**Assignment 3 : SeqTrack Training**

**Team 16**

**GitHub:** **https://github.com/Amr20004/assignment\_3**

**Hugging-Face:** **https://huggingface.co/Amr20004/assignment\_3/tree/main**

**-:Selected Dataset:-**

We picked **Airplane class** and **Bicycle class** from laSOT dataset.

* Airplane class has around 86,000 samples. We used 16 sequence from the total 20 sequence and the left 4 for testing
* Bicycle class has around 40,772 samples. We used 16 sequence from the total 20 sequence and the left 4 for testing

We used train-test split file to prevent the model from training on the whole data

A screen shot of a computer

AI-generated content may be incorrect.

Here we updated lasot.py file from line 59 to line 70 to force the model to train the model on the specified training samples only.

**-:installed packages:-**

torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0

PyYAML

easydict

cython

opencv-python

pandas

tqdm

pycocotools

jpeg4py

tb-nightly

tikzplotlib

thop <## instead of pip install --upgrade git+https://github.com/Lyken17/pytorch-OpCounter.git

colorama

lmdb

scipy

visdom

vot-toolkit==0.5.1 <## instead of pip install git+https://github.com/votchallenge/vot-toolkit-pythontimm

yacs

**List of filenames and corresponding line numbers**

**Limiting Training to 2 Specific Classes (airplane, bicycle) and Using Train/Test Split Files:**

**File:** SeqTrack/lib/train/dataset/lasot.py

**Line ~43 (Class List):** Replaced the original line self.class\_list = [f for f in os.listdir(self.root)] with self.class\_list = ['airplane', 'bicycle'] to hardcode your chosen classes.

**Line ~59-69 (Sequence List Generation):** Replaced the original logic (or our previous hardcoding of range(1, 21)) with the block that reads from your custom split files (training\_set.txt or testing\_set.txt) based on the split argument. This ensures only the sequences listed in the appropriate file are used for training or testing.

**Adjusting Log Printing Format:**

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

**Line 45~49 (Added Helper Function):** Added the \_format\_time(self, seconds) function to convert seconds into the required H:M:S hours format.

**Line ~126 (Initialize Last Log Time):** Added self.last\_log\_time = self.start\_time inside the \_init\_timing function to track time between logs.

**Lines ~162-204 (Replaced \_print\_stats Function):** Deleted the original \_print\_stats function and replaced it with a new version that calculates time\_for\_last\_batch, time\_since\_beginning, and time\_left\_to\_finish, formats them using \_format\_time, and prints the required detailed log message to both the console and the log file.

**Automatic Checkpoint Uploading to Hugging Face:**

We encountered problems with Hugging Face Access Token within Hugging Face API so we couldn’t upload checkpoint automatically

We uploaded it manually as proof for our progress.

**Ensuring Seamless Training Resumption (Saving Full State):**

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

**Lines ~131-133 (Saving States):** Modified the state dictionary within the save\_checkpoint function to add keys and save the states for 'lr\_scheduler', 'rng\_state', and 'rng\_state\_cuda'.

**Lines ~219-224 (Loading States):** Modified the for key in fields: loop within the load\_checkpoint function to add elif conditions that correctly load the state dictionaries for 'lr\_scheduler', 'rng\_state', and 'rng\_state\_cuda' when resuming.

**Enabling the Resume Feature (Command Line Argument):**

**File:** SeqTrack/lib/train/run\_training.py

**Line ~75 (Add Argument):** Added parser.add\_argument('--resume', ...) to define the command-line flag.

**Line ~89 (Pass Argument):** Added resume=args.resume to the run\_training(...) function call.

**Line ~26 (Accept Argument):** Added resume=False to the run\_training(...) function definition.

**File:** SeqTrack/tracking/train.py

**Line ~18 (Add Argument):** Added parser.add\_argument('--resume', ...) to the parse\_args function.

**Lines ~30 & ~34 (Pass Argument):** Added --resume %d and , args.resume to the train\_cmd strings for both single and multiple GPU modes.

**Resume from a specific epoch number:**

**File: SeqTrack/tracking/train.py**

**Line ~19 (Add Argument):** Added parser.add\_argument('—resume\_epoch', ...) to the parse\_args function.

**Lines ~30 & ~34 (Pass Argument):** Added –resume\_epoch %s and , args.resume\_epoch to the train\_cmd strings for both single and multiple GPU modes.

