Data Science Workflow

Lecture 3: Training on a Cluster GPU

M.Sc. Teodor Chiaburu

FB 6, Data Science, Berliner Hochschule für Technik

19. Juni 2025

Introduction

To check the capabilities of our cluster's machines:

```
kubectl get nodes

kubectl describe nodes

kubectl describe node cl-worker17 # or any other node
```

Overview Workers and GPUs

Tabelle: GPU Cluster Configuration

Type	Node name(s)	#GPUs	RAM[Gb]/GPU
K80	cl-worker17 (w17)	8	11.441
P100	w19	8	16.384
V100	w20	2	32.768
	w22	2	
	w23	2	
	w25	1	
A100	w24	1	81.92
	w27	8	40.96
	w28	8	
	w29	1	
H100	w34	1	81.559
H200	w37	4	143.771
B200	w36	8	183.359

Adjusting the Deployment's yaml

In *remote-deployment.yml* you just need to (un)comment these lines:

```
...
resources:
  requests:
    cpu: "1" # minimum number of cpus you want
    #memory: 5Gi

    nvidia.com/gpu: 1 # Request one GPU, if available
  limits:
    #cpu: "2" # maximum number of cpus you want
    #memory: 10Gi # Shut down if it exceeds this RAM

    nvidia.com/gpu: 1 # Always set limit to gpus, otherwise you get
         allocated all the available ones
...
nodeSelector:
    gpu: k80 # For now, please only switch between k80, p100 and
        v100
...
```

Installing PyTorch

We will be developing and training our models in PyTorch. When installing PyTorch, you need to take note of the CUDA version of your GPU (see table above). Once connected to a pod with GPU, you can also check the dependencies and usage of your GPU via:

```
nvidia-smi
```

A PyTorch installation that works for all our machines is:

```
pip install torch==2.2.0 torchvision==0.17.0 torchaudio==2.2.0 \
--index-url https://download.pytorch.org/whl/cu118
```

You may try to install newer versions of PyTorch, but these may not be compatible with all the workers on our cluster.
Check the official installation guide for more details.

Installing PyTorch

Recommendations

1. You may notice quite soon, that you have reached the maximum storage size of your PV. Recall that the original request in *storage.yml* was for only 5Gb. Increase that size in the yml-file to e.g. 50Gb and just reapply the PVC:

```
kubectl apply -f storage.yml
```

   This will overwrite the previous request and increase the maximum size in your PV. You don't need to delete the old PVC. **Don't forget that you can't decrease the size of this PV from now on; you would have to create a new one.**

2. You may want to create (multiple) different *.venv* folders (for instance, for different PyTorch versions, that don't work on all machines).

Practice Session 1

Preliminaries

Task 1: Download the folder *Lecture_3* from Moodle into your local course folder and push it to your remote repository.

Task 2: From your local VSCode: Extend the size of your PV to 50Gb and send the request to the cluster. Check that it's been successfully resized.

Task 3: Modify the deployment file to request access to a GPU and send it to the cluster. Check before which GPUs are free and afterwards whether you received the desired one.

Task 4: Connect to your pod and open a VSCode window on it. Pull the code from your remote repo into your PV.

Task 5: Install PyTorch in your environment (see above). Also install the new requirements under *Lecture_3/Model_Training*.

Task 6: Download the folder *data* with the bees images and masks from the Google Drive link in *main.py*.

Localizing Bees in Images

- KInsecta Project
- Fine Grained Image Analysis
- beexplainable
- A look at the data:
  - Images scraped from iNaturalist
  - iNat Challenge
  - Segmentations in Label Studio
  - Annotated dataset publicly available:
    - Zenodo
    - Paper

A Look at the Model

- Mask R-CNN
  - capable of proposing bounding box regions, segmenting and classifying objects
  - for the bees, only the segmentation head was used
  - Approach:
    1. Train segmenter on the manually annotated bee dataset
    2. Download a couple of thousands more images from iNat
    3. Generate masks on the new images with the pretrained segmenter

Note: Nowadays, many people use *Segment Anything Model* (SAM) for segmenting images. It was trained by Facebook on 11M images with 1B+ masks and has become the foundational segmentation model in AI. However, there are some problems:

- It's very large (about 640M parameters) - can't run on edge devices without quantization
  - Meanwhile, there is a 'light' version: Mobile-SAM
- It only works off-the-shelf for images belonging to 'ImageNet-like' distributions - if your input images are not plentiful on the internet e.g. soil profile fotos or proprietary medical images, you need to fine-tune

How Do We Evaluate?

- For now, we are only interested in the quality of the masks (no bounding boxes or classification)
- **IoU** (Intersection over Union) - how well the predicted and the ground-truth masks overlap
- For now, just one fixed train-validation split
- More on HPO and Model Evaluation next week

Practice Session 2

Training on GPU, Debugging, Inference

Task 1: Set 1-2 breakpoints in the code and inspect the intermediate states of the objects with the Debugger e.g. in *dataset.py* or *metrics.py*.

Task 2: Run the script in *main.py* till the final epoch. If you're on a slower machine, reduce the number of epochs to not wait for too long. Optional: start a *screen* session before the training; once the script is running, you can close the window and the port-forwarding. When you reconnect and reattach to the screen session you will get back the training logs.

Task 3: Pick a couple of bee images and masks from the validation set, use your trained model for inference and plot the predicted segmentation masks (in *inference.py*).

Task 4: Once you're done, commit and push everything to remote.

Task 5: And as always...

And as always...

Don't forget to shut down your pod once you're finished!

```
kubectl scale deployment dsw --replicas=0
```

Notes

1. If you have a Deployment running already, for instance on a K80, and you reapply the same deployment yaml but requesting a different machine, say P100, this will not overwrite the previous deployment, but create a new one in a separate pod. So whenever you want to switch machines, first shut down your current application, then reapply the new deployment.
2. You can't shut down a deployment by just deleting its pod:

   ```
   kubectl delete pod [pod-identifier]
   ```

   K8s will simply reallocate it to a new pod. The reason for this is, that K8s will take care to keep your apps running, even if something happens to the pod. Notice that your deployment's Status says 'Running'. With Jobs, however, that have the Status 'Completed', you can delete their pod and K8s will not create a new one in its place.