Exercise Sheet 03
MA-INF 2218 Video Analytics SS25
Action Segmentation with TCNs

May 29, 2025

Deadline: 13.06.2025 - 23:59
Source code and data: Download Links
Solutions: submit solutions to araujo@em.uni-frankfurt.de with subject *"VA - Exercise Sheet 03 - Last Name, First Name"*

In this sheet, you will implement an action segmentation model using temporal convolutional networks (TCNs). Given an input video $\mathbf{x}_1^T = (x_1, \ldots, x_T)$ with T frames, the goal of action segmentation is to predict the action label for each frame $\mathbf{c}_1^T = (c_1, \ldots, c_T)$, where c_t is the action label for frame t. Your system for this sheet will be based on the model described in [1]. The input $\mathbf{x}_1^T$ is also the same frame-wise I3D features that are used in [1], where $x_t \in \mathbb{R}^{2048}$. Your implementation should be using Python 3 and PyTorch >= 1.1.

Download the training data using the provided link in the `README` file and place it in the same directory of your `main.py` file. For details on the features and annotations, please refer to the `README` file.

You are going to report frame-wise accuracy (*MoF*), *Edit*, and *F1*@{10, 25, 50} scores (more info on these metrics can be found at [1]). For details on how to prepare the data for evaluation look at the `eval.py`.

1. Implement a single-stage TCN as described in Section 3.1 in [1]. However, replace the dilation factors with a dilation factor that increases linearly with the number of layers (*i.e.* 1, 2, 3, 4, 5, ...)

   - Train a network with 10 layers and 64 filters in each layer for 50 epochs with a cross-entropy loss. Use Adam optimizer with a learning rate 0.001 and a batch size of 4.
   - Evaluate the trained model on the test set and report the frame-wise accuracy, edit distance and F1-scores. Use the provided helper functions for evaluation.
   - Why, in your opinion, the performances are worst in this case compared with using the [1] dilation factor?

   (6 Points)

2. Implement a multi-stage TCN as described in Section 3.2 in [1] using the single-stage TCN from Question 1. However, instead of passing only the predicted probabilities to the next stage, concatenate the predicted probabilities with the input features before passing it to the next one.

   - Train a multi-stage model with 4 stages for 50 epochs with a cross-entropy loss. Use Adam optimizer with a learning rate 0.001 and a batch size of 4.
   - Evaluate the trained model on the test set and report the frame-wise accuracy, edit distance and F1-scores.

   (4 points)

3. Repeat Question 2 with an additional video-level loss. The new loss computes the binary cross-entropy between a multi-class video level prediction and a multi-class target that indicates which classes are present in the video. The target is a vector with a dimension equals the number of classes in the dataset. The *i-th* element of this vector is 1 if the *i-th* class is present in the video, otherwise it should be 0. To get the video level prediction apply a max pooling on the temporal dimension of the predicted frame-wise logits. Note that the final loss function is the sum of the video level loss and the frame-wise cross-entropy loss that is used in Question 2.

   (4 points)

4. Implement a multi-scale model using the single-stage TCN with 10 layers from Question 1. The model should contain three parallel branches, where each branch is a single-stage TCN that receives a downsampled version of the input features and predicts a downsampled version of the frame-wise labels. The first branch operates on the full temporal resolution whereas the other branches operate on a downsampled version with factors 4 and 8 respectively. To get the final prediction, upsample the output of the last dilated layer from each branch (the output before the last 1×1 convolution layer, *i.e.* the classification layer) and pass the average of these outputs to another 1×1 convolution that predicts the logits for the output classes. Train your model using cross-entropy loss on the final output and on the output of each branch as well. Report the results on the test set.

   (6 points)

Important remarks

Exclude the `data` folder from your submission due to its size. However, submit any file you created yourself. Poorly written and not-commented code will be **penalized**. Please submit your code in the following requested format. This will make grading much easier for me. You should have at least these files:

- `model.py` file for implementation of the specific models as PyTorch *nn.Module*
- `dataset.py` file for loading the data
- `eval.py` and `metrics.py` provided files for evaluation, that you should complete
- `main.py` file

- `report.pdf` file, that contains a table that summarizes the results from all the questions

If you have any questions, feel free to contact me (**araujo@em.uni-frankfurt.de**).

[1] Y. Abu Farha and J. Gall, **MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation**, CVPR 2019.