DLCV HW4

TAs (劉致廷,吳致緯,劉彥廷)
ntudlcvta2019@gmail.com

吳致緯
Mon. 13:30 ~ 15:30
BL-421

劉致廷
Tue. 15:00 ~ 17:00
BL-421

劉彥廷
Wed. 15:30 ~ 17:30
BL-527

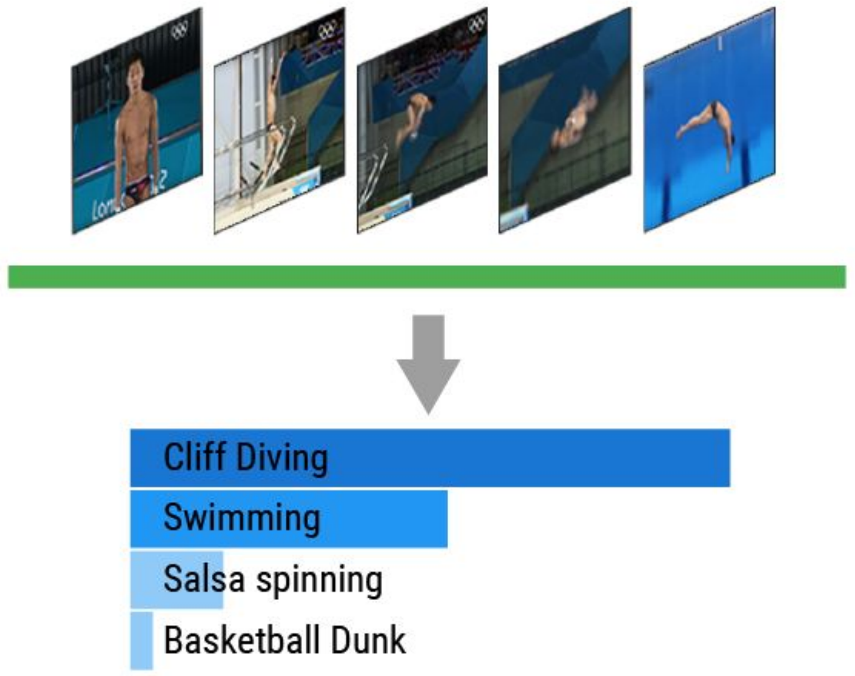
2019/05/15

Goal

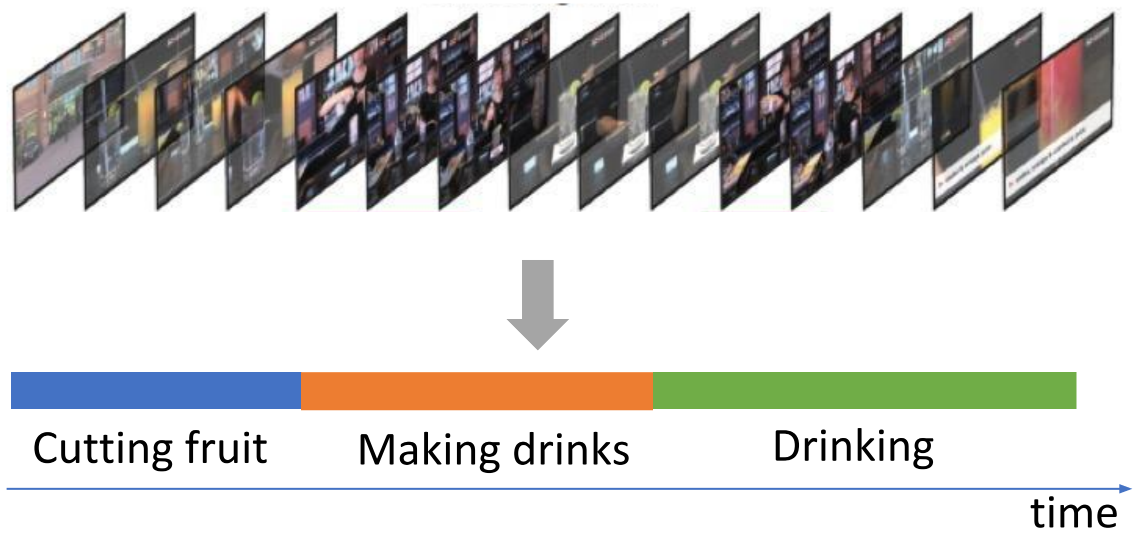
- Ability to extract state-of-the-art deep CNN features
- Implement recurrent neural networks (RNN) for action recognition
- Extend RNN models for solving sequence-to-sequence problems

Task Description

- In this assignment, you will learn to perform both trimmed action recognition and temporal action segmentation in full-length videos.



Trimmed action recognition



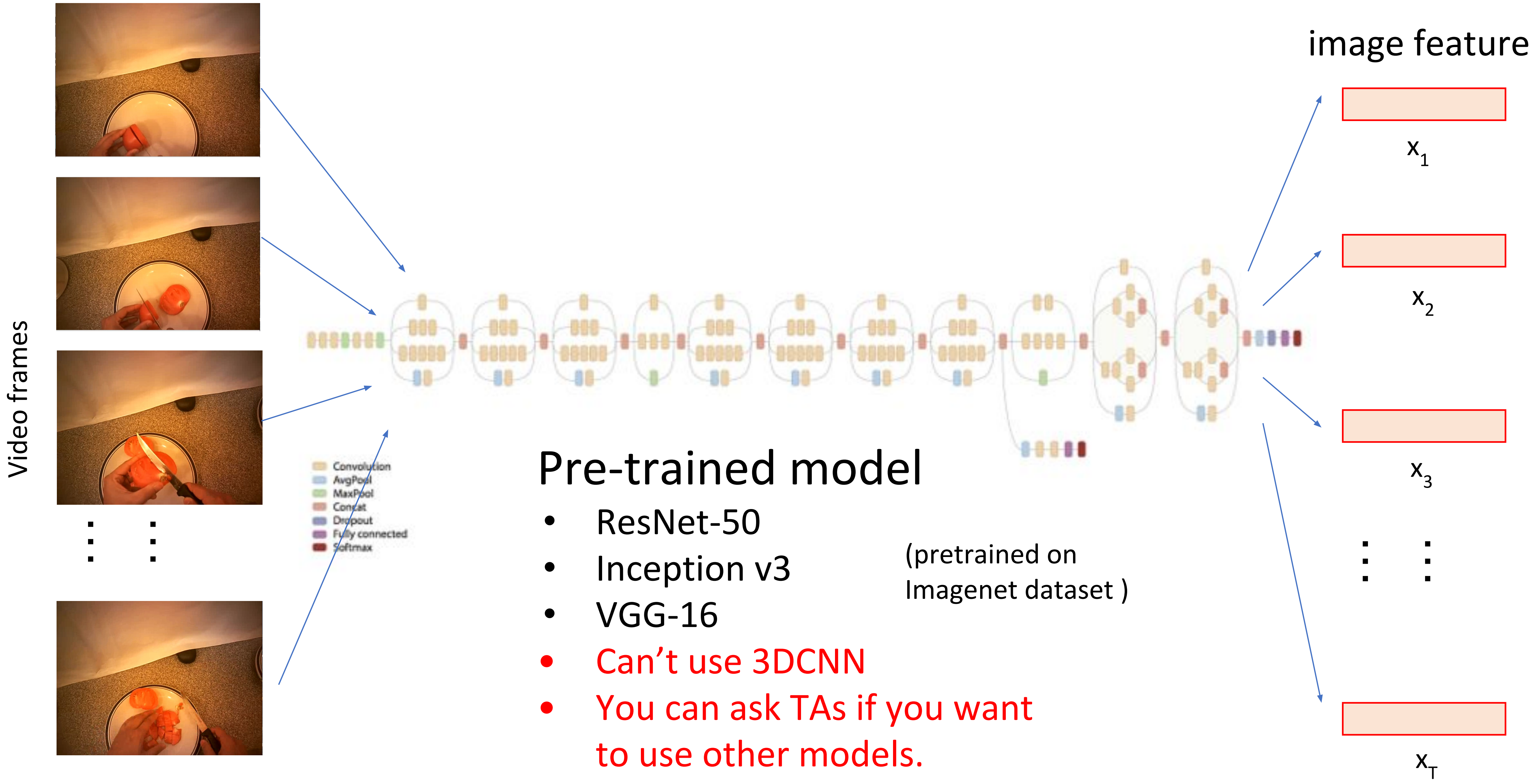
Temporal action segmentation

Task Description

- Task 1 : Data preprocessing
  - Extract state-of-the-art CNN features for action recognition
- Task 2 : Trimmed action recognition
  - Training your RNN model with sequences of CNN features and labels
- Task 3 : Temporal action segmentation
  - Extend your RNN model for sequence-to-sequence prediction

