Introduction to Deep Learning in Computer Vision

Project 4.2: Video classification

In this project you will be classifying videos according to their actions. You will work on a subset of the ucf-101 dataset for video action recognition derived from this Kaggle version of UCF-101. You can find the designated dataset as well as the file `datasets.py` on Learn.

The dataset and provided tools

The provided dataset includes 720 videos (500/120/120 for train/val/test) of 10 balanced classes related to workout, mode precisely including the classes ['BodyWeightSquats', 'HandstandPushups', 'HandstandWalking', 'JumpingJack', 'JumpRope', 'Lunges', 'PullUps', 'PushUps', 'TrampolineJumping', 'WallPushups']

BodyWeightSquats

HandstandPushups

HandstandWalkin

JumpingJack

JumpRope

Lunges

PullUps

PushUps

TrampolineJumping

WallPushups

For each video, we provide:

- 10 uniformly sampled frames
- csv files which include the video name, relative path, actions, and labels.

Moreover, we provide example dataset classes and dataloaders in `datasets.py` to help you get started with both 2D or 3D modeling. You may find the dataset of `frames` useful for training per-frame models and the `videos` dataset useful either for testing on videos or training early/late fusion models (if stack_frames=True returns the sampled frames are stacked as [C, T, H, W], else as a list).

Your tasks

Task from project 4.1

In this first part of the project, we your task is to explore different models for video classification, namely:

- aggregation of per-frame models
- late fusion
- early fusion

… all for 2D CNN models, along with

- 3D CNNs.

Please report your comparative results in your poster. To work with the models – in particular, the 3D CNN – you may have to subsample the images. Please include any such details in the poster and discuss the potential effects thereof.

Next week, you will receive additional tasks based on the associated lecture.

New tasks: Project 4.2

Task 4.2.1: Information leakage between train/val/test splits

Your first task will address the problem of **information leakage between training/validation/test splits**. As you already know, you should not repeat images between splits, and you should only use the test set for final testing – not for model selection.

However, what might be less evident is that you **need to sample your splits independently** from the data distribution. In particular, when your data comes from humans, different instances of the same human are often closely correlated and you therefore should take care **not to include the same subject in multiple splits**. Splits must be created on individual-level to ensure accurate evaluation.

The dataset you analyzed in Project 4.1 was retrieved, splits included, from Kaggle. However, the Kaggle splits were different from the original UCF101 splits, and in particular there was leakage between the Kaggle version's splits: Different splits included videos of the same person carrying out a given action. In the first part of project 4.2 we therefore ask you to select one of your well-performing models from project 4.1 and retrain the model using the updated dataset `ucf101_noleakage`, which you can now find on `/dtu/datasets1/02516` both as a directory and as a zip file (in case you want to download it elsewhere).

Please report both your performance on the splits with leakage from Project 4.1, as well as your updated performance when splits do not have leakage. It's enough to observe the difference for one model, but please select one that performed well with the leakage-affected splits.

Task 4.2.2: Dual-stream networks

Now that you have most likely observed a drop in performance evaluating on splits without leakage, let's try a more powerful model: Dual-stream networks that include optical flow as input:

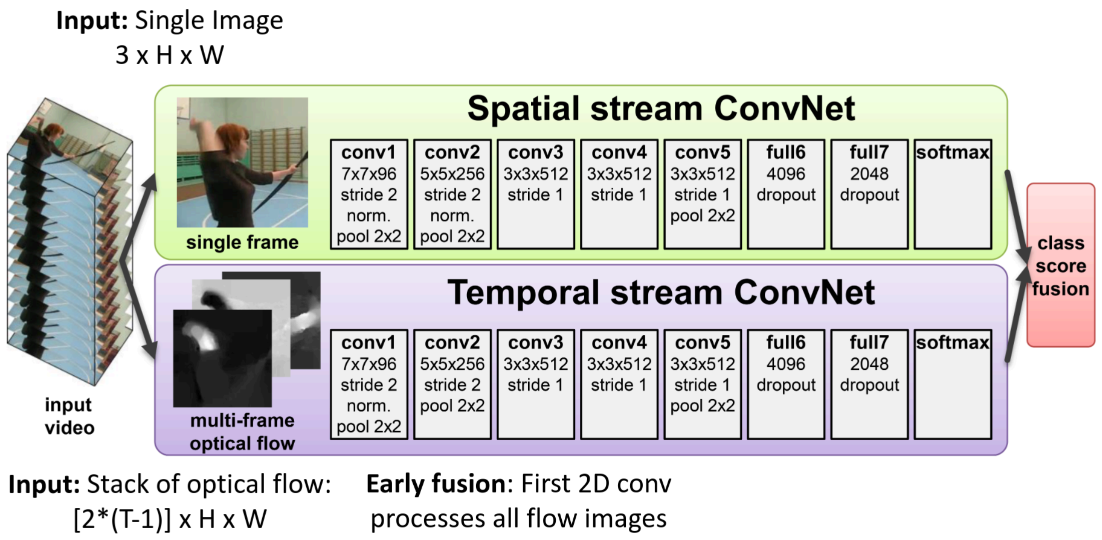

Figure from slides by Justin Johnson.

To support you in this endeavor, the updated dataset `ucf101_noleakage` found on `/dtu/datasets1/02516` contains pre-computed optical flows between the 10 uniformly sampled frames. This optical flow has been generated using the RAFT model[1]. You can inspect the optical flows in the `flows_png` folder, an example is also shown below:

[1] RAFT: https://arxiv.org/abs/2003.12039, https://pytorch.org/vision/0.12/generated/torchvision.models.optical_flow.raft_large.html

To train the temporal stream model you need to stack the 2-channel optical flows that are provided in the ‘flows’ folder as ‘npy’ files. To help you get started we provide a new dataset class ‘FlowVideoDataset’ in the updated `datasets.py`.

Please implement and train the dual-stream network and compare its performance to what you saw in task 4.2.1.

Again, your project and results should be summarized on a poster to be presented December 4, with submission deadline Tuesday 3.12 at 22:00.