

Intel Unnati Industrial Training Program 2024

Project Report

Introduction to GenAI, Simple LLM Inference on CPU and Finetuning of LLM Models to Create a Custom Chatbot

Submitted by

Name: Rudransh Singh

Email – rudransh.singh2504@gmail.com

Github Link -[Supervised finetuning project](#)

Google Drive Link(Recording)- [Recordings](#)

Institute – Manipal Institute of Technology, Manipal

Industrial Mentor – Dr Rashmi Laxmikant Malghan

Table of Contents

Sl. No	Content	Pg No.
1.	Introduction	3
2.	Problem Statement	3
3.	Objective	4
4.	Abstract	4
5.	Inference On CPU	5
6.	Finetuning for Text Generation	8
7.	Outputs	10
8.	Process	12
9.	Future Prospects	13
10.	Conclusion	14

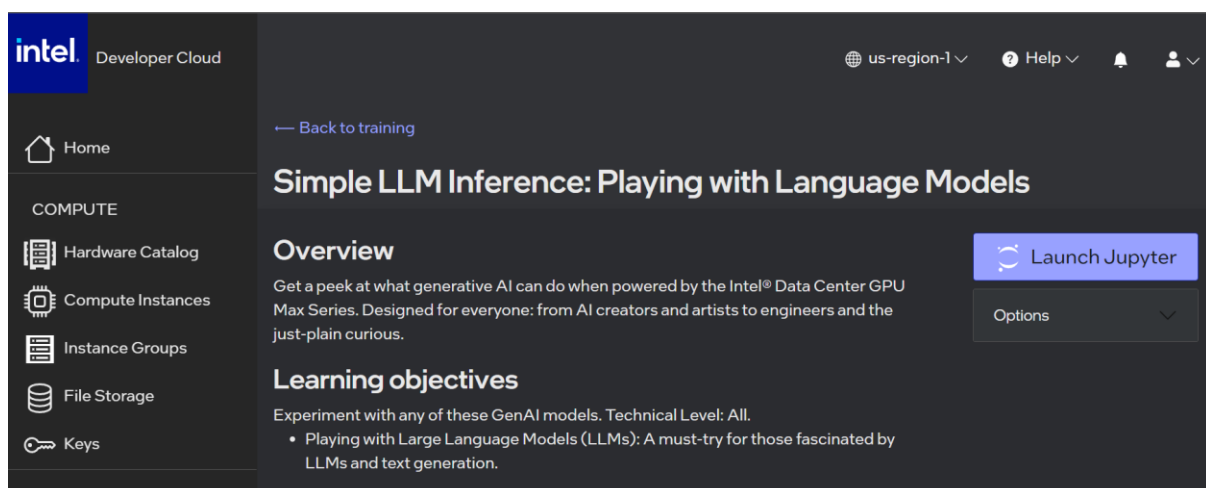
Introduction

The Intel Unnati Industrial Training program provided a unique opportunity to delve into the dynamic field of Generative AI (GenAI). This training was centered around understanding and implementing key concepts of GenAI, specifically focusing on **Simple Large Language Model (LLM) inference on CPUs and the fine-tuning of LLM models to create custom chatbots**. Utilizing the Intel Developer Cloud Console platform, we were able to harness advanced computational resources to develop and optimize our models, gaining hands-on experience with state-of-the-art AI technology.

Throughout the training, we explored the fundamentals of GenAI, including the architecture and functioning of LLMs. Our practical sessions involved deploying simple LLM inference tasks on CPUs, demonstrating the feasibility and efficiency of running complex AI models on accessible hardware. Furthermore, we delved into the customization of these models through fine-tuning techniques, ultimately developing a bespoke chatbot tailored to specific requirements. This program not only enhanced our technical skills but also provided invaluable insights into the practical applications of GenAI, equipping us with the knowledge to innovate and excel in the rapidly evolving AI landscape.

Problem Statement

The problem statement provided to us by Intel was **Simple Large Language Model (LLM) inference on CPUs and the fine-tuning of LLM models to create custom chatbots**.



Fig(1). Preview of Intel Developer Cloud

Objective

- Gain a comprehensive understanding of the process and importance of fine-tuning pre-trained language models.
- Learn to optimize training parameters and configurations to improve the efficiency and effectiveness of the model training process.
- Address and resolve challenges related to memory usage and runtime errors to ensure smooth and efficient model training.
- Gain practical experience of GenAI and LLM models by fine-tuning a model and performing Inference on CPU, understanding its architecture, and enhancing its performance.
- Enhance problem-solving skills by troubleshooting and resolving issues encountered during the fine-tuning process.