Task Description

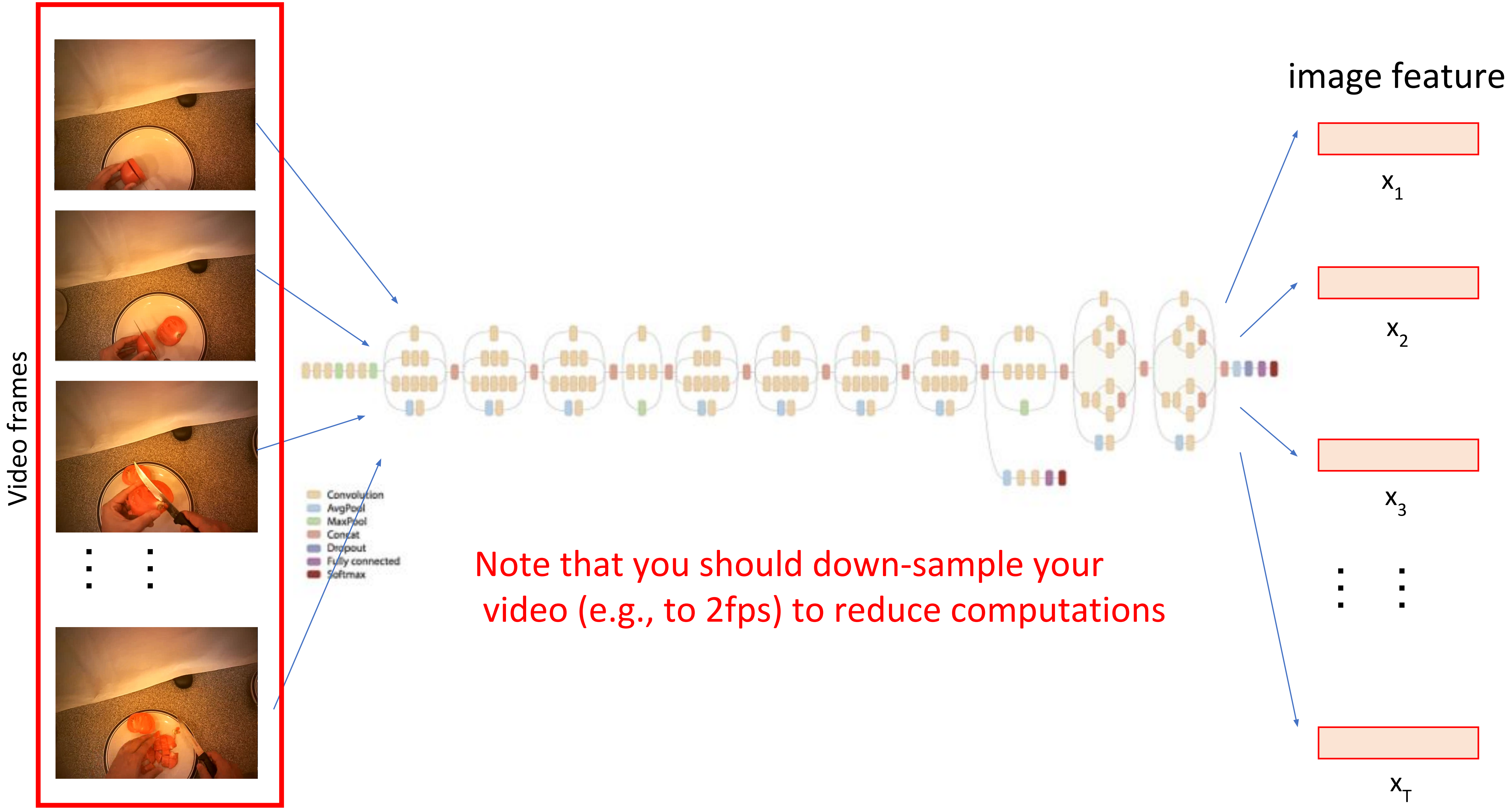
- Task 1 : Data preprocessing
  - Extract state-of-the-art CNN features for action recognition
- Task 2 : Trimmed action recognition
  - Training your RNN model with sequences of CNN features and labels
- Task 3 : Temporal action segmentation
  - Extend your RNN model to achieve sequence-to-sequence prediction



Pre-trained model

- ResNet-50
- Inception v3
- VGG-16
- Can’t use 3DCNN
- You can ask TAs if you want to use other models.

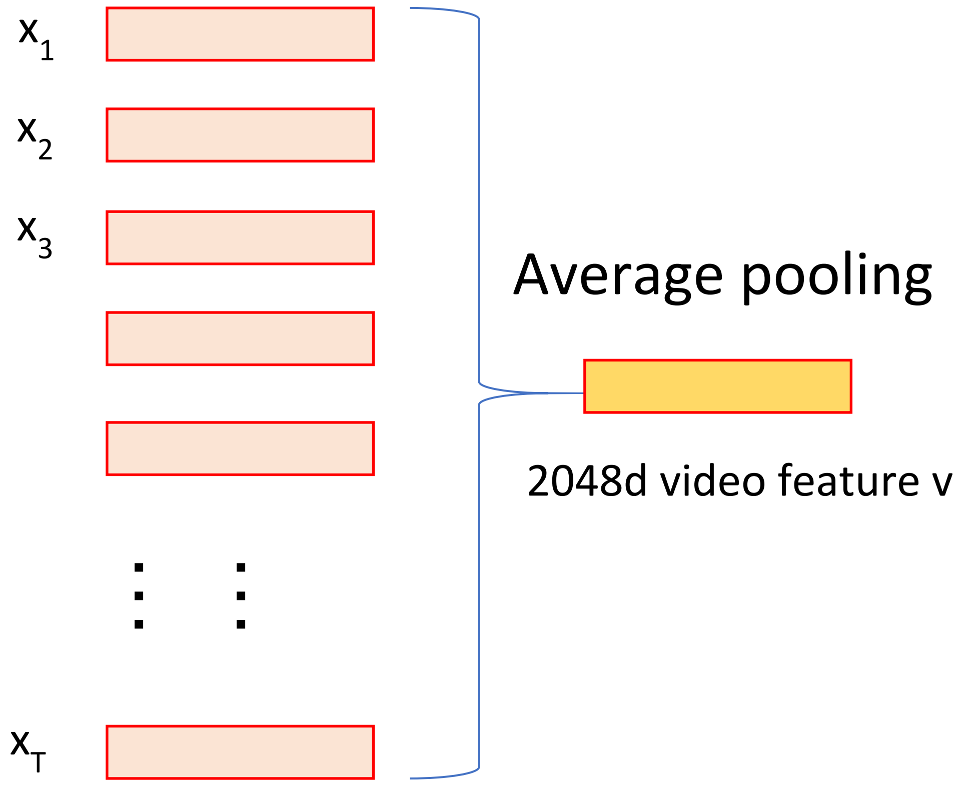
(pretrained on Imagenet dataset)



Note that you should down-sample your video (e.g., to 2fps) to reduce computations

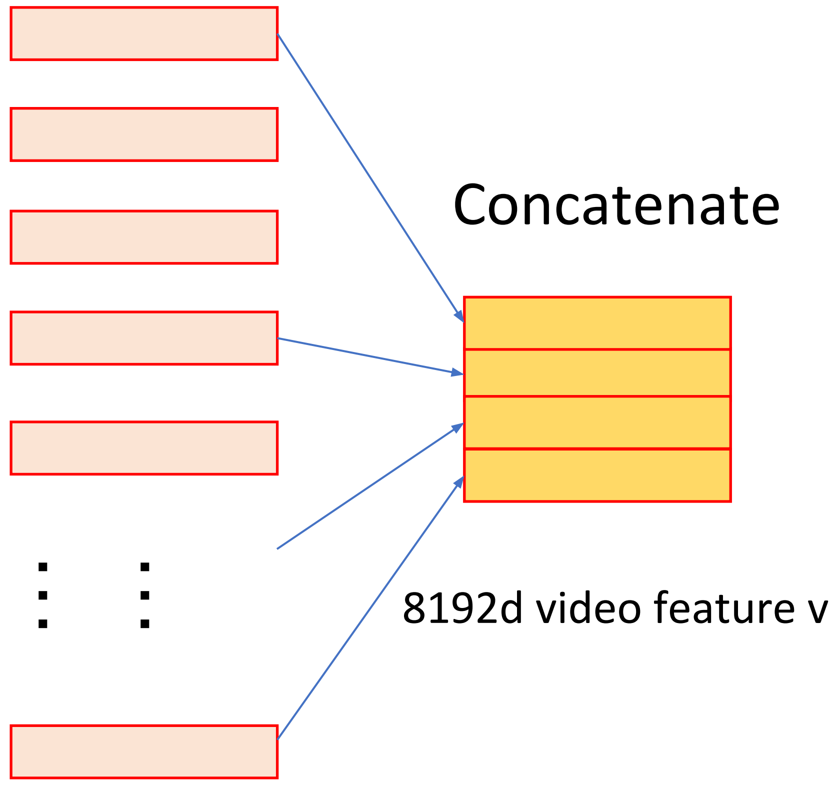
CNN-based video features

Strategy 1: Average pooling across all video frames



2048d image feature x

Strategy 2: Sample & concatenate selected frames, e.g., the first, middle and last one.

