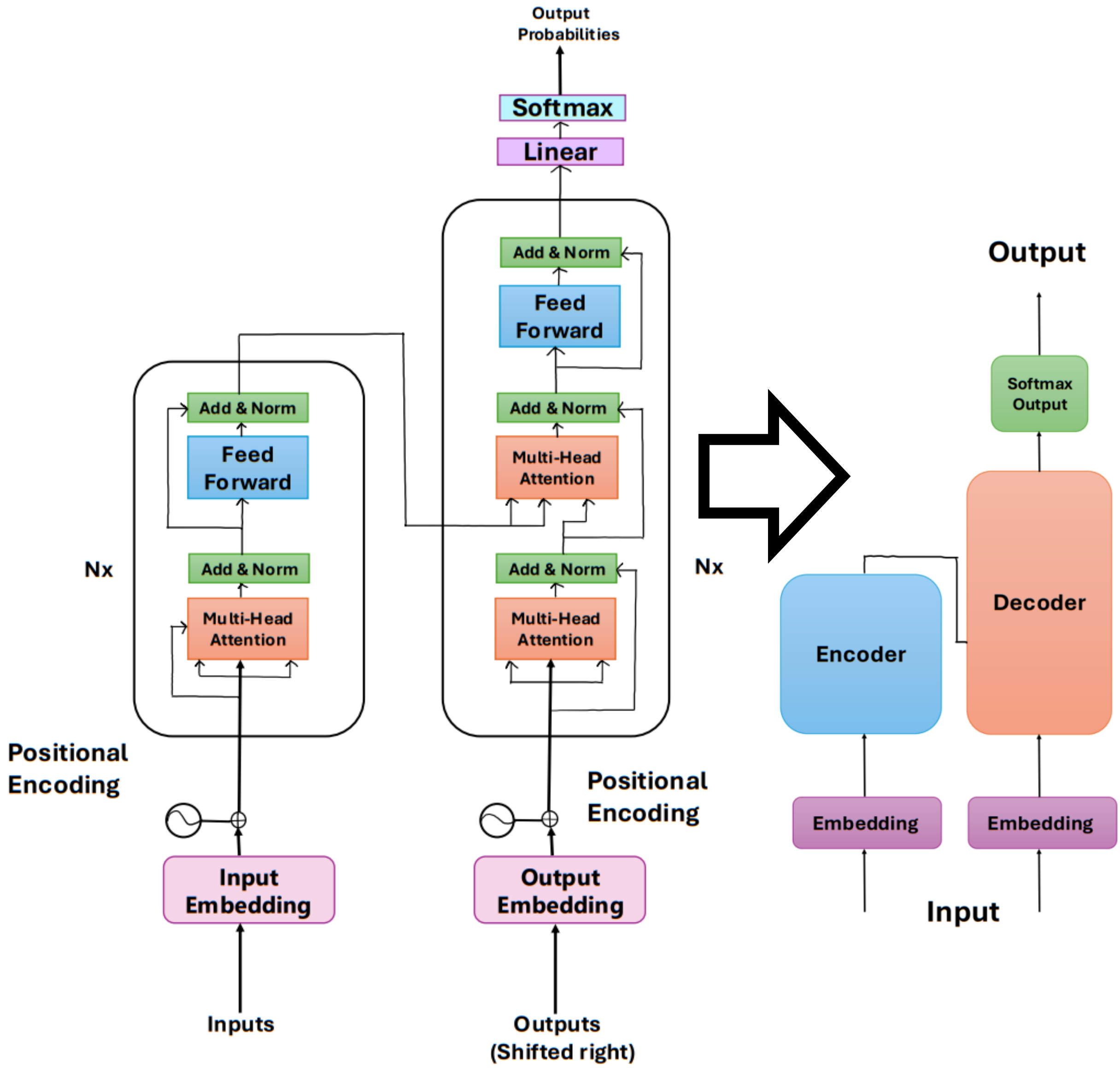


Fig. 1: Transformers Architecture and it’s simplified form

Transformers come in different types, each tailored for specific tasks:

- **Encoder Only:** These transformers focus solely on encoding input data. For example, BERT (Bidirectional Encoder Representations from Transformers) learns contextualized representations of words in a text. It's widely used for tasks like text classification, named entity recognition, and question answering[21].
- **Decoder Only:** On the other hand, decoder-only transformers specialize in generating output sequences based on encoded input. Models like GPT (Generative Pre-trained Transformer), such as GPT-2 and GPT-3, excel at tasks such as text generation, story writing, and completing prompts [22].
- **Encoder-Decoder:** This type combines both encoding and decoding capabilities. A well-known example is the Transformer model used in machine translation tasks, such as Google's Transformer model for Google Translate. It encodes the source language input and decodes it into the target language output, enabling accurate and fluent translation [20].

Each type of transformer architecture is designed to handle different aspects of natural language processing tasks, whether it involves understanding context, generating responses, or translating languages, showcasing their versatility across various applications.

Training an LLM involves feeding it vast amounts of text data and adjusting the model parameters to minimize errors in predicting the next word in a sequence. This process uses intensive computation and requires powerful GPUs or CPUs and large amounts of memory.

Our finetuned LLM model is primarily based on **text generation**, excelling in creating coherent and contextually relevant text from a given prompt. This capability is incredibly useful in various applications. For instance, in content creation, LLMs can help writers by generating articles, blog posts, or social media updates quickly and efficiently. In the realm of storytelling, these models can craft engaging narratives, offering creative assistance to authors and scriptwriters. Additionally, LLMs are valuable for drafting emails, providing users with polished and professional messages with minimal effort. By leveraging their ability to understand and generate human-like text, LLMs significantly enhance productivity and creativity across multiple domains.

