VIDEO & ACTION Classification

Snehashis MAJHI

Email: snehashis.majhi@inria.fr

Ph.D. Candidate @STARS Team INRIA

Collaboration with TOYOTA Motor Europe

Ezem EKMEKCI

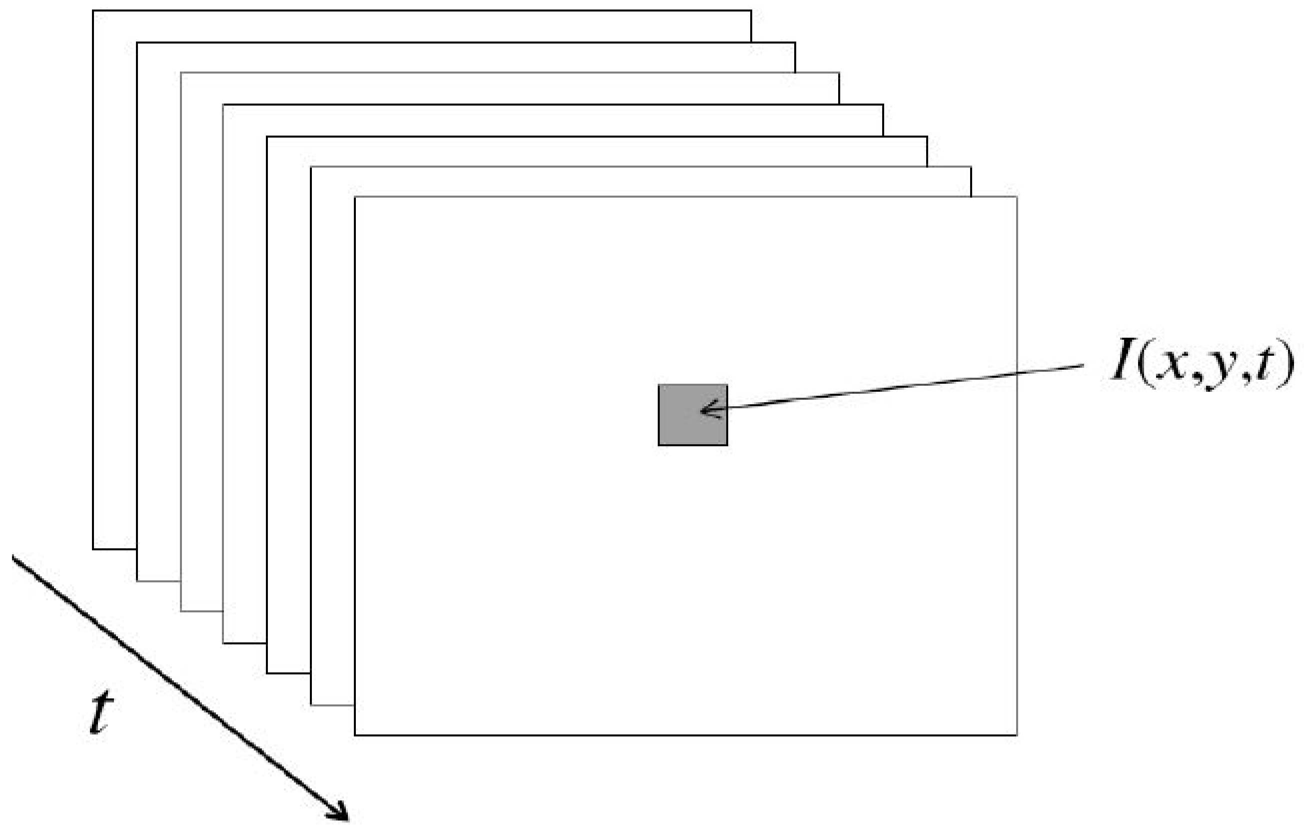
Email: ezem-sura.ekmekci@inria.fr

Joint Ph.D. Student@STARS and EPIONE Team INRIA

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:

BBC Motion Gallery

~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

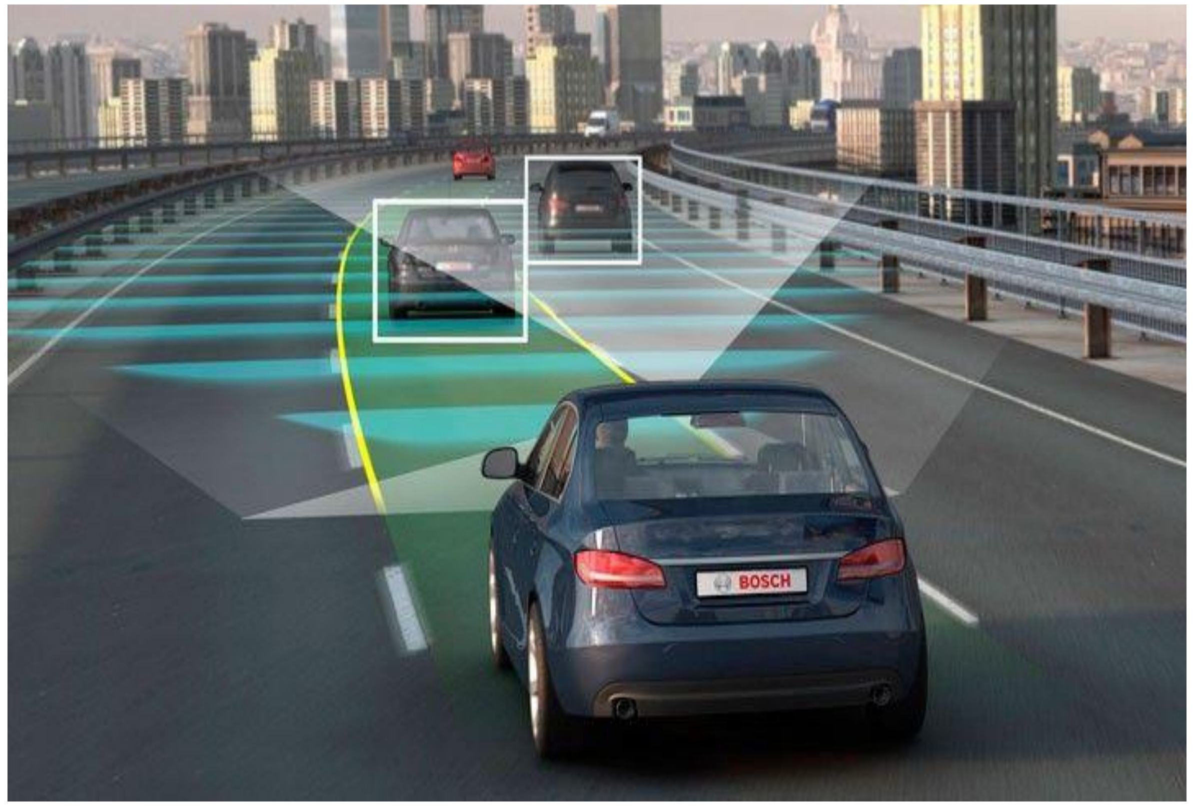
And even more with future wearable devices

Real-world Applications ::

