VIDEO Classification

Snehashis MAJHI

Email: snehashis.majhi@inria.fr

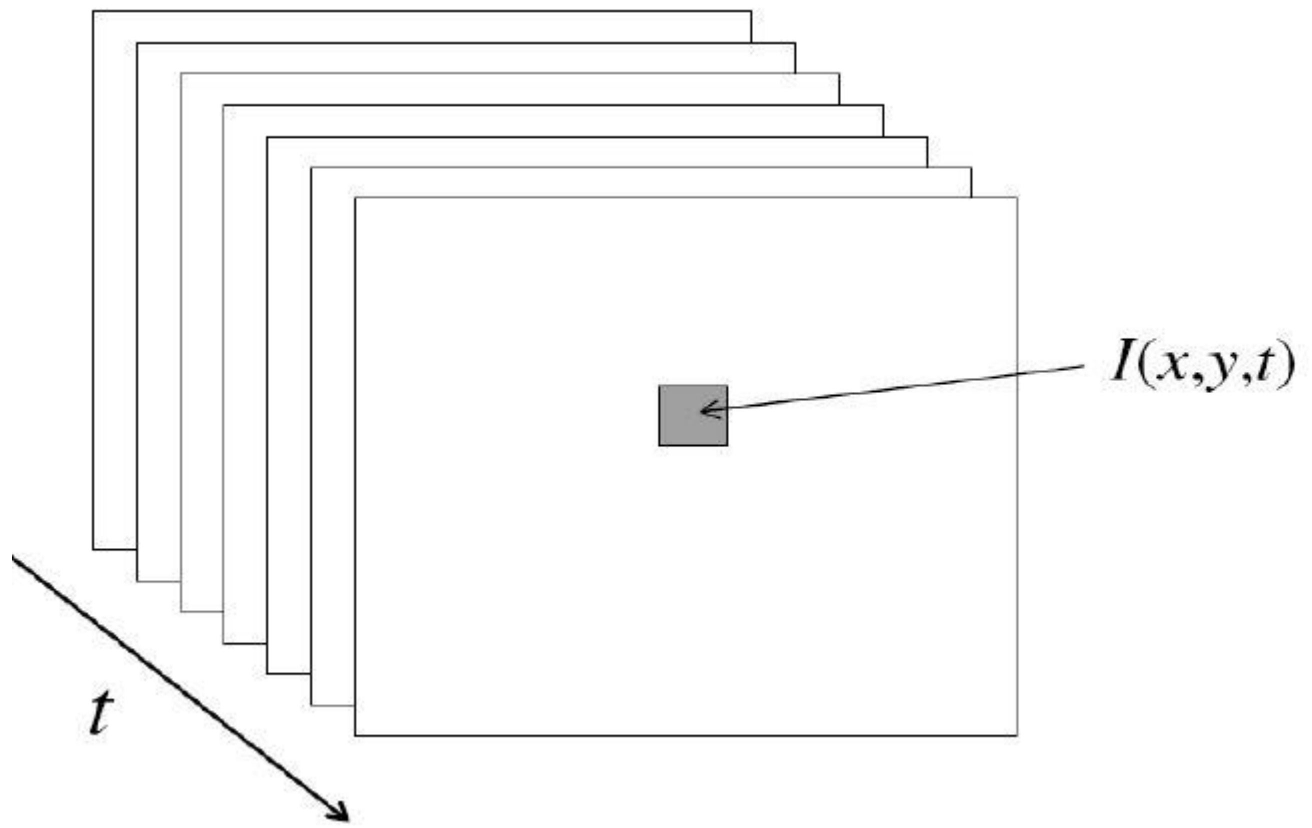
Ph.D. Candidate @STARS Team INRIA

Collaboration with TOYOTA Motor Europe

TABLE OF CONTENT

Video ::

- Formally, a **video is a 3D signal** with:
    - **Spatial Coordinates:** x, y
    - **Temporal Coordinates:** t



- If we fix ‘t’, we obtain an image (a.k.a frame). So video can be seen as a **sequence of Images/Frames.**

Real-world Applications ::

Data:

~2.5 Billion new images / month

TV-channels recorded since 60's

~5K image uploads every min.

>34K hours of video upload every day

~30M surveillance cameras in US
=> ~700K video hours/day

And even more with future wearable devices

Real-world Applications ::

Applications:

First appearance of N. Sarkozy on TV

Sociology research: Influence of character smoking in movies

Education: How do I make a pizza?

Where is my cat?

Predicting crowd behavior
Counting people

Motion capture and animation

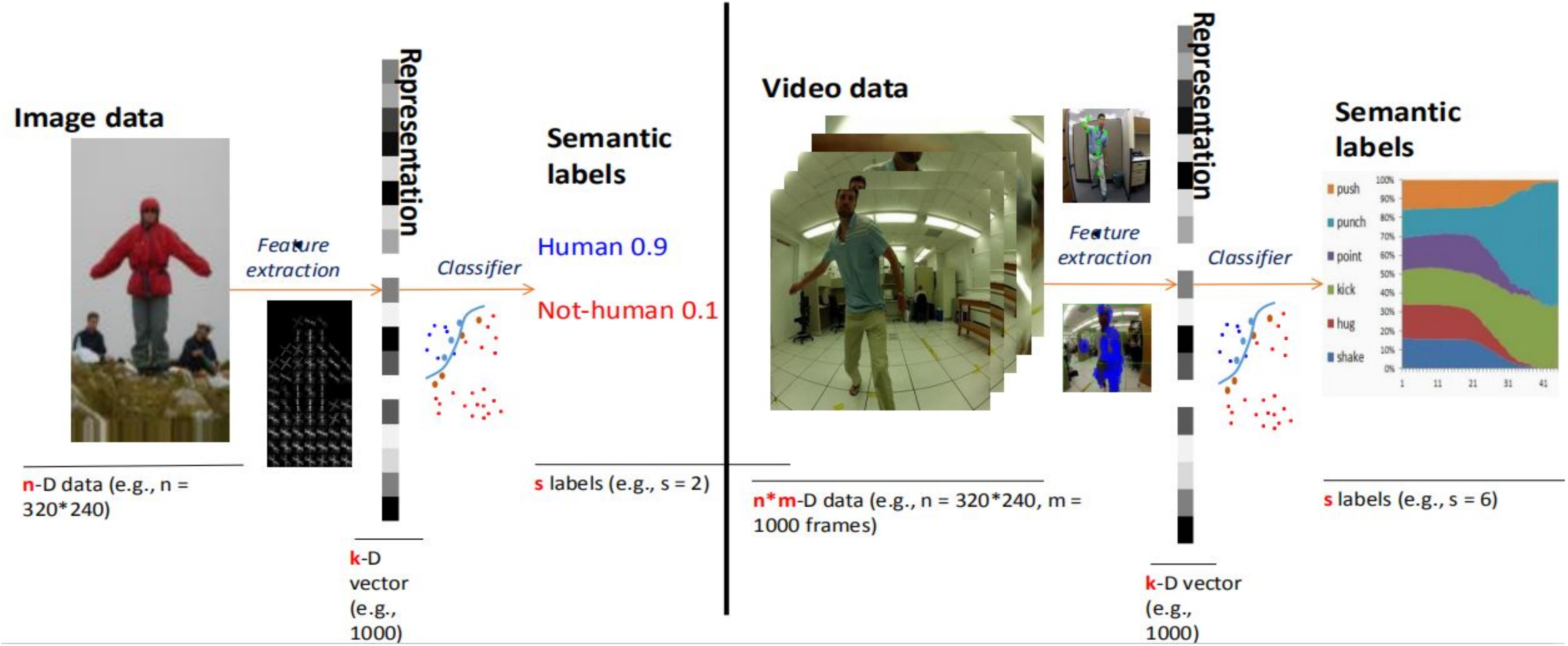
Real-world Applications ::

Amazon go

Assistive Robot

Waiter Robot!

Image Vs. Video Classification ::

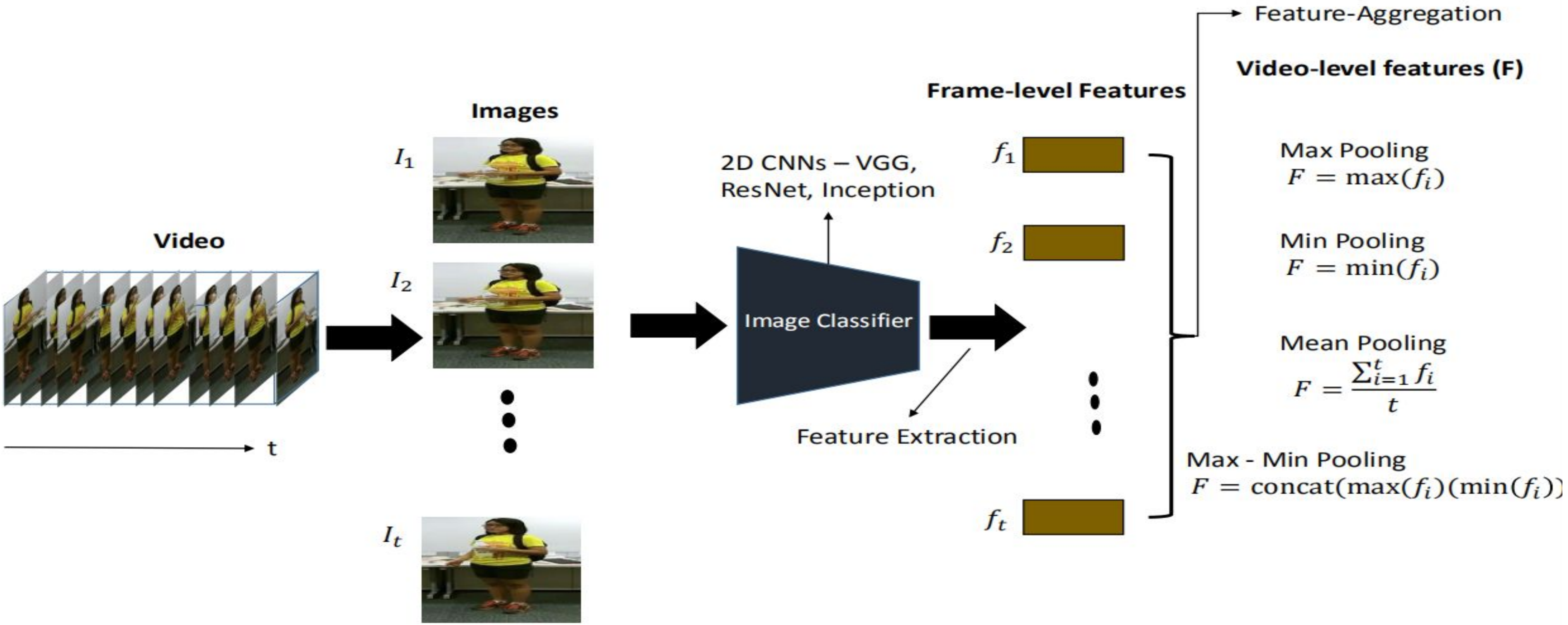


Video Classification Techniques ::

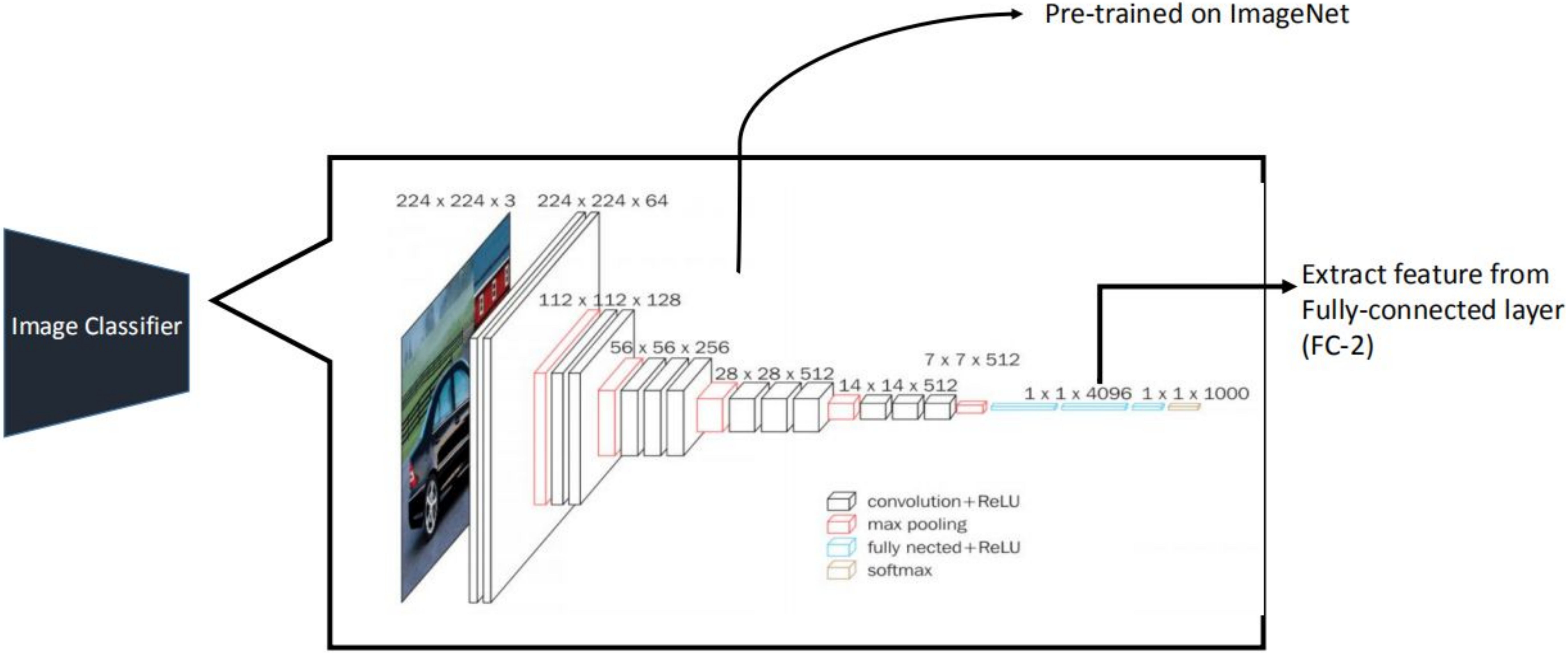
Evolution of Research

1. **Frame-level aggregation of 2D Convolutional Networks**
   a. Aggregating the frame-level information using pooling
   b. Temporal information is lost
2. **Two-Stream 2D Convolutional Networks**
   a. Perform convolution separately on both spatial and temporal modalities
   b. Complexity involved in obtaining multiple modalities
3. **Recurrent Neural Networks and Temporal Convolution Networks**
   a. Model the temporal evolution of the frames using gating functions and 1D convolutional kernels respectively
   b. Do not handle space-time simultaneously
4. **3D Convolutional Networks**
   a. Perform convolution across space-time simultaneously
   b. Too rigid to capture subtle information

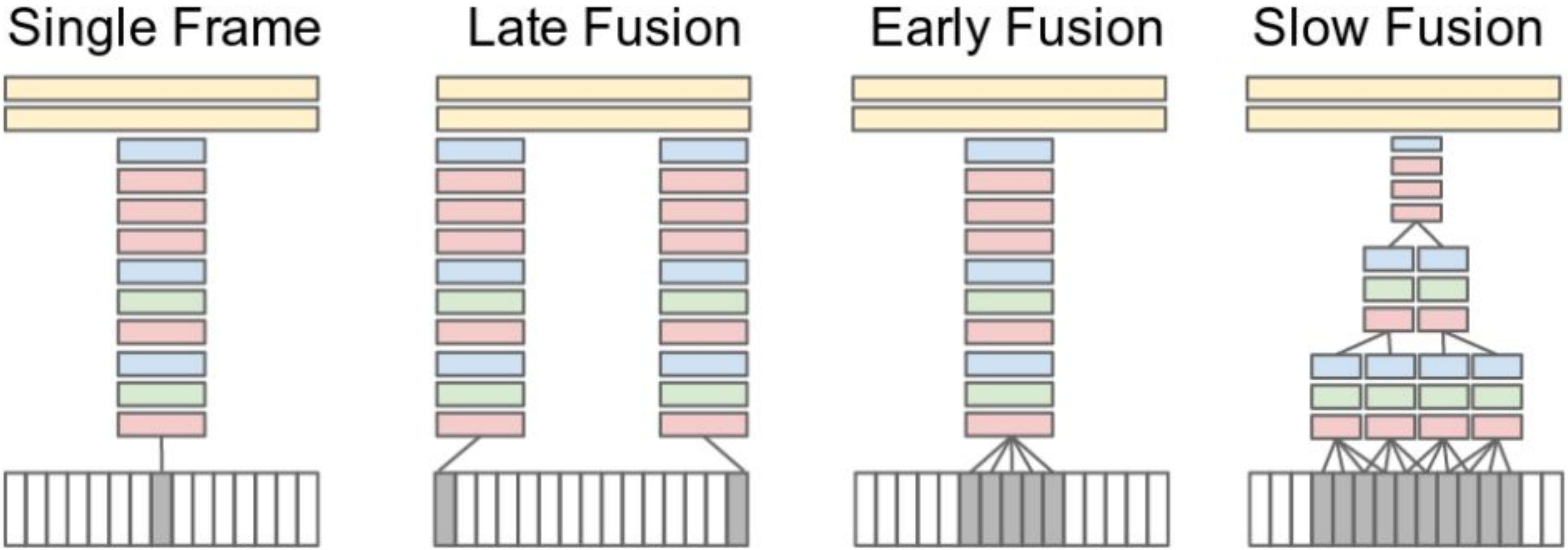
1. Frame-Level Aggregation of 2D CNN ::



How to Extract Frame-Level Features?



