GRAPHCORE

Paperspace: Getting Started with IPUs

Version latest

Graphcore Ltd

Mar 05, 2024

CONTENTS

This guide describes how to get up and running with Graphcore IPUs on Paperspace. We show you how to train your first model, using one of our Hugging Face Optimum vision transformer (ViT) notebooks as an example. We also guide you through other features of Paperspace Gradient Notebooks so you can run more tutorials and start building your own models for the IPU.

CHAPTER

ONE

OVERVIEW

Paperspace now offers access to Graphcore Intelligence Processing Units (IPUs) via Gradient Notebooks, a web-based Jupyter IDE.

The IPU is a completely new kind of massively parallel processor specifically designed for AI and machine learning applications. Each IPU has 1,472 processor cores, running nearly 9,000 independent parallel program threads. For example, the IPU-POD_{16} system available on Paperspace gives you access to 4 petaFLOPS of AI compute.

The advantage of accessing IPUs on Paperspace is that the environment to run the notebook is ready-to-use — you don't have to set anything up.

More information is available on the Graphcore blog Getting Started with IPUs on Paperspace and the Paperspace blog Up and Running with Graphcore IPUs on Paperspace.

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.

Watch the videos on the Paperspace YouTube channel to learn more about Paperspace and Gradient.

CHAPTER

TWO

PREREQUISITES

In order to use IPUs on Paperspace, you must be familiar with developing machine-learning applications using industry-standard frameworks such as PyTorch or TensorFlow. You must also have a knowledge of Python and Jupyter notebooks.

The Graphcore Developer page has links to a variety of resources to help you get started with programming for the IPU.

2.1 Paperspace terminal

If you want to use the terminal in Paperspace then it can be beneficial if you have some familiarity with the Linux command line and some common Linux tools. Some of the tasks that you may need to perform are:

- 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 (for example, setting paths and performing actions that need to be done on every login).
- 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 with `ssh`, configuring `ssh` and using `scp` to copy files to and from the vPod.
- 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 such as conflicting versions and dependencies.

CHAPTER

THREE

ACCESS IPUS ON PAPERSPACE

This section describes how to access Paperspace.

Note: You can also access IPUs on Paperspace from VS Code with the setup described in the notebook Using VS Code in Paperspace Notebooks.

3.1 Sign into Paperspace

Before you can start using IPUs on Paperspace, you must be signed into a Paperspace account.

If you don't already have an account, you can create one on the sign in page. You can choose to sign up with an email address or with your GitHub or Google account.

3.2 Create a project

After you have created your account, you will need to create a project.

To create a project:

1. In the drop-down on the top-left, make sure the *Gradient ML Platform* is selected. You are now in the Gradient console.
2. Click on *CREATE A PROJECT*.
3. Change the name of the project, if you wish.
4. Click on *CREATE*.

Now you are ready to use IPUs to run a Jupyter notebook on Paperspace Gradient (Section 4, Quick start for Gradient Notebooks.

CHAPTER

FOUR

QUICK START FOR GRADIENT NOTEBOOKS

There are three routes to use IPUs to run a Jupyter notebook on Paperspace Gradient:

1. *From the Graphcore Model Garden*
2. *From the list of IPU-powered notebooks*
3. *From the setup procedure in this guide*

Note: You will be required to *sign into Paperspace* before you can run any notebooks.

You can also access IPUs on Paperspace from VS Code with the setup described in the notebook Using VS Code in Paperspace Notebooks.

4.1 From the Graphcore Model Garden

1. Go to the Graphcore Model Garden.
2. Filter by "Paperspace".
3. Click on *Try on Paperspace* for the notebook you want to run.
4. The notebook opens in Paperspace Gradient.
5. For example to run the notebook described in Section 6.2, Running the ViT model on a medical image dataset, filter by "Paperspace", "PyTorch" and "Computer Vision" and click on *Try on Paperspace*:

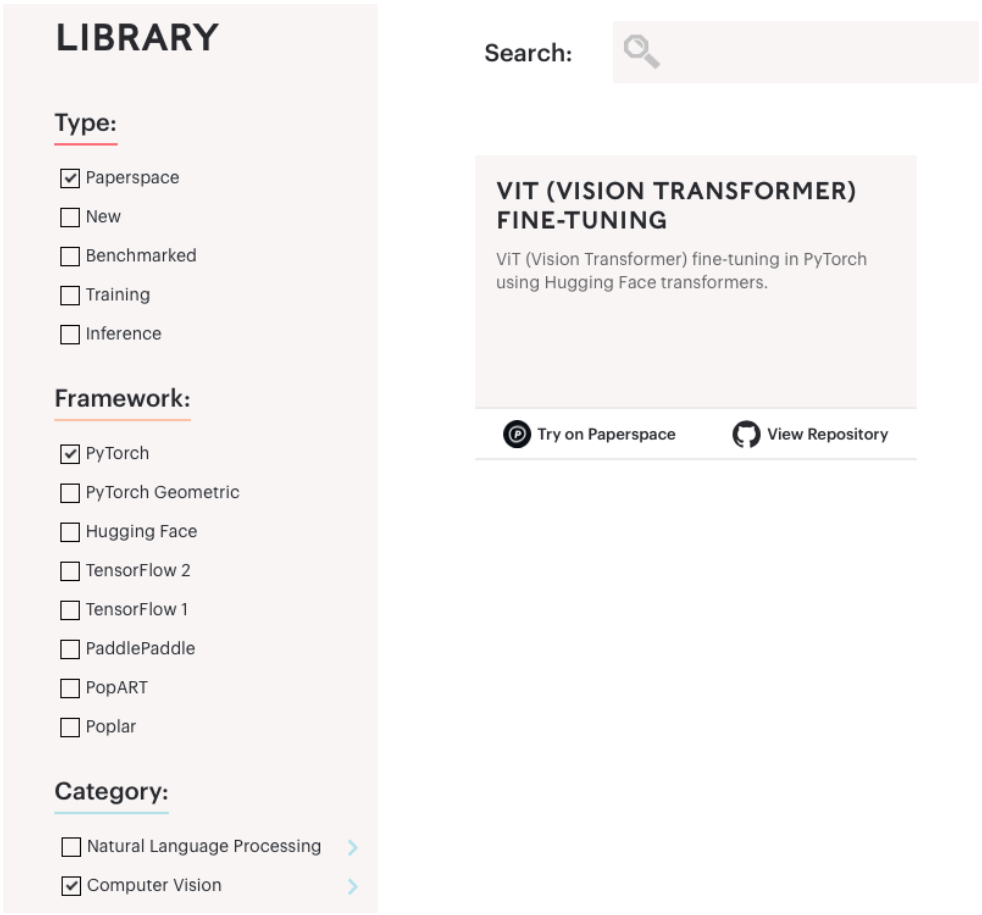

Fig. 4.1: Filter and click on *Try on Paperspace* from the Model Garden

4.2 From the list of IPU-powered notebooks

1. Go to the list of IPU-powered Jupyter Notebooks.
2. Click on *Run on Gradient* for the notebook you want to run.
3. The notebook opens in Paperspace Gradient.
4. For example to run the notebook described in Section 6.2, Running the ViT model on a medical image dataset, find the notebook and click on *Run on Gradient*:

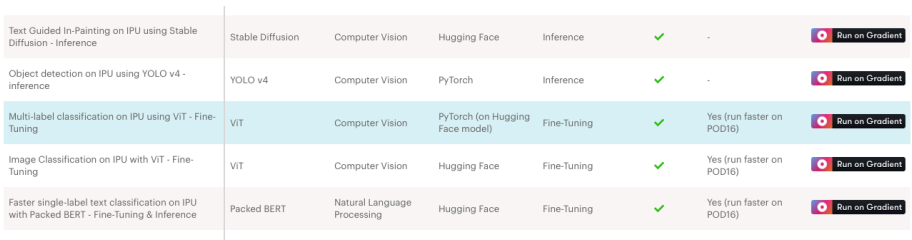

Fig. 4.2: Find and click on *Run on Gradient* from the list of IPU-powered notebooks

4.3 From the setup procedure in this guide

Follow the procedure described in Section 5, Create a Gradient Notebook and try it out on the example notebook described in Section 6, Run a Jupyter notebook in a Gradient Notebook.

CHAPTER

FIVE

CREATE A GRADIENT NOTEBOOK

In this section, we will learn how to create a Gradient Notebook by using a pre-built, IPU-optimised runtime. In Section 6, Run a Jupyter notebook in a Gradient Notebook we take you through selecting and running a Jupyter notebook in a Gradient Notebook created with the *PyTorch on IPU runtime*.

5.1 Start Gradient Notebook creation

There are two routes to start creating a notebook:

1. In the project card in the Gradient console, click on *CREATE NOTEBOOK* (Fig. 5.1).

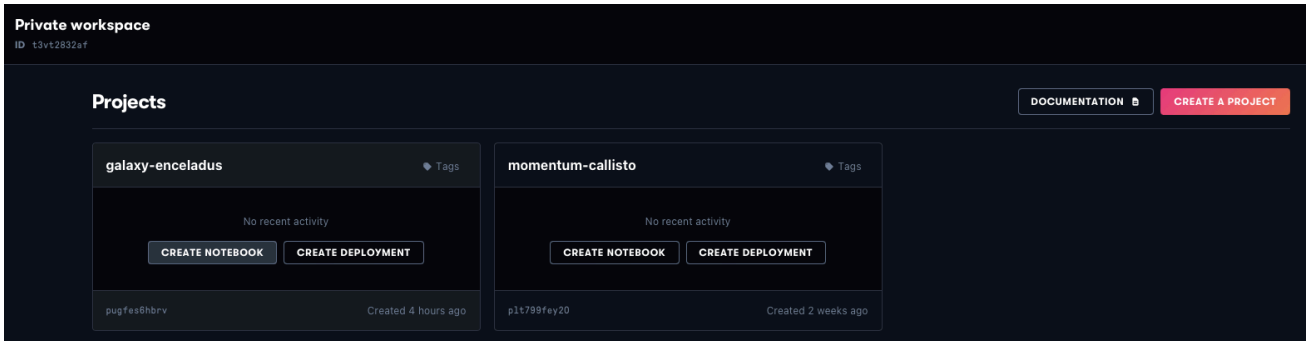

Fig. 5.1: Create a Gradient Notebook from the Gradient console

2. Open the project and click on *CREATE* (Fig. 5.2).

Fig. 5.2: Create a Gradient Notebook from a project

5.2 Select an IPU runtime

In the *Select a runtime* section, you can select one of four IPU-optimised runtimes. These are:

- Hugging Face Transformers on IPU
- PyTorch on IPU
- TensorFlow 2 on IPU
- PyTorch Geometric on IPU

Starting the runtime loads the pre-configured Docker container and a GitHub repository of a curated list of Jupyter notebooks. The runtime basically takes care of everything under the hood so you can jump straight to running the code.

