Maximizing run:ai gains:

An intro for Data Scientists

Maximizing run:ai gains

Who is this for?

Data Scientist/Researchers who are now using run:ai

What is it for?

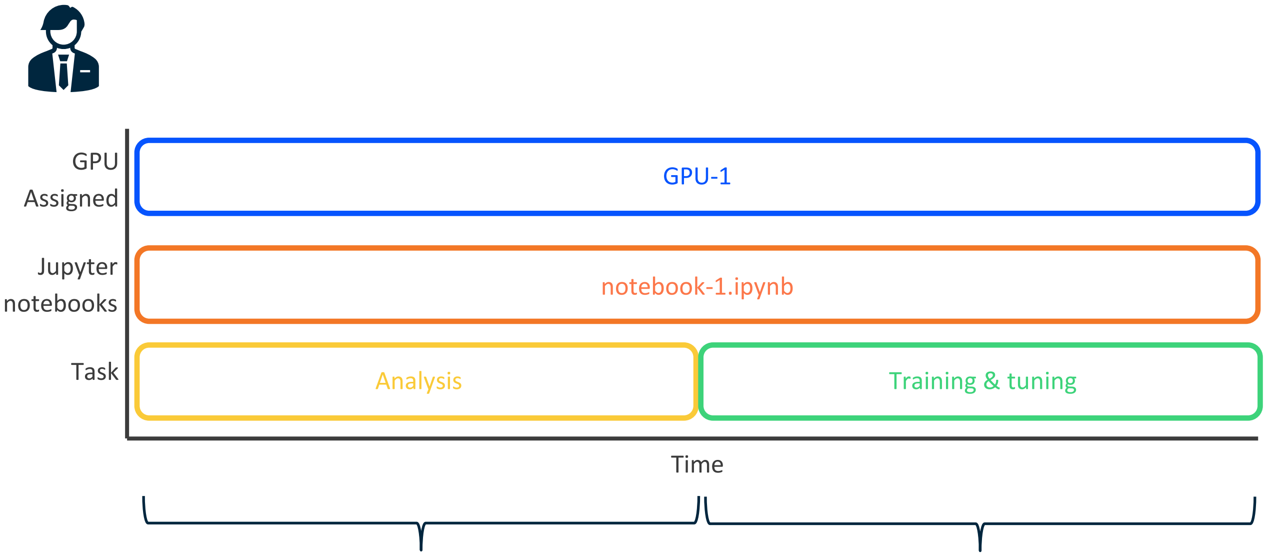
Inform on **best practices** with run:ai