Abstract

The process began with the selection and preparation of the dataset, followed by the configuration of the training environment. We employed techniques to optimize memory usage and runtime efficiency, such as gradient checkpointing and parameter-efficient fine-tuning. The Llama-2 model was then fine-tuned using the **Supervised Fine-Tuning (SFT)** approach, which allowed us to adapt the model's capabilities to specific tasks relevant to our dataset.

Our method offers notable advantages over existing approaches. By using a targeted dataset and optimizing the training process, we were able to achieve more relevant and precise results. The fine-tuned model outperformed general-purpose models in specific tasks, showcasing the effectiveness of our approach. This project not only provided practical experience in model fine-tuning but also highlighted the potential of customized NLP solutions in addressing specific problems more effectively than generic models.

Inference on CPU

During the Intel Unnati Industrial Training program, we had the opportunity to perform simple LLM inference on CPUs using Jupyter notebooks, which facilitated an interactive and hands-on learning experience. We set up our custom environment and kernel within these notebooks to ensure the effective functioning of the provided code and models. This setup allowed us to execute LLM inference tasks seamlessly, demonstrating how even resource-constrained environments like CPUs can handle substantial AI workloads efficiently. By configuring our environment and kernel appropriately, we were able to optimize the performance of the models, thus gaining a deeper understanding of the practical aspects of deploying and managing AI systems on various hardware configurations.

LLM inference on CPU involves running pre-trained large language models on a central processing unit (CPU) to generate predictions or outputs based on input data. Unlike GPUs, which are traditionally favoured for their parallel processing capabilities, CPUs can still effectively handle LLM inference, albeit at potentially slower speeds. The process involves loading the model into memory, feeding it input text, and processing this input to produce coherent and contextually relevant responses.

1. Input 1: Tell me about Intel Xeon Scalable Processors.

```
[1]: # BF16 Optimization
from intel_extension_for_transformers.neural_chat import build_chatbot, PipelineConfig
from intel_extension_for_transformers.transformers import MixedPrecisionConfig
config = PipelineConfig(optimization_config=MixedPrecisionConfig())
chatbot = build_chatbot(config)
response = chatbot.predict(query="Tell me about Intel Xeon Scalable Processors.")
print(response)
```


```
/home/u17a883cf852b9f81bb6e3bb6ee3b080/.conda/envs/itrex/lib/python3.10/site-packages/transformers/deepspeed.py:23: FutureWarning: transformers.deepspeed module is deprecated and will be removed in a future version. Please import deepspeed modules directly from transformers.integrations
warnings.warn(
/home/u17a883cf852b9f81bb6e3bb6ee3b080/.conda/envs/itrex/lib/python3.10/site-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
warnings.warn(
Loading model Intel/neural-chat-7b-v3-1
Error displaying widget: model not found
The Intel Xeon Scalable Processors represent a family of high-performance central processing units (CPUs) designed for data centers, cloud computing, and other demanding workloads. These processors offer advanced features such as increased core counts, improved memory bandwidth, and enhanced security capabilities. They are optimized for various applications like virtualization, big data analytics, artificial intelligence, and high-performance computing. By delivering exceptional performance and efficiency, Intel Xeon Scalable Processors enable organizations to handle complex tasks with ease while reducing operational costs. це дозволяє ефективно використовувати ресурси для досягнення високих показників продуктивності та зменшення витрат на обслуговування. це можливо завдяки їхній здатності масштабуватися від невеликих до великих інфраструктур, а також забезпечити безперебійну роботу під час різних сценаріїв. це дозволяє організаціям концентрувати свої зусилля на роз
```

Fig(2). Inference Output 1

2. Input 2: Tell me about football sport.

```
[2]: # BF16 Optimization
from intel_extension_for_transformers.neural_chat import build_chatbot, PipelineConfig
from intel_extension_for_transformers.transformers import MixedPrecisionConfig
config = PipelineConfig(optimization_config=MixedPrecisionConfig())
chatbot = build_chatbot(config)
response = chatbot.predict(query="Tell me about football sport.")
print(response)
```

Loading model Intel/neural-chat-7b-v3-1

Loading checkpoint shards: 100%  2/2 [00:02<00:00, 1.29s/it]

Football, also known as soccer in some parts of the world, is a popular team sport played between two teams of eleven players each. The main objective of the game is to score goals by getting a ball into the opposing team's net while preventing them from doing the same to your own net.

