EPS Machine Learning Server

# General information

The Aston EPS Machine Learning Server has the following specifications:

* 10x [NVIDIA RTX A6000 48GB GPUs](https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/proviz-print-nvidia-rtx-a6000-datasheet-us-nvidia-1454980-r9-web%20(1).pdf) (each: 10,752 CUDA cores, 336 3rd-gen Tensor cores, 84 2nd-gen RT cores)
* 2x Intel Xeon Gold 5218R - 2.1GHz - 20 cores/40 threads - 27.5MB cache.
* 768GB (12x64GB) 2933MHz DDR4 RAM.
* 28TB storage (7x 4TB Intel DC4510 NVME)

The server mission, as stated in the grant proposal that funded this machine, is:

***Access will be prioritised towards proof of principle work for future proposals, funded projects and new collaborations across different research groups at Aston. In all cases, early career academics will be favoured.***

*MSc students can use the server* ***if*** *their work cannot be done using other free alternatives (e.g., Google Colab, Kaggle notebooks).*

# Steering group

Server management:

* Dr Luis Manso ([l.manso@aston.ac.uk](mailto:l.manso@aston.ac.uk))
* Dr Martin Rudorfer ([m.rudorfer@aston.ac.uk](mailto:m.rudorfer@aston.ac.uk))

Additional support:

* Kieran Saunders ([190229315@aston.ac.uk](mailto:190229315@aston.ac.uk))

# How to start

1. Join the Slack workspace using the following link: <https://join.slack.com/t/astoncs-gpuserver/shared_invite/zt-2mw1wopht-y~9VJEg819m2tEJSDn6Ydg>. If the link doesn’t work, ask Luis Manso to invite you. Everyone accessing the server must be part of that channel, which is the official communication venue for all discussions related to requesting and booking resources, as well as using the machine.
2. **On Slack**, ask Luis Manso to register your username on the server.
3. If you don’t have Unix attributes (MSc students), **you will need to ask your supervisor** to request Unix attributes for you via Solve.
4. Create an account in the booking system see below (“*Registering into the resource booking system*”). To activate your user, you will have to ask anyone in the admin team to activate your account in the booking system.
5. After this, you are all set up. You can book resources and use the server (see **Booking Resources** below).
   1. Every month you will be able to request a budget for the following month (see **Requesting Resources** below).

# Available bookable resources

These resources are divided into **bookable groups**, as follows:

* Bookable GPU groups:
  + GPU Group 1: GPUs {0, 1} + 2 cores / 4 threads
  + GPU Group 2: GPUs {2, 3} + 2 cores / 4 threads
  + GPU Group 3: GPUs {4, 5} + 2 cores / 4 threads
  + GPU Group 4: GPUs {6, 7} + 2 cores / 4 threads
  + GPU Group 5: GPUs {8, 9} + 2 cores / 4 threads
* Bookable CPU groups
  + CPU Group 1: 5 cores/ 10 threads
  + CPU Group 2: 5 cores/ 10 threads

Storage and RAM are distributed as follows:

* **RAM**: For each GPU or CPU group booked, it's assumed that no more than 96GB RAM will be used (192 GB if 2 groups are booked together, 288 GB if 3 groups, etc.).
* **Storage**: Each user can store up to 1TB of data to be used between sessions (anything above this limit needs to be explicitly requested and approved by the steering group).
  + **It's considered good practice to delete all data that's no longer being used**.
  + If the total available storage space in the server goes below 1TB, users will get a warning asking them to delete some data. Any data above 1TB (unless explicitly approved) may be deleted after 1 week.
  + **IMPORTANT: users that are not using the server for more than 3 consecutive months may get any residual data deleted without warning**.

Note that the bookable groups and resource allocation are designed to always leave the server with some spare RAM and processing power to be able to function properly. Usage control is done via a mix of honour-code (you pledge not to use more than was booked), peer monitoring (if you detect someone overusing, please flag to [l.manso@aston.ac.uk](mailto:l.manso@aston.ac.uk)), and a monitoring program that will report unbooked usage via Slack. **The monitoring script will notify the users of the lack of adequate booking and eventually kill the processes if the situation is not fixed.**

**IMPORTANT**: users who go beyond their booked resources (be it in use time or in amount of resources used) will be treated under a progressive penalties policy:

* First and second violations: user receives warnings from admin group (+ support from the group to prevent this from happening again)
* Further violations: 1-month ban for each successive violation of resource constraints: 1 month ban in the 3rd violation, 2 months in the 4th, 3 months in the 5th. After 6 violations the user is permanently banned from the server.

# Registering into the resource booking system

Before being able to book or use the ML server, you will need to register in the booking system: <http://57.128.172.217:8000/>**.** When registering you will need to provide:

* Name: Use your full name.
* Email: Your **Aston** email address.
* Slack id: Your Slack user id (see below).
* Notes: **Specify the name of your supervisor** (MSc/PhD students).

