**Introduction to Large Language Models**

Introduction to Large Language Models

How’s it going? I’m Megha. Today I’m going to be talking about large language models. Don’t know what those are? Me either. Just kidding, I actually know what I’m talking about. I’m a customer engineer here at Google Cloud, and today, I’m going to teach you everything you need to know about LLMs, that’s short for large language models. In this course, you’re going to learn to ***define large language models, describe LLM use cases, explain prompt tuning and describe Google’s generative AI development tools***. Let’s get into it. Large language models, or LLMs, are a subset of deep learning. To find out more about deep learning, check out our Introduction to Generative AI course video. LLMs and generative AI intersect and they are both a part of deep learning. Another area of AI you may be hearing a lot about is generative AI. This is a type of artificial intelligence that can produce new content, including text, images, audio, and synthetic data. All right, back to LLMs. So, what are large language models? Large language models refer to large, general‑purpose language models that can be pre‑trained and then fine‑tuned for specific purposes. What do pre‑trained and fine‑tuned mean? Great questions. Let’s dive in. Imagine training a dog. Often you train your dog basic commands such as sit, come, down, and stay. These commands are normally sufficient for everyday life and help your dog become a good canine citizen, good boy. But, if you need special service dogs, such as a police dog, a guide dog, or a hunting dog, you add special trainings, right? The similar idea applies to large language models. These models are trained for general purposes to solve common language problems, such as text classification, question answering, document summarization, and text generation across industries. The models can then be tailored to solve specific problems in different fields, such as retail, finance, and entertainment, using a relatively small size of field datasets.

So now that you’ve got that down, let’s further break down the concept into three major features of large language models. We’ll start with the word, large. **Large** indicates two meanings. *First* is the enormous size of the training dataset, sometimes at the petabyte scale. *Second*, it refers to the parameter count. In machine learning, parameters are often called hyperparameters. Parameters are basically the memories and the knowledge the machine learned from the model training. Parameters define the skill of a model in solving a problem, such as predicting text. So that’s why we use the word, large. What about general purpose? **General purpose** is when the models are sufficient to solve common problems. Two reasons led to this idea. *First* is the commonality of human language regardless of the specific tasks, and *second* is the resource restriction. Only certain organizations have the capability to train such large language models with huge datasets and a tremendous number of parameters. How about letting them create fundamental language models for others to use? So this leaves us with our last terms, pre‑trained and fine‑tuned, which mean to ***pre‑train*** *a large language model for a general purpose with a large dataset*, and then ***fine‑tune*** *it for specific aims with a much smaller dataset*.

So now that we’ve nailed down the definition of what large language models (LLMs) are, we can move onto describing **LLM use cases**.

The *benefits* of using large language models are straightforward.

1. A single model can be used for different tasks. This is a dream come true. These large language models that are trained with petabytes of data and generate billions of parameters are smart enough to solve different tasks, including language translation, sentence completion, text classification, question answering, and more.
2. Large language models require minimal field training data when you tailor them to solve your specific problem. Large language models obtain decent performance even with little domain training data. In other words, they can be used for few‑shot or even zero‑shot scenarios. In machine learning, few‑shot refers to training a model with minimal data, and zero‑shot implies that a model can recognize things that have not explicitly been taught in the training before.
3. The performance of large language models is continuously growing when you add more data and parameters.

Large language models are almost ***exclusively based on transformer models***. Let me explain what that means. A transformer model consists of an encoder and a decoder. The encoder encodes the input sequence and passes it to the decoder, which learns how to decode the representations for a relevant task.

We’ve come a long way from traditional programming, to neural networks, to generative models. In traditional programming, we used to have to hard code the rules for distinguishing a cat; type: animal, legs: four, ears: two, or yes, likes yarn and catnip. In the wave of neural networks, we could give the network pictures of cats and dogs and ask, is this a cat? And it would predict a cat. What’s really cool is that in the generative wave, we as users can generate our own content, whether it be text, images, audio, video, or more. For example, models like Gemini, Google’s multimodal AI model, or LaMDA (Language Model for Dialogue Applications) ingest very, very large data from multiple sources across the internet and build foundation language models we can use simply by asking a question, whether typing it into a prompt or verbally talking into the prompt itself. So, when you ask it, what’s a cat? It can give you everything it has learned about a cat.