Dataset

The significance of datasets in finetuning large language models (LLMs) cannot be overstated. Datasets provide the foundational knowledge upon which LLMs are built and refined. High-quality datasets ensure that models are exposed to a diverse range of linguistic patterns, vocabulary, and contextual nuances, which are crucial for improving their performance in specific tasks. Fine-tuning LLMs on domain-specific datasets allows the models to adapt their general language understanding to the particularities of a given field, whether it be legal language, medical terminology, or financial jargon. For instance, Gururangan *et al. (2020)* demonstrated that task-specific datasets significantly enhance the performance of LLMs on downstream tasks, enabling them to generate more accurate and contextually relevant responses [23].

Moreover, the process of fine-tuning on high-quality datasets helps in mitigating biases present in general pre-training datasets. By curating and using balanced and representative datasets during the fine-tuning phase, developers can address issues related to fairness and ethical AI. The impact of the dataset quality on model performance is underscored by studies such as that by Lee *et al. (2021)*, which highlighted that model fine-tuned on carefully selected datasets exhibited marked improvements in accuracy and robustness across various NLP benchmarks [21]. These findings emphasize that the choice and curation of datasets are critical steps in the fine-tuning process, as they directly influence the reliability and applicability of the resulting models in real-world scenarios.

The Alpaca dataset was developed by Stanford University by fine-tuning Meta's LLaMA 7B model to enhance language model capabilities using instruction tuning. The dataset comprises various types of inputs and their corresponding outputs, designed to teach the model to follow human instructions accurately.

The Alpaca dataset is based on the LLama model developed by Meta (formerly known as Facebook AI Research). The LLaMA (Large Language Model Meta AI) model is a family of state-of-the-art language models. These models are designed to perform a variety of natural language processing (NLP) tasks with high accuracy and efficiency. The dataset is designed to train language models to understand and respond to various types of instructions more effectively. This enhances the model's performance in interactive applications, such as virtual assistants and customer support systems.

The dataset consists of thousands of sets of inputs and outputs. These pairs cover a large range of topics and instructions which trains the models efficiently. The Alpaca dataset is used to fine-tune language models for tasks like question-answering, summarization, and other applications requiring precise and contextually appropriate responses. Some prominent uses of Alpaca Datasets are illustrated in Table 2.

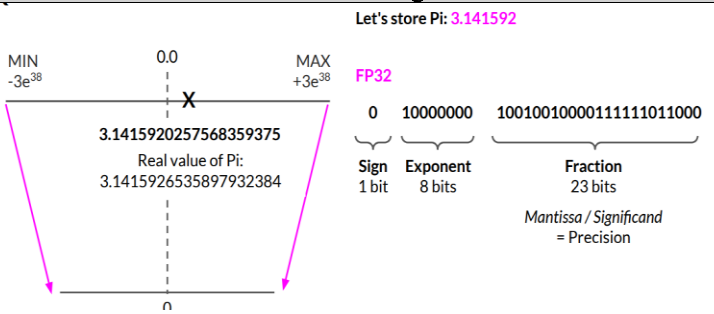
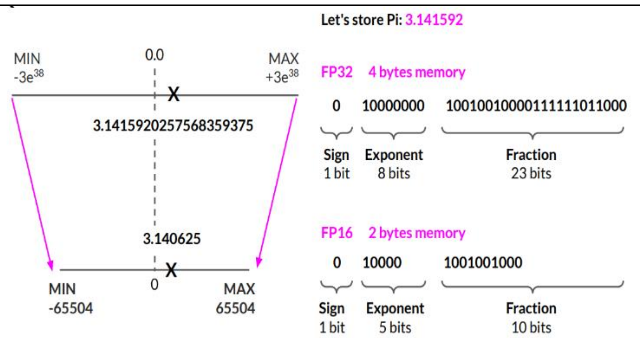
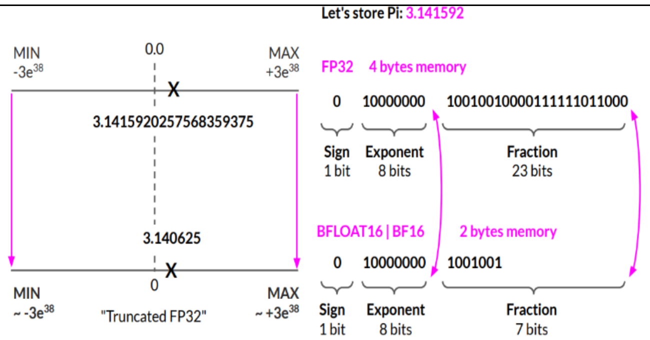
Table 2: Applications of Alpaca Dataset in Fine-Tuned Language Models

Application	**Description**
Virtual Assistants	Fine-tuned models have better responses to queries and instructions, making virtual assistants like Siri and Google Assistant more efficient.
Customer Support	These models can be used in chatbots and automated customer support systems to provide accurate responses to customer queries.
Research Assistance	Academics and researchers can use these models to quickly find relevant information, summarize research papers, and generate hypotheses, accelerating the research process.
Educational Tools	Language models fine-tuned with the Alpaca dataset can be used in educational platforms to provide personalized tutoring, answer student questions, and explain complex concepts in an easy-to-understand way.
Content Creation	These models can assist writers, journalists, and marketers by generating content ideas, drafting articles, and creating marketing copy that adheres to specific instructions and guidelines.