3 . Collective Activity Understanding

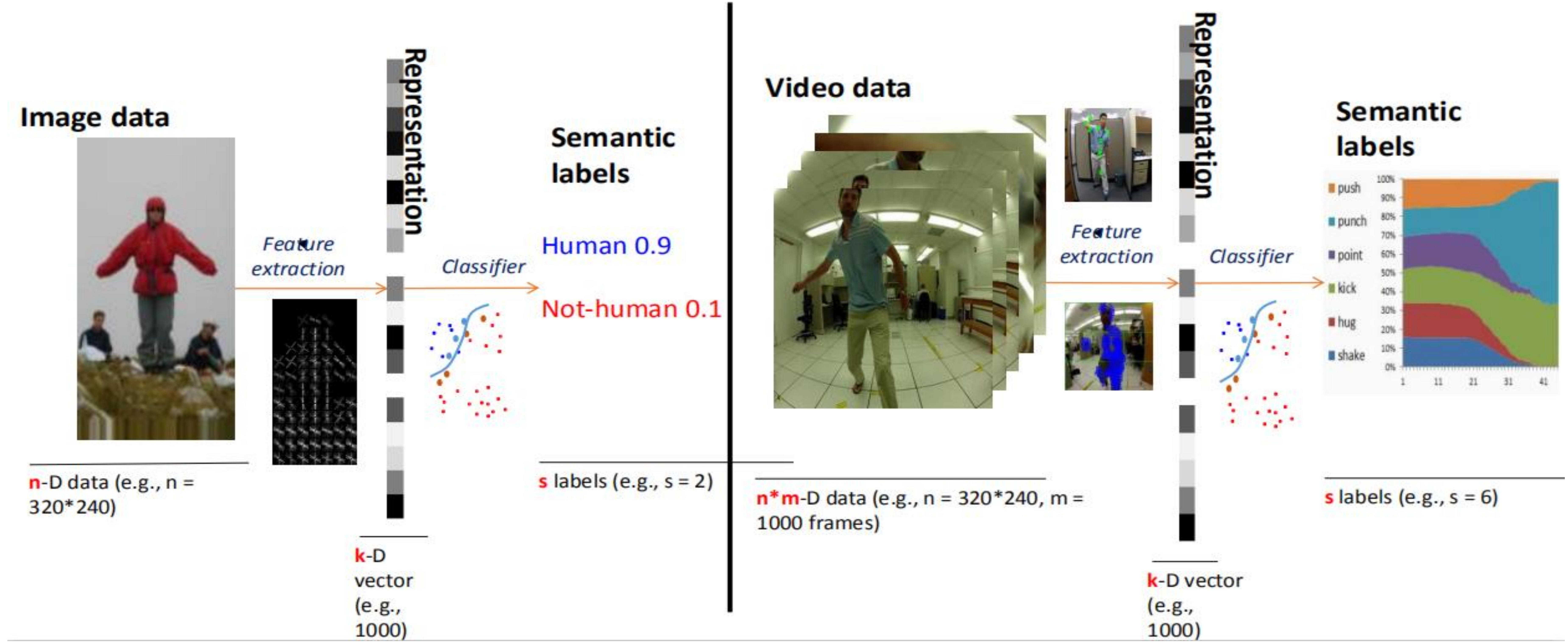
1 . Robotics and Manipulation

2 . Self Driving Cars



4 . Event Classification

Image Vs. Video Classification ::

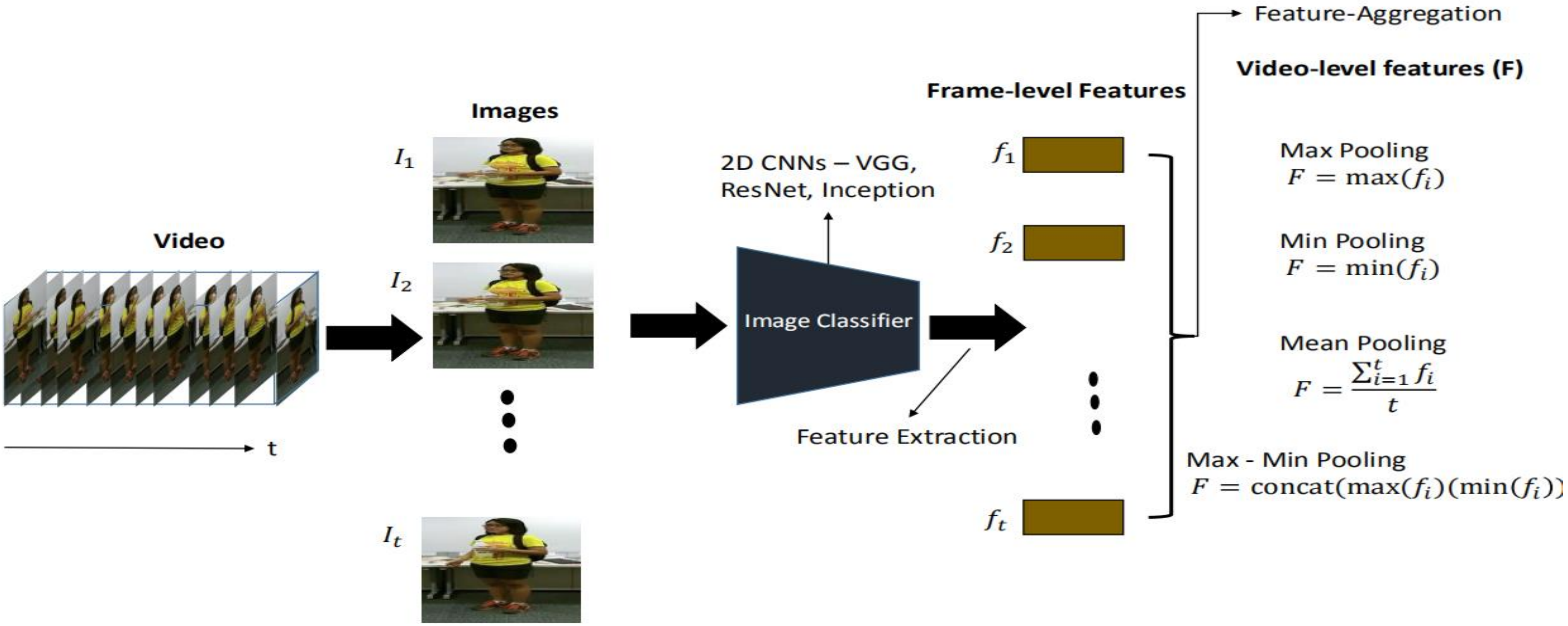


Video Classification Techniques ::