Types of Frame-level Feature Aggregation



Observation:

- These frame-level pooling mechanisms provide a video descriptor which **encourages the salient frames in the video**.

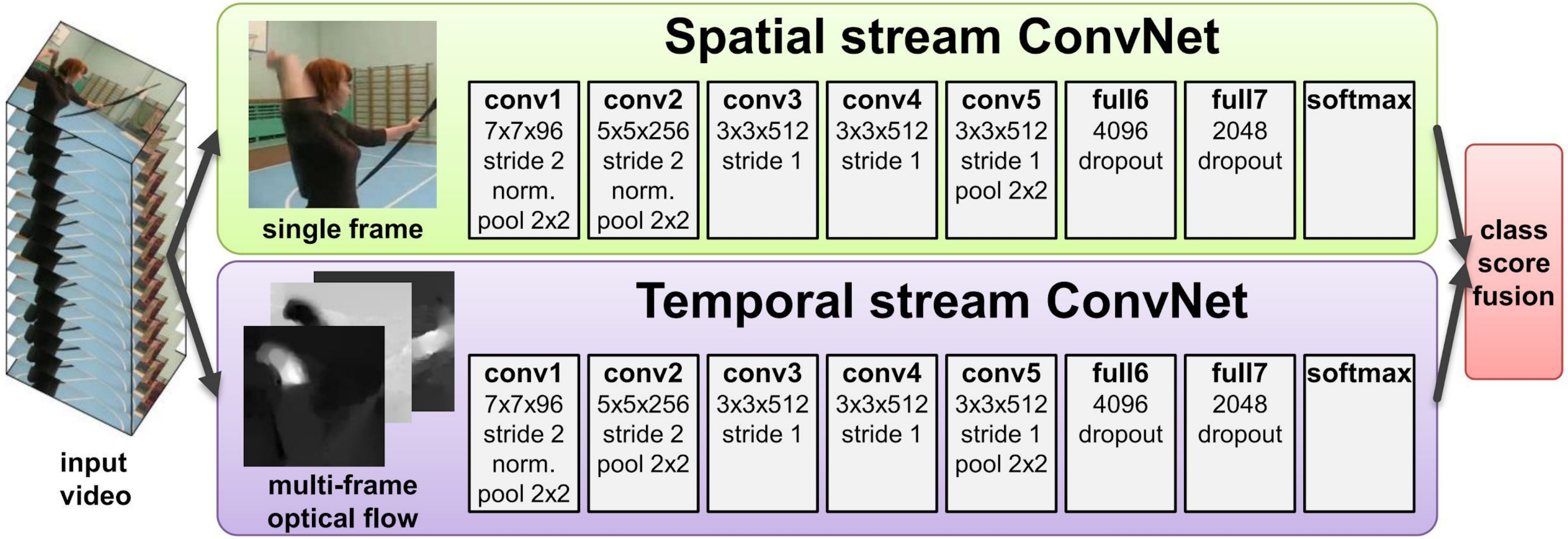
- The video descriptors for each videos are treated as data samples for a classifier (like SVM) for classifying the videos.

- These **video descriptors do not model temporal information and only relies on the salient frame-level features**.

- Then how should we model temporal information???

2. Two Stream 2D CNN ::

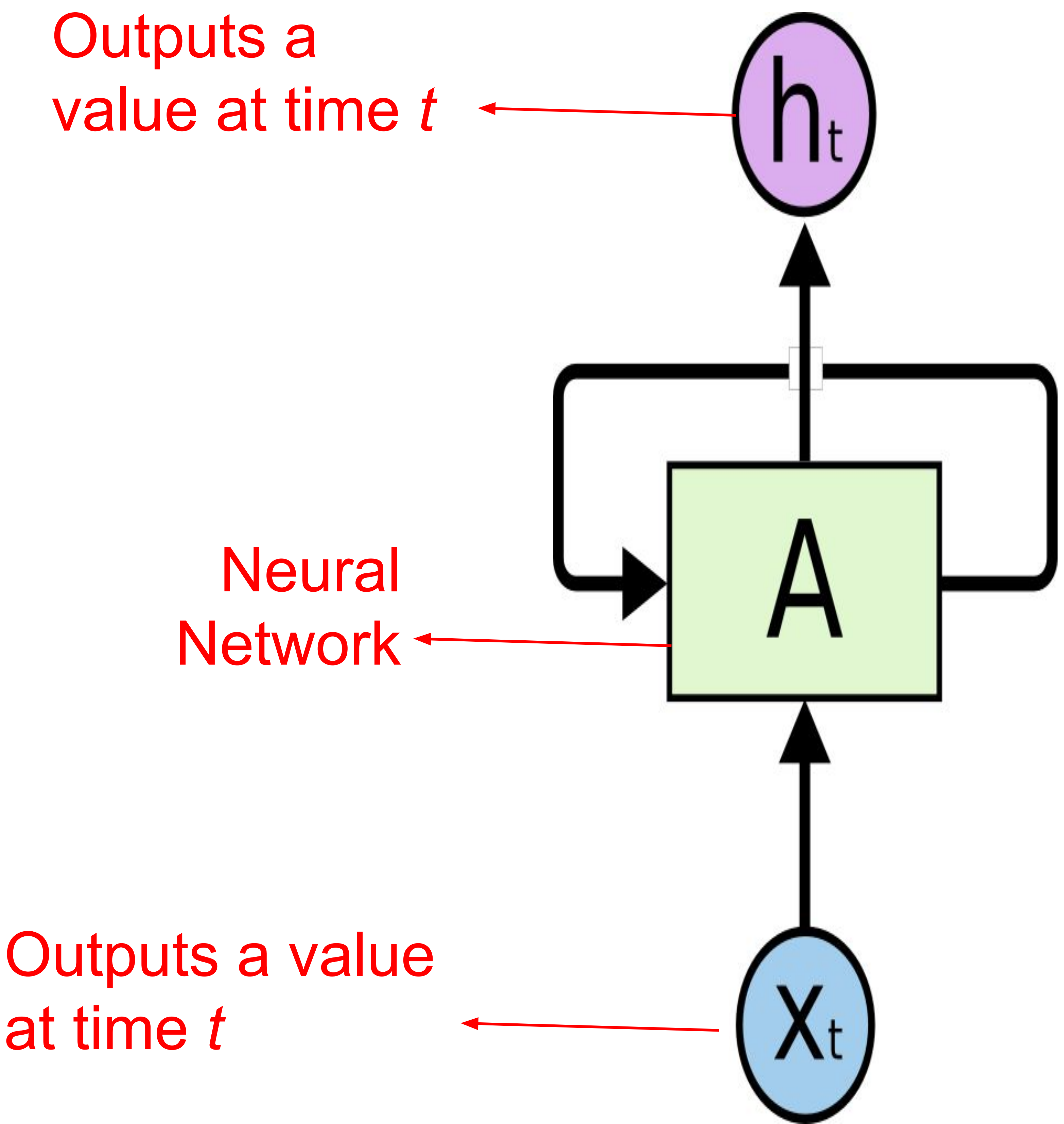
- Idea: To combine both Appearance and motion representations.
- Previous work: Failed because of the difficulty in learning implicite motion.



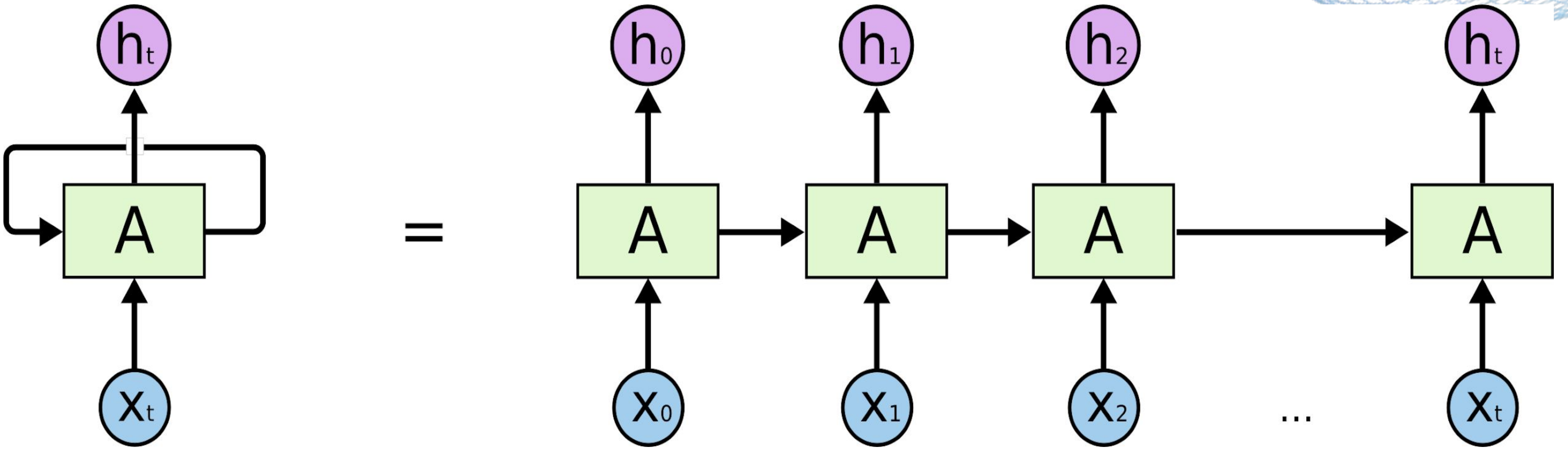
- Separate the ***Motion*** *(multi-frame)* from ***static appearance*** *(single frame).*
- **The appearance and motion stream are not aligned.**
- Optical flow can only capture shirt tern temporal dynamics

3. Recurrent Neural Network::

- RNNs address the issue of temporal dependency modeling in videos.
- They are networks with loops in them, allowing information to persist.
- A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.