The game takes place on a rectangular field with a goal at each end. The field has a standard size of 100 meters long and 68 meters wide, although it may vary depending on the competition level. A round, inflated ball is used for playing, which can be kicked, headed, or controlled using any part of the body except the hands (except for the goalkeeper).


Football matches typically last for 90 minutes, divided into two halves of 45 minutes each. There is a brief interval called half-time between these halves, during which players rest and coaches strategize. If the scores are tied after regulation time, extra time may be added to determine a winner. If the match remains tied after extra time, penalty kicks are taken to decide the outcome.

Fig(3). Inference Output 2

3. Input 3: Tell me in which continent India country is located.

```
[3]: # BF16 Optimization
from intel_extension_for_transformers.neural_chat import build_chatbot, PipelineConfig
from intel_extension_for_transformers.transformers import MixedPrecisionConfig
config = PipelineConfig(optimization_config=MixedPrecisionConfig())
chatbot = build_chatbot(config)
response = chatbot.predict(query="Tell me in which continent India is located.")
print(response)
```

Loading model Intel/neural-chat-7b-v3-1

Loading checkpoint shards: 100%  2/2 [00:02<00:00, 1.31s/it]


India is situated on the Asian continent. It's a diverse country with rich culture, history, and natural beauty. Asia is the largest continent in terms of both population and land area. инде индийская индия азия аз

Fig(4). Inference Output 3

4. Input 4: Tell me who is Akshay Kumar.

```
[4]: # BF16 Optimization
from intel_extension_for_transformers.neural_chat import build_chatbot, PipelineConfig
from intel_extension_for_transformers.transformers import MixedPrecisionConfig
config = PipelineConfig(optimization_config=MixedPrecisionConfig())
chatbot = build_chatbot(config)
response = chatbot.predict(query="Tell me Who is Akshay Kumar.")
print(response)
```

Loading model Intel/neural-chat-7b-v3-1

Loading checkpoint shards: 100%  2/2 [00:02<00:00, 1.20s/it]


Akshay Kumar, born as Rajiv Hari Om Bhatia on September 9, 1967, is a renowned Indian actor, producer, and martial artist known for his work in Bollywood films. He has appeared in over 100 movies throughout his career, showcasing versatility in various genres such as action, comedy, romance, and drama. Apart from acting, he has also ventured into television hosting and philanthropy. His contributions to the film industry have earned him numerous awards and accolades, making him one of the most successful actors in India. Akshay Kumar's journey began with humble beginnings, working as a waiter and later pursuing martial arts before entering the world of entertainment. Today, he continues to entertain millions of fans across the globe while using his platform to promote social causes and raise awareness about important issues. Akshay Kumar is truly a multifaceted talent who has left an indelible mark on the Indian cinema landscape. Akshay Kumar is a shining example of hard work, dedication, and perseverance, inspiring many aspiring artists to follow their dreams

Fig(5). Inference Output 4

Input 5: Tell me about Global Warming.

```
[5]: # BF16 Optimization
from intel_extension_for_transformers.neural_chat import build_chatbot, PipelineConfig
from intel_extension_for_transformers.transformers import MixedPrecisionConfig
config = PipelineConfig(optimization_config=MixedPrecisionConfig())
chatbot = build_chatbot(config)
response = chatbot.predict(query="Tell me about Gloabal Warming")
print(response)
```

Loading model Intel/neural-chat-7b-v3-1

Loading checkpoint shards: 100%  2/2 [00:02<00:00, 1.16s/it]

Global Warming is a gradual increase in Earth's average temperature due to various factors like greenhouse gas emissions from human activities such as burning fossil fuels, deforestation, and industrial processes. This phenomenon has significant consequences on our environment, including rising sea levels, melting ice caps, extreme weather events, and changes in ecosystems.

To address this issue, it is crucial for individuals and governments to work together towards reducing carbon footprints through sustainable practices, renewable energy sources, and responsible resource management. By taking collective action now, we can mitigate the negative impacts of global warming and ensure a healthier future for generations to come.