What is often done

Common workflow for data scientist

1. Provision GPU resources

2. Create Jupyter notebook



3. Perform exploratory analysis

- Modify code blocks
- Test code blocks

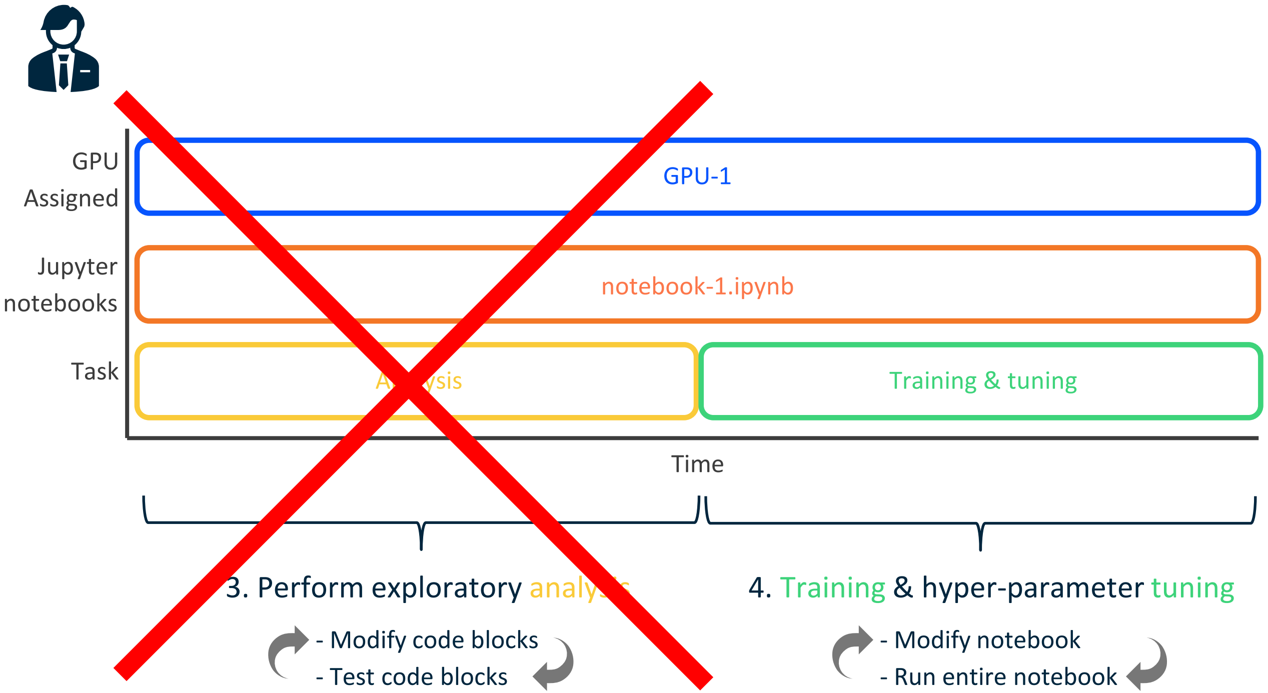
4. Training & hyper-parameter tuning

- Modify notebook
- Run entire notebook

Common workflow for data scientist

1. Provision GPU resources

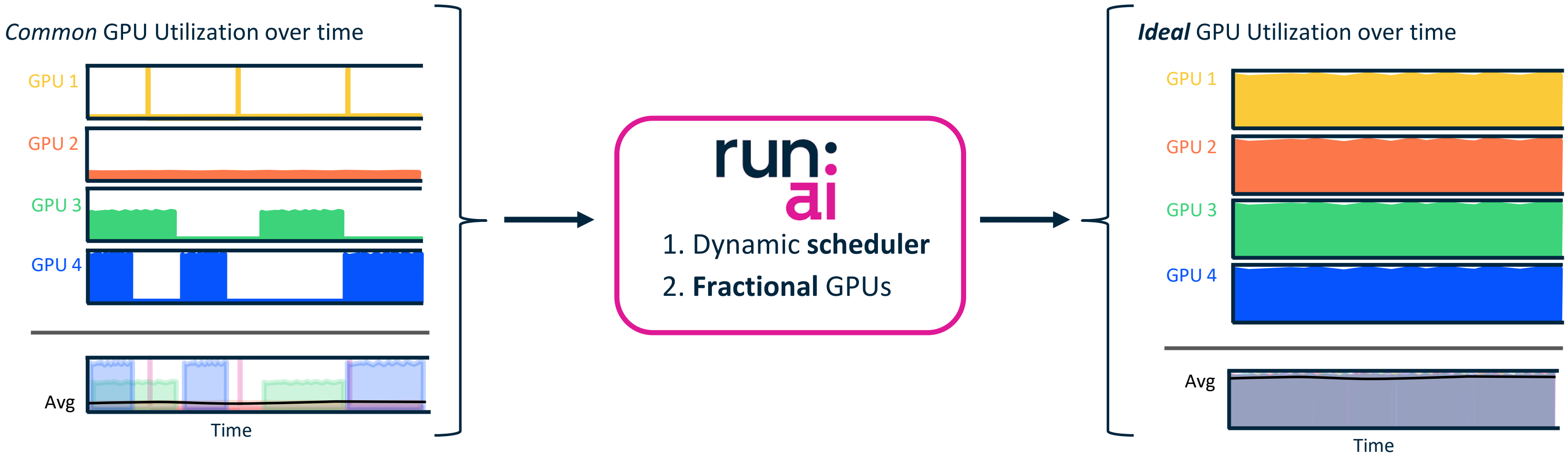
2. Create Jupyter notebook



Why not to do it

Understanding goal of run:ai

The run:ai software seeks to **maximize GPU utilization**



Some run:ai terminology

Two types of ‘jobs’

