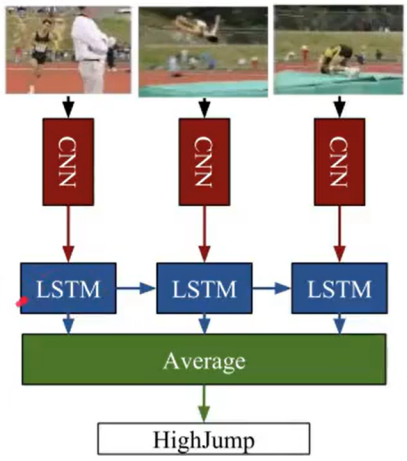
✅

Lecture 14

Models

- Long Term Recurrent Convolutional Network

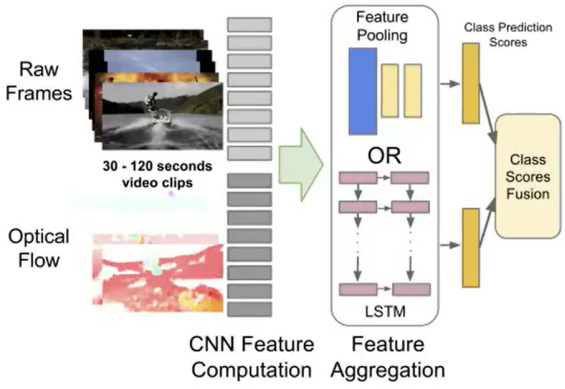
  - Long-term Recurrent Convolutional Network
    - A direct application of LSTM idea to action recognition (classification).
  - Architecture:
    - Input: 16×227×227 video
    - Each frame is encoded by a pre-trained CNN (CaffeNet -- a variation of AlexNet, VGG).
    - For every input frame, the LSTM cell predicts activity classes.
    - Per-frame predictions are aggregated by averaging.
  - Applied it to image/video captioning as well.
    - (Will be covered later in this course.)



https://arxiv.org/pdf/1411.4389.pdf

- Beyond short snippets

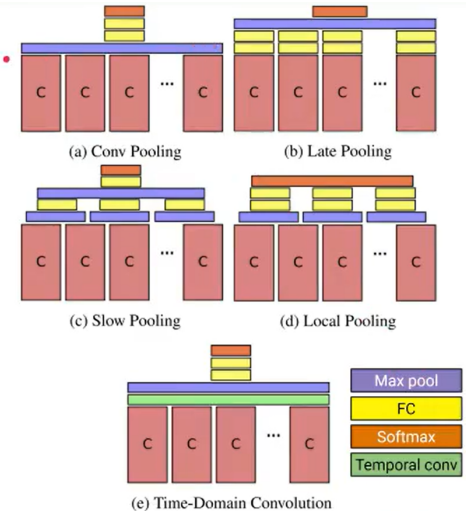
  - Most previous works take short video snippets (e.g., 16 frames) as input.
  - This work explores video understanding with **longer** but more **sparsely sampled** sequence of frames.
    - 1 fps, up to 300 frames (5 min)
    - To compensate for motion information loss, optical flow was extracted from 15 fps samples around each main frame.
  - For frame aggregation, tried both pooling and LSTMs.



    - Pooling

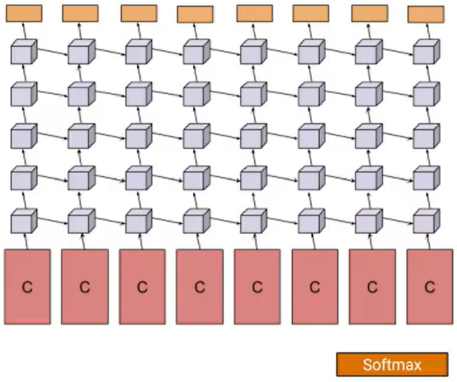
- Given a sequence of conv-features:
  - **Conv pooling**: taking max over time
  - **Late pooling**: adding a few FC layers before taking max
  - **Slow pooling**: taking max hierarchically, with an FC layer in-between.
  - **Local pooling**: taking max only locally, then local features are concatenated before softmax.
  - **Time-domain conv**: 1×1 convolution (dimension reduction) performed before max pooling

Method	Clip Hit@1	Hit@1	Hit@5
Conv Pooling	**68.7**	**71.1**	**89.3**
Late Pooling	65.1	67.5	87.2
Slow Pooling	67.1	69.7	88.4
Local Pooling	68.1	70.4	88.9
Time-Domain Convolution	64.2	67.2	87.2



