# DLCV HW5

2018/05/15

# Reminder

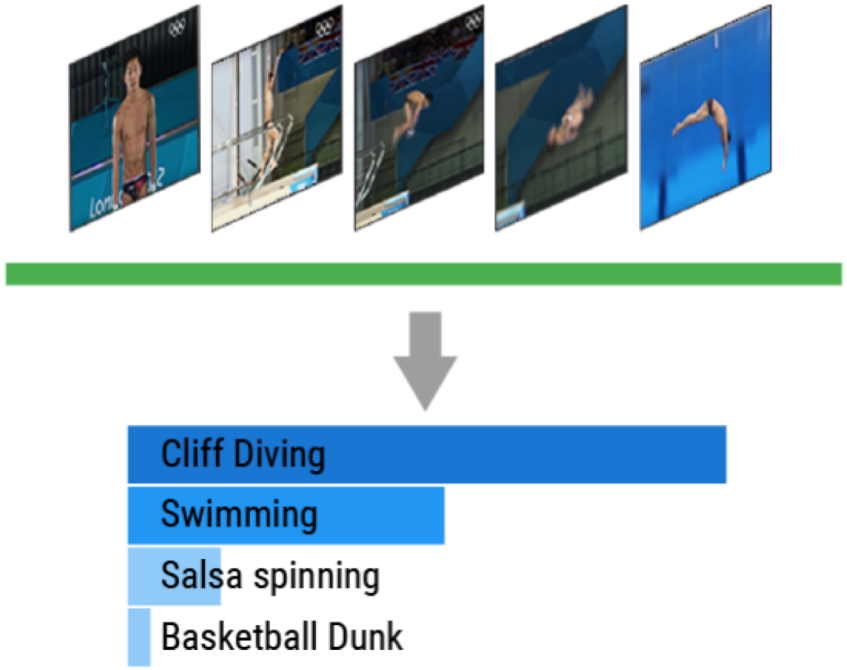
- Last Homework! 🤟
- 3.5 weeks long
- 5/25 停修截止

# Goal

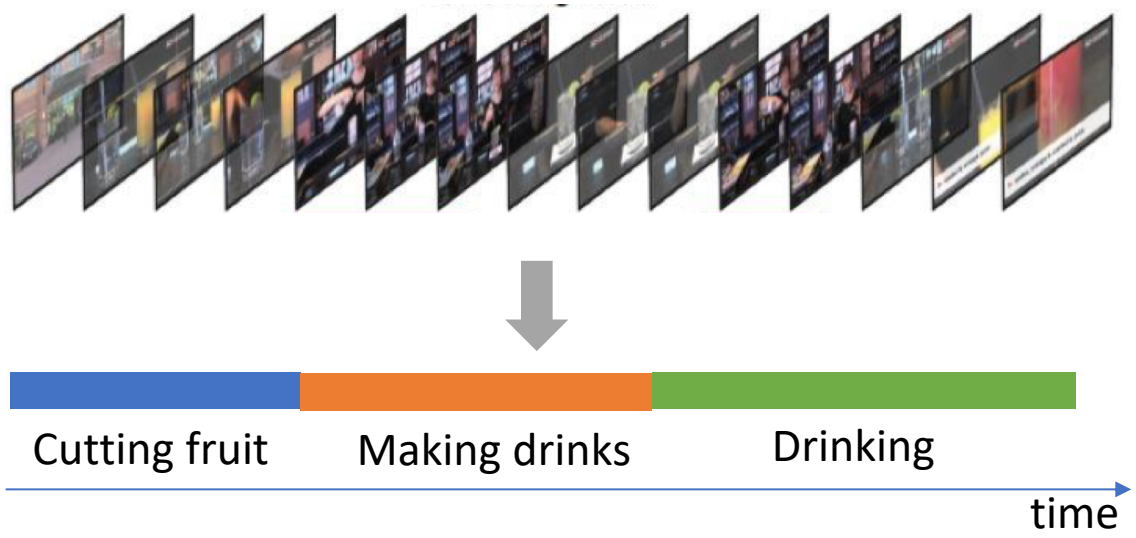
- Ability to extract state-of-the-art deep CNN features
- Implement recurrent neural networks (RNN) for action recognition
- Extend RNN models for solving sequence-to-sequence problems