For both, resources are allocated as long as the job exists

‘**Interactive**’ jobs

- Jobs that run indefinitely
- Only stops if user deletes the job

<u>Resources can be tied up forever</u>

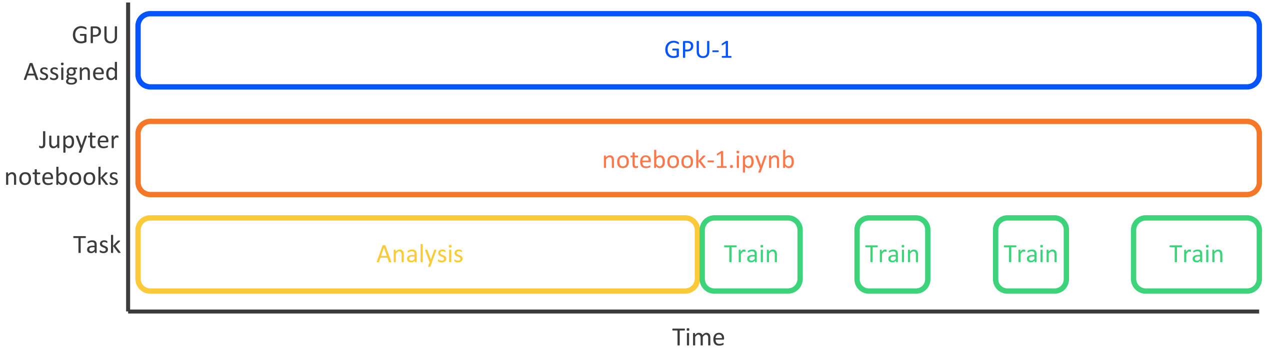
‘**training**’ jobs

- Jobs that run until script finishes
- Will automatically stop once done

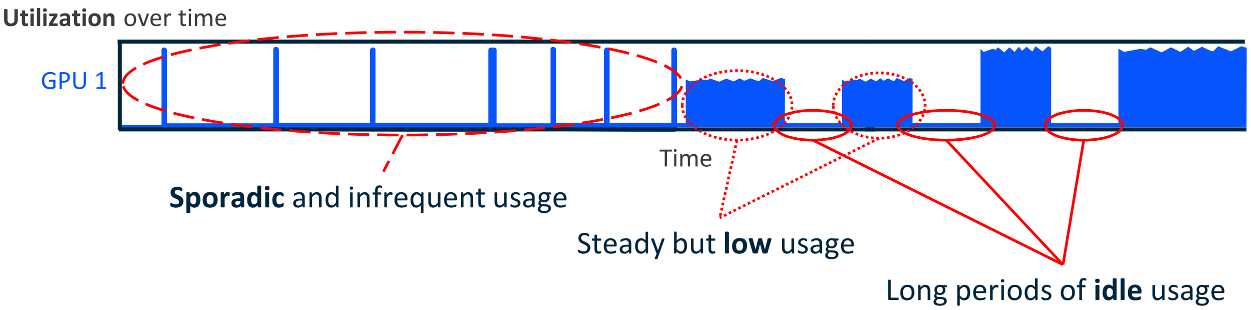
Resources are only tied up while job is running

