As we have seen certain models can successfully perform zero shot inference by recognizing instructions in a prompt, but others—like smaller LLMs—may not be able to complete the task, as seen in this example. You also saw that one shot, or few shot inference, which involves giving the model one or more examples of what you want it to perform, can be sufficient to help it recognize the task and produce a good completion.

However, there are a few disadvantages to this tactic. First, even with five or six samples, it isn't always effective for smaller models. Second, the context window's available size is limited by whatever examples you put in your prompt, leaving less room for extra helpful information. Fortunately, there is an alternative: you can further train a base model by utilizing a technique called fine-tuning. Fine-tuning is a supervised learning procedure where you utilize a data collection of labeled examples to update the weights of the LLM, as opposed to pre-training, when you train the LLM using enormous volumes of unstructured textual data using self supervised learning.

One method that works very well for enhancing a model's performance on a range of tasks is called instruction fine tuning. Let's examine how this operates in more detail. Instruction fine-tuning uses examples to show the model how to react to a particular instruction. Here are a few sample questions that illustrate this concept. Classifying this review is the instruction in both examples, and the intended outcome is a text string that begins with a sentiment and is followed by either positive or negative. Numerous pairs of prompt completion examples for the job of interest, each with an instruction, are included in the training data set.

Full fine-tuning is a type of instruction fine-tuning in which every weight in the model is changed. A revised model with the updated weights is the end outcome of the operation. As with pre-training, it is crucial to remember that complete fine tuning necessitates sufficient memory and computational resources to store and process all of the gradients, optimizers, and other elements that are being modified throughout training. The fine-tuning process results in a new version of the base model, often called an instruct model that is better at the tasks you are interested in. Fine-tuning with instruction prompts is the most common way to fine-tune LLMs these days.

Even though LLMs are well-known for being able to handle a variety of linguistic jobs within a single model, your application might only need to handle one activity. In this situation, a pre-trained model can be adjusted to perform better on just the task that interests you. For instance, summarizing a task using a dataset of instances. It's interesting to note that comparatively few examples can yield good results. In contrast to the billions of texts the model viewed during pre-training, 500–1,000 instances can frequently produce good performance. However, focusing only on fine-tuning one activity may have drawbacks. Catastrophic forgetting is a phenomenon that could result from the procedure.

Catastrophic forgetting is a phenomenon that could result from the procedure. Because the weights of the original LLM are altered during the fine-tuning phase, catastrophic forgetting occurs. This can result in poor performance on other tasks, even when it produces excellent performance on the single fine-tuning work. For instance, fine-tuning can make a model more capable of performing sentiment analysis on a review and producing high-quality results, but it may also cause the model to lose its capacity to perform other tasks.

In conventional machine learning, a model's performance on training and validation data sets, where the output is known, can be used to gauge how well it is performing. Because the models are deterministic, you can compute basic metrics like accuracy, which expresses the percentage of all forecasts that are accurate. However, language-based evaluation becomes considerably more difficult with complex language models, whose output is non-deterministic.

However, an automated and organized method of measurement is required when training a model on millions of phrases. Two popular measures for evaluating various tasks are ROUGE and BLEU. The main purpose of ROUGE, or recall oriented under study for jesting evaluation, is to evaluate the quality of automatically generated summaries by contrasting them with reference summaries that were created by humans. However, BLEU, or bilingual evaluation understudy, is an algorithm that compares machine-translated text against human-generated translations in order to assess the quality of the translation. The French word for blue is now BLEU.

Recall, accuracy, and F1 can be used to carry out basic metric computations that resemble other machine-learning tasks. The number of words or unigrams that match between the reference and the produced output, divided by the number of words or unigrams in the reference, is the recall metric. Since every created word in this instance matches a word in the reference, it receives a perfect score of 1 Precision is calculated by dividing the output size by the number of unigram matches. The F1 score is the harmonic mean of both of these values.

Bigram matches can now be used to compute the recall, precision, and F1 score rather than individual words. The scores are lower than the ROUGE-1 scores, as you can see. Longer sentences increase the likelihood that bigrams won't match, which could result in even lower scores. Let's try something else instead of carrying on with ROUGE numbers getting larger to n-grams of three or fours. Rather, search for the longest common subsequence that appears in both the reference output and the created output.

Parameter-efficient fine tuning techniques simply update a restricted subset of parameters, as opposed to full fine-tuning, which updates every model weight throughout supervised learning. Certain route strategies concentrate on fine-tuning a subset of the current model parameters, such as specific layers or components, while freezing the majority of the model weights. Other

methods just add a few new parameters or layers and adjust the new elements, leaving the previous model weights completely unaltered. The majority, if not all, of the LLM weights are maintained frozen when using PEFT.

Consequently, there are far fewer trained parameters than there were in the original LLM. Only 15–25% of the initial LLM weights in certain instances. This greatly reduces the amount of memory needed for training. In actuality, PEFT can sometimes be completed with just one GPU. Additionally, PEFT is less vulnerable to the disastrous forgetting issues of full fine-tuning because the original LLM is only marginally altered or left unaltered. Every task you train on yields a new version of the model after complete fine-tuning.

The original model can be effectively adapted to numerous tasks by simply switching out the PEFT weights, which are trained for each task. You may fine-tune parameters efficiently using a variety of techniques, each with trade-offs for model quality, inference costs, training speed, memory economy, and parameter efficiency. Let's examine the three primary PEFT method classes. Techniques that only adjust a portion of the initial LLM parameters are known as selective techniques. You can choose the parameters you wish to alter using a variety of methods. You can choose to train just particular model elements, layers, or even different kinds of parameters. Researchers have discovered that there are notable trade-offs between compute efficiency and parameter efficiency, and that various approaches perform inconsistently. They will not be the subject of this course. By developing new low rank transformations of the original network weights, reparameterization techniques decrease the amount of parameters to train while still utilizing the original LLM parameters. We'll go into more detail about LoRA, a popular method of this kind, in the upcoming video. Finally, by adding new trainable components and maintaining all of the initial LLM weights frozen, additive approaches do fine-tuning.