Model Optimization for Deployment

A diagram of a diagram

Description automatically generated

* Large language models present inference challenges in terms of computing and storage requirements, as well as ensuring low latency for consuming applications.
* These challenges persist whether you're deploying on premises or to the cloud, and become even more of an issue when deploying to edge devices.
* One of the primary ways to improve application performance is to reduce the size of the LLM. This can allow for quicker loading of the model, which reduces inference latency.
* However, the challenge is to reduce the size of the model while still maintaining model performance.
* Some techniques work better than others for generative models, and there are trade-offs between accuracy and performance. You'll learn about three techniques in this section.
* Distillation uses a larger model, the teacher model, to train a smaller model, the student model. You then use the smaller model for inference to lower your storage and compute budget.
* Similar to quantization aware training, post training quantization transforms a model's weights to a lower precision representation, such as a 16- bit floating point or eight bit integer. As you learned in week one of the course, this reduces the memory footprint of your model.
* The third technique, Model Pruning, removes redundant model parameters that contribute little to the model's performance.

Distillation

A diagram of a student model

Description automatically generated

* Model Distillation is a technique that focuses on having a larger teacher model train a smaller student model.
* The student model learns to statistically mimic the behavior of the teacher model, either just in the final prediction layer or in the model's hidden layers as well.
* You'll focus on the first option here.
* You start with your fine tune LLM as your teacher model and create a smaller LLM for your student model.
* You freeze the teacher model's weights and use it to generate completions for your training data.
* At the same time, you generate completions for the training data using your student model.
* The knowledge distillation between teacher and student model is achieved by minimizing a loss function called the distillation loss.
* To calculate this loss, distillation uses the probability distribution over tokens that is produced by the teacher model's softmax layer.
* Now, the teacher model is already fine-tuned on the training data. So the probability distribution likely closely matches the ground truth data and won't have much variation in tokens.
* That's why Distillation applies a little trick adding a temperature parameter to the softmax function. As you learned in lesson one, a higher temperature increases the creativity of the language the model generates.
* With a temperature parameter greater than one, the probability distribution becomes broader and less strongly peaked.
* This softer distribution provides you with a set of tokens that are similar to the ground truth tokens.
* In the context of Distillation, the teacher model's output is often referred to as soft labels and the student model's predictions as soft predictions.
* In parallel, you train the student model to generate the correct predictions based on your ground truth training data.
* Here, you don't vary the temperature setting and instead use the standard softmax function. Distillation refers to the student model outputs as the hard predictions and hard labels.
* The loss between these two is the student loss. The combined distillation and student losses are used to update the weights of the student model via back propagation.
* The key benefit of distillation methods is that the smaller student model can be used for inference in deployment instead of the teacher model.
* In practice, distillation is not as effective for generative decoder models. It's typically more effective for encoder only models, such as Burt that have a lot of representation redundancy.
* Note that with Distillation, you're training a second, smaller model to use during inference. You aren't reducing the model size of the initial LLM in any way.

Post Training Quantization or PTQ

A graph on a white background

Description automatically generated

Pruning

A diagram of weights with black text

Description automatically generated with medium confidence

Generative AI Project Lifecycle Cheat sheet

A white table with blue text

Description automatically generated