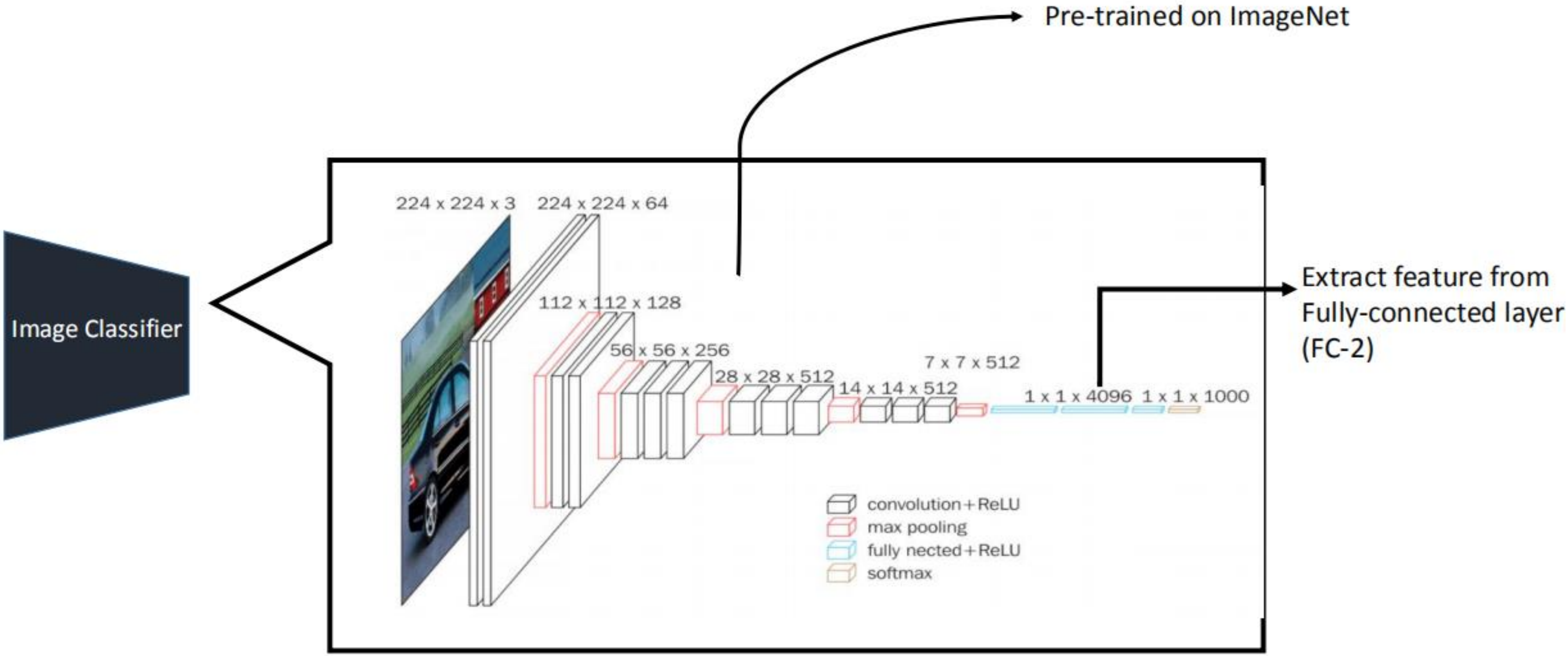
Evolution of Research

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

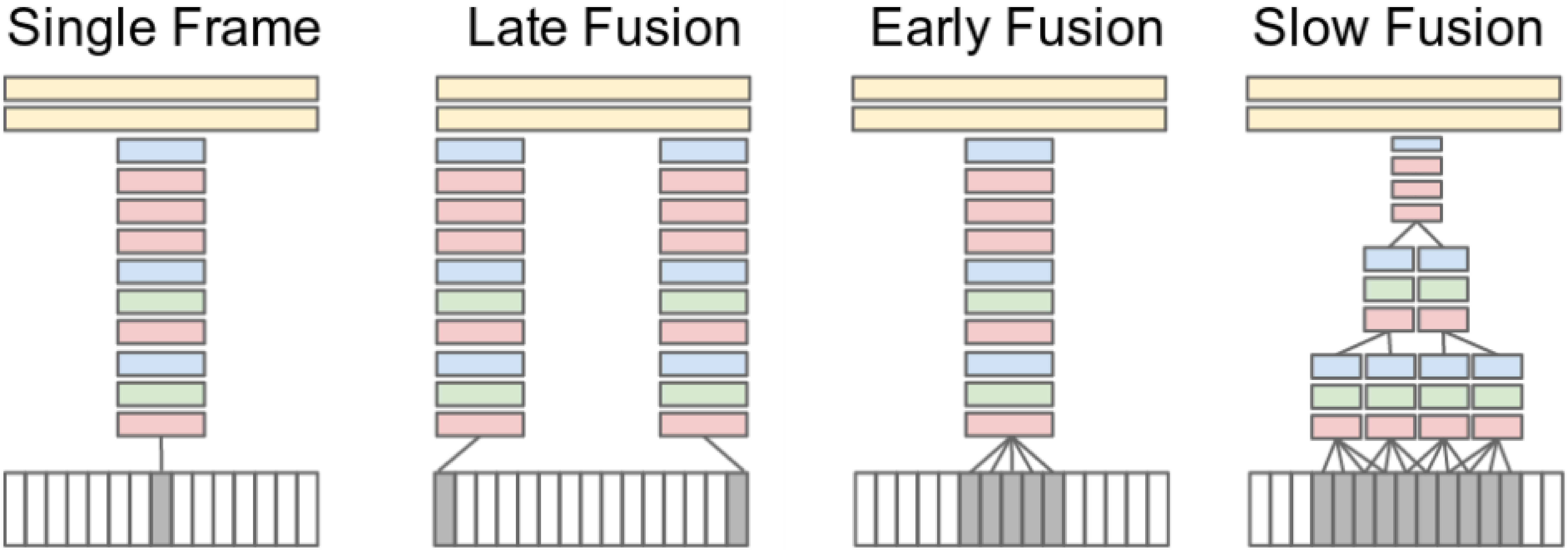
1.Frame-Level Aggregation of 2D CNN ::



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**.

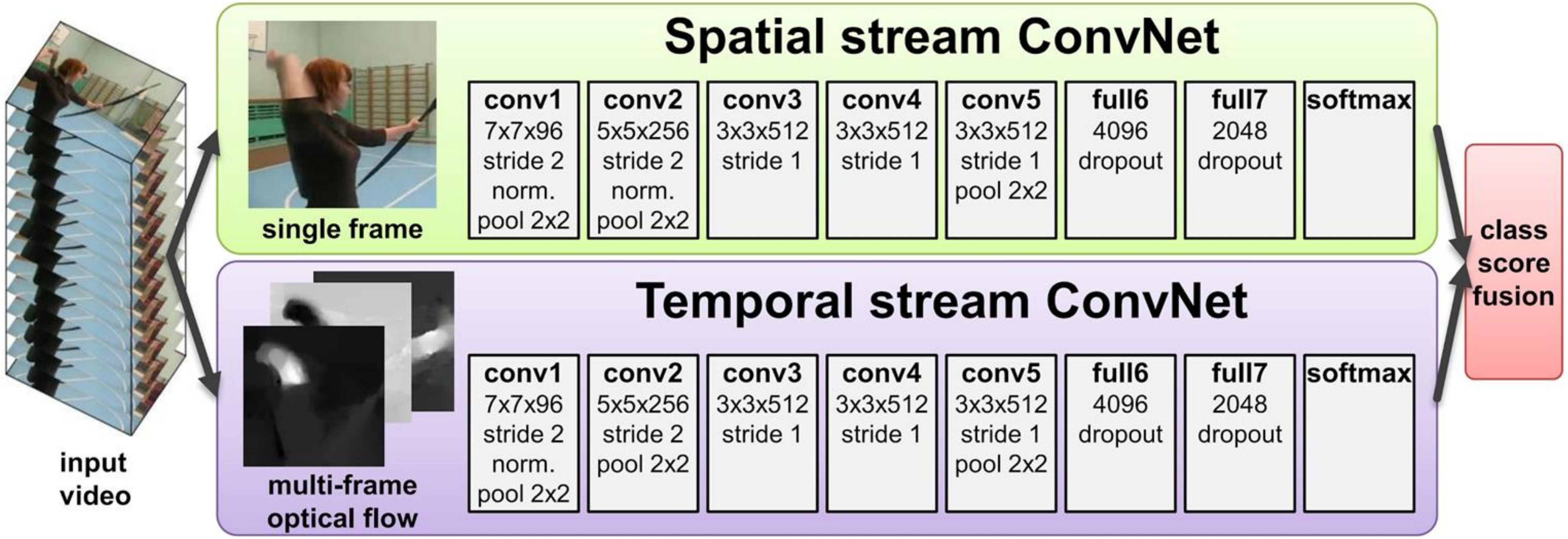
- The video descriptors for each videos are treated as data samples for a classifier (like SVM) for classifying the videos.

- 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 ::

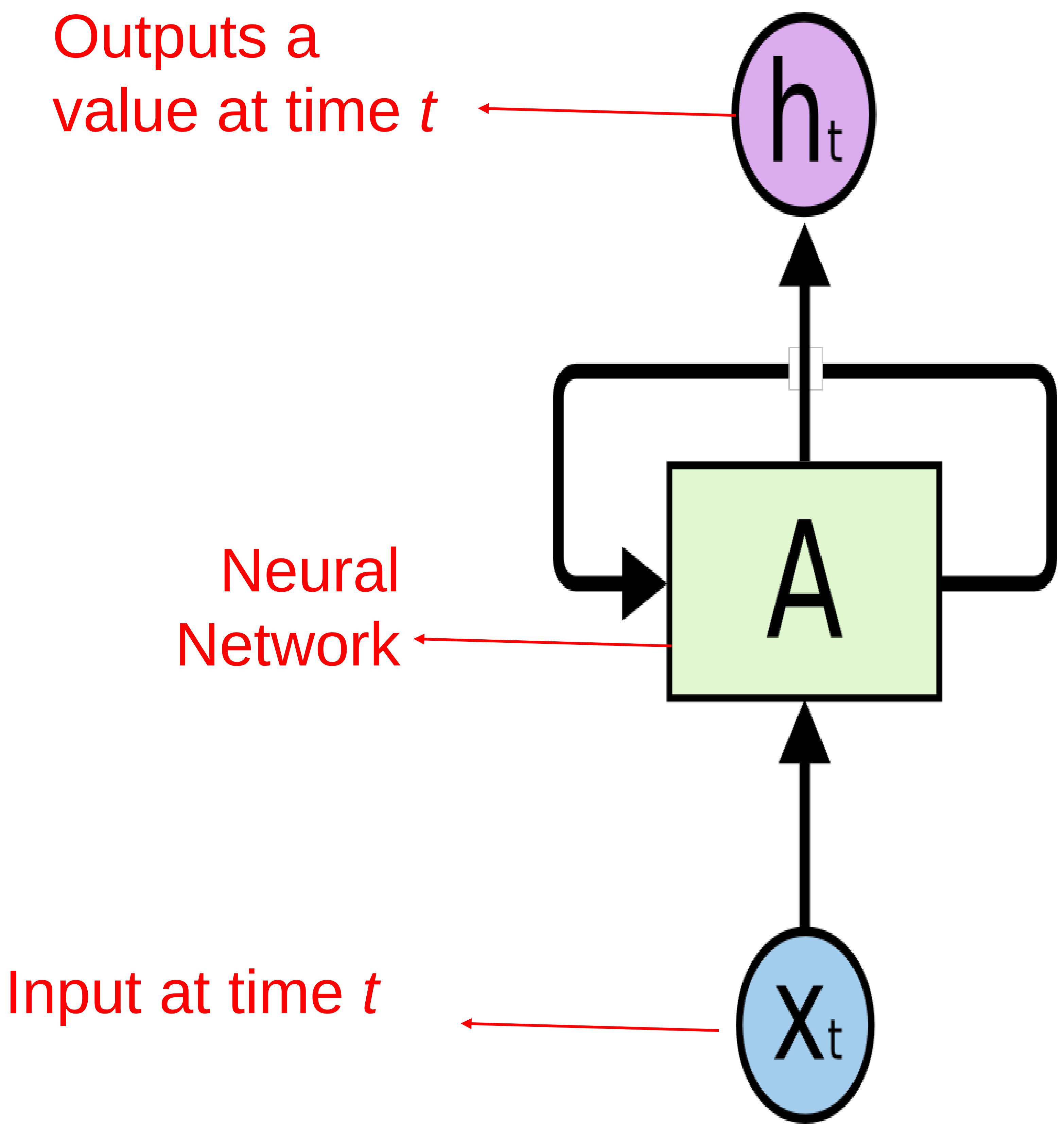
- **Idea: To combine both Appearance and motion representations.**
- **Previous work: Failed because of the difficulty in learning implicite motion.**