# 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

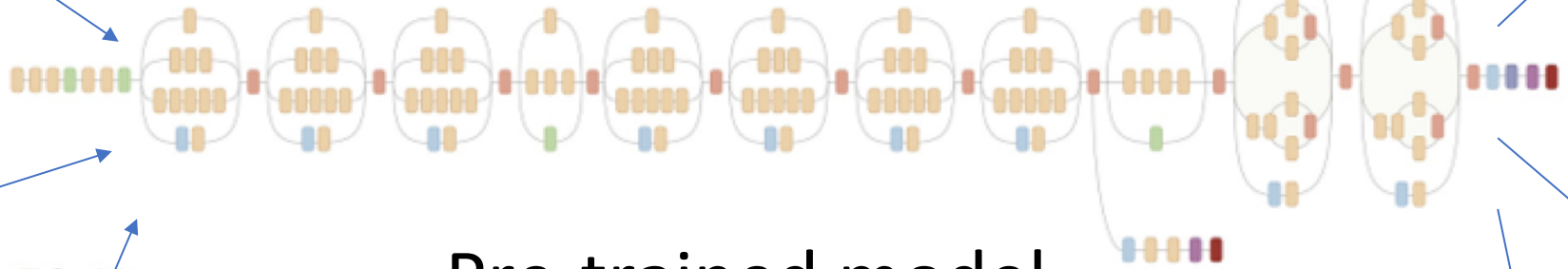
# Task Description

- 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

Video frames




## Pre-trained model

- ResNet-50
- Inception v3
- VGG-16
- Can’t use 3DCNN

image feature

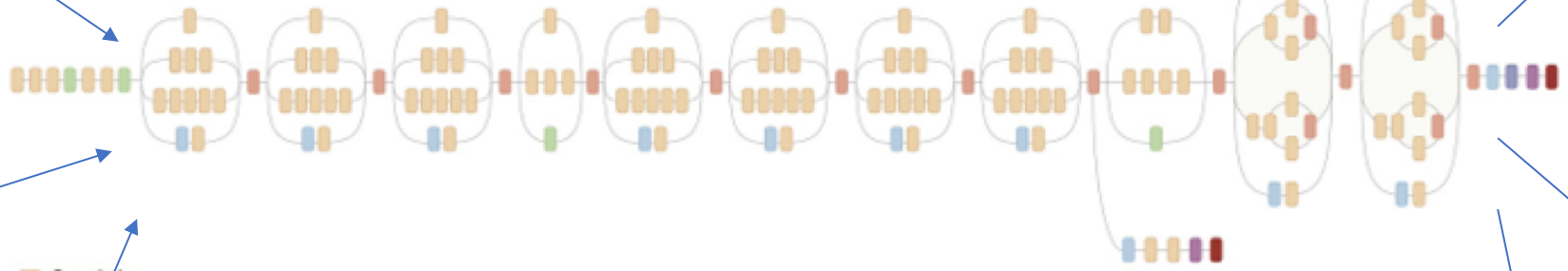
$x_1$

$x_2$

$x_3$

$x_T$

Video frames




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

image feature

$x_1$

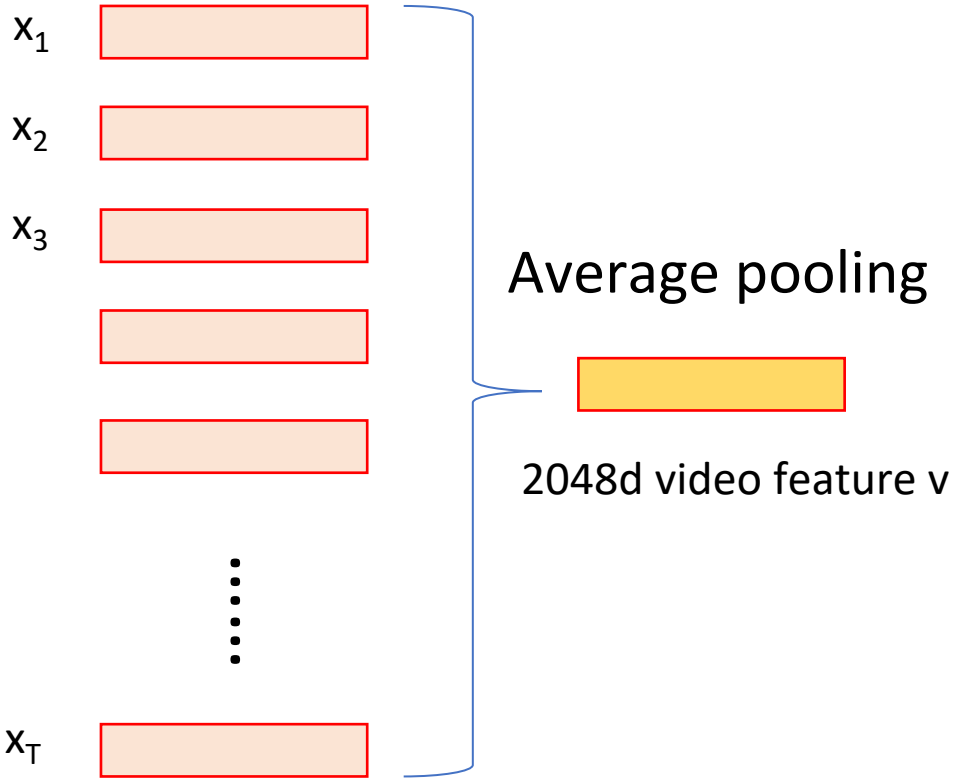
$x_2$

$x_3$

$x_T$

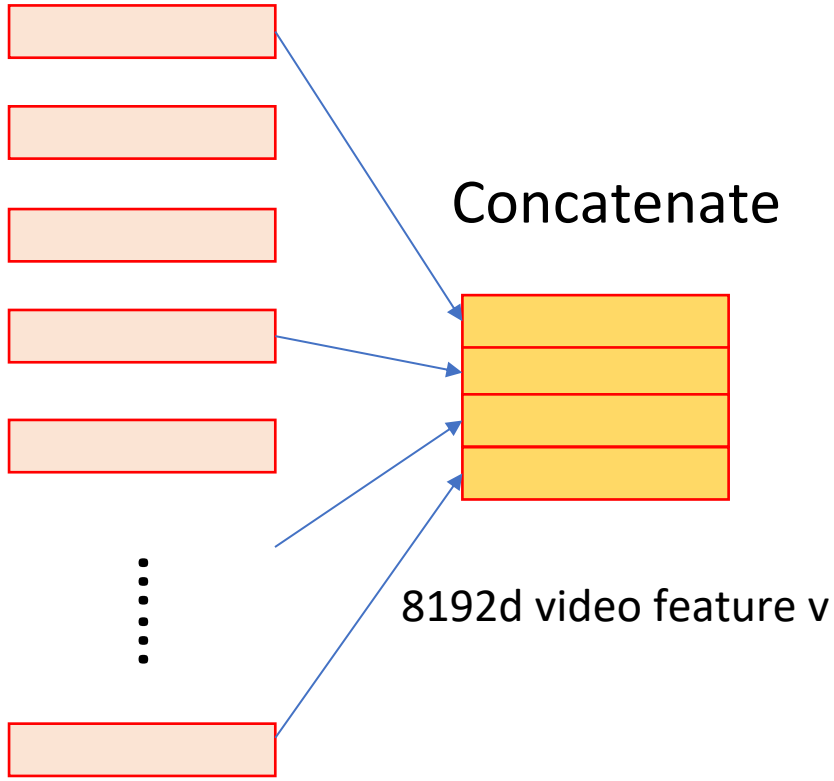
# CNN-based video features

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

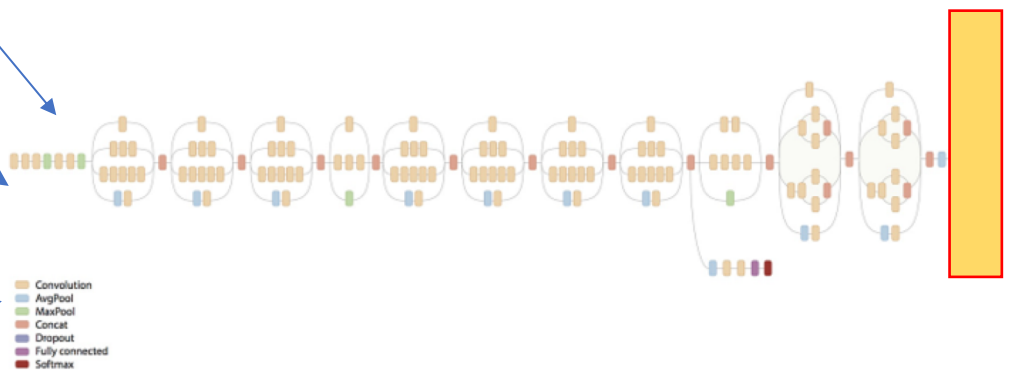
# CNN-based video features

- 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.)