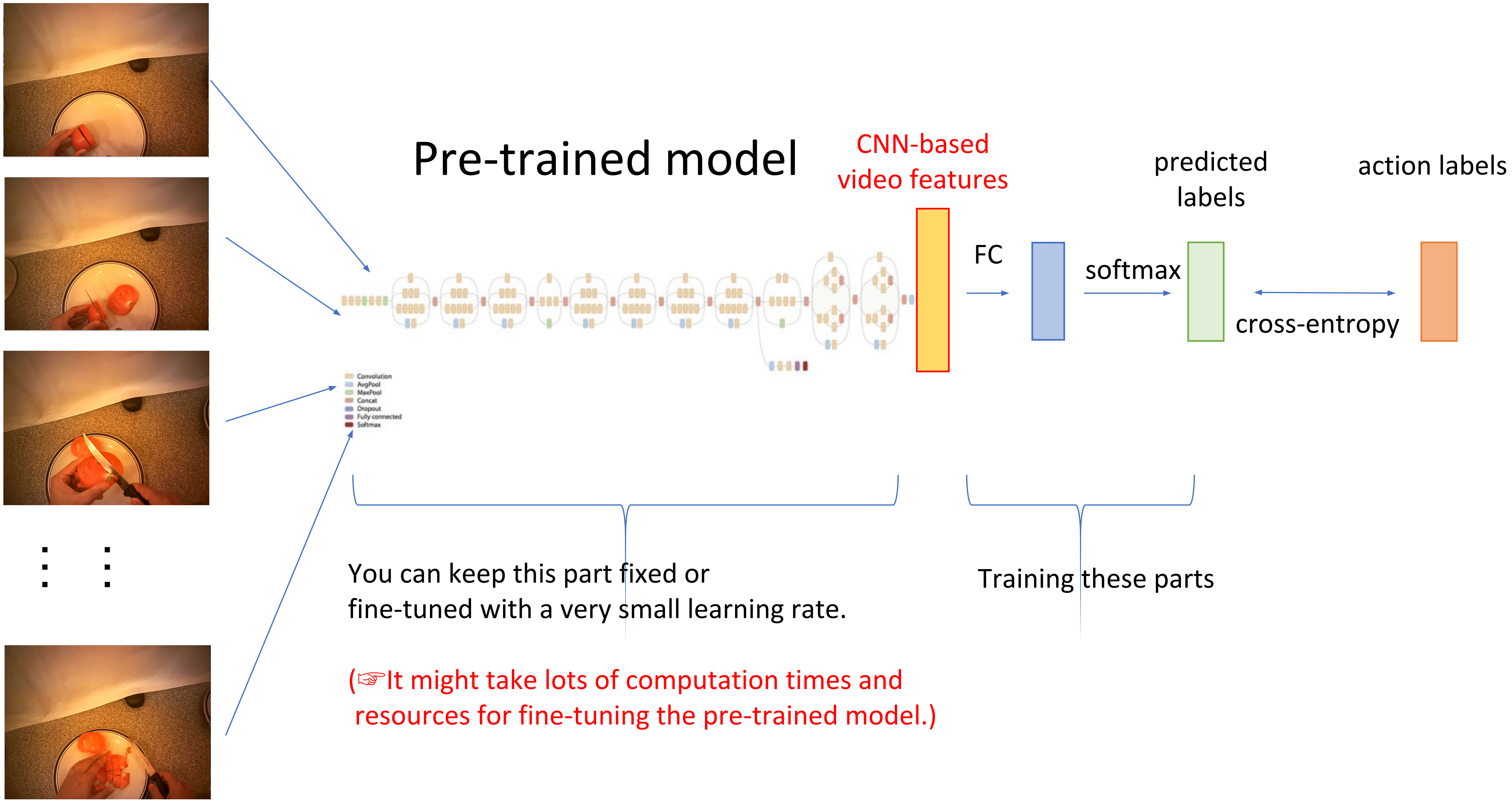


2048d image feature x

CNN-based video features

- You are welcome to design your own feature selection strategies. (Please provide details in your report.)

- Some common preprocessing techniques
  - Average pooling
  - Concatenate
  - Fusion (Sum up image features by some weights)
  - Dimension reduction (PCA, etc.)
  - You cannot use RNN in this part



You can keep this part fixed or
fine-tuned with a very small learning rate.

Training these parts

(☞It might take lots of computation times and
resources for fine-tuning the pre-trained model.)

Task Description

- Task 1 : Data preprocessing
  - Extract state-of-the-art CNN features for action recognition
- Task 2 : Trimmed action recognition
  - Training your RNN model with sequences of CNN features and labels
- Task 3 : Temporal action segmentation
  - Extend your RNN model to achieve sequence-to-sequence prediction