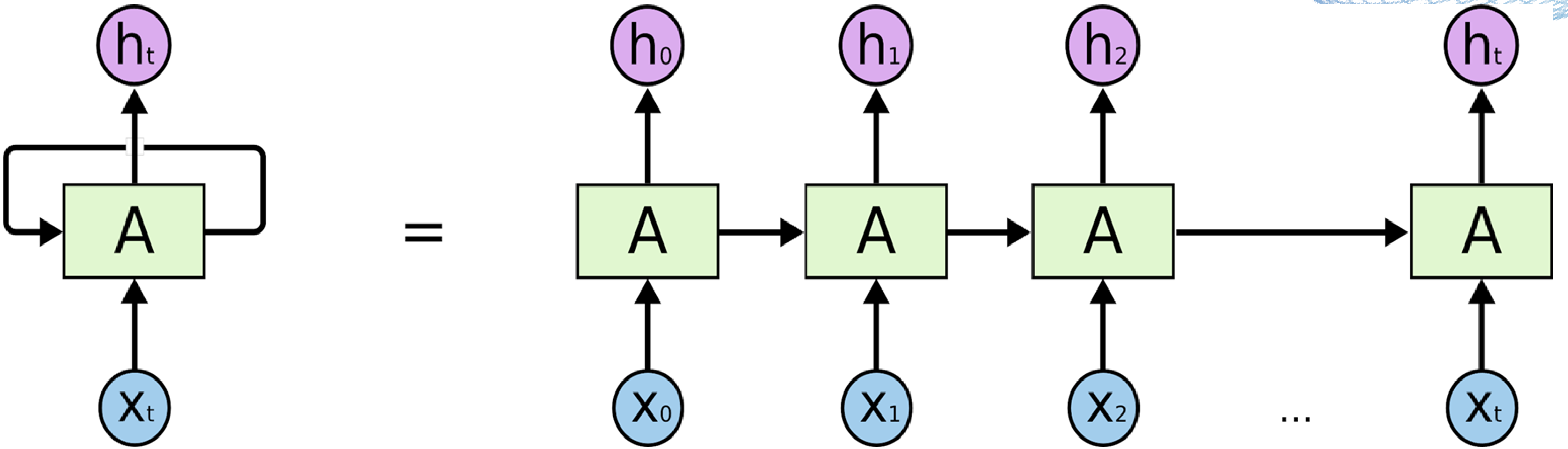
- **Separate the *Motion (multi-frame)* from *static appearance* *(single frame)*.**
- **The appearance and motion stream are not aligned.**
- **Optical flow can only capture short term temporal dynamics**

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.



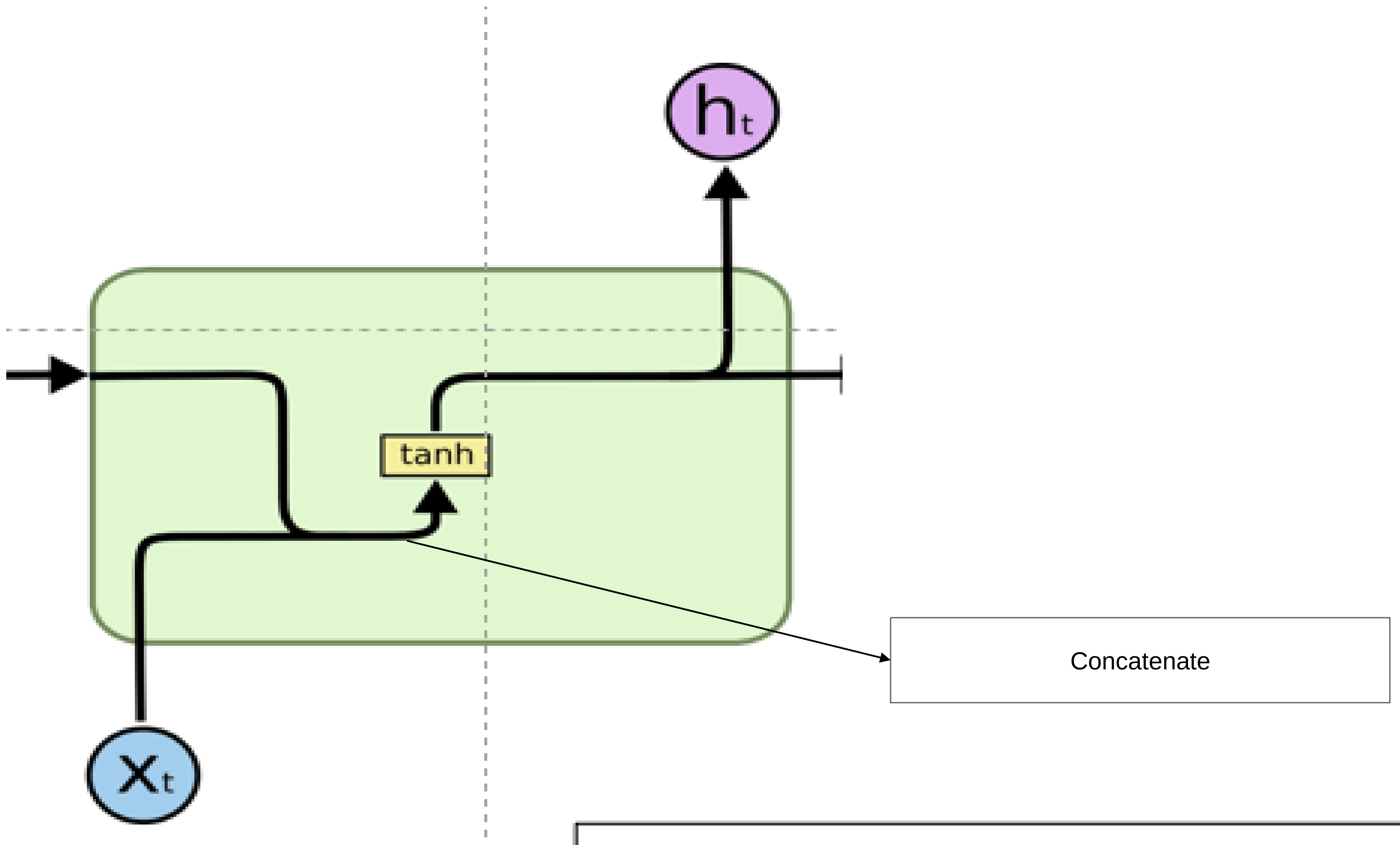
3. Recurrent Neural Network::



$$\boxed{h_t = f_W(h_{t-1}, x_t)} \xrightarrow{\text{A typical example}} \boxed{h_t = tanh(W_{hh}h_{t-1}, W_{hx}x_t)}$$