- LSTM
  - multi-layer LSTM을 사용함

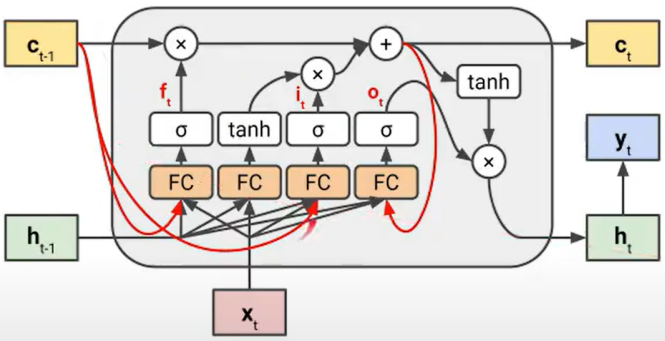
- Similar to LRCN, with the following differences:
  - **Multi-layer LSTM** (instead of a single layer) is applied on top of conv-features extracted from each frame.
  - To **aggregate frame-level predictions**, they tried:
    - Taking the prediction at the last step
    - Max aggregation
    - Average aggregation (summing)
    - Weighted averaging
  - All methods perform similarly, but weighted averaging slightly the best. (See Section 3.3 of the paper for more details.)



- 실험 결과 : LSTM이 pooling보다 결과가 더 좋았다

- FC-LSTM

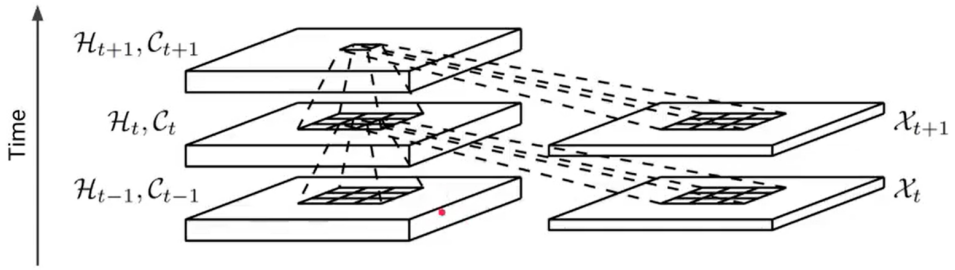
Additional connections from cell state (**c**) to the FC layers for each gate.



$$+\mathbf{V}_f\mathbf{c}_{t-1}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$+\mathbf{V}_i\mathbf{c}_{t-1}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$+\mathbf{V}_o\mathbf{c}_t$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{c}_t = \mathbf{f} \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

https://arxiv.org/pdf/1308.0850.pdf

- ConvLSTM
  - conv는 공간적인 정보 + LSTM은 시간적인 정보 → spatio-temporal dynamic한 모델링



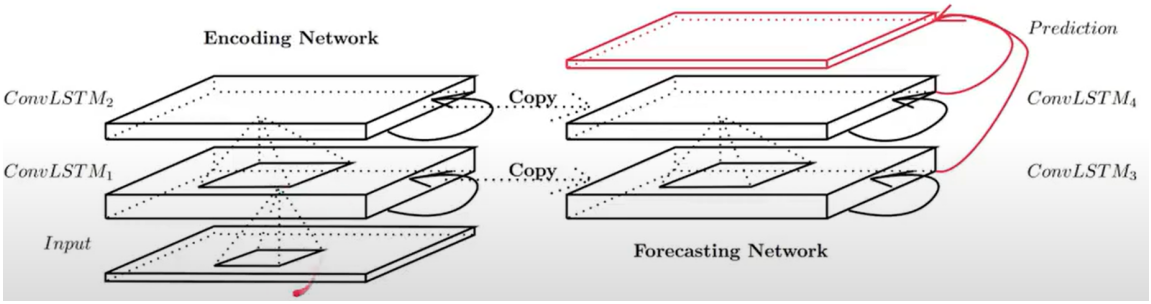
ConvLSTM

$$i_t = \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f)$$
$$\mathcal{C}_t = f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c)$$
$$o_t = \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o)$$
$$\mathcal{H}_t = o_t \circ \tanh(\mathcal{C}_t)$$