# Pre-trained model




CNN-based video features

FC

predicted labels

softmax

cross-entropy

action labels

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

Training these parts

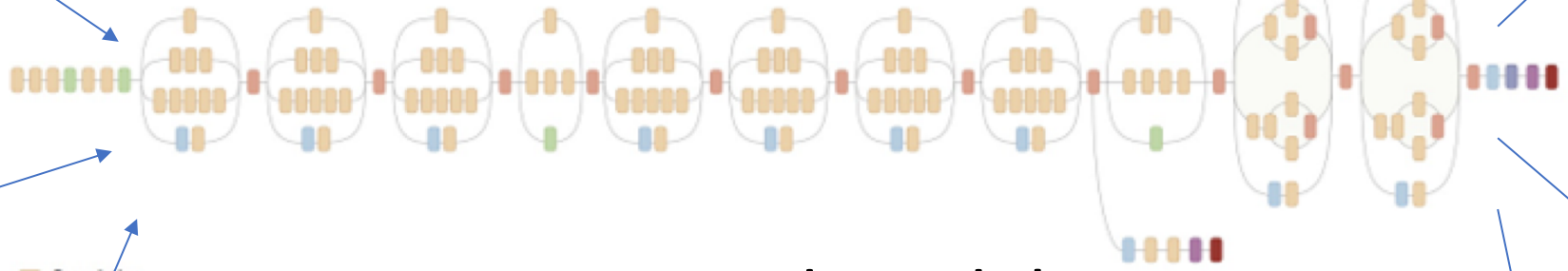
(☞It might take lots of computation times and resources for fine-tuning the pre-trained model.)