# How to find your Slack user id

Make sure you follow steps 1 to 4. It will copy your Slack user identifier into the clipboard:

| **Step 1:** On the Aston EPS - GPU Slack workspace, click on your profile icon on the bottom left: | **Step 2:** Click on ‘Profile’: |
| --- | --- |
| **Step 3:** Click on the three dots on the right: | **Step 4:** Click on ‘Copy member ID’: |

# Requesting resources

Requests for credits are collected every month up to 5 days before the end of the calendar month. Requests are made using the booking service: <http://57.128.172.217:8000/>. By default, requests need to be made monthly - e.g., if you have unused resources for July these don’t roll over to August.

* When registering on the booking system, you will need to provide your Slack ID. Otherwise you won’t be able to receive notifications, so your registration will not be accepted.

The steering group will consider the requests made and allocate the available resources to the users. The basic resource budget for staff and PhD students is currently set as 3500 credits for PhD students and 2000 for MSc students. Each GPU slot hour is worth 10 credits, each CPU slot hour is worth 15 credits.

These budgets may vary with demand: months with higher demand may result in reduced budgets, while we can give users increased budgets in low-demand months. We'll strive to provide resources to everyone, but in cases of particularly high demand we may need to refuse some requests. Given the stated mission of the server, *priority will always be given to early-career researchers doing proof of principle work for future proposals, funded projects and new collaborations across different research groups at Aston.* Our voluntary support team (listed in the *Steering Group*) is also granted some level of priority (5000 credits instead of 3500).

# Booking resources

Once you receive the monthly budget (communicated via the email) you need to book the resources using the booking system <http://57.128.172.217:8000/>.

In the booking system, you will be able to go to check the availability of the resources you'd like to book. Once you're ready to book, please select the resources you'd like to use and the time period, and make your booking.

**IMPORTANT**: When booking resources, your booking **must** adhere to the time restrictions below:

* Start time:
  + 9:00 (am)
  + 17:00 (pm)
  + The current hour.
* End time:
  + 9:00 (am)
  + 17:00 (pm)

This means that you can start your bookings at either 9:00 (am), 17:00 (pm) or the current hour (to avoid spending credits unnecessarily in the past). The bookings **must** always end at 9:00 (am) or 17:00 (pm) only. Also, please be mindful of others - if your slot ends at 9am, try to plan ahead to get everything finished by 8:30am, so that the next person has the resources available at the start of their slot.

# IMPORTANT: Acknowledgements in publications

By using the server you agree to acknowledge its use in any publication that may be produced as a result. The following acknowledgement is suggested, but you are free to adapt it if necessary:

*“Experiments were run on Aston EPS Machine Learning Server, funded by the EPSRC Core Equipment Fund, Grant EP/V036106/1."*

# General How-Tos

**How to connect to the server?**

The server name is *csml01*. **If you're connecting from outside Aston’s network,**  first you need to connect through the VPN. Check <https://solve.aston.ac.uk/tas/public/ssp/content/detail/knowledgeitem?unid=7a1f6638-394f-4bd5-bf25-113a7bda24aa> for guidance.

Once you're connected through the Aston network (either by being on campus or connecting through the VPN) just run:

[ssh username@csml01.campus.aston.ac.uk](mailto:username@csml01.campus.aston.ac.uk)

where *username* is the one you use to log in the university systems.

**How to transfer files?**

To transfer a local folder to the server you need to use the *scp* command:

scp -r **local\_folder\_name** csml01:/**your\_home\_directory**

where **local\_folder\_path** is the folder you want to upload, and **your\_home\_directory** is the path to your home folder in the server. To verify which is your home directory, when connected to the server just execute the command *pwd*.

If you only want to transfer a single file you can run:

scp **local\_file\_name** csml01:/**your\_home\_directory**

(i.e., just remove the –r option)

**How to install Conda (Python)?**

1: Download conda:

wget https://repo.anaconda.com/archive/Anaconda3-2021.05-Linux-x86\_64.sh

(replace the URL by the latest available Linux version in https://repo.anaconda.com/archive/)

2: After the download, run:

bash ./Anaconda3-2021.05-Linux-x86\_64.sh

and follow the instructions (if needed, replace the file name by the one you downloaded).

