GRAPHCORE

Jupyter Notebook Quick Start

Version latest

Graphcore Ltd

Feb 27, 2024

CONTENTS

This Quick Start guide describes how to use IPUs from a Jupyter Notebook.

CHAPTER

ONE

OVERVIEW

This Quick Start guide describes how to run the MNIST PyTorch application (from the Graphcore examples repository) in a Jupyter Notebook on the IPU. We start a Jupyter server on a host with access to IPUs. Then, we set up port forwarding so that you can access the Jupyter server from a browser running on your local machine.

We suggest watching the Fundamentals of the IPU and Poplar video, which introduces the IPU architecture and programming model.

You can also read the IPU Programmer's Guide and Switching from GPUs to IPUs for Machine Learning Models for more details on these topics.

1.1 Prerequisites

Before following the instructions in this Quick Start guide, you must be able to log into a system with access to IPUs. Details are given in the getting started guide for your system.

There are several ways you can run Jupyter Notebook applications:

1. Directly on a cloud-based system or on IPU Pod hardware using the setup described in this document.
2. In VS Code following the setup instructions in the tutorial Using VS Code.
3. In a Docker container that has already been setup. The Demo-in-a-Box GitHub repository contains scripts and Docker files to help you get set up quickly.

For more information about using the Graphcore Docker images for Jupyter, refer to Using IPUs from Docker.

1.2 Summary of this document

This document will guide you through the steps required to *prepare for running the application described in this guide*, while also leading you through the steps required to *prepare for running your own Jupyter Notebook application* on any IPU host.

Step-by-step, this guide will show you how to:

- On the IPU host:
    1. *Clone the Graphcore examples repo*
    2. *Define environment variables*
    3. *Enable the Poplar SDK*
    4. *Create a Python virtual environment* and activate it
    5. *Install the ML framework wheels*
    6. *Install Jupyter*
    7. *Start the Jupyter server*

- On your local machine:
  1. *Connect to the Jupyter server*
  2. *Run the application*

You can expect the setup steps to take no more than 30 minutes.

CHAPTER

TWO

SETUP FOR THIS QUICK START GUIDE

This chapter describes the setup steps that are specific for running the application in this Quick Start guide.

- *Clone the Graphcore examples repo*.
- *Define environment variables*.

2.1 Clone the Graphcore examples repo

You need to clone the Graphcore examples repository on some systems as detailed in Table 2.1.

Table 2.1: Systems that need the Graphcore tutorials and examples repositories to be cloned

System	Clone repos?	Comment
Pod system	Yes	You can clone the tutorials and examples repos in any location.
Graphcloud	Yes	You can clone the tutorials and examples repos in any location.
Gcore Cloud	No	The tutorials and examples have already been cloned in ~/graphcore/tutorials and ~/graphcore/examples respectively.

If you don't need to clone the examples repository, then go straight to Section 2.2, Define environment variables.

You can clone the examples repository into a location of your choice.

To clone the examples repository for the latest version of the Poplar SDK:

```
$ cd ~/[base_dir]
$ git clone https://github.com/graphcore/examples.git
```

where [base_dir] is a location of your choice. This will install the contents of the examples repository under ~/[base_dir]/examples. The tutorials are in ~/[base_dir]/examples/tutorials.

Note: If you are using a version of the Poplar SDK prior to version 3.2, then refer to Section 7, Install examples and tutorials for older Poplar SDK versions for how to install examples and tutorials.

2.2 Define environment variables

In order to simplify running the tutorials, we define the environment variable POPLAR_TUTORIALS_DIR that points to the location of the cloned tutorials.