# 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



Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

Pre-trained model

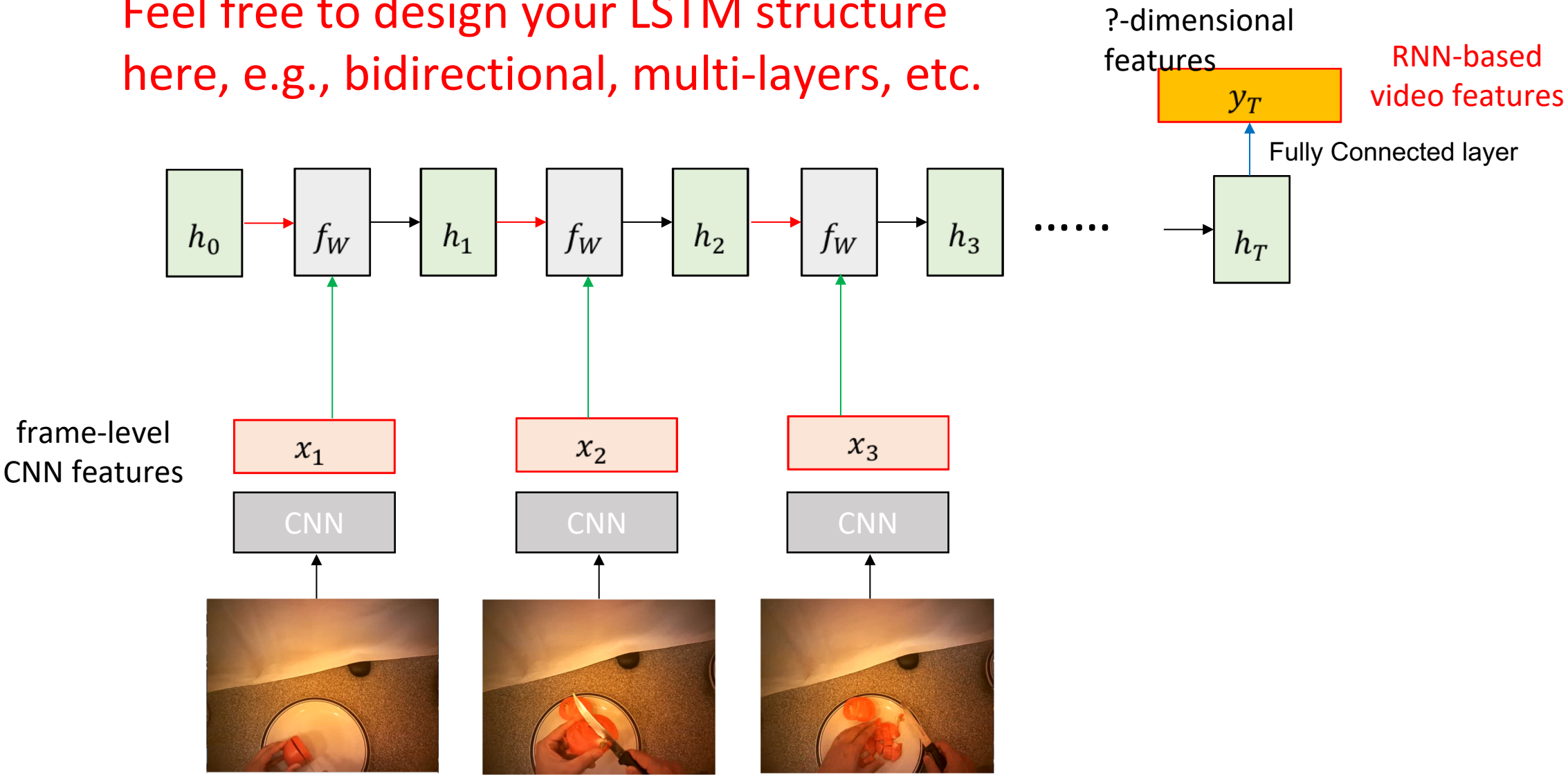
$x_1$

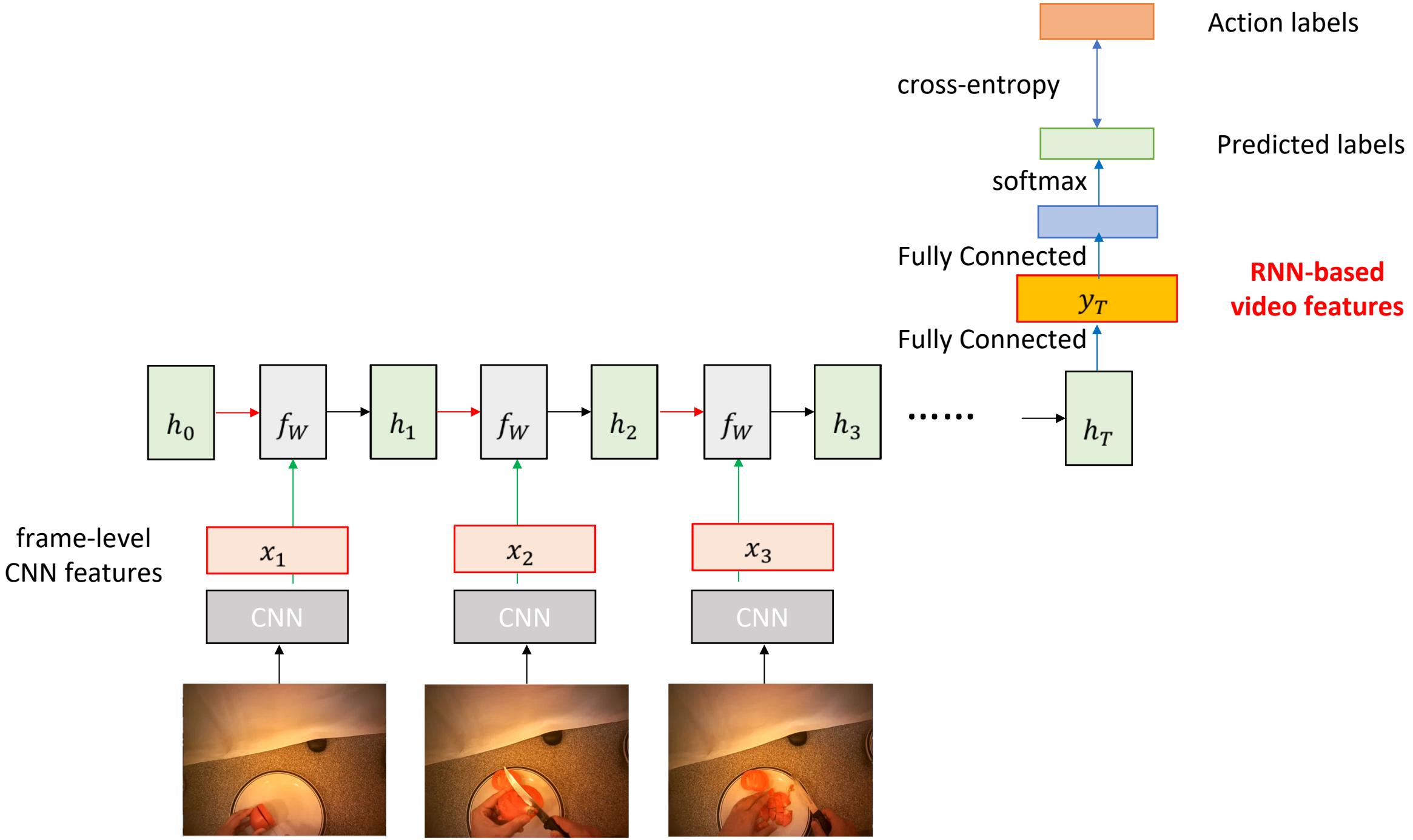
$x_2$

$x_3$

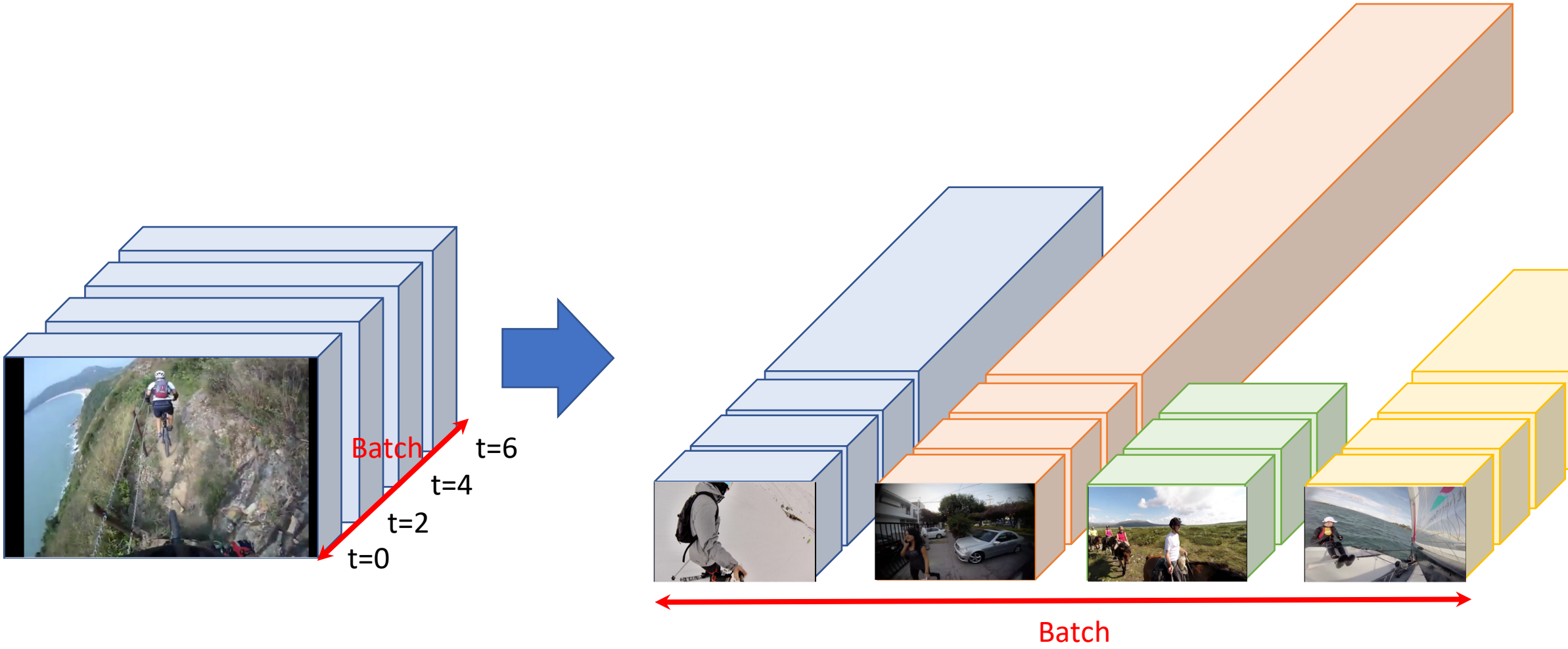
$x_T$

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



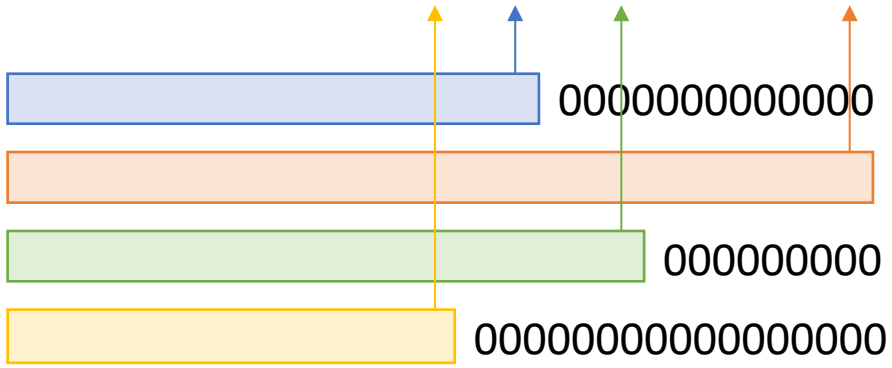

Action labels
cross-entropy
Predicted labels
softmax
Fully Connected
RNN-based
video features
$y_T$
Fully Connected
$h_0$
$f_W$
$h_1$
$f_W$
$h_2$
$f_W$
$h_3$
.....
$h_T$
frame-level
CNN features
$x_1$
$x_2$