**File: SeqTrack/lib/train/run\_training.py**

**Line ~76 (Add Argument):** Added parser.add\_argument('—resume\_epoch', ...) to define the command-line flag.

**Line ~89 (Pass Argument):** Added resume\_epoch=args.resume\_epoch to the run\_training(...) function call.

**Line ~26 (Accept Argument):** Added resume\_epoch=None to the run\_training(...) function definition.

**Line ~65 ~78(Process the Argument):** check if it will automatically resume or resume from a specific epoch

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

**Line ~62(add start epoch):** update train function to make it accept the start\_epoch.

**Line ~83(use start epoch):** use start\_epoch in train function

**Setting the Fixed Seed at the Start of Each Epoch**

To meet the requirement of setting the seed at the beginning of every epoch for maximum reproducibility, the original seeding logic was moved from running once at the start of the script to running at the beginning of the epoch loop.

**File: SeqTrack/lib/train/run\_training.py**

**Line ~57 (Store Seed):** Added settings.base\_seed = base\_seed to store the main seed number (the team number) in the settings object, making it accessible to the trainer.

**Lines ~42-46 (Remove Old Call):** The original block of code that called init\_seeds once at the beginning of the script was removed to prevent redundant seeding.

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

**Line ~8-9 (Add Imports):** Added import random and import numpy as np to the top of the file.

**Lines ~11-27 (Add Helper Function):** The init\_seeds(seed) function was copied into this file to make it locally available to the trainer.

**Lines ~94-101 (Call Seed Function in Loop):** Inside the train function, a new block was added at the very beginning of the for epoch in ...: loop. This block checks for the base\_seed in settings and calls init\_seeds() with that number. This action ensures that the random number generators for PyTorch, NumPy, and Python are reset to the exact same state at the start of every single epoch, fulfilling the requirement.

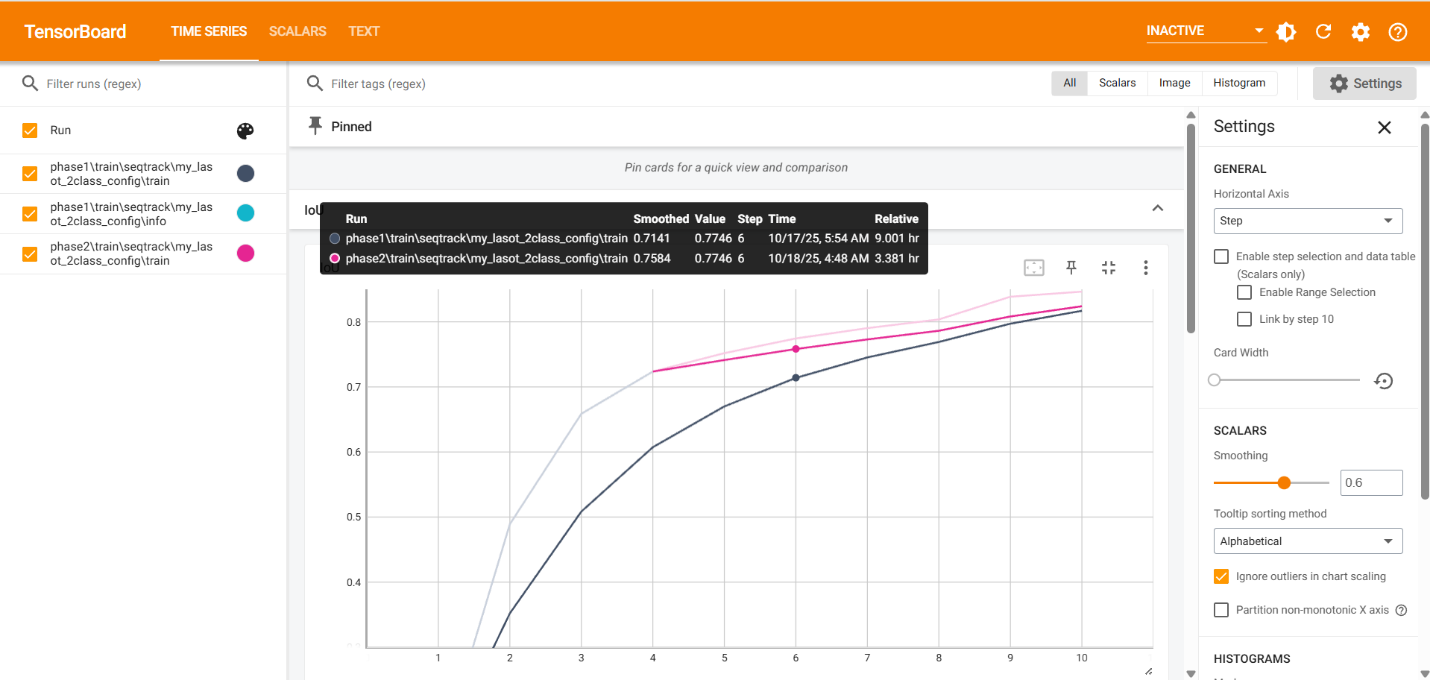
**Loss and IoU Graphs**

**IoU Graph:-**

A graph on a screen

AI-generated content may be incorrect.