Pod systems

Graphcloud

Gcore Cloud

```
$ export POPLAR_TUTORIALS_DIR=~/[base_dir]/examples/tutorials
```

[base_dir] is the location where you *installed the Graphcore tutorials*.

```
$ export POPLAR_TUTORIALS_DIR=~/[base_dir]/examples/tutorials
```

[base_dir] is the location where you *installed the Graphcore tutorials*.

```
$ export POPLAR_TUTORIALS_DIR=~/graphcore/tutorials
```

We also use the environment variable POPLAR_SDK_ENABLED. This environment variable is defined when Poplar is enabled (Section 3.1, Enable the Poplar SDK) and defines the location of the poplar directory in the SDK directory.

CHAPTER

THREE

SETUP FOR JUPYTER NOTEBOOKS

Before you run any application on a Jupyter Notebook, you must complete the following setup steps:

1. *Enable the Poplar SDK*
2. *Create a Python virtual environment* and activate it
3. *Install the ML framework wheels*
4. *Install Jupyter*
5. *Start the Jupyter server*
6. *Connect to the Jupyter server*

This assumes you do not have any of the software installed already. If you do, you can just skip the appropriate sections.

3.1 Enable the Poplar SDK

Note: It is best if you use the latest version of the Poplar SDK.

On some systems you must explicitly enable the Poplar SDK before you can use PyTorch or TensorFlow for the IPU, or the Poplar Graph Programming Framework. On other systems, the SDK is enabled as part of the login process.

Table 3.1 defines whether you have to explicitly enable the SDK and where to find it.

Table 3.1: Systems that need the Poplar SDK to be enabled and the SDK location

System	Enable SDK?	SDK location
Pod system	Yes	The SDK is in the directory where you extracted the SDK tarball.
Graphcloud	Yes	/opt/gc/poplar_sdk-ubuntu_18_04-[poplar_ver]+[build] where [poplar_ver] is the software version number of the Poplar SDK and [build] is the build information.
Gcore Cloud	No	The SDK has been enabled as part of the login process.

To enable the Poplar SDK:

SDK Versions 2.6 and later

SDK Versions earlier than 2.6

For SDK versions 2.6 and later, there is a single enable script that determines whether you are using Bash or Zsh and runs the appropriate scripts to enable both Poplar and PopTorch/PopART.

Source the single script as follows:

```
$ source [path_to_SDK]/enable
```

where [path_to_SDK] is the location of the Poplar SDK on your system.

For SDK versions earlier than 2.6, there are only Bash scripts available and you have to source the Poplar and PopART scripts separately.

Note: You only have to source the PopART enable script if you are using PopTorch or PopART.

Source the scripts as follows:

```
$ source [path_to_SDK]/poplar-ubuntu_[os_ver]-[poplar_ver]+[build]/enable.sh
$ source [path_to_SDK]/popart-ubuntu_[os_ver]-[poplar_ver]+[build]/enable.sh
```

where [path_to_SDK] is the location of the Poplar SDK on your system. [os_ver] is the version of Ubuntu on your system, [poplar_ver] is the software version number of the Poplar SDK and [build] is the build information.

Note: You must source the Poplar enable script for each new shell. You can add this source command to your .bashrc (or .zshrc for SDK versions later than 2.6) to do this on a more permanent basis.

If you attempt to run any Poplar software without having first sourced this script, you will get an error from the C++ compiler similar to the following (the exact message will depend on your code):

```
fatal error: 'poplar/Engine.hpp' file not found
```

If you try to source the script after it has already been sourced, then you will get an error similar to:

```
ERROR: A Poplar SDK has already been enabled.
Path of enabled Poplar SDK: /opt/gc/poplar_sdk-ubuntu_20_04-3.2.0-7cd8ade3cd/poplar-ubuntu_20_04-3.2.0-7cd8ade3cd
If this is not wanted then please start a new shell.
```

You can verify that Poplar has been successfully set up by running:

```
$ popc --version
```

This will display the version of the installed software.

3.2 Create a Python virtual environment

It is good practice to work in a different Python virtual environment for each framework or even for each application. This section describes how you create and activate a Python virtual environment.

Note: You must activate the Python virtual environment before you can start using it.