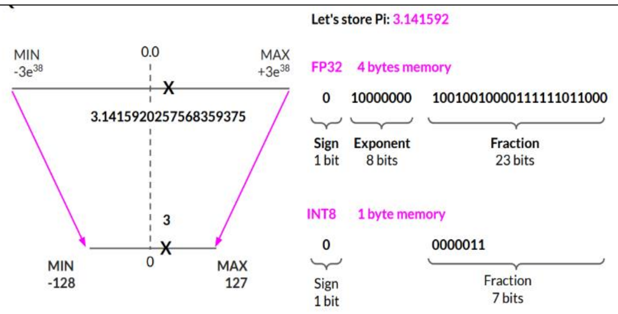
The Alpaca dataset is extensive, comprising 52,000 rows of instruction prompts designed for model fine-tuning. To optimize the training process and reduce the computational load, we selected the first 10,000 rows for our project. This approach ensures that we achieve a balance between model performance and training efficiency, allowing us to fine-tune the model effectively within a reasonable timeframe for the project. By focusing on a representative subset of the dataset, we can still capture a wide range of instructions and contexts, ensuring robust and reliable model training.

Quantization of Model

Quantization is a technique used in machine learning to reduce the computational and memory requirements of large language models (LLMs) without significantly compromising their performance. This is achieved by reducing the precision of the model's parameters (weights and activations) from higher precision (e.g., 32-bit floating point) to lower precision (e.g., 8-bit integer). Different type of quantization techniques are used in industrial practice like:

<table>
<tr><th>Quantization Method</th><th>Schematic Diagram</th></tr>
<tr><td>32-bit floating point refers to a numerical format in computing that stores numbers with a decimal point, commonly used for representing real numbers with precision and efficiency in mathematical calculations.</td><td>
</td></tr>
<tr><td>16-bit floating point is a numerical format in computing that uses half the bits of a standard floating-point representation, balancing storage efficiency and numerical precision for certain applications.</td><td>
</td></tr>
<tr><td>Brain Floating Point (BF-16) is a numerical format that uses 16 bits to represent floating-point numbers, designed to optimize deep learning and AI computations by balancing precision and efficiency.</td><td>
</td></tr>
</table>

Integer 8 (INT-8) is a numerical format that uses 8 bits to represent integer numbers, commonly used in machine learning and AI for efficient storage and computation of model parameters.



Prompt Engineering

Prompt engineering is a method of fine-tuning where carefully crafted inputs (prompts) are used to guide a pre-trained model's behavior and responses, effectively instructing it to perform specific tasks without altering the model's parameters. This approach leverages the model's existing capabilities by providing context or examples that lead to the desired output.

One-shot inference involves giving the model a single example to learn from and then generating the desired response based on this example. Two-shot inference provides the model with two examples before prompting it to generate a response, helping improve the model's understanding and performance. Multi-shot inference involves multiple examples, typically enhancing the model's ability to grasp more complex tasks and generate more accurate responses [24].

Fine-Tuning

Fine-Tuning is a crucial process in the field of machine learning and artificial intelligence, particularly when dealing with pre-trained models like Large Language Models (LLMs).This process is like giving an already trained model a specialized skill set by training it on a specialized dataset. For instance, a medical chatbot trained in the medical field would readily provide useful and relevant information about diseases like diabetes. If we ask about diabetes symptoms, the chatbot would offer detailed information on common symptoms, potential complications, recommended lifestyle changes, and when to seek medical help. This chatbot can be used for customer support and information retrieval in the medical field[25].

Pre-trained Models: Begin with a model that's already been trained on a vast amount of general data. Think of this model as having a solid understanding of language in general like language patterns, grammar, and general knowledge [21].

Specialized Training: Then, you train this model further on a smaller, more focused dataset that relates to the specific task you want it to perform well in, such as sentiment analysis, medical diagnosis, customer support, or any other specialized domain [22].

Fine-tuning pre-trained models offers several advantages. Firstly, it's more efficient than training a model from scratch due to its strong foundation of general knowledge, requiring less data and time to adapt. Secondly, models fine-tuned for specific tasks generally exhibit improved performance compared to those that are not tailored [26]. Thirdly, fine-tuning enhances versatility by allowing a single pre-trained model to adapt to multiple tasks. For instance, a general language model can be fine-tuned separately for tasks such as translation, summarization, and question-answering, optimizing its utility across various domains [24].

Fine-tuned models are transforming various sectors with their specialized capabilities. In healthcare, these models are invaluable for diagnosing diseases from medical records, utilizing their ability to sift through vast amounts of data and identify critical patterns. Customer service benefits from chatbots fine-tuned on relevant data, as they can provide more accurate and responsive support to inquiries. Finance relies on these models to analyze intricate financial trends and detect anomalies like fraud, offering essential insights for decision-makers. Legal professionals leverage fine-tuned models to expedite document review processes, ensuring thoroughness and precision in legal analyses. Furthermore, in creative fields such as writing, these models assist by generating text in specific styles or genres, aiding writers in content creation and adaptation.

Generative AI Model Development Lifecycle

A general generative AI model development lifecycle is depicted in Figure 2.

