Intel Unnati Industrial Training Report

Problem Statement: Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU and fine-tuning of LLM Models using Intel® OpenVINO™

Date:- 11/07/2024

Submitted By:- **Tejasvee Dwivedi**
Registration Number:- 210968158
Learner ID:- tejasvee.Dwivedi@learner.manipal.edu
Mentor:- Mr. Manjunath Vishweshwar Hegde (MAHE-MIT)

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Problem Statement

- Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU and fine-tuning of LLM Models using Intel® OpenVINO™.
- This problem statement is crafted to introduce the fascinating domain of Generative Artificial Intelligence (GenAI) through a series of interactive exercises. We need to run a pretrained transformer conduct basic Large Language Model (LLM) inference on a CPU, perform inference with OpenVINO, and develop a personalized Chatbot.
- LLMs or Large Language Models use a structure called as transformers which is a type or neural network architecture which is the foundation for many state-of-the-art Natural Language Processing models.
- Many popular LLMs include-
    - GPT3 (Generative Pre-Training V3)
    - T5 (Made by google Text-To-Text Transfer Transformer)
    - BERT (Bidirectional Encoder Representation from Transformers)

Challenges:-

1. Large Model File Sizes: Pre-trained language models often have large file sizes, necessitating considerable storage space and memory for loading and running them efficiently.

2. Learning LLM Inference on CPU: Mastering the process of running large language models (LLMs) on CPUs, including optimizing performance and resource usage, can be complex.

Unique Idea Brief (Solution)

Intel OpenVINO is an open-source toolkit used for accelerating, optimizing and deploying LLMs (Large Language Models) on the edge and on the cloud irrespective of the hardware present on the computing machines.

It also allows very large language models which require a lot of memory and space to be run locally on any intel machine.

Using OpenVINO to optimize and run the language model can significantly affect CPU usage in the following ways:

Imagine you have a traditional model and an optimized model:

- **Traditional Model**: Requires a lot of CPU power to process data, causing high CPU usage and potentially slower response times. And we have to shift towards dedicated GPUs.
- **Optimized Model with OpenVINO**: Uses less CPU power by making the computations simpler and more efficient, resulting in lower CPU usage and faster response times.

The **Tiny LLaMA 1B chat model** is used by us as it balances performance and efficiency, providing robust conversational capabilities while being lightweight enough to run on less powerful hardware. It offers faster inference times and requires less memory, making it suitable for applications where resources are limited. Additionally, its smaller size facilitates easier deployment and integration into various systems. However, the trade-off is that it may not perform as well as larger models in terms of generating highly nuanced or contextually rich responses.

Model Size: Approximately 1 billion parameters.

Using INT4 and INT8 precision for better compression and less load over the CPU.

Features Offered

Tiny Llama 1b Chat model-

- **Model Size**: Approximately 1 billion parameters.
- **Inference Speed**: Optimized for faster inference on CPUs, achieving speeds suitable for real-time applications.
- **Memory Usage**: Requires significantly less memory compared to larger language models, enhancing efficiency on resource-constrained devices.
- **Performance**: Demonstrates competitive performance metrics in conversational AI tasks, balancing accuracy with computational efficiency.
- **Deployment Time**: Quick deployment due to its smaller size and optimized architecture, facilitating integration into various applications.

OpenVINO-

Model Quantization: Implementation of various model quantizations (INT8, INT4) to reduce model size and improve performance.

Without OpenVINO**:**

- Running a high-precision model (e.g., FP32) directly on the CPU might result in high CPU usage because the model computations are more complex and resource-intensive.

With OpenVINO:

- By using INT8 or INT4 compressed models and optimized inference with OpenVINO, the same tasks can be performed with lower CPU usage. This means the CPU can handle more requests simultaneously or perform other tasks while running the model.

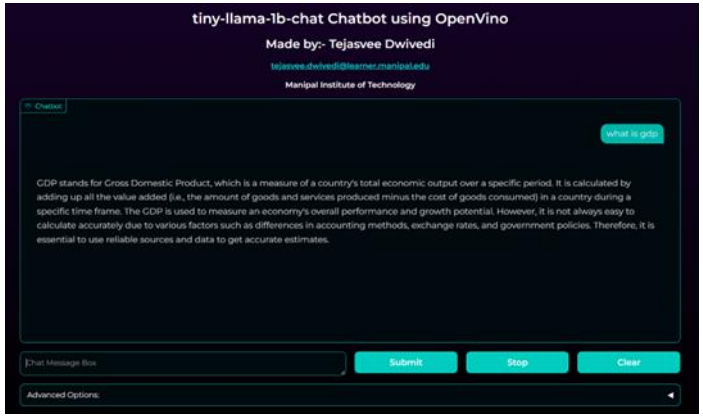
The conversation is stored in such a way that the context of the topics is retained and the model may be able to answer the questions with respect to the details provided along with it and not just the answer.