$x_3$
CNN
CNN
CNN

# Input with varying length in a batch



# Solution

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




```
tf.nn.dynamic_rnn(
    cell,
    inputs,
    sequence_length=None,
    initial_state=None,
    dtype=None,
    parallel_iterations=None,
    swap_memory=False,
    time_major=False,
    scope=None
)
```

`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.

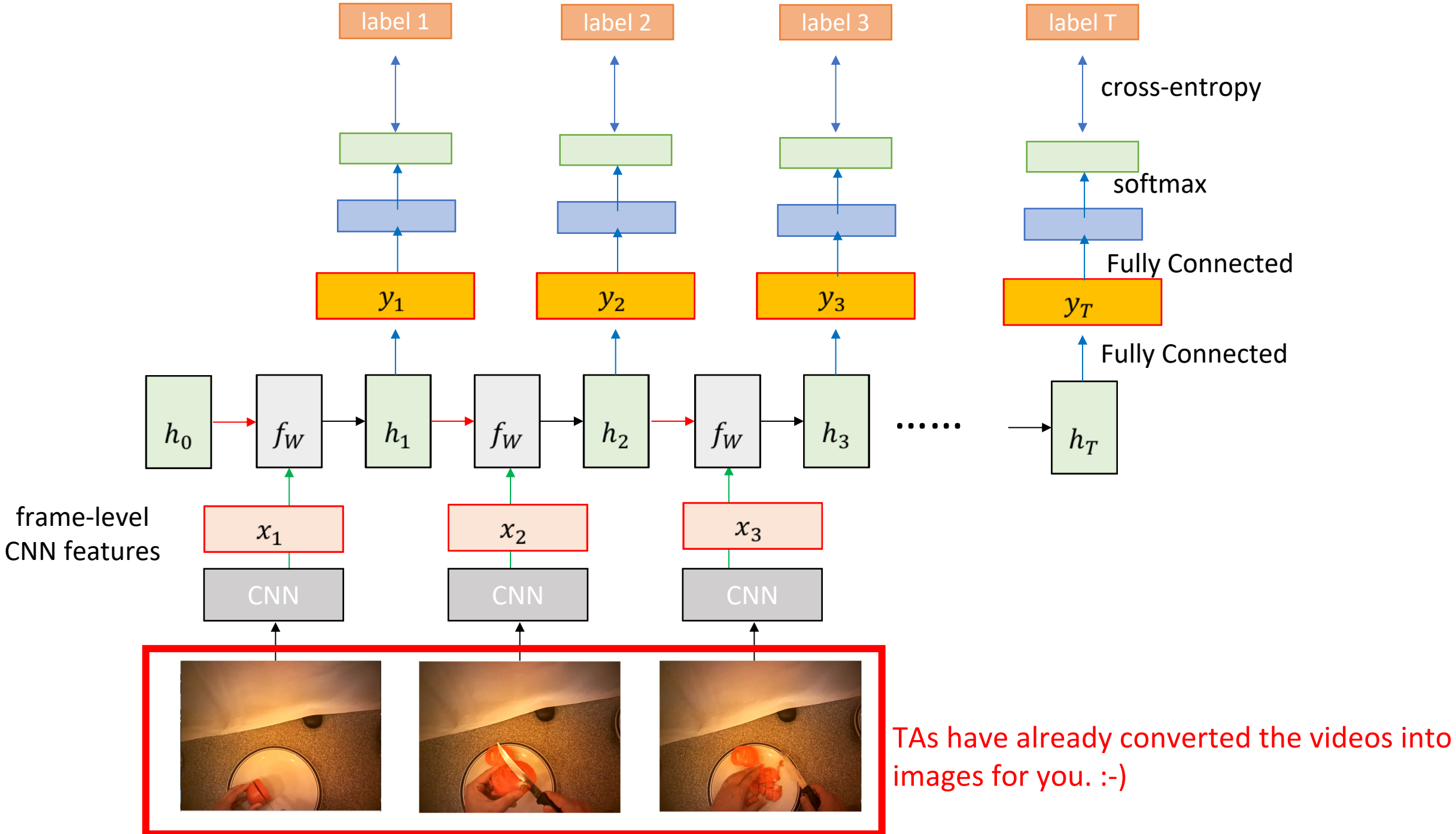
The sequences should be sorted by length in a decreasing order, i.e. `input[:,0]` should be the longest sequence, and `input[:,B-1]` the shortest one.