3. Recurrent Neural Network::



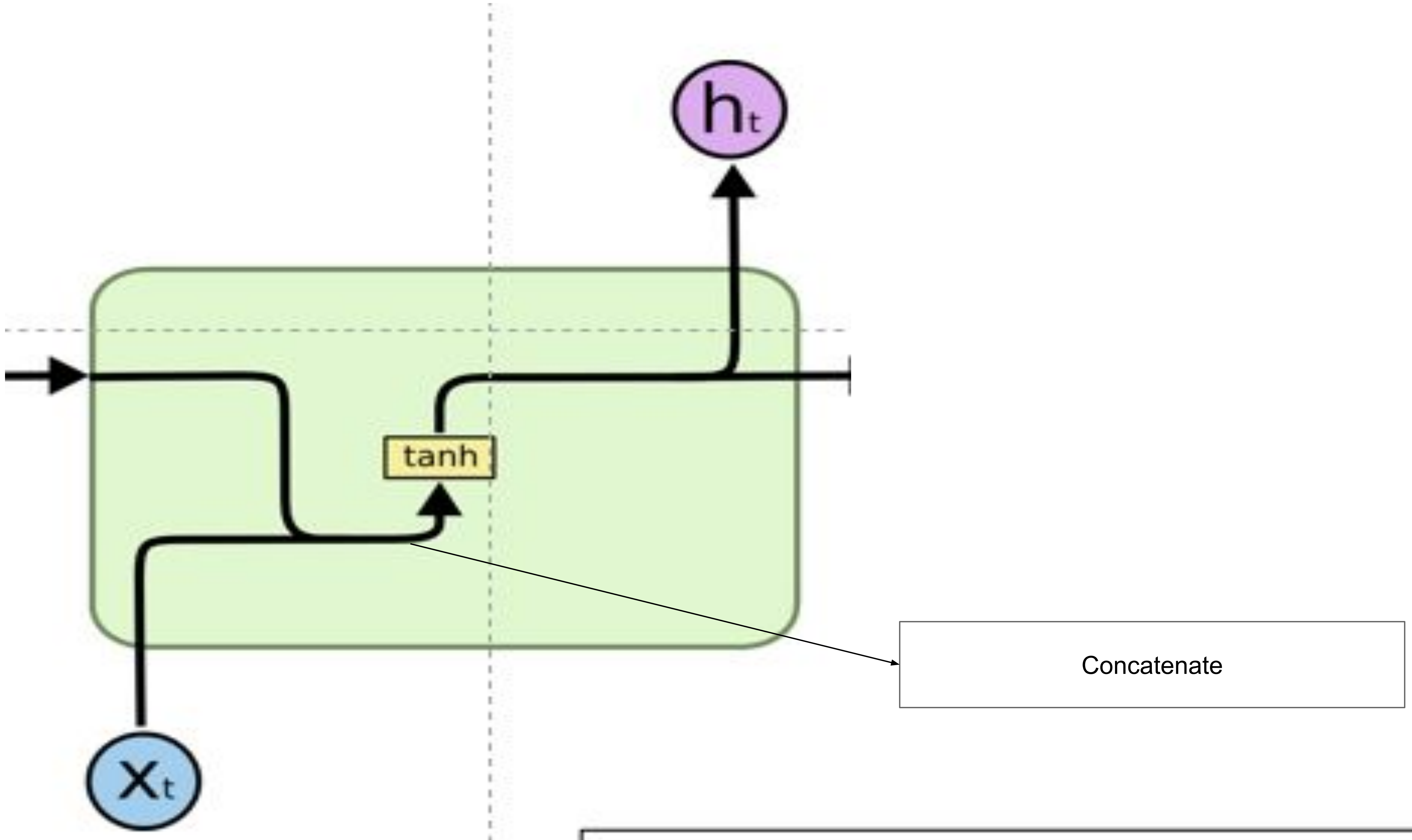
$$\boxed{h_t = f_W(h_{t-1}, x_t)}$$

A typical example

$$\boxed{h_t = tanh(W_{hh}h_{t-1}, W_{hx}x_t)}$$

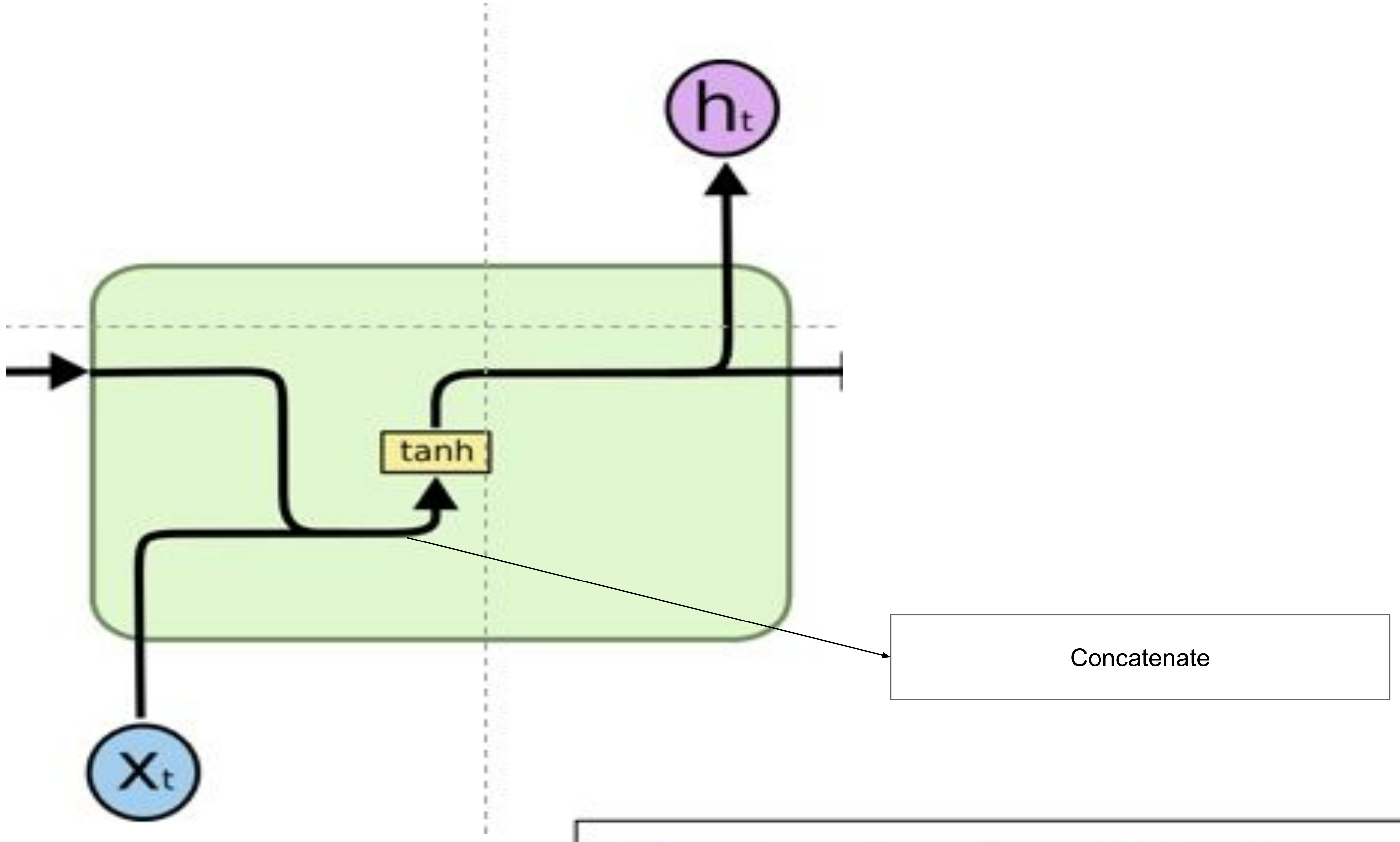
Some function with parameter W

3. Single RNN Unit::



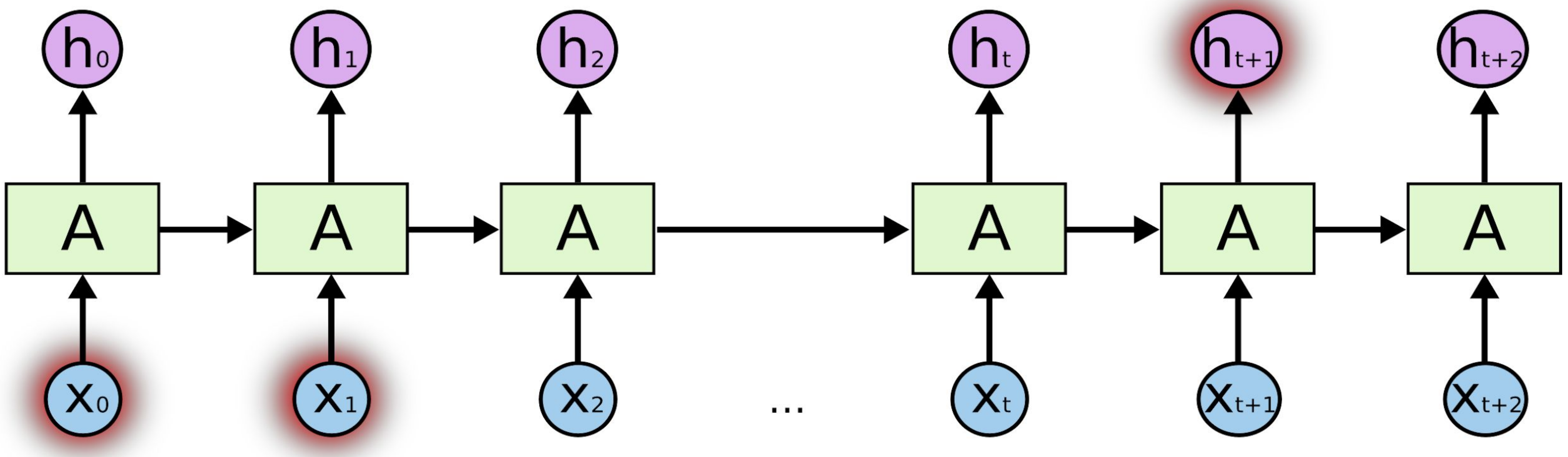
$$h_t = tanh(W_{hh}h_{t-1}, W_{hx}x_t)$$

3. Single RNN Unit::



$$h_t = tanh(W_{hh}h_{t-1}, W_{hx}x_t)$$

3. Limitation of RNN ::



Not capable of learning long-term dependencies because of gradient vanishing factor.

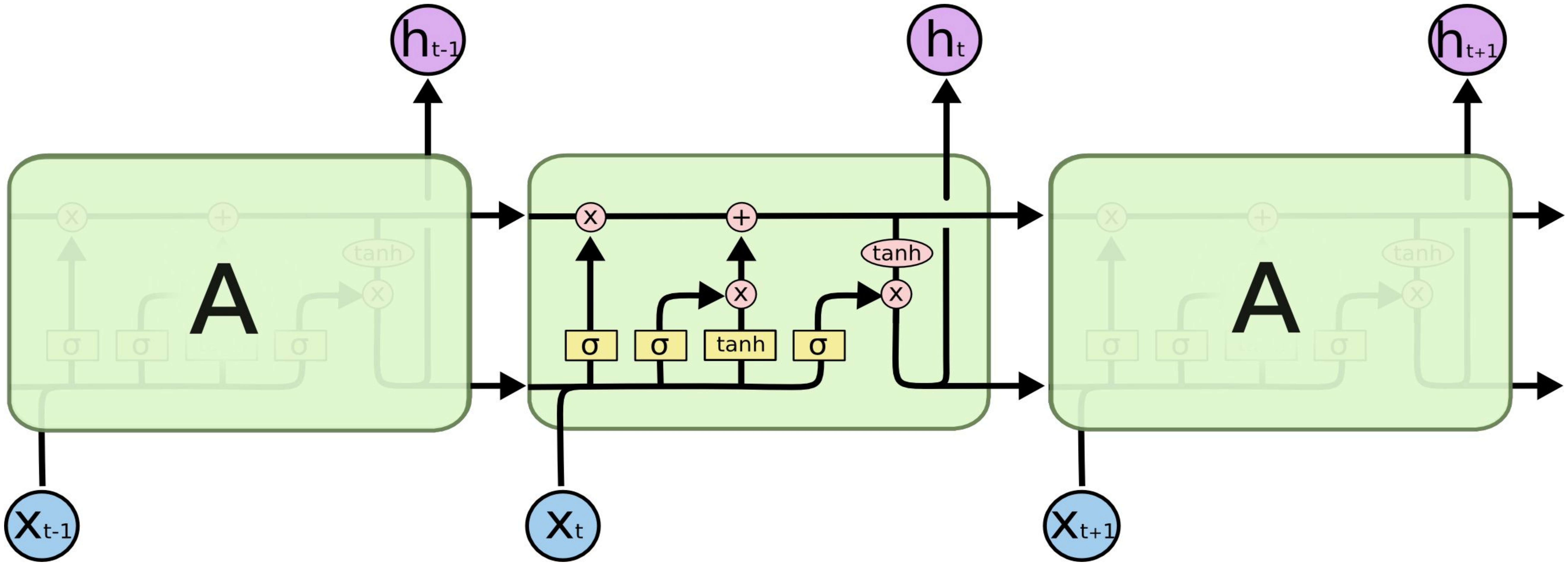
3. Long-short Term Memory (LSTM)::

Two major characteristics of LSTM:

- **Information Persistence :** Done using Cell States. These are like conveyor belts that runs across time through which information flows.

- **Prioritizing Information :** This means which deciding information is useful for future and which are useless and can be erased. Done using gates similar to digital logic, but are controlled by neural networks.

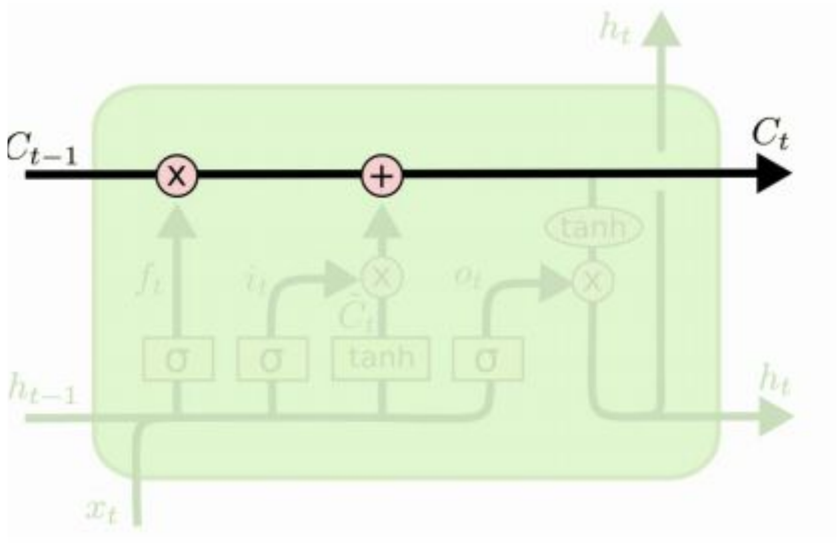
3. Long-short Term Memory (LSTM)::



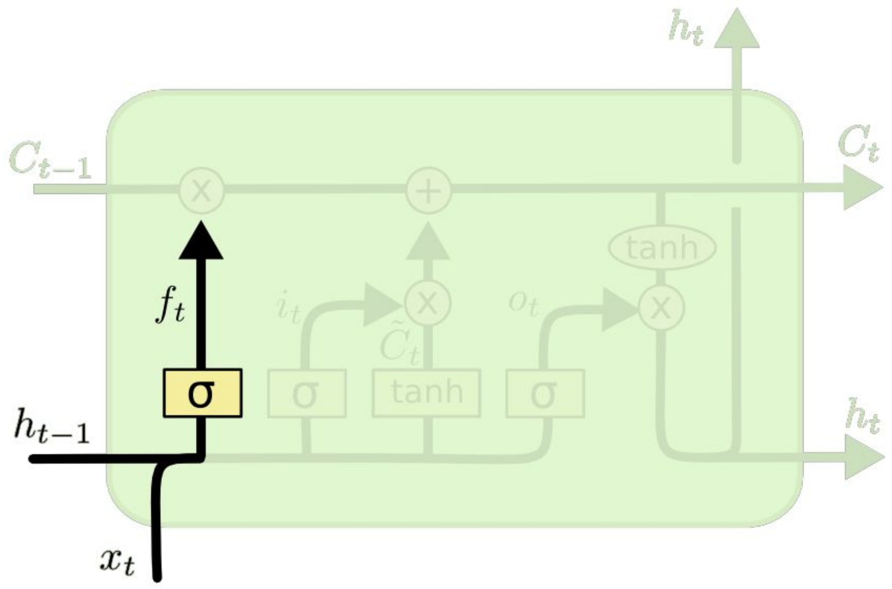
3. Different Modules of LSTM::