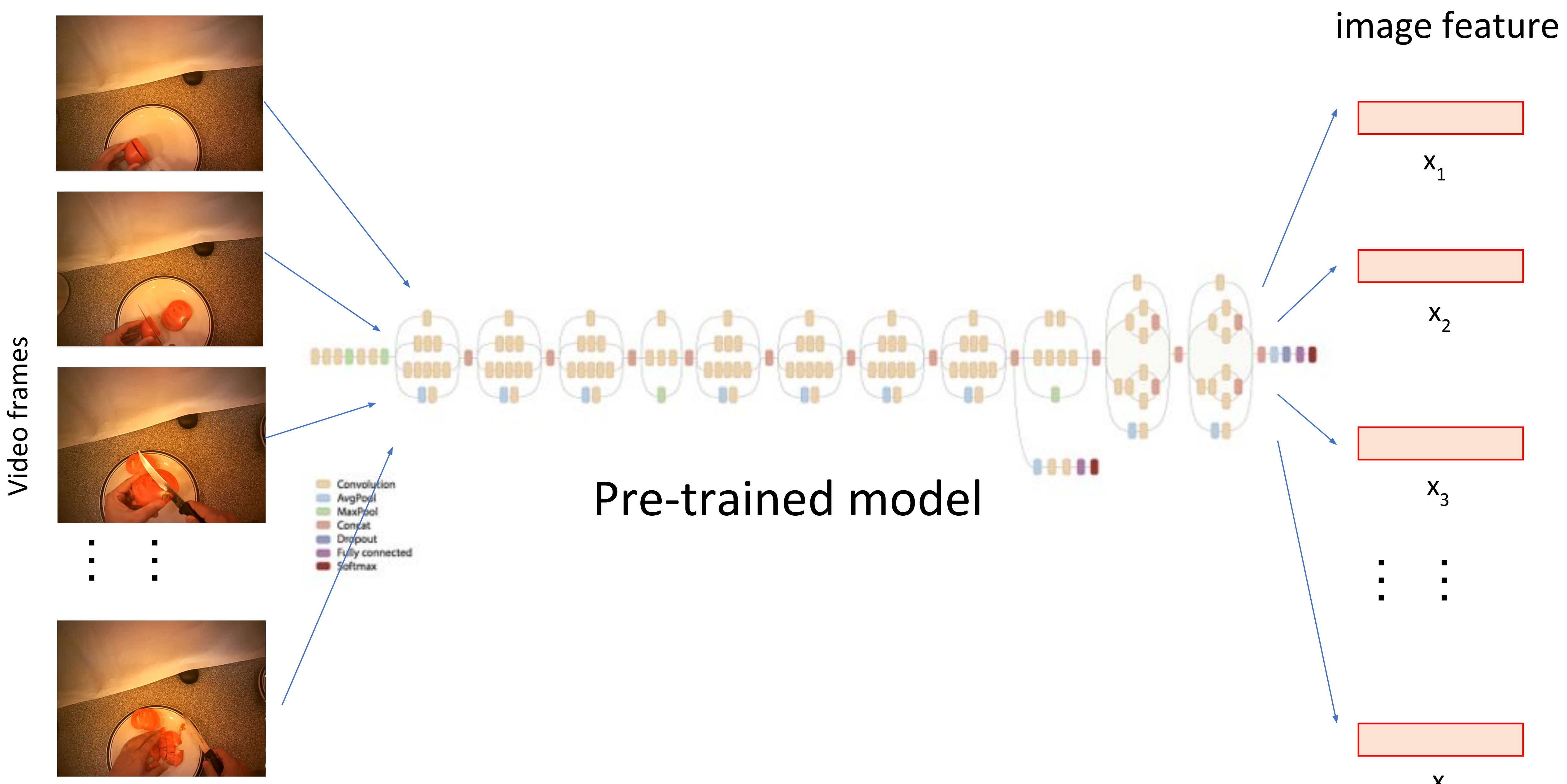
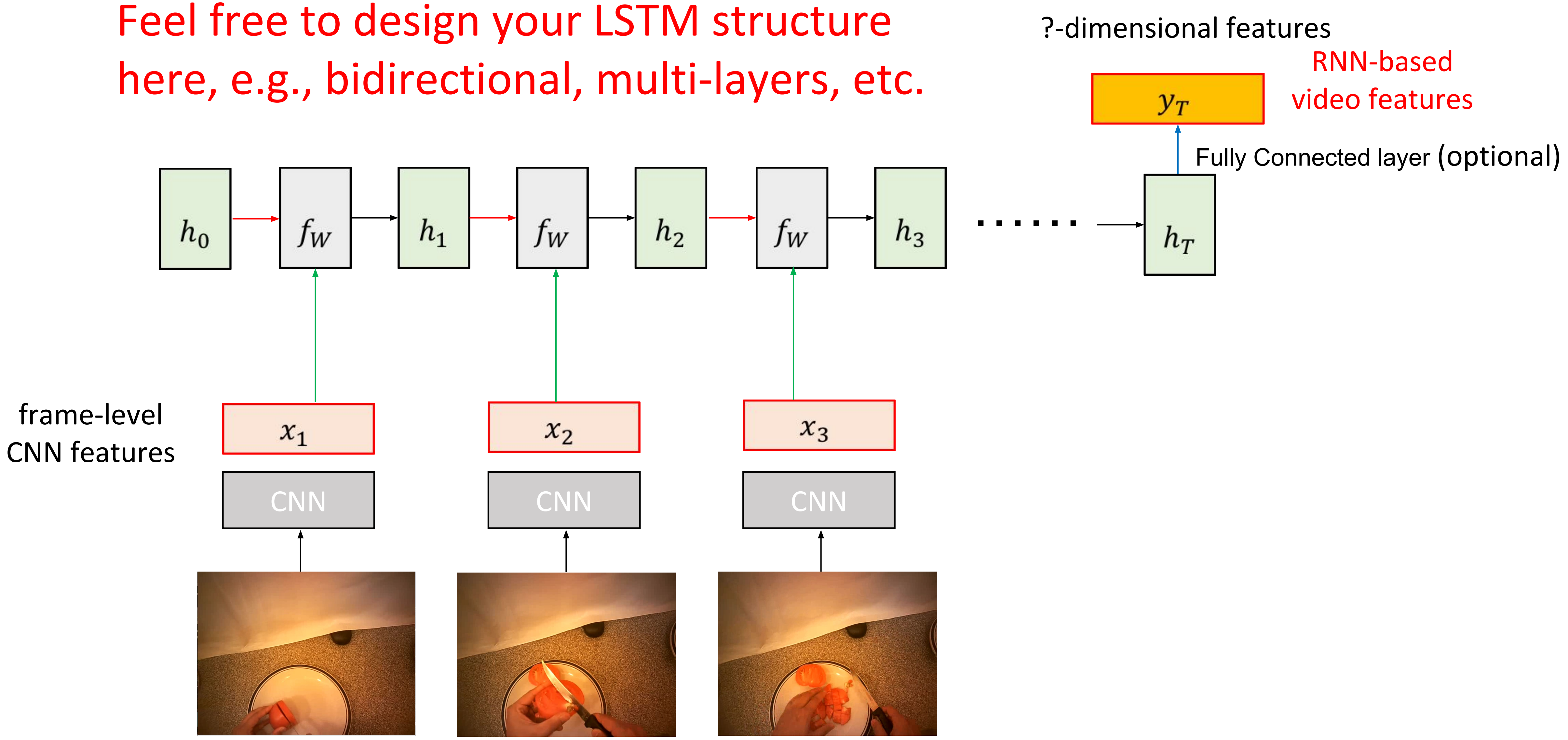
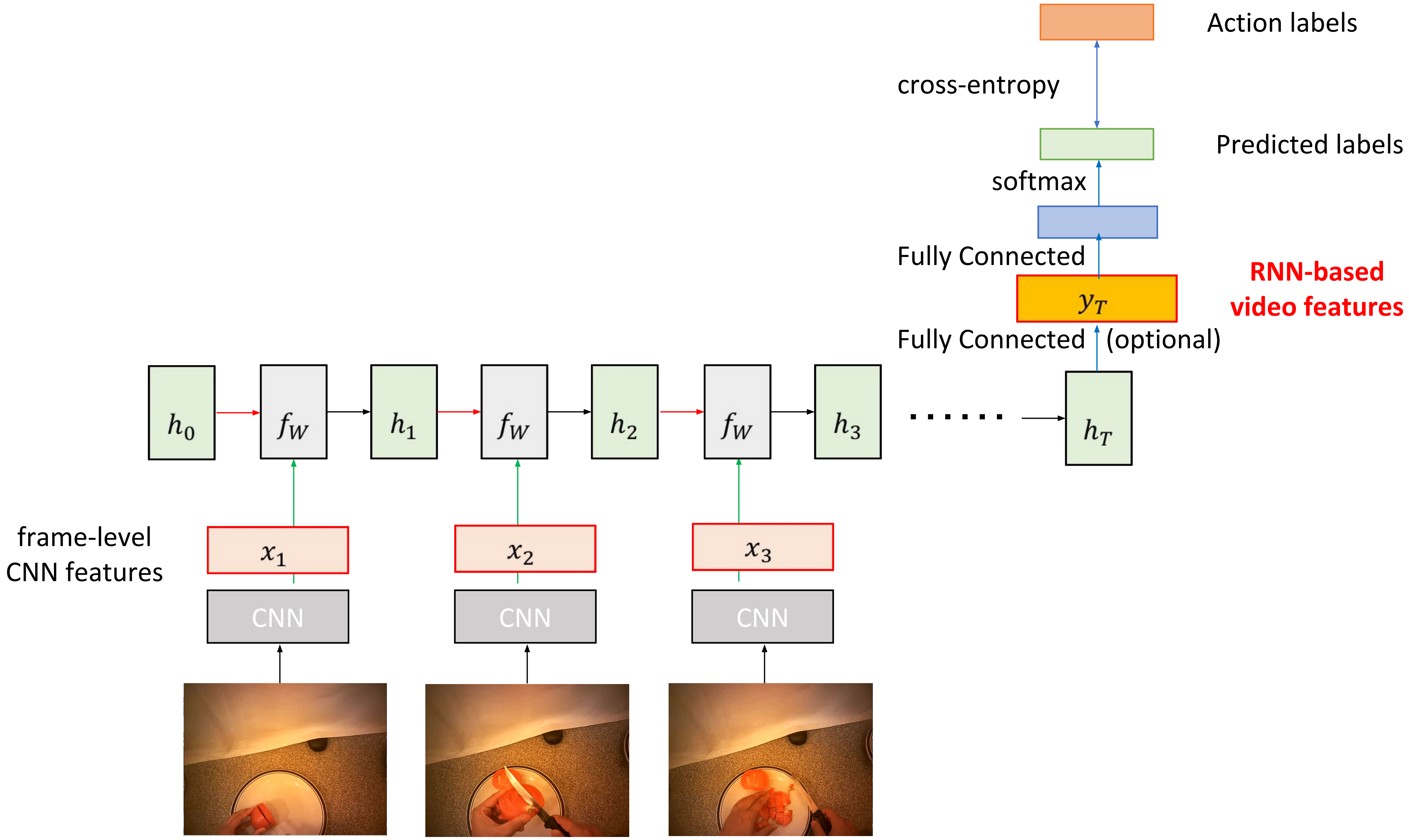