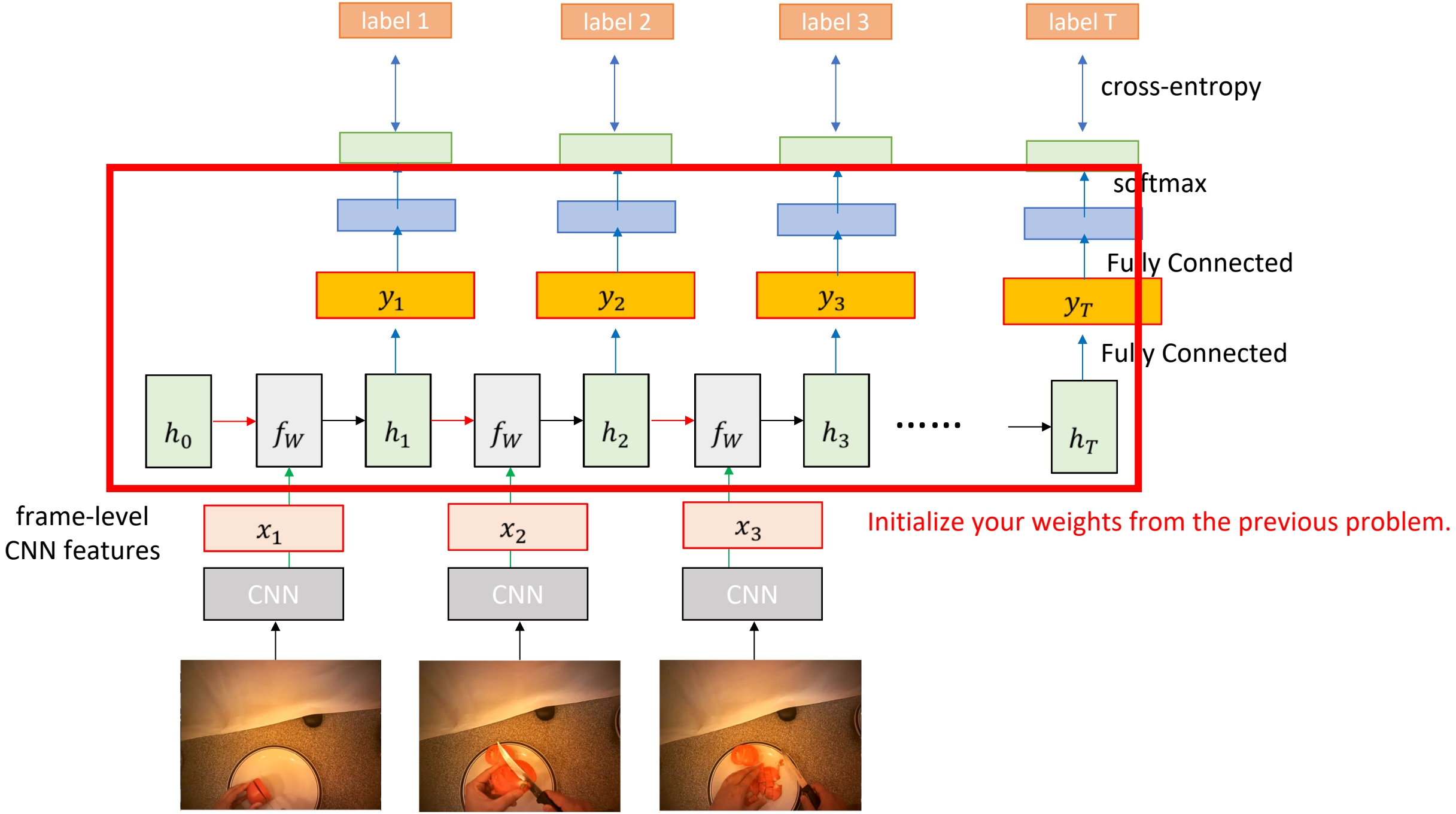
# Reference of variable sequence length

- 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


label 1
label 2
label 3
label T
cross-entropy
softmax
Fully Connected
Fully Connected
$y_1$
$y_2$
$y_3$
$y_T$
$h_0$
$f_W$
$h_1$
$f_W$
$h_2$
$f_W$
$h_3$
.....
$h_T$
frame-level
CNN features
$x_1$
$x_2$
$x_3$
CNN
CNN
CNN
TAs have already converted the videos into images for you. :-)


label 1
label 2
label 3
label T
cross-entropy
softmax
Fully Connected
Fully Connected
$y_1$
$y_2$
$y_3$
$y_T$
$h_0$
$f_W$
$h_1$
$f_W$
$h_2$
$f_W$
$h_3$
.....
$h_T$
frame-level
CNN features
$x_1$
$x_2$
$x_3$
CNN
CNN
CNN
Initialize your weights from the previous problem.