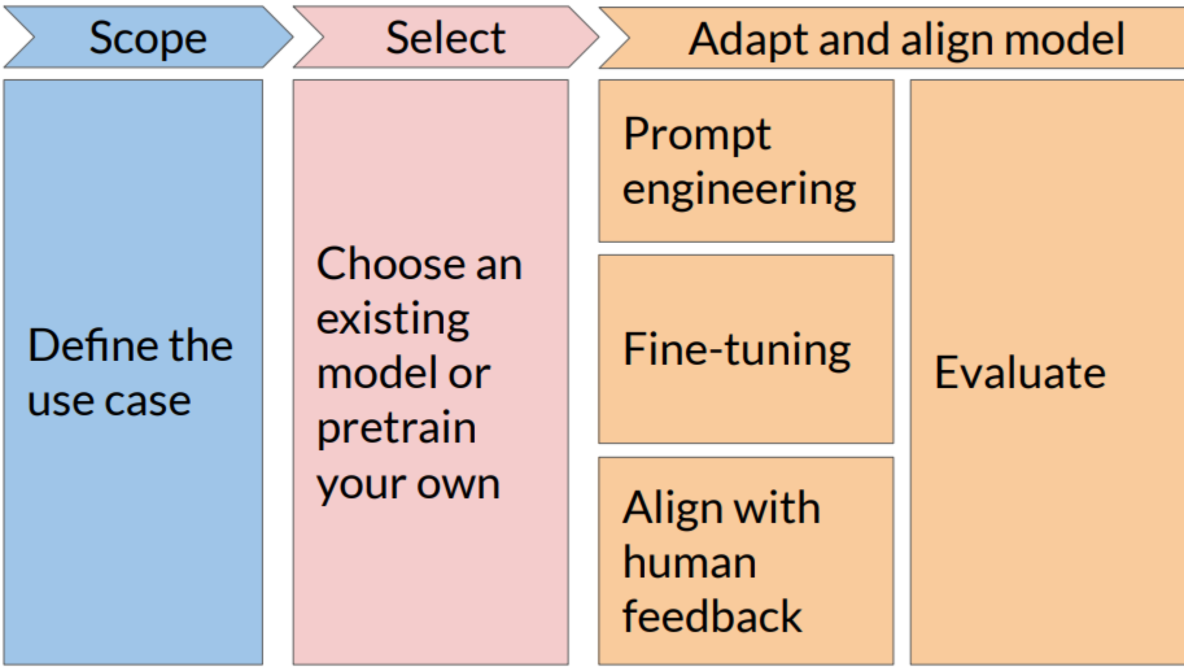


Figure 2: Generative AI Model Development Lifecycle

1. Define the use case: We need to define the use of our model and answer the following questions:
    a. What is the use of this model?
    b. Where will it be used?
    c. Who are the end-users of the model?
    d. What problem is the model solving?
2. Choose an existing model or pre-train your own model: To choose a pre-trained model we should take the following things into account:
    a. Task Suitability: Assess if the model can handle your specific tasks effectively.
    b. Performance: Check if the model achieves desired benchmarks and performance metrics.

   c. Resource Requirements: Evaluate computational needs for training, fine-tuning, and deployment.
   d. Community Support: Consider availability of documentation, tutorials, and community support.
   e. Ethical Considerations: Ensure the model aligns with ethical guidelines and regulatory requirements.
3. Adapt and Align the Model: During this stage, we customize the general-purpose model to fit our specific use-case by employing techniques such as:
   a. Prompt Engineering
   b. Fine-Tuning
   c. Incorporating Human Feedback

   We then evaluate the model across various parameters to ensure it meets our performance criteria.

Model Development

In the first phase of developing our Large Language Model (LLM), we've defined its primary use-case as a creative text-generation model. This entails empowering the model to generate text creatively in response to prompts. For instance, when prompted with "Once upon a time," the model is expected to produce a coherent and engaging narrative or continuation of the story. Key objectives include enabling the model to handle and interpret prompts effectively, ensuring generated text flows naturally and maintains coherence, and evaluating its outputs for creativity and linguistic quality. This phase sets the foundation for refining the model's capabilities to excel in creative storytelling applications, where natural and imaginative text generation is crucial for enhancing user experiences and creating engaging content.

Model Choice

We have chosen the Dolly-v2-3b model for our text generation tasks due to several compelling reasons. Firstly, Dolly-v2-3b is optimized with 3 billion parameters, making it computationally lighter compared to larger models while still delivering high-quality responses. This efficiency ensures faster inference times and lower resource requirements, which are crucial for practical applications where responsiveness and cost-effectiveness are priorities.

Moreover, Dolly-v2-3b benefits from being fine-tuned on a specialized corpus of approximately 15,000 instruction/response pairs, curated by Databricks employees across various capability domains. This fine-tuning process enhances the model's ability to follow instructions and generate coherent responses, particularly in structured tasks like classification, closed QA, summarization, and more. This specialization makes Dolly-v2-3b particularly suitable for applications requiring precise and accurate text generation based on specific prompts.

Additionally, Dolly-v2-3b is supported by Intel Extension for transformers, further optimizing its performance on hardware configurations that support these enhancements. This support ensures that the model can leverage hardware acceleration effectively, enhancing both speed and efficiency during inference. By leveraging Dolly-v2-3b, we strike a balance between model size, performance, and specialization, enabling us to achieve reliable and efficient text generation capabilities tailored to our application needs.

Adapt & Align Model

In this phase, we fine-tune our model, leveraging the first 10,000 rows of the Alpaca Dataset, to enhance its performance and alignment with our specific objectives. This dataset serves as a foundational source of diverse and relevant text examples, enabling the model to learn and adapt to the nuances required for effective storytelling and narrative continuation.

To guide the fine-tuning process, we employ a carefully crafted system prompt designed to encourage creative and imaginative storytelling. Each prompt begins with:

"You are a creative and imaginative storyteller. Your task is to continue stories in a captivating and coherent manner. Ensure that your narratives are engaging, appropriate for all audiences, and maintain a positive tone. Avoid any content that is harmful, unethical, racist, sexist, toxic, dangerous, or illegal. Strive to create stories that are socially unbiased and enjoyable. If a prompt is unclear or does not make sense, provide a creative and sensible continuation while maintaining coherence."