Since we will be deploying our model using gradio thus it will have a minimal UI, just simply type the question which you want to ask and then you will get the answer generated by the model

Process flow

Steps Involved:-

1. Configuring and Fetching the LLM Configuration File.
2. Importing SUPPORTED_LLM_MODELS and Selecting Model Language.
3. Generating and Executing Commands for Model Conversion to INT8 and INT4.
4. Setting up OpenVINO configuration using certain parameters.
5. Setting up Conversational AI with OpenVINO Model Integration.
6. Setting up Gradio interface, to use the chatbot.



Process flow

Setting Up Environment and Installing Required Packages:
The notebook outlines the installation of essential packages, including OpenVino, PyTorch, and other dependencies required for setting up the environment.

Configuring and Fetching the LLM Configuration File:
It includes a script to fetch and configure the LLM (Large Language Model) configuration file from a specified URL or a local path. This ensures the model has the necessary configuration to run correctly.

Preparing Model and Tokenizer:
Instructions are provided to load the model and tokenizer from pre-trained weights. The notebook uses the transformers library for this purpose.

Configuring Chatbot Interface using Gradio:
The main functionality involves creating a chatbot interface using Gradio, a library for creating user interfaces for machine learning models. It covers setting up the input and output fields, and various controls such as temperature, top-k sampling, and repetition penalty.

Launching the Gradio Interface:
Finally, the notebook includes code to launch the Gradio interface, allowing users to interact with the chatbot. There are options to stop the chatbot, clear the conversation, and customize various parameters for text generation.

Architecture Diagram

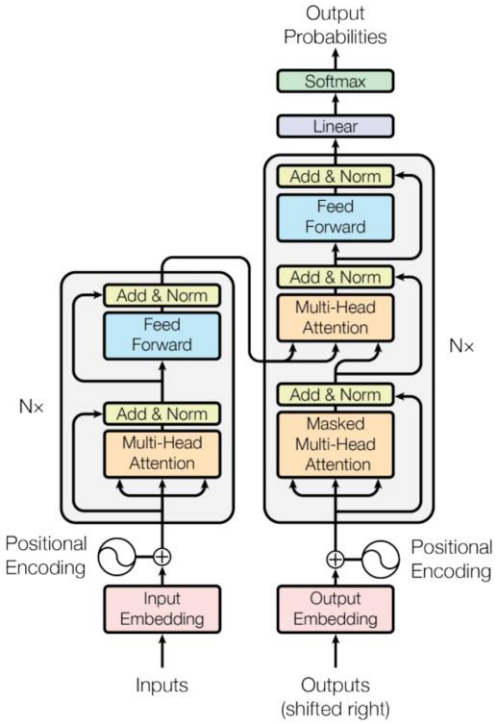
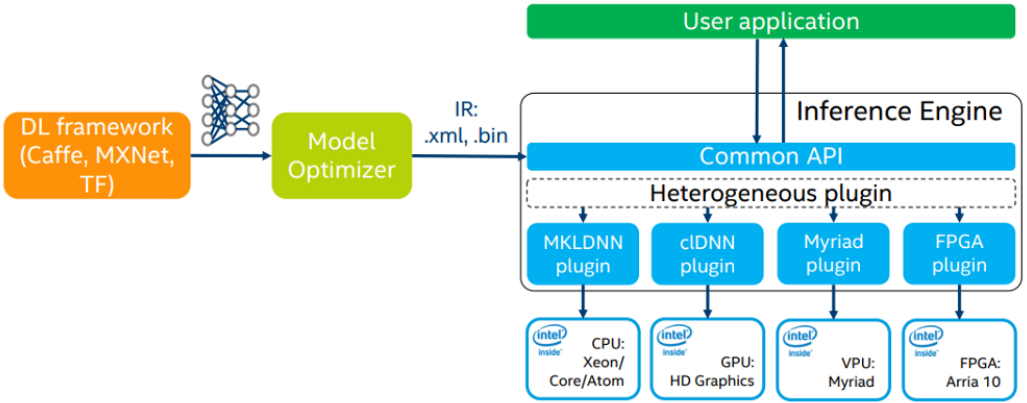


Figure 1: The Transformer - model architecture.