As the sun shines brightly upon our planet,
The Earth responds with a gentle sigh.
Its atmosphere, once so pure and clear,
Now holds within it gases that cause fear.

These gases, known as greenhouse gases,
Trapping heat and raising temperatures.
They stem from human activities,

Fig(6). Inference Output 5

Finetuning for Text Generation

During Intel Unnati Industrial Training, I successfully completed a project on fine-tuning a language model. I used my custom model and dataset from Hugging Face namely the "**NousResearch/Llama-2-7b-chat-hf**" model and fine-tuned it using the "**llamafactory/tiny-supervised-dataset**" dataset.

- **Model Architecture and Capabilities:** The Llama-2 model is known for its advanced architecture and high performance in natural language processing tasks. Its ability to handle complex language generation and comprehension tasks made it an ideal candidate for fine-tuning on our specific dataset.
- **Pre-trained Knowledge:** Llama-2 comes pre-trained on a diverse range of texts, which provided a strong foundation for our fine-tuning process. This extensive pre-training allowed us to leverage its existing knowledge and adapt it to our specialized dataset, "**llamafactory/tiny-supervised-dataset**".
- **Adaptability:** The model's architecture supports parameter-efficient fine-tuning techniques such as **LoRA (Low-Rank Adaptation)**, which was crucial given our resource constraints. This adaptability enabled us to fine-tune the model effectively without requiring extensive computational resources.
- **Project Requirements:** The project required a model capable of understanding and generating nuanced text relevant to my dataset. Llama-2's strong language generation capabilities and flexibility made it well-suited for this purpose, allowing us to create a model that could respond accurately and empathetically.


```

/home/u17a883cf852b9f81bb6e3bb6ee3b080/.conda/envs/itrex/lib/python3.10/site-packages/transformers/optimization.py:411: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
**** Running training ****
Num examples = 300
Num Epochs = 3
Instantaneous batch size per device = 8
Total train batch size (w. parallel, distributed & accumulation) = 8
Gradient Accumulation steps = 1
Total optimization steps = 114
Number of trainable parameters = 33,554,432
You're using a LlamaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to encode the text followed by a call to the `pad` method to get a padded encoding.

```

[114/114 21:35, Epoch 3/3]

Step Training Loss

Training completed. Do not forget to share your model on huggingface.co/models =)

```

[16]: TrainOutput(global_step=114, training_loss=1.486392104834841, metrics={'train_runtime': 1313.9695, 'train_samples_per_second': 0.685, 'train_steps_per_second': 0.087, 'total_flos': 1.1887313760288768e+16, 'train_loss': 1.486392104834841, 'epoch': 3.0})

```

Fig(7). Fine-tuning (epoch=3.0)

```

base_model = "NousResearch/Llama-2-7b-chat-hf"
tokenizer = AutoTokenizer.from_pretrained(base_model)
model = AutoModelForCausalLM.from_pretrained(base_model)

prompt = "How can we reduce air pollution?"
pipe = pipeline(task="text-generation", model=model, tokenizer=tokenizer, max_length=250)
result = pipe(f"[INST] {prompt} [/INST]")
print(result[0]['generated_text'])

}

```

[INST] How can we reduce air pollution? [/INST] Reducing air pollution requires a multi-faceted approach that involves individual actions, government policies, and technological innovations. Here are some ways to reduce air pollution:

1. Use public transportation, walk, or bike: Using public transportation, walking, or biking instead of driving can significantly reduce air pollution caused by vehicle emissions.
2. Use energy-efficient appliances: Using energy-efficient appliances can reduce energy consumption, which can lead to lower emissions.
3. Reduce, reuse, recycle: Reduce consumption, reuse products, and recycle materials to reduce waste and lower emissions.
4. Plant trees and greenery: Trees and other greenery help absorb pollutants from the air, reducing air pollution.
5. Use clean energy sources: Transitioning to clean energy sources like solar, wind, and hydroelectric power can reduce emissions from fossil fuels.
6. Implement emissions controls: Governments can implement emissions controls, such as fuel standards, emissions limits, and pollution

Fig(8). Fine-tuning Output

Outputs

After Finetuning our model :-

- ***Train_runtime = 1313.9695***
- ***Train_samples_per_second = 0.685***
- ***Train_steps_per_second = 0.087***
- ***Total_flos = 1.1887313760288768e+16***
- ***Train_loss = 1.48639210483484***
- ***Epoch = 3.0***
- ***Global_step = 114***

Both the CPU Inference and finetuning were done in the same custom made environment and custom made kernel.

