GRAPHCORE

# Getting Started with Graphcloud

*Version latest*

**Graphcore Ltd**

**Aug 25, 2021**

# CONTENTS

CHAPTER

ONE

# GRAPHCLOUD OVERVIEW

Graphcloud is a secure, cloud-based, commercial machine-learning (ML) platform running on Graphcore Mk2 IPU-POD systems hosted by US-based Cirrascale in partnership with Graphcore.

Customers are able to purchase weekly, monthly or annual access to IPU-POD$_{16}$ systems (comprised of four IPU-M2000 and a host server) and IPU-POD$_{64}$ systems (comprised of 16 IPU-M2000 and four host servers). Other options, including larger scale-out systems will be available during 2021.

Invited customers can apply for preview access to Graphcloud – a two week free of charge, 'try before you buy' evaluation on a Graphcore Mk2 IPU-POD$_{16}$. After the two week trial period, you can pay for extended access.

Full pricing details are available from the Cirrascale website or from Graphcore sales.

Graphcloud is fully supported by Graphcore's Poplar SDK to provide a complete, scalable platform for accelerated machine intelligence development. Online support from Graphcore is included with access to Graphcloud.

## 1.1 Prerequisites

This document assumes you are familiar with developing Machine Learning applications using industry-standard frameworks such as TensorFlow or PyTorch. As such it also assumes a knowledge of programming languages such as Python or C++.

The Graphcore Developer page has links to a variety of resources to assist you getting started programming for the IPU. A good place to start is the Fundamentals of the IPU and Poplar video, which introduces the IPU architecture and programming model.

### 1.1.1 Linux and Python commands

You will also need to have some familiarity with the Linux command line and some common Linux tools. Some of the tasks that you may need to perform are summarised below:

- Navigating the file system using commands such as `ls` and `cd`, and shortcuts such as `~`.
- Using login files such as `.profile` and `.bashrc` to configure your environment (setting paths, performing actions that need be done on every login, and so on).
- Understanding file ownership and permissions for users and groups.
- Changing file permissions to allow sharing between multiple users in your organisation (commands: `chmod` and `chown`).
- Logging in to Graphcloud with `ssh`, configuring `ssh` and using `scp` to copy files to and from Graphcloud.
- Using `cp` and `rsync` to copy public examples and data to your user workspace.
- Creating a Python "virtual environment" with the `virtualenv` command; this keeps the Python packages that you install for an application or framework separate. This can avoid problems with conflicting versions and dependencies, for example.

### 1.1.2 Managing the IPU hardware resources

The command `vipu` is the main interface to the Virtual-IPU (V-IPU) software that manages the IPU-POD hardware behind Graphcloud. The partition that provides a set of IPUs for you has already been configured so you should not need to use this.

However, you can use `vipu` to monitor the status of the IPUs and, if necessary, reset them. See Section 5, Troubleshooting for more information.

CHAPTER

TWO

# ACCESSING GRAPHCLOUD

When you have been granted access to the two-week free Graphcloud trial period by Graphcore, you'll be provided with a login to the Graphcore support portal and asked to accept the Graphcore End User License Agreement. You will also be asked to accept Cirrascale's terms and conditions for Graphcloud. Your Graphcore sales contact will agree with you when the trial period will start and finish.

You will then be set up with a user account for Graphcloud. This will require you to use SSH key-based authentication, so you'll need to provide us with a public key.

## 2.1 Your account

### 2.1.1 Logging in

After you have agreed the terms and conditions with Graphcore for the use of the Graphcloud service, you'll need to follow the instructions below to gain access to the vPOD instance, and to register for and use the support portal on the Graphcore website.

**Note:** Graphcore uses SSH encryption for connections to the Graphcloud service, and provides the latest versions of all security certificates to enable you to keep your connection and data secure while you use the service. However, responsibility for the correct use of these tools, and the consequent security of your assets, is your responsibility, and Graphcore will accept no liability for any losses to your data that is incurred as a result of your use of the service.

The secure connection to the server on which your vPOD instance is provisioned is provided using SSH key security. To gain access to your vPOD instance, you will be asked to send an SSH public key to Graphcore.