For more information about runtimes in general, refer to the Paperspace documentation on runtimes.

Note: You may have to click on *SHOW MORE* under the list (Fig. 5.3) to access all the IPU-optimised runtimes.

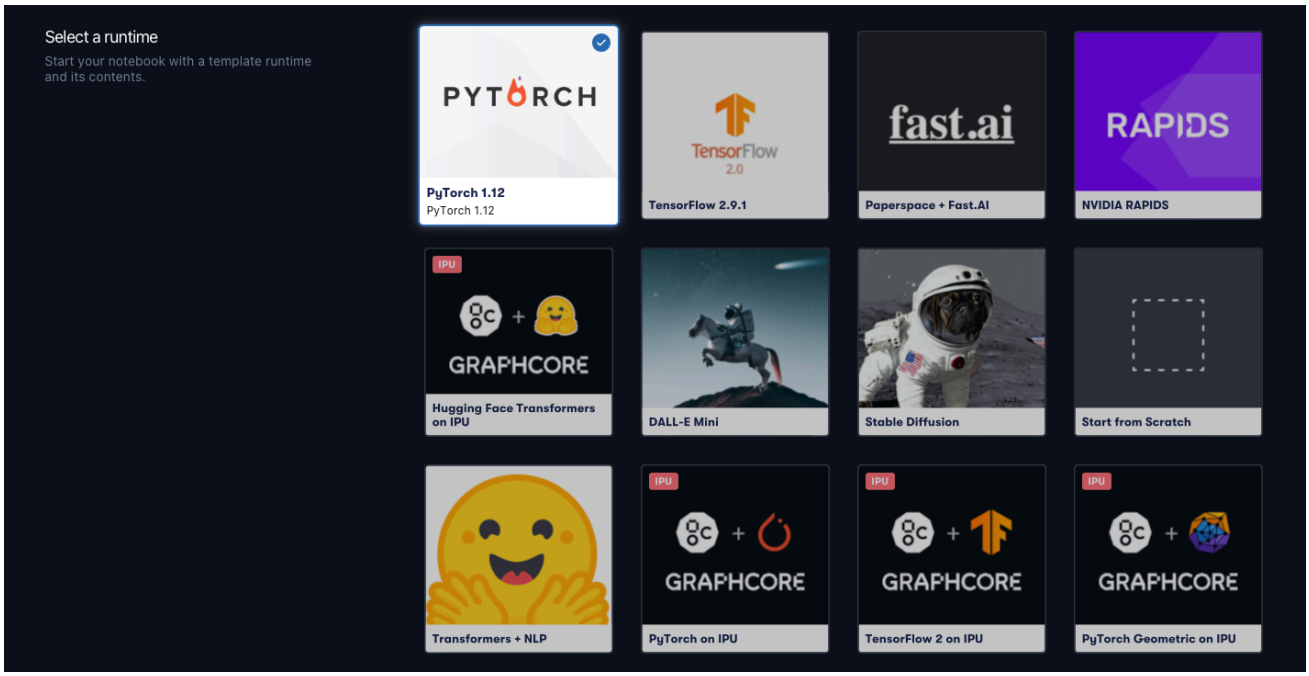

Fig. 5.3: Other IPU-optimised runtimes

5.3 Select a machine

Once you have selected an IPU-optimised runtime, you can select a machine containing IPUs. For example, you can access a free IPU-POD$_4$ or pay and access Pod systems with more resources such as an IPU-POD$_{16}$, a Bow Pod$_{16}$ or a Bow Pod$_{64}$.

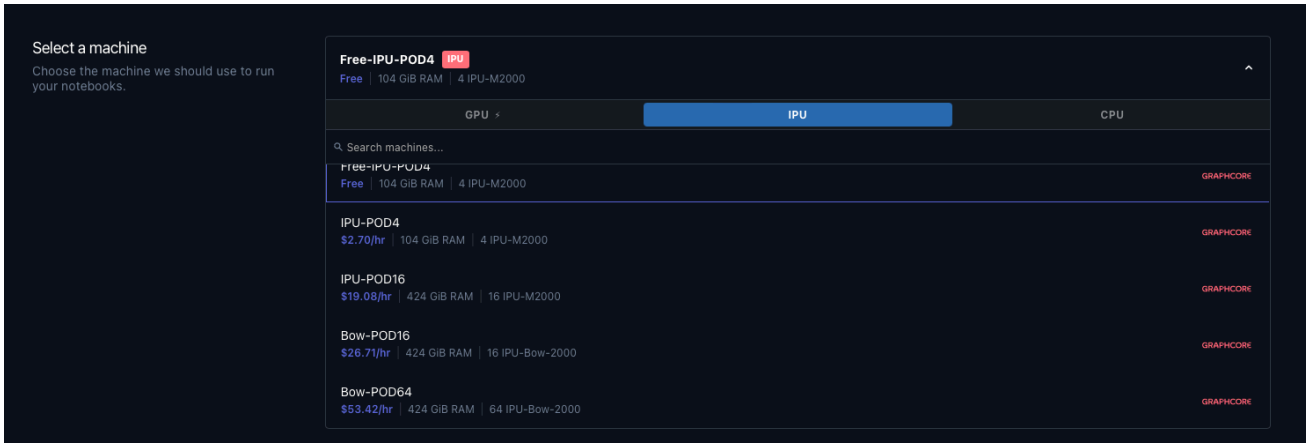

Fig. 5.4: Examples of available machines

5.4 Select auto-shutdown timeout

The free machines auto-shutdown after a maximum of 6 hours. For more information, refer to the Paperspace documentation on machine autoshutdown.

