## Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot

Intel® Unnati Industrial Training Project,2024

**By**

**19Bits**

**Joshua Sunny Ninan (B23CS1140)**

**Rohith N S (B23CS2156)**

**Avin Joy (B23CS2123)**

**Edwin K Mathew (B23CS1128)**

## 

Mentored by

Dr. Tessy Mathew

**MAR BASELIOS COLLEGE OF ENGINEERING & TECHNOLOGY**

**(Autonomous)**

**MAR IVANIOS VIDYA NAGAR, NALANCHIRA, THIRUVANANTHAPURAM, 695 015**

**MAY, 2024**

ABSTRACT

Large Language Models, or LLMs, are a type of Generative Artificial Intelligence (GenAI), and this research investigates their potential for commercial use. It looks into whether LLM inference can be performed on CPUs that are easily accessible and shows how to optimize an LLM to build a personalized chatbot.  
  
An overview of GenAI principles and LLM capabilities is provided in the report. After that, it looks at methods for effective LLM inference on CPU architectures, emphasizing the benefits of CPU-based solutions in business contexts. Lastly, the project shows how to refine a pre-trained LLM using a Stanford alpaca dataset to create a unique chatbot. The possible advantages and factors to take into account while implementing GenAI and customized chatbots in industrial settings are covered in the report's conclusion.

**INTRODUCTION:**

The frontiers of Artificial Intelligence (AI) are rapidly expanding, with Generative AI (GenAI) and Large Language Models (LLMs) leading the charge. GenAI pushes the boundaries of content creation, crafting entirely new forms of text, music, and even code. Imagine a system analyzing vast amounts of music to compose a fresh song inspired by the style. That's the power of GenAI! It devours massive datasets, learning the underlying patterns and relationships within the data. This allows GenAI models to transcend mere imitation, generating entirely novel outputs fueled by their training.

LLMs, a specialized form of GenAI, excel at manipulating and generating text. Think of them as incredibly powerful language experts. Trained on colossal amounts of text data, they perform impressive feats like generating human-quality text formats (news articles, code!), translating languages with remarkable accuracy, and summarizing lengthy documents. Imagine providing an LLM a simple prompt and receiving a complete news story! LLMs can even act as a powerful knowledge source, offering informative answers to your questions.

This report dives into the core principles of GenAI and LLM functionalities. This exploration will establish a foundation in these concepts, delve into the growing importance of CPU-based LLM inference methods, and showcase a practical application: fine-tuning pre-trained LLMs for specific tasks. This structured approach equips you with a comprehensive understanding of these groundbreaking AI tools.

**LITERATURE REVIEW**

Generative Artificial Intelligence (GenAI) and Large Language Models (LLMs):

Generative Artificial Intelligence (GenAI) has emerged as a potent tool in the field of Artificial Intelligence (AI), which has seen a notable movement towards generative models. Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two examples of approaches that fall under the umbrella of GenAI. These techniques have the ability to produce unique data points and learn underlying data distributions. Large Language Models (LLMs) have attracted a lot of interest in GenAI because of their capacity to process and produce text that is similar to that of a human.

To accomplish tasks like text production, translation, question answering, and code authoring, LLMs are essentially neural networks that have been trained on large volumes of text data. Research has demonstrated that LLMs can produce a wide range of realistic and coherent text formats, including emails, screenplays, musical compositions, and poems. Because of its adaptability, LLMs are being used in a variety of industries. Examples of these include automated customer support chatbots and content creation assistants for creative writing.

**Challenges and Considerations for Industrial Applications:**

Despite the remarkable capabilities of LLMs, deploying them in industrial settings presents two main challenges: computational cost and customization.

**Computational Cost:** Because of the intricate computations required, traditional LLM inference—running the model to produce outputs for fresh data points—heavily depends on strong Graphics Processing Units (GPUs). Although GPUs perform better, their high cost and power consumption prevent them from being used in industrial settings with limited resources.

**Customization:** Although adaptable, pre-trained LLMs might not be the best fit for certain industrial requirements. Large datasets that may not contain the domain-specific knowledge needed for commercial applications are used to train these models. This lack of customization may make LLMs less efficient at handling problems and duties unique to a certain sector.