Four Major modules

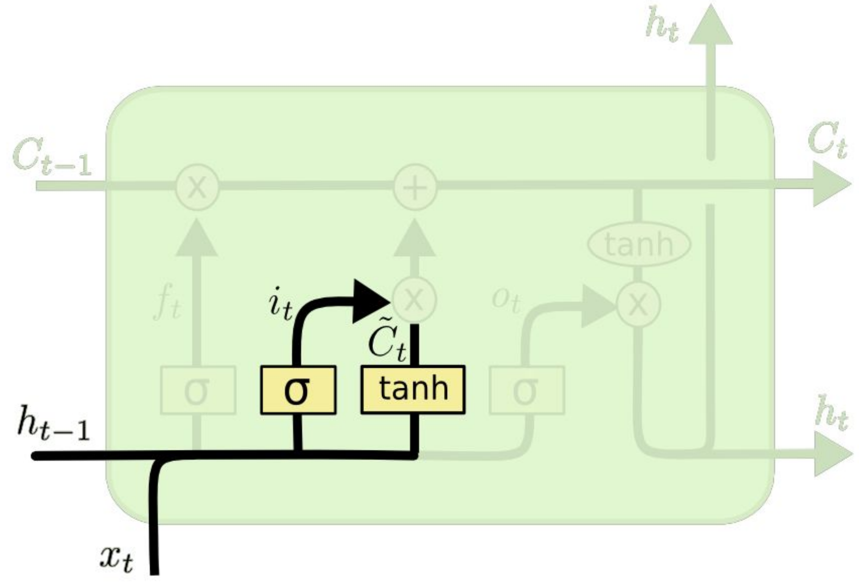
1.Cell State



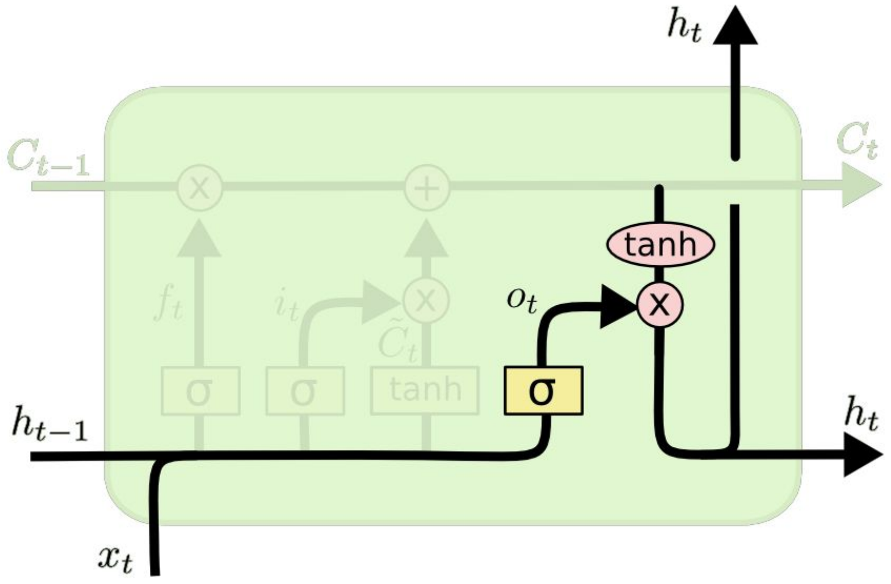
2. Forget Gate



3.Input Gate

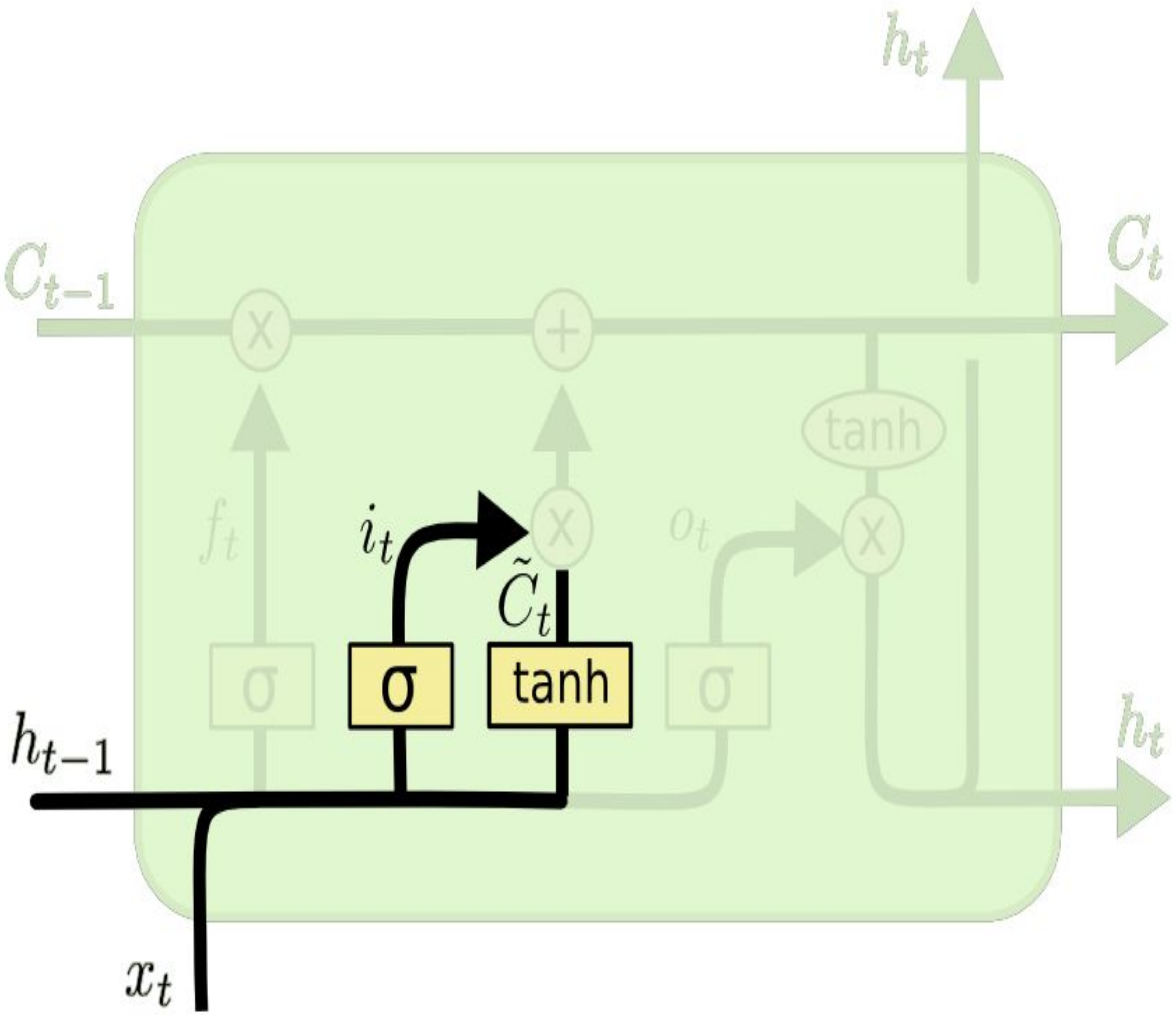


4.Output Gate



3. Working of LSTM::

Input Gate : This gate selects which of the new information is useful.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

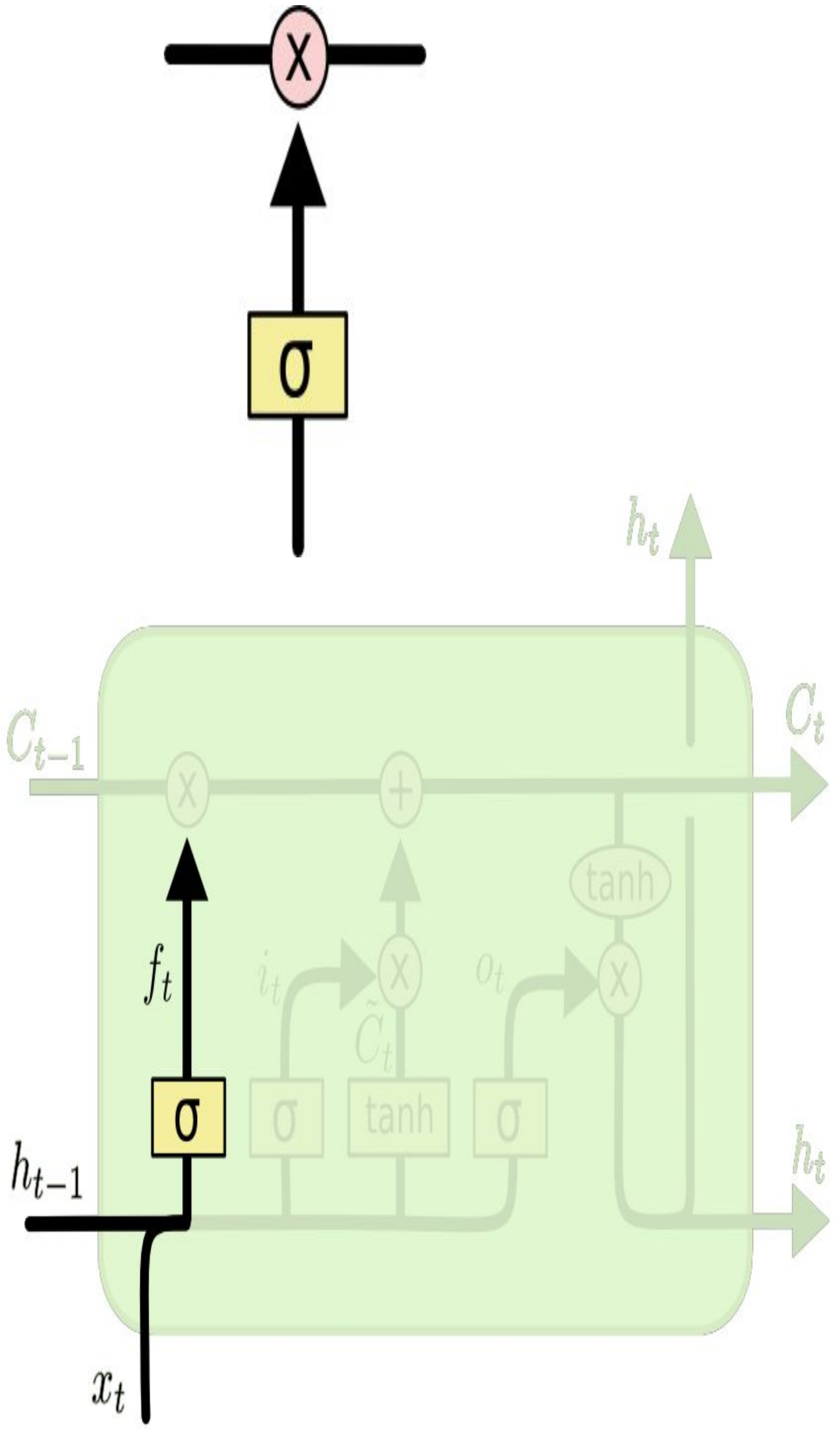
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

3. Working of LSTM::

Forget Gate :

- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

- The first step in the LSTM is to decide what information we're going to throw away from the cell state.



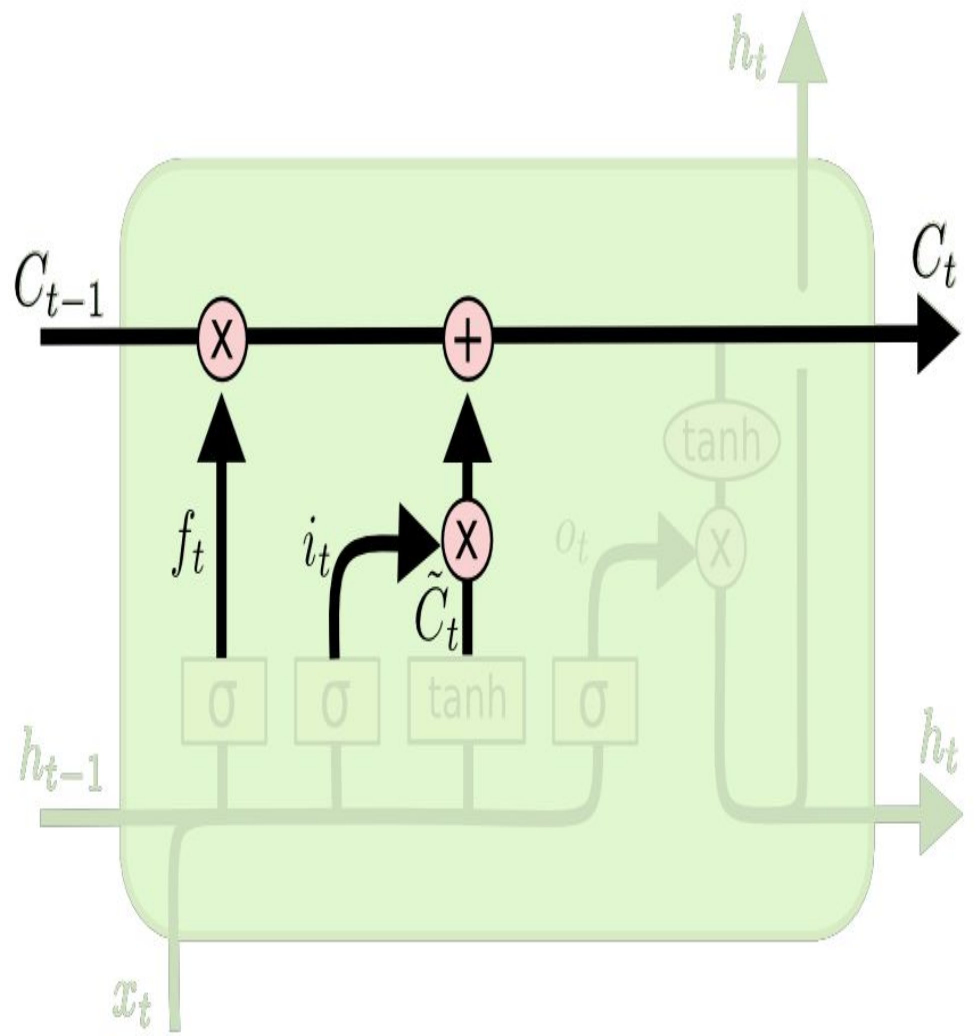
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

3. Working of LSTM::

Cell State :

- It's now time to update the old cell state, C_{t-1}, into the new cell state C_t

- The horizontal line, the cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

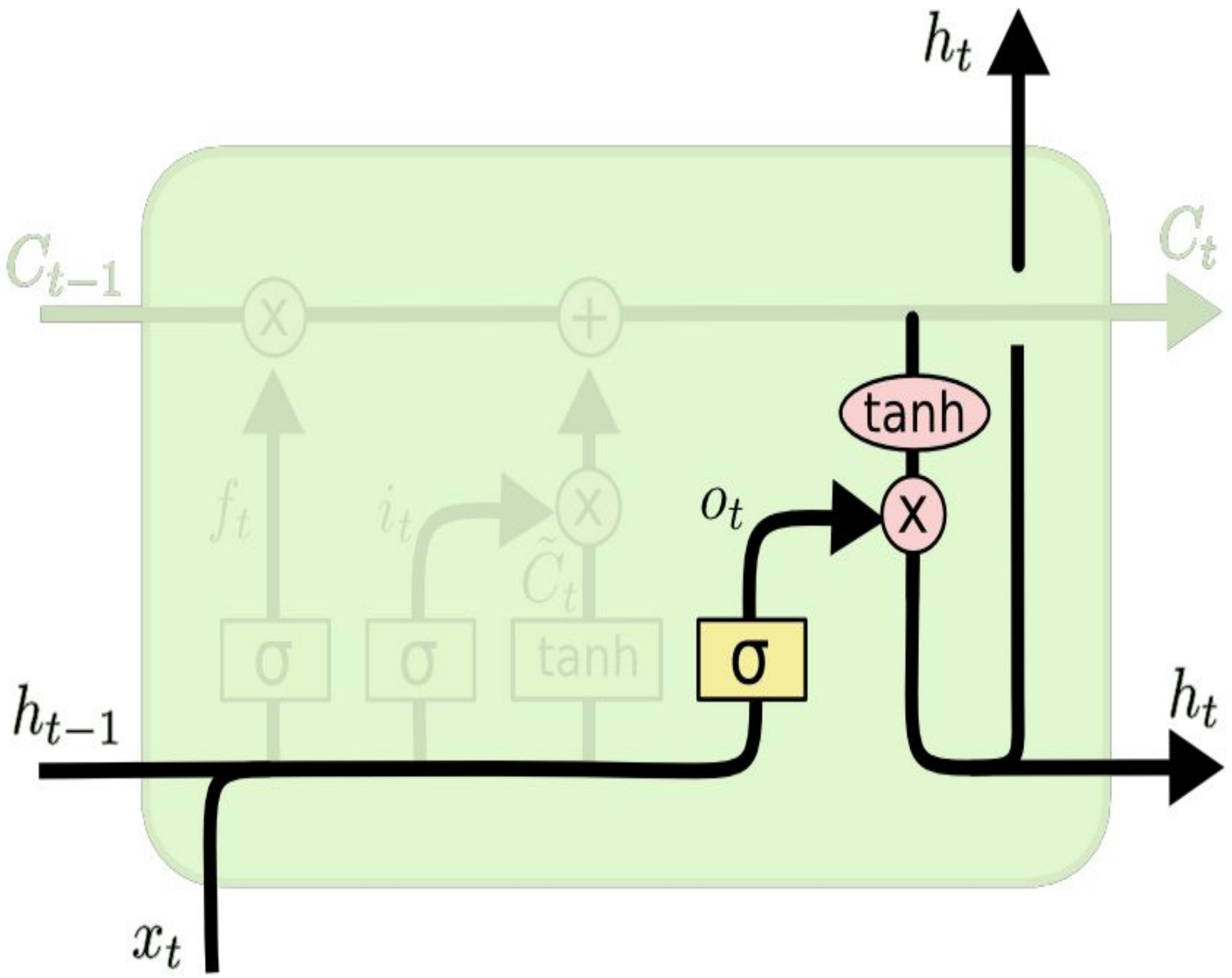


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3. Working of LSTM::

Output Gate :

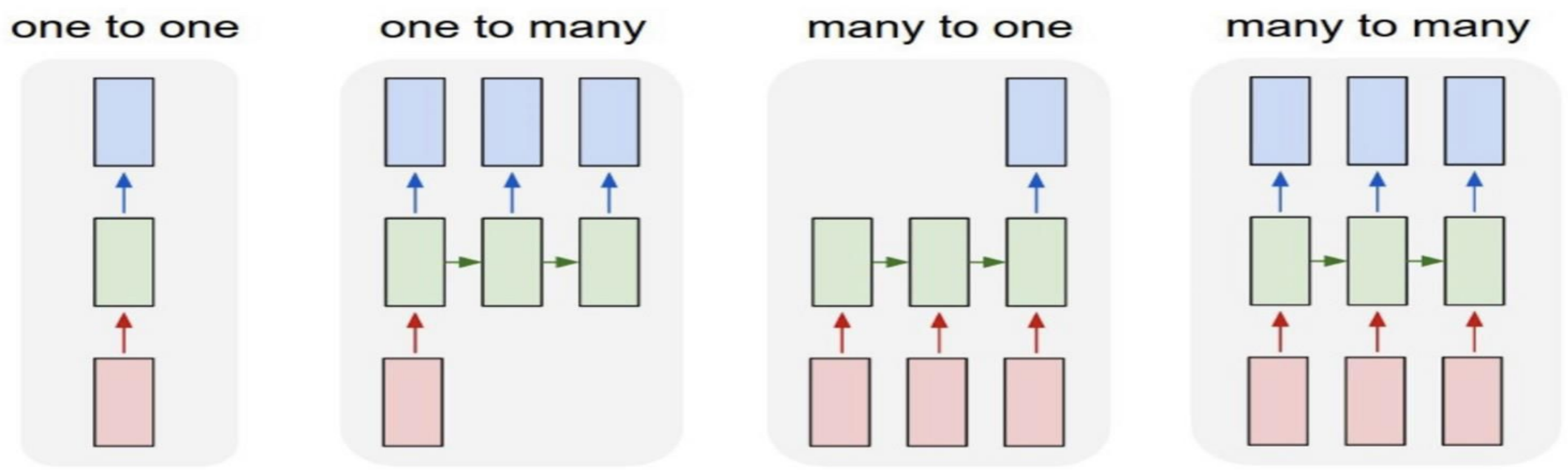
- Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