This systematic approach ensures that our model learns to generate narratives that are not only entertaining and engaging but also uphold ethical standards and inclusivity. By embedding this prompt with each training example, we mitigate bias and promote the creation of socially conscious and enjoyable storytelling outputs. This fine-tuning process aims to optimize the model's ability to respond creatively and sensibly to a wide range of narrative prompts, thereby enhancing its overall performance and relevance for our application.

Evaluation

We have employed several evaluation metrics to assess the performance of our machine learning models. Key metrics include *eval_loss*, which measures the difference between predicted and actual values during training epochs. Achieving a lower *eval_loss* indicates better model accuracy. Another crucial metric is *eval_ppl* (perplexity), which evaluates the model's predictive capability based on the sequence of tokens. A lower *eval_ppl* signifies improved prediction quality. Additionally, *eval_samples_per_second* provides insights into the model's computational efficiency by measuring how many input samples it processes per second. These metrics guide our optimization efforts, helping us refine model performance and enhance its effectiveness in real-world applications. In addition to these evaluation metrics, we utilize benchmark techniques to further assess our models' performance.

Observations – Fine Tuning

In our endeavor to optimize and refine our model's performance for chatbot applications, the integration of the Intel Extension for Transformers played a pivotal role. This collaboration allowed us to harness advanced features seamlessly interfaced with Hugging Face's Transformers library, amplifying our capabilities beyond conventional boundaries. Central to our approach was the strategic configuration of key components, starting with `ModelArguments`, which anchored our efforts on the robust "databricks/dolly-v2-3b" model, recognized for its efficacy in complex conversational tasks.

Our methodology extended to meticulous data handling through `DataArguments`, facilitating the incorporation of diverse datasets. This meticulous integration ensured our model's proficiency in discerning intricate patterns and nuances essential for accurate and contextually aware responses in real-world scenarios. Complementing these efforts, `TrainingArguments` enabled a finely tuned training regimen, optimizing parameters such as batch sizes, epochs, and leveraging `bf16` precision where applicable, to enhance computational efficiency and performance reliability.

The culmination of our approach was realized in `TextGenerationFinetuningConfig`, where all configurations harmonized into a cohesive framework. This comprehensive setup not only streamlined the fine-tuning process but also facilitated an optimized deployment of our enhanced model. The Intel Extension for Transformers proved instrumental in pushing the boundaries of our model's capabilities, delivering a solution poised to redefine interactive AI experiences in chatbot applications.

The entire model was fine-tuned on the hardware configurations on the Intel Developer Cloud as depicted in Figure 3.

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         46 bits physical, 57 bits virtual
  Byte Order:            Little Endian
CPU(s):                  192
  On-line CPU(s) list:   0-191
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Xeon(R) Platinum 8468V
    CPU family:          6
    Model:               143
    Thread(s) per core:  2
    Core(s) per socket:  48
    Socket(s):           2
    Stepping:            8
    Frequency boost:     enabled
    CPU max MHz:         2401.0000
    CPU min MHz:         800.0000
    BogoMIPS:            4800.00
```

Figure 3: Hardware Configuration of Jupyter Notebook environment on IDC

As evident form figure 4(a), the support from Intel Extension for Transformers was instrumental in enabling us to complete the fine-tuning work within an impressive time frame of 2:41:35, all while achieving a significantly low training loss. This efficiency was largely attributed to the advanced optimization features provided by Intel, which seamlessly integrated with our existing Hugging Face infrastructure. The utilization of bf16 precision, for instance, not only reduced training time but also maintained model accuracy, ensuring that our fine-tuning process was both swift and effective.

As evident from figure 4(b) without the Intel Extension for Transformers, we attempted to execute the same script, which approximated a completion time of 6:11:09, nearly 2.3 times longer than with the Intel enhancements. This stark difference underscores the transformative impact of Intel's optimizations on our workflow. Although the script got stuck underway due to frequent server disconnection issue on the IDC Jupyter Notebook environment.

[3711/3711 2:41:35, Epoch 2/3]

Step	Training Loss
500	0.587900
1000	0.503800
1500	0.474200
2000	0.457300
2500	0.472000
3000	0.430000
3500	0.420800

Figure 4(a) Training Time & Training Loss with Intel Extension for Transformers

[1312/3375 3:55:41 < 6:11:09,

Step	Training Loss	Validation Loss
500	0.933800	0.946631
1000	0.771800	0.882140

Figure 4(b) Anticipated Training Time & Training Loss without Intel Extension for Transformers

The step v/s training loss graph can thus be plotted in Figure 5.

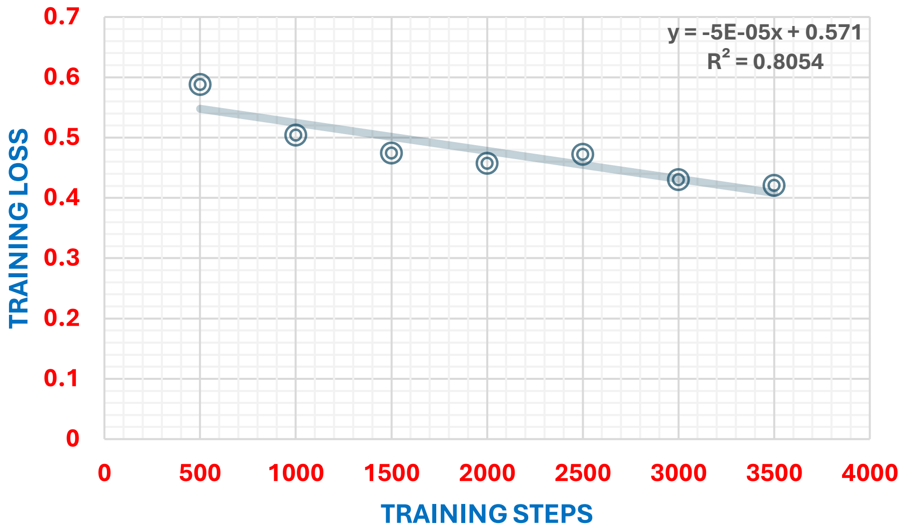


Figure 5: Training loss v/s Training Steps scatter plot

The regression line equation clearly indicates a negative slope, indicating a decrease in the training loss with increase in the training steps in the studied range.

Observations – Benchmarking

Based on the evaluation metrics obtained after training the model, the Intel extension for transformers has clearly demonstrated its role in enhancing both performance metrics and efficiency. As evident in Figure 6, the model achieved a low evaluation loss of 0.4564 and a perplexity (eval_ppl) of 1.5786, indicating its effectiveness in accurately predicting target outcomes. During evaluation, the model processed 100 samples in approximately 14.16 seconds, achieving an average throughput of 7.061 samples per second and 1.765 steps per second.

In terms of inference performance, the average latency per inference was measured at 4.0511 seconds, highlighting the model's efficiency in generating predictions swiftly for input data. Furthermore, the throughput of 505.54 tokens per second underscores the model's capability to process input tokens rapidly during inference, which is critical for applications requiring real-time responsiveness.

These results clearly demonstrate that the Intel extension for transformers has significantly improved the model's efficiency and performance metrics. While achieving impressive evaluation metrics such as low loss and perplexity, further optimization could focus on fine-tuning architectural aspects, leveraging hardware capabilities more effectively, and exploring techniques like batching to further enhance inference speed without compromising accuracy. This showcases the extension's pivotal role in optimizing and accelerating transformer model performance across various applications.