5.5 Advanced options (optional)

You can configure the source repository and Graphcore Docker container in your Gradient Notebook. In general, you will keep the settings defined in your selected IPU runtime.

Select *Advanced options* to see the pre-loaded configuration associated with a Gradient runtime and modify it for your own use.

Looking at the Hugging Face Transformers on IPU runtime for example, you can see from Fig. 5.5 that it is configured to load the gradient-ai/Graphcore-Huggingface repo and run it with a PyTorch Docker container from the Graphcore Docker Hub containing all the necessary libraries and the Poplar SDK.

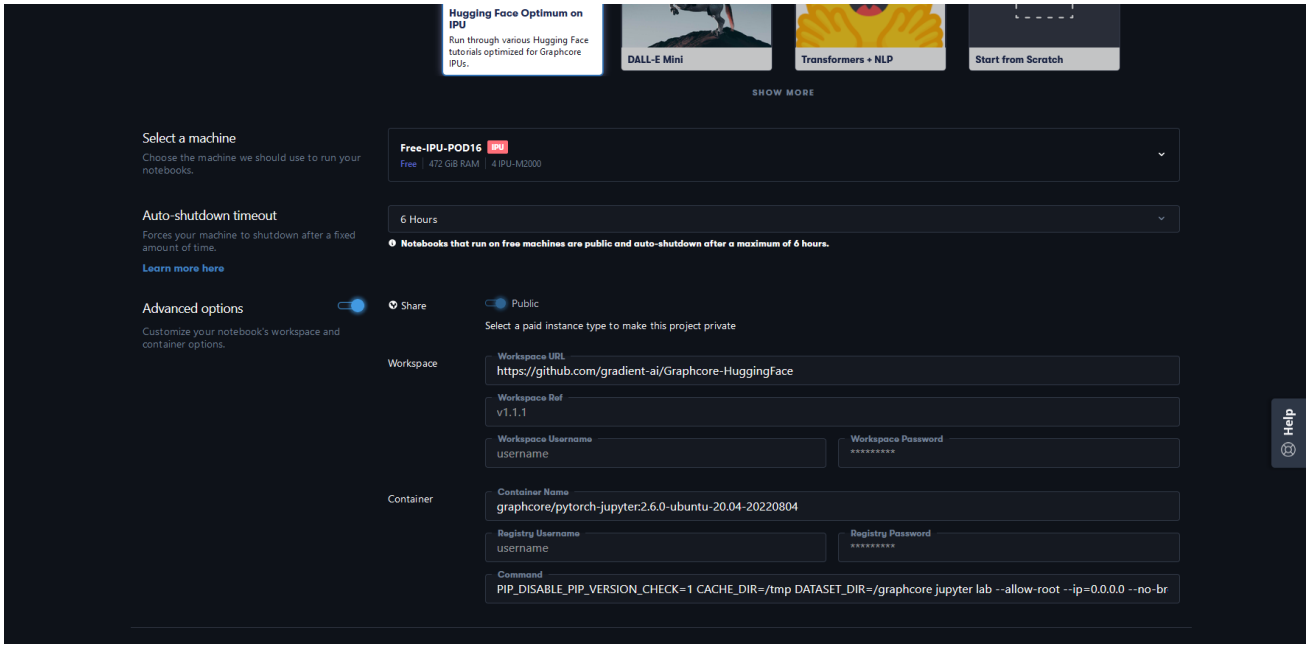

Fig. 5.5: Hugging Face Transformers on IPU advanced options

You can easily modify the runtime's advanced options in order to load your own repository, and run it in a different Docker image. Fig. 5.6 shows how you can change the repository field to use the Graphcore application examples repo and select a TensorFlow 2 image from Graphcore's official Docker Hub releases.

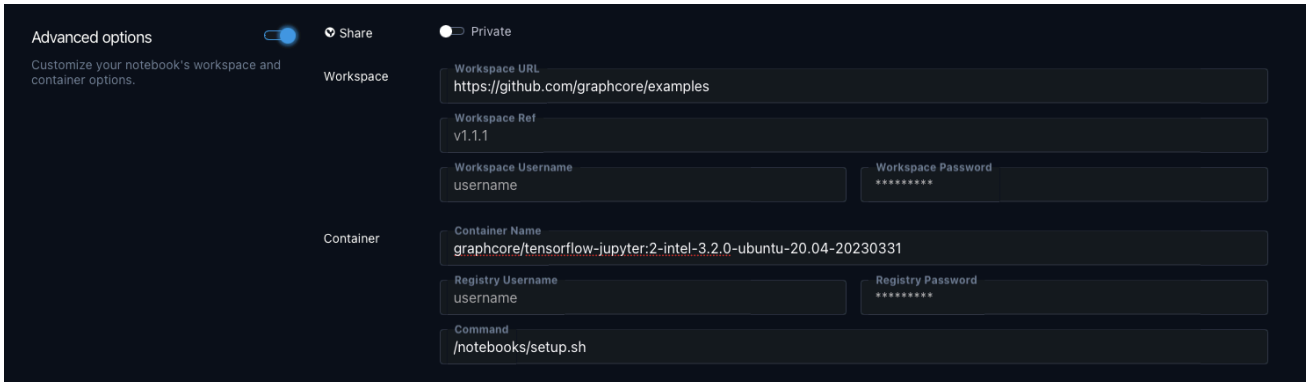

Fig. 5.6: Selecting Graphcore examples repository

This will let you run any TensorFlow 2 code from the Graphcore application examples repo in this Gradient Notebook.