FC-LSTM

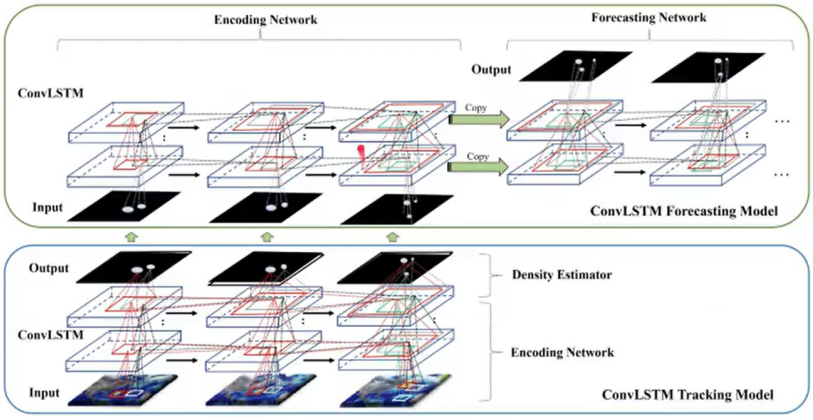
$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$
$$\mathbf{c}_t = \mathbf{f} \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t)$$

- ConvLSTM layers may be **stacked** multiple times, similar to multi-layer LSTMs.
- Applying to the **seq2seq structure**, inputs are encoded in the last hidden matrix (**H**), then decoder can **predict the future**.
- Applied to precipitation nowcasting.



- Hurricane Tracking & Forecasting

  - **Tracking**: density estimation from a climate video to hurricane probability
  - **Forecasting**: video prediction from the sequence of hurricane density



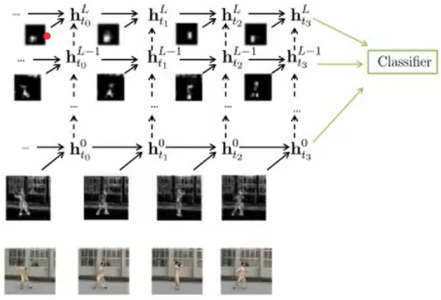
http://www.joonseok.net/papers/wacv19.pdf

- ConvGRU

- When stacking multiple ConvGRU layers, each layer takes
  - 1) 2D input feature (**x**),
  - 2) one-layer lower hidden state ($\mathbf{h}^{l-1}$)
  - 3) one-step previous hidden state ($\mathbf{h}_{t-1}$).
- This is different from common practice that each layer takes
  - 1) one-layer lower hidden state as its input ($\mathbf{x} = \mathbf{h}^{l-1}$), and 2) one-step previous hidden state ($\mathbf{h}_{t-1}$).

$$
\begin{aligned}
\mathbf{z}_l^l &= \sigma(\mathbf{W}_z^l * \mathbf{x}_t^l + \mathbf{W}_{z^l}^l * \mathbf{h}_t^{l-1} + \mathbf{U}_z^l * \mathbf{h}_{t-1}^l), \\
\mathbf{r}_t^l &= \sigma(\mathbf{W}_r^l * \mathbf{x}_t^l + \mathbf{W}_{r^l}^l * \mathbf{h}_t^{l-1} + \mathbf{U}_r^l \mathbf{h}_{t-1}^l), \\
\tilde{\mathbf{h}}_t^l &= \tanh(\mathbf{W}^l * \mathbf{x}_t^l + \mathbf{U}^l * (\mathbf{r}_t \odot \mathbf{h}_{t-1}^l), \\
\mathbf{h}_t^l &= (1 - \mathbf{z}_t^l)\mathbf{h}_{t-1}^l + \mathbf{z}_t^l \tilde{\mathbf{h}}_t^l,
\end{aligned}
$$

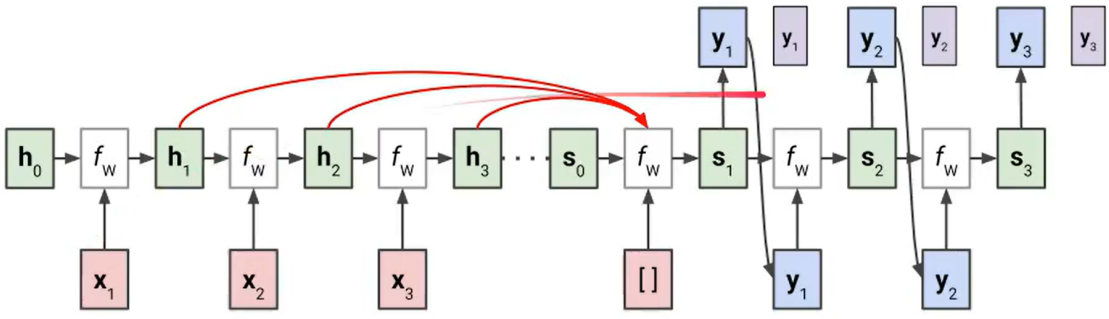
https://arxiv.org/pdf/1511.06432.pdf



Attention Mechanism

- Information Loss with RNN
  - 하나의 임베딩에다가 모두 저장한다는 것은 불가능
- Attention Idea

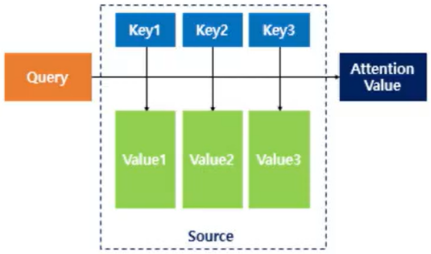
  - The decoder takes into account **hidden states at all input steps**, with closer attention on **more relevant input tokens**.