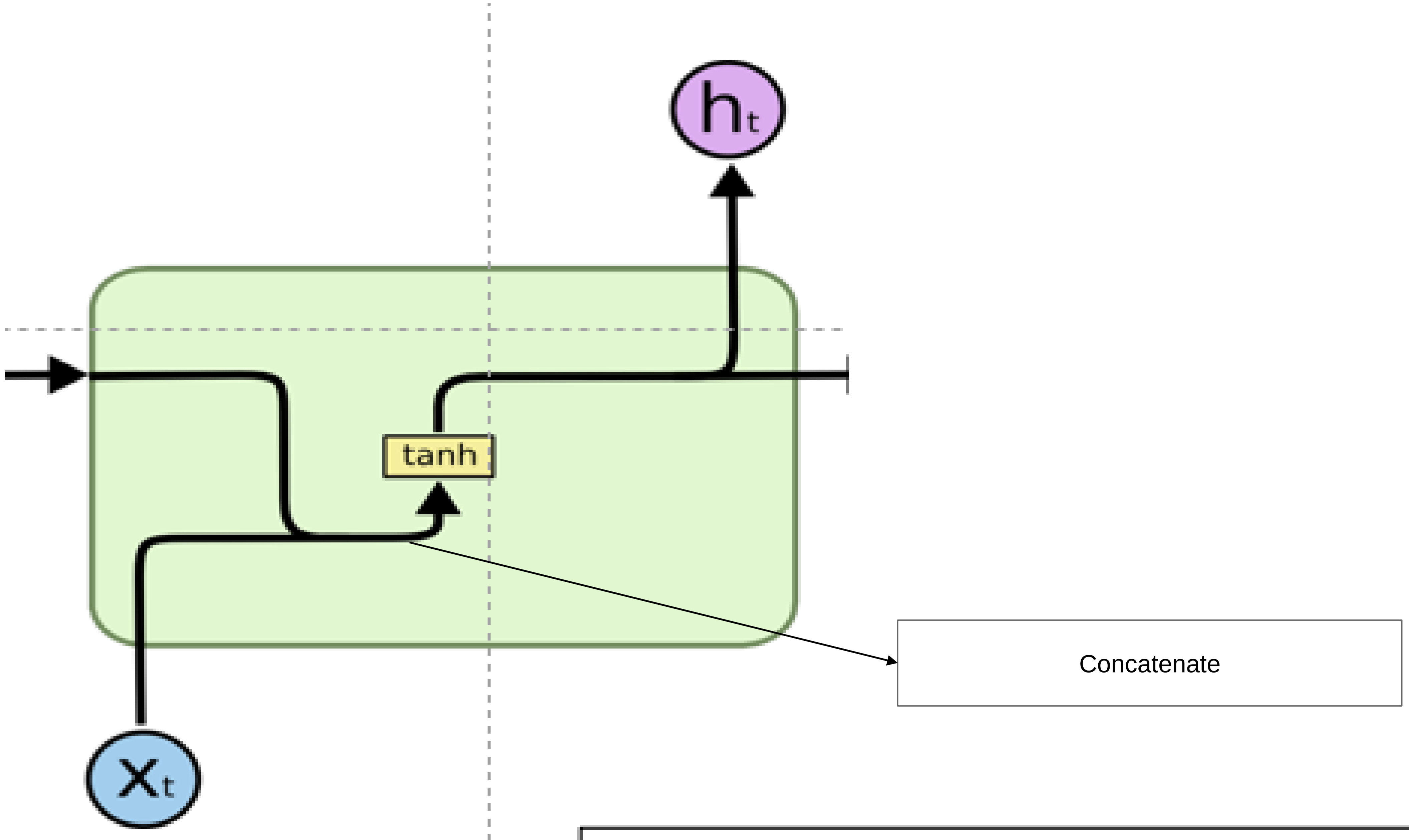
Some function with parameter W

3. Single RNN Unit::



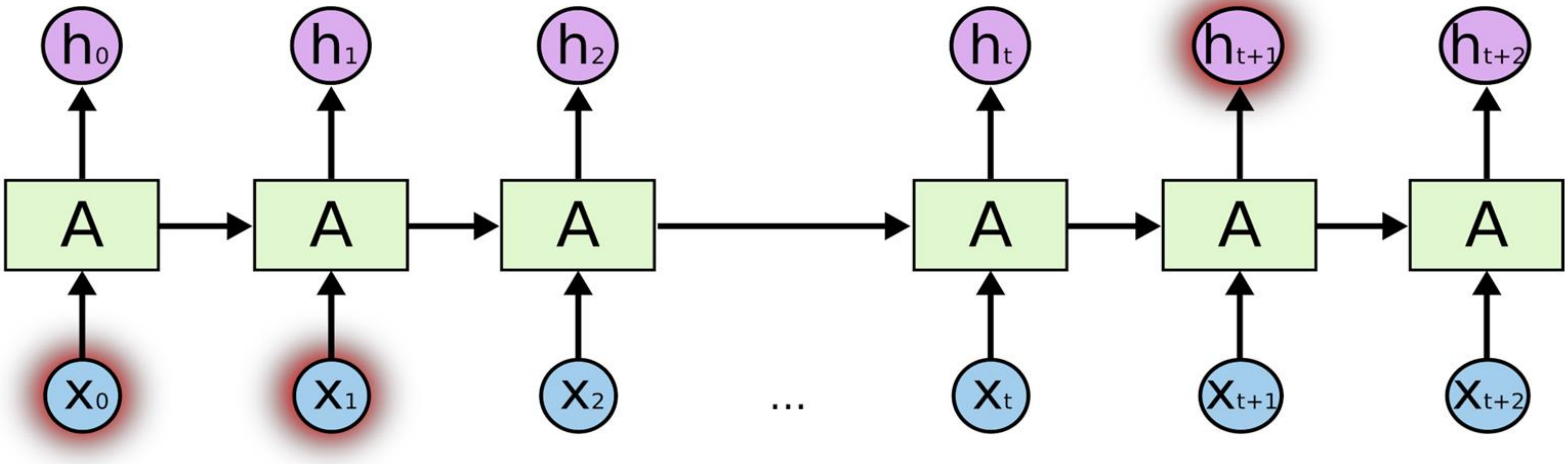
$$h_t = tanh(W_{hh}h_{t-1}, W_{hx}x_t)$$

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 vanishing gradient factor.

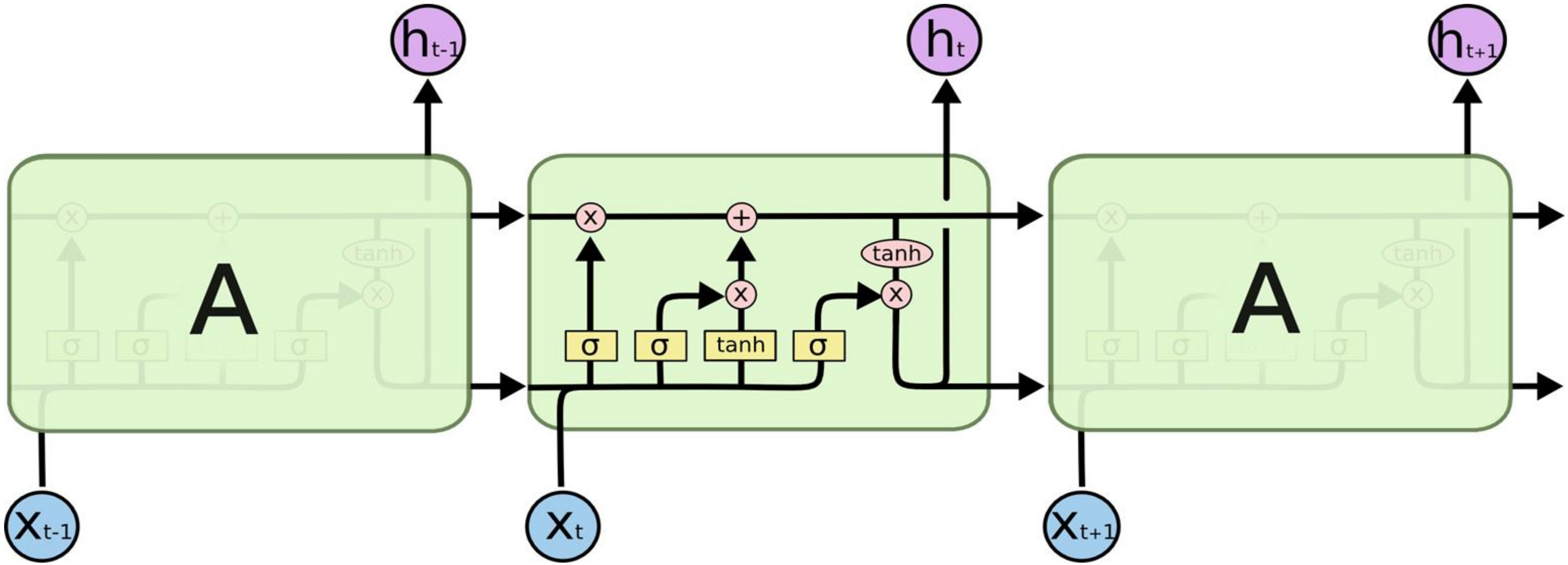
3. Long-short Term Memory (LSTM)::

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.

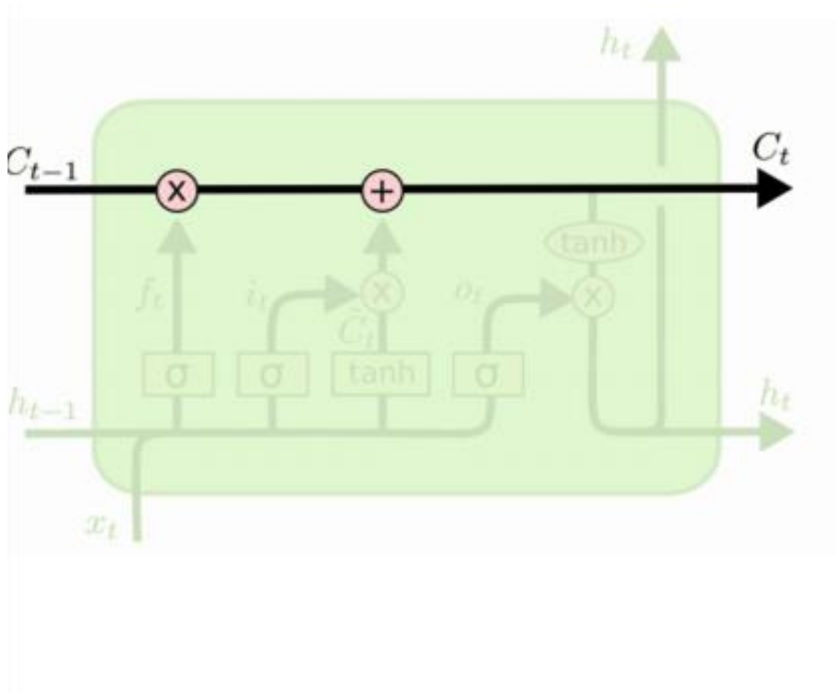
3. Long-short Term Memory (LSTM)::

