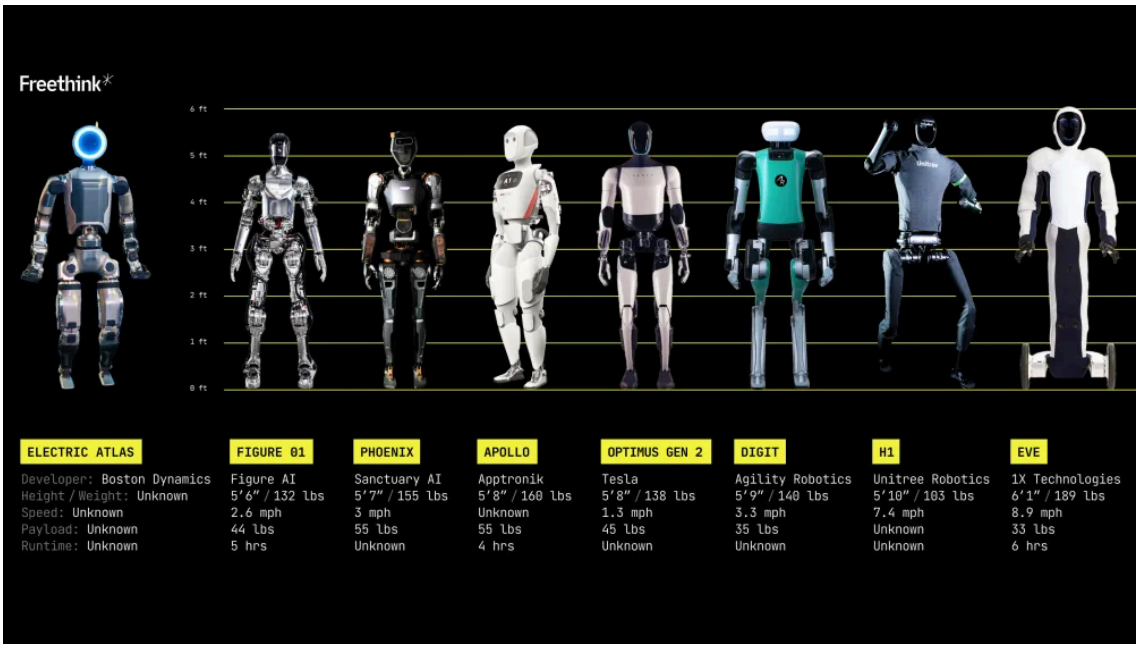
Master the Future of Tech: Become a Certified Cloud Native Applied Generative AI Engineer



Build Custom GPTs, AI Agents, Humanoids, and Fine-Tune LLMs

Version: 12 (Implementation and adoption starting from July 1, 2024)

Today's pivotal technological trends are Cloud Native (CN), Generative AI (GenAI), and Physical AI. Cloud Native technology offers a scalable and dependable platform for application operation, while AI equips these applications with intelligent, human-like capabilities. Physical AI aims to bridge the gap between digital intelligence and physical capability, creating systems that can understand and interact with the world in a human-like manner. Our aim is to train you to excel as a Cloud Native Applied Generative and Physical AI developer globally.

The Cloud Native Applied Generative AI Certification program equips you to create leading-edge Cloud Native AI and Physical AI solutions using a comprehensive cloud-native, AI, and Physical AI platform.

Everything will soon be represented by a conversational interface, or to put it another way, a personal AI, we will cover it extensively in this program. () Currently, OpenAI Custom GPT Platform is the best platform to develop personal AI.

We will also be covering AI agents, which are autonomous programs or entities that perceive their environment through sensors, process this information, and take actions to achieve specific goals or tasks. They can operate independently, adapt to

changing conditions, and make decisions based on their observations and objectives.

Material to Understand the Coming AI Age:

- Watch the Overview Video of Our Program
- Watch Short Video
- What Is an AI Anyway? Mustafa Suleyman The Coming Wave: Technology, Power, and the 21st Century's Greatest Dilemma
- The Worlds I See: Curiosity, Exploration, and Discovery at the Dawn of AI
- Ethan Mollick's Substack
- David Autor Lecture
- Conversation between Suleyman, Yuval Noah Harari, and Zanny Minton Beddoes

This one and a half year program equips you with the skills to thrive in the age of Generative AI (GenAI), Physical AI, and cloud native computing (CN). You will become an expert Custom GPT, AI Agent, and Humanoid Robotics Developer.

Why This Program?

- **Cutting-Edge Skills:** Develop in-demand skills to build intelligent, scalable cloud applications using Generative AI and Cloud Native technologies.
- **Industry-Ready:** Prepare for global certifications, startup and freelance opportunities after just six months.
- **Future-Proof Your Career:** Stay ahead of the curve in a rapidly evolving tech landscape.

What You'll Learn:

- **Custom GPTs and Multi AI Agent Systems:** Learn to fine-tuning foundational AI models, and market them in GPT stores. Learn key principles of designing effective AI agents, and organising a team of AI agents to perform complex, multi-step tasks. Apply these concepts to automate common business processes.
- **Develop AI Powered Microservices:** Master Python, build APIs using FastAPI, SQLModel, Postgres, Kafka, Kong, and leverage cutting-edge GenAI APIs like OpenAI, and Open Source AI LLMs.
- **Cloud Native Expertise:** Design and deploy cloud-native applications using Docker, DevContainers, TestContainers, Kubernetes, Terraform, and GitHub Actions.
- **Distributed System Design**: Designing systems that run on multiple computers (or nodes) simultaneously, interacting and coordinating their actions by passing messages over a network.

- **Designing AI Solutions using Design Thinking and Behaviour Driven Development (BDD)**: We will learn to leverage these methodologies to create AI solutions that are not only technically sound but also highly user-centric and aligned with real-world needs.
- **Fine-Tuning Open-Source Large Language Models using PyTorch, and Fast AI**: We will learn to fine-tuning of open-source Large Language Models (LLMs) like Meta LLaMA 3 using PyTorch and Fast AI, with a focus on cloud-native training and deployment. We will set up development environments, preprocess data, fine-tune models, and deploy them using cloud native platforms.
- **Physical AI and Humanoid Robotics**: We will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions.

Flexible Learning:

- **Earn While You Learn:** Start freelancing or contributing to projects after the third quarter.

Program Structure (4+2 = 6 Quarters):

Foundation Level

- **Quarter 1: Fundamentals of Prompt Engineering, Docker, GitHub, and Modern Python Programming:**

  We begin the course by understanding the basics of GenAI and Prompt Engineering. Then we will understand the basics of Linux, Docker, VSCode, Devcontainer, and GitHub. The main focus will be on mastering the fundamentals of Modern Python with Typing, the go-to language for AI.

  - **Certification:**
    - Certified Professional Python Programmer (CPPP1)

  Learning Repo:
  https://github.com/panaversity/learn-cloud-native-modern-python

- **Quarter 2: Prompt Engineering, Developing Custom GPTs and Multi AI Agent Systems:**

  With this course, you'll start by building a strong understanding of generative AI and learn how to apply Large language models (LLMs) and diffusion models practically. We will introduce a set of principles known as prompt engineering, which will help developers to work efficiently with AI. Learn to create custom AI models and GPTs using OpenAI, Azure, and Google

technologies. Use open source libraries, like Langchain, CrewAI, and LangGraph to automate repeatable, multi-step tasks and automate business processes that are typically done by a group of people.

Certifications:

- Microsoft Certified: Azure AI Engineer Associate
- Certified crewAI Engineer

Learning Repo:
https://github.com/panaversity/learn-prompt-eng-gpts-ai-agents

- **Quarter 3: Cloud Native AI Powered Microservices Design, Development, and Deployment:**

  Build scalable AI Powered APIs using FastAPI, Postgres, Kafka, Kong, GenAI APIs like OpenAI Chat Completion APIs, Assistant APIs, LangChain and Open Source AI LLMs, develop them using Containers and Dev Containers, and deploy them using Docker Compose locally and Kubernetes Powered Serverless Container Services on the cloud.

