First-Person Action Recognition

TA: Mirco Planamente

OVERVIEW

The goal of the project is to become familiar with the task of action recognition. The student should understand the general approach to tackle the video recognition task and the problems that they may have dealing with the first-person data. Before starting, the student should read [1] in order to get familiar with the following terms: Optical Flow, Recurrent Neural Network (RNN) and Class Activation Maps (CAM). The student should be able to replicate some experiments proposed in [1]. As the next step, the student should implement the self-supervised task as described in [2]. For the last part of the project, the student should propose a variation for the project, taking inspiration from a set of possible ideas.

GOALS

1. Read [1,2] and get familiar with Optical Flow, Recurrent Neural Network (RNN), Class Activation Maps (CAM) and Self-Supervision.
2. Replicate some experiments proposed in [1] and [2]
3. Propose, implement and test a variation for the project.

REQUIREMENTS

For this project, you can assume, for simplicity, that your validation set is the same as the test set, therefore:

Dataset: GTEA61 GTEA61.zip

Training sets = S1 S3 S4 (labeled)

Validation set = test set = S2.

- Complete the table below using the same parameters of [1] (run these experiments with fewer than 25 frames, for example 7 - 16 for RGB and only 5 for WarpFlow)

Configurations	**Accuracy%** 7 Frames	**Accuracy%** 16 Frames
ConvLSTM		

ConvLSTM-attention		
Temporal-warp flow		
two-stream (joint train)		

- Implement the self-supervised task proposed in [2] and integrate it on the second stage of the Ego-RNN network (with the same training procedure proposed in [1]).
- CAM visualization like in [1] when the self-supervised task is added.
- Implement the same self-supervised task as a regression problem. This variant requires you to choose a correct activation function for the regressor and a suitable loss function for the problem.
- Run an appropriate hyperparameter optimization for the self-supervised task: for each method you have to choose at least 3 different sets of hyperparameters.
- Choose a variation of the project as described in the following VARIATIONS section.
- Deliver PyTorch scripts for all the required steps.
- Write a complete pdf report. The report should contain a brief introduction, a related works section, a methodological section for describing the algorithms you are going to use, an experimental section with all the results and discussions. End the report with a brief conclusion.

VARIATIONS

You can refer to this section for ideas of possible variations of the project.

Optical Flow colorization

The idea is to extend the concept of depth-colorization to optical flow. You can start to read this work [3] to understand the concept. Feel free to change the architecture of the DECO module in order to better learn the colorization. The goal is to learn a mapping/colorization that transforms the 2-channel optical flow (in your case warp flow) in three-channel, in order to use the same architecture that you use for RGB frames.

A different Self-Supervised task

You can choose to implement a different self-supervised task.

You can refer to a list of self-supervised tasks and implementations in the last paragraph. If this variation is chosen, you have to repeat hyperparameter optimization also for the new task.

Exploiting Optical flow like Attention on RGB

The main idea is to exploit the optical flow information to perform attention on the RGB information in order to help the Appearance Network. There are many ways to implement this idea.

DETAILS

Details and useful information for completing the project follow.

Before starting

Before starting it is mandatory to take time to familiarize yourself with Optical Flow, Recurrent Neural Network (RNN), Class Activation Maps (CAM) and Self-Supervision, otherwise the project will be extremely difficult. It is also important to understand the difference between RGB and optical flow information in terms of meaning, dimensions and network used to process both. You should understand how works the Recurrent Neural Network(RNN) in particular the Long Short-Term Memory (LSTM) and its variation with the convolutions. If you want, you can start from [https://github.com/swathikirans/ego-rnn]. This is the GitHub repository of Ego-RNN. You can use all the python scripts in the repository but it's extremely important that you not use the three model .pth that the authors make available. You have to train your network from scratch (obviously for the backbone you have to start from Imagenet pre-train model).

The datasets

Dataset: GTEA61

You can find and download the dataset at this link. [GTEA61.zip]

Each video of the dataset is already divided into frames, scaled at 256 with the extracted warp flow information. You can also find the Ground-Truth motion maps to implement the self-supervised task [2].

Inside the dataset you can find 3 folders: **processed_frames, flow_x_processed** and **flow_y_processed**

- gtea_61/processed_frames2 ➜ RGB frames and Ground-Truth motion maps
- gtea_61/flow_x_processed and flow_y_processed ➜ x and y Warp Flow data.

The general path for RGB frames/ground-truth motion maps is:

- “Subject_N/Action/Sample/rgb” or “mmaps” (The general path for warp flow is the same, without the last folder)

Questions you should be able to answer at the end of the project.

What are the differences between RGB and Optical Flow network?

How does it work the attention mechanism?

Why in this work [1] do they use a ConvLSTM and how does it work?

Which is the motivation behind the use of the self-supervised task in [2]?

What are the differences between Optical Flow and Warp Flow?

What are the main limitations of this method?

What are the main limitations on working with videos?

Hyperparameter optimization

You can start with the hyperparameters declared in the paper and optimize them further to get better results. Good hyperparameters to choose for optimization are: learning rate, batch size, number of frames, number of epochs, weight decay. You need also to optimize the weight of the loss of the auxiliary task.

Additional useful resources

- Implementations of popular self-supervised algorithms available at:
  - https://github.com/jason718/awesome-self-supervised-learning
- Learning features by watching objects move.
  - https://arxiv.org/pdf/1612.06370.pdf
- More info about the dataset
  - http://cbs.ic.gatech.edu/fpv/

Hint on Self-supervised block:

General idea of self-supervised block implementation:

feat (BS,512x7x7) --> relu() = feat (BS,512x7x7)
feat (BS,512x7x7) --> CONV[100, k=1, p=0] = feat(BS,100x7x7)
*feat(BS,100x7x7) --> feat(BS,100*7*7)*
*feat(BS,100*7*7) --> FC[100*7*7, 2*7*7] = feat(BS, 2*7*7)*

*feat(BS, 2*7*7) --> feat(BS, 2x7x7)*
cross_entropy(feat, maps)

BS is the batch size
100*7*7 *= 4900*
2*7*7 *= 98*
Feel free to prove some variation of this implementation.
(If you have already implemented it in different way, don't worry)

References:

[1] Attention is All We Need: Nailing Down Object-centric Attention for Egocentric Activity Recognition https://arxiv.org/pdf/1807.11794.pdf BMVC18

[2] Joint Encoding of Appearance and Motion Features with Self-supervision for First Person Action Recognition https://arxiv.org/pdf/2002.03982.pdf

[3] (DE)^2CO: Deep Depth Colorization https://arxiv.org/pdf/1703.10881.pdf ICRA18