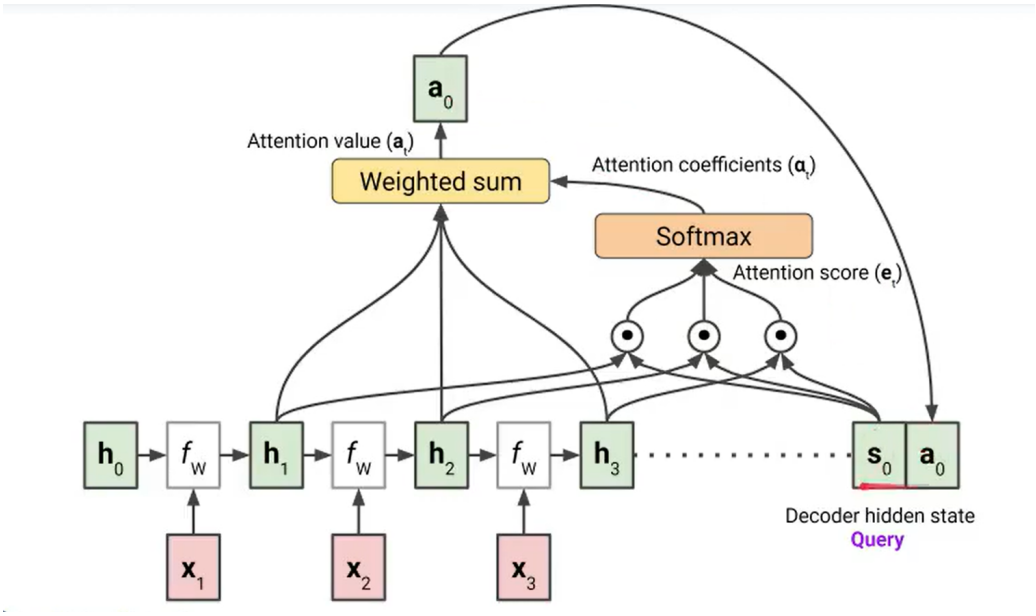
  - Attention function

- **Attention function**: Attention (**Q**, **K**, **V**) = Attention value
  For a query and reference key-value pairs, attention value is the weighted average of values, where each weight is proportional to the similarity between the query and the corresponding key.

- In a seq2seq model:
  - **Q** (Query): the **decoder** hidden state at time *t*
  - **K** (Keys): the **encoder** hidden states at all times
  - **V** (Values): the **encoder** hidden states at all times



- attention coefficient : query가 value와 얼마나 다른가를 score로 나타내고 이를 softmax로 normalize한 것
- attention value : weighted average of encoder hidden states



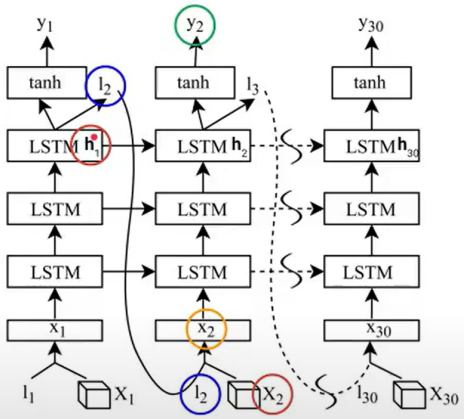
◦ 다른 attention method

- Multiple options for scoring similarity:
  - Dot-product: score(**s**, **h**) = $\mathbf{s}^T\mathbf{h}$
  - Learnable weighted dot-product: score(**s**, **h**) = $\mathbf{s}^T\mathbf{W}\mathbf{h}$
    - The weight matrix W is learned from data.
  - Concatenation and additional layer: score(**s**, **h**) = $W_2{}^T tanh(W_1[\mathbf{s}; \mathbf{h}])$
- Dot-product is simpler, with less parameters. In many applications, dot-product is a default choice as it is easier to train.