  We will also learn to integrate design thinking and Behavior-Driven Development (BDD) in developing AI systems. We will learn to create AI solutions that are deeply aligned with user needs and expectations. Design thinking ensures a thorough understanding of the user and problem space, while BDD provides a structured approach to defining and validating the desired behaviours of the AI system. Together, these methodologies lead to the development of AI solutions that are not only technically robust but also highly user-centric and effective in solving real-world problems.

  - **Certifications:**
    - PostgreSQL 13 Associate Certification
    - Confluent Certified Developer for Apache Kafka (CCDAK)
    - Design Thinking Professional Certificate (DTPC)
    - Test and Behavior Driven Development (TDD/BDD)

  Learning Repo:

  https://github.com/panaversity/learn-cloud-native-ai-powered-microservices/

We Will Be Using Microsoft Azure as our Default Cloud Platform

Amazon is still the cloud king based on market share. But many analysts agree: In the battle for the cloud, AI is now a game-changer — and Amazon's main competitors, particularly Microsoft, have the momentum.

In our program we will be using Azure as our default provider for teaching and deployment. We will be using using these services:

Get a free Azure Account now:
https://azure.microsoft.com/en-us/free
Note: Use GitHub Account to start an Azure free trial

Azure Container Apps (We will Start from this service using Dapr and Keda)
https://azure.microsoft.com/en-us/products/container-apps
Get started with the free tier: The first 180,000 vCPU per second, 360,000 GiB/s, and 2 million requests each month are free.
Watch: https://www.youtube.com/watch?v=0HwQfsa03K8
Deploy:
https://learn.microsoft.com/en-us/azure/container-apps/code-to-cloud-options

Azure Container Registry
https://azure.microsoft.com/en-us/products/container-registry/

Deploy to Azure Container Apps with GitHub Actions
https://learn.microsoft.com/en-us/azure/container-apps/github-actions

Azure Kubernetes Service (AKS)
https://azure.microsoft.com/en-us/products/kubernetes-service

GitHub
https://azure.microsoft.com/en-us/products/github/

GitHub Actions for AKS
https://learn.microsoft.com/en-us/azure/aks/kubernetes-action

Azure OpenAI Service
https://azure.microsoft.com/en-us/products/ai-services/openai-service

Azure Database for PostgreSQL
https://azure.microsoft.com/en-us/products/postgresql/

Kafka

https://cloudatlas.me/5-different-ways-you-can-run-apache-kafka-on-azure-973a18925ac7

Advance Level

- **Quarter 4: Generative AI with PyTorch:**

  Generative AI tools like ChatGPT, Gemini, and DALL-E have revolutionised our professional landscape. This hands-on course, “Master Generative AI with PyTorch,” guides you through the exciting process of building and training AI models using Python and the versatile, open-source PyTorch framework, all with the hardware you already have. You’ll delve into the core concepts of Generative Adversarial Networks (GANs), Transformers, Large Language Models (LLMs), variational autoencoders, diffusion models, LangChain, and more. Along the way, you’ll gain practical experience and a deep understanding of these cutting-edge technologies.

  Learning Repo: https://github.com/panaversity/genai-with-pytorch

- **Quarter 5: Fine-Tuning Open-Source Large Language Models:**

  This comprehensive course is designed to guide learners through the process of fine-tuning open-source Large Language Models (LLMs) such as Meta LLaMA 3 using PyTorch and Fast AI, with a particular emphasis on cloud-native training and deployment. The course covers everything from the fundamentals to advanced concepts, ensuring students acquire both theoretical knowledge and practical skills.

  The journey begins with an introduction to LLMs, focusing on their architecture, capabilities, and the specific features of Meta LLaMA 3. Students will also set up their development environment, including tools like Anaconda, Jupyter Notebooks, PyTorch, and Fast AI, to prepare for hands-on learning.

  Next, the course dives into PyTorch fundamentals, teaching students how to perform basic operations with tensors and build simple neural networks. This foundation is crucial for understanding the mechanics behind LLMs. The Fast AI library is introduced subsequently, highlighting its powerful data block API and tools for building and fine-tuning models. Practical sessions are integrated to ensure that students can apply these concepts effectively in real-world scenarios.

Data preparation is a crucial aspect of training models. The course covers comprehensive data collection and preprocessing techniques, such as tokenization and text normalisation. These steps are essential for preparing datasets suitable for fine-tuning LLMs like Meta LLaMA 3. Through practical exercises, students learn how to handle and preprocess various types of text data, ensuring they can prepare their datasets for optimal model performance.

Fine-tuning Meta LLaMA 3 with PyTorch forms a significant part of the course. Students will delve into the architecture of Meta LLaMA 3, learn how to load pre-trained models, and apply fine-tuning techniques. The course covers advanced topics such as regularisation and optimization strategies to enhance model performance. Practical sessions guide students through the entire fine-tuning process on custom datasets, emphasising best practices and troubleshooting techniques.

Building on these skills, the course then focuses on utilising Fast AI for NLP tasks. Students will learn how to leverage Fast AI's user-friendly interfaces and robust features to fine-tune Meta LLaMA 3 efficiently. Hands-on exercises ensure that students can effectively integrate Fast AI tools into their workflows, making the fine-tuning process more streamlined and accessible.

A critical aspect of this course is its focus on cloud-native training and deployment. Students will gain an understanding of cloud-native principles and infrastructure, setting up cloud environments on platforms like AWS, GCP, or Azure using Kubernetes. The course teaches how to train models using cloud resources, employing tools like Kubernetes and Kubeflow to optimise and distribute training workloads. Furthermore, students learn how to deploy models using Docker and Kubernetes, set up monitoring and maintenance tools, and ensure their models are scalable and efficient.

To round off the learning experience, the course includes an in-depth segment on exporting models for inference and building robust inference pipelines. Students will deploy models on cloud platforms, focusing on practical aspects of setting up monitoring tools to maintain model performance and reliability.

The course culminates in a capstone project, where students apply all the skills they have learned to fine-tune and deploy Meta LLaMA 3 on a chosen platform. This project allows students to demonstrate their understanding and proficiency in the entire process, from data preparation to cloud-native deployment.

Learning Repo:

https://github.com/panaversity/learn-fine-tuning-llms

- **Quarter 6: Physical AI and Humanoid Robotics Development:**

Artificial intelligence (AI) has experienced remarkable advancements in recent years. However, the future of AI extends beyond the digital space into the physical world, driven by robotics. This new frontier, known as “Physical AI,” involves AI systems that can function in the real world and comprehend physical laws. This marks a notable transition from AI models confined to digital environments. Humanoid robots are poised to excel in our human-centred world because they share our physical form and can be trained with abundant data from interacting in human environments.

This course provides an in-depth exploration of humanoid robotics, focusing on the integration of ROS 2 (Robot Operating System), Open Source Meta Llama 3, and OpenAI technologies. Students will learn to design, simulate, and deploy advanced humanoid robots capable of natural interactions. The curriculum covers essential topics such as ROS 2 for robotic control, simulations with Gazebo and Unity, and using OpenAI’s GPT models for conversational AI. Through practical projects and real-world applications, students will develop the skills needed to drive innovation in humanoid robotics.

Learning Repo:
https://github.com/panaversity/learn-physical-ai-humanoid-robotics

Horizontal Specialization Level

- **Quarter 7: Cloud Native Microservices Deployment and Distributed System Design (Optional):**

  Master Kubernetes, Terraform, and GitHub Actions to deploy your AI APIs and microservices in the cloud. We will cover distributed system design involving creating systems that are distributed across multiple nodes, focusing on scalability, fault tolerance, consistency, availability, and partition tolerance.