```
***** eval metrics *****
  epoch                   =     2.9988
  eval_loss               =     0.4564
  eval_ppl                =     1.5786
  eval_runtime            = 0:00:14.16
  eval_samples            =        100
  eval_samples_per_second =      7.061
  eval_steps_per_second   =      1.765
```

```
Average Latency: 4.0511 seconds
Throughput: 505.54 tokens/second
```

Figure 6: Evaluation and Benchmarking Parameters

Average Latency, recorded at 4.0511 seconds, indicates the average time taken for the model to process individual tasks or requests. Lower latency is crucial for ensuring responsive performance in applications. Throughput, measured at 505.54 tokens/second, reveals the model's processing speed, indicating its capacity to handle a high volume of data efficiently. Together, these metrics and benchmarks provide a comprehensive understanding of our models' capabilities, guiding our decisions in refining and deploying them effectively.

Observations – Inferencing

We tried inferencing this model on five different prompts to check for the outputs. The maximum tokens were fixed at 75 tokens. The outcomes from various prompts are analyzed and documented in Table 3 for shorter prompts and Table 4 for longer prompts.

These outputs were generated to evaluate the model's response across varied prompts within the constraints of a maximum of 75 tokens per inference. Each prompt aimed to gauge the model's ability to generate coherent and relevant responses based on the input provided.

Table 3: Inferencing on Prompts with smaller context

Sr. No.	Prompt	Response
1	Computer Engineering is	Computer Engineering is a broad field that includes many different disciplines. Some of the more popular computer engineering disciplines include computer architecture, computer engineering, computer graphics, and computer science. Computer engineering is the study of the design, construction, and operation of electronic systems and devices. Computer engineering students typically focus on the design of hardware and software, and the integration of these components into larger systems.
2	Once upon a time	Once upon a time, there was a little girl who loved to read. She loved to read books about princesses, fairies, and unicorns. She loved to read about the world she created in her head. She loved to read until one day, she found a book that changed her life. It was a book about a little girl who loved to read. The book was called
3	Long ago lived a king	Long ago lived a king and queen. They had three children: a boy and two girls. The boy was very handsome and the girls were beautiful. The king and queen loved their children very much. They were so proud of them. The king and queen were especially proud of their son. He was a great warrior and a hero. He was strong and brave. He was a true leader. He
4	Once upon a time in Stanford	Once upon a time in Stanford's English classroom, there was a young man named Harry Potter. He was a very smart wizard, and he was very good at his job. He was also a very bad boy. He would do things like drink excessively, get into fights, and generally cause trouble. One day, Harry Potter was walking down the street when he saw a beautiful girl. He was immediately attracted

Table 4: Inferencing on Prompts with larger context

Sr. No.	Prompt:
5	Once upon a time, in a faraway kingdom, there lived a princess named Snow White. She was as beautiful as a blooming rose, with skin as white as snow, lips as red as blood, and hair as black as ebony. Her mother, the queen, passed away when Snow White was very young, and her father, the king, remarried a vain and wicked woman who was obsessed with her own beauty. The new queen owned a magical mirror that she would often consult, asking who is the fairest of them all. The mirror always replied that she, the queen, was the fairest of them all. However, as Snow White grew older, her beauty surpassed that of the queen, and one day, when the queen asked her mirror the familiar question, it replied that Snow White is the fairest of them all. Consumed with jealousy, the queen ordered a huntsman to take Snow White into the forest and kill her. She demanded that he bring back Snow White's heart as proof. The huntsman, however, could not

	bring himself to harm the innocent princess. Instead, he let her go, advising her to run far away and never return. To deceive the queen, he brought back the heart of a wild boar.
	Response:
	Once upon a time, in a faraway kingdom, there lived a princess named Snow White. She was as beautiful as a blooming rose, wit h skin as white as snow, lips as red as blood, and hair as black as ebony. Her mother, the queen, passed away when Snow White was very young, and her father, the king, remarried a vain and wicked woman who was obsessed with her own beauty. The new que en owned a magical mirror that she would often consult, asking who is the fairest of them all. The mirror always replied that she, the queen, was the fairest of them all. However, as Snow White grew older, her beauty surpassed that of the queen, and o ne day, when the queen asked her mirror the familiar question, it replied that Snow White is the fairest of them all. Consume d with jealousy, the queen ordered a huntsman to take Snow White into the forest and kill her. She demanded that he bring bac k Snow White's heart as proof. The huntsman, however, could not bring himself to harm the innocent princess. Instead, he let her go, advising her to run far away and never return. To deceive the queen, he brought back the heart of a wild boar. Snow W hite fled to a nearby castle, where she was welcomed by the king and his daughter, the princesses. The king and princesses lo ved Snow White dearly, and they decided to hide her from the queen. They dressed her up in beautiful clothes and made her the ir own personal lady-in-waiting. The queen, however, did not give up. She