**How to install pip**?

In case you don’t want to install conda and just want to have pip locally, follow the next steps:

wget https://bootstrap.pypa.io/get-pip.py

python3 get-pip.py --user

Add the following line to your *.bashrc* file:

export PATH=”$PATH:~/.local/bin”

(NOTE: It is better to replace ~ with the absolute path to your user home folder to avoid warnings.)

source ~/.bashrc

Then use the following syntax to install pip packages:

python3 –m pip install --user package\_name

**Install Pytorch**

Using conda (note: the command below is a single line):

conda install pytorch torchvision torchaudio   
 cudatoolkit=11.1 -c pytorch -c nvidia

Using pip (note: the command below is a single line):

python3 -m pip install --user torch==1.8.1+cu111  
 torchvision==0.9.1+cu111 torchaudio==0.8.1 -f   
 https://download.pytorch.org/whl/torch\_stable.html

**Notice that the versions informed in the strings above may change. Please ask on the Slack channel for more information.**

To verify if the installation is ok, execute the ***Python*** code below. The output must be **10**.

import torch  
torch.cuda.device\_count()

**Install TensorFlow**

Using conda:

conda install tensorflow-gpu

**Set up personal R package library**

1. Start R.
2. Run this R command:  
   dir.create(Sys.getenv("R\_LIBS\_USER"), recursive = TRUE)  
   This will create a personal \Users\$user\My Documents\R\win-library\x.y for version x.y of R. It only needs to be run once, to make sure that the directory structure exists.
3. You can install packages at this point in the usual way.
4. You will need to restart R to see the packages you installed in the previous step in the Load Packages dialog.
5. You will need to update the personal library if the version changes.

**Running long tasks in the server**

Frequently, tasks will take many hours to finish and users may want to log out from the SSH sessions or their computers. By default, processes will stop after the SSH session is closed. We suggest two possible ways of ensuring that the processes keep running after logging out: using tmux, or using nohup. There are other alternatives, such as screen, but these two are the ones recommended.

**Running tasks in detachable sessions using tmux**

In addition to making the sessions detachable, **tmux** allows creating multiple windows in the same SSH session.

* Once you're logged into the server, type ***tmux*** to enter a **tmux** session. You should see a green bar at the bottom of the screen. That means that you have entered a **tmux** session.
* Start whatever time-consuming task you need to run.
* You can now “detach” your session and close the ssh connection. To detach your session you need to press Ctrl+B followed by D. After detaching the session, you should no longer see the green bar, and you should see the command prompt. Don’t worry, your process is still running in the background. You can close the session and come back later.
* If your SSH connection fails at any point, or if you have detached your tmux session, you may want to attach it again. To do so, you must run the command:
  + **tmux attach**
* **Extra points:**
  + **tmux** has support for sub-windows, which are very helpful. You can read more about tmux in <https://hamvocke.com/blog/a-quick-and-easy-guide-to-tmux/>.
  + **tmux** allows using the mouse to change between windows. To activate the mouse, you need to press Ctrl+B followed by : . That will take tmux into command mode. To enable the mouse you need to type “set -g mouse on”.

**Running tasks using nohup**

Using nohup is the simplest alternative. A command `command` can be run using nohup by typing:

nohup command &

The textual output of the command will be written into `nohup.out`. You can visualise its output by running `cat nohup.out` or `tail nohup.out`.

**Running a process in a specific GPU**

There are several ways of achieving this. Usually, each framework (Pytorch, Tensorflow, sklearn...) has its own functions to send the data to a specific GPU. Here, a general method (framework agnostic) is explained in case it can help you. This method consists in setting an environment variable (CUDA\_VISIBLE\_DEVICES) that will tell the system which GPU is available for a specific terminal session. For example, if you want that your process only has GPUs with id 2 and 3 available, you have to set the variable in this way:

export CUDA\_VISIBLE\_DEVICES=2,3

If you want to check if the variable has been correctly set, you can run:

echo $CUDA\_VISIBLE\_DEVICES

The output should be "2,3". That will ensure any process that you run on that specific terminal will see only GPUs 2 and 3. Please bear in mind that if you open a new terminal you will have to set the variable again.

**Limiting thread usage**

When using PyTorch, it is suggested to use the below to limit the thread usage of PyTorch operations:

torch.set\_num\_threads(1)

We can also limit the thread usage for processes like NumPy and similar libraries with the below:

OMP\_NUM\_THREADS=1 NUMEXPR\_NUM\_THREADS=1 MKL\_NUM\_THREADS=1 python train.py …

Or:

import os

os.environ["MKL\_NUM\_THREADS"] = "1" # noqa F402

os.environ["NUMEXPR\_NUM\_THREADS"] = "1" # noqa F402

os.environ["OMP\_NUM\_THREADS"] = "1" # noqa F402

The above should be placed within the main.py (where your imports are).

For those who are limited by the speed of their data loaders, I would strongly suggest looking into [NVIDIA DALI](https://docs.nvidia.com/deeplearning/dali/user-guide/docs/index.html).

**Limiting core usage**

This is normally not necessary, but if you find the above did not work, try this:

An example to limit the code to only run with core 0 is below:

taskset -c 0 python train.py

Another example shows how to use multiple cores (cores 0, 1, 2 and 3):

taskset -c 0-3 python train.py

**How to install Julia?**

First, please follow the instructions in **How to install Conda (Python)** to install Conda (Miniconda works as well and it will require less disk space). Once installed, you can install Julia in any Conda environment by running the following command:

conda install -c conda-forge julia