**<https://github.com/ollama/ollama/blob/main/docs/import.md>**

**Date: 20/5/2024 Version: 0.1.38**

**Import a model**

This guide walks through importing a GGUF, PyTorch or Safetensors model.

**Importing (GGUF)**

**Step 1: Write a Modelfile**

Start by creating a Modelfile. This file is the blueprint for your model, specifying weights, parameters, prompt templates and more.

FROM ./mistral-7b-v0.1.Q4\_0.gguf

(Optional) many chat models require a prompt template in order to answer correctly. A default prompt template can be specified with the TEMPLATE instruction in the Modelfile:

FROM ./mistral-7b-v0.1.Q4\_0.gguf

TEMPLATE "[INST] {{ .Prompt }} [/INST]"

**Step 2: Create the Ollama model**

Finally, create a model from your Modelfile:

ollama create example -f Modelfile

**Step 3: Run your model**

Next, test the model with ollama run:

ollama run example "What is your favourite condiment?"

**Importing (PyTorch & Safetensors)**

Importing from PyTorch and Safetensors is a longer process than importing from GGUF. Improvements that make it easier are a work in progress.

**Setup**

First, clone the ollama/ollama repo:

git clone git@github.com:ollama/ollama.git ollama

cd ollama

and then fetch its llama.cpp submodule:

git submodule init

git submodule update llm/llama.cpp

Next, install the Python dependencies:

python3 -m venv llm/llama.cpp/.venv

source llm/llama.cpp/.venv/bin/activate

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

Then build the quantize tool:

make -C llm/llama.cpp quantize

**Clone the HuggingFace repository (optional)**

If the model is currently hosted in a HuggingFace repository, first clone that repository to download the raw model.

Install [Git LFS](https://docs.github.com/en/repositories/working-with-files/managing-large-files/installing-git-large-file-storage), verify it's installed, and then clone the model's repository:

git lfs install

git clone https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1 model

**Convert the model**

Note: some model architectures require using specific convert scripts. For example, Qwen models require running convert-hf-to-gguf.py instead of convert.py

python llm/llama.cpp/convert.py ./model --outtype f16 --outfile converted.bin

**Quantize the model**

llm/llama.cpp/quantize converted.bin quantized.bin q4\_0

**Step 3: Write a Modelfile**

Next, create a Modelfile for your model:

FROM quantized.bin

TEMPLATE "[INST] {{ .Prompt }} [/INST]"

**Step 4: Create the Ollama model**

Finally, create a model from your Modelfile:

ollama create example -f Modelfile

**Step 5: Run your model**

Next, test the model with ollama run:

ollama run example "What is your favourite condiment?"

**Publishing your model (optional – early alpha)**

Publishing models is in early alpha. If you'd like to publish your model to share with others, follow these steps:

1. Create [an account](https://ollama.com/signup)
2. Copy your Ollama public key:

* macOS: cat ~/.ollama/id\_ed25519.pub | pbcopy
* Windows: type %USERPROFILE%\.ollama\id\_ed25519.pub
* Linux: cat /usr/share/ollama/.ollama/id\_ed25519.pub

1. Add your public key to your [Ollama account](https://ollama.com/settings/keys)

Next, copy your model to your username's namespace:

ollama cp example <your username>/example

Note: model names may only contain lowercase letters, digits, and the characters ., -, and \_.

Then push the model:

ollama push <your username>/example

After publishing, your model will be available at https://ollama.com/<your username>/example.

**Quantization reference**

The quantization options are as follow (from highest highest to lowest levels of quantization). Note: some architectures such as Falcon do not support K quants.

* q2\_K
* q3\_K
* q3\_K\_S
* q3\_K\_M
* q3\_K\_L
* q4\_0 (recommended)
* q4\_1
* q4\_K
* q4\_K\_S
* q4\_K\_M
* q5\_0
* q5\_1
* q5\_K
* q5\_K\_S
* q5\_K\_M
* q6\_K
* q8\_0
* f16