  Certifications:

  - Certified Kubernetes Application Developer (CKAD)
  - HashiCorp Certified: Terraform Associate

  Learning Repo:
  https://github.com/panaversity/learn-cloud-native-microservices-deployment

- **Quarter 8: Front-end Web GUI Development using Next.js and TypeScript (Optional):**

Next.js is designed to handle complex front-end applications well, making it a good fit for AI applications that might grow in features and data usage over time. Next.js offers features like API routes and file-based routing, which can streamline development for AI applications that need to interact with backend APIs and manage different application views. While Next.js and TypeScript aren't the only options for building AI application frontends, their focus on performance, scalability, and developer experience makes them a compelling choice for many developers.

Learning Repo:
https://github.com/panaverse/learn-nextjs

In 2015, Klaus Schwab, founder of the World Economic Forum, asserted that we were on the brink of a “Fourth Industrial Revolution,” one powered by a fusion of technologies, such as advanced robotics, artificial intelligence, and the Internet of Things.

“[This revolution] will fundamentally alter the way we live, work, and relate to one another,” wrote Schwab in an essay published in Foreign Affairs. “In its scale, scope, and complexity, the transformation will be unlike anything humankind has experienced before.”

Generative AI is set to revolutionise our daily lives and work environments. According to McKinsey & Company, generative AI could contribute an annual economic value of $2.6 trillion to $4.4 trillion across various sectors by enhancing automation, bolstering decision-making, and providing personalised experiences.

Investor Cathie Wood predicts that the market for humanoid robots could grow to $1 trillion by 2030.

Cloud native is an approach in software development that enables application creation, deployment, and management in cloud environments. It involves constructing applications as a collection of small, interconnected services known as microservices, a shift from traditional monolithic structures. This modular approach enhances the agility of cloud-native applications, allowing them to operate more efficiently with fewer resources.

Cloud Native has already been adopted by the majority of the companies, by 2024, more than 90% of global organisations will be running containerized applications in production. The adoption of Docker and Kubernetes has seen significant growth over recent years. As of 2022, about 61% of organisations reported using Kubernetes for container orchestration. This number has been steadily increasing as more

companies realise the benefits of these technologies for managing containerized applications [OBJ] [OBJ].

Technologies such as Kubernetes, Docker, serverless containers, APIs, SQL Databases, and Kafka support developers in swiftly constructing cloud-native applications. These tools offer a standardised platform for application development and management across various cloud services like Azure, Google Cloud, and AWS.

This revolution is pivotal for technology and job landscapes, making it essential knowledge in fast-evolving tech cycles. The rapid emergence of Gen AI-powered and Physical AI technologies, and the evolving demand for skills necessitate extensive and timely professional training.

Vertical Specialization Level

Students will have the option of selecting one of the following specialisations after the completion of sixth quarter i.e. in the seventh quarter:

1. **Healthcare and Medical GenAI Specialization**
2. **Web3, Blockchain, and GenAI Integration Specialization**
3. **Metaverse, 3D, and GenAI Integration Specialization**
4. **GenAI for Accounting, Finance, and Banking Specialization**
5. **GenAI for Engineers Specialization**
6. **GenAI for Sales and Marketing Specialization**
7. **GenAI for Automation and Internet of Things (IoT) Specialisation**
8. **GenAI for Cyber Security**

Common Questions (FAQs) with Detailed Answers

1. **What is Cloud Native Applied Generative AI Engineering?**
   Cloud Applied Generative AI Engineering (GenEng) is the application of generative AI technologies to solve real-world problems in the cloud.
   - Generative AI is a type of artificial intelligence that can create new data or content from existing data.
   - Cloud Native computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet (“the cloud”).

   By combining generative AI with cloud native computing, businesses can solve a variety of problems, such as:
   - Creating personalised experiences for customers
   - Automating tasks

- Improving decision-making
- Detecting fraud
- Developing new products and services

The potential applications of cloud native-applied generative AI are endless. As generative AI and cloud native computing continue to develop, we can expect to see even more innovative and groundbreaking uses for this technology.

2. **How valuable are the Cloud Native Applied Generative AI developers?**
   Developers with expertise in Cloud Native Applied Generative AI were in extremely high demand due to the increasing adoption of GenAI technologies across various industries. However, the supply of developers skilled specifically in this niche area might not have been as abundant compared to more generalised AI or cloud computing roles.

   The demand for AI developers, especially those proficient in applying generative AI techniques within cloud native environments, has been rising due to the growing interest in using AI for creative applications, content generation, image synthesis, natural language processing, and other innovative purposes.

   According to some sources, the average salary for a Cloud Native Applied Generative AI developer in the global market is around $150,000 per year. However, this may vary depending on the experience level, industry, location, and skills of the developer. For example, a senior Cloud Applied Generative AI developer with more than five years of experience can earn up to $200,000 per year. A Cloud Applied Generative AI developer working in the financial services industry can earn more than a developer working in the entertainment industry. A Cloud Applied Generative AI developer working in New York City can earn more than a developer working in Dubai. In general, highly skilled AI developers, especially those specialising in applied generative AI within cloud environments, tend to earn competitive salaries that are often above the average for software developers or AI engineers due to the specialised nature of their skills. Moreover, as generative AI technology becomes more widely adopted and integrated into various products and services, the demand for Cloud Applied Generative AI developers is likely to increase.

   Therefore, Cloud Applied Generative AI developers are valuable professionals who have a bright future ahead of them. They can leverage their creativity and technical skills to create innovative solutions that can benefit various industries and domains. They can also enjoy very competitive salary and career growth opportunities.

3. **What is the potential for Cloud Applied Generative AI Developers to start their own companies?**
   Cloud Applied Generative AI Developers have a significant potential to start their own companies due to several factors:

1. **Emerging Field:** Generative AI, particularly when applied within cloud environments, is still an evolving field with immense potential for innovation. Developers who understand the intricacies of both generative AI techniques and cloud technologies can identify unique opportunities to create novel products, services, or solutions.
2. **Market Demand:** There is a growing demand for AI-driven applications, especially those that involve generative capabilities such as image generation, content creation, style transfer, etc. Developers with expertise in this area can leverage this demand to create specialized products that cater to specific industries or consumer needs.
3. **Innovation and Differentiation:** The ability to develop unique and innovative solutions using generative AI in the cloud can set apart these developers' startups from more conventional companies. They can explore new ways of generating content, enhancing user experiences, or solving complex problems with AI-generated solutions.
4. **Access to Cloud Resources:** Cloud platforms provide scalable and cost-effective resources that are crucial for AI development. Developers starting their own companies can leverage cloud services to access powerful computing resources, storage, and AI-related services without significant upfront investment.
5. **Entrepreneurial Opportunities:** Developers with entrepreneurial spirit and a deep understanding of AI technologies can identify gaps in the market and build startups to fill those gaps. They can create platforms, tools, or services that simplify the adoption of generative AI for businesses or developers.
6. **Collaboration and Partnerships:** These developers can collaborate with other experts in AI, domain specialists, or businesses to create innovative solutions or explore new application areas for generative AI in the cloud.

   However, starting a company, especially in a specialised field like Cloud Applied Generative AI, requires more than technical expertise. It also demands business acumen, understanding market needs, networking, securing funding, managing resources effectively, and navigating legal and regulatory landscapes.

   Successful entrepreneurship in this domain involves a combination of technical skills, innovation, a deep understanding of market dynamics, and the ability to transform technical expertise into viable products or services that address real-world challenges or opportunities.

Developers aspiring to start their own companies in the Cloud Applied Generative AI space can do so by conducting thorough market research, networking with industry experts, building a strong team, and developing a clear business plan that highlights the unique value proposition of their offerings.