# How to handle very long sequence

- Max time steps should be about 250-500
- Use Truncated Backpropagation Through Time (TBTT)
  - https://www.tensorflow.org/tutorials/recurrent#truncated_backpropagation
  - https://discuss.pytorch.org/t/implementing-truncated-backpropagation-through-time/15500/4
- 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)
- 21 action classes
- # of videos (Full / Trimmed)
  Training : 23 / 3236
  Validation : 5 / 517
  Test : 3 / 398

# 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
```

# 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 and helper functions on ceiba
- Helper function to read videos as ndarray
  - sudo pip install sk-video
  - sudo apt-get install ffmpeg
- Helper function to read csv file as dictionary

# Grading

- **Problem 1 : Data preprocessing (20%)**
  - Report your CNN performance and training strategies
- **Problem 2 : Trimmed action recognition (55%)**
  - Pass the validation baseline and the test baseline
  - Draw t-SNE graph for CNN-based (in Problem 1) & RNN-based video features (in Problem 2)
- **Problem 3 : Seq-to-Seq prediction in full-length videos (25%)**
  - Report your performance and make visualization figures
- **Bonus: Attention mechanisms (up to 20%)**

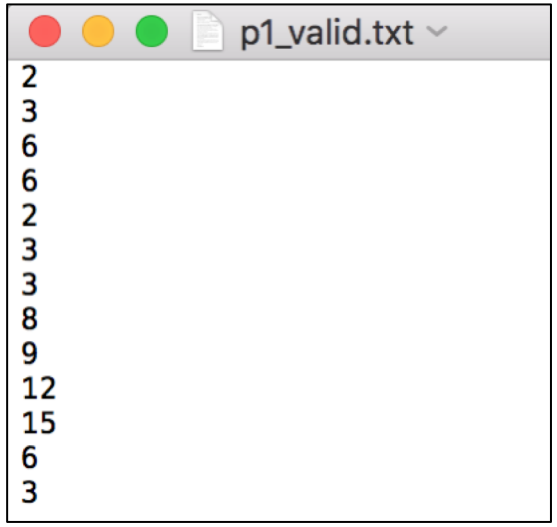
# Grading

- Problem 1 : Data preprocessing (20%)
  - Describe your strategies of extracting CNN-based video features, training the model and other implementation details. (5%)
  - Report your video recognition performance using CNN-based video features (10%) and plot the learning curve of your model (5%).

    Note that your code need to generate a txt file [p1_valid.txt] which contains 517 lines of numbers. Each number indicates the action label of the corresponding validation video.

# Example of [p1_valid.txt]

Action label of #1 video




You need to have 517 lines

# Grading

- Problem 2 : Trimmed action recognition (55%)
  - Describe your RNN models and implementation details for action recognition. (5%)
  - Your model should pass the baseline (valid: XXXXX / test: XXXXX )
    validation set (15%) / test set (20%, only TAs have the test set).

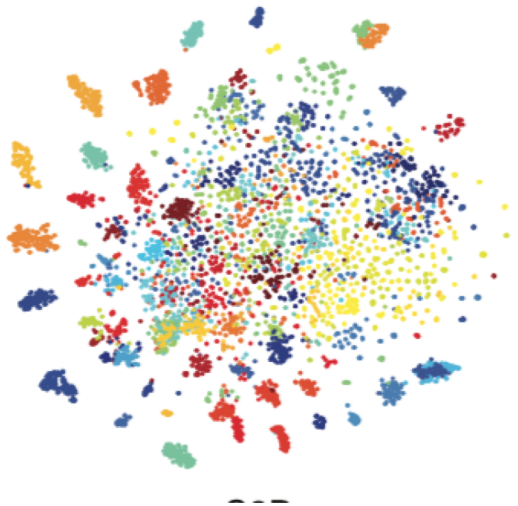
    Your code need to generate [p2_result.txt] output file. Note that [p2_result.txt] would consist of either 517 lines for validation videos or 398 lines for test videos.

# Grading

- Problem 2 : Trimmed action recognition (55%)
