

Run AI Software Locally

Do you want to run AI models locally? You can. No specialised hardware is required, although you need a reasonably powerful "desktop" computer. Nevertheless, you can try it on low-end consumer machines. You can run some AI software suits on a low-power computer like mine; the only problem with this approach is that the software will sorely test your patience. You need a computer that has at least 16GB DDR3 RAM or better, a quad-core (four cores + four threads) CPU capable of running at 2.1GHz, 8.5GB storage space for models, and 200MB for the software. No fancy GPU is required. I tested it on my Windows desktop computer. Everything went okay.

My PC's specs:

```
Processor:      Intel(R) Core(TM) i5-4570S CPU @ 2.90GHz    2.90 GHz
                Haswell 22nm Technology
Installed RAM:  16.0 GB (15.5 GB usable) Dual-Channel
                DDR3 @ 781MHz
Board:          Gigabyte Technology Co. Ltd. H81M-S
Graphics:       ACER S271HL (1366x768@60Hz)
                Intel HD Graphics 4600 (Gigabyte)
                512MB NVIDIA GeForce 9600 GT (XFX Pine Group)
                ForceWare version: 342.01
                SLI Disabled
System type:    64-bit operating system, x64-based processor
Pen and touch:  No pen or touch input is available for this display
```

OS:

```
Edition:        Windows 10 Pro
Version:        20H2
Installed on:    9/12/2020
OS build:        19042.1055
Experience:      Windows Feature Experience Pack 120.2212.2020.0
```

GPT4All

Luckily, [GPT4All](#) offers you the "easy" solution.

Check whether your PC's CPU supports AVX2:

[GPT4All](#) is a GUI application. The models must be downloaded separately. No worry! You will find links to those models on their website. The models [GPT4All](#) supports are also supported by

another program [LlamaGPTJ-chat](#). The program [LlamaGPTJ-chat](#) is a command-line program. It is a model runner that also checks whether your CPU supports AVX2. On top of that, you will need this small C++ program later if you wish to use the models via this program from your console. Many people love console applications, too. So, there's no loss.

Step 1:

Download [LlamaGPTJ-chat](#)'s compiled executable files (depending on your OS) from the 'Release' section. For example: if you are on Windows, you'll need 1. chat-windows-latest-avx.exe and 2. chat-windows-latest-avx2.exe.

Open a terminal emulator of your choice, cmd.exe, WezTerm, Alacritty etc., in the directory where you've downloaded the executable files.

Type `chat-windows-latest-avx2.exe` .

```
C:\Users\USERNAME>chat-windows-latest-avx2.exe
LlamaGPTJ-chat (v. 0.2.1)
Your computer supports AVX2
LlamaGPTJ-chat: loading .\models\ggml-vicuna-13b-1.1-q4_2.bin
.
```

If you see `Your computer supports AVX2` , that means you are good to go with the regular versions of the model runner applications (both the [GPT4All](#) and [LlamaGPTJ-chat](#)).

If not, Download the "AVX Only" version of [GPT4All](#) and keep the "chat-windows-latest-avx.exe" version of [LlamaGPTJ-chat](#), since your CPU has no support for AVX2.

Step 2:

Download models:

The basic models:

1. [ggml-mpt-7b-chat.bin](#)
2. [ggml-gpt4all-j-v1.3-groovy.bin](#)

There is no shortage of pre-trained models on their site for your use. These are the models [GPT4All](#) will first search for. Download the models with a download manager like [Free Download Manager](#). Usually, downloading such large files (4.52GB and 3.92GB, respectively) without the torrent protocol requires a download manager. [FDM](#) is a closed-source, proprietary, free-to-use file downloader application.

Step 3:

Install [GPT4All](#). It's fairly easy.

Copy chat-windows-latest-avx.exe or chat-windows-latest-avx2.exe (depending on the situation) into the `bin` directory of the installation folder of [GPT4All](#), generally, `%USERPROFILE%\gpt4all\bin`.

Step 4:

Add the `%USERPROFILE%\gpt4all\bin` folder to PATH:

```
WINDOWS + R -> type systempropertiesadvanced -> ENTER.
```

Advanced Tab -> Environment Variables... -> System variables... -> Path -> New. Paste the actual path there, `C:\Users\YOUR_USERNAME\gpt4all\bin`.

Step 5:

Create a hard symbolic link to a folder in another location of your hard drive where you will store the large models. I wanted to keep the downloaded model files (`*.bin`) in `N:\GPT4All`. The program expects the model files to be inside

`C:\Users\YOUR_USERNAME\AppData\Local\nomic.ai\GPT4All\`. Sorry! I don't want those large files to occupy the system drive of my machine. I created a symbolic link. `mklink` is used to create symlinks in Windows.

```
mklink /Option Link Target
Or,
mklink /Option Virtual Actual

Options:
/D -> soft link
/H -> hard link
/J -> hard link pointing to a directory (directory junction)
```

Here, in my case, it was:

```
mklink /J "C:\Users\YOUR_USERNAME\AppData\Local\nomic.ai\GPT4All\" "N:\GPT4All"
```

Start `cmd.exe` in Administrator mode.

```
WINDOWS + R -> type cmd -> CTRL + SHIFT + ENTER.
```

Issue the appropriate command depending on where you want to store the downloaded models.