To sum up, the potential for Cloud Applied Generative AI Developers to start their own companies is high.

- Generative AI is a rapidly growing field with a high demand for skilled professionals.
- The Certified Generative AI (GenEng) Developer and Engineering Program provides students with the skills and knowledge they need to develop and apply cutting-edge generative AI technologies.
- The program also teaches students how to start and run a successful business.
- Graduates of the program will be well-positioned to start their own companies and capitalise on the growing demand for generative AI solutions.

4. **Is the program not too long, eighteen months is a long time?**
   The length of the program is eighteen months which is broken down into sixF quarters of three months each. The program covers a wide range of topics including Python, GenAI, Microservices, Database, Cloud Development, Fine-tuning, DevOps, GPTs, AI Agents, and Humanoids. The program is designed to give students a comprehensive understanding of generative AI and prepare them for careers in this field. Nothing valuable can be achieved overnight, there are no shortcuts in life.

5. **Why don't we use TypeScript (Node.js) to develop APIs instead of using Python?**
   We will not use Typescript in GenAI API development because Python is a priority with the AI community when working with AI and if any updates come in libraries they will first come for Python. Python is always a better choice when dealing with AI and API.

   - **Python is the de facto standard for AI Development**.
   - TypeScript is a more modern language that is gaining popularity for Web Development, but Python is more widely used and has a larger ecosystem of libraries and frameworks available, especially for AI.
   - TypeScript is used for web user interfaces, while Python is used for APIs.

- In the second quarter, students will learn to develop APIs using Python instead of TypeScript.
- Python is a more commonly used language for AI and API development, and it has a larger ecosystem of libraries and frameworks available for these purposes.
- TypeScript is a more modern language that is becoming increasingly popular for API development also, but it is still not as widely used as Python, especially for AI applications and development.

6. **What is the difference between OpenAI Completion API, OpenAI Assistant API, Google Gemini Multi-Modal API, and LangChain?**
   The difference between OpenAI Completion API, OpenAI Assistant API, Google Gemini Multi-Modal API, and LangChain is that they are different ways of using artificial intelligence to generate text, images, audio, and video based on some input, but they have different features and applications. Here is a summary of each one:

   OpenAI Completion API is the most fundamental OpenAI model that provides a simple interface that's extremely flexible and powerful. You give it a prompt and it returns a text completion, generated according to your instructions. You can think of it as a very advanced autocomplete where the language model processes your text prompt and tries to predict what's most likely to come next. The Completion API can be used for various tasks such as writing stories, poems, essays, code, lyrics, etc. It also supports different models with different levels of power suitable for different tasks.

   OpenAI Assistant API is an interface to OpenAI's most capable model (gpt-4) and their most cost-effective model (gpt-3.5-turbo). It provides a simple way to take text as input and use a model like gpt-4 or gpt-3.5-turbo to generate an output. The Assistant API allows you to build AI assistants within your applications. An Assistant has instructions and can leverage models, tools, and knowledge to respond to user queries. The Assistant API currently supports three types of tools: Code Interpreter, Retrieval, and Function calling.

   Google Gemini Multi-Modal API is a new series of foundational models built and introduced by Google. It is built with a focus on multimodality from the ground up. This makes the Gemini models powerful against different combinations of information types including text, images, audio, and video. Currently, the API supports images and text. Gemini has proven by reaching state-of-the-art performance on the benchmarks and even beating the ChatGPT and the GPT4-Vision models in many of the tests.
   There are three different Gemini models based on their size, the Gemini Ultra, Gemini Pro, and Gemini Nano in decreasing order of their size.

LangChain is a platform that allows you to interact with various language models from different providers such as OpenAI, Google Gemini, Hugging Face Transformers, etc. You can use LangChain to create applications that leverage the power of natural language processing without having to deal with the complexity of APIs or SDKs. LangChain provides a user-friendly interface that lets you choose the model you want to use, customize the parameters you want to apply, and see the results in real-time.

7. **Why don't we use Flask or Django for API development instead of FastAPI?**
    - **FastAPI is a newer and more modern framework than Flask or Django.** It is designed to be fast, efficient, and easy to use. FastAPI is also more scalable than Flask or Django, making it a better choice for large-scale projects.
    - **FastAPI is also more feature-rich than Flask or Django.** It includes several built-in features that make it easy to develop APIs, such as routing, validation, and documentation.
    - **Overall, FastAPI is a better choice for API development than Flask or Django.** It is faster, more scalable, and more feature-rich.

8. **Why do we need to learn Cloud technologies in a Generative AI program?**
   Cloud technologies are essential for developing and deploying generative AI applications because they provide a scalable and reliable platform for hosting and managing complex workloads.

    - Cloud computing offers a vast pool of resources that can be provisioned on demand, which is ideal for generative AI applications that can be computationally intensive.
    - Cloud providers offer a wide range of services that can be used to support generative AI applications, including storage, computing, networking, and machine learning.
    - Cloud services are typically more cost-effective than on-premises infrastructure, which can be a significant advantage for generative AI applications that are often used for large-scale projects.

   The Certified Generative AI (GenEng) Developer and Engineering Program teaches you how to use cloud native services, including containers and Kubernetes, to deploy your applications to the cloud. You will also learn how to use **Docker containers** to package and deploy your applications, and how to use Terraform to manage your cloud infrastructure.

By the end of the program, you will be able to:

- Use Docker containers to package and deploy your applications
- Develop and deploy generative AI applications to the cloud
- Manage your cloud infrastructure using Terraform

9. **What is the purpose of Docker Containers and what are the benefits of deploying them with Docker Compose, and Kubernetes?**
    - **Docker Containers** are a way to package software into a single unit that can be run on any machine, regardless of its operating system. It is used to create a Dockerfile, which is a text file that describes how to build a Docker image. The image is then used to create a container, which is a running instance of the image. This makes them ideal for deploying applications on a variety of platforms, including cloud-based services.
    - **Docker Compose** is a tool provided by Docker that allows you to define and manage multi-container Docker applications locally. It enables you to use a YAML file to configure the services, networks, and volumes needed for your application's setup. With Docker Compose, you can describe the services your application requires, their configurations, dependencies, and how they should interact with each other, all in a single file. This makes it easier to orchestrate complex applications locally composed of multiple interconnected containers.
    - **Kubernetes** is a container orchestration system that automates the deployment, scaling, and management of containerized applications. It allows you to run multiple containers on a single machine or across multiple machines. It is an open source and can be deployed in your data centre or the cloud.

10. **What is the purpose of learning to develop APIs in a Generative AI program?**
    APIs (Application Programming Interfaces) are used to connect different software applications and services together. They are the building blocks of the internet and are essential for the exchange of data between different systems.

    In the third quarter of the Certified Generative AI (GenEng) Developer and Engineering Program, students will learn to develop APIs not just as a backend for their front end but also as a **product** itself. In this model, the API is at the core of the business's value.

- APIs are used to make it possible for different software applications to communicate with each other.
- APIs are used to access data from a remote server.
- APIs are used to create new services or applications that are integrated with existing systems.
- APIs are used to improve the security of applications by providing a way to control access to data.
- By learning to develop APIs, students will gain the skills necessary to create powerful and efficient software applications that can be used to solve a variety of business problems.

11. **What is the purpose of using Python-based FastAPI and related technologies in Quarter 3?**
    In the third quarter of the Engineering Program, students will learn how to use Python-based FastAPI as a core library for API development.

