**Installing TensorFlow on macOS using ARM64 architecture/GPU**

The following is what I did to successfully install and run TensorFlow on my Macbook Pro M1 Max running macOS Ventura 13.1. Note that Apple gives the following requirements for running TensorFlow:

* Mac computers with Apple silicon or AMD GPUs
* macOS 12.0 or later
* Python 3.8 or later
* Xcode command line tools (In your terminal: xcode-select –install to install, but I think it should be there already)

1. Completely uninstall all conda/Anaconda/miniconda/miniforge environments. I did this because I thought I had conflicts between several environments that were preventing me from successfully installing TensorFlow.
2. **Install miniforge3**

In your web browser, navigate to the miniforge3 download site <https://github.com/conda-forge/miniforge/releases> and download the release for OSX/arm64 (Apple Silicon) release. At the time of this writing, the file is Miniforge3-MacOSX-arm64.sh, which also installs Python 3.10 in your base environment.

Once the shell script has been downloaded, open a terminal window, set the directory to the download folder, and enable the shell script for execution with the command:

chmod +x Miniforge3-MacOSX-arm64.sh

Execute the script using the following shell command in the terminal window:

sh Miniforge3-MacOSX-arm64.sh

You should be prompted to set the installation directory for miniforge3. I set it to my Applications folder under my user home directory, e.g., “~/Applications/miniforge3”. Note that this is NOT the same directory as the Applications folder on your macOS hard drive.

At the end of the installation, the installer will ask about letting it do the initialization of the Conda environment. I let it do that. You can run the command conda info in the terminal to see the environment information for the installed conda, and conda --version to check the installed version. I also find that closing the terminal and reopening it helps to make sure the terminal is updated, too.

**3. Conda setup**

Configure conda for arm64 by verifying that the miniforge3 installer installed the following code in the ~/.zshrc file. (It did this automatically for me on installation.) If it hasn’t, enter this code in the file. You can open the file with the command open ~/.zshrc

# Create ARM conda environment

create\_ARM\_conda\_environment () {

# example usage: create\_ARM\_conda\_environment myenv\_x86 python=3.9

CONDA\_SUBDIR=osx-arm64 conda create -n $@

conda activate $1

}

Be sure to save the file if you have edited it before closing it.

Note that this shortcut will allow you to create an arm64 environment where Python packages can be installed. The conda\_subdir environment variable ensures that conda will install packages from the osx-arm64 sub-directories in your package download channels.

**4. Create conda environment**

Using the shortcut command created above, use your terminal window to create a local virtual environment for your TensorFlow packages:

create\_ARM\_conda\_environment <environment\_name> python=<python\_version>

Replace <environment\_name> with your own name for the environment that reflects something about the environment. For example, I used “python3.9\_tf\_ARM” to note that the environment is for python version 3.9.x using TensorFlow (“tf”) and uses arm64.

Replace <python\_version> with the version of python that is compatible with TensorFlow. For my initial installation I used version 3.9.15 because this seemed to be supported by the developer community on the various boards.

You will be prompted with the list of packages to be installed as asked whether to proceed (default is “yes”, so just hit Return).

Note: In the messages during the installation of python I noticed that it recommended that the version of conda installed during the miniconda3 installation was old and should be updated. I have not done this yet.

Whenever you want to use this environment, activate the environment using:

conda activate <environment\_name>

You should see a prompt with the environment name.

**5. TensorFlow installation**

If it isn’t already active, activate the environment you just created using the conda activate <environment\_name> command in your terminal window.

Install arm64-specific TensorFlow dependencies:

conda install -c apple tensorflow-deps

Note that “-c” refers to the channel, which is “apple” in this case.

You will get various messages about the installation as it is proceeding, including a listing of the files to be installed and whether to proceed. Verify that you are installing in the arm64 environment you created. When it finishes, you should not have any error messages and should be back to the environment prompt.

Install TensorFlow for macOS:

pip install tensorflow-macos==2.9

Note that for now you need to specify the version shown rather than the latest version (2.11) to make this work. I’m going to keep trying to make an environment that uses the latest version, but for now there seems to be general agreement on the development message boards that this version works, while the latest version has issues.

When the installation finishes, you should receive a message showing all the files that were successfully installed.

Install the libraries to make your TensorFlow utilize your arm64/GPU architecture:

pip install tensorflow-metal==0.5.0

Note that for now you need to specify the version shown rather than the latest version (0.7.0) to make this work. There seems to be general agreement on the development message boards that this version works.

When the installation finishes, you should receive a message showing all the files that were successfully installed.

**6. Verify TensorFlow installation**

Activate the environment you created and launch the python environment:

conda activate <environment\_name> && python

In your terminal window, you should see the python prompt (“>>>”). As python is launching, you probably will see output showing the version of python that is being used (version 3.9.15 if you followed the instructions above).

In a terminal window, import TensorFlow and create a constant:

>>> import tensorflow as tf

>>> tf.constant([1,2,3])

Metal device set to: Apple M1 Max

systemMemory: 32.00 GB

maxCacheSize: 10.67 GB

.

.

.

<tf.Tensor: shape=(3,), dtype=int32, numpy=array([1, 2, 3], dtype=int32)>

>>>

To check if TensorFlow is using GPU acceleration on your machine:

>>> tf.config.list\_physical\_devices('GPU')

[PhysicalDevice(name='/physical\_device:GPU:0', device\_type='GPU')]

>>>

I also ran this [example TensorFlow program](https://developer.apple.com/metal/tensorflow-plugin/) from Apple to verify everything was working:

import tensorflow as tf

cifar = tf.keras.datasets.cifar100

(x\_train, y\_train), (x\_test, y\_test) = cifar.load\_data()

model = tf.keras.applications.ResNet50(

include\_top=True,

weights=None,

input\_shape=(32, 32, 3),

classes=100,)

loss\_fn = tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True)

model.compile(optimizer="adam", loss=loss\_fn, metrics=["accuracy"])

model.fit(x\_train, y\_train, epochs=5, batch\_size=64)