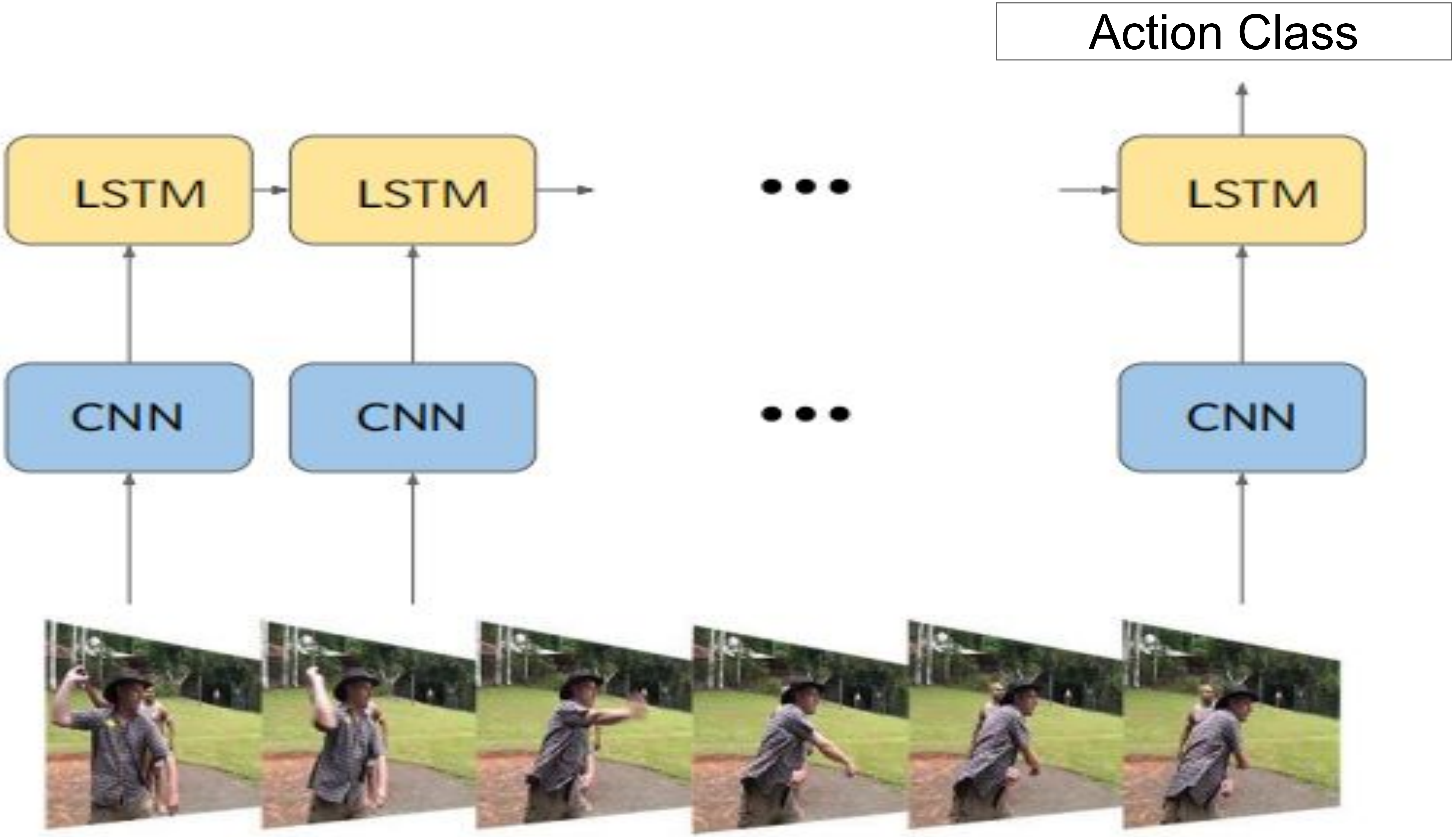
$$h_t = o_t * \tanh\left(C_t\right)$$

3. Types of LSTM::



Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification)	From fixed-sized input to Sequence output (e.g. image captioning: takes an image as input and outputs a sentence of words)	From Sequence input to fixed-sized output (e.g. Video Classification: takes sequence of frames/images as input and outputs a class label)	From Sequence input to Sequence output (e.g. Video Event Detection: takes sequence of frames/images as input and outputs a sequence event labels for each frame)

3. Temporal Dependency modeling with LSTM::

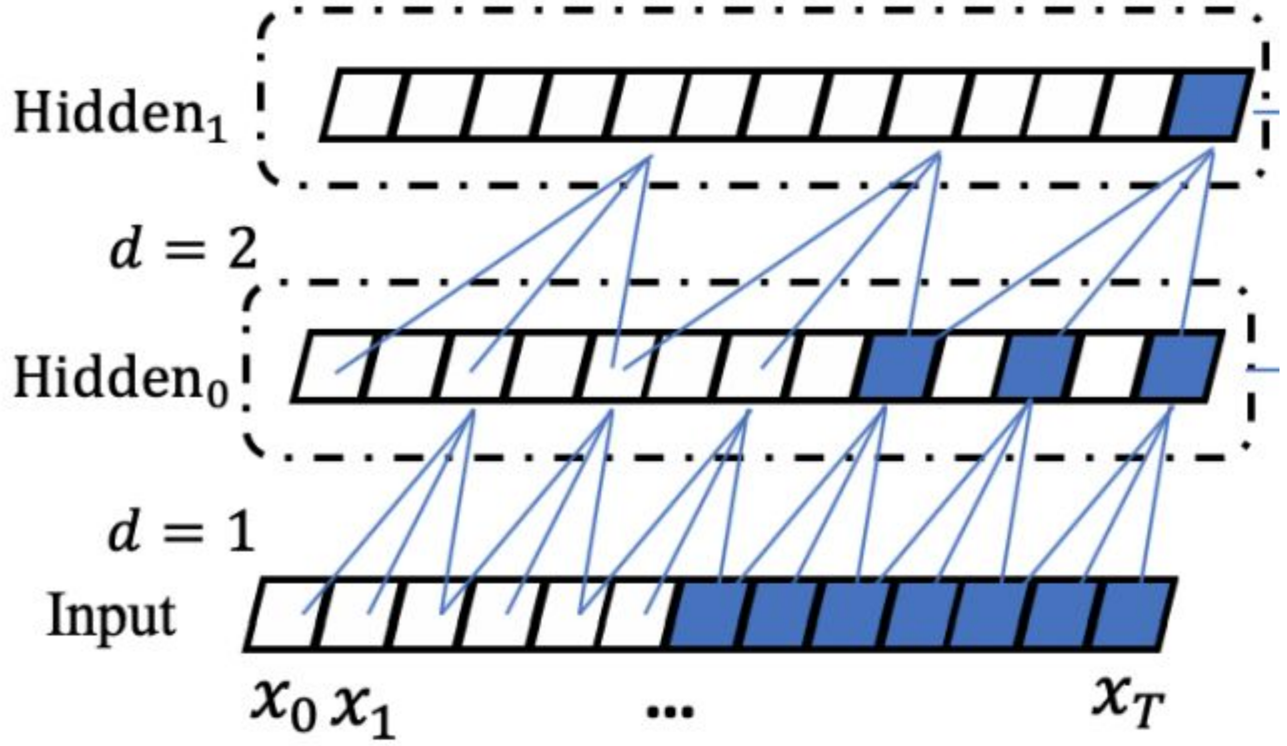


3. Drawback of RNN/LSTM::

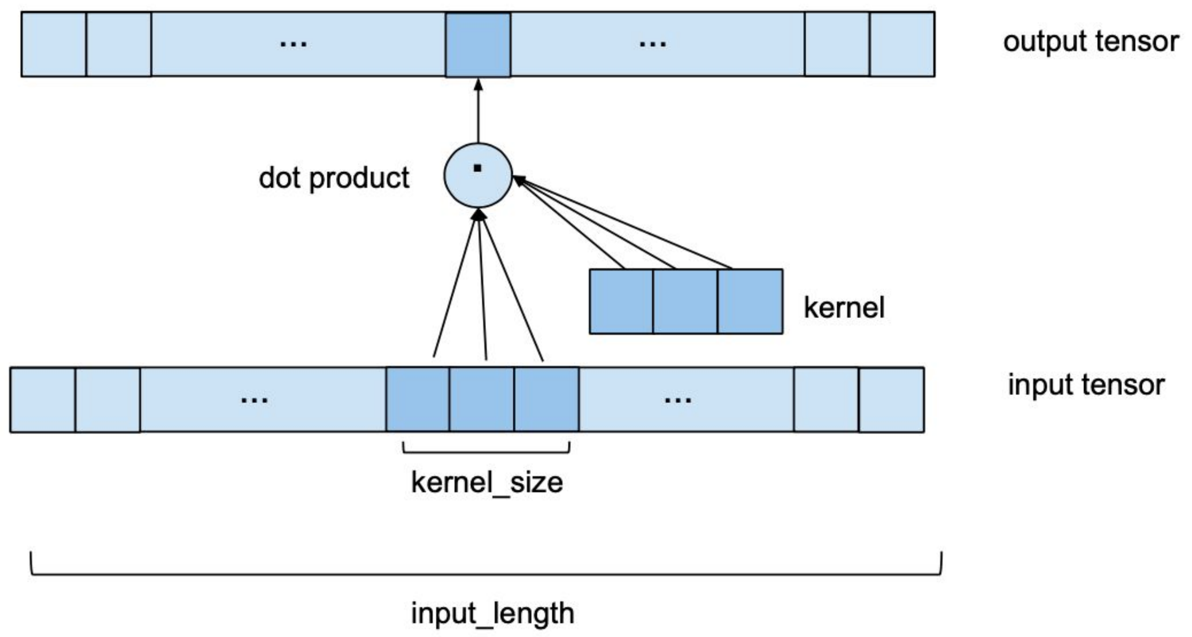
- RNN/LSTM are **sequential and can not be parallelized**.
- RNNs/LSTMs **can only capture strong temporal change of the image level features** and the subtle features are ignored.
- **Vanishing gradient issue** (Can not remember long term temporal information).
- **Not much efficient on small datasets** (pre-training is not a good idea as they change the statistics learned by the gates).

3. Temporal Convolution Network (TCN)::

- TCN encodes temporal dependencies by learning 1D convolution filters across temporal dimension.



- Inputs and outputs a 3-dimensional tensors.
  - Input shape: (**Batch_size**, **Temporal_length**, **Feature_size**) and
  - output shape: (**Batch_size**, **Temporal_length**, **Output_size**).

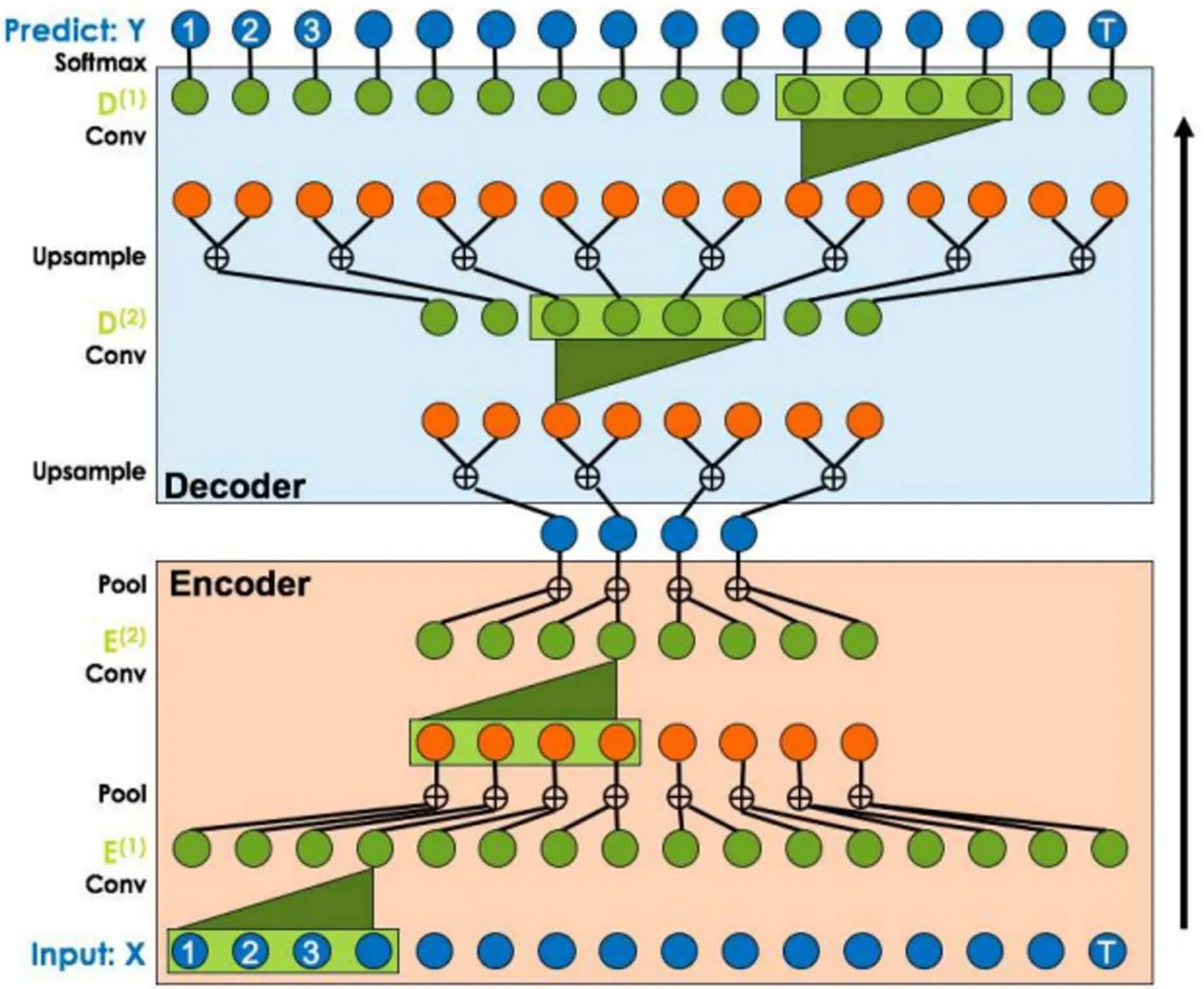


- TCN can be causal (no information leakage from the future to the past)