How common workflow can inhibit run:ai

Working like this…

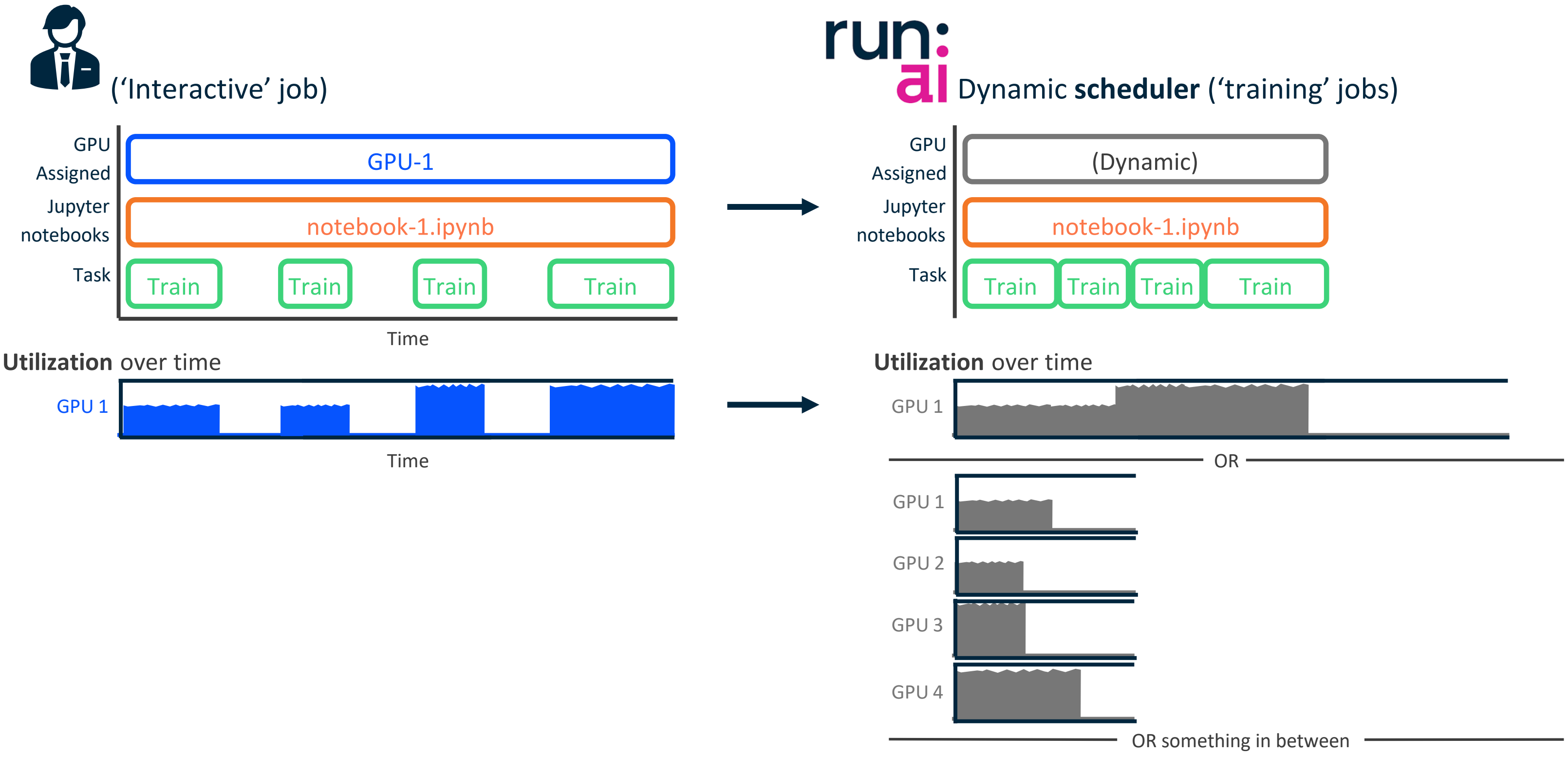


…can lead to this utilization

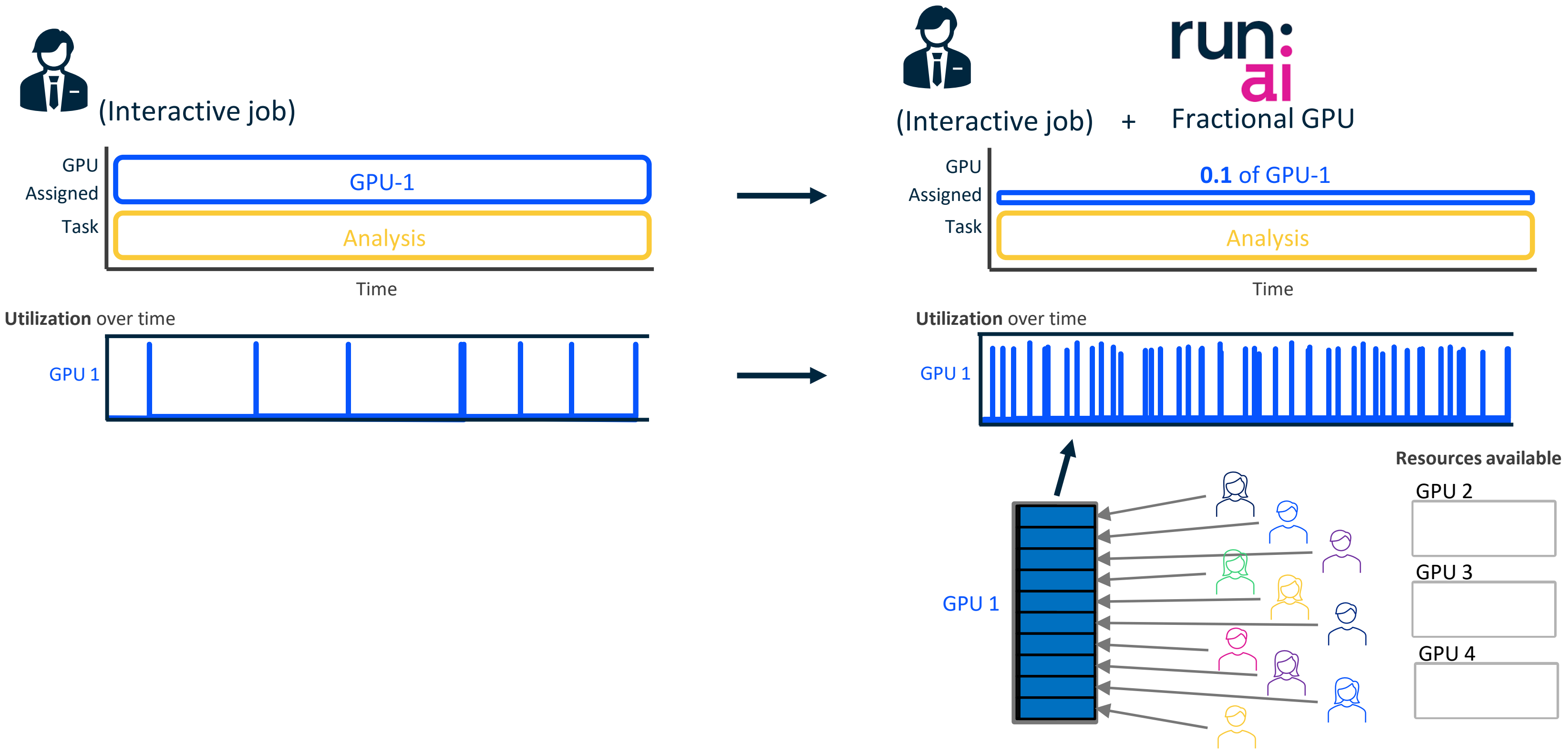


How to do it right (with run:ai)

Use the run:ai **scheduler** for training & tuning