**CPU-based LLM Inference:**

The limits of GPU-based inference have prompted studies on executing LLMs on Central Processing Units (CPUs) that are widely accessible. GPUs are more suitable for deep learning workloads than traditional CPU architectures. To optimize LLM inference for CPUs, however, recent research has looked into methods like quantization and knowledge distillation. While knowledge distillation includes moving knowledge from a pre-trained teacher model (often a GPU-based LLM) to a smaller student model (designed for CPU), quantization entails decreasing the precision of the model weights. Thanks to these methods, deploying LLMs in industrial settings is feasible as long as CPUs remain widely available and affordable.

**Fine-tuning LLMs for Chatbot Development**:

Customization gaps can be filled in part by optimizing previously trained LLMs. A pre-trained model is fine-tuned by applying it to a new, smaller dataset that is unique to the intended job. In the context of industrial applications, this can entail optimizing an LLM using a dataset that includes conversation data unique to a given industry in order to build a personalized chatbot. After that, the chatbot might be employed for customer service, question-answering, or even helping with process control in a manufacturing environment. Promising outcomes from research on optimizing LLMs for chatbot creation have opened the door for more specialized and efficient chatbots across a range of sectors.

**PROBLEM STATEMENT**

Problem Statement (PS-04): Introduction to GenAI and Simple LLM Inference on CPU and finetuning of LLM Model to create a Custom Chatbot

**METHODOLOGY**

**Methodology for Fine-Tuning TinyLLama on Intel Xeon SPR**

**1. Environment Setup**

**Objective:** Set up a Python environment with the necessary dependencies.

**Steps:**

1. **Create and activate a Conda environment:**

conda create -n itrex-1 python=3.10 -y

conda activate itrex-1

*This creates a new Conda environment named itrex-1 with Python 3.10 and activates it.*

1. **Install required Python packages:**

pip install intel-extension-for-transformers

**2. Cloning the Repository**

**Objective:** Clone the GitHub repository containing the fine-tuning scripts.

**Steps:**

1. **Clone the repository:**

git clone https://github.com/eternalflame02/Single-Node-FInetuning-of-Tiny-LLama-using-Intel-Xeon-SPR.git

1. **Navigate to the fine-tuning directory:**

cd./Single-Node-FInetuning-of-Tiny-LLama-using-Intel-Xeon-SPR/Fine Tuning/

**3. Installing Additional Dependencies**

**Objective:** Install additional dependencies required for fine-tuning.

**Steps:**

1. **Install dependencies from the requirements.txt file:**

pip install -r requirements.txt

1. **Install Jupyter and IPython kernel:**

python3 -m pip install jupyter ipykernel

python3 -m ipykernel install --name neural-chat --user

**4. Setting Up Hugging Face Authentication**

**Objective:** Authenticate with Hugging Face to access and download models.

**Steps:**

1. **Login to Hugging Face:**

huggingface-cli login

**5. Downloading Data**

**Objective:** Download the dataset required for fine-tuning.

**Steps:**

1. **Download the Alpaca dataset:**

Curl -O https://github.com/tatsu-lab/stanford\_alpaca/raw/main/alpaca\_data.json

**6. Fine-Tuning the Model**

**Objective:** Fine-tune the TinyLLama model using the downloaded dataset.

**Steps:**

1. **Import necessary modules:**

python

from transformers import TrainingArguments

from intel\_extension\_for\_transformers.neural\_chat.config import (

ModelArguments,

DataArguments,

FinetuningArguments,

TextGenerationFinetuningConfig,

)

from intel\_extension\_for\_transformers.neural\_chat.chatbot import finetune\_model

import os

1. **Define model, data, and training arguments:**

python

# Define model arguments

model\_args = ModelArguments(model\_name\_or\_path="TinyLlama/TinyLlama-1.1B-Chat-v1.0")

# Define data arguments

data\_args = DataArguments(train\_file="alpaca\_data.json", validation\_split\_percentage=1)

# Define training arguments

training\_args = TrainingArguments(

output\_dir='tinyllama',

overwrite\_output\_dir=True,

do\_train=True,

do\_eval=True,

per\_device\_train\_batch\_size=4,

per\_device\_eval\_batch\_size=4,

gradient\_accumulation\_steps=2,

learning\_rate=4e-5, # Adjust learning rate

weight\_decay=0.01, # Add weight decay

num\_train\_epochs=1, # Increase epochs for better training

save\_steps=500,

save\_total\_limit=2,

logging\_steps=100,

evaluation\_strategy="steps",

load\_best\_model\_at\_end=True,

seed=42, # Set random seed

bf16=True,

resume\_from\_checkpoint=True # Load from the latest checkpoint if finetuning fails after creating a checkpoint