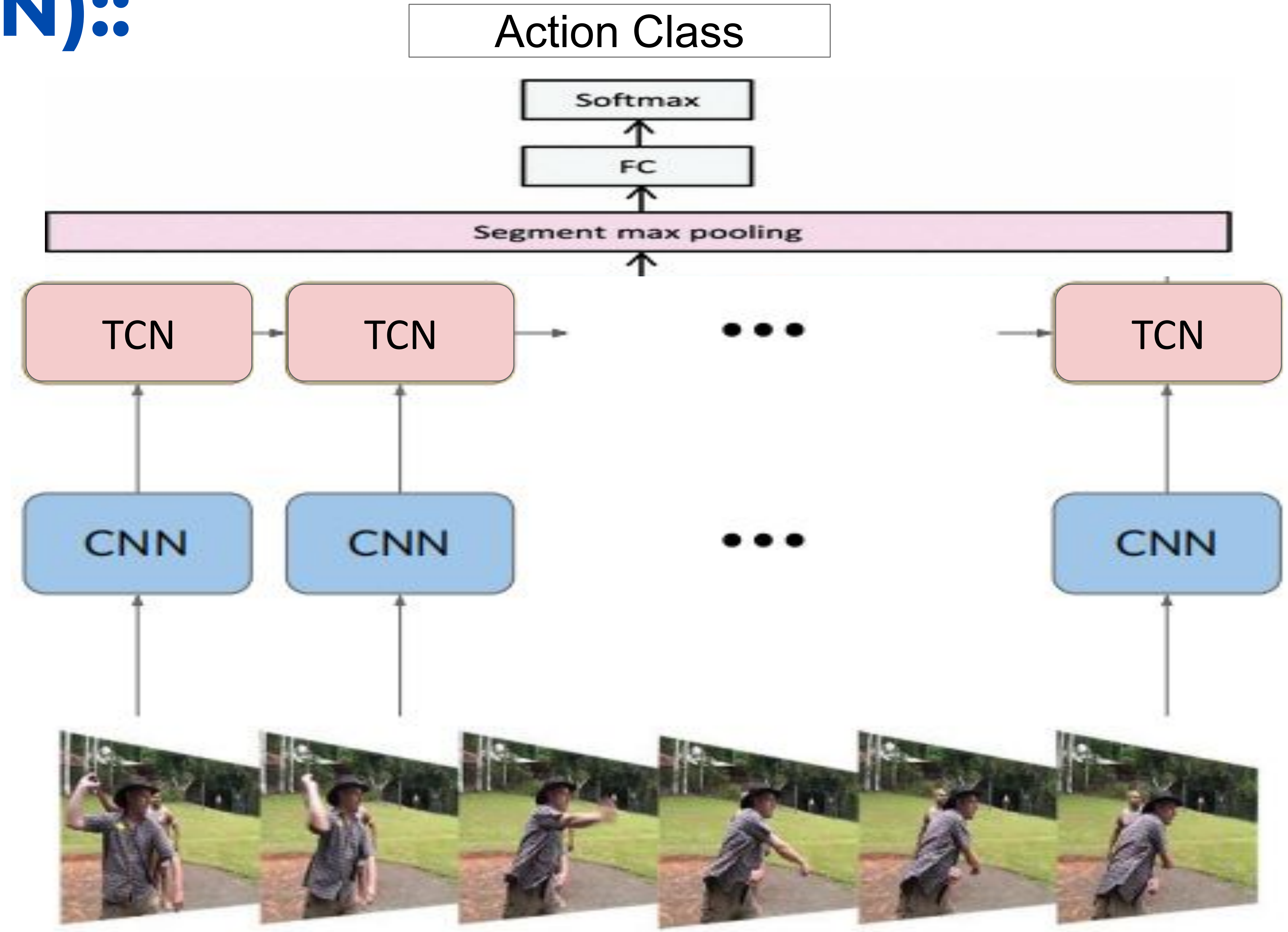
- TCN can use a very-deep network with the help of residual connections, and it can look very far into the past to predict with the help of dilated convolutions

3. Temporal Convolution Network (TCN)::

- TCN can follow Encoder-Decoder design to model the dependency among temporally neighbour and distant feature maps.



3. Temporal Convolution Network (TCN)::



3. TCN Vs. LSTM::

- Parallelism
- Flexible Receptive Field Size
- Stable Gradient
- Low Memory Requirement
- Knowledge Transfer between Domain can be possible

- NO Parallelism
- Fixed Receptive Field Size
- Vanishing Gradient Problem
- High Memory Requirement as it maintain Hidden State
- Not Possible to Knowledge Transfer between Domain(Pre-training LSTM is not a good Idea)

4. 3D Convolutional Neural Networks::

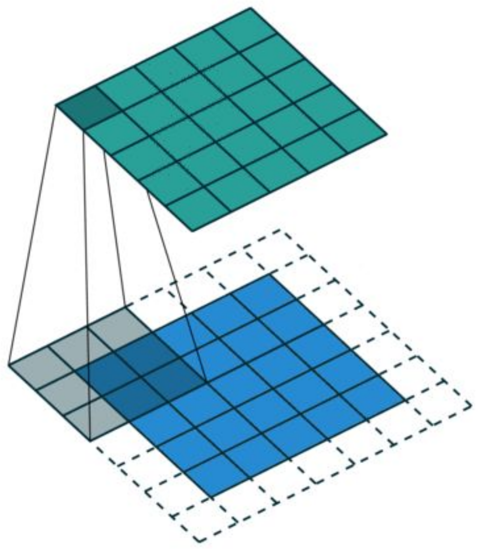
- 3DCNN uses three dimensional convolution filters to capture spatio-temporal features in a short-snippet of video.

2D Convolution (XY)

Input: [,,]

Output: [, , *#Kernel*]

Kernel move in H,W direction



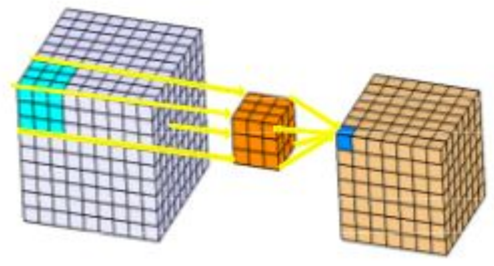
$$H_{out} = \frac{H_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$

$$W_{out} = \frac{W_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$

3D Convolution (XYT)

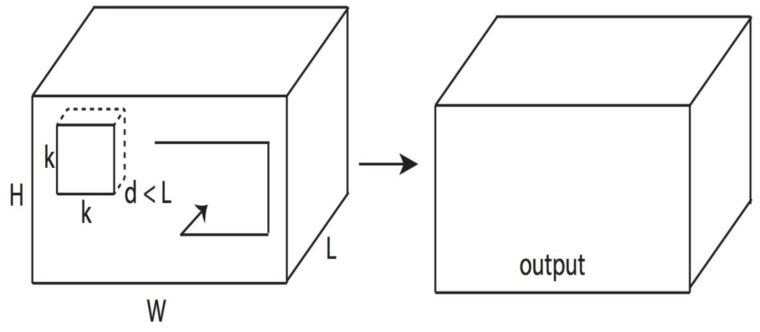
Input: [,,,]

Output: [, , , *#Kernel*]

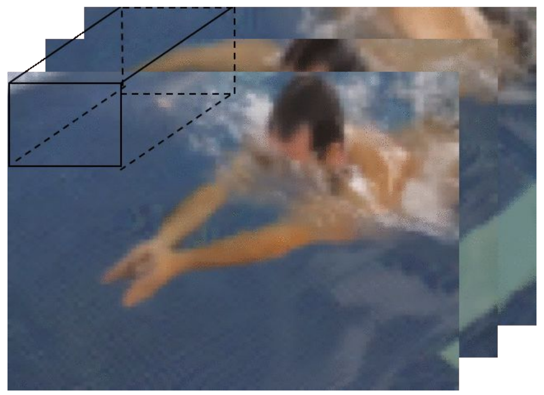


$$H_{out} = \frac{H_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$

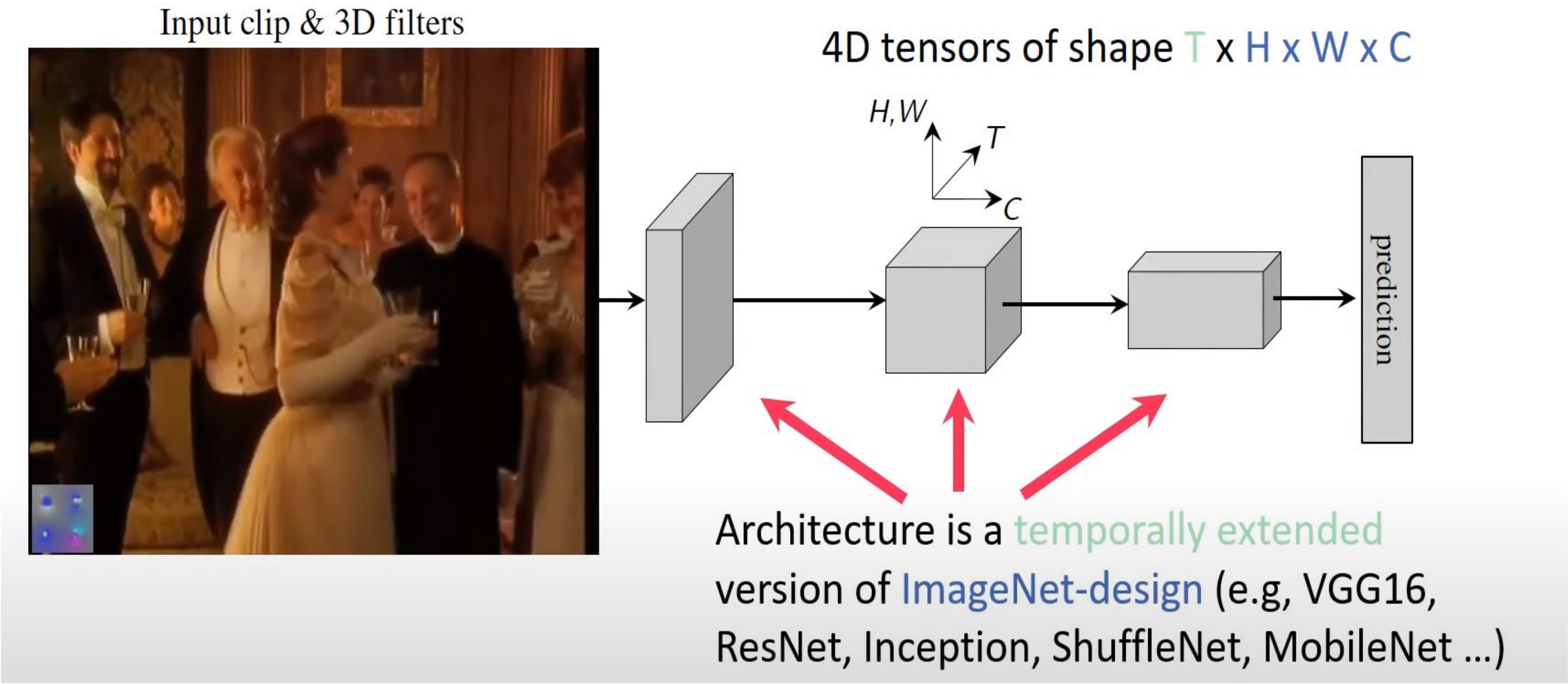
$$W_{out} = \frac{W_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$



$$\boxed{T_{out} = \frac{T_{in} + 2 \times padding - dilation \times (kernel_{size} - 1) - 1}{stride} + 1}$$

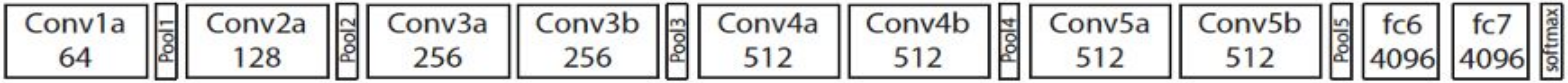


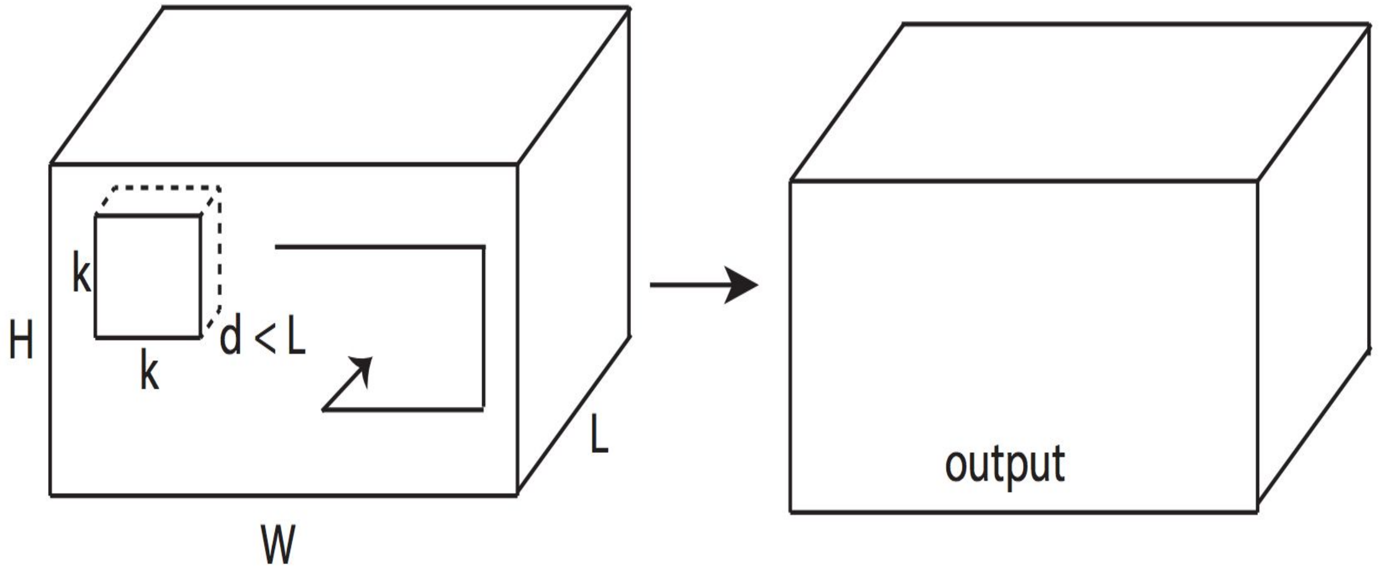
4. 3D Convolutional Neural Networks::

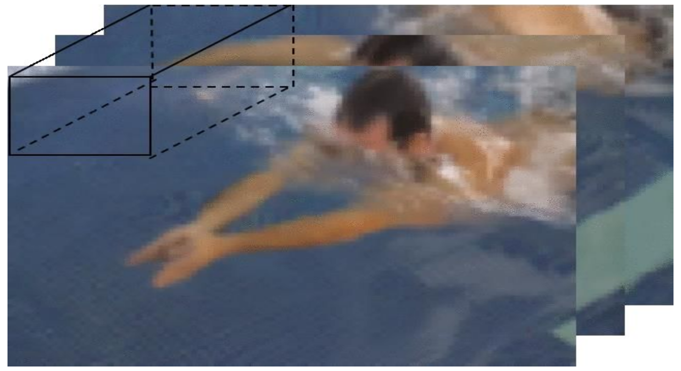


4. C3D Architecture::

- C3D contains 3 x 3 x 3 convolutional kernels followed by 2 x 2 x 2 pooling at each layer.
- The network architecture contains 8 convolutional, 5 pooling layers and 2 fully connected layers.
- It considers 16-frames snippets to extract spatio temporal feature representation.







