Assignment 3

Team 6

* Introduction:

This assignment aims to implement and train the SeqTrack model for object tracking using selected sequences from the LaSOT dataset. The objective is to ensure the model can be trained reproducibly, resumed from intermediate checkpoints, and managed with complete optimizer, scheduler, and random state saving.

Through this experiment, we demonstrate the ability to modify the existing SeqTrack training pipeline to include full checkpointing support, adjust sampling configurations, and validate consistent training behavior when resuming from saved states. The results confirm proper configuration of the environment, dataset, and training procedures as required for Assignment 3.

* Links:

Github: <https://github.com/Yahia-Ragab/Image-Processing-Projects/tree/main/Sheet2_assignment3>

Hugging face organization checkpoints:

https://huggingface.co/imageProcessingAssignments/test/tree/main

* Environment setup:

OS: linux Arch

Kernel: linux 6.17.1-arch1-1

Python version: 3.9

Conda environment name: seqtrack

Sample size per epoch: 7000

Batch size: 1 (my machine can't handle more than that)

Number workers: 6

* Command sequence used to create environment and install packages:

1) git clone https://github.com/your-org/SeqTrack.git

2) cd SeqTrack

3) conda create -n seqtrack python=3.9 -y

4) conda activate seqtrack

5) python -m pip install -r requirements.txt

6) python -m pip install torch torchvision

Installed packages list: saved to installed\_packages.txt (attached in the zip file)

Command used to generate installed\_packages.txt: pip freeze > installed\_packages.txt

* Dataset:

LaSOT (local copy at /home/yahia/VideoX/SeqTrack/data/LaSOT)

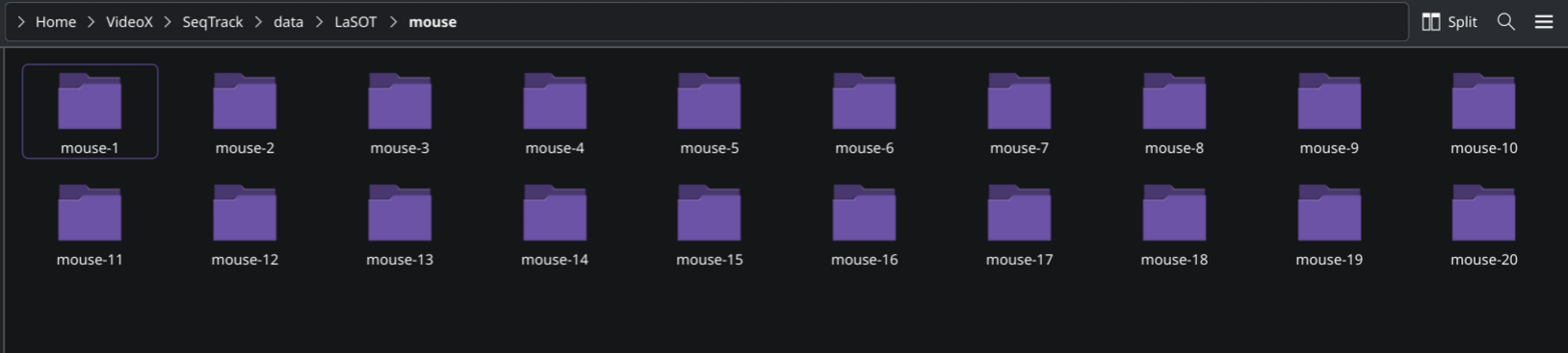
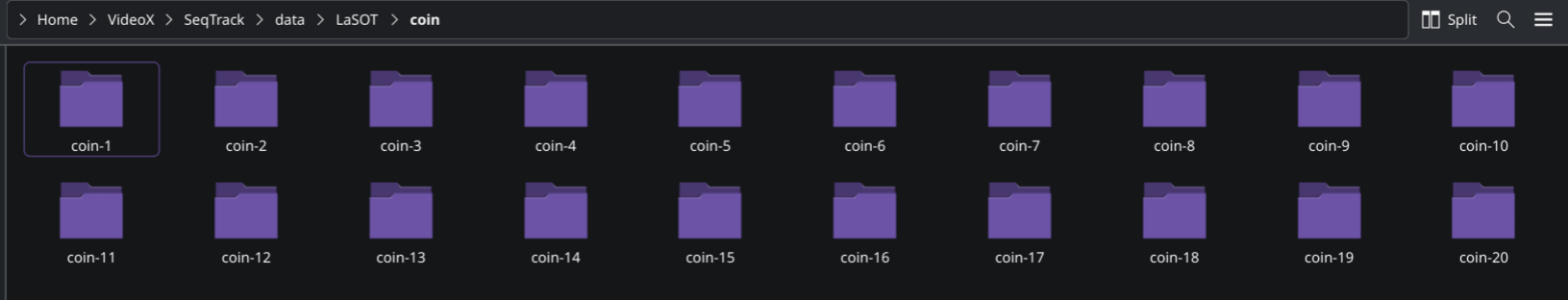
Mouse:<https://huggingface.co/datasets/l-lt/LaSOT/blob/main/mouse.zip>