The virtual environment must be created for the Python version you will be using. This cannot be changed after creation. Create a new Python virtual environment with:

```
$ virtualenv -p python3 [venv_name]
```

where [venv_name] is the location of the virtual environment.

Note: Make sure that the version of Python that is installed is compatible with the version of the Poplar SDK that you are using. See Supported tools in the Poplar SDK release notes for information about the supported operating systems and versions of tools.

To start using a virtual environment, activate it with:

```
$ source [venv_name]/bin/activate
```

where [venv_name] is the location of the virtual environment.

Now all subsequent installations will be *local* to that virtual environment.

3.3 Install the ML framework wheels

You need to install the Poplar SDK wheels if you are using PyTorch, PyTorch Geometric or TensorFlow in the virtual environment you have activated. Refer to the PyTorch Quick Start and TensorFlow 2 Quick Start for details of installing the relevant wheels. For this Quick Start guide, we are using PyTorch.

This section describes how to install a wheel file containing PopTorch, a set of extensions to enable PyTorch models to run on the IPU.

Note: You should activate the Python virtual environment you created for PyTorch (Section 3.2, Create a Python virtual environment) before performing the setup in this section.

The wheel file is included with the Poplar SDK and has a name of the form:

```
poptorch-[sdk_ver]+*.whl
```

where [sdk_ver] is the version of the Poplar SDK. An example of the wheel file is:

```
poptorch-3.2.0+109946_bb50ce43ab_ubuntu_20_04-cp38-cp38-linux_x86_64.whl
```

Install the PopTorch wheel file using the following command:

```
$ python -m pip install ${POPLAR_SDK_ENABLED?}/../poptorch-*.whl
```

where POPLAR_SDK_ENABLED is the location of the Poplar SDK defined when the SDK was enabled. The ? ensures that an error message is displayed if Poplar has not been enabled.

To confirm that PopTorch has been installed, you can use:

```
pip list | grep poptorch
```

For the example wheel file, the output will be:

```
poptorch        3.2.0+109946
```

You can also test that the module has been installed correctly by attempting to import it in Python, for example:

```
$ python3 -c "import poptorch; print(poptorch.__version__)"
```

3.4 Install Jupyter

After enabling the SDK and installing the desired wheel, install Jupyter Notebook and the IPython kernel:

```
$ python -m pip install jupyter ipykernel
```

Note: You can also use JupyterLab. Install JupyterLab and the IPython kernel with:

```
$ python -m pip install jupyter-lab ipykernel
```

3.5 Start the Jupyter server

You need to start the Jupyter server in the directory where you will be accessing the notebook. This will be the top most directory that you can access from your Jupyter server. To avoid restarting the notebook for every tutorial, it is simplest to navigate to the folder containing all tutorials. For this Quick Start guide, navigate to:

```
$ cd $POPLAR_TUTORIALS_DIR
```

Then, start a Jupyter server in "no browser" mode on a specified port:

```
$ jupyter-notebook --no-browser --port <port number>
```

For example:

```
$ jupyter-notebook --no-browser --port 44300

 [I 11:57:42.444 NotebookApp] Jupyter Notebook 6.4.4 is running at:
 [I 11:57:42.444 NotebookApp] http://localhost:44300/?token=cb37a6b1ffb29dc99dea3d8e4153cef232b9f067ecd50f64
 [I 11:57:42.444 NotebookApp]  or http://127.0.0.1:44300/?token=cb37a6b1ffb29dc99dea3d8e4153cef232b9f067ecd50f64
 [I 11:57:42.444 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
 [C 11:57:42.447 NotebookApp]

     To access the Notebook, copy and paste one of these URLs:
     http://localhost:44300/?token=cb37a6b1ffb29dc99dea3d8e4153cef232b9f067ecd50f64
     or http://127.0.0.1:44300/?token=cb37a6b1ffb29dc99dea3d8e4153cef232b9f067ecd50f64
```

Note: If you are using JupyterLab, then start it with:

```
$ jupyter-lab --no-browser --port <port number>
```

The URLs that appear are going to be used to connect to this server later.