Note: When setting advanced options manually, you need to match the IPU host server processor type (in this case, AMD) to the correct Docker tags (tf2-amd).

5.6 Start notebook

Click on *START NOTEBOOK* to complete the Gradient Notebook creation process.

Note: Depending on the machine you have selected, this process can take up to 25 minutes.

Once your Gradient Notebook has been created, you will be able to run a Jupyter notebook, as described in Section 6, Run a Jupyter notebook in a Gradient Notebook.

CHAPTER

SIX

RUN A JUPYTER NOTEBOOK IN A GRADIENT NOTEBOOK

In this section we create a Gradient Notebook with the **PyTorch on IPU** runtime and run the fine-tuning on a ViT Hugging Face model.

6.1 Setup

We create a Gradient notebook with the PyTorch on IPU runtime.

6.1.1 About the runtime

The *PyTorch on IPU* runtime uses a PyTorch-Jupyter container from the Graphcore Docker Hub, which ensures that the Poplar SDK and PyTorch for IPU libraries are pre-installed and compatible with the IPU system in Paperspace. This runtime also loads a curated repo of Jupyter notebooks, and takes care of various environment and data configurations to optimise ease of use, so you can focus on learning about IPUs and model development.

6.1.2 Create the Gradient Notebook

Follow the instructions in Section 5, Create a Gradient Notebook and select *PyTorch on IPU* in the *Select a runtime* section.

It will take a few minutes to spin up the virtual machine. Once it's ready, navigate to the vit-model-training folder (shown on the left pane in Fig. 6.1) and open the ViT fine-tuning notebook (walkthrough.ipynb).

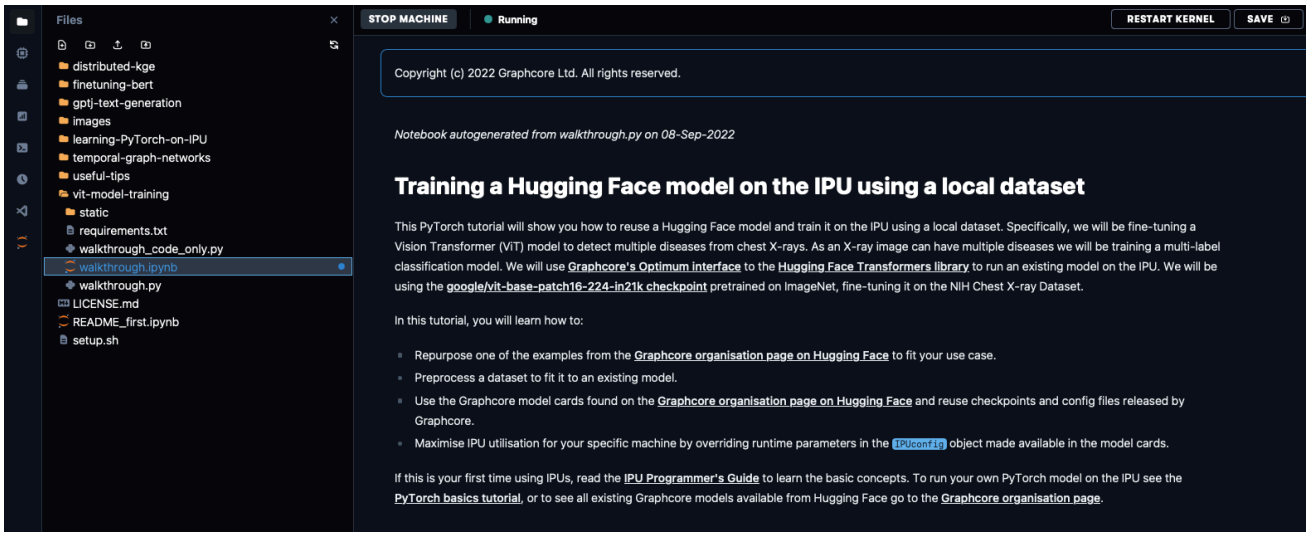

Fig. 6.1: ViT fine-tuning notebook

6.2 Running the ViT model on a medical image dataset

This is a good example of the type of workflow you'll commonly encounter when developing your own models using IPUs and Hugging Face Optimum. For a bit more context about this specific ViT implementation (and ViT models in general), check out this deep dive on running ViT in Hugging Face Optimum Graphcore.

You can follow the notebook step-by-step as it takes you through how to pre-process a large, widely-used dataset of chest x-ray images and how to run a ViT model in Hugging Face Optimum on this dataset. We'll cover some highlights of what you can do with this notebook. Full details are given in the notebook.

6.2.1 Training the model

To train the ViT model on the IPU we need to use the IPUTrainer class, which takes the same arguments as the original Hugging Face Transformer Trainer class, in tandem with the IPUConfig object which specifies the behaviour for compilation and execution on the IPU. Running through the notebook will include downloading IPUConfig which is made available through Graphcore/vit-base-ipu, using the pre-trained ViT checkpoints model card found in google/vit-base-patch16-224-in21k, and fine-tuning it using the chest x-ray dataset. This lets you run your training on IPUs using optimised runtime configurations.