    - FastAPI is a high-performance, lightweight, and easy-to-use framework for building APIs.
    - It is designed to be fast, scalable, and secure.
    - FastAPI is compatible with a wide range of programming languages and frameworks, making it a good choice for developers with different skill sets.
    - Students will also learn about the following related technologies:
    - **Pydantic:** Pydantic is a Python library that helps to improve the quality of your code by checking for errors and potential problems.
    - **SQLModel:** SQLModel is a Python library that provides an object-relational mapping (ORM) layer for working with databases.
    - **PostgreSQL:** PostgreSQL is a free and open-source relational database management system (RDBMS) that can be used for development. Highly scalable database systems compatible with it have also been deployed by all the major cloud platforms.

    By the end of the quarter, students will be able to use Python-based FastAPI to develop APIs that are fast, scalable, and secure.

12. **What does the API-as-a-Product model entail?**
    API-as-a-Product is a type of Software-as-a-Service that monetizes niche functionality, typically served over HTTP. In this model, the API is at the core of the business's value. The API-as-a-Product model is different from the traditional API model, where APIs are used as a means to access data or functionality from another application. In the API-as-a-Product model, the API itself is the product that is being sold.

The benefits of the API-as-a-Product model include:

- **Increased flexibility:** APIs can be used to access data or functionality from any application, regardless of the underlying platform or technology. This gives businesses greater flexibility in how they integrate APIs into their applications.
- **Reduced development costs:** APIs can be reused by multiple applications, which can save businesses the time and expense of developing their custom APIs.
- **Improved scalability:** APIs can be scaled up or down as needed, which makes them well-suited for businesses with fluctuating or unpredictable traffic demands.
- **Enhanced security:** APIs can be more secure than traditional methods of data exchange, as they can be protected by a variety of security measures, such as encryption and access control.

13. **What are the benefits of using Docker Containers for development, testing, and deployment?**
    Docker Containers are a fundamental building block for development, testing, and deployment because they provide a consistent environment that can be used across different systems. This eliminates the need to worry about dependencies or compatibility issues, and it can help to improve the efficiency of the development process. Additionally, Docker Containers can be used to isolate applications, which can help to improve security and make it easier to manage deployments.

14. **What is the advantage of using open Docker, Kubernetes, and Terraform technologies instead of using AWS, Azure, or Google Cloud technologies?**
    Using open-source technologies like Docker, Kubernetes, and Terraform offers several advantages over relying solely on proprietary cloud services from AWS, Azure, or Google Cloud. Here's a detailed comparison:

    Advantages of Using Docker, Kubernetes, and Terraform (Open Technologies)

    1. Portability and Flexibility:
    - Vendor Agnostic: These tools are cloud-agnostic, meaning you can run your applications on any cloud provider or on-premises infrastructure without being locked into a specific vendor.

- Ease of Migration: Applications packaged in Docker containers can easily be moved across different environments, and Kubernetes provides a consistent orchestration layer, ensuring seamless transitions.

2. Cost Efficiency:
- Avoid Vendor Lock-In: Being locked into a single cloud provider can lead to higher costs over time. Using open technologies allows you to leverage competitive pricing from multiple providers or even use on-premises resources.
- Optimised Resource Utilisation: Kubernetes helps in efficiently managing resources through automated scaling and load balancing, potentially reducing costs.

3. Community and Ecosystem:
- Open Source: These tools are backed by strong, active open-source communities that continuously improve the software, provide support, and share best practices.
- Ecosystem: A rich ecosystem of tools and integrations is available, providing flexibility to choose the best components that fit your specific needs.

4. Standardisation and Consistency:
- Unified Platform: Using Docker for containerization, Kubernetes for orchestration, and Terraform for infrastructure as code (IaC) provides a standardised way to deploy, manage, and scale applications across different environments.
- Consistency Across Environments: These tools ensure that your development, staging, and production environments are consistent, reducing bugs and deployment issues.

5. Customization and Control:
- Full Control: Open-source tools give you complete control over your infrastructure and deployment pipelines. You can customise and extend the functionality to suit specific requirements.
- Transparency: Access to the source code means you can audit and modify the software to meet your security and compliance needs.

Advantages of Using AWS, Azure, or Google Cloud Technologies

1. Managed Services:
- **Ease of Use:** Cloud providers offer a wide range of managed services that abstract away the complexity of setting up and managing infrastructure. This can save time and reduce operational overhead.

- Integrated Solutions: These platforms provide integrated services and tools, such as databases, machine learning, analytics, and monitoring, which can be easily combined to build complex applications.

2. Scalability and Reliability:
- Global Infrastructure: Cloud providers have extensive global infrastructure, ensuring high availability, redundancy, and low latency.
- Auto-Scaling: Advanced auto-scaling capabilities can dynamically adjust resources to meet changing demands, ensuring optimal performance.

3. Security and Compliance:
- Built-In Security: Cloud providers offer robust security features, including identity and access management, encryption, and compliance certifications, helping to protect your data and meet regulatory requirements.
- Automatic Updates: Managed services often include automatic updates and patches, reducing the risk of security vulnerabilities.

4. Innovation and Support:
- Cutting-Edge Technology: Major cloud providers continuously innovate and introduce new services and features, allowing you to leverage the latest technologies without significant investment.
- Support and SLA: Comprehensive support services and Service Level Agreements (SLAs) ensure that you have access to expert help and guaranteed uptime.

Conclusion

Choosing between open-source technologies like Docker, Kubernetes, and Terraform versus proprietary cloud services from AWS, Azure, or Google Cloud depends on your specific needs and priorities.

- Open Technologies: Offer portability, cost efficiency, customization, and control, making them ideal for multi-cloud strategies, avoiding vendor lock-in, and having more control over your infrastructure.
- Cloud Providers: Provide ease of use, managed services, scalability, security, and access to cutting-edge technology, which can be advantageous for rapid development, scaling, and leveraging advanced services.

In many cases, a hybrid approach that combines the strengths of both open-source tools and cloud provider services can provide the best of both worlds, allowing you to optimise for cost, flexibility, and innovation.

15. **Why in this program are we not learning to build LLMs ourselves? How difficult is it to develop an LLM like ChatGPT 4 or Google's Gemini?**

Developing an LLM like ChatGPT 4 or Google Gemini is extremely difficult and requires a complex combination of resources, expertise, and infrastructure. Here's a breakdown of the key challenges:

Technical hurdles:

Massive data requirements: Training these models requires an immense amount of high-quality data, often exceeding petabytes. Compiling, cleaning, and structuring this data is a monumental task.
Computational power: Training LLMs demands incredible computational resources, like high-performance GPUs and specialised AI hardware. Access to these resources and the ability to optimise training processes are crucial.
Model architecture: Designing the LLM's architecture involves complex decisions about parameters, layers, and attention mechanisms. Optimising this architecture for performance and efficiency is critical.
Evaluation and bias: Evaluating the performance of LLMs involves diverse benchmarks and careful monitoring for biases and harmful outputs. Mitigating these biases is an ongoing research challenge.

Resource and expertise:

Team effort: Developing an LLM like ChatGPT 4 or Google Gemini requires a large team of experts across various disciplines, including AI researchers, machine learning engineers, data scientists, and software developers.
Financial investment: The financial resources needed are substantial, covering costs for data acquisition, hardware, software, and talent. Access to sustained funding is critical.

Additionally:

Ethical considerations: LLMs raise ethical concerns like potential misuse, misinformation, and societal impacts. Responsible development and deployment are crucial.
Rapidly evolving field: The LLM landscape is constantly evolving, with new research, models, and benchmarks emerging. Staying abreast of these advancements is essential.

Therefore, while ChatGPT 4 and Google Gemini have made impressive strides, developing similar LLMs remains a daunting task accessible only to a handful of organisations with the necessary resources and expertise.

In simpler terms, it's like building a skyscraper of knowledge and intelligence. You need the right materials (data), the right tools (hardware and software),

the right architects (experts), and a lot of hard work and attention to detail to make it stand tall and function flawlessly.