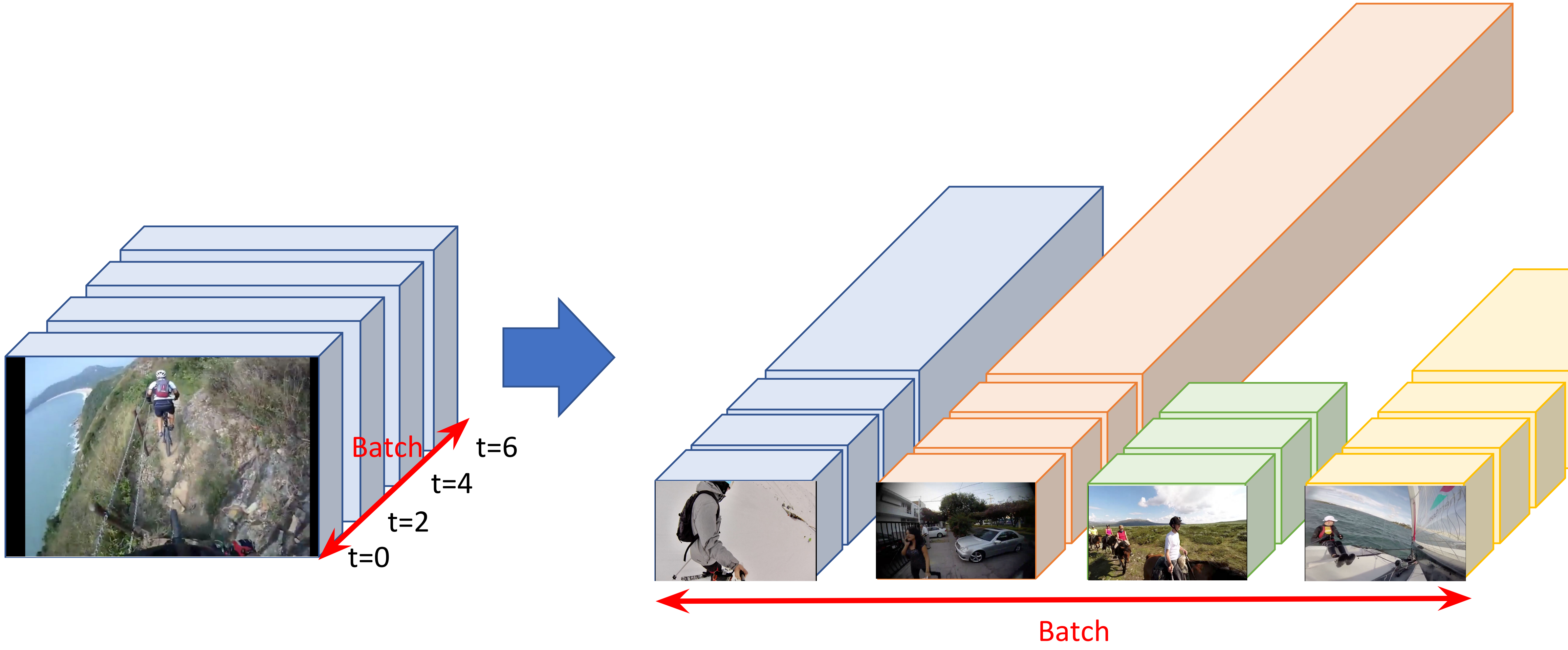
image feature
Video frames
Pre-trained model
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax
x_1
x_2
x_3
x_T

Feel free to design your LSTM structure here, e.g., bidirectional, multi-layers, etc.



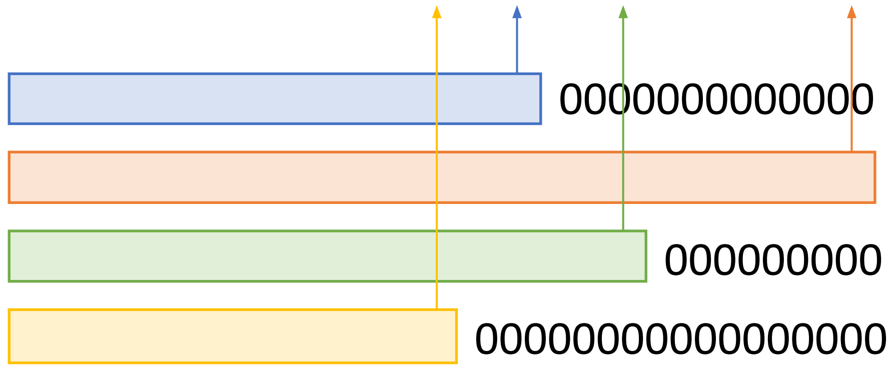

Action labels
cross-entropy
Predicted labels
softmax
Fully Connected
RNN-based video features
y_T
Fully Connected
(optional)
h_0
f_W
h_1
f_W
h_2
f_W
h_3
....
h_T
frame-level CNN features
x_1
x_2
x_3
CNN
CNN
CNN

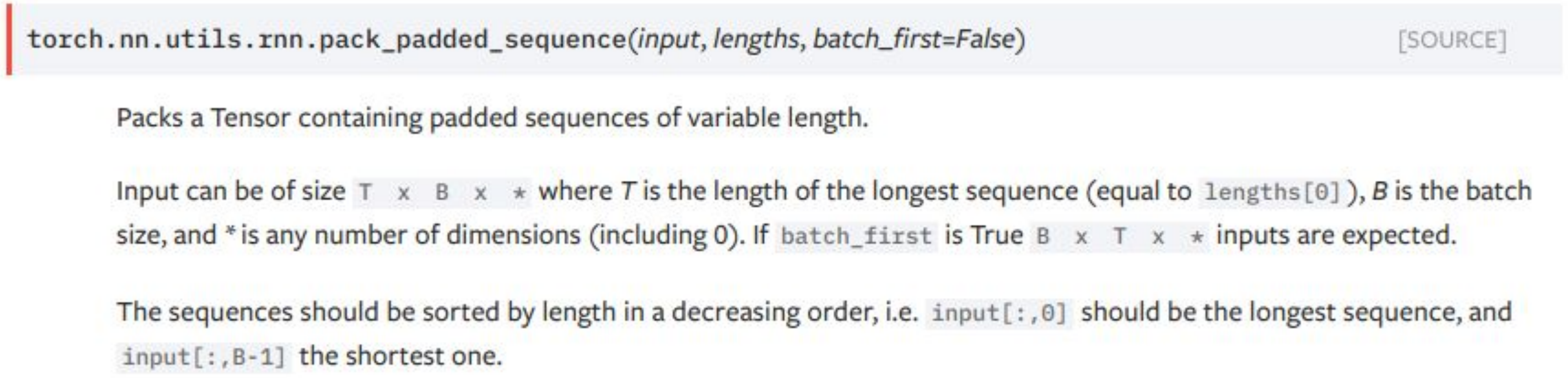
Input with varying length in a batch



Solution

- Batch size = 1
- Zero padding and take valid output only





torch.nn.utils.rnn.pack_padded_sequence(*input, lengths, batch_first=False*) [SOURCE]

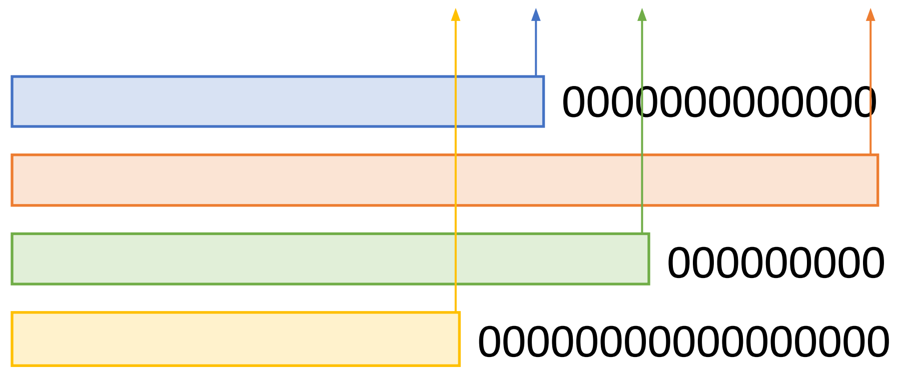
Packs a Tensor containing padded sequences of variable length.

Input can be of size `T x B x *` where *T* is the length of the longest sequence (equal to `lengths[0]`), *B* is the batch size, and * is any number of dimensions (including 0). If `batch_first` is True `B x T x *` inputs are expected.

The sequences should be sorted by length in a decreasing order, i.e. `input[:,0]` should be the longest sequence, and `input[:,B-1]` the shortest one.