)

# Define finetuning arguments

finetune\_args = FinetuningArguments()

# Create finetuning configuration

finetune\_cfg = TextGenerationFinetuningConfig(

model\_args=model\_args,

data\_args=data\_args,

training\_args=training\_args,

finetune\_args=finetune\_args,

)

1. **Start the fine-tuning process:**

python

finetune\_model(finetune\_cfg)

**7. Building the Chatbot**

**Objective:** Build a chatbot using the fine-tuned model.

**Steps:**

1. **Import necessary modules for building the chatbot:**

python

from intel\_extension\_for\_transformers.neural\_chat import build\_chatbot

from intel\_extension\_for\_transformers.neural\_chat import PipelineConfig

from intel\_extension\_for\_transformers.neural\_chat.config import LoadingModelConfig

1. **Create a pipeline configuration:**

python

config = PipelineConfig(

model\_name\_or\_path="./tinyllama",

loading\_config=LoadingModelConfig(

peft\_path="./tinyllama" # Path to the PEFT model (fine-tuned model)

)

)

1. **Build the chatbot instance:**

python

chatbot = build\_chatbot(config)

1. **Generate a response to a query:**

python

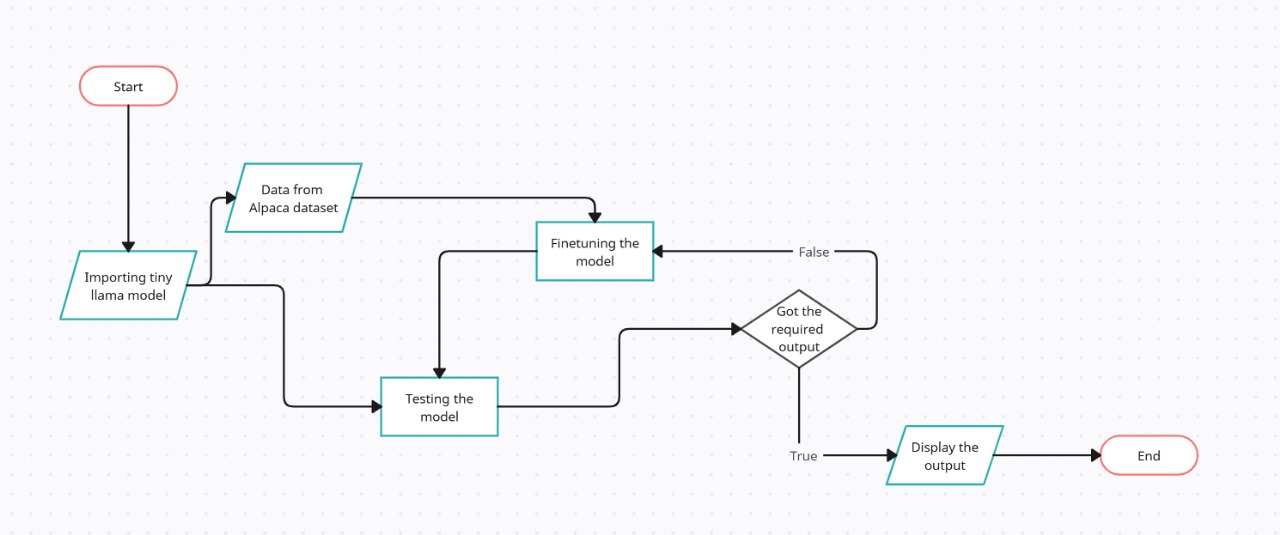
query = "Tell me about AI."

response = chatbot.predict(query=query)