6	Prompt:
	Once upon a time, in a peaceful meadow, there lived a speedy rabbit who boasted about his swift running abilities, often mocking the slow-moving tortoise for his sluggish pace. The tortoise, calm and patient, ignored the rabbit's taunts. One day, tired of the rabbit's bragging, the tortoise challenged him to a race. Amused, the rabbit accepted. As the race began, the rabbit sprinted ahead confidently, leaving the tortoise far behind. He took a nap under a shady tree, sure he had time to spare. Meanwhile, the tortoise plodded steadily, focused on the finish line and unfazed by the rabbit's antics.
	Response:
	Once upon a time, in a peaceful meadow, there lived a speedy rabbit who boasted about his swift running abilit often mocking the slow-moving tortoise for his sluggish pace. The tortoise, calm and patient, ignored the rabb taunts. One day, tired of the rabbit's bragging, the tortoise challenged him to a race. Amused, the rabbit acc d. As the race began, the rabbit sprinted ahead confidently, leaving the tortoise far behind. He took a nap un shady tree, sure he had time to spare. Meanwhile, the tortoise plodded steadily, focused on the finish line an azed by the rabbit's antics. As the race drew near, the rabbit awoke and set off at a sprint once again. He was sure to beat the tortoise t ime. But as the rabbit crossed the finish line, he saw the tortoise had already arrived. The rabbit was crestf n, but the tortoise was elated. He had won the race! The rabbit was crestfall

Future Prospects of the Model:

Our intention is to elevate the capabilities of this model in text generation through extensive training across diverse datasets and refinement through human feedback integration. With a solid foundation in Generative AI and Language Models (LLMs), our team is committed to crafting bespoke solutions using cutting-edge tools like the Intel Extension for Transformers. Among the envisioned applications of these LLM-based solutions is the development of a student-friendly chatbot tailored to provide comprehensive university information. However, our ambitions extend beyond this specific use case, encompassing a broad spectrum of potential applications where sophisticated text generation can significantly enhance user experiences and operational efficiencies.

We also intend to leverage the Retrieval Augmented Generation (RAG) approach to address a diverse array of challenges. By integrating information retrieval with advanced generation techniques, RAG models enable us to enhance solutions across multiple domains. These include but are not limited to creating intelligent assistants capable of retrieving and synthesizing knowledge from vast datasets in real-time, developing robust question-answering systems for complex queries, and constructing interactive content generators that adapt to user preferences. This approach not only improves the accuracy and relevance of generated content but also empowers applications in education, healthcare, customer service, and beyond, by efficiently combining the strengths of retrieval and generation mechanisms.

Takebacks:

The Intel Unnati Industrial Training Program – 2024 has equipped us comprehensively in transformers, Generative AI, Language Model (LLMs), fine-tuning processes, and the Intel Extension for Transformers library. Our goal is to apply this acquired expertise practically to address everyday challenges. By leveraging our skills in these advanced technologies, we aim to develop innovative solutions that enhance efficiency and effectiveness in various domains. These include but are not limited to creating tailored conversational agents, optimizing data processing workflows, and enhancing decision-making through predictive analytics. Our approach integrates cutting-edge techniques with practical applications, driving impactful solutions across industries and sectors.

Conclusion:

In conclusion, the generative AI model development lifecycle outlined in this report provides a structured framework for effectively harnessing the capabilities of advanced language models (LLMs) like the Dolly-v2-3b model. By meticulously defining use cases, selecting appropriate models, and customizing them through fine-tuning and adaptation, we ensure that our solutions are tailored to specific needs and optimized for performance. The integration of Intel Extension for Transformers has been pivotal in enhancing our model's efficiency and scalability, enabling significant reductions in training time and inference latency.

Looking forward, the insights gained from this development lifecycle and the advancements made through the Intel Unnati Industrial Training Program – 2024 will continue to drive our efforts towards solving real-world problems. From creating innovative conversational agents to optimizing data workflows and beyond, our commitment to leveraging cutting-edge technologies remains steadfast. As we explore future prospects such as retrieval augmented generation (RAG) and expand into diverse applications, we are poised to deliver impactful solutions that redefine interactive AI experiences across various domains. Through continuous innovation and integration of best practices, we strive to push the boundaries of what is possible in generative AI, driving positive change and enhancing user experiences worldwide.