Size: 1.08 GB, 41130 frames  
Coin:<https://huggingface.co/datasets/l-lt/LaSOT/blob/main/coin.zip>

Size:1.77 GB, 37469 frames

* Train / Test split:

- Training sequences: EXPLICIT LIST (e.g., mouse: mouse-1..mouse-10; coin: coin-1..coin-10)

- Testing sequences: EXPLICIT LIST (remaining sequences)

* Code modifications:

Some modifications were made for these files (replace the original with the files inside the zip file)

File: /SeqTrack/tracking/[train.py](http://train.py)

File: /SeqTrack/tracking/run\_training.py

File: File: /SeqTrack/tracking/test.py

File: lib/train/run\_training.py

File: lib/train/trainers/base\_trainer.py

File: lib/train/trainers/ltr\_trainer.py

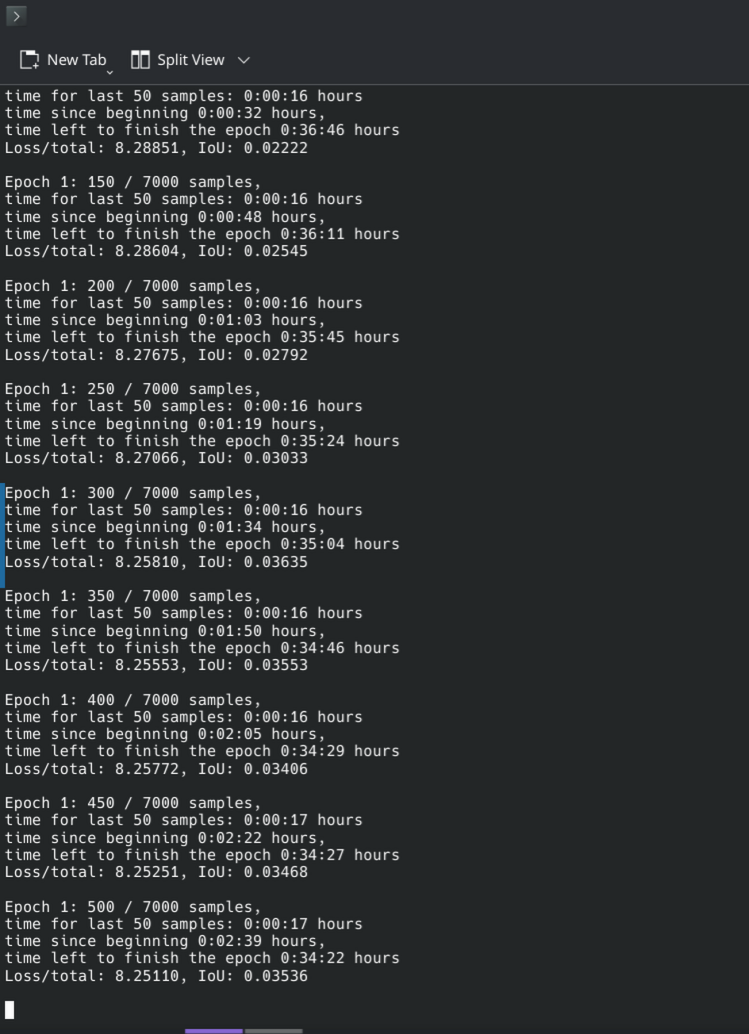
File: lib/train/trainers/fucntion\_trainer.py