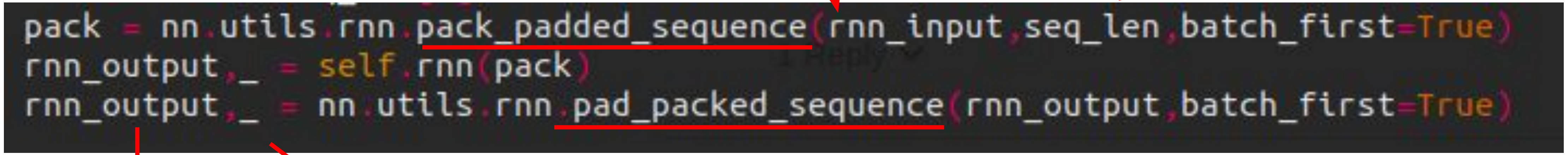
Example

If I have a RNN with hidden size = 100



Padded , Sorted by length, Batch first(if batch_first=True)**,**
ex. shape = (32, 50 , 128)
(B X T X D)

list -> [50, 48, 30, ……]



```
pack = nn.utils.rnn.pack_padded_sequence(rnn_input,seq_len,batch_first=True)
rnn_output,_ = self.rnn(pack)
rnn_output,_ = nn.utils.rnn.pad_packed_sequence(rnn_output,batch_first=True)
```

tensor([50,48,30,....])

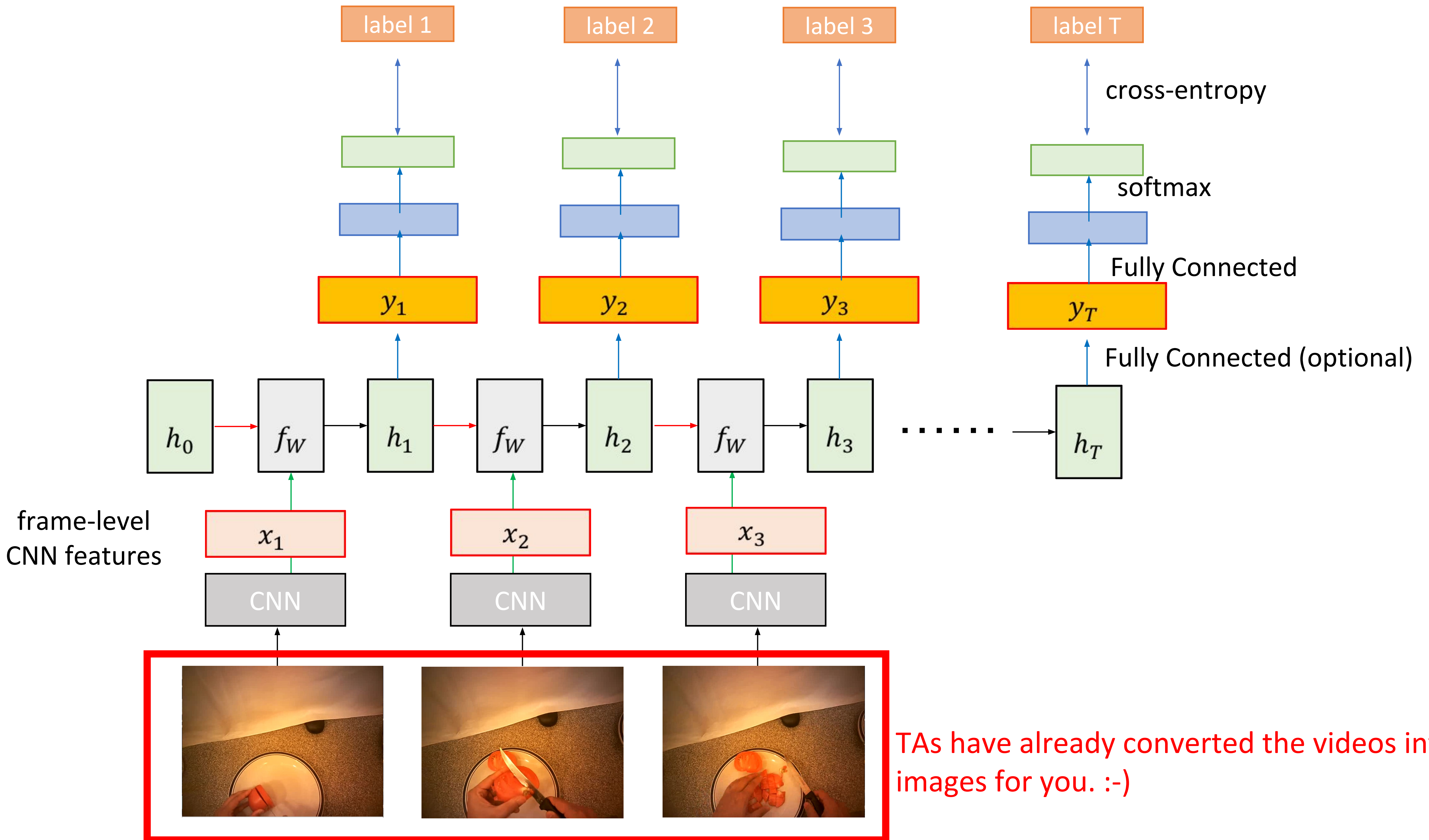
Tensor's shape = (32, 50 , 100)

Reference of variable sequence length

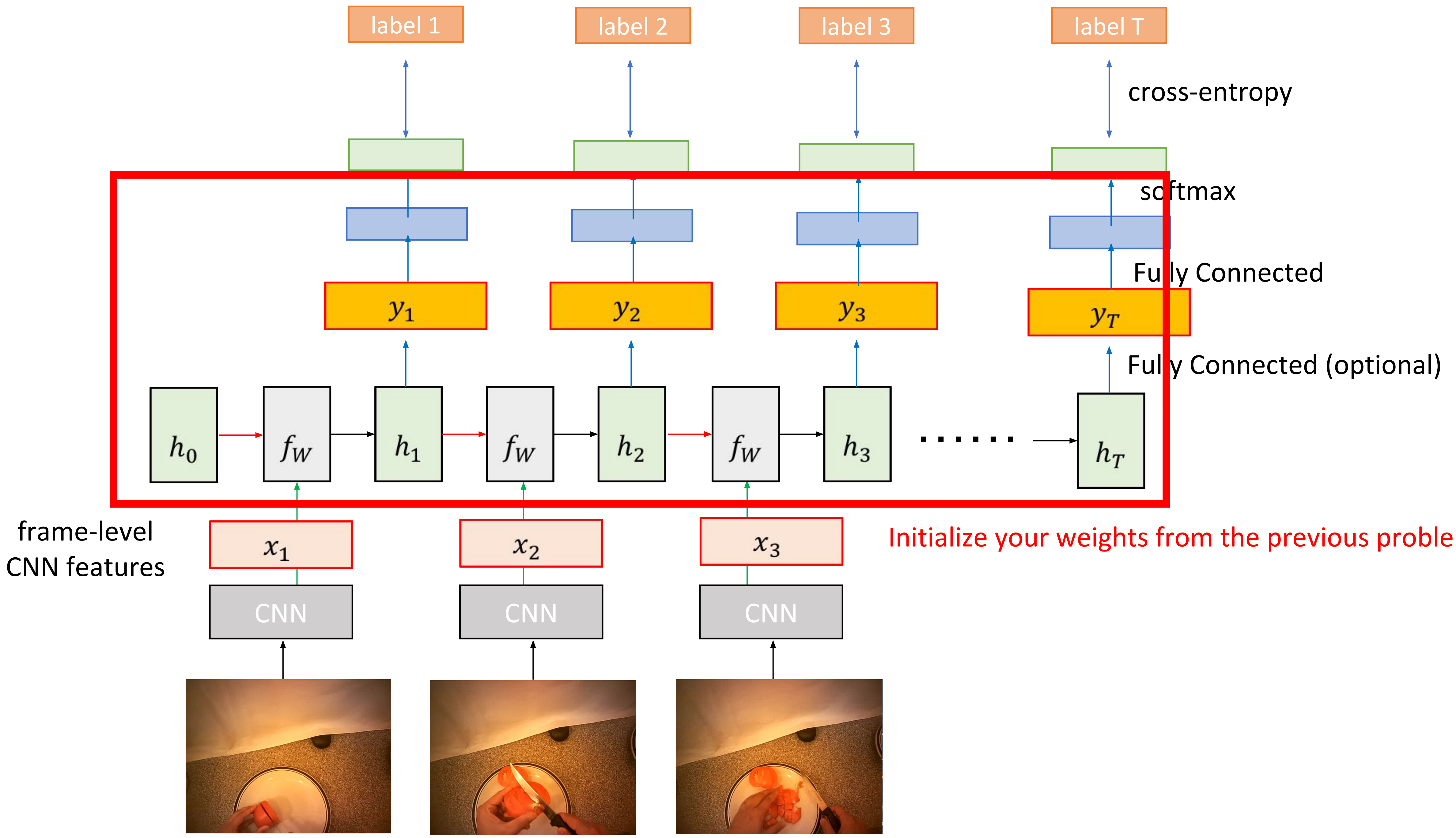
- TensorFlow
  - http://www.wildml.com/2016/08/rnns-in-tensorflow-a-practical-guide-and-undocumented-features/
- Pytorch
  - https://zhuanlan.zhihu.com/p/34418001
  - https://discuss.pytorch.org/t/understanding-pack-padded-sequence-and-pad-packed-sequence/4099
- Keras
  - https://github.com/keras-team/keras/issues/40