Acknowledgements:

The members of the team extend their heartfelt gratitude to several individuals and entities whose support was instrumental in the completion of this project. First and foremost, we express our sincere appreciation to our esteemed mentors: Mr. Ajit Pasayat, whose guidance as our Faculty Mentor was invaluable; Mr. Abhishek Nandy, our External Mentor, for his expertise, continuous encouragement, and prompt resolution of our queries; and Ms. Vasudha Kumari, our Industry Mentor, whose insights and detailed sessions enriched our project significantly. We are also thankful to KIIT Deemed to be University for providing us with the platform and resources necessary for our work. Special recognition goes to the Intel Unnati Team and all others who contributed their time, expertise, and encouragement along the way.

References:

[1] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *Adv Neural Inf Process Syst*, vol. 33, pp. 1877–1901, 2020, Accessed: Jun. 26, 2024. [Online]. Available: https://commoncrawl.org/the-data/

[2] E. Frantar and D. Alistarh, "SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot," *International Conference on Machine Learning*, 2023, doi: 10.48550/ARXIV.2301.00774.

[3] P. Yu, H. Xu, X. Hu, and C. Deng, "Leveraging Generative AI and Large Language Models: A Comprehensive Roadmap for Healthcare Integration," *Healthcare 2023, Vol. 11, Page 2776*, vol. 11, no. 20, p. 2776, Oct. 2023, doi: 10.3390/HEALTHCARE11202776.

[4] R. Taori *et al.*, "Stanford Alpaca: An Instruction-following LLaMA model," *GitHub repository*. GitHub, 2023.

[5] M. Conover *et al.*, "Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM." 2023. [Online]. Available: https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm

[6] K. Huang, H. Yin, H. Huang, and W. Gao, "Towards Green AI in Fine-tuning Large Language Models via Adaptive Backpropagation," Sep. 2023, Accessed: Jun. 26, 2024. [Online]. Available: https://arxiv.org/abs/2309.13192v2

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners." [Online]. Available: https://github.com/codelucas/newspaper

[8] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." [Online]. Available: https://github.com/tensorflow/tensor2tensor

[9] T. Karras, S. Laine, and T. Aila, "A Style-Based Generator Architecture for Generative Adversarial Networks," *IEEE Trans Pattern Anal Mach Intell*, vol. 43, no. 12, pp. 4217–4228, Dec. 2021, doi: 10.1109/TPAMI.2020.2970919.

[10] "MuseNet | OpenAI." Accessed: Jun. 26, 2024. [Online]. Available: https://openai.com/index/musenet/

[11] E. Bilokopytov, "Disjointly non-singular operators on order continuous Banach lattices complement the unbounded norm topology," *J Math Anal Appl*, vol. 506, no. 1, p. 125556, Feb. 2022, doi: 10.1016/J.JMAA.2021.125556.

[12] A. van den Oord *et al.*, "WaveNet: A Generative Model for Raw Audio," in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[13] K. Mo *et al.*, "StructureNet: Hierarchical graph networks for 3D shape generation," *ACM Trans Graph*, vol. 38, no. 6, Nov. 2019, doi: 10.1145/3355089.3356527/SUPPL_FILE/A242-MO.ZIP.

[14] X. Gu, K. M. Yoo, and J. W. Ha, "DialogBERT: Discourse-Aware Response Generation via Learning to Recover and Rank Utterances," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, pp. 12911–12919, May 2021, doi: 10.1609/AAAI.V35I14.17527.

[15] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-October, pp. 2242–2251, Dec. 2017, doi: 10.1109/ICCV.2017.244.

[16] C. Saharia *et al.*, "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding."

[17] J. Shen *et al.*, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, pp. 4779–4783, Sep. 2018, doi: 10.1109/ICASSP.2018.8461368.

[18] S. Makridakis, F. Petropoulos, and Y. Kang, "Large Language Models: Their Success and Impact," *Forecasting 2023, Vol. 5, Pages 536-549*, vol. 5, no. 3, pp. 536–549, Aug. 2023, doi: 10.3390/FORECAST5030030.

[19] B. Agüera Y Arcas, "Do Large Language Models Understand Us?," *Daedalus*, vol. 151, no. 2, pp. 183–197, May 2022, doi: 10.1162/DAED_A_01909.

[20] A. Vaswani *et al.*, "Attention is All you Need," *Adv Neural Inf Process Syst*, vol. 30, 2017.

[21] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*, pp. 4171–4186, 2019, doi: 10.18653/V1/N19-1423.

[22] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners.” [Online]. Available: https://github.com/codelucas/newspaper

[23] S. Gururangan *et al.*, “Don’t Stop Pretraining: Adapt Language Models to Domains and Tasks,” *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 8342–8360, 2020, doi: 10.18653/V1/2020.ACL-MAIN.740.

[24] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” *Adv Neural Inf Process Syst*, vol. 33, pp. 1877–1901, 2020, Accessed: Jun. 27, 2024. [Online]. Available: https://commoncrawl.org/the-data/

[25] J. Howard and S. Ruder, “Universal Language Model Fine-tuning for Text Classification,” *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 328–339, 2018, doi: 10.18653/V1/P18-1031.

[26] M. E. Peters *et al.*, “Deep Contextualized Word Representations,” *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 2227–2237, 2018, doi: 10.18653/V1/N18-1202.

In case of any clarification
Please reach out to

Team Leader	:	**Kushagra Agrawal**
Address	:	School of Computer Engineering, KIIT Deemed to be University Patia, Bhuneshwar, Odisha
Email	:	kush4409@gmail.com 2205044@kiit.ac.in kushagra.agrawal@ieee.org
LinkedIn	:	https://www.linkedin.com/in/kushagra-agrawal-51a3121ab/

COMPILER CREW TEAM