CS 490/548: Assignment 04

Programming Assignments (80%)

The goal of this assignment is to write code to **train and evaluate PyTorch neural network models to predict classes from a SUBSET of the HMDB51 dataset.**

You must implement at LEAST one approach.

***CS 490:* You must handle 2 classes (running and walking).**
***CS 548:* You must handle 4 classes (running, walking, falling, climbing stairs).**

Setup

You must install two additional packages to your python environment:

conda install conda-forge::unrar
pip install prettytable

Once you have done this, **you MUST run Prepare_A04.py ONCE**. This will download the data you need. It will ask you whether you want the undergraduate or graduate version. You CAN download both, but you only need one.

If you want the data to be stored somewhere OTHER than your project direction, change the value of **CORE_DATA_DIR** in Prepare_A04.

A04.py

You will write the following functions in this python script:

- **get_approach_names()**
- **get_approach_description(approach_name)**
- **get_data_transform(approach_name, training)**
- **get_batch_size(approach_name)**
- **create_model(approach_name, class_cnt)**
- **train_model(approach_name, model, device, train_dataloader, test_dataloader)**

Note that most of these functions rely on the approach_name parameter. This is how you will code different behavior per approach.

get_approach_names()

Returns a list of the names of all combinations you will be testing. It is largely up to you what names you use, but self-documenting names are encouraged.

A rather boring example: ["CNN0", "CNN1"]

get_approach_description(approach_name)

Given the approach_name, return a text description of what makes this approach distinct. This does not have to be very long: one sentence is sufficient.

get_data_transform(approach_name, training)

Given the approach_name and whether this is for training data, return the appropriate dataset transform. You may add data augmentation if you like for training data transforms, BUT:

- Do NOT do data augmentation for non-training data!

At minimum, you must perform the following:

(0.15) data_transform = v2.Compose([v2.ToImageTensor(), v2.ConvertImageDtype()])
(0.16) data_transform = v2.Compose([v2.ToImage(), v2.ToDtype(torch.float32, scale=True)])

get_batch_size(approach_name)

Given the approach_name, return the preferred batch size. This can be hardcoded to a single value if you do not have a preference, BUT this allows you to use a smaller batch size if your architecture is large enough where that is a concern.

create_model(approach_name, class_cnt)

Given the approach_name and output class_cnt, build and return a PyTorch neural network that takes a video of shape [batch, frames, channels, height, width] and outputs a vector with class_cnt elements.

You do not have to move this model to the GPU (that is done for you in the provided scripts).

Any of your networks should have the following attributes:

1. The number of frames per video are 30 frames.
2. The size of the image can be controlled by adding a v2.Resize() transformation to your data transforms.
3. The output layer should have class_cnt nodes.
4. The network must have at least 3 layers, not including the last output layer.
5. **You are free to use ANY of the layer types available in PyTorch**.
6. You may choose to use one of the existing pre-trained networks and fine-tune it.

train_model(approach_name, model, device, train_dataloader, test_dataloader)

Given the provided model, the device it is located, and the relevant dataloaders, train this model and return it.

A couple of caveats:

- **You can ONLY train on the data from train_dataloader.**
- You may print out how the network is doing on test_dataloader. **However, you may NOT use test_dataloader as a validation set** (e.g., you cannot use it to tune learning rates dynamically).
- You are free to choose any optimizer you wish.
- You are free to decide how many epochs of training you use.
- **Your loss SHOULD be CrossEntropyLoss.** You are, however, welcome to add other losses if you believe it will help.

Note that, in general, differences here do NOT count as acceptable approach differences, such as:

- **Epoch counts**
- **Different optimizers**

Acceptable Approach Differences

If you have more than one approach, your approaches should be sufficiently different from each other! Acceptable differences include:

- Different sizes of filters
- Notably different filter counts (32 vs. 64, not 32 vs. 33)
- Notably different node counts
- Consistently different activation functions.
- Adding 1 or more additional layers
- Adding skip connections
- With or without significant data augmentation (or sufficiently different variations of data augmentation)

I actively encourage you to look for examples of working networks. HOWEVER, if you do use code from another source, cite it in the body of the code like so:

 # Taken from: https://keras.io/getting-started/sequential-model-guide/

NOTE: Even if you cite it, you may NOT use code from another student (whether presently in the course or from a previous semester).

However, any code in the slides for this course is freely usable.

If you are unsure whether any variants you intend to experiment with are different enough, ASK ME.

TRAINING CAN BE VERY TIME-CONSUMING. DO NOT LEAVE THIS UNTIL LAST MINUTE!

Provided Files

Prepare_A04.py

- Asks you if you want the undergraduate or graduate version of the data.
- If the data folder already exists, asks whether you want to delete it and start over.
- Downloads the necessary RAR files.
- Unpacks the necessary RAR files and deletes any extra files.
    - Undergraduate folder size: ~233MB
    - Graduate folder size: ~296MB
- NOTE: It may need more intermediate space to download everything.

General_A04.py

- Contains all shared functions for training and evaluation scripts.

Train_A04.py

- Asks you if you want the undergraduate or graduate version of the data.
- Asks you which approach you wish to train.
- Loads the data.
- Trains the model.
- Saves the model to assign04/output

Eval_A04.py

- Asks you if you want the undergraduate or graduate version of the data.
- Asks you which approach you wish to evaluate (if -1, evaluates all).
- Evaluates the model(s) on training and testing data.
- Prints and saves the results for each approach to assign04/output with filename: <approach name>_RESULTS.txt or ALL_RESULTS.txt.

Output Files (20%)

You must run Eval_A04.py on all your approaches and get the ALL_RESULTS.txt file. You must commit this file to your repo.

Your model files (model_*.pth) are likely to be very large. ***If they are less than 10MB, you can (and should) commit them to your repo.***

If they are LARGER than 10MB:

- Add those files to your .gitignore
- Include a text file in your repo, **assign04/output/links_models.txt,** that contains links to download your trained models from some other location (e.g., OneDrive, Google drive, Dropbox, etc.)

BONUS POINTS

For the entire assignment, bonus points will be awarded to the submission that has the following:

- (+%5) Best **testing accuracy** in class
- (+%5) Best **testing F1 score** in the class
- (+%5) Experimenting with 3 variants or more

"Best in class" will be assessed separately for undergraduate and graduate students.

Submission

Make sure the following is committed on your repo:

- **A04.py**
- **assign04/output/ALL_RESULTS.txt**
- **assign04/output/links_models.txt**
- **Models smaller than 10MB**

PLEASE add the following to your .gitignore file:

- data/
    - This is where the HMDB data is downloaded by default
- assign04/output/bigmodel.pth
    - Any models over 10MB

Grading

- 80% Programming
- 20% Results Folder