Task Description

- Task 1 : Data preprocessing
    - Extract state-of-the-art CNN features for action recognition
- Task 2 : Trimmed action recognition
    - Training your RNN model with sequences of CNN features and labels
- Task 3 : Temporal action segmentation
    - Extend your RNN model for sequence-to-sequence prediction



TAs have already converted the videos into images for you. :-)



Initialize your weights from the previous problem.

How to handle very long sequence

- Max time steps should be about 250-500
- Cut / Down-sample full-length videos to suitable training size
- Reference
  - https://machinelearningmastery.com/handle-long-sequences-long-short-term-memory-recurrent-neural-networks/
  - https://arxiv.org/abs/1803.04831

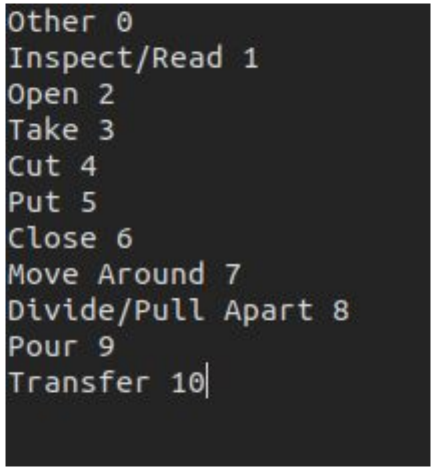
Dataset

- Total 37 full-length videos (each 5-20 mins in 24 fps)
- Split into 4151 trimmed videos (each 5-20 secs in 24 fps)
- 11 action classes
- # of videos (Full / Trimmed)
  Training : 23 / 2653
  Validation : 7 / 769
  Test : 7 / 729

Dataset

- Trimmed videos (For Task 1 & For Task 2)
  - train,valid – 240x320 trimmed videos are named as:
    <Video_category>/<Video_name><some_index>.mp4
  - gt_train.csv/gt_valid.csv
    <Video_index>, <Video_name>, <Video_category>, <Start_times>, <End_times>, <Action_labels>, <Nouns>

action labels



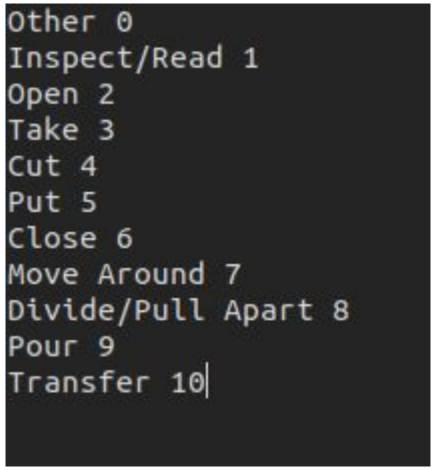
```
Other 0
Inspect/Read 1
Open 2
Take 3
Cut 4
Put 5
Close 6
Move Around 7
Divide/Pull Apart 8
Pour 9
Transfer 10
```

You cannot first convert videos into images and load the saved images as input !
(We have no space for generating images of all students)

Dataset

- Full-length videos (For Task 3)
  - train,valid – 240x320 video frames in folder are named as: <Video_category>/<Frame_index>.jpg
  - groundtruth - <Video_category>.txt
    sequence of action labels correspond to their frame index.

action labels



```
Other 0
Inspect/Read 1
Open 2
Take 3
Cut 4
Put 5
Close 6
Move Around 7
Divide/Pull Apart 8
Pour 9
Transfer 10
```

Provided data

- Download the dataset here
- Helper function (in reader.py) to read videos as ndarray
  - sudo pip install sk-video
  - sudo apt-get install ffmpeg
- Helper function (in reader.py) to read csv file as dictionary

Grading

- **Problem 1 : Data preprocessing (20%)**
- **Problem 2 : Trimmed action recognition (40%)**
- **Problem 3 : Seq-to-Seq prediction in full-length videos (40%)**
- **Bonus: Attention mechanisms (up to 20%)**

Grading

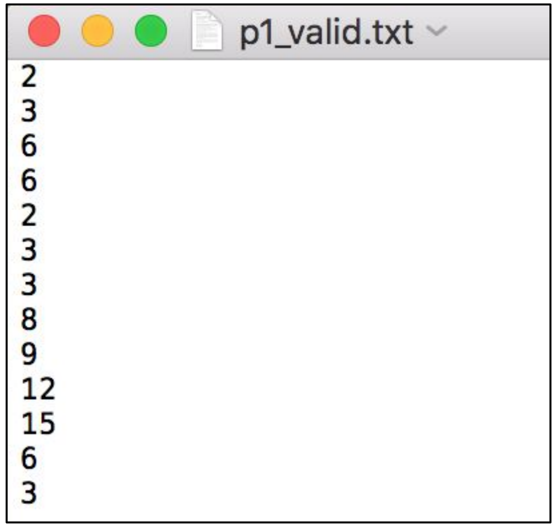
- Problem 1 : Data preprocessing (20%)
  - Describe your strategies of extracting CNN-based video features, training the model and other implementation details (which pretrained model) and plot your learning curve. (5%)

  - Report your video recognition performance (valid) using CNN-based video features. (5%)

  Note that your code need to generate a txt file [p1_valid.txt] which contains 769 lines of numbers. Each number indicates the action label of the corresponding validation video.

Example of [p1_valid.txt]

Action label of #1 video



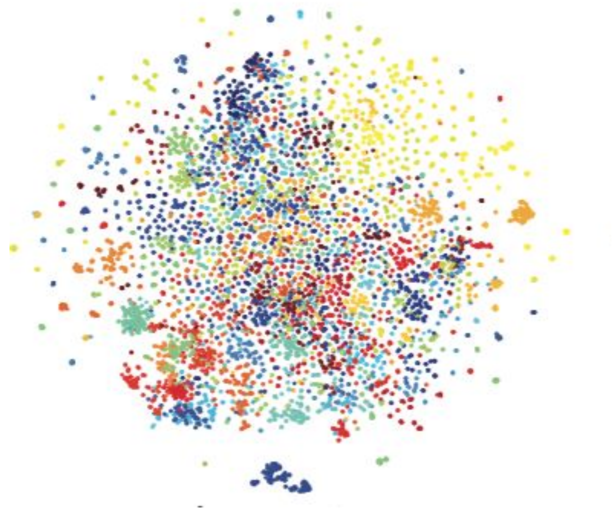
```
2
3
6
6
2
3
3
8
9
12
15
6
3
```

You need to have 769 lines

Grading

- Problem 1 : Data preprocessing (20%)
  - Visualize CNN-based video features to 2D space (with tSNE) in your report. You need to color them with respect to different action labels.(10%)

Example visualization:



CNN-based features

Grading

You cannot directly implement attention model for problem 2.
You have to implement a normal RNN!

- Problem 2 : Trimmed action recognition (40%)
  - Describe your RNN models and implementation details for action recognition and plot the learning curve of your model. (5%)
  - Your model should pass the baseline (valid: 0.45 / test: 0.43)
    validation set (10%) / test set (15%, only TAs have the test set).