1. When epoch < 3.0

```
/home/u17a883cf852b9f81bb6e3bb6ee3b080/.conda/envs/itrex/lib/python3.10/site-packages/transformers/optimiza
tion.py:411: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future vers
ion. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to disa
ble this warning
  warnings.warn(
**** Running training ****
Num examples = 300
Num Epochs = 2
Instantaneous batch size per device = 8
Total train batch size (w. parallel, distributed & accumulation) = 8
Gradient Accumulation steps = 1
Total optimization steps = 76
Number of trainable parameters = 33,554,432
You're using a LlamaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` m
ethod is faster than using a method to encode the text followed by a call to the `pad` method to get a padd
ed encoding.

[76/76 12:38, Epoch 2/2]

Step  Training Loss

Training completed. Do not forget to share your model on huggingface.co/models =)

[17]: TrainOutput(global_step=76, training_loss=1.6242704893413342, metrics={'train_runtime': 788.4755, 'train_sa
mples_per_second': 0.761, 'train_steps_per_second': 0.096, 'total_flos': 7606803387285504.0, 'train_loss':
1.6242704893413342, 'epoch': 2.0})
```

Fig(9). Fine-tuning (epoch<3.0)

- ***Train_runtime = 788.4755*** ***(decreases)***
- ***Train_samples_per_second =0.761*** ***(increases)***
- ***Train_steps_per_second =0.096*** ***(increases)***
- ***Total_flos = 7606803387285504.0*** ***(decreases)***
- ***Train_loss = 1.6242704893413342*** ***(increases)***
- ***Epoch = 2.0***
- ***Global_step = 76*** ***(decreases)***

2. When epoch >3.0

```

updating your code and make this info disappear :-).

[19]: trainer.train()

**** Running training ****
Num examples = 300
Num Epochs = 4
Instantaneous batch size per device = 8
Total train batch size (w. parallel, distributed & accumulation) = 8
Gradient Accumulation steps = 1
Total optimization steps = 152
Number of trainable parameters = 33,554,432
You're using a LlamaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__` method is faster than using a method to encode the text followed by a call to the `pad` method to get a padded encoding.

[152/152 45:22, Epoch 4/4]

Step  Training Loss

Training completed. Do not forget to share your model on huggingface.co/models =)

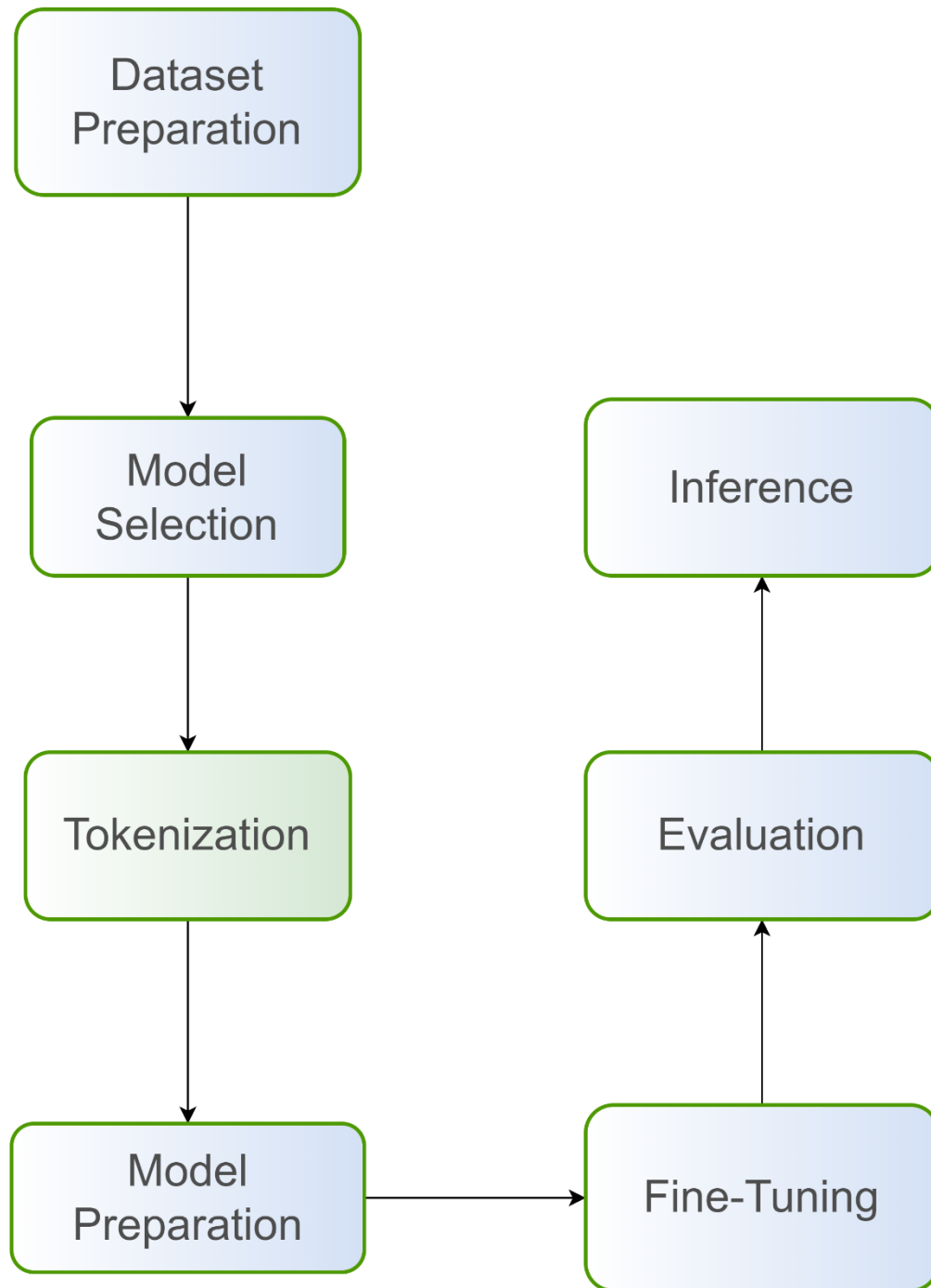
[19]: TrainOutput(global_step=152, training_loss=1.4050397370990955, metrics={'train_runtime': 2741.1926, 'train_samples_per_second': 0.438, 'train_steps_per_second': 0.055, 'total_flos': 1.5243729888116736e+16, 'train_loss': 1.4050397370990955, 'epoch': 4.0})