**Note:** Your secure connection will only work if the corresponding private key(s) are installed and correctly located on the machine you are using to connect to the service.

You will be sent confirmation that Graphcore has installed the public key on the Graphcloud server and the IP address to use to connect to your vPOD instance over SSH. You can make this easier to use by adding it to your SSH configuration file. For example:

```
host gc-graphcloud
hostname 123.456.789.012
user myusername
identityfile ~/.ssh/gclogin.pem
```

You can then log in over SSH using that name. For example:

```
$ ssh gc-graphcloud
```

Note that you will not have root access on the Graphcloud server.

### 2.1.2 Adding more users

You can add more users to your instance by sending their emails, names and SSH public keys to Graphcore, as described above. We will then install these on the server(s) to which your organisation has been granted access.

Note that each user must use their own public SSH key. Multiple users must not use the same key, as this could raise security problems, and also make it difficult to analyse and audit any security issues that may arise from the use of the service.

### 2.1.3 Extending access to Graphcloud

Please let us know as soon as possible if you would like to extend your access to Graphcloud by switching to a paid-for service. Due to high demand, we cannot guarantee that you will automatically be able to switch to paid-for access to IPU-POD$_{16}$ or IPU-POD$_{64}$ in Graphcloud and, therefore, request that you give us much notice as possible. Cirrascale will provide full details of the payment options available.

### 2.1.4 Closing your account

When you close your Graphcloud account all of your data will be wiped from the server. This includes:

- Software and data files
- Any log files

You need to explicitly copy any data from the sever that you want to keep (using `scp`, for example).

## 2.2 Configuration overview

You will be allocated your IPU-POD$_{16}$ as an entity called “vPOD16” with the following properties:

- Secure IPU-POD$_{16}$ (four IPU-M2000), fully isolated by VLAN
- One Dell R6525 server
- V-IPU admin & server software (preconfigured)
- Local NVME storage (4 TB); this will be shared by all users accessing your IPU-POD$_{16}$
- Read-only access to a shared drive with code examples and public data sets (see Section 2.3, Software and datasets)
- No root access or Docker support

The vPOD is provisioned as a single Virtual-IPU partition. A Virtual-IPU partition represents some number of IPUs which can communicate with one another. They are isolated so that all communication from physically neighbouring devices that are not in the same partition is prohibited.

### 2.2.1 Local storage

You have 4 TB of high-speed storage at `/localdata`. You should run tests and store data there. This is much faster than the disk used for `/mnt/public` (see below), which is read-only.

For convenience, you can create a directory in `/localdata` and create a symbolic link to this in your home directory:

```
$ mkdir /localdata/username/workspace
$ ln -s /localdata/username/workspace ~/workspace
```

## 2.3 Software and datasets

When you sign into your vPOD instance, there are several software packages available including:

- C++ compilers
- Python (versions 2.7 and 3.6)
- Graphcore Poplar SDK, including TensorFlow and PyTorch for the IPU, and documentation in `/opt/gc`. See Section 3, Software setup for information on how to configure the SDK before use.
- Datasets for use with the Graphcore examples (see Section 2.4.2, Installing the Graphcore examples) in `/localdata/datasets`. There are README files with more information on using each of these datasets. The same datasets are also on the shared drive `/mnt/public/data`, which can be used to restore the data if necessary.

**Licensing of datasets**

Some datasets are provided locally for ease of use. These are made available for **non-commercial use only** and should not be downloaded from Graphcloud. Please take time to read the full terms & conditions of use and license information which you will find alongside the datasets.

You can also download your own data, within the available disk space. Alternatively, you can use synthetic data to test model performance without the overhead of data transfer. See the relevant software framework documentation for enabling synthetic data on the software documentation page.

## 2.4 Making efficient use of local storage

You should find that there is a directory for each user under `/localdata`. For example, `/localdata/alice` or `/localdata/bob`. This provides fast access to a large amount of storage. By default, this personal workspace is private to each user.

### 2.4.1 Shared workspace

You may also want to create a shared area that can be used by other members of your team. You can do this with the following sequence of commands:

```
$ mkdir /localdata/shared
$ chown :ipupod /localdata/shared
$ chmod 2770 /localdata/shared
```

The first command creates the directory, the second sets the group ownership to `ipupod`, the group that all users are in, and finally the permissions are changed so that all members of the group can read and write to the directory, and ensures that all files created in the directory have the same permissions (the `setguid` bit).