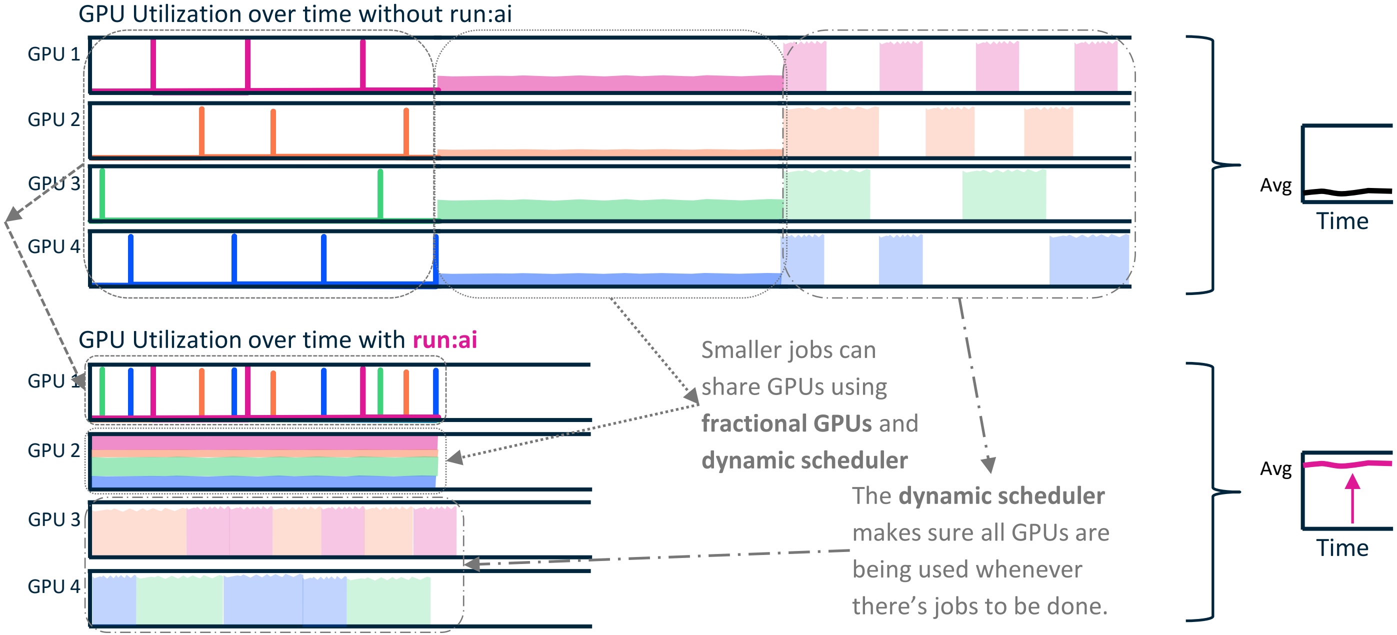


Use the run:ai **fractional GPU** for analysis



Gains in GPU Utilization with run:ai

‘Analysis’ work to be done concurrently on one GPU with the **fractional GPUs** feature