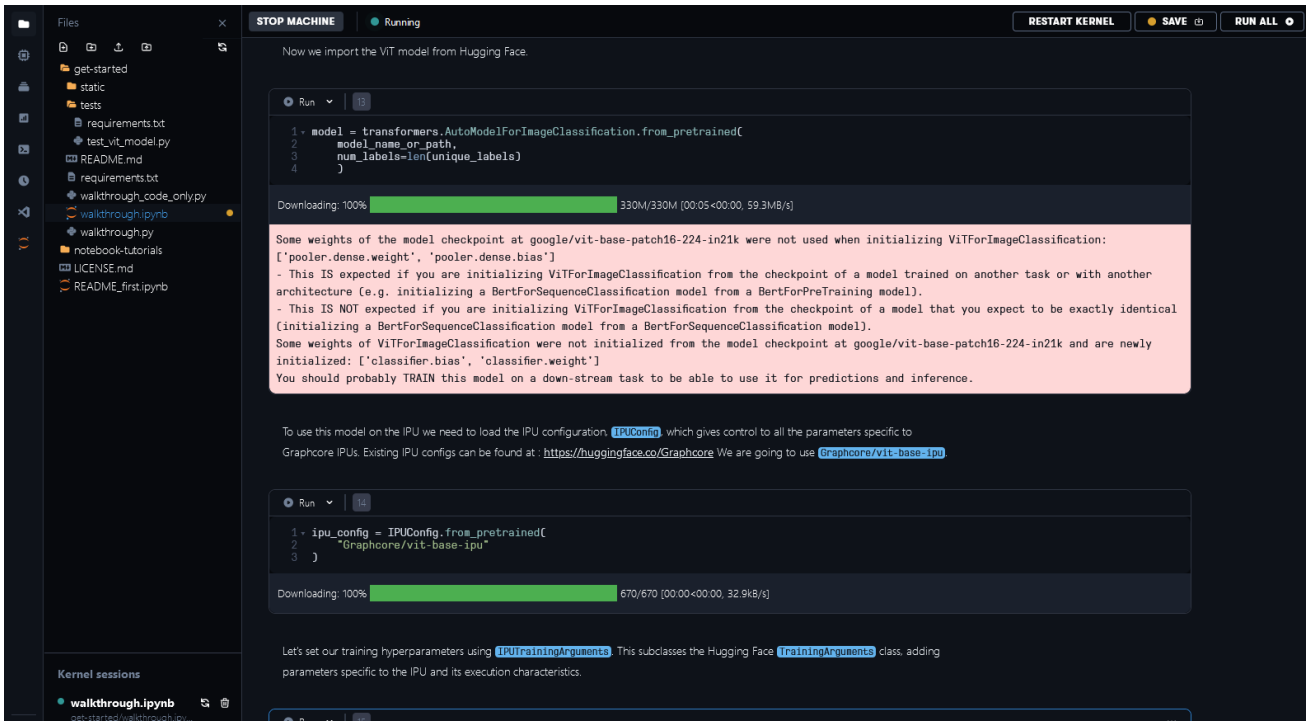

Fig. 6.2: Using weights and IPUConfig files from Hugging Face Hub

The PyTorch source code is compiled into a graph program which dictates how the program can be executed on the IPU hardware. The time to perform this compilation step depends on the model complexity.

In order to save time, a pre-compiled execution graph can be loaded so the model doesn't need to be recompiled. You can see the documentation on pre-compilation and caching to find out how it works.

We've made executable files available for some of the configurations in our notebook examples in Paperspace. This step can take a few minutes without the pre-compilation files or when it's necessary to recompile because of changes in the model.