You can create project directories under this directory, that can be accessed by all users in the group:

```
$ mkdir /localdata/shared/test_project
```

The pre-installed public datasets are available for all users under `/localdata/datasets`.

### 2.4.2 Installing the Graphcore examples

There are several example programs and tutorials available in the Graphcore GitHub repositories examples and tutorials. These are listed in the document index. You can clone these repositories to the shared area or to your private workspace. For example:

```
$ cd /localdata/shared
$ git clone https://github.com/graphcore/examples.git
Cloning into 'examples'...
$ git clone https://github.com/graphcore/tutorials.git
Cloning into 'tutorials'...
```

This will install all the examples and benchmarks to the fast disk storage under `/localdata/shared/examples`. These files will be accessible to everyone in the `ipupod` user group.

### 2.4.3 Temporary storage

Each user has a `tmp` subdirectory (for example, `/localdata/alice/tmp`) which is configured (in the ~/.profile file) to be used as temporary storage. This provides fast storage and avoids the more limited system temp directory filling up.

CHAPTER

# THREE

# SOFTWARE SETUP

The Poplar SDK is already installed on the Graphcloud server. You will need to do some initial configuration to be able to use it, as described below.

## 3.1 Setting up the SDK environment

To use the Graphcore tools and Poplar libraries, several environment variables (such as library and binary paths) need to be set up, as shown below:

```
$ cd /opt/gc/poplar_sdk-ubuntu_18_04-[ver]
$ source poplar-ubuntu_18_04-[ver]/enable.sh
$ source popart-ubuntu_18_04-[ver]/enable.sh
```

Where `[ver]` is the current software version number of each package.

You will need to source both the Poplar and the PopART enable scripts if you are using PyTorch or PopART.

If you attempt to run any Poplar software without having first enabled these scripts, you'll 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
```

**Note:** These scripts must be sourced for each new Bash shell. You can add these commands to your `.bashrc` to do this.

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

```
$ popc --version
```

This will display the version number of the installed software.

No further set up is required to use PopART and Poplar. PopTorch and TensorFlow for the IPU are provided as Python wheel files that can be installed using `pip` as described in the following sections.

## 3.2 Setting up PyTorch for the IPU

PopTorch is part of the Poplar SDK. It provides functions to allow PyTorch models to run on the IPU.

Before running PopTorch, you must source the `enable.sh` scripts for Poplar and PopART as described in Section 3.1, Setting up the SDK environment.

PopTorch is packaged as a Python wheel file that can be installed using `pip`.

**Note:** PopTorch requires `pip` version 18.1 or later, so it important to make sure you have the latest version before installing PopTorch.

We recommend creating a virtual environment, using `virtualenv`, to isolate your PopTorch environment from the system Python environment. You can create a virtual environment in a `workspace` directory and install PopTorch as shown below:

```
$ virtualenv -p python3 ~/workspace/poptorch_env
$ source poptorch_env/bin/activate
$ pip3 install -U pip
$ pip3 install poptorch-[ver].whl
```

Where `[ver]` is the SDK version version.

To confirm that PopTorch has been installed, you can use `pip list`, which should include the `poptorch` package in the output.

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__)"
```

For more information, refer to PyTorch for the IPU: User Guide.

## 3.3 Setting up TensorFlow for the IPU

Before running TensorFlow, you must source the `enable.sh` scripts for Poplar as described in Section 3.1, Setting up the SDK environment.

To use the Graphcore port of TensorFlow, you must set up a Python virtual environment.

You can create a virtual environment in a `workspace` directory and install TensorFlow as shown below:

```
$ virtualenv -p python3.6 ~/workspace/tensorflow_env
```

Then activate it.

```
$ source tensorflow_env/bin/activate
```

Now all installations will be local to that virtual environment.

We support TensorFlow 1 and TensorFlow 2. There are versions of these compiled for Intel and AMD processors to provide the best performance on those hosts. As a result, there are four Python wheel files that can be installed with `pip`.