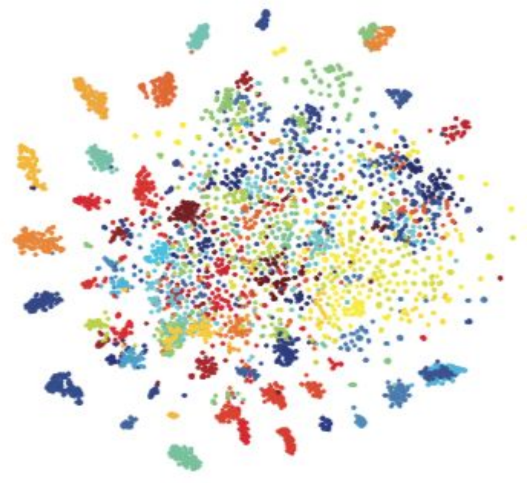
    Your code need to generate [p2_result.txt] output file. Note that [p2_result.txt] would consist of either 769 lines for validation videos or 729 lines for test videos.

Grading

- Problem 2 : Trimmed action recognition (40%)
  - Visualize RNN-based video features to 2D space (with tSNE) in your report. You need to color them with respect to different action labels.

    Do you see any improvement for action recognition compared to CNN-based video features ? Why? Please explain your observation (10%).

Example visualization:



RNN-based features

本圖不代表實際data分佈情形，僅供同學參考

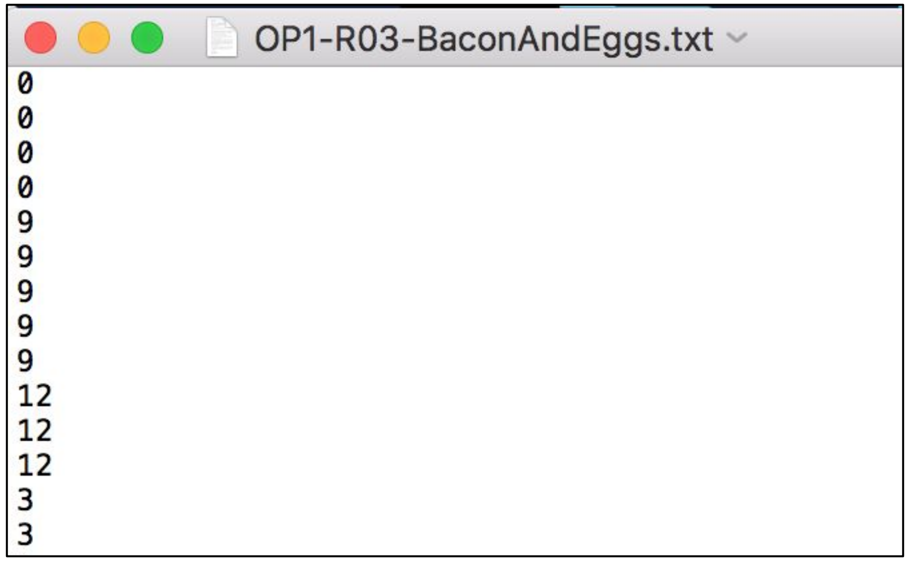
Grading

- Problem 3 : Temporal action segmentation (40%)
  - Describe any extension of your RNN models, training tricks, and post-processing techniques you used for temporal action segmentation. (5%)
  - Report validation accuracy in your report. (20%)

    For each video, you need to generate [<Video_category>.txt] which contains a sequence of action labels corresponding to each frame. Note that you need to generate 7 files for validation set in total.

Example of [<Video_category>.txt]

Action label of #1 frame

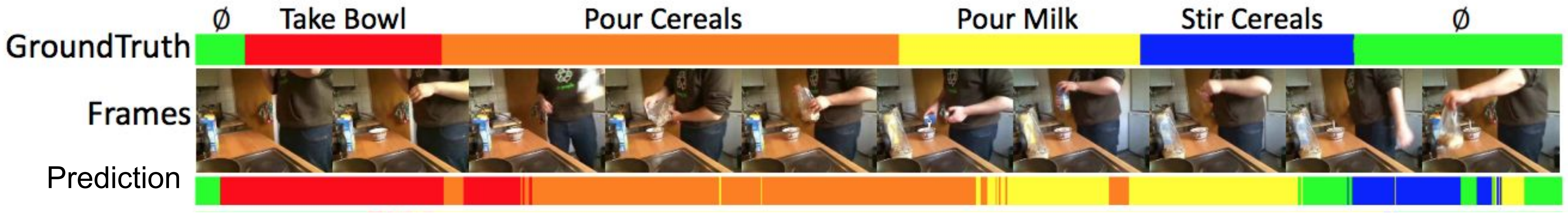

OP1-R03-BaconAndEggs.txt

```
0
0
0
0
9
9
9
9
9
12
12
12
3
3
```

Total # of lines must match the ground-truth file

Grading

- Problem 3 : Temporal action segmentation (40%)
  - Choose one video from the 7 validation videos to visualize the best prediction result in comparison with the ground-truth scores in your report. Please make your figure clear and explain your visualization results (You need to plot at least 500 continuous frames). (15%)



Example of visualization results

Bonus (up to 20%)

- Extra points will be given, if you implement and integrate attention mechanisms to **improve** both trimmed action recognition and temporal action segmentation. Please note that you CANNOT use any pre-defined attention functions or models for this part, i.e., you need to implement attention mechanism in each time step.

- You will be graded by the details of your implementation. You need to show reasonable experiment results plus detailed discussions in your report. 0 point will be given if negligible improvement. Thus, do not try to work on bonus problem without making efforts.

Homework Policy - Submission

https://classroom.github.com/a/Ip0FJlsz

- Deadline : **108/06/05 (Wed.) 01:00 AM (GMT+8)**
- Your GitHub repository should include the following files:
  - hw4_YourStudentID.pdf
  - hw4_p1.sh (for Problem 1)
  - hw4_p2.sh (for Problem 2)
  - hw4_p3.sh (for Problem 3)
  - your python files (e.g., Training code & Testing code)
  - your model files (can be loaded by your python file)

Homework Policy - Submission

- If your model is larger than GitHub’s maximum capacity (100MB), you can upload your model to another cloud service (e.g., Dropbox). However, you script file should be able to download the model automatically.
- Dropbox tutorial: link

Homework Policy - Execution

- TA will run your code as shown below
  - bash hw4_p1.sh $1 $2 $3
    - $1: directory of trimmed validation videos folder (ex. path to TrimmedVideos/video/valid)
    - $2: path of the gt_valid.csv (ex. path to TrimmedVideos/label/gt_valid.csv)
    - $3: directory of output labels folder (ex. ./output)
  - bash hw4_p2.sh $1 $2 $3
    - $1: directory of trimmed validation/test videos folder
    - $2: path of the gt_valid.csv/gt_test.csv (No labels in gt_test.csv file XD)
    - $3: directory of output labels folder
  - bash hw4_p3.sh $1 $2
    - $1: directory of full-length validation videos folder
    - $2: directory of output labels folder

- **Do not use "python" in your scripts. Use "python3" instead.**

Homework Policy - Packages

- Python : 3.5+
- Tensorflow : 1.13
- Keras : 2.2+
- Pytorch : 1.0
- h5py : 2.9.0
- Numpy : 1.16.2
- Pandas : 0.24.0
- torchvision==0.2.2, opencv, Matplotlib, Scikit-image, Pillow, skvideo ,Scipy, Python standard Lib.
- **E-mail or ask TA first if you want to import other packages.**

Rules

- Delay quota: Deducted 300% each day excluding using 5 free late day quota this semester
- Academic Ethics: Discussion between classmates is encouraged, however, please do NOT copy (or let someone copy your) homework. TA will check the similarity of every submission.
- Rules Violation: Violation of any format/execution specification will result in zero points. Please follow homework spec carefully and ask without any hesitation.
- External Dataset: Using external dataset is forbidden for this homework

Academic Integrity

- Can discuss HW with peers, but DO NOT copy and/or share code
    - **任一次作業抄襲/被抄襲者**，按校規論且**本課程學期成績為F!**
    - This is university policy and not negotiable.
- Do not directly use code from Internet unless you have permissions.
    - If not sure, ask!
    - If so, do specify in your HW/project.
- Do not use your published work as your final project.
    - However, you are encouraged to turn your high-quality projects into publications.

Very kind reminder from TA

- If you have any question, you can:
  - Use TA hours (please check course website for time/location)
  - Contact TAs by e-mail (ntudlcvta2019@gmail.com)
  - Post your question under hw4 FAQ section in FB group
  - Useful website: link
  - **DO NOT** directly message TAs (we will ignore any direct message)

Notes for Grading

P1:

For the 5%, you cannot print screen your “code”. You can draw with block diagrams to describe your architecture.

P2:

Same as P1.

P3:

How to get the 20% of the accuracy score? A: I will check your results and the visualization figure. If your model all predict one specific label (ex “0”), I will not give you the 20% score although you can reproduce it.

Bonus:

According to p36., you have to “improve” your prediction score.