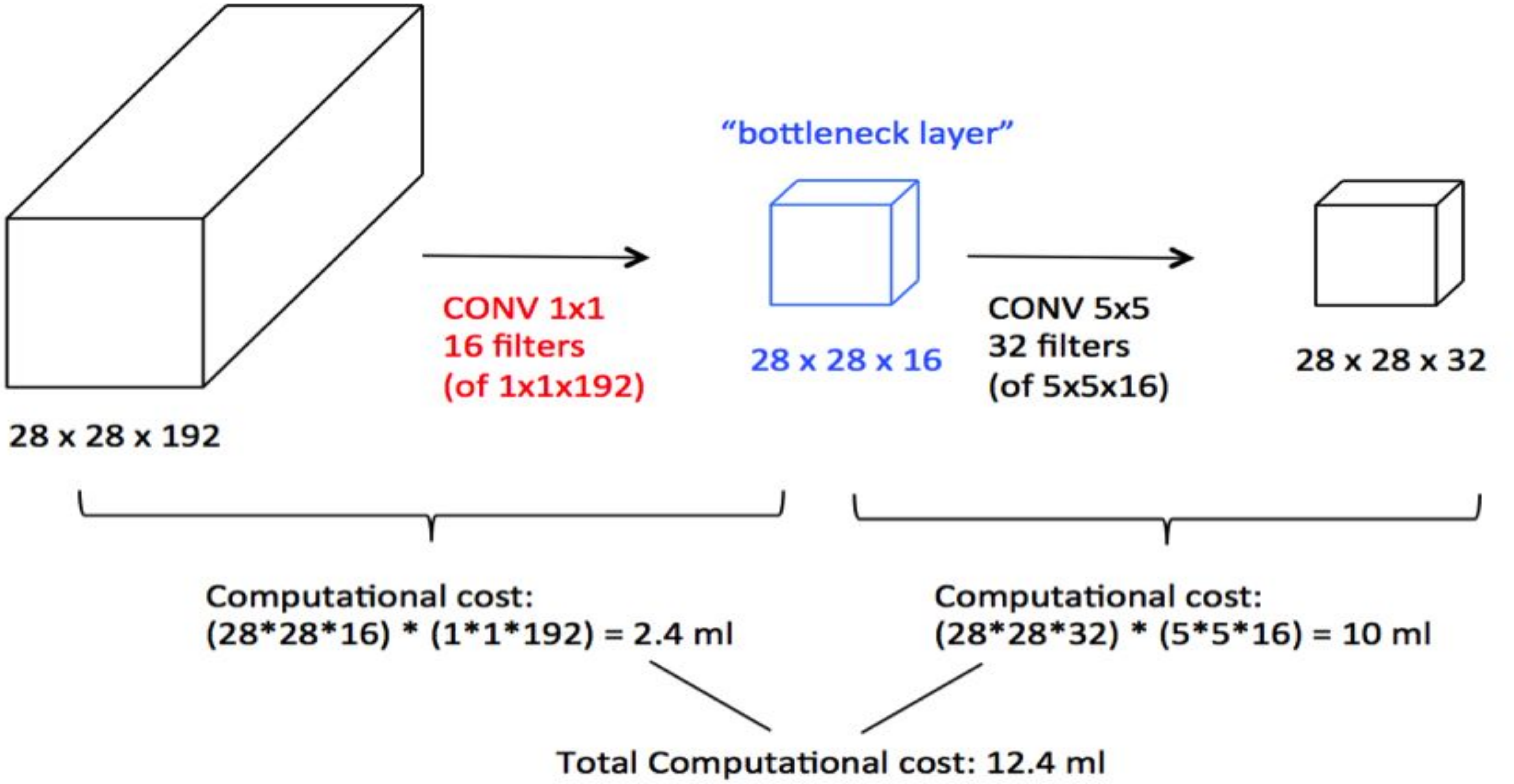
C3D is Temporally extended version of VGG16

4. I3D Architecture::

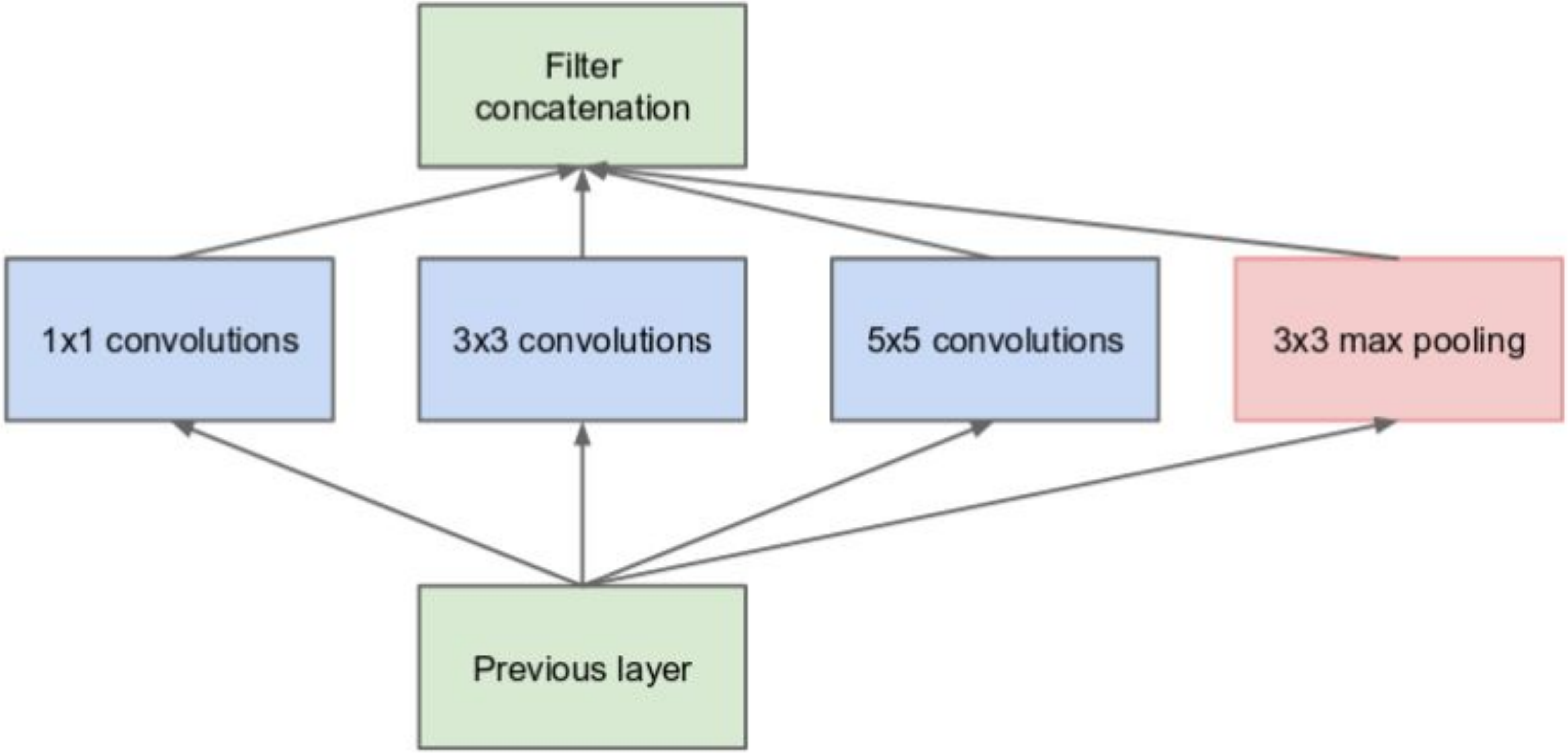
- I3D is designed by replacing the 2D kernels of GoogleNet by 3D kernels.
- It is extended by inflation from the spatial domain.
- Unlike C3D it allows branching in the network architecture.
- Two major component of I3D:
    - **Bottleneck Block**
    - **Inception Block**
- It considers 16/ 64-frames clip for spatio-temporal feature extraction.

I3D is a 3DCNN version of GoogleNet (InceptionV1)

4. Bottleneck Block ::

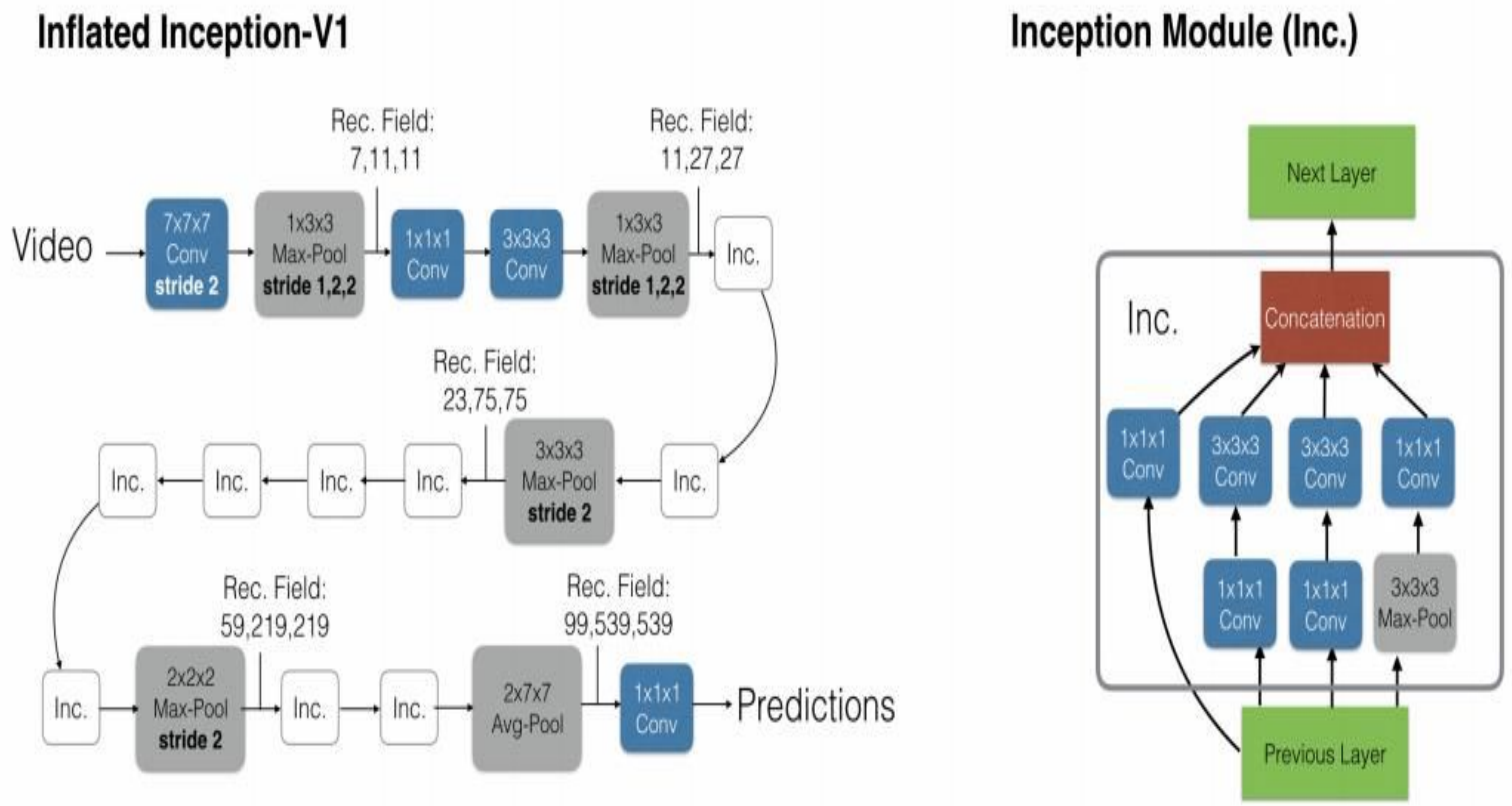


4. Inception Block ::

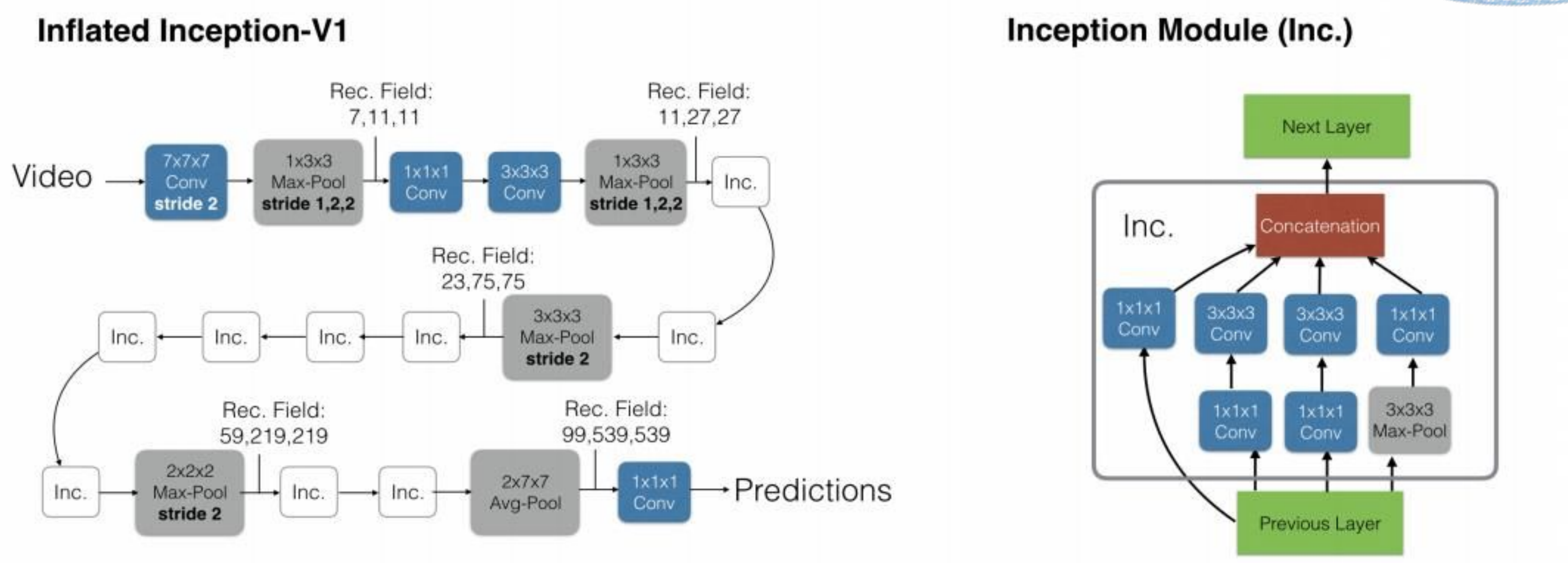


(a) Inception module, naïve version

4. I3D Network ::



4. I3D Network ::



Limitations of 3DCNN

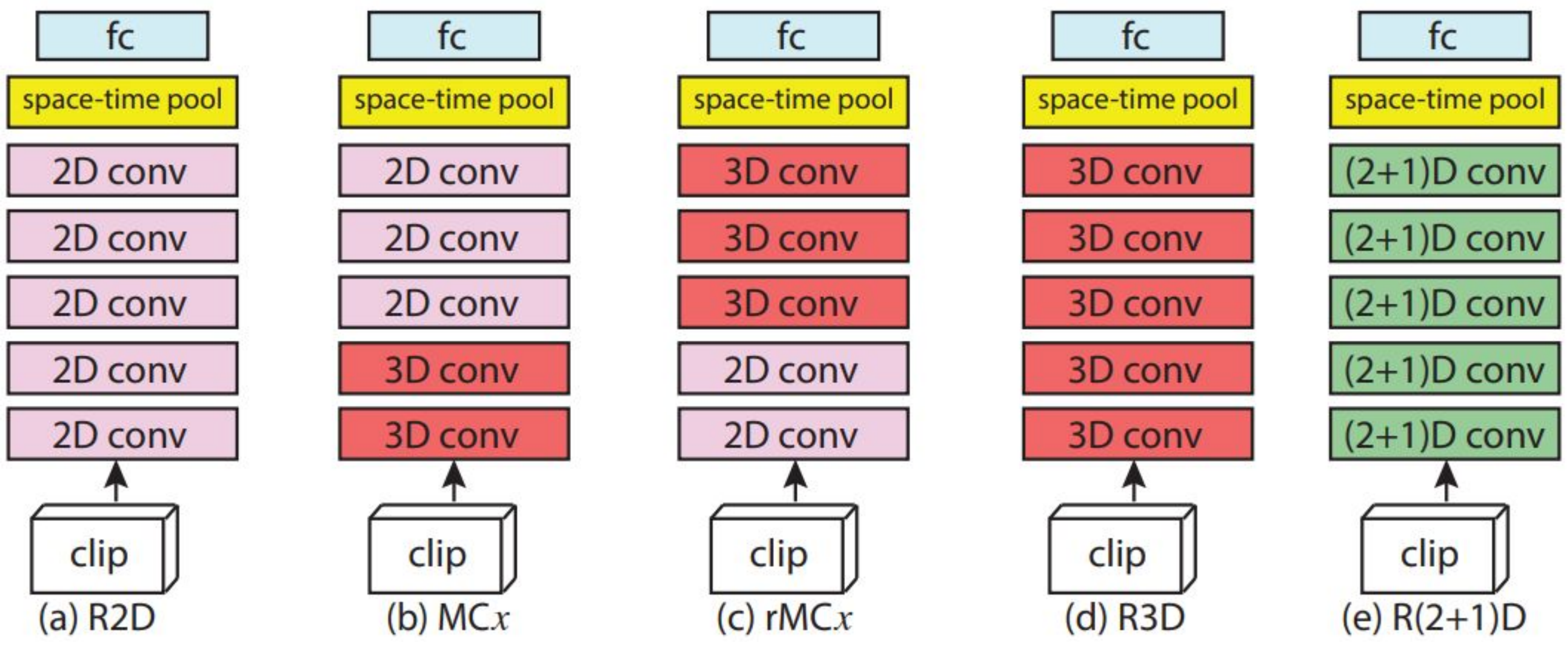
- Rigid spatio-temporal Kernels limiting them to capture subtle motion.
- No specific operation for discriminative feature representations.