**Warning:** You must install the correct wheel file for your host CPU. You can use the command `lscpu` to determine the CPU type, if you are not sure.

For example, to install Graphcore's TensorFlow distribution, compatible with v2.1.0 of TensorFlow, you would use a command similar to the following:

```
$ pip install tensorflow-2.1.0+[ver]+[arch].whl
```

Where `[ver]` is the TensorFlow version you have downloaded, and `[arch]` is the host CPU architecture (Intel or AMD).

To confirm that `tensorflow` has been installed, you can use `pip list`, which should include the `tensorflow` package in the output, for example:

```
(tensorflow_env) jsp$ pip list
Package        Version
-------------  ----------
future         0.18.2
numpy          1.19.5
pip            20.3.3
pkg-resources  0.0.0
tensorflow_env 2.1.0
setuptools     51.1.2
torch          1.6.0+cpu
wheel          0.36.2
```

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

```
$ python -c "from tensorflow.python import ipu"
```

For the next steps with TensorFlow, refer to the appropriate user guide:

- Targeting the IPU from TensorFlow 1.
- Targeting the IPU from TensorFlow 2.

## 3.4 Verifying the hardware configuration

Poplar needs a configuration file that describes how to connect to the IPU-M2000s that you have been allocated. This file is in the directory `.ipuof.config.d` in your home directory and is read-only.

This file contains information about:

- The number of IPUs assigned to you as a "vPOD"
- Networking details for how to connect to the vPOD

The file name is decided by the Graphcloud admin when setting up your account and can vary. There will only be one configuration file in this directory.

If this file is missing, then please contact your Graphcloud support team or use the resources on the Graphcore support portal https://www.graphcore.ai/support.

You can use the IPU command line tools to check what IPU hardware can be seen by the system. For example, `gc-info` will list information about the available IPU hardware made available through the use of this config file. For example:

```
$ gc-info -a
Graphcore device listing:

-+- Id:  [0], target:    [Fabric], PCI Domain: [3]
-+- Id:  [1], target:    [Fabric], PCI Domain: [2]
-+- Id:  [2], target:    [Fabric], PCI Domain: [1]
-+- Id:  [3], target:    [Fabric], PCI Domain: [0]
-+- Id:  [4], target:    [Fabric], PCI Domain: [3]
-+- Id:  [5], target:    [Fabric], PCI Domain: [2]
-+- Id:  [6], target:    [Fabric], PCI Domain: [1]
-+- Id:  [7], target:    [Fabric], PCI Domain: [0]
-+- Id:  [8], target:    [Fabric], PCI Domain: [3]
-+- Id:  [9], target:    [Fabric], PCI Domain: [2]
-+- Id: [10], target:    [Fabric], PCI Domain: [1]
-+- Id: [11], target:    [Fabric], PCI Domain: [0]
-+- Id: [12], target:    [Fabric], PCI Domain: [3]
-+- Id: [13], target:    [Fabric], PCI Domain: [2]
-+- Id: [14], target:    [Fabric], PCI Domain: [1]
-+- Id: [15], target:    [Fabric], PCI Domain: [0]
-+- Id: [16], target: [Multi IPU]
|--- PCIe Id:  [0], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [1], DNC Id: [1], PCI Domain: [2]
-+- Id: [17], target: [Multi IPU]
```

(continues on next page)