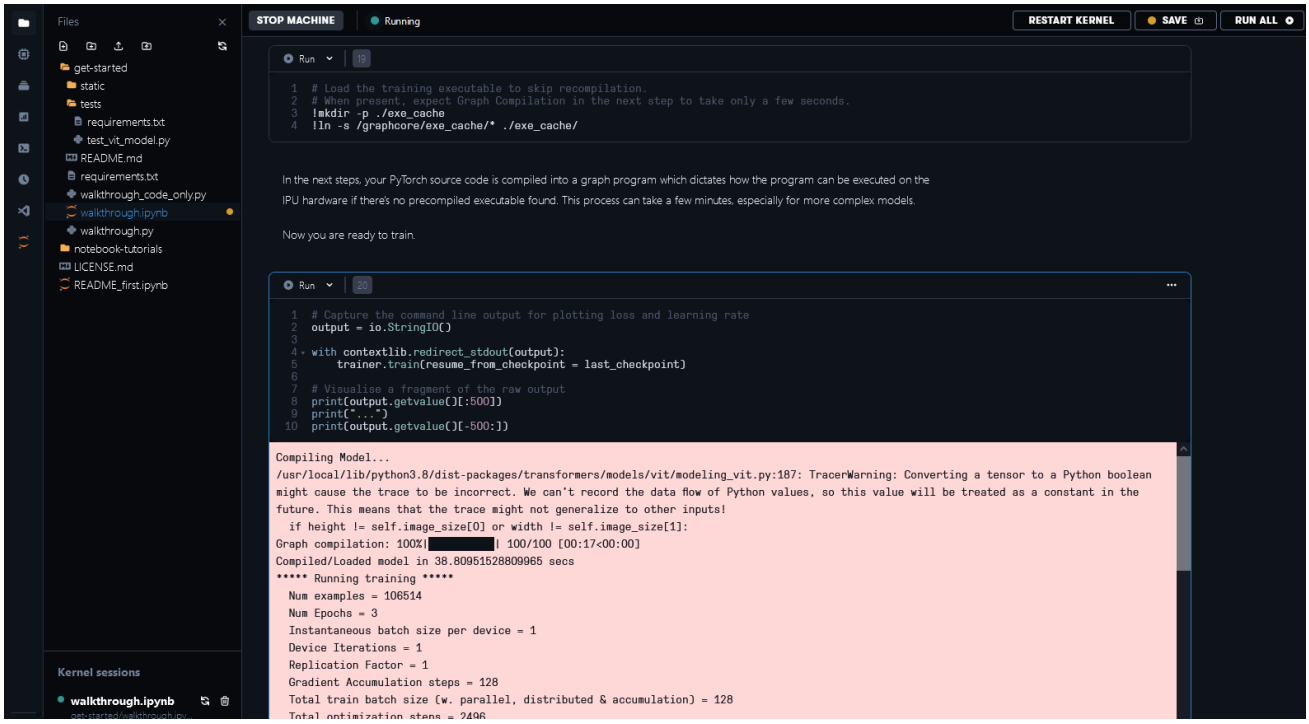

Fig. 6.3: Loading cached executable and model training

We have implemented a custom metric for the area under the ROC (receiver operating characteristic) curve (AUC_ROC). It is a commonly used performance metric for multi-label classification tasks because it is insensitive to class imbalance and easy to interpret. After completing training and evaluation, you can see in Fig. 6.4 the validation AUC_ROC score across 3 epochs to be 0.7811 for this 14-category multi-label classification task.

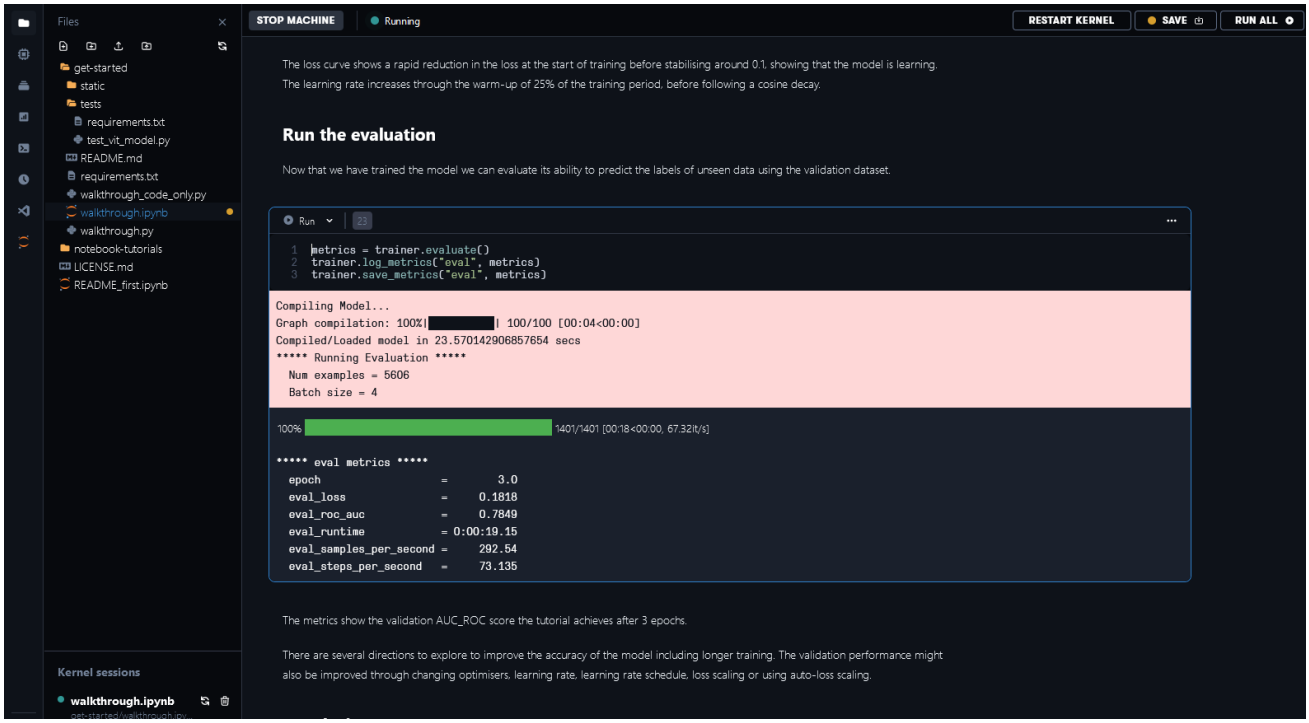

Fig. 6.4: Evaluation metrics

CHAPTER

SEVEN

GRADIENT FEATURES

This section highlights a few features of Gradient. For more information, refer to the Gradient documentation on Paperspace.

7.1 About Gradient

Paperspace Gradient is a supercharged MLOps platform that lets you go through the model development and deployment lifecycle faster. One of its most important features is its data store. If you are running code that requires generating or downloading large amounts of data, we recommend that you store it in the $/\mathrm{tmp}$ folder so as not to fill your local data storage limits.

7.2 About projects

The project in Paperspace lets you access various MLOps features, and most importantly lets you spin up a Gradient Notebook. The IDE handles provisioning a VM and setting up of the Docker container, the Poplar SDK, and code repository.

7.3 Storage

Running an IPU in Gradient gives you an initial 10 GB of data storage in the free tier (5 GB more than other runtimes), with the option to get more free storage. Additionally, a range of other storage options are provided for users on Gradient's paid plans. If you need permanent and shareable storage, you can use high performance data storage through Paperspace Datasets, which lets you keep and access data across all the runtimes in your project.

7.4 Advanced CLI jobs

While Gradient focuses on providing a notebook-style interface, it is a complete IDE with access to the terminal. This powerful feature lets you run the rest of the scripts that are available in Graphcore's examples repository.