- Visual attention

- **Spatial** attention: where should we focus on the **2D image space** to classify the video correctly?
- At each time step,
  - Using the last LSTM's hidden representation ($\mathbf{h}_{t-1}$) as query, attends spatially on the last conv-layer representation of an input image ($\mathbf{X}_t$).
  - From this spatial attention, we get spatial attention coefficients ($\mathbf{l}_t$).
  - $\mathbf{X}_t$ and $\mathbf{l}_t$ are weighted-summed to $\mathbf{x}_t$ (attention value), then fed into 3 LSTM layers.
  - Outputs class prediction ($\mathbf{y}_t$), and continue with the last hidden layer ($\mathbf{h}_t$) for the next frame.

https://arxiv.org/pdf/1511.04119.pdf

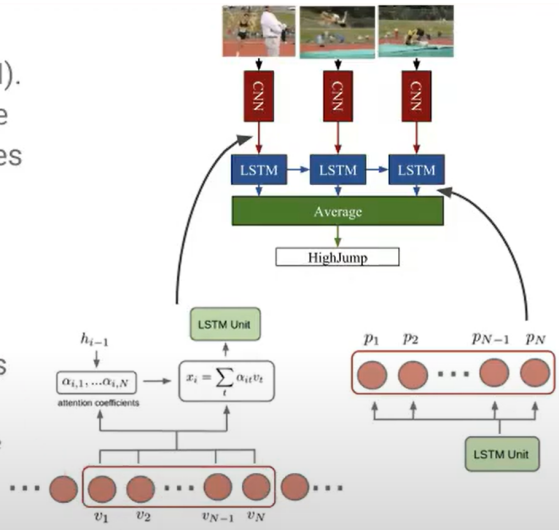


MultiLSTM

- multiLSTM

- Basically, the model is similar to the LSTM-based video classifier (e.g., LRCN).
- **Multiple input**: instead of taking a single frame as input to the LSTM cell, it applies attention to *N* recent frames.
  - **Query**: previous hidden state $\mathbf{h}_{t-1}$ of LSTM
  - **Key**, **Value**: *N* recent input frame features
  - **Attention value**: weighted sum of recent *N* frame features
- **Multiple output**: each LSTM cell outputs predictions on *N* most recent frames.
  - These are averaged later to finalize per-frame prediction.

https://arxiv.org/pdf/1507.05738.pdf



Case study

Dataset & open-source TensorFlow code

- http://research.google.com/youtube8m/
- https://github.com/google/youtube-8m/

7 Million	450,000	3.2 Billion	4716	3.4
Video URLs	Hours of Video	Audio/Visual Features	Classes	Avg. Labels / Video

Kaggle Competition (2017, 2018, 2019)

- https://www.kaggle.com/c/youtube8m
- $100,000 prize pool
- $30,000 in Google Cloud credits for participants

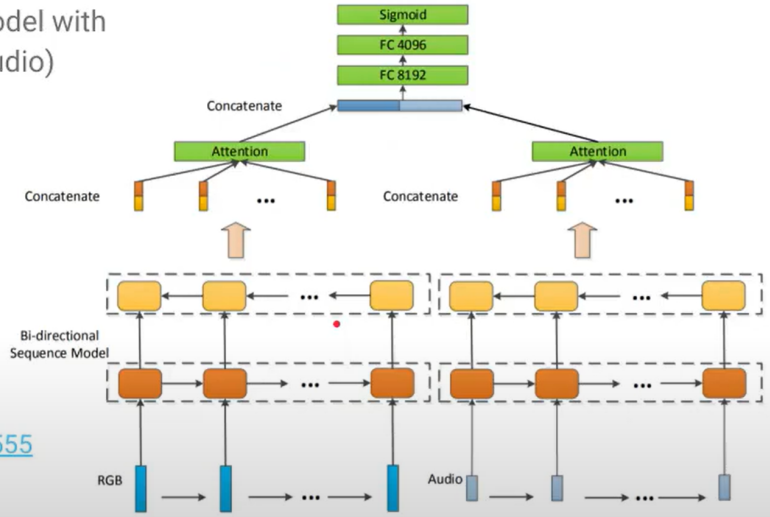
742 Teams **926** Competitors

CVPR'17, ECCV'18, ICCV'19 Workshops

- https://research.google.com/youtube8m/workshop2017/index.html
- https://research.google.com/youtube8m/workshop2018/index.html
- https://research.google.com/youtube8m/workshop2019/index.html

- proposed ideas

- **Bidirectional LSTM** model with two streams (visual, audio)



https://arxiv.org/abs/1707.04555
Team offline, 2017 3rd place