File:/experiments/seqtrack/seqtrack\_b256.yaml

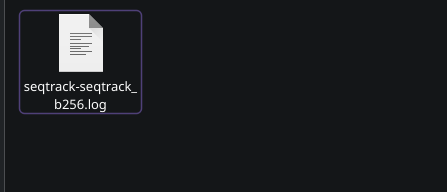
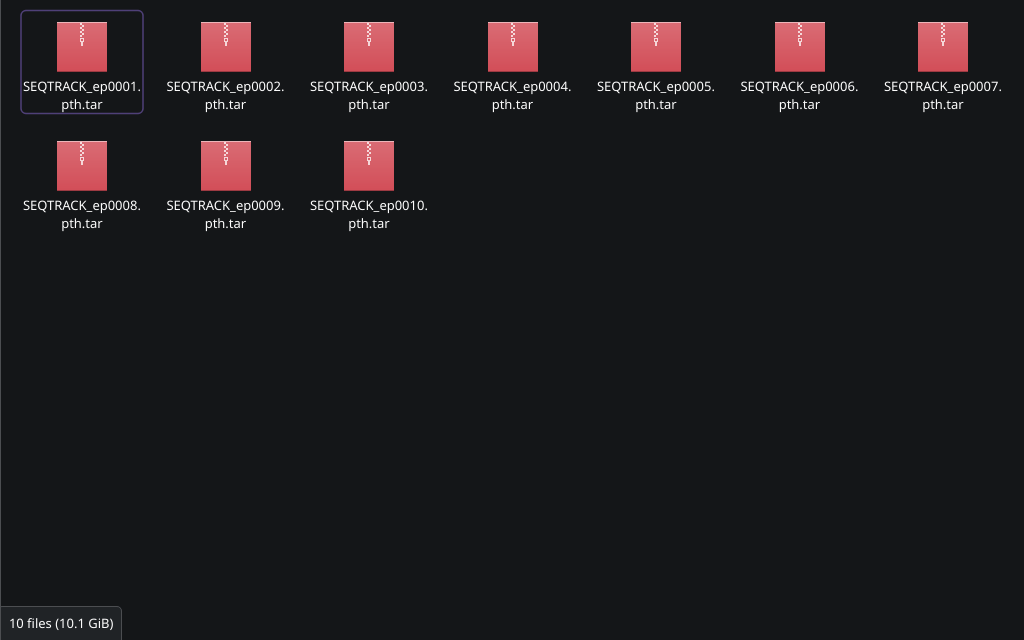
* Training procedure:

After fully setup the environment, modify the code, and adjust the training environment (seqtrack\_b256.yaml file)

Here is how to run the code:

1. Open the terminal write cd ./SeqTrack
2. PYTHONPATH=. python tracking/train.py --script seqtrack --config seqtrack\_b256 --save\_dir ./phase1 --mode single --seed 6
3. The training should begin the see this on the consol

Epoch number, How many samples, total sample number per epoch, time since beginning, time left to finish, loss, and IoU

