bartowski/Llama-3.2-3B-Instruct-GGUF

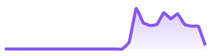
like 39

Text Generation | Transformers | GGUF | PyTorch | 8 languages | facebook | meta | llama | llama-3 | Inference Endpoints | imatrix | conversational | License: llama3.2

Train | Deploy | Use this model

Model card | Files | Community 3

Downloads last month
34,217



GGUF | Model size 3.21B params | Architecture llama

3-bit	Q3_K_L Q3_K_XL
4-bit	IQ4_XS Q4_K_S Q4_0 Q4_0 Q4_0 Q4_0 Q4_K_M Q4_K_L
5-bit	Q5_K_S Q5_K_M Q5_K_L
6-bit	Q6_K Q6_K_L
8-bit	Q8_0
16-bit	F16

View +1 file

Inference Examples

Text Generation

This model does not have enough activity to be deployed to Inference API (serverless) yet. Increase its social visibility and check back later, or deploy to Inference Endpoints (dedicated) instead.

Model tree for bartowski/Llama-3.2-3B-Instruct-GGUF

Base model	meta-llama/Llama-3.2-3B-Instruct
Quantized (57)	this model
Quantizations	1 model

Spaces using bartowski/Llama-3.2-3B-Instruct-GGUF 2

Masterdqqq/bartowski-Llama-3.2-3B-Instruct-GGUF | sergey21000/chatbot-rag

Edit model card

Llamacpp imatrix Quantizations of Llama-3.2-3B-Instruct

Using llama.cpp release b3821 for quantization.

Original model: https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

All quants made using imatrix option with dataset from here

Run them in LM Studio

Prompt format

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

Cutting Knowledge Date: December 2023
Today Date: 26 Jul 2024

{system_prompt}<|eot_id|><|start_header_id|>user<|end_header_id|>

{prompt}<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Download a file (not the whole branch) from below:

Filename	Quant type	File Size	Split	Description
Llama-3.2-3B-Instruct-f16.gguf	f16	6.43GB	false	Full F16 weights.

Filename	Quant type	File Size	Split	Description
Llama-3.2-3B-Instruct-Q8_0.gguf	Q8_0	3.42GB	false	Extremely high quality, generally unneeded but max available quant.
Llama-3.2-3B-Instruct-Q6_K_L.gguf	Q6_K_L	2.74GB	false	Uses Q8_0 for embed and output weights. Very high quality, near perfect, *recommended*.
Llama-3.2-3B-Instruct-Q6_K.gguf	Q6_K	2.64GB	false	Very high quality, near perfect, *recommended*.
Llama-3.2-3B-Instruct-Q5_K_L.gguf	Q5_K_L	2.42GB	false	Uses Q8_0 for embed and output weights. High quality, *recommended*.
Llama-3.2-3B-Instruct-Q5_K_M.gguf	Q5_K_M	2.32GB	false	High quality, *recommended*.
Llama-3.2-3B-Instruct-Q5_K_S.gguf	Q5_K_S	2.27GB	false	High quality, *recommended*.
Llama-3.2-3B-Instruct-Q4_K_L.gguf	Q4_K_L	2.11GB	false	Uses Q8_0 for embed and output weights. Good quality, *recommended*.
Llama-3.2-3B-Instruct-Q4_K_M.gguf	Q4_K_M	2.02GB	false	Good quality, default size for must use cases, *recommended*.
Llama-3.2-3B-Instruct-Q4_K_S.gguf	Q4_K_S	1.93GB	false	Slightly lower quality with more space savings, *recommended*.
Llama-3.2-3B-Instruct-Q4_0_8_8.gguf	Q4_0_8_8	1.92GB	false	Optimized for ARM inference. Requires 'sve' support (see link below).
Llama-3.2-3B-Instruct-Q4_0_4_8.gguf	Q4_0_4_8	1.92GB	false	Optimized for ARM inference. Requires 'i8mm' support (see link below).
Llama-3.2-3B-Instruct-Q4_0_4_4.gguf	Q4_0_4_4	1.92GB	false	Optimized for ARM inference. Should work well on all ARM chips, pick this if you're unsure.
Llama-3.2-3B-Instruct-Q4_0.gguf	Q4_0	1.92GB	false	Legacy format, generally not worth using over similarly sized formats
Llama-3.2-3B-Instruct-Q3_K_XL.gguf	Q3_K_XL	1.91GB	false	Uses Q8_0 for embed and output weights. Lower quality but usable, good for low RAM availability.
Llama-3.2-3B-Instruct-IQ4_XS.gguf	IQ4_XS	1.83GB	false	Decent quality, smaller than Q4_K_S with similar performance, *recommended*.
Llama-3.2-3B-Instruct-Q3_K_L.gguf	Q3_K_L	1.82GB	false	Lower quality but usable, good for low RAM availability.
Llama-3.2-3B-Instruct-IQ3_M.gguf	IQ3_M	1.60GB	false	Medium-low quality, new method with decent performance comparable to Q3_K_M.