  - Visualize CNN-based video features and RNN-based video features to 2D space (with tSNE) in your report. You need to generate two separate graphs and color them with respect to different action labels. Do you see any improvement for action recognition? Please explain your observation (15%).

Example visualization:

CNN-based features



RNN-based features

本圖不代表實際data分佈情形，僅供同學參考

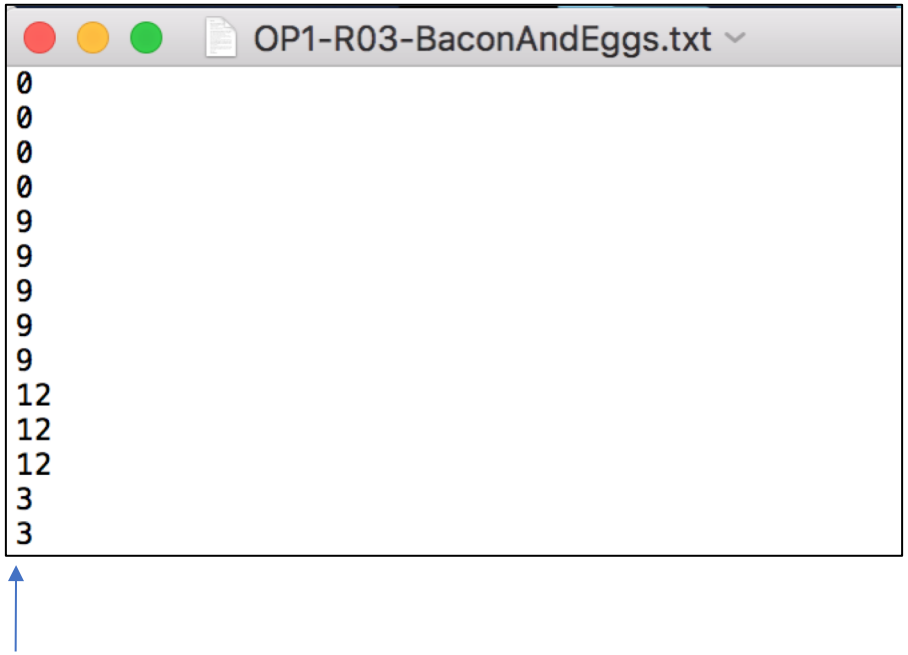
# Grading

- Problem 3 : Temporal action segmentation (25%)
  - Describe any extension of your RNN models, training tricks, and post-processing techniques you used for temporal action segmentation. (5%)
  - Report validation accuracy and plot the learning curve (10%) in your report.

    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 5 files for validation set in total.

# Example of [<Video_category>.txt]

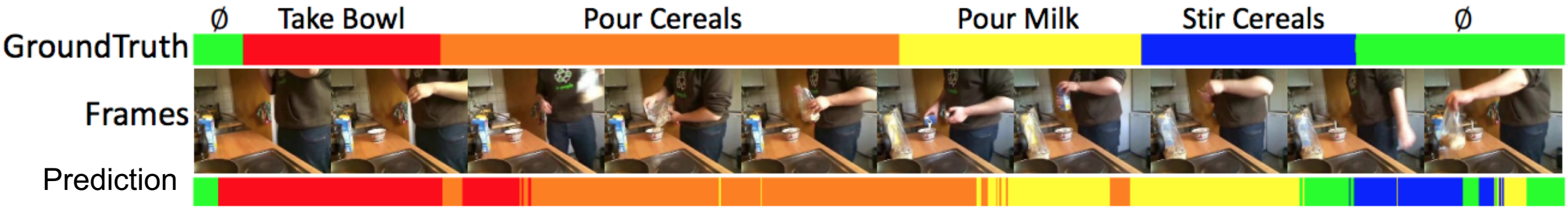
Action label of #1 frame




Total # of lines must match the ground-truth file

# Grading

- Problem 3 : Temporal action segmentation (25%)
  - Choose one video from the 5 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 300 continuous frames (2.5 mins). (10%)




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

- DLCV2018SPRING/hw5 on your GitHub repository should include the following files:
  - hw5_YourStudentID.pdf (report template)
  - hw5_p1.sh (for Problem 1)
  - hw5_p2.sh (for Problem 2)
  - hw5_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

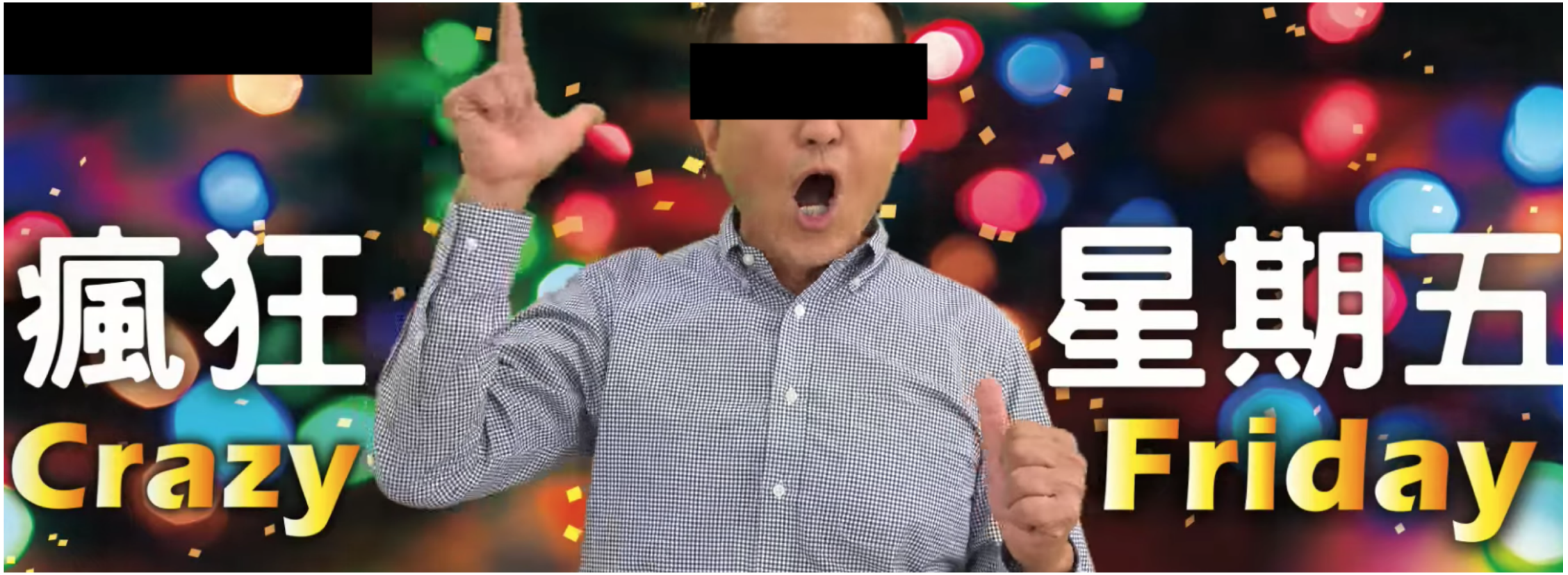
# Homework Policy - Execution

- TA will run your code as shown below
  - bash hw5_p1.sh $1 $2
    - $1: directory of trimmed validation videos folder
    - $2: directory of output labels folder
  - bash hw5_p2.sh $1 $2
    - $1: directory of trimmed validation/test videos folder
    - $2: directory of output labels folder
  - bash hw5_p3.sh $1 $2
    - $1: directory of full-length validation videos folder
    - $2: directory of output labels folder

# Homework Policy - Packages

- Python : 3.6
- Tensorflow : 1.6
- Keras : 2.1.5
- Pytorch : 0.4.0
- h5py : 2.7.1
- Numpy : 1.14.2
- Pandas : 0.22.0
- Matplotlib, Scikit-image, Pillow, Scipy, any video IO Lib, Python standard Lib.
- E-mail or ask TA first if you want to import other packages.

# Deadline




2018/06/08 (Fri) 23:59:59 (GMT+8)

# Rules

- Delay quota: Deducted 30% each day excluding using 3 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

- TA hours is not 24/7:
  同學有任何問題TA都很樂意討論與回答，
  不過請同學遵守TA hours的時間規定。
- TA不是通靈少女，請不要貼error或learning curve叫TA Debug...
  - Google你的error message可以解決90%的疑慮
  - 請先看過Facebook 討論版置頂Q&A，同學有問過的問題都會更新
  - 在Facebook 討論版詳細描述你的問題，TA跟熱心的同學都會幫忙回答
  - 寄信詳細描述你的問題到TA信箱，並耐心等候TA的回覆
- 請不要私訊給任何一位TA！只會得到已讀訊息...