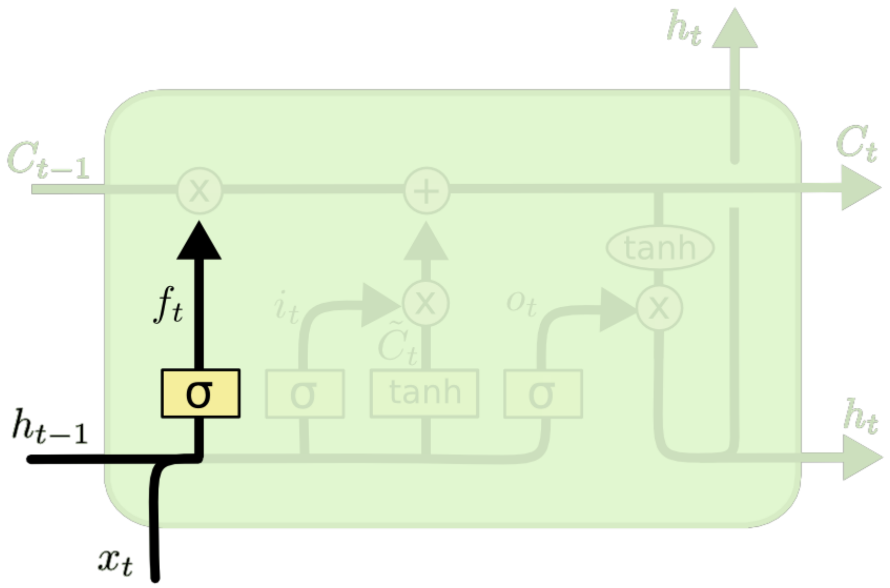


3. Different Modules of LSTM::

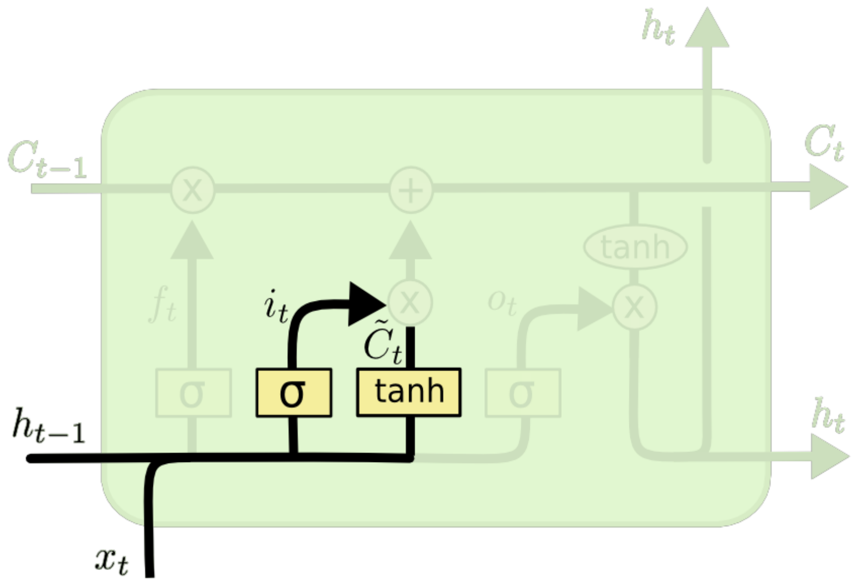
Four Major modules

1.Cell State

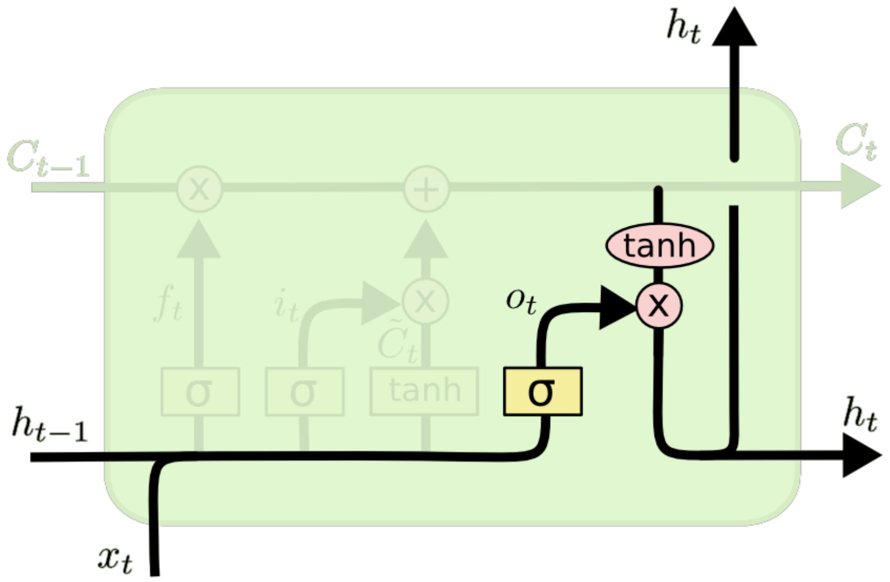




2. Forget Gate

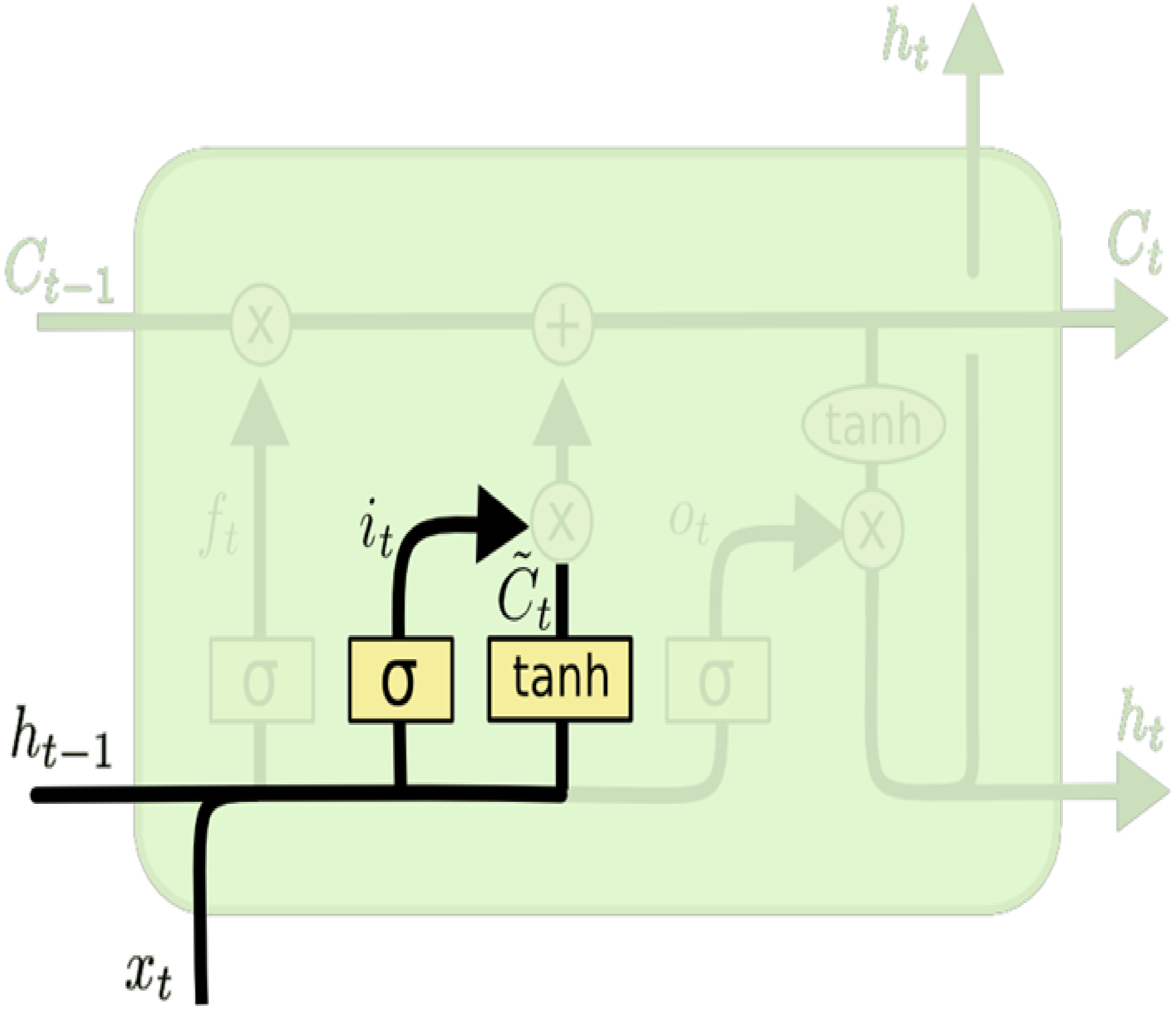


4.O



3. Working of LSTM::

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