OpenVINO inference engine works by converting the deep learning model into an intermediate representation that the openVINO API can understand

Technologies used

- We are using intel **OpenVINO** an open source toolkit which allows optimization and quantization of large language models such that these heavy models can be ran on legacy hardware and on the edge.
- We are using **optimum-intel** and **NNCF** for weight compression and to export our model with INT8 precision
- We are using jupyter notebook and **python** language to run and interface with the model
- The main framework which is the backbone to this project is **hugging face**, **hugging face hub** and **hugging face pipeline**. Using all these frameworks we were able to build our model and bind a flow of control to the model.
- We are also using **tokenizer** and **detokenizer** to form the correct input and output generated by the model.
- INT8 and INT4 Formats: Converting the model to lower-precision formats (INT8 and INT4) reduces the computational complexity of the model. This means the model requires fewer CPU resources to perform the same tasks, resulting in lower CPU load during inference
- We have used **github** to store and maintain our code.
- We are using **Gradio,** which is a framework used to build GUIs over the internet.

Team members and contribution:

Individual Project

Made By- **Tejasvee Dwivedi (210968158)**

Mentor- **Mr. Manjunath Vishweshwar Hegde (MAHE-MIT)**

References-

1. Intel Github Repositories:- **https://github.com/openvinotoolkit/openvino**
2. Intel Dev Cloud for Edge**:-https://www.intel.com/content/www/us/en/developer/tools/devcloud/edge/overview.html**
3. **https://huggingface.co/Intel/neural-chat-7b-v3-3**
4. Gradio themes**:- https://huggingface.co/spaces/gradio/theme-gallery**
5. Python Langchain**:- https://python.langchain.com/v0.2/docs/integrations/llms/openvino/**
6. YouTube:-https://www.youtube.com/@AbhishekNandy/videos
7. **https://medium.com/@abhishek.nandy81**

Conclusion

Environment Setup: The notebook provides a thorough guide for setting up the environment and installing necessary packages, ensuring that users can replicate the setup on their machines.

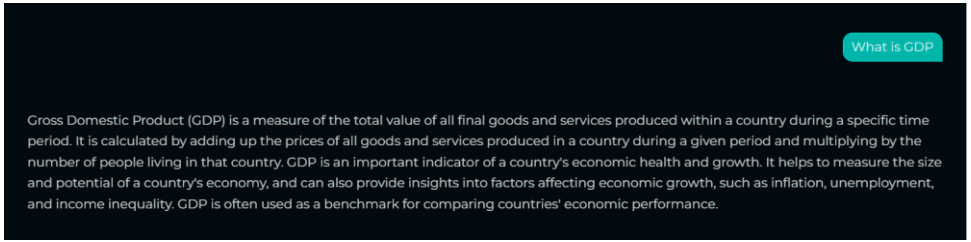
Model Configuration: It highlights the importance of fetching and configuring the model correctly, which is crucial for running the chatbot effectively.

User Interaction: By utilizing Gradio, the notebook offers an interactive way for users to engage with the chatbot, making it user-friendly and accessible.

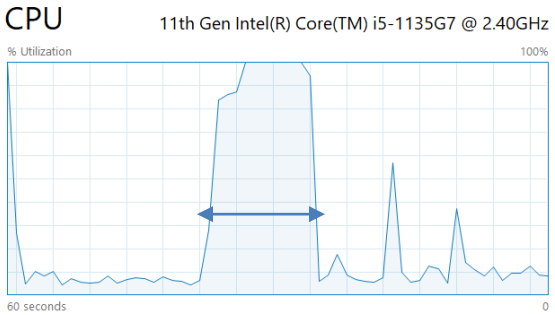
Customization: The provided controls for text generation parameters demonstrate the flexibility of the chatbot, allowing users to fine-tune the output according to their preferences.

Conclusion

Increased CPU usage-



Prompt entered in chatbot



CPU usage while the answer is generated

Images of the deployed chatbot