# Print the generated response

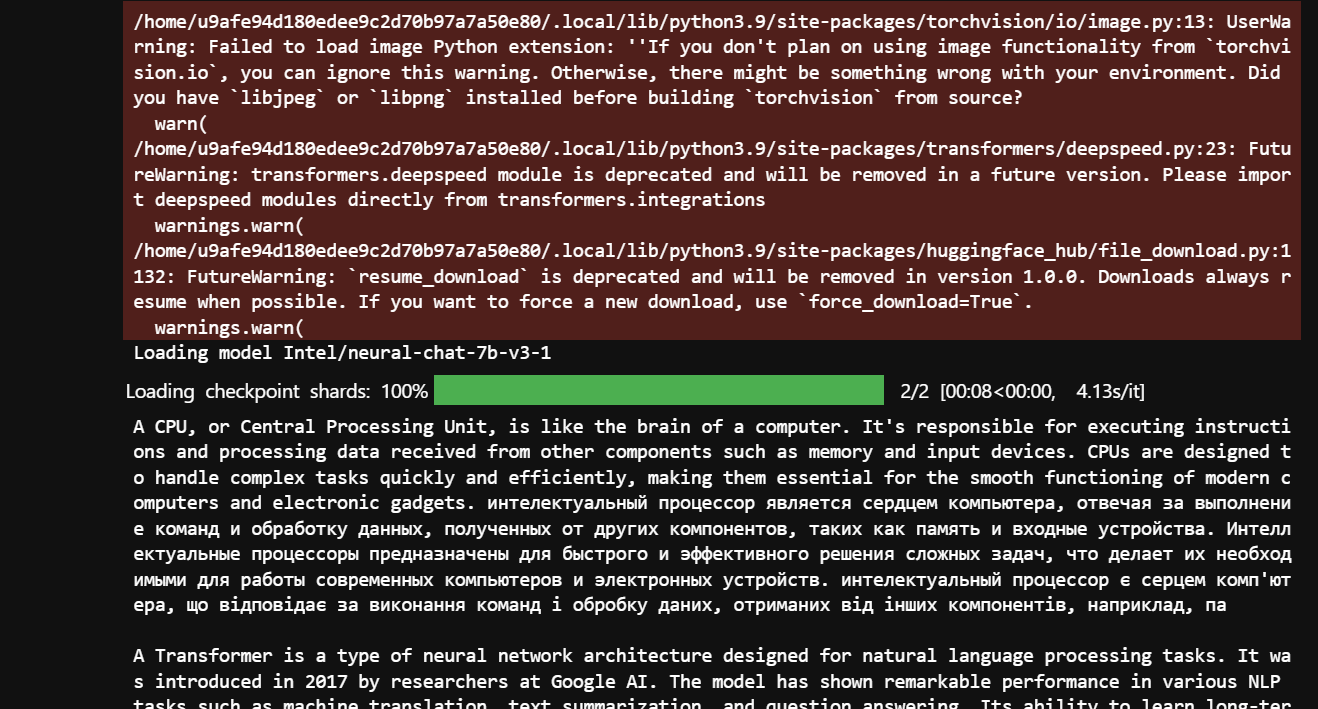
print(response)