Net result of optimization with run:ai

- Resources become free much sooner
- More work gets done in less time

Summary

Summary of run:ai best practices

Fractional GPUs
(‘*interactive*’ jobs)

Analysis

- When GPUs **usage is sporadic**/intermittent
- For example, when you are writing scrips/notebooks or
- when you are **testing segments/blocks** of **code**

Dynamic scheduler
(‘*training*’ jobs)

Training/tuning/ETL

- When GPUs **usage is consistent** or steady
- When you are **running entire scripts or notebooks** until they finish

Fractional + Scheduler
(‘*training*’ jobs)

Training/tuning/ETL

- When GPUs **usage is consistent** or steady, **but low**
- For example, when training small models that **do not fully saturate** a **GPU**

Full GPU
(‘*interactive*’ jobs)

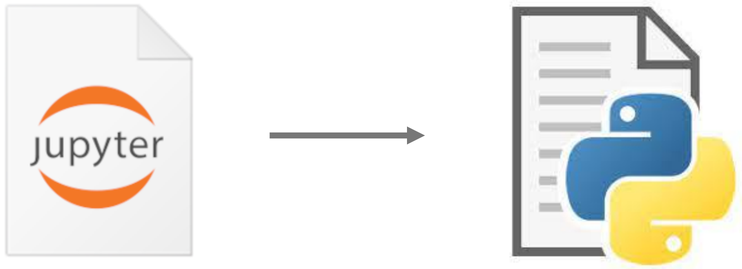
- **This should be done *rarely***
- For example, this can be done if you are trying to figure out max batch size to saturate a GPU
- ‘Interactive’ job should be stopped after max batch size is determined; ‘training’ job should be submitted afterwards

Tips

Tips

1. You can turn a jupyter notebook into a python script with the *jupyter nbconver* command

```
jupyter nbconvert --to script my-notebook.ipynb
```



my-notebook.**ipynb** my-notebook.**py**

---

2. Then run the python script (note that any blocks with ‘cell magic’ will not convert properly; comment out or remove ‘cell magic’ lines before converting the notebook)

```
python my-notebook.py
```

Tips

If in your workflow, you modify and run your notebook multiple times….

Training & hyper-parameter tuning

- Modify notebook
- Run entire notebook

my-notebook-1.ipynb

… Instead, create multiple copies of the notebook, modify each as needed…

my-notebook-1.ipynb

my-notebook-2.ipynb

my-notebook-3.ipynb

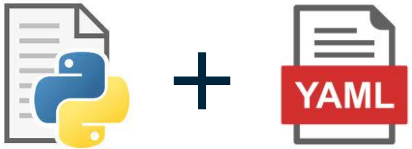
… and after converting them to python scripts, submit each to the job scheduler.

```
python  my-notebook-1.py
python  my-notebook-2.py
python  my-notebook-3.py
```

run:ai scheduler

Tips

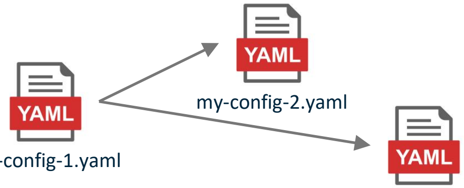
1 If you pass in configuration files (e.g. a yaml file) to your python scripts…



my-script.py + my-config-1.yaml

```
python  my-script.py  --config  my-config-1.yaml
```

2 …you can create multiple configuration files…



my-config-1.yaml

my-config-2.yaml

my-config-3.yaml

3 … and submit each configuration to the job scheduler.

```
python  my-script.py  --config  my-config-1.yaml
python  my-script.py  --config  my-config-2.yaml
python  my-script.py  --config  my-config-3.yaml
```

run:ai
scheduler

Thank you!

info@run.ai | www.run.ai