(continued from previous page)

```
|--- PCIe Id:  [2], DNC Id: [0], PCI Domain: [1]
|--- PCIe Id:  [3], DNC Id: [1], PCI Domain: [0]
-+- Id: [18], target: [Multi IPU]
|--- PCIe Id:  [4], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [5], DNC Id: [1], PCI Domain: [2]
-+- Id: [19], target: [Multi IPU]
|--- PCIe Id:  [6], DNC Id: [0], PCI Domain: [1]
|--- PCIe Id:  [7], DNC Id: [1], PCI Domain: [0]
-+- Id: [20], target: [Multi IPU]
|--- PCIe Id:  [8], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [9], DNC Id: [1], PCI Domain: [2]
-+- Id: [21], target: [Multi IPU]
|--- PCIe Id: [10], DNC Id: [0], PCI Domain: [1]
|--- PCIe Id: [11], DNC Id: [1], PCI Domain: [0]
-+- Id: [22], target: [Multi IPU]
|--- PCIe Id: [12], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id: [13], DNC Id: [1], PCI Domain: [2]
-+- Id: [23], target: [Multi IPU]
|--- PCIe Id: [14], DNC Id: [0], PCI Domain: [1]
|--- PCIe Id: [15], DNC Id: [1], PCI Domain: [0]
-+- Id: [24], target: [Multi IPU]
|--- PCIe Id:  [0], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [1], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id:  [2], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id:  [3], DNC Id: [3], PCI Domain: [0]
-+- Id: [25], target: [Multi IPU]
|--- PCIe Id:  [4], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [5], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id:  [6], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id:  [7], DNC Id: [3], PCI Domain: [0]
-+- Id: [26], target: [Multi IPU]
|--- PCIe Id:  [8], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [9], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id: [10], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id: [11], DNC Id: [3], PCI Domain: [0]
-+- Id: [27], target: [Multi IPU]
|--- PCIe Id: [12], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id: [13], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id: [14], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id: [15], DNC Id: [3], PCI Domain: [0]
-+- Id: [28], target: [Multi IPU]
|--- PCIe Id:  [0], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [1], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id:  [2], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id:  [3], DNC Id: [3], PCI Domain: [0]
|--- PCIe Id:  [4], DNC Id: [4], PCI Domain: [3]
|--- PCIe Id:  [5], DNC Id: [5], PCI Domain: [2]
|--- PCIe Id:  [6], DNC Id: [6], PCI Domain: [1]
|--- PCIe Id:  [7], DNC Id: [7], PCI Domain: [0]
-+- Id: [29], target: [Multi IPU]
|--- PCIe Id:  [8], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [9], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id: [10], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id: [11], DNC Id: [3], PCI Domain: [0]
|--- PCIe Id: [12], DNC Id: [4], PCI Domain: [3]
|--- PCIe Id: [13], DNC Id: [5], PCI Domain: [2]
|--- PCIe Id: [14], DNC Id: [6], PCI Domain: [1]
|--- PCIe Id: [15], DNC Id: [7], PCI Domain: [0]
-+- Id: [30], target: [Multi IPU]
|--- PCIe Id:  [0], DNC Id: [0], PCI Domain: [3]
|--- PCIe Id:  [1], DNC Id: [1], PCI Domain: [2]
|--- PCIe Id:  [2], DNC Id: [2], PCI Domain: [1]
|--- PCIe Id:  [3], DNC Id: [3], PCI Domain: [0]
|--- PCIe Id:  [4], DNC Id: [4], PCI Domain: [3]
|--- PCIe Id:  [5], DNC Id: [5], PCI Domain: [2]
|--- PCIe Id:  [6], DNC Id: [6], PCI Domain: [1]
|--- PCIe Id:  [7], DNC Id: [7], PCI Domain: [0]
|--- PCIe Id:  [8], DNC Id: [8], PCI Domain: [3]
|--- PCIe Id:  [9], DNC Id: [9], PCI Domain: [2]
|--- PCIe Id: [10], DNC Id: [10], PCI Domain: [1]
```

(continues on next page)

(continued from previous page)

```
|--- PCIe Id: [11], DNC Id: [11], PCI Domain: [0]
|--- PCIe Id: [12], DNC Id: [12], PCI Domain: [3]
|--- PCIe Id: [13], DNC Id: [13], PCI Domain: [2]
|--- PCIe Id: [14], DNC Id: [14], PCI Domain: [1]
|--- PCIe Id: [15], DNC Id: [15], PCI Domain: [0]
```

If this command does not bring up a list of devices or shows an error when trying to connect to the IPU resources, then you should contact your Graphcloud support team or use the resources on the Graphcore support portal https://www.graphcore.ai/support.

You can also run `gc-monitor` to view more IPU-related details. See the IPU Command Line Tools document for more details about other driver-level diagnostic tools available.

CHAPTER

FOUR

# RUNNING A PROGRAM ON THE IPU

You can now run one of the example programs from the SDK. The Poplar examples are in the directory `/opt/gc/poplar_sdk-ubuntu_18_04-[ver]/poplar-ubuntu_18_04-[ver]/examples`.