At this point the Jupyter server is running and will block your terminal. You will need to keep the server running while you want your sessions to remain active. For more information on how to manage long running tasks on linux, check out this guide on how to run and control background processes on Linux.

3.6 Connect to the Jupyter server

In order to connect to the Jupyter server we will need to connect via SSH with port forwarding to your IPU host. To do this you must add the -NL option to the ssh command that you run from your local machine:

```
$ ssh -NL <port_on_local_machine>:localhost:<port_on_server> <user>@<ip>
```

where <port_on_local_machine> is the port on your local machine which the Jupyter server will forward to, <port_on_server> is the port on which the Jupyter server is running and <user>@<ip> is your username and the server IP address.

For example:

```
$ ssh -NL 8888:localhost:44300 joes@123.456.789.012
```

where 8888 is the port on your local machine, 44300 is the port on which the Jupyter server is running, joes is the username and 123.456.789.012 is the IP address of the IPU host.

Note: This command must run continuously while you access the Jupyter server from your browser.

Now you are ready to *Run the application*.

CHAPTER

FOUR

RUN THE APPLICATION

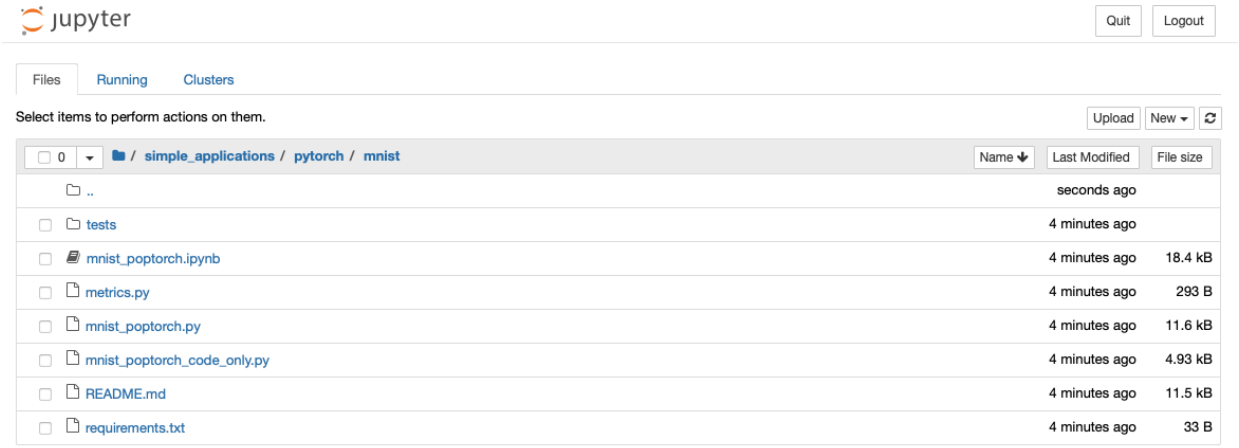
This section describes how you run the MNIST PyTorch application.

1. Open a browser on your local machine and paste one of the URLs that were printed when the server started. For our example, we use:

   ```
   http://localhost:8888/?token=cb37a6b1ffb29dc99dea3d8e4153cef232b9f067ecd50f64
   ```

2. Your browser will connect to the Jupyter server.

3. Navigate in your browser to the Notebook for the PyTorch MNIST application: simple_applications > python > mnist.



4. Follow the instructions in the mnist_poptorch.ipynb Notebook to run the application.

CHAPTER

FIVE

NEXT STEPS

This page lists sources of information to help you learn about programming for, and running code on, the IPU.

5.1 Documentation

All documentation is on the Graphcore documentation portal. There you can find user guides for hardware and software, API references and technical notes.

The IPU Programmer's Guide provides an introduction to the IPU architecture, programming model and tools available.