Developing similar models would be a daunting task for individual developers or smaller teams due to the enormous scale of resources and expertise needed. However, as technology progresses and research findings become more accessible, it might become incrementally more feasible for a broader range of organisations or researchers to work on similar models, albeit at a smaller scale or with fewer resources. At that time we might also start to focus on developing LLMs ourselves.

To sum up, the focus of the program is not on LLM model development but on applied Cloud GenAI Engineering (GenEng), application development, and fine-tuning of foundational models. The program covers a wide range of topics including Python, GenAI, Microserices, API, Database, Cloud Development, and DevOps, which will give students a comprehensive understanding of generative AI and prepare them for careers in this field.

16. **Business wise does it make more sense to develop LLMs ourselves from scratch or use LLMs developed by others and build applications using these tools by using APIs and/or fine-tuning them?**
Whether it makes more business sense to develop LLMs from scratch or leverage existing ones through APIs and fine-tuning depends on several factors specific to your situation. Here's a breakdown of the pros and cons to help you decide:

Developing LLMs from scratch:

Pros:

Customization: You can tailor the LLM to your specific needs and data, potentially achieving higher performance on relevant tasks.
Intellectual property: Owning the LLM allows you to claim intellectual property rights and potentially monetize it through licensing or other means.
Control: You have full control over the training data, algorithms, and biases, ensuring alignment with your ethical and business values.

Cons:

High cost: Building and training LLMs require significant technical expertise, computational resources, and data, translating to high financial investment.
Time commitment: Developing an LLM is a time-consuming process, potentially delaying your go-to-market with your application.

Technical expertise: You need a team of highly skilled AI specialists to design, train, and maintain the LLM.

Using existing LLMs:

Pros:

Lower cost: Leveraging existing LLMs through APIs or fine-tuning is significantly cheaper than building them from scratch.
Faster time to market: You can quickly integrate existing LLMs into your applications, accelerating your launch timeline.
Reduced technical burden: You don't need a large team of AI specialists to maintain the LLM itself

Cons:

Less customization: Existing LLMs are not specifically designed for your needs, potentially leading to lower performance on some tasks.
Limited control: You rely on the data and biases of the existing LLM, which might not align with your specific requirements.
Dependency on external parties: You are dependent on the availability and maintenance of the LLM by its developers.

Here are some additional factors to consider:

The complexity of your application: Simpler applications might benefit more from existing LLMs, while highly complex ones might require the customization of a dedicated LLM.
Your available resources: If you have the financial and technical resources, developing your own LLM might be feasible. Otherwise, existing options might be more practical.
Your competitive landscape: If your competitors are using LLMs, you might need to follow suit to remain competitive.
Ultimately, the best decision depends on your specific needs, resources, and business goals. Carefully evaluating the pros and cons of each approach will help you choose the strategy that best aligns with your success.

17. **What are Custom GPTs?**
"Custom GPTs" refers to specialised versions of the Generative Pre-trained Transformer (GPT) models that are tailored for specific tasks, industries, or data types. These custom models are adapted from the base GPT architecture, which is a type of language model developed by OpenAI. Custom GPT models are trained or fine-tuned on specific datasets or for

particular applications, allowing them to perform better in those contexts compared to the general-purpose models.

Here are some examples of what custom GPT models might be used for:

1. **Industry-Specific Needs**: A custom GPT for legal, medical, or financial industries could be trained on domain-specific texts to understand and generate industry-specific language more accurately.

2. **Language and Localization**: Models can be customised for different languages or dialects that might not be well-represented in the training data of the base model.

3. **Company-Specific Applications**: Organisations might develop a custom GPT model trained on their own documents and communications to assist with internal tasks like drafting emails, generating reports, or providing customer support.

4. **Educational Purposes**: Educational institutions might develop custom GPTs trained on educational material to assist in creating teaching materials or providing tutoring in specific subjects.

5. **Creative Writing and Entertainment**: Custom models could be trained on specific genres of literature or scripts to assist in creative writing or content creation.

6. **Technical and Scientific Research**: A custom GPT model could be trained on scientific literature to assist researchers in summarising papers, generating hypotheses, or even drafting new research.

These custom models are created through a process of fine-tuning, where the base GPT model is further trained (or 'fine-tuned') on a specific dataset. This process allows the model to become more adept at understanding and generating text that is relevant to the specific use case. Fine-tuning requires expertise in machine learning and natural language processing, as well as access to relevant training data.

18. **What are Actions in GPTs?**
    Actions are a way to connect custom GPTs to external APIs, allowing them to access data or interact with the real-world. For example, you can use actions to create a GPT that can book flights, send emails, or order pizza. **Actions are defined using the OpenAPI specification**, which is a standard for describing APIs. You can import an existing OpenAPI specification or create a new one using the GPT editor.

19. What are AI Agents and how do they differ from Custom GPTs?

AI Agents and Custom GPTs are both tools that utilise artificial intelligence to perform tasks, but they have distinct functionalities and use cases. Here's a breakdown of their differences:

AI Agents

AI Agents are autonomous programs that can perceive their environment, make decisions, and act upon them to achieve specific goals. They often interact with other systems or users, continuously learning and adapting based on their experiences.

Key Characteristics:
1. Autonomy: AI Agents operate independently without continuous human intervention.
2. Learning: They often employ machine learning algorithms to improve performance over time.
3. Interactivity: AI Agents can interact with their environment, other systems, and users.
4. Goal-Oriented: They are designed to achieve specific objectives and can adapt their actions to optimise towards these goals.
5. Multi-Modal Capabilities: AI Agents can incorporate various forms of AI, such as computer vision, natural language processing, and decision-making algorithms.

Examples:
- Robotics: Autonomous robots that navigate and perform tasks.
- Virtual Assistants: Programs like Siri or Alexa that interact with users and perform tasks based on voice commands.
- Game AI: Non-player characters (NPCs) that adapt and react to player actions.

Custom GPTs

Custom GPTs are tailored instances of OpenAI's ChatGPT, launched in late 2022. They are designed for specific purposes and enhanced with context. Each custom GPT can have a unique "personality," including tone of voice, language complexity, and responsiveness to specific topics. For example, a financial institution's custom GPT could be trained on financial reports and industry-specific terminology, while a healthcare provider's version might focus on medical literature and health policy documents

Key Differences

1. Autonomy:
   - AI Agents: Operate autonomously and continuously interact with their environment.
   - Custom GPTs: Typically respond to specific inputs and generate outputs accordingly, but don't operate autonomously beyond text generation tasks.

2. Learning and Adaptation:
   - AI Agents: Often incorporate continuous learning and adaptation mechanisms.
   - Custom GPTs: Rely on pre-training and fine-tuning phases, with limited continuous learning capabilities.

4. Interactivity:
   - AI Agents: Can interact with both digital and physical environments.
   - Custom GPTs: Primarily interact through text-based inputs and outputs.

In summary, while both AI Agents and Custom GPTs utilise AI, AI Agents are designed for autonomous, goal-oriented actions in diverse environments, and Custom GPTs are specialised in generating and understanding human-like text for specific applications.

20. Do we need to use Design Thinking and BDD for designing custom GPTs and AI Agents?

Design Thinking and Behavior-Driven Development (BDD) are methodologies that can greatly enhance the process of designing custom GPTs and AI Agents, though they are not strictly necessary. Here's how each can be beneficial:

Design Thinking

Design Thinking is a user-centred approach to innovation and problem-solving that involves understanding the user, challenging assumptions, redefining problems, and creating innovative solutions through iterative prototyping and testing.

Benefits for Custom GPTs and AI Agents:
1. User-Centric Focus: Ensures that the AI solutions are tailored to the actual needs and pain points of users.
2. Empathy: Helps in understanding the context and environment in which the AI will be used, leading to more relevant and effective solutions.
3. Iterative Development: Encourages continuous testing and refinement of ideas, leading to more robust and user-friendly AI models.
4. Collaboration: Promotes cross-disciplinary collaboration, which can bring diverse perspectives and expertise to the design process.

