Video Analytics - Exercise Sheet 02 - Convolutional Neural Networks for Action Recognition

Deadline: 01.06.2021 - 23:55

Exercise Class: 04.06.2021 - 12:15

Summary

In this exercise, we will focus on using Convolutional Neural Networks (CNNs) for action recognition, i.e. classification of the action being performed in a short video.

The dataset used is a subset of the UCF101 [1] dataset with 25 classes which we call *miniUCF*. The train-validation split is already provided as well as pre-computed optical flows. RGB frames have to be extracted from the AVI files.

You should implement 2 different models for action segmentation: TSN [2] and 3D-ResNet [3]. The input to each can be RGB frames or optical flow. Each network can also be randomly initialized or initialized from ImageNet.

miniUCF

The first 25 classes of the UCF101 [1] dataset have been selected for this exercise sheet. The dataset is provided as 2 zip files. One containing the original AVI videos which you can use to extract the RGB frames. And another one containing pre-computed optical flows. Alongside the video files, 3 `.txt` files are also provided containing the following information:

- `classes.txt`: Contains the class id, class name mapping.

- `train.txt`: Contains the list of videos that should be used for training.
- `validation.txt`: Contains the list of videos that should be used for evaluation.

Each line in `train.txt` or `validation.txt` is a *video identifier*, something like

```
CLASS_NAME/VIDEO_NAME
```

Where `CLASS_NAME` is the name of the class that the video belongs to and you can find the mapping to class id in `classes.txt` (e.g. `ApplyEyeMakeup`). And where `VIDEO_NAME` is the name of the video (e.g. `v_ApplyEyeMakeup_g08_c01`).

Using the video identifier one can access the video AVI file by accessing:

```
CLASS_NAME/VIDEO_NAME.avi
```

relative into the root folder in which you extract the video zip files.

The flow files for each video identifier can be accessed from

```
CLASS_NAME/VIDEO_NAME/flow_x_0001.jpg
CLASS_NAME/VIDEO_NAME/flow_x_0002.jpg
...
CLASS_NAME/VIDEO_NAME/flow_x_N.jpg
```

and

```
CLASS_NAME/VIDEO_NAME/flow_y_0001.jpg
CLASS_NAME/VIDEO_NAME/flow_y_0002.jpg
...
CLASS_NAME/VIDEO_NAME/flow_y_N.jpg
```

where `N` is the maximum number of frames in each video. Notice that the flow in each direction is saved as a compressed single-channel `.jpg` file which you can use OpenCV to load.

Requirements

Use Python3.

You should use PyTorch >= 1.1 and `torchvision`.

Deliverable

Please consider writing clear and well-documented code. The easier to read and understand your code is, the lower the probability of incorrectly getting a lower grade than you deserve.

Please submit your code in the following requested format. This will make grading much easier for me.

Have 2 folders: `TSN` and `3DResNet`. Inside each includes a file named `dataset.py` for implementation of the PyTorch `Dataset` that you will use to load the model. This could potentially be 2 classes, 1 for RGB frames and 1 for optical flow or 1 class that takes the modality as an initializer argument.

Also, include a file named `model.py` for implementation of the specific model as a PyTorch `nn.Module`.

Include a file named `main.py` that by running it I can reproduce the training, evaluation part of each exercise. In this file, you should import and use the dataset and model from `dataset.py` and `model.py`.

Finally include a file `report.pdf` that you describe the results you obtained, training and validation loss figures as the training progress goes on, final classification accuracy, etc. Training and validation loss should be calculated as the average training and validation loss for each epoch.

Other than the `.py` files (potentially you could have additional python files) and the `report.pdf` please don't submit anything else, especially please don't submit the dataset as it is very large.

1. TSN

(10 points)

1.1 RGB TSN

Implement a TSN network only for RGB signals with a [ResNet-18 backbone] which is initialized from ImageNet. Use the following hyper-parameters:

1. Number of segments = 4 or 8.
2. During training randomly select the snippet inside each segment.
3. During testing select the middle snippet inside each segment.
4. Use the same input size and input preprocessing for the RGB frames.

1.2. Optical Flow TSN

Implement a TSN network only for optical flow. Try 2 settings: initialization from ImageNet (how should one initialize the first layer?), initialize randomly. Compare the training and validation loss figures of the 2 settings. (Between the 2 settings only change the kind of initialization, keep all other hyper-parameters the same)

Use the following hyper-parameters:

1. Number of segments = 4 or 8. (keep the same as RGB)
2. Use a stack of 5 consecutive x,y flows as input. This results in a 10 channel input (as compared to the RGB settings that you have 3 channel input)

1.3. RGB vs. Optical Flow and Fusion

Please compare the performance of RGB and Optical Flow models for each class.

Use late fusion (averaging the probability estimate of RGB and optical flow models at test time) to obtain a single prediction from the 2 models trained on the 2 different modalities. Can we achieve higher performance?

2. 3D ResNet

(10 points)

Note: You are not allowed to use the 3D ResNet available in PyTorch or its weights.

2.1. RGB 3D ResNet

Implement the 3D equivalent of the ResNet-18 backbone. Compare initializing it using weights using **inflation** [4] from the equivalent 2D model and initializing it randomly (the default PyTorch initialization).

To prevent overfitting we need augmentation. Use spatial random crop augmentation and well as temporal random cropping during training. At test time, perform multi-view testing (4 temporal views is enough, usually more views result in higher performance).

3. *Extra

If you wanted to improve the performance of action recognition on the provided dataset, beyond what is mentioned in this exercise sheet, what would you do? Please share potential ideas for improvement and the reasoning behind it.

References

[1] Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." *arXiv preprint arXiv:1212.0402* (2012).

[2] Wang, Limin, et al. "Temporal segment networks: Towards good practices for deep action recognition." *European Conference on Computer Vision*. 2016.

[3] Hara, Kensho, Hirokatsu Kataoka, and Yutaka Satoh. "Learning spatio-temporal features with 3d residual networks for action recognition." *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2017.

[4] Carreira, Joao, and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

Questions

If you have questions or find any mistakes or ambiguities, please contact Yaser Souri souri@iai.uni-bonn.de.