Now you can drop all model files in that folder on another partition. Do that now.

Step 6:

Create a `json` file for [LlamaGPTJ-chat](#) to avoid typing lots of commands before using [LlamaGPTJ-chat](#).

The file should be created in `C:\Users\YOUR_USERNAME\gpt4all\`.

```
type nul >> "%USERPROFILE%\gpt4all\chat.json"
```

```
notepad "%USERPROFILE%\gpt4all\chat.json"
```

Paste the contents given below.

```
{
// CMD in Admin mode
// mklink /J "C:\Users\AppuRaja\AppData\Local\nomic.ai\GPT4All\" "N:\GPT4All"
// https://gpt4all.io/
// https://github.com/kuvaus/LlamaGPTJ-chat
// chat-windows-latest-avx2.exe -j %userprofile%\gpt4all\chat.json

// "model": "N:\GPT4All\ggml-gpt4all-j-v1.3-groovy.bin",
// "model": "N:\GPT4All\ggml-mpt-7b-chat.bin",
// "model": "N:\GPT4All\ggml-nous-gpt4-vicuna-13b.bin",
// "model": "N:\GPT4All\ggml-vicuna-7b-1.1-q4_2.bin",
// "model": "N:\GPT4All\ggml-wizardLM-7B.q4_2.bin",
// "model": "N:\GPT4All\ggml-nous-gpt4-vicuna-13b.bin",

"load_json": "N:\GPT4All\gpt4all\chat.json"
"load_log": "N:\GPT4All\chatlog.log",
"save_log": "N:\GPT4All\chatlog.log",
"n_predict": 100,
"top_p": 1.0,
"top_k": 50400,
"temp": 0.9,
"batch_size": 0.9,
"repeat_penalty": 1.1,
"n_batch": 20,
"threads": 4,
"model": "N:\GPT4All\ggml-mpt-7b-chat.bin",
"no-interactive": "false"
}
```

Step 7:

Run [GPT4All](#). Find it in your Start Menu.

To run [LlamaGPTJ-chat](#) command-line model runner, issue this command:

```
chat-windows-latest-avx2.exe -j %USERPROFILE%\gpt4all\chat.json
```

Or, simply:

```
chat-windows-latest-avx2.exe -m "N:\GPT4All\ggml-mpt-7b-chat.bin" -t 4
```

A WinBATCH/BASH/FISH/ZSH script should work as well. The AI model `ggml-wizardLM-7B.q4_2.bin` generated almost perfect code, the last time I checked.

Feel free to experiment with the parameters. Change models and parameters in the `json` file. You need the `json` file to quickly insert flags into the [LlamaGPTJ-chat](#) model runner. The `json` file is unrelated to the [GPT4All](#) GUI model runner program.

You might want to keep an eye on a similar program [koboldcpp](#). It can run many AI model formats. The program is small, nearly 20MB.

For more details on [koboldcpp](#):

<https://github.com/LostRuins/koboldcpp>

<https://huggingface.co/TheBloke/WizardCoder-15B-1.0-GGML/blob/main/README.md>

<https://huggingface.co/TheBloke/WizardCoder-15B-1.0-GPTQ>

<https://huggingface.co/TheBloke/MPT-7B-Storywriter-GGML/discussions/2#6475d914e9b57ce0caa68888>

Running [koboldcpp](#):

```
koboldcpp.exe --help
```

```
koboldcpp.exe --model "N:\GPT4All\ggml-wizard-13b-uncensored.bin" --smartcontext --nommap
```

```
koboldcpp.exe --model "N:\GPT4All\WizardCoder-15B-1.0.ggmlv3.q4_1.bin" --smartcontext --nommap
```

Drop [koboldcpp](#)'s compiled binary in the same folder where you have your [LlamaGPTJ-chat](#) (`chat-windows-latest-avx2.exe`).

NOTE:

1. Too **high CPU/RAM usage**. Jokes apart, be prepared to heat up your room with your computer.
2. Responses are generated very slowly, nearly one word per second in my setup. These two runner apps do not support GPUs. If they had support for at least the integrated graphics processor of the CPU (e.g., the iGPU in Intel CPUs), generating responses would have been way faster.

3. Requires a reasonably capable machine; at least 16GB DDR3 RAM and a 2GHz quad-core CPU. That requirement is the real minimum.
4. The models cannot be integrated into commercial chatbots, like AI chatbots in customer care services. Most of these models are derivative works of Meta's LLaMA model. Some of them are fine-tuned using OpenAI's responses.