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

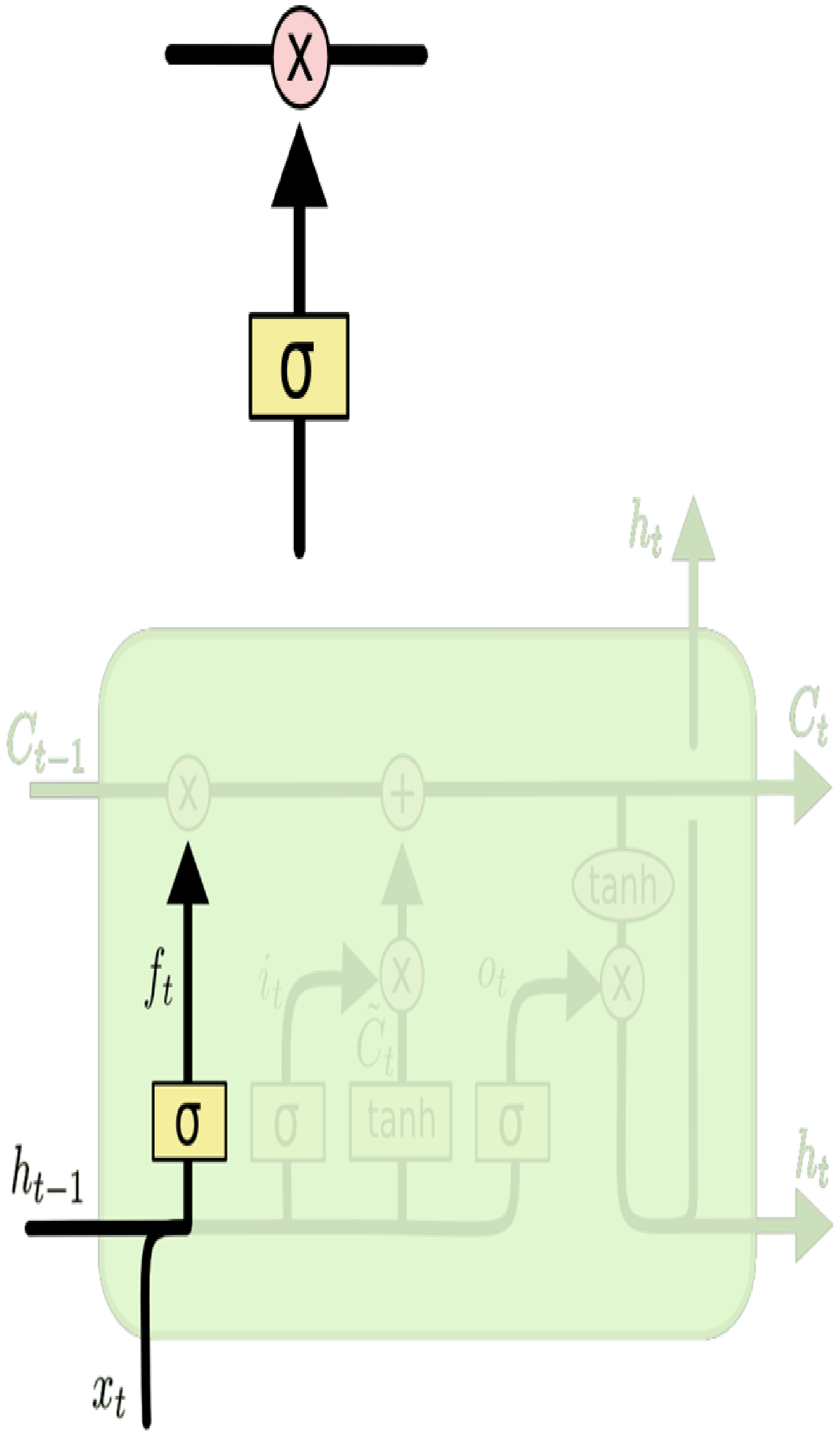
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

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.



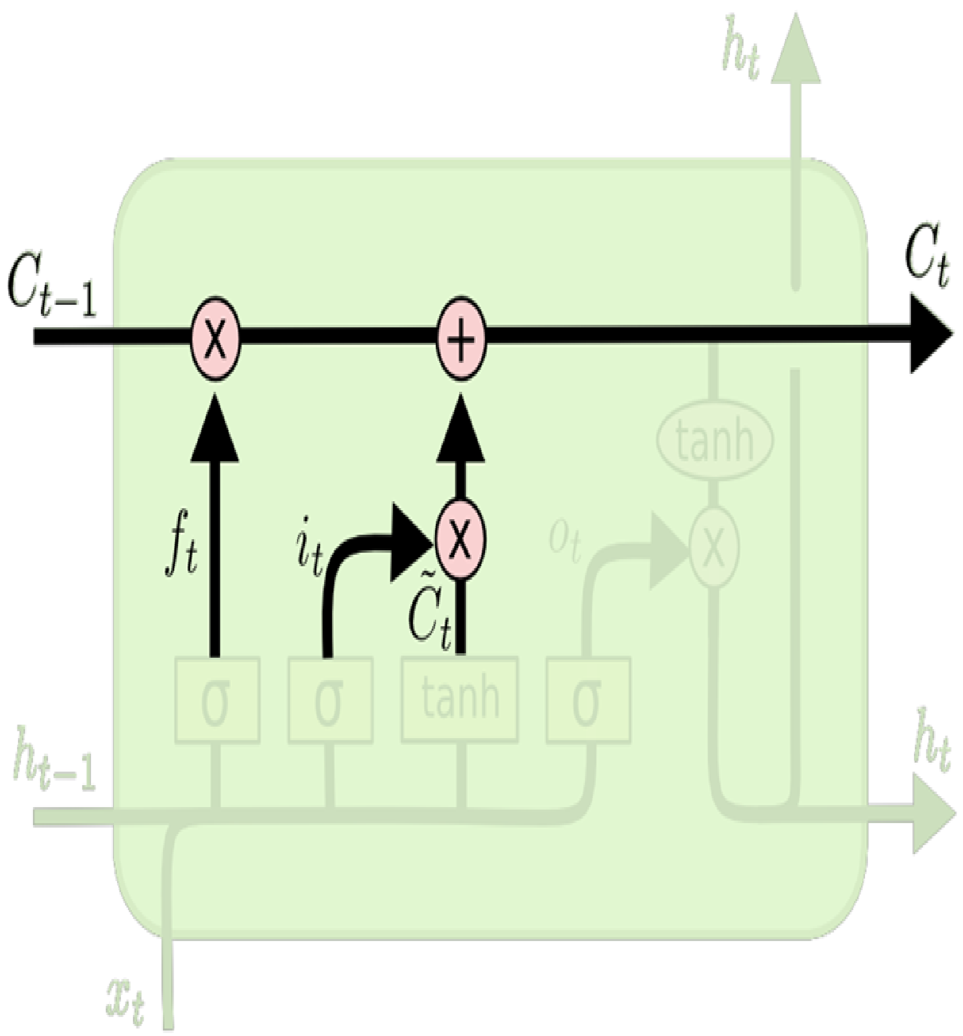
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

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.

