Søren L Kristiansen
Aug 8, 2022 · 5 min read · Listen

TensorFlow with GPU support on Apple Silicon Mac with Homebrew and without Conda / Miniforge

TLDR; Run `brew install hdf5`, then `pip install tensorflow-macos` and finally `pip install tensorflow-metal`. You're done ✅.

With the release of Apple Silicon Macs, we finally have a way to (easily) install and run TensorFlow with GPU support on macOS. Unfortunately, Apple's installation instructions are not very clear, and they expect you to use a mix of `conda` and `pip`. If you are like me and already have a setup for running virtual environments that does not involve Conda, and you'd like to keep it that way, Apple's instructions are not very helpful. In this short post, I will show you how to get TensorFlow up and running with GPU support on your Apple Silicon Mac without installing Miniforge or anything else related to Conda!

First a quick bit of background. I have nothing against Conda. But, for all my projects, both work and personal, I use Homebrew for managing non-Python dependencies and pip for Python-dependencies. For managing virtual environments, I use venv which is Python's relatively new, built-in implementation of virtual environments. This setup works well for me, and I'd like to keep it. This post explains how I managed to do that. Note that you do not need to use venv to follow my way, you could just as well use virtualenv (or no virtual environment at all if you are such inclined, though I would not recommend that).

Apple's instructions for installing TensorFlow will tell you to install Miniforge which is a minimal installer for Conda. However, Conda is only used for installing a single package, namely tensorflow-deps. This package is not available on PyPI, so that's our first roadblock. However, if you download the tensorflow-deps package file and inspect it, you will see that all it does is install a few dependencies that can easily be installed using pip! As you can see in the snippet below, those packages are grpcio, h5py and NumPy — and all of these are available on PyPI (and hence, installable through pip).

```
package:
  name: tensorflow-deps
  version: 2.9.0
build:
  number: '0'
  string: '0'
requirements:
  build: []
  run:
    - grpcio >=1.37.0,<2.0
    - h5py >=3.6.0,<3.7
    - numpy >=1.22.3,<1.23.0
    - python
about:
  license: Apache2
  license_family: Apache
  summary: Metapackage for installing dependencies of TensorFlow
extra:
  copy_test_source_files: true
  final: true
```

meta.yaml hosted with ♥ by GitHub — view raw

So, instead of installing Conda, we could just install these packages from PyPI. In fact, as it turns out, we do not even need to install them explicitly since they are dependencies of TensorFlow, so we will get them automatically when installing TensorFlow. There are two things to be aware of, though:

1. The h5py package depends on hdf5 which cannot be installed through pip.
2. The Conda package meta.yaml file specifies maximum version numbers for the three packages, grpcio, h5py and NumPy. Letting the TensorFlow setup procedure install those packages automatically as dependencies may install packages that are exceed the specified version numbers (and will indeed do so for two of the packages).

Let's first handle the h5py package and its non-Python dependency, hdf5. Since hdf5 cannot be installed through pip, simply running `pip install h5py` will fail because of the missing dependency. Luckily, there is a Homebrew package for hdf5. Thus, assuming you have Homebrew installed, all you need to do is run the following:

```
brew install hdf5
```

In case you do not already have Homebrew, you should install it first by following the instructions. It really is very easy and it's useful for all sorts of packages.

Having installed hdf5, we can move on to the second point above, namely the maximum version numbers specified for grpcio, h5py and NumPy in the tensorflow-deps package (see the meta.yaml file above). As of this writing, if you install TensorFlow and let the setup procedure automatically install the dependencies, both the h5py and NumPy versions will exceed the maximum version specified in the YAML. Apple makes no indication as to why they chose to limit the versions in the Conda package but if you follow Apple's instructions for installing on an **Intel** mac, they apply no version limits. In short, I have chosen to ignore those version limits and just let the TensorFlow setup procedure itself handle its dependencies, trusting that it knows best which versions of the dependencies it supports. So far I have seen no problems. If you do experience problems, you may want to install the three packages manually (remember to do it inside the relevant virtual environment, assuming you do use those), specifying the version requirements from the meta.yaml file in Apple's tensorflow-deps package:

```
pip install "grpcio>=1.37.0,<2.0" "h5py>=3.6.0,<3.7" "numpy>=1.22.3,<1.23.0"
```

⚠️ **NOTE:** Before running the above, you should check the actual meta.yaml file in the tensorflow-deps package and make sure that you use the version numbers specified there as they may have changed since the time of writing.

We now need to install TensorFlow itself using pip. The regular tensorflow package on PyPI does not install on Apple Silicon. Instead, you should install the tensorflow-macos package. (This aligns with Apple's own installation instructions). If you are using any kind of virtual environments, this is the time to create and activate a new one. Then run the following:

```
pip install tensorflow-macos
```

Behold! 👀 You now have TensorFlow on your Apple Silicon Mac! We are not there yet, though, as you will not yet have GPU support. You can see that by running `tf.config.list_physical_devices("GPU")` in Python to list the GPUs available to TensorFlow. As shown in the screenshot below, this will result in an empty list:

```
>>> import tensorflow as tf
>>> tf.config.list_physical_devices("GPU")
[]
>>>
```

You should now install GPU acceleration by installing tensorflow-metal as follows:

```
pip install tensorflow-metal
```

You can now run the above Python command again, and this time the returned list will contain your GPU:

```
>>> import tensorflow as tf
>>> tf.config.list_physical_devices("GPU")
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
>>>
```

Congratulations! 🎉 Now you have TensorFlow with GPU acceleration on your Apple Silicon Mac 🖥️!

. . .

To get a quick glimpse of the impact of training with a GPU, I downloaded the code and data for the Keras Image segmentation with a U-Net-like architecture example. I ran it on both my M1 MacBook Pro, my Intel Mac Pro (AMD Radeon Pro W5700X 16 GB) and my AMD Ryzen PC (NVIDIA RTX 3090). On both Macs, I have run with and without installing the tensorflow-metal package to compare GPU and no GPU. On the PC, I tested with GPU acceleration only. The results should be taken with a large grain of salt due to different CPUs, amounts of RAM and SSD speed, etc., which may cause different bottlenecks on the different machines.

- MacBook Pro, Apple Silicon M1, GPU using Metal: 83 seconds / epoch
- MacBook Pro, Apple Silicon, M1, CPU: 214 seconds / epoch
- Mac Pro, Intel (Radeon W5700X), GPU using Metal: 166 seconds / epoch
- Mac Pro, Intel (Radeon W5700X), CPU: 328 seconds / epoch
- **Windows PC, Intel (RTX 3090), GPU using CUDA: 21 seconds / epoch**

It is unsurprising that the RTX 3090 is the fastest. But interesting to see that the M1 is twice as fast as the Mac Pro with Radeon W5700X.

. . .

Finally, to sum up, all you need to get TensorFlow running with GPU support on your M1 or M2 Mac is to install hdf5 through Homebrew and then install both tensorflow-macos and tensorflow-metal through pip.

Good luck! 🍀

TensorFlow · Apple Silicon · Machine Learning · Gpu · Tutorial