Switching from GPUs to IPUs for Machine Learning Models provides a high-level overview of the programming changes required when switching from GPUs to IPUs.

Technical notes cover a range of topics about implementing and optimising models for the IPU.

5.2 Running applications in Docker

You can run Poplar applications in Docker on a Linux machine using one or more physical IPU devices.

Refer to Using IPUs from Docker for more information.

5.3 Tutorials, examples and applications

You can browse the Graphcore Model Garden to find applications that run on IPUs. You can filter by category, model type and framework.

There are more applications in the examples repository on GitHub.

The latest tutorials and simple code examples are also in the examples repository:

- Tutorials
- Simple applications
- Feature examples

Note: If you are using a version of the Poplar SDK prior to version 3.2, then refer to Section 7, Install examples and tutorials for older Poplar SDK versions for how to install examples and tutorials.

5.4 Other support

- When looking for answers or asking questions on StackOverflow, use the tag "ipu".
- You can request support on the Graphcore Support portal by clicking on the **Submit a ticket** link.
- For general help, discussions and announcements, please join our Graphcore Slack Community.

CHAPTER

SIX

TRADEMARKS & COPYRIGHT

Graphcloud®, Graphcore®, Poplar® and PopVision® are registered trademarks of Graphcore Ltd.

Bow™, Bow-2000™, Bow Pod™, Colossus™, In-Processor-Memory™, IPU-Core™, IPU-Exchange™, IPU-Fabric™, IPU-Link™, IPU-M2000™, IPU-Machine™, IPU-POD™, IPU-Tile™, PopART™, PopDist™, PopLibs™, PopRun™, PopTorch™, Streaming Memory™ and Virtual-IPU™ are trademarks of Graphcore Ltd.

All other trademarks are the property of their respective owners.

Copyright © 2022 Graphcore Ltd. All rights reserved.

CHAPTER

SEVEN

INSTALL EXAMPLES AND TUTORIALS FOR OLDER POPLAR SDK VERSIONS

This section describes how to install the Graphcore examples and tutorials if you are using a version of the Poplar SDK prior to version 3.2.

7.1 Clone the Graphcore tutorials

If you are using a version of the Poplar SDK prior to version 3.2, you will need to use the old GitHub tutorials repository. You will also have to checkout a branch of the tutorials repository corresponding to the version of the Poplar SDK you are using.

You can clone the tutorials repository into a location of your choice.

```
$ cd ~/[base_dir]
$ git clone https://github.com/graphcore/tutorials.git
$ cd tutorials
$ git checkout sdk-release-[poplar-ver]
```

where [base_dir] is a location of your choice and [poplar-ver] is the version of the Poplar SDK that you are using. This will install the contents of the tutorials under ~/[base_dir]/tutorials. The tutorials are in ~/[base_dir]/tutorials.

For example to checkout the tutorials repo for Poplar SDK version 3.1:

```
$ cd ~/[base_dir]
$ git clone https://github.com/graphcore/tutorials.git
$ cd tutorials
$ git checkout sdk-release-3.1
```

7.2 Clone the Graphcore examples

If you are using a version of the Poplar SDK prior to version 3.2, you will need to checkout a specific tagged version of the examples repository corresponding to the version of the Poplar SDK you are using.

You can clone the examples repository into a location of your choice.

```
$ cd ~/[base_dir]
$ git clone https://github.com/graphcore/examples.git
$ cd examples
$ git checkout tags/[tag_name]
```

where [base_dir] is a location of your choice and [tag_name] is the name of the tagged commit corresponding to the version of the Poplar SDK that you are using. The tags are of the form v3.2.0 for the code compatible with each SDK release.

This will install the contents of the examples repository under ~/[base_dir]/examples.

There is a list of tagged versions.

For example, to checkout the tagged version corresponding to Poplar SDK 3.1:

```
$ cd ~/[base_dir]
$ git clone https://github.com/graphcore/examples.git
$ cd examples
$ git checkout tags/v3.1.0
```