Converting HuggingFace Models to GGUF/GGML

August 31, 2023 · 3 min read

Sam Stoelinga
Engineer

Llama.cpp is a great way to run LLMs efficiently on CPUs and GPUs. The downside however is that you need to convert models to a format that's supported by Llama.cpp, which is now the GGUF file format. In this blog post you will learn how to convert a HuggingFace model (Vicuna 13b v1.5) to GGUF model.

At the time of writing, Llama.cpp supports the following models:

- LLaMA 🦙
- LLaMA 2 🦙🦙
- Falcon
- Alpaca
- GPT4All
- Chinese LLaMA / Alpaca and Chinese LLaMA-2 / Alpaca-2
- Vigogne (French)
- Vicuna
- Koala
- OpenBuddy 🐶 (Multilingual)
- Pygmalion 7B / Metharme 7B
- WizardLM
- Baichuan-7B and its derivations (such as baichuan-7b-sft)
- Aquila-7B / AquilaChat-7B

At a high-level you will be going through the following steps:

- Downloading a HuggingFace model

- Running llama.cpp `convert.py` on the HuggingFace model
- (Optionally) Uploading the model back to HuggingFace

Downloading a HuggingFace model

There are various ways to download models, but in my experience the `huggingface_hub` library has been the most reliable. The `git clone` method occasionally results in OOM errors for large models.

Install the `huggingface_hub` library:

```
pip install huggingface_hub
```

Create a Python script named `download.py` with the following content:

```
from huggingface_hub import snapshot_download
model_id="lmsys/vicuna-13b-v1.5"
snapshot_download(repo_id=model_id, local_dir="vicuna-hf",
                  local_dir_use_symlinks=False, revision="main")
```

Run the Python script:

```
python download.py
```

You should now have the model downloaded to a directory called `vicuna-hf`. Verify by running:

```
ls -lash vicuna-hf
```

Converting the model

Now it's time to convert the downloaded HuggingFace model to a GGUF model. Llama.cpp comes with a converter script to do this.

Get the script by cloning the llama.cpp repo:

```
git clone https://github.com/ggerganov/llama.cpp.git
```

Install the required python libraries:

```
pip install -r llama.cpp/requirements.txt
```

Verify the script is there and understand the various options:

```
python llama.cpp/convert.py -h
```

Convert the HF model to GGUF model:

```
python llama.cpp/convert.py vicuna-hf \
  --outfile vicuna-13b-v1.5.gguf \
  --outtype q8_0
```

In this case we're also quantizing the model to 8 bit by setting `--outtype q8_0`. Quantizing helps improve inference speed, but it can negatively impact quality. You can use `--outtype f16` (16 bit) or `--outtype f32` (32 bit) to preserve original quality.

Verify the GGUF model was created:

```
ls -lash vicuna-13b-v1.5.gguf
```

Pushing the GGUF model to HuggingFace

You can optionally push back the GGUF model to HuggingFace.

Create a Python script with the filename `upload.py` that has the following content:

```python
from huggingface_hub import HfApi
api = HfApi()

model_id = "substratusai/vicuna-13b-v1.5-gguf"
```

```
api.create_repo(model_id, exist_ok=True, repo_type="model")
api.upload_file(
    path_or_fileobj="vicuna-13b-v1.5.gguf",
    path_in_repo="vicuna-13b-v1.5.gguf",
    repo_id=model_id,
)
```

Get a HuggingFace Token that has write permission from here: https://huggingface.co/settings/tokens

Set your HuggingFace token:

```
export HUGGING_FACE_HUB_TOKEN=<paste-your-own-token>
```

Run the `upload.py` script:

```
python upload.py
```

Interested in learning how to automate flows like this? Checkout our open source project:
Star 133

Tags: llama.cpp gguf

Edit this page