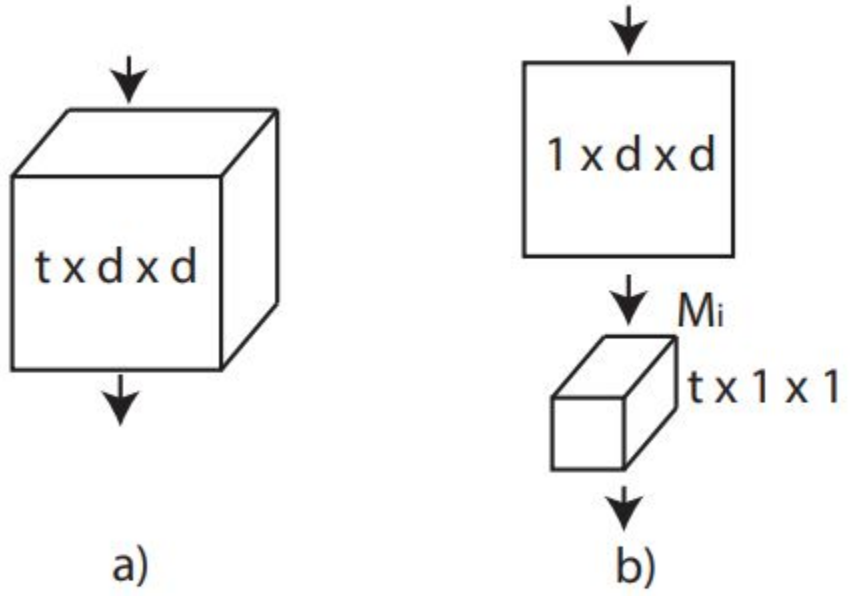
4. R(2+1)D Architecture::

- R(2+1)D factorizes the 3D convolutional filters into separate 2D spatial and 1D temporal convolution.

- It has almond double additional nonlinearity compared to standard 3D blocks with same parameters
  - Thus renders the model capable of representing more complex representation

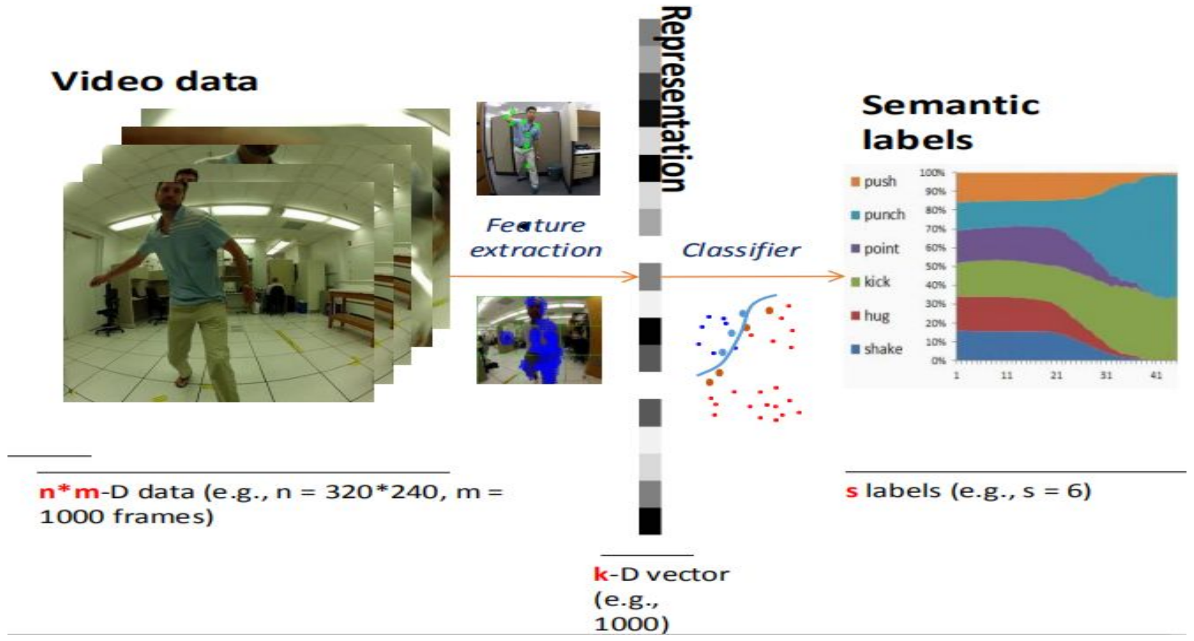
- Easier to optimize.

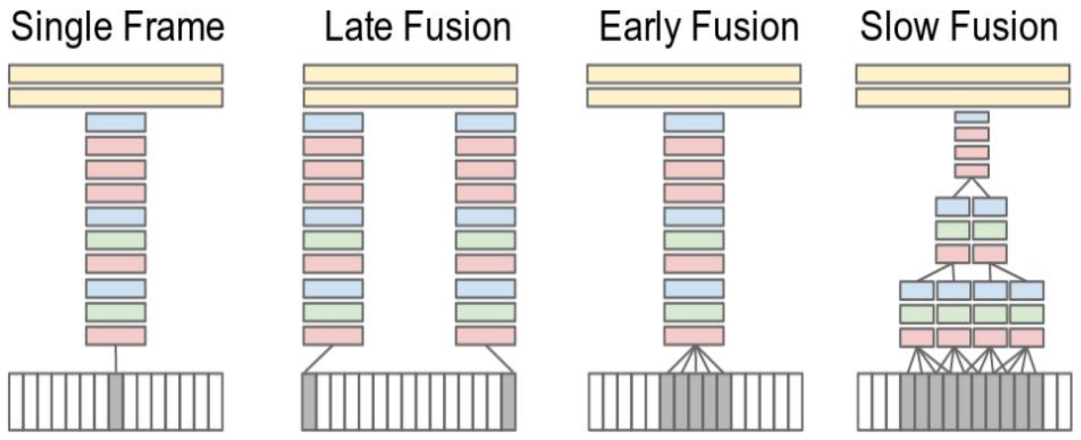




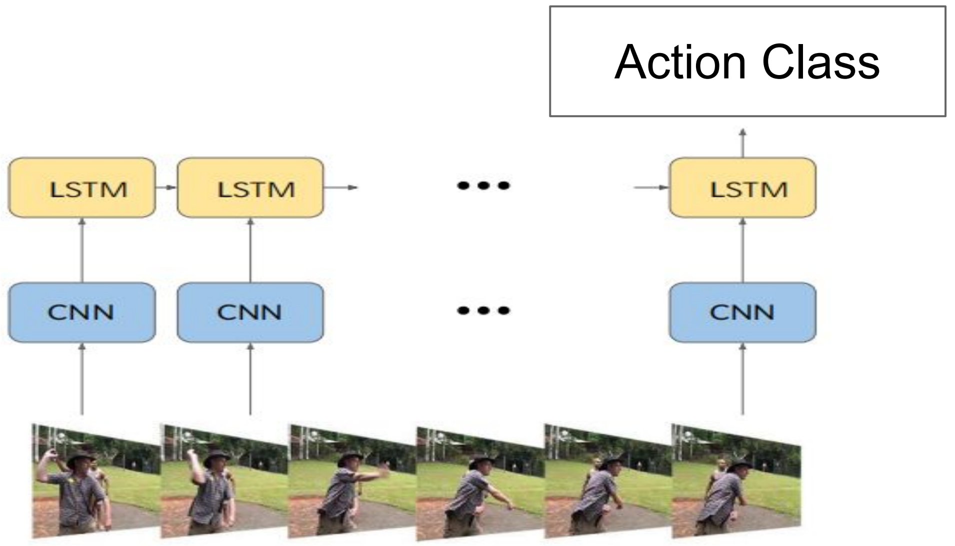
Summary ::

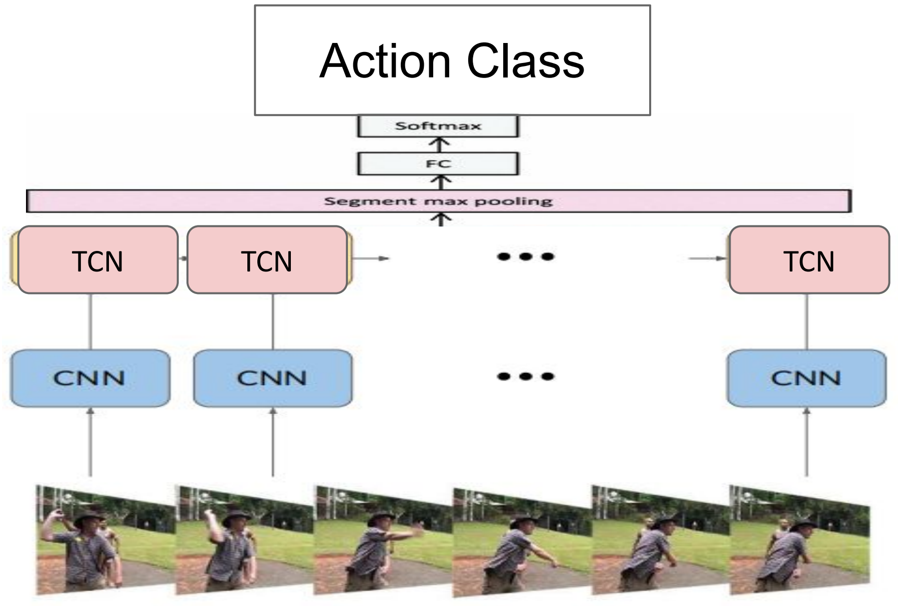
Classical Image Models



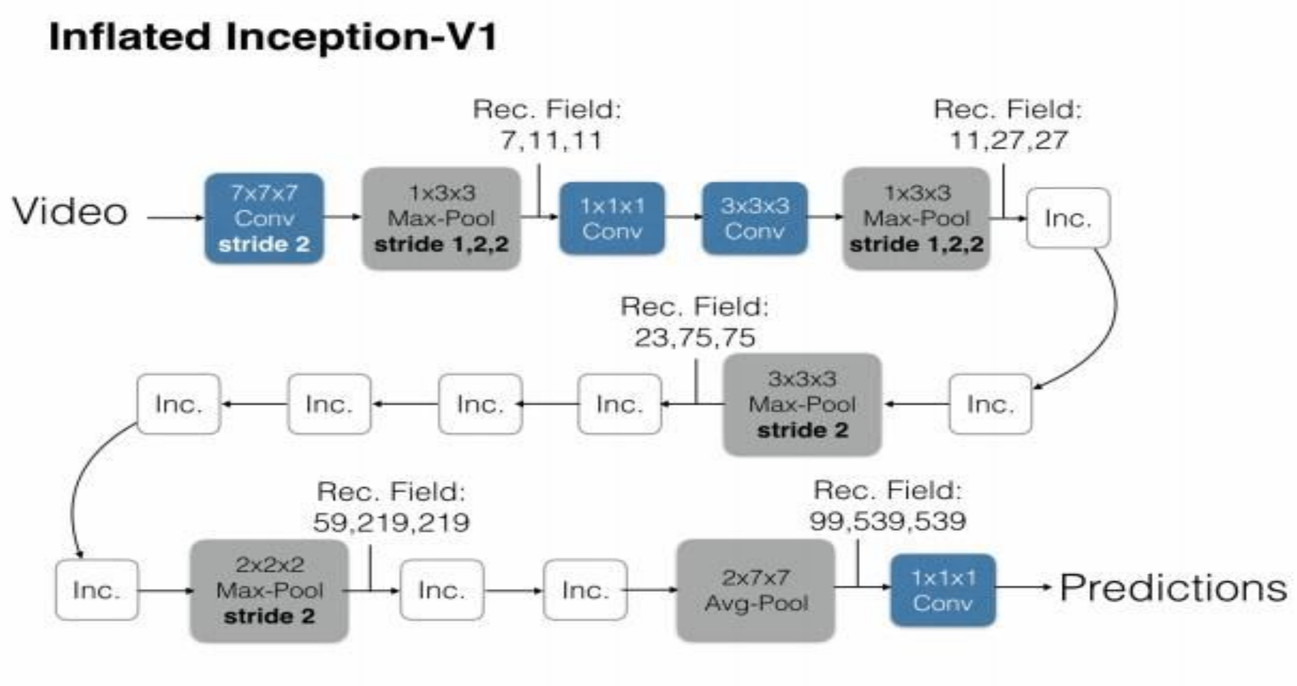


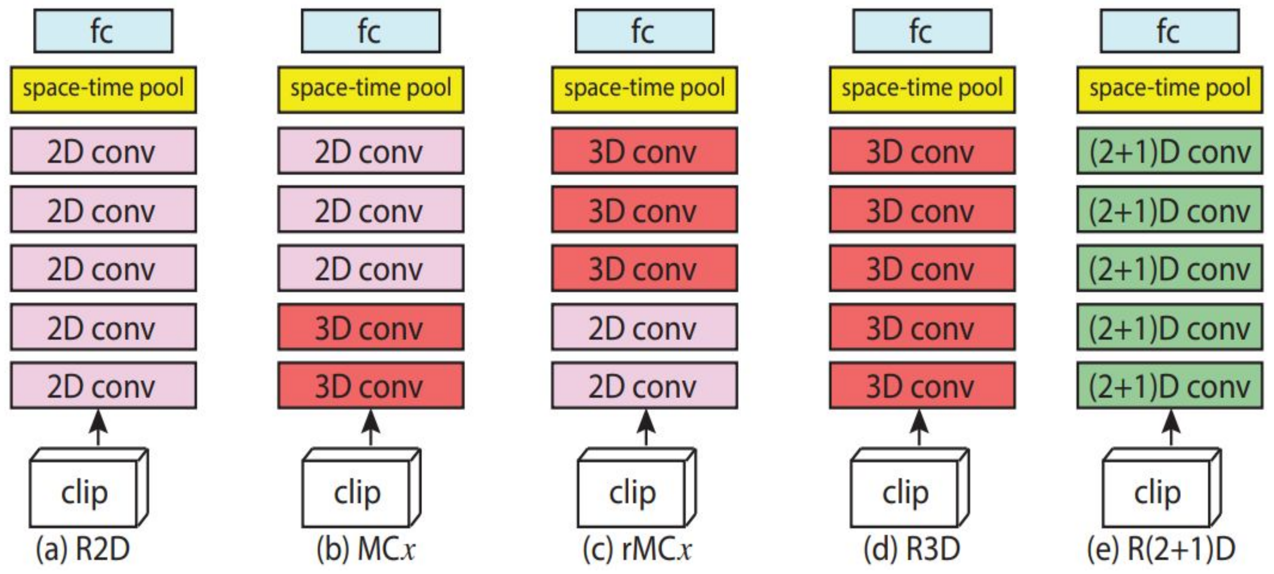
Classical Image Models with Temporal Models





Classical Video Models





Upcoming Agenda

- Introduction to HAR: Human Action Recognition
- Multiple Modalities in HAR
- Attentions in HAR (Spatial, Temporal, Self Attention)
- Recent Popular Techniques
  - Transformer Models (ViT, ViviT, Swin, VideoSwin)
  - Self-supervised Models (MAE, VideoMAE, DiT)
  - Vision and Language Models (CLIP)

Thank you for your attention!