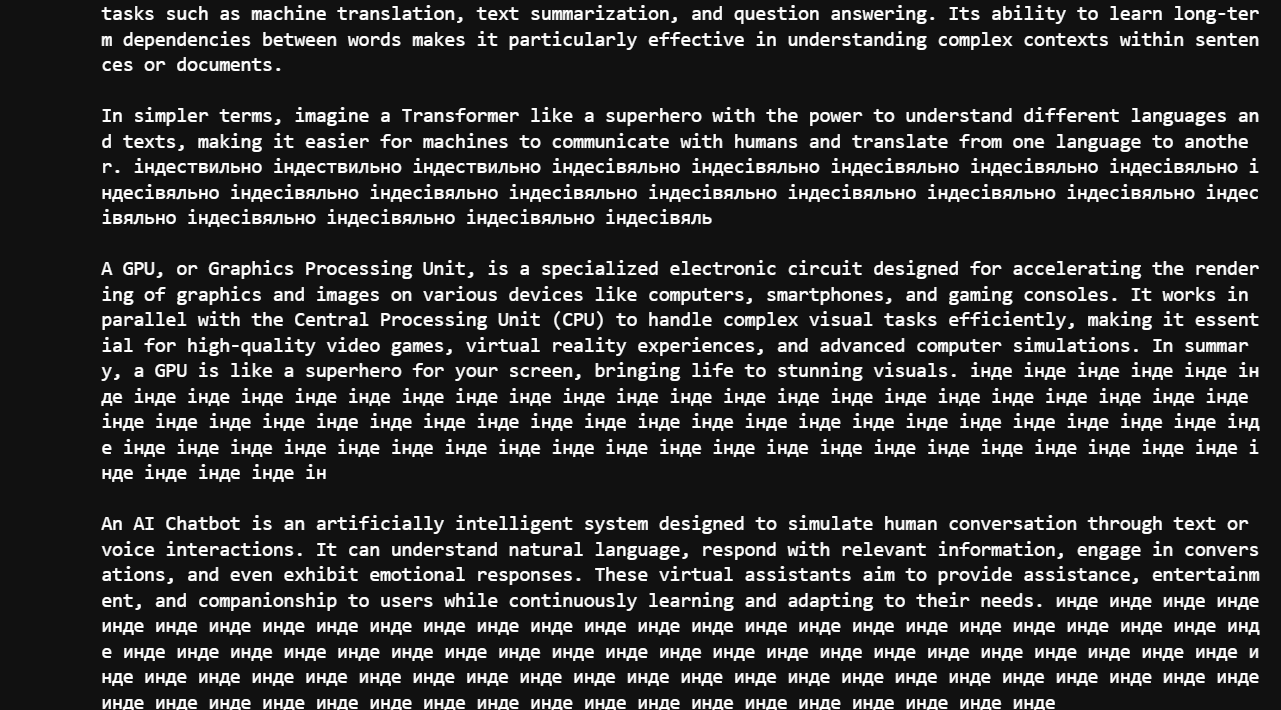
**DATA FLOW DIAGRAM**

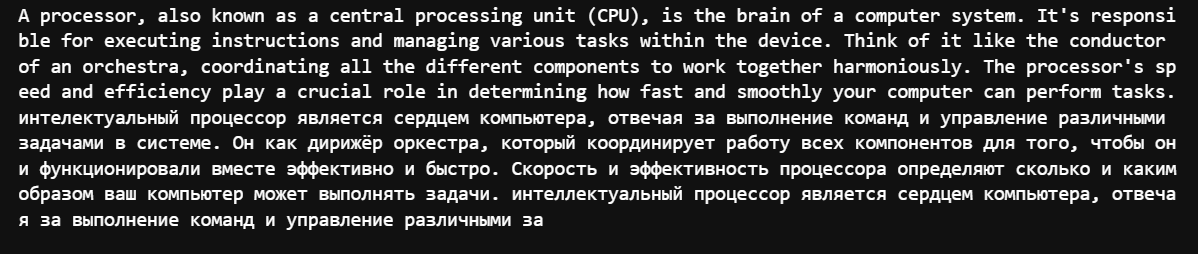


**RESULTS:**

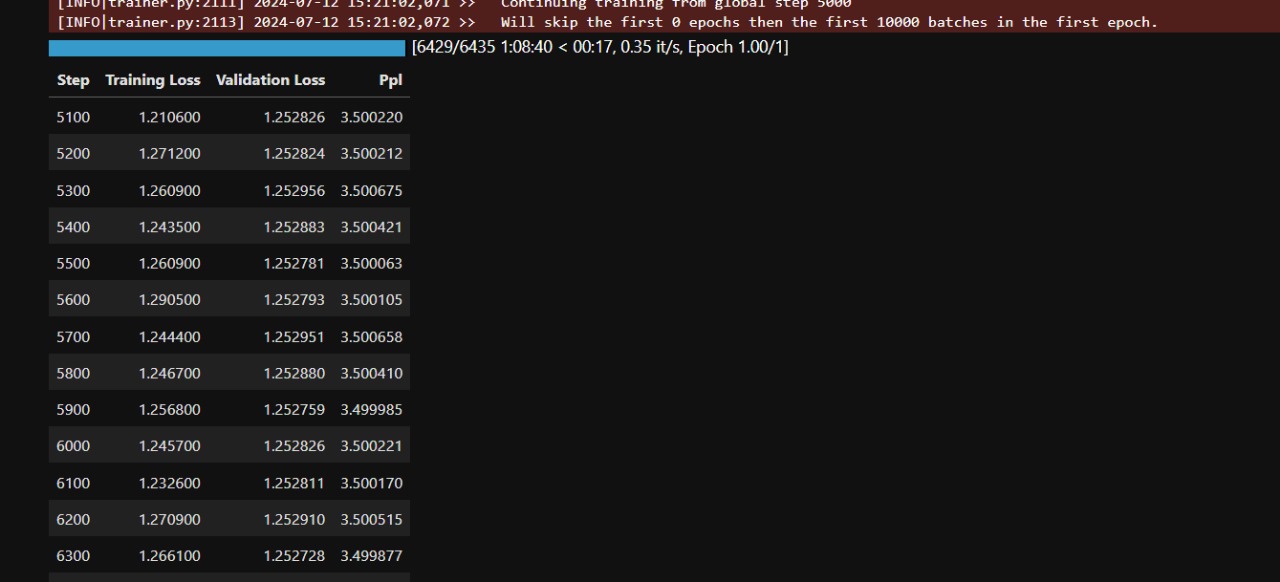
* **The fundamental concepts of Transformers, Generative AI, and its applications are understood.**
* This project involved performing a simple LLM inference on a CPU and understood the process of fine-tuning LLMs for custom applications.
* Result from the **build\_chatbot\_on\_spr.ipynb** notebook :