Click on the terminal icon on the left (Fig. 7.1) to launch a shell interface. From there, you can replicate one of Graphcore's published examples or run the tutorials.

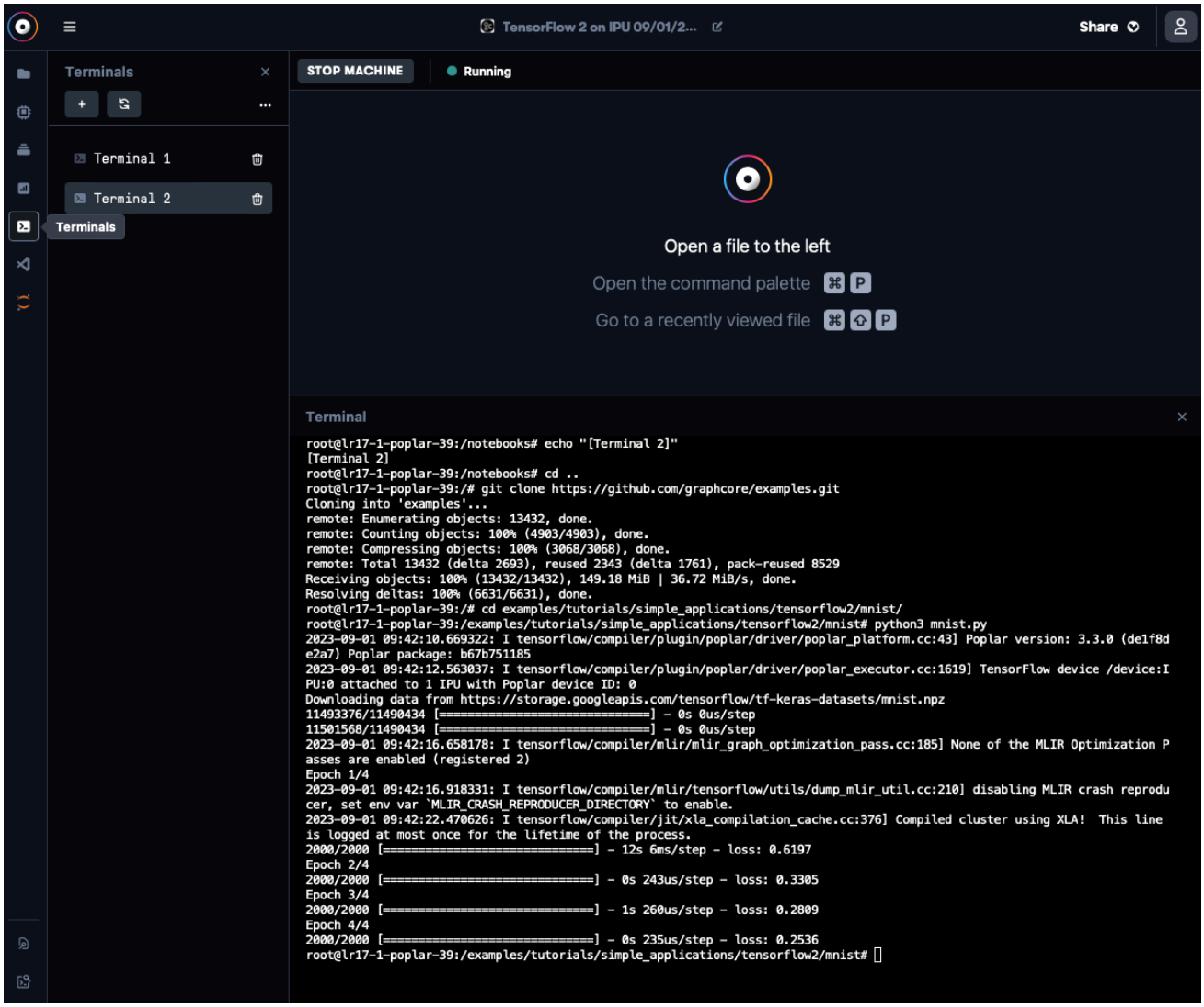

Fig. 7.1: Shell interface

CHAPTER

EIGHT

NEXT STEPS

There are plenty of resources to help you progress with developing on IPUs.

8.1 Support

Support is included with your access. You can request support by clicking on the **Submit a ticket** link on the Graphcore Support portal.

For general help, discussions and announcements, please join our Graphcore Slack Community.

When looking for answers or asking questions on StackOverflow, use the tag "ipu".

Note: If you want to use IPUs to test your own models or in a production environment, please contact us using the form on the Graphcore Contact page and we will help you to get set up.

8.2 Graphcore tutorials

There are Graphcore tutorials available in the Graphcore GitHub Examples repository. You will have to modify the advanced options for the the Gradient Notebook you create to specify the *Workspace URL* as "https://github.com/graphcore/examples/" and enter a suitable container from the Graphcore Docker hub into *Container Name* as described in Section 5.5, Advanced options (optional). Some tutorials that you might find interesting to run are:

- Porting a simple PyTorch model to the IPU
- PyTorch Geometric on IPUs at a glance
- Using half and mixed precision in PyTorch

8.3 Further reading

Graphcore's documentation website contains useful documents for learning how to work with IPUs. Some examples are:

- The IPU Programmer's Guide
- Switching from GPUs to IPUs for Machine Learning Models
- PyTorch for the IPU: User Guide

CHAPTER

NINE

TRADEMARKS & COPYRIGHT