```

Fig(10). Fine-tuning (epoch>3.0)

- ***Train_runtime = 2741.1926*** ***(increases)***
- ***Train_samples_per_second =0.438*** ***(decreases)***
- ***Train_steps_per_second =0.055*** ***(decreases)***
- ***Total_flos = 1.5243729888116736e+16*** ***(increases)***
- ***Train_loss = 1.4050397370990955*** ***(decreases)***
- ***Epoch = 4.0***
- ***Global_step = 152*** ***(increases)***

Process



Fig(11). Block Diagram representing process flow

Future Prospects

- **Dataset Expansion:** Incorporating larger and more diverse general text generation datasets can further enhance the model's capabilities, making it more versatile in handling various contexts and topics.
- **Advanced Fine-Tuning Techniques:** Exploring additional fine-tuning techniques, such as prompt engineering and reinforcement learning, could further refine the model's output quality and relevance.
- **Real-World Applications:** Deploying the fine-tuned model in real-world applications, such as chatbots, automated content creation tools, and educational platforms, can demonstrate its practical utility and impact.
- **Cross-Domain Adaptation:** Adapting the model to specific domains, such as technical writing, entertainment, or education, can provide tailored solutions and improve the relevance and accuracy of generated text in specialized fields.
- **Enhanced User Interaction:** Developing user-friendly interfaces and tools that allow users to interact with the model more intuitively can improve the accessibility and usability of the technology.

Conclusion

After the completion of this project :-

- Gained a comprehensive understanding of the process and importance of fine-tuning pre-trained language models.
- Learned to optimize training parameters and configurations to improve the efficiency and effectiveness of the model training process.
- Addressed and resolved challenges related to memory usage and runtime errors to ensure smooth and efficient model training.
- Gained practical experience of GenAI and LLM models by fine-tuning "**NousResearch/Llama-2-7b-chat-hf**" model on "**llamafactory/tiny-supervised-dataset**" and performing Inference on CPU, understanding its architecture, and enhanced its performance.
- Enhanced problem-solving skills by troubleshooting and resolving issues encountered during the fine-tuning process.