Embed/output weights

Some of these quants (Q3_K_XL, Q4_K_L etc) are the standard quantization method with the embeddings and output weights quantized to Q8_0 instead of what they would normally default to.

Some say that this improves the quality, others don't notice any difference. If you use these models PLEASE COMMENT with your findings. I would like feedback that these are actually used and useful so I don't keep uploading quants no one is using.

Thanks!

Downloading using huggingface-cli

First, make sure you have hugginface-cli installed:

```
pip install -U "huggingface_hub[cli]"
```

Then, you can target the specific file you want:

```
huggingface-cli download bartowski/Llama-3.2-3B-Instruct-GGUF --include
```

If the model is bigger than 50GB, it will have been split into multiple files. In order to download them all to a local folder, run:

```
huggingface-cli download bartowski/Llama-3.2-3B-Instruct-GGUF --include
```

You can either specify a new local-dir (Llama-3.2-3B-Instruct-Q8_0) or download them all in place (./)

Q4_0_X_X

These are *NOT* for Metal (Apple) offloading, only ARM chips.

If you're using an ARM chip, the Q4_0_X_X quants will have a substantial speedup. Check out Q4_0_4_4 speed comparisons on the original pull request

To check which one would work best for your ARM chip, you can check AArch64 SoC features (thanks EloyOn!).

Which file should I choose?

A great write up with charts showing various performances is provided by Artefact2 here

The first thing to figure out is how big a model you can run. To do this, you'll need to figure out how much RAM and/or VRAM you have.

If you want your model running as FAST as possible, you'll want to fit the whole thing on your GPU's VRAM. Aim for a quant with a file size 1-2GB smaller than your GPU's total VRAM.

If you want the absolute maximum quality, add both your system RAM and your GPU's VRAM together, then similarly grab a quant with a file size 1-2GB Smaller than that total.

Next, you'll need to decide if you want to use an 'I-quant' or a 'K-quant'.

If you don't want to think too much, grab one of the K-quants. These are in format 'QX_K_X', like Q5_K_M.

If you want to get more into the weeds, you can check out this extremely useful feature chart:

llama.cpp feature matrix

But basically, if you're aiming for below Q4, and you're running cuBLAS (Nvidia) or rocBLAS (AMD), you should look towards the I-quants. These are in format IQX_X, like IQ3_M. These are newer and offer better performance for their size.

These I-quants can also be used on CPU and Apple Metal, but will be slower than their K-quant equivalent, so speed vs performance is a tradeoff you'll have to decide.

The I-quants are *not* compatible with Vulcan, which is also AMD, so if you have an AMD card double check if you're using the rocBLAS build or the Vulcan build. At the time of writing this, LM Studio has a preview with ROCm support, and other inference engines have specific builds for ROCm.

Credits

Thank you kalomaze and Dampf for assistance in creating the imatrix calibration dataset

Thank you ZeroWw for the inspiration to experiment with embed/output

Want to support my work? Visit my ko-fi page here: https://ko-fi.com/bartowski