GPU support

GPU support is available for Ubuntu and Windows with CUDA®-enabled cards.

TensorFlow GPU support requires an assortment of drivers and libraries. To simplify installation and avoid library conflicts, we recommend using a TensorFlow Docker image with GPU support (https://www.tensorflow.org/install/docker) (Linux only). This setup only requires the NVIDIA® GPU drivers (https://www.nvidia.com/drivers).

These install instructions are for the latest release of TensorFlow. See the tested build configurations (https://www.tensorflow.org/install/source#linux) for CUDA® and cuDNN versions to use with older TensorFlow releases.

Pip package

See the pip install guide (https://www.tensorflow.org/install/pip) for available packages, systems requirements, and instructions. The TensorFlow `pip` package includes GPU support for CUDA®-enabled cards:

```
$ pip install tensorflow
```

This guide covers GPU support and installation steps for the latest *stable* TensorFlow release.

Older versions of TensorFlow

For releases 1.15 and older, CPU and GPU packages are separate:

```
$ pip install tensorflow==1.15      # CPU
$ pip install tensorflow-gpu==1.15  # GPU
```

Hardware requirements

The following GPU-enabled devices are supported:

- NVIDIA® GPU card with CUDA® architectures 3.5, 5.0, 6.0, 7.0, 7.5, 8.0 and higher than 8.0. See the list of CUDA®-enabled GPU cards (https://developer.nvidia.com/cuda-gpus).

- For GPUs with unsupported CUDA® architectures, or to avoid JIT compilation from PTX, or to use different versions of the NVIDIA® libraries, see the Linux build from source (https://www.tensorflow.org/install/source) guide.
- Packages do not contain PTX code except for the latest supported CUDA® architecture; therefore, TensorFlow fails to load on older GPUs when `CUDA_FORCE_PTX_JIT=1` is set. (See Application Compatibility (http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#application-compatibility) for details.)

The error message "Status: device kernel image is invalid" indicates that the TensorFlow package does not contain PTX for your ecture. You can enable compute capabilities by building TensorFlow from source (https://www.tensorflow.org/install/source).

Software requirements

The following NVIDIA® software must be installed on your system:

- NVIDIA® GPU drivers (https://www.nvidia.com/drivers) —CUDA® 11.0 requires 450.x or higher.
- CUDA® Toolkit (https://developer.nvidia.com/cuda-toolkit-archive) —TensorFlow supports CUDA® 11 (TensorFlow >= 2.4.0)
- CUPTI (http://docs.nvidia.com/cuda/cupti/) ships with the CUDA® Toolkit.
- cuDNN SDK 8.0.4 (https://developer.nvidia.com/cudnn) cuDNN versions (https://developer.nvidia.com/rdp/cudnn-archive)).
- *(Optional)* TensorRT 6.0 (https://docs.nvidia.com/deeplearning/sdk/tensorrt-install-guide/index.html) to improve latency and throughput for inference on some models.

Linux setup

The `apt` instructions below are the easiest way to install the required NVIDIA software on Ubuntu. However, if building TensorFlow from source (https://www.tensorflow.org/install/source), manually install the software requirements listed above, and consider using a `-devel` TensorFlow Docker image (https://www.tensorflow.org/install/docker) as a base.

Install CUPTI (http://docs.nvidia.com/cuda/cupti/) which ships with the CUDA® Toolkit. Append its installation directory to the `$LD_LIBRARY_PATH` environmental variable:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64
```

Install CUDA with apt

This section shows how to install CUDA® 11 (TensorFlow >= 2.4.0) on Ubuntu 16.04 and 18.04. These instructions may work for other Debian-based distros.

n: Secure Boot (https://wiki.ubuntu.com/UEFI/SecureBoot) complicates installation of the NVIDIA driver and is beyond the scop instructions.

Ubuntu 18.04 (CUDA 11.0)

```
# Add NVIDIA package repositories
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu18
$ sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu
$ sudo add-apt-repository "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1
$ sudo apt-get update

$ wget http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/nv
(http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/nvidia-machine-learning-repo-ub

$ sudo apt install ./nvidia-machine-learning-repo-ubuntu1804_1.0.0-1_amd64.deb
$ sudo apt-get update

$ wget https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86_64/l
$ sudo apt install ./libnvinfer7_7.1.3-1+cuda11.0_amd64.deb
$ sudo apt-get update

# Install development and runtime libraries (~4GB)
$ sudo apt-get install --no-install-recommends \
    cuda-11-0 \
    libcudnn8=8.0.4.30-1+cuda11.0  \
    libcudnn8-dev=8.0.4.30-1+cuda11.0

# Reboot. Check that GPUs are visible using the command: nvidia-smi

# Install TensorRT. Requires that libcudnn8 is installed above.
$ sudo apt-get install -y --no-install-recommends libnvinfer7=7.1.3-1+cuda11.0 \
    libnvinfer-dev=7.1.3-1+cuda11.0 \
    libnvinfer-plugin7=7.1.3-1+cuda11.0
```

Ubuntu 16.04 (CUDA 11.0)

```
# Add NVIDIA package repositories
# Add HTTPS support for apt-key
$ sudo apt-get install gnupg-curl
$ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/cuda-ubuntu16
$ sudo mv cuda-ubuntu1604.pin /etc/apt/preferences.d/cuda-repository-pin-600
$ sudo apt-key adv --fetch-keys https://developer.download.nvidia.com/compute/cuda/repos/ubuntu
$ sudo add-apt-repository "deb https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1
```

```
$ sudo apt-get update
$ wget https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64/r
$ sudo apt install ./nvidia-machine-learning-repo-ubuntu1604_1.0.0-1_amd64.deb
$ sudo apt-get update
$ wget https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64/l
$ sudo apt install ./libnvinfer7_7.1.3-1+cuda11.0_amd64.deb
$ sudo apt-get update

# Install development and runtime libraries (~4GB)
$ sudo apt-get install --no-install-recommends \
    cuda-11-0 \
    libcudnn8=8.0.4.30-1+cuda11.0  \
    libcudnn8-dev=8.0.4.30-1+cuda11.0


# Reboot. Check that GPUs are visible using the command: nvidia-smi

# Install TensorRT. Requires that libcudnn7 is installed above.
$ sudo apt-get install -y --no-install-recommends \
    libnvinfer7=7.1.3-1+cuda11.0 \
    libnvinfer-dev=7.1.3-1+cuda11.0 \
    libnvinfer-plugin7=7.1.3-1+cuda11.0 \
    libnvinfer-plugin-dev=7.1.3-1+cuda11.0
```

Windows setup

See the hardware requirements (#hardware_requirements) and software requirements (#software_requirements) listed above. Read the CUDA® install guide for Windows (https://docs.nvidia.com/cuda/cuda-installation-guide-microsoft-windows/).

Make sure the installed NVIDIA software packages match the versions listed above. In particular, TensorFlow will not load without the `cuDNN64_8.dll` file. To use a different version, see the Windows build from source (https://www.tensorflow.org/install/source_windows) guide.

Add the CUDA®, CUPTI, and cuDNN installation directories to the `%PATH%` environmental variable. For example, if the CUDA® Toolkit is installed to `C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0` and cuDNN to `C:\tools\cuda`, update your `%PATH%` to match:

```
C:\>  SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\bin;%PATH%
C:\>  SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\extras\CUPTI\lib64;%PAT
C:\>  SET PATH=C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.0\include;%PATH%
C:\>  SET PATH=C:\tools\cuda\bin;%PATH%
```

Last updated 2021-04-07 UTC.