Behaviour-Driven Development (BDD)
BDD is a software development methodology that encourages collaboration between developers, QA, and non-technical stakeholders through the use of natural language descriptions of the desired behaviour of the software.

Benefits for Custom GPTs and AI Agents:
1. Clear Requirements: Ensures that the requirements are clearly understood and agreed upon by all stakeholders.
2. Testable Scenarios: Facilitates the creation of testable scenarios that can validate the AI's behaviour against the expected outcomes.
3. Documentation: Provides clear and comprehensive documentation of the AI's intended behaviour, which is useful for future maintenance and enhancements.
4. Alignment: Ensures that the development stays aligned with business goals and user expectations.

Application in Designing Custom GPTs and AI Agents

For Custom GPTs:
- Design Thinking:
  - Understand the specific use cases and user interactions where the GPT will be applied.
  - Iterate on the model's performance by gathering user feedback and refining the fine-tuning process.
  - Prototype different conversation flows and evaluate their effectiveness with real users.

- BDD:
  - Define the expected behaviours of the GPT in natural language scenarios.
  - Create automated tests that validate the GPT's responses against these scenarios.
  - Ensure that the GPT's behaviour aligns with user stories and business requirements.

For AI Agents:
- Design Thinking:
  - Map out the user journey and identify critical interaction points where the AI Agent will provide value.
  - Prototype and test the agent's interactions in various environments to ensure robustness and usability.
  - Use empathy maps and personas to better understand and anticipate user needs and behaviours.

- BDD:
  - Write behaviour scenarios that describe how the AI Agent should react in different situations.
  - Develop tests that simulate these scenarios to verify the agent's decision-making and learning processes.
  - Continuously refine the agent's behaviour based on test results and user feedback.

While not strictly necessary, Design Thinking and BDD can significantly enhance the design and development process of custom GPTs and AI Agents by ensuring a user-centred approach, clear requirements, and continuous improvement through iterative testing and feedback. These methodologies help in creating more effective, reliable, and user-friendly AI solutions.

21. **When Fine-Tuning Open-Source Large Language Models, how is FastAI and PyTorch important and what role do these libraries play?**
Fine-tuning open-source Large Language Models (LLMs) like Meta LLaMA 3 involves adapting pre-trained models to specific tasks or domains by continuing the training process on a smaller, task-specific dataset. FastAI and PyTorch play crucial roles in this process due to their powerful features and ease of use. Here's how they contribute:

PyTorch
PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab. It provides a flexible and efficient platform for building and training neural networks.

Importance in Fine-Tuning LLMs:
1. Dynamic Computation Graphs: PyTorch's dynamic computation graph (also known as define-by-run) allows for flexibility and ease of debugging. This is particularly useful when experimenting with different model architectures and training strategies.
2. Extensive Libraries and Tools: PyTorch has a wide range of libraries and tools that facilitate various deep learning tasks, including natural language processing (NLP). Libraries like Hugging Face's Transformers are built on top of PyTorch, providing pre-trained models and utilities for fine-tuning.
3. GPU Acceleration: PyTorch supports GPU acceleration, which is essential for handling the large computational requirements of fine-tuning LLMs.
4. Community and Ecosystem: PyTorch has a strong community and extensive documentation, making it easier to find resources, tutorials, and support for fine-tuning tasks.

Role in Fine-Tuning:

- Model Customization: Allows users to modify the architecture of LLMs to better fit specific tasks.
- Efficient Training: Provides efficient backpropagation and optimization routines to train large models on specialised datasets.
- Experimentation: Facilitates rapid experimentation with different hyperparameters and training setups due to its flexible framework.

FastAI

FastAI is a deep learning library built on top of PyTorch that aims to simplify training neural networks by providing high-level abstractions and best practices.

Importance in Fine-Tuning LLMs:

1. Ease of Use: FastAI provides high-level APIs that simplify many complex tasks involved in training and fine-tuning models, making it accessible to both beginners and experts.
2. State-of-the-Art Techniques: FastAI incorporates many state-of-the-art techniques and best practices in deep learning, such as learning rate schedules, data augmentation, and transfer learning, which can improve the performance of fine-tuned models.
3. Rapid Prototyping: FastAI's concise and readable code allows for rapid prototyping and iteration, which is crucial for experimenting with different fine-tuning approaches.
4. Integrated Data Handling: FastAI offers powerful data handling and preprocessing utilities, which streamline the process of preparing datasets for training.

Role in Fine-Tuning:

- High-Level API: Simplifies the process of loading pre-trained models, defining custom datasets, and setting up training loops.
- Training Utilities: Provides utilities for monitoring training progress, adjusting learning rates, and saving model checkpoints.
- Best Practices: Encourages the use of best practices in model training, such as using discriminative learning rates and employing effective data augmentation techniques.

Combined Workflow for Fine-Tuning LLMs

1. Model Loading: Use PyTorch (and libraries like Hugging Face Transformers) to load a pre-trained LLM.
2. Data Preparation: Leverage FastAI's data handling capabilities to preprocess and load the fine-tuning dataset.

3. Training Setup: Define the model architecture and training parameters using PyTorch, and use FastAI's high-level APIs to simplify this process.
4. Fine-Tuning: Utilise FastAI's training utilities to fine-tune the model on the specific dataset, adjusting hyperparameters as needed.
5. Evaluation and Iteration: Evaluate the fine-tuned model using PyTorch's evaluation utilities, and iterate on the fine-tuning process by leveraging FastAI's rapid prototyping capabilities.

Both PyTorch and FastAI play crucial roles in the fine-tuning of open-source LLMs. PyTorch provides the foundational tools and flexibility needed for deep learning, while FastAI builds on top of PyTorch to offer high-level abstractions and best practices that streamline the fine-tuning process. Together, they enable efficient, effective, and user-friendly fine-tuning of large language models for specific tasks and domains.

22. **In this course both PyTorch and FastAI play crucial roles in the fine-tuning of open-source LLMs, why don't we use TensorFlow instead?**
While TensorFlow is a powerful and widely-used deep learning framework, PyTorch and FastAI offer several advantages that make them particularly well-suited for fine-tuning open-source Large Language Models (LLMs). Here's a detailed comparison highlighting why PyTorch and FastAI might be preferred over TensorFlow for this specific task:

PyTorch vs. TensorFlow

1. Dynamic Computation Graphs:
- PyTorch: Uses dynamic computation graphs (define-by-run), which allow for greater flexibility and ease of debugging. This is especially useful when experimenting with new models and training strategies.
- TensorFlow: Initially used static computation graphs (define-and-run). Although TensorFlow 2.0 introduced eager execution to support dynamic graphs, PyTorch's implementation is often considered more intuitive and easier to work with for dynamic tasks.

2. Ease of Use:
- PyTorch: Known for its simplicity and clear, Pythonic code, which makes it easier to learn and use, especially for research and prototyping.
- TensorFlow: While TensorFlow 2.0 improved usability, it is still considered more complex compared to PyTorch, particularly for newcomers.

3. Community and Ecosystem:
- PyTorch: Has seen rapid adoption in the research community, leading to a rich ecosystem of tools, libraries, and community support. Libraries like

Hugging Face's Transformers are built primarily for PyTorch, offering extensive support for LLMs.
  - TensorFlow: Has a strong industrial presence and is widely used in production environments. However, the research community has increasingly favoured PyTorch.

4. Integration with Hugging Face:
  - PyTorch: Hugging Face's Transformers library, which is a go-to for working with LLMs, is deeply integrated with PyTorch. This library provides pre-trained models, tokenizers, and utilities that simplify the process of fine-tuning LLMs.
  - TensorFlow: Although Hugging Face provides TensorFlow support, the integration is not as seamless or feature-rich as it is with PyTorch.