1. Let’s compare LLM development using pre‑trained models with traditional ML development. With LLM development, you don’t need to be an expert. You don’t need training examples, and there is no need to train a model. All you need to do is think about prompt design, which is a process of creating a prompt that is clear, concise, and informative. It is an important part of natural language processing, or NLP for short. In traditional machine learning, you need expertise, training examples, compute time, and hardware. That’s a lot more requirements than LLM development. Let’s take a look at an example of a text generation use case to really drive the point home. Question answering, or QA, is a subfield of natural language processing that deals with the task of automatically answering questions posed in natural language. QA systems are typically trained on a large amount of text and code, and they are able to answer a wide range of questions, including factual, definitional, and opinion‑based questions. The key here is that you needed domain knowledge to develop these question answering models. Let’s make this clear with a real‑world example. Domain knowledge is required to develop a question answering model for customer IT support, or healthcare, or supply chain. But using generative QA, the model generates free text directly based on the context; there’s no need for domain knowledge. Let me show you a few more examples of how cool this is. Let’s look at three questions given to Gemini, a large language model chatbot developed by Google AI. Question one. This year’s sales are $100,000. Expenses are $60,000. How much is net profit? Gemini first shares how net profit is calculated, then performs the calculation. Then, Gemini provides the definition of net profit. Here’s another question. Inventory on hand is 6000 units. A new order requires 8000 units. How many units do I need to fill to complete the order? Again, Gemini answers the question by performing the calculation. And our last example, we have 1000 sensors in 10 geographic regions. How many sensors do we have on average in each region? Gemini answers the question with an example on how to solve the problem and some additional context. So how is it that in each of our questions a desired response was obtained? This is due to prompt design. Fancy. Prompt design and prompt engineering are two closely related concepts in natural language processing. Both involve the process of creating a prompt that is clear, concise, and informative, but there are some key differences between the two. Prompt design is the process of creating a prompt that is tailored to the specific task the system is being asked to perform. For example, if the system is being asked to translate a text from English to French, the prompt should be written in English and should specify that the translation should be in French. Prompt engineering is the process of creating a prompt that is designed to improve performance. This may involve using domain‑specific knowledge, providing examples of the desired output, or using keywords that are known to be effective for the specific system. In general, prompt design is a more general concept, while prompt engineering is a more specialized concept. Prompt design is essential, while prompt engineering is only necessary for systems that require a high degree of accuracy or performance. There are three kinds of large language models: generic language models, instruction‑tuned, and dialogue‑tuned. Each needs prompting in a different way. Let’s start with generic language models. Generic language models predict the next word based on the language in the training data. Here is a generic language model. In this example, the cat sat on, the next word should be the, and you can see that the is most likely the next word. Think of this model type as an autocomplete in search. Next, we have instruction‑tuned models. This type of model is trained to predict a response to the instructions given in the input. For example, summarize the text of X. Generate a poem in the style of X. Give me a list of keywords based on semantic similarity for X. In this example, classify text into neutral, negative, or positive. And finally, we have dialogue‑tuned models. This model is trained to have a dialogue by the next response. Dialogue‑tuned models are a special case of instruction‑tuned, where requests are typically framed as questions to a chatbot. Dialogue‑tuning is expected to be in the context of a longer back‑and‑forth conversation and typically works better with natural question‑like phrasings. Chain of thought reasoning is the observation that models are better at getting the right answer when they first output text that explains the reason for the answer. Let’s look at the question. Roger has five tennis balls. He buys two more cans of tennis balls. Each can has three tennis balls. How many tennis balls does he have now? This question is posed initially with no response. The model is less likely to get the correct answer directly. However, by the time the second question is asked, the output is more likely to end with the correct answer. But there is a catch, there’s always a catch. A model that can do everything has practical limitations. But, task‑specific tuning can make LLMs more reliable. Vertex AI provides task‑specific foundation models. Let’s get into how you can tune with some real‑world examples. Let’s say you have a use case where you need to gather how your customers are feeling about your product or service, you can use a sentiment analysis task model. Same provision tasks. If you need to perform occupancy analytics, there is a task‑specific model for your use case. Tuning a model enables you to customize the model response based on examples of the task that you want the model to perform. It is essentially the process of adapting a model to a new domain or a set of custom use cases by training the model on new data. For example, we may collect training data and tune the model specifically for the legal or medical domain. You can also further tune the model by fine‑tuning, where you bring your own dataset and retrain the model by tuning every weight in the LLM. This requires a big training job and hosting your own fine‑tuned model. Here’s an example of a medical foundation model trained on healthcare data. The tasks include question answering, image analysis, finding similar patients, etc. Fine‑tuning is expensive and not realistic in many cases, so, are there more efficient methods of tuning? Yes. Parameter‑efficient tuning methods (PETM) are methods for tuning a large language model on your own custom data without duplicating the model. The base model itself is not altered. Instead, a small number of add‑on layers are tuned, which can be swapped in and out at inference time. I’m going to tell you about three other ways Google Cloud can help you get more out of your LLMs. The first is Vertex AI Studio. Vertex AI Studio lets you quickly explore and customize generative AI models that you can leverage in your applications on Google Cloud. Vertex AI Studio helps developers create and deploy generative AI models by providing a variety of tools and resources that make it easy to get started. For example, there is a library of pre‑trained models, a tool for fine‑tuning models, a tool for deploying models to production, and a community forum for developers to share ideas and collaborate. Next, we have Vertex AI, which is particularly helpful for those of you who don’t have much coding experience. You can build generative AI search and conversations for customers and employees with Vertex AI Agent Builder, formerly Vertex AI Search and Conversation. Build with little or no coding and no prior machine learning experience. Vertex AI can help you create your own chatbots, digital assistants, custom search engines, knowledge bases, training applications, and more. Gemini is a multimodal AI model. Unlike traditional language models, it’s not limited to understanding text alone, it can analyze images, understand the nuances of audio, and even interpret programming code. This allows Gemini to perform complex tasks that were previously impossible for AI. Due to its advanced architecture, Gemini is incredibly adaptable and scalable, making it suitable for diverse applications. Model Garden is continuously updated to include new models. See, I told you way back in the beginning of this video that I knew what I was talking about when it came to large language models, and now you do too. Thank you for watching our course, and make sure to check out our other videos if you want to learn more about how you can use AI.

Introduction to Large Language Models: Reading