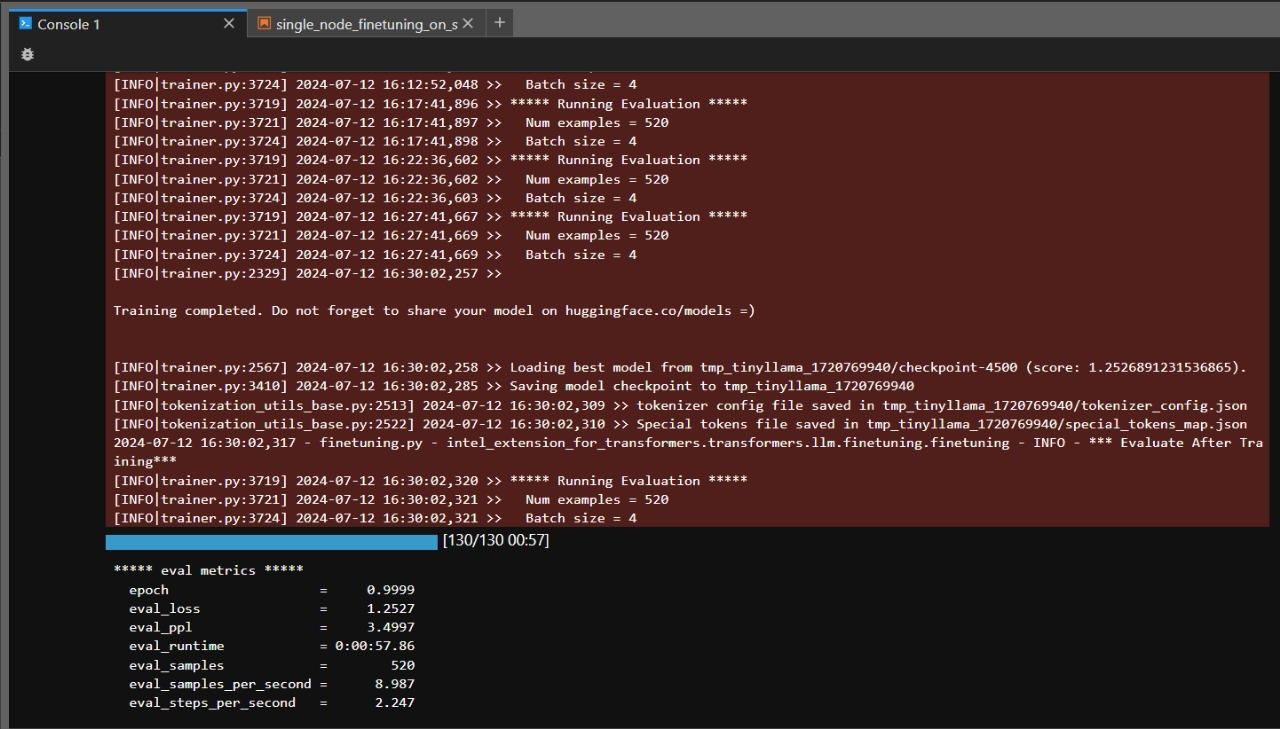






* Result from **single\_node\_finetuning\_on\_spr.ipynb** notebook





CONCLUSION

This project explored the potential of Generative AI (GenAI), specifically Large Language Models (LLMs), for industrial applications. A successful simple LLM inference was achieved on a CPU, showcasing the potential for LLM deployment in resource-constrained environments. Additionally, we gained a practical understanding of the fine-tuning process, culminating in the development of a custom chatbot.

The findings suggest that CPU-based LLM inference, coupled with fine-tuning techniques, can bridge the gap between cutting-edge GenAI technology and practical industrial applications. This approach offers a cost-effective and accessible solution for leveraging the power of LLMs in various industrial settings.

1. REFERENCES

[Hugging Face – The AI community building the future.](https://huggingface.co/)

<https://github.com/intel/intel-extension-for-transformers/>

<https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0>

<https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>