This figure capture IoU curve for Phase 1 (full training) and Phase 2 (resumed from epoch 3).



This figure proof that the two phases have identical IoU values in epoch 7

**Loss Graph:-**

A graph on a screen

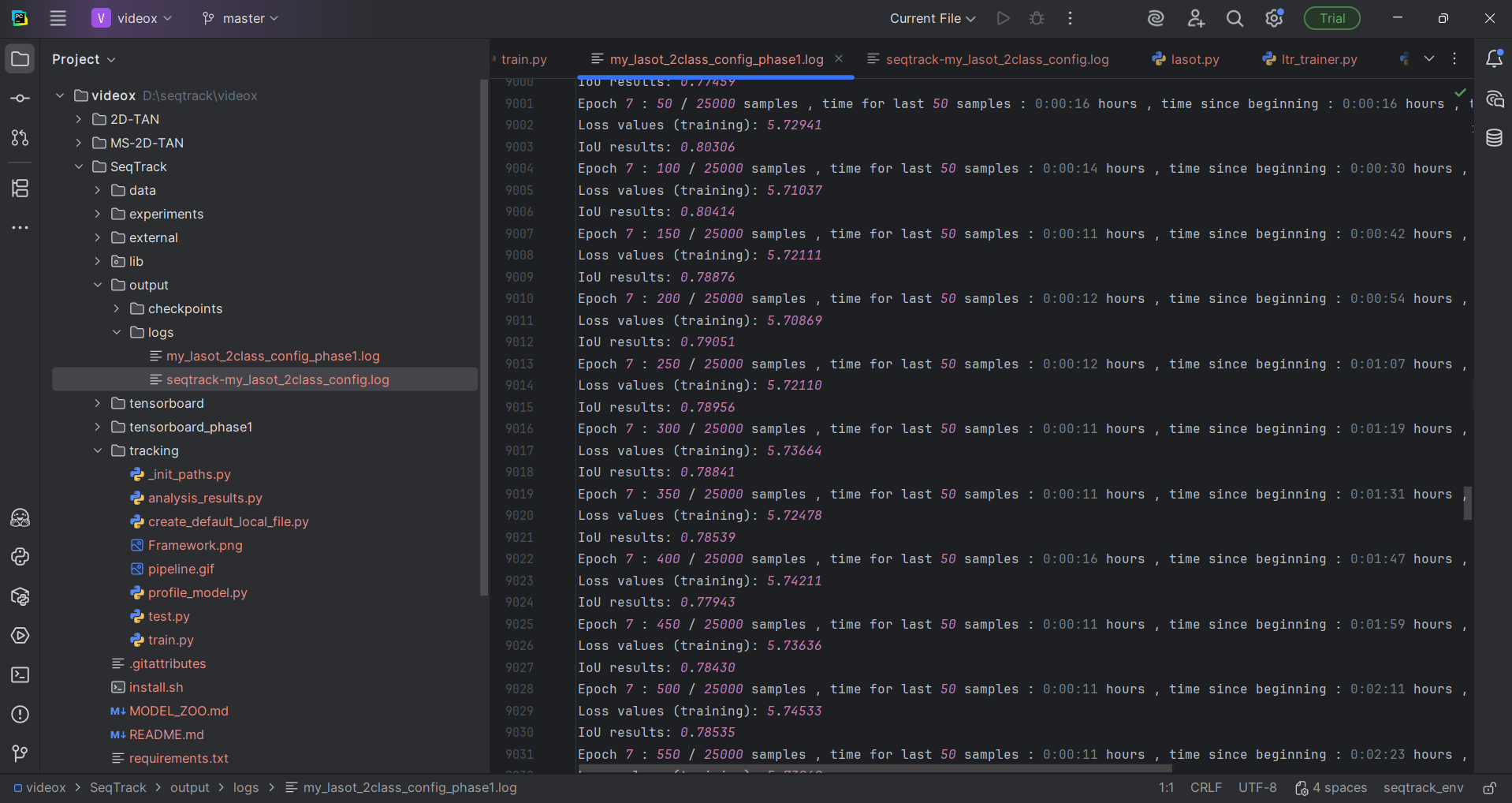
AI-generated content may be incorrect.

This figure capture Loss curve for Phase 1 (full training) and Phase 2 (resumed from epoch 3).

A screen shot of a graph

AI-generated content may be incorrect.

This figure proof that the two phases have identical loss values in epoch 7



A screenshot of a computer program

AI-generated content may be incorrect.

Those two captures from log files from phase 1 and phase 2 prove they are identical with the same loss and IoU values.