The program `adder_ipu.cpp` builds a simple graph to add two vectors together and return the result.

1. Make a copy of the `adder` example directory in your working directory
2. Compile the program

```
$ make
```

3. Run the program. It will produce the following output:

```
$ ./adder_ipu
Creating graph
Building engine
v1 sum = 10
v2 sum = 65
```

More resources for developing software to run on the IPU can be found on the Graphcore developer site and the Graphcore examples and tutorials repositories on GitHub.

## 4.1 Hardware and software monitoring

The IPU hardware, and any programs running on it, can be monitored and analysed in various ways.

### 4.1.1 Command line tools

The IPU driver software includes a number of software tools that provide information on the current status of the connected hardware. These include:

- `gc-info`: determine what IPU devices are present in the system.
- `gc-monitor`: passively monitor IPU activity and telemetry such as power draw, temperature and clock speed.
- `gc-reset`: reset one or more IPU devices.

These are in the directory `/opt/gc/poplar_sdk-ubuntu_18_04-[ver]/poplar-[os]-[ver]/bin`.

The use of these tools is described in the IPU Command Line Tools document.

### 4.1.2 Trace output

When running a program, you have the option to output a trace from the running code which allows you to see the phases that the graph compiler uses when preparing and compiling the graph processes to execute on the IPU. Going back to the simple adder example in *Running a program on the IPU*, try running:

```
$ POPLAR_LOG_LEVEL=DEBUG ./adder_ipu
```

You will see each stage of the process listed before the program output appears. The logging options are documented in the Poplar & PopLibs User Guide.

### 4.1.3 Execution profiling

The PopVision analysis tools are available on the Graphcore software download portal.

The PopVision Graph Analyser provides a graphical view of the graph execution trace, showing memory use, tile mapping and other vital information to help you optimise your application.

The PopVision System Analyser provides information about the behaviour of the host-side application code. It shows an interactive graphical view of the timeline of execution steps, helping you to identify any bottlenecks between the CPUs and IPUs.

For more information see the PopVision User Guide.

CHAPTER

FIVE

# TROUBLESHOOTING

If you encounter an issue where the IPU-M2000s get into an unusable state and using the `gc-reset` tool does not resolve it then you can reset the Virtual-IPU partition using the Virtual-IPU command line interface.

Resetting the partition will:

- ensure that the IPUoF server is running on the IPU-M2000s
- configure the IPUs in the partition and the IPU links between them

To reset the partition:

1. Identify the partition name (this is the first field of the output)

```
$ vipu -H localhost list partition
```

2. Reset the partition

```
$ vipu -H localhost reset partition <partition_name>
```

3. Check that the partition is ready for use

   Resetting the partition can take a few minutes.

```
$ vipu -H localhost list partition
```

   The partition is ready for use when the state is `ACTIVE`.

For more information about Virtual-IPU, see the V-IPU User Guide.

CHAPTER

SIX

# NEXT STEPS

Full documentation for the Poplar software is available on the Graphcore documentation site: https://docs.graphcore.ai/. More information can be found on the Graphcore developer pages: https://www.graphcore.ai/developer.

For an overview of the Poplar SDK and development tools, see the SDK Overview.

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

If you are interested in running TensorFlow on the IPU, there are user guides and API references for the IPU implementation of TensorFlow 1 and TensorFlow 2.

You can also run PyTorch on the IPU.

Graphcore also has GitHub repositories with further examples:

- https://github.com/graphcore/examples:
  - TensorFlow, PyTorch and PopART versions of commonly used machine learning models for training and inference, including CNNs such as ResNet, ResNeXt and EfficientNet
  - Training data for the above models
- https://github.com/graphcore/tutorials:
  - Tutorials
  - Examples of using Poplar and IPU features
  - Examples of simple models
  - Source code from videos, blogs and other documents
  - Benchmarks for performance testing of layer types on your IPU system

You can use the tags "ipu", "poplar" and "popart" when asking questions or looking for answers on StackOverflow.

Support is available from the Graphcore customer engineering team via the Graphcore support portal.

CHAPTER

SEVEN

# TRADEMARKS & COPYRIGHT