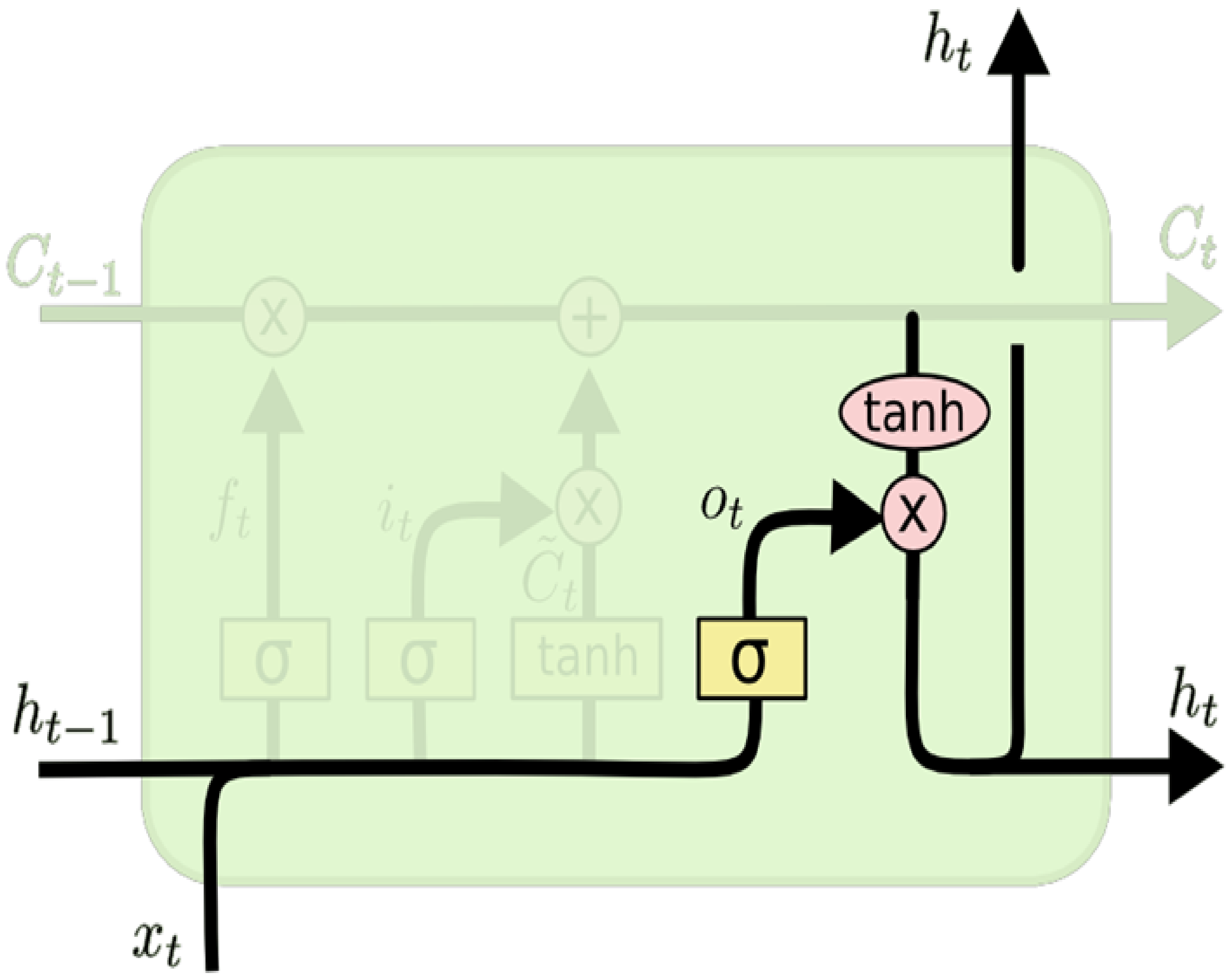


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

3. Working of LSTM::

Output Gate :

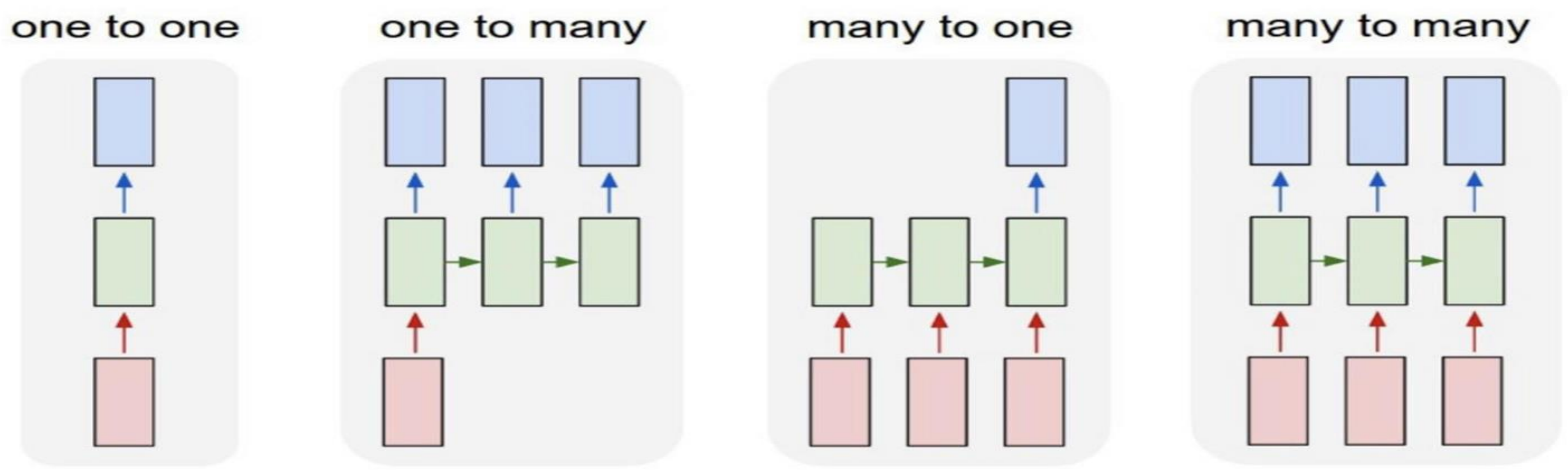
- Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version.



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

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

3. Types of LSTM::



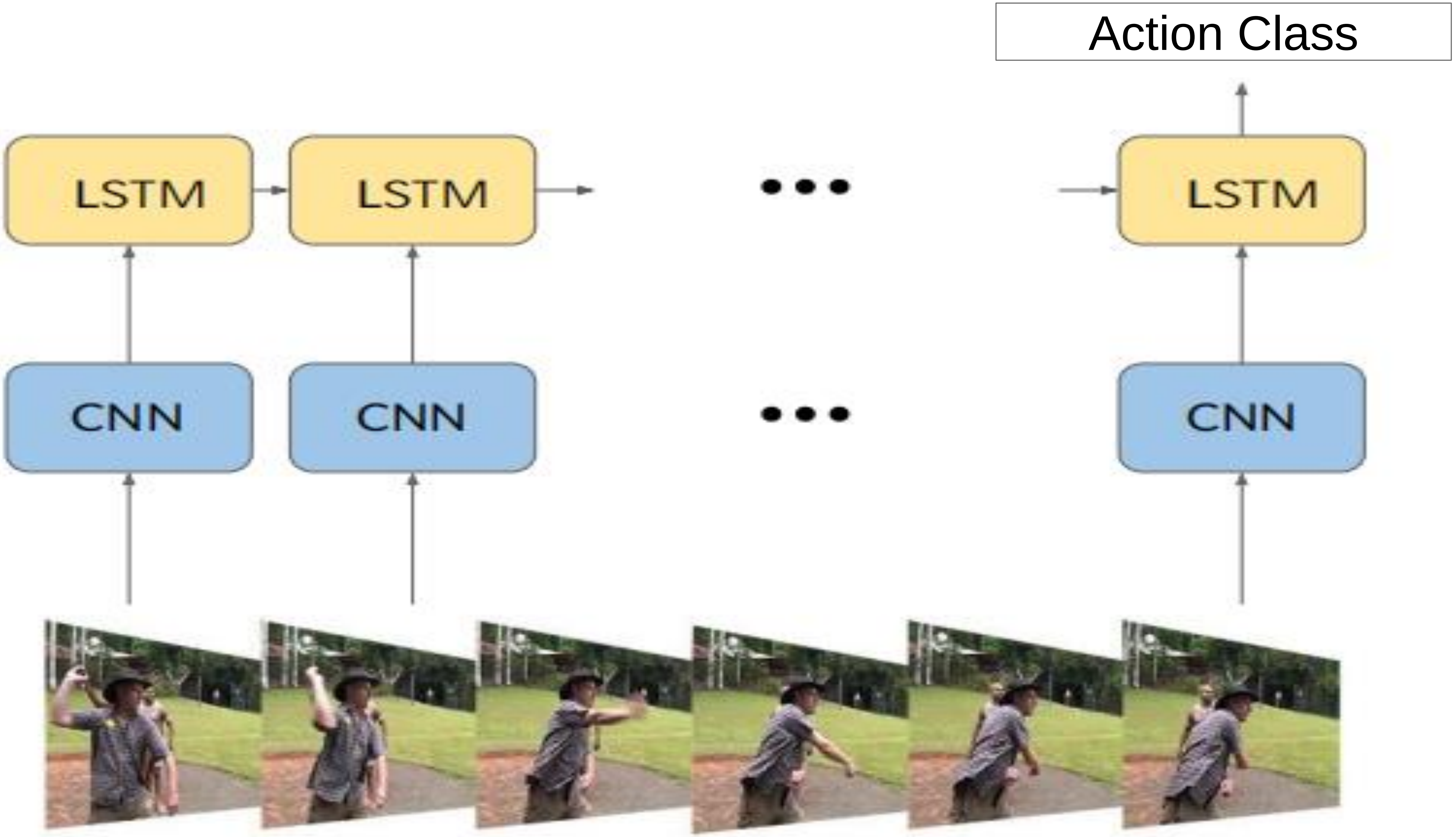
Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification)

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::