FastAI vs. TensorFlow

1. High-Level Abstractions:
  - FastAI: Provides high-level abstractions that simplify complex tasks in model training and fine-tuning, making it accessible and efficient for users. These abstractions are built on top of PyTorch, leveraging its flexibility.
  - TensorFlow: While Keras (part of TensorFlow) offers high-level APIs, FastAI's APIs are often considered more intuitive and tailored towards rapid experimentation and prototyping.

2. Best Practices:
  - FastAI: Incorporates state-of-the-art techniques and best practices in deep learning, such as learning rate schedules, transfer learning, and data augmentation, making it easier to achieve high performance with minimal effort.
  - TensorFlow: Requires more manual effort to implement many of these best practices, which can be a barrier for quick iteration and experimentation.

3. Community and Learning Resources:
  - FastAI: Has an active community and a wealth of educational resources, including courses and documentation that focus on practical, hands-on learning.
  - TensorFlow: Also has extensive documentation and resources, but the learning curve can be steeper, especially for those new to deep learning.

Summary of Why PyTorch and FastAI Might Be Preferred

- Flexibility and Debugging: PyTorch's dynamic computation graph is more flexible and easier to debug.
- User-Friendly: PyTorch and FastAI offer more intuitive and user-friendly interfaces, making them suitable for rapid prototyping and experimentation.

- Research and Development: PyTorch is preferred in the research community, leading to faster adoption of new techniques and tools like Hugging Face's Transformers.
- Seamless Integration: Hugging Face's deep integration with PyTorch provides a robust ecosystem for working with LLMs.
- Educational Resources: FastAI's focus on best practices and practical education makes it easier for users to get started and achieve good results quickly.

While TensorFlow remains a powerful framework, particularly in production environments, PyTorch and FastAI provide a combination of flexibility, ease of use, and community support that make them particularly well-suited for the fine-tuning of open-source LLMs.

23. What is Physical AI?

Physical AI refers to the integration of artificial intelligence with physical entities, such as robots, that can operate and interact in the real world. This concept involves AI systems that not only process data and make decisions but also perform physical actions and understand the laws of physics.

Key Characteristics:

1. Real-World Interaction:
   - Physical AI systems can perceive their environment through sensors, process this information, and take appropriate actions using actuators.

2. Embodiment:
   - Unlike purely digital AI, Physical AI involves AI embedded in physical bodies, like humanoid robots, which can navigate and manipulate the physical world.

3. Understanding Physics:
   - These AI systems are designed to comprehend and adhere to the physical laws that govern real-world interactions, such as gravity, friction, and object dynamics.

4. Human-like Functionality:
   - Humanoid robots are a prime example of Physical AI, as they are built to perform tasks in environments designed for humans, utilising a form factor that mirrors human anatomy.

5. Data-Driven Training:

- Physical AI leverages vast amounts of real-world data to train AI models, enabling robots to improve their performance through machine learning and interaction experiences.

Applications:
- **Healthcare**:
  - Assistive robots that help with patient care, rehabilitation, and surgery.
- **Service Industry**:
  - Robots that perform tasks such as cleaning, delivery, and customer service.
- **Manufacturing**:
  - Industrial robots that assemble products, manage inventory, and ensure quality control.
- **Exploration**:
  - Robots designed for exploration in environments like space, underwater, or disaster zones.

Physical AI represents a significant shift from traditional AI applications confined to virtual environments. It aims to bridge the gap between digital intelligence and physical capability, creating systems that can understand and interact with the world in a human-like manner. This evolution has the potential to revolutionise various industries by enhancing automation, improving efficiency, and enabling new forms of human-machine collaboration.

24. What are the different specialisations offered at the end of the program and what are their benefits?

At the end of the GenEng certification program we offer six specialisations in different fields:

Healthcare and Medical GenAI: This specialisation will teach students how to use generative AI to improve healthcare and medical research. This is relevant to fields such as drug discovery, personalised medicine, and surgery planning.

Benefits:

- Learn how to use generative AI to identify diseases, develop new drugs, and personalise treatment plans.
- Gain a deeper understanding of the ethical implications of using generative AI in healthcare.
- Prepare for a career in a growing field with high demand for skilled professionals.

Web3, Blockchain, and GenAI Integration: This specialisation will teach students how to integrate generative AI with Web3 and blockchain technologies. This is relevant to fields such as finance, healthcare, and supply chain management.
Benefits:

- Learn how to create smart contracts and decentralised applications (dApps).
- Gain a deeper understanding of the potential of blockchain technology and how it can be used to improve business processes.
- Develop the skills necessary to work in a rapidly growing field with high demand for skilled professionals.

Metaverse, 3D, and GenAI Integration: This specialisation will teach students how to create and use 3D models and other immersive content manually and with generative AI. This is relevant to fields such as gaming, marketing, and architecture.
Benefits:

- Learn how to use generative AI to create realistic and immersive 3D models.
- Develop the skills necessary to work in the growing field of virtual reality (VR) and augmented reality (AR).
- Apply generative AI to solve real-world problems in areas such as product design, marketing, and education.

GenAI for Accounting, Finance, and Banking: This specialisation will teach students how to use generative AI to improve accounting, finance, and banking processes. This is relevant to fields such as fraud detection, risk management, and investment analysis.
Benefits:

- Learn how to use generative AI to automate tasks, identify patterns, and make predictions.
- Gain a deeper understanding of the financial industry and how generative AI can be used to improve its processes.
- Prepare for a career in a growing field with high demand for skilled professionals.

GenAI for Engineers: This specialisation will teach students how to use generative AI to improve engineering design and problem-solving. This is relevant to fields such as manufacturing, construction, and product development.
Benefits:

- Learn how to use generative AI to create simulations, optimize designs, and predict failures.

- Gain a deeper understanding of the engineering design process and how generative AI can be used to improve it.
- Prepare for a career in a growing field with high demand for skilled professionals.

GenAI for Sales and Marketing: This specialisation will teach students how to use generative AI to improve sales and marketing campaigns. This is relevant to fields such as advertising, public relations, and customer service. Benefits:

- Learn how to use generative AI to create personalised marketing messages, generate leads, and track campaign performance.
- Gain a deeper understanding of the latest marketing trends and how generative AI can be used to improve them.
- Prepare for a career in a growing field with high demand for skilled professionals.

GenAI for Automation and Internet of Things (IoT):

- **Provide Multi-Modal User Interface for the IoT systems:** Multimodal interaction exploits the synergic use of different modalities to optimise the interactive tasks accomplished by the users. This allows a user to use several input modes such as speech, touch, and visual to interact with IoT systems.
- **Improve efficiency and accuracy of industrial processes:** By implementing GenAI in automation and IoT systems, industries can optimise their processes, reduce manual labour, and increase productivity while ensuring higher accuracy and consistency.
- **Enhance decision-making:** GenAI can analyse vast amounts of data collected by IoT sensors to derive valuable insights, enabling businesses to make informed decisions regarding operations, maintenance, and resource allocation.
- **Personalise user experiences:** GenAI can leverage IoT data to understand user preferences and behaviours, enabling the creation of personalised experiences across smart devices and IoT-enabled systems.

GenAI for Cyber Security:

- **Strengthen threat detection and response:** GenAI can be used to rapidly detect and respond to cyber threats by analysing large volumes of security data in real time, identifying anomalies, and suggesting appropriate countermeasures.
- **Enhance security monitoring and analysis:** GenAI can assist security analysts in monitoring and analysing security logs, automating threat detection, and providing insights into security risks and vulnerabilities.

- **Improve threat intelligence:** GenAI can be used to gather and analyse threat intelligence from various sources, enabling organisations to stay informed about the latest threats and trends and proactively strengthen their security posture.