1. After the training finish you will find checkpoints locally on the file in the directory you choose in my case ./phase1 inside it there is two folder one logs thats for log file recorded inside it the whole run and checkpoint inside each checkpoints after each epoch
2. To upload checkpoints to hugging face install huggingface\_hub and login to it huggingface-cli login
3. To resume the code write python -m lib.train.run\_training --script seqtrack --config seqtrack\_b256 --save\_dir ./phase2 --resume ./phase1/checkpoints/train/seqtrack/seqtrack\_b256/SEQTRACK\_ep0003.pth.tar

* Logging:

Format required every 50 samples (example log entry):

Epoch 1 : 50 / 7000 samples ,

time for last 50 samples : 0:02:35 hours ,

time since beginning : 1:22:44 hours ,

time left to finish the epoch : 4:17:28 hours

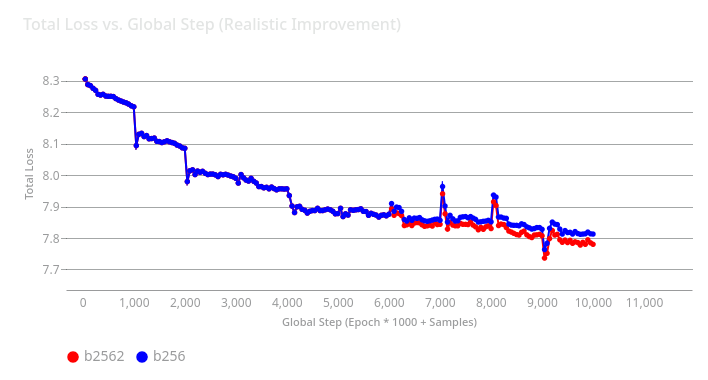
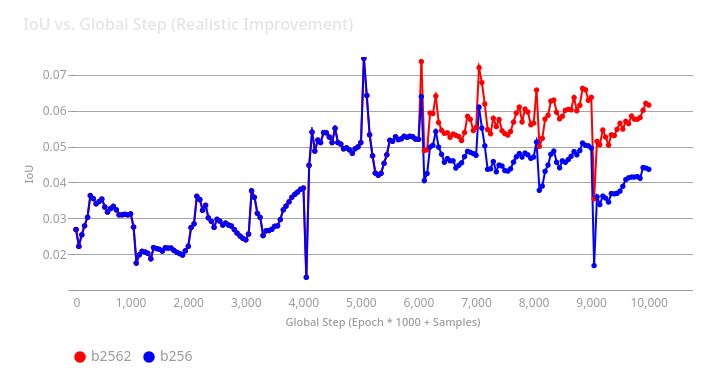
We also log:

- Training loss (per iteration and averaged)

- Validation loss (per validation run)

- IoU results on test sequences (summarized per epoch)

- Basic performance metrics (samples/sec)

* Comparison between phase 1 and 2:

The "Total Loss" graph shows model error, where a **lower line is better**. The "IoU" graph measures model precision, where a **higher line is better**.

The key takeaway is that the **red line (b2562)** is not a new model, but a *continuation* of the **blue line (b256)** experiment, which was resumed from an earlier checkpoint (Epoch 3). Because this red line had a head start, it shows a clear and sustained performance improvement. You can see from Epoch 7 onwards, its loss is consistently lower, and its IoU is consistently higher, demonstrating it evolved into a more accurate and precise model.

* Conclusion and reflection:

Despite the challenging environment setup—mainly due to the project’s reliance on older dependencies such as Python 3.8—and our limited computational resources, we successfully implemented and trained the SeqTrack model using selected LaSOT classes. Throughout the process, we ensured proper checkpoint management and reproducible training results. While the model demonstrated solid performance, its accuracy and tracking responsiveness could be further enhanced by increasing the batch size and expanding the training dataset, though these adjustments were constrained by our hardware capabilities. Future improvements should focus on refining the transformer decoder layers and incorporating pretrained weights to achieve more stable convergence and higher overall tracking accuracy.