To save and load a vision language model (VLM) after quantization while preserving its complete architecture, follow these steps. This guide will utilize the Hugging Face Transformers library as an example, which is commonly used for handling such models.

Steps to Quantize, Save, and Load a Vision Language Model

1. **Quantization Process**

First, you need to quantize your model. Depending on whether you choose Post-Training Quantization (PTQ) or Quantization-Aware Training (QAT), the implementation will differ slightly.

Example of Post-Training Quantization (PTQ)

python

**from** transformers **import** AutoModelForCausalLM, AutoTokenizer

**from** optimum.gptq **import** GPTQQuantizer

**import** torch

*# Load your pre-trained vision language model*

model\_name = "your\_vision\_language\_model"

tokenizer = AutoTokenizer.from\_pretrained(model\_name)

model = AutoModelForCausalLM.from\_pretrained(model\_name, torch\_dtype=torch.float16)

*# Initialize the quantizer*

quantizer = GPTQQuantizer(bits=4, dataset="c4", block\_name\_to\_quantize="model.decoder.layers", model\_seqlen=2048)

*# Quantize the model*

quantized\_model = quantizer.quantize\_model(model, tokenizer)

2. **Save the Quantized Model**

After quantizing the model, you can save it along with its configuration using the save method provided by the GPTQQuantizer.

python

*# Specify the folder where you want to save the model*

save\_folder = "/path/to/save\_folder/"

*# Save the quantized model and its configuration*

quantizer.save(quantized\_model, save\_folder)

3. **Load the Quantized Model**

To load the quantized model later, you will need to initialize an empty model and then load the quantized weights.

python

**from** accelerate **import** init\_empty\_weights

**from** optimum.gptq **import** load\_quantized\_model

*# Initialize an empty model with the same architecture*

**with** init\_empty\_weights():

empty\_model = AutoModelForCausalLM.from\_pretrained(model\_name, torch\_dtype=torch.float16)

empty\_model.tie\_weights() *# Ensure weights are tied if necessary*

*# Load the quantized weights into the empty model*

quantized\_model = load\_quantized\_model(empty\_model, save\_folder=save\_folder, device\_map="auto")

4. **Evaluate and Use the Loaded Model**

Once loaded, you can evaluate or use your quantized vision language model as needed.

python

*# Example usage: Generate text or perform inference with the loaded quantized model*

input\_text = "What is the capital of France?"

inputs = tokenizer(input\_text, return\_tensors="pt")

**with** torch.no\_grad():

outputs = quantized\_model(\*\*inputs)

**print**(outputs)

Conclusion

By following these steps, you can effectively quantize a vision language model and save it along with its complete architecture. The process involves loading a pre-trained model, applying quantization techniques (either PTQ or QAT), saving the quantized version, and later loading it for inference or further evaluation. This approach ensures that you maintain both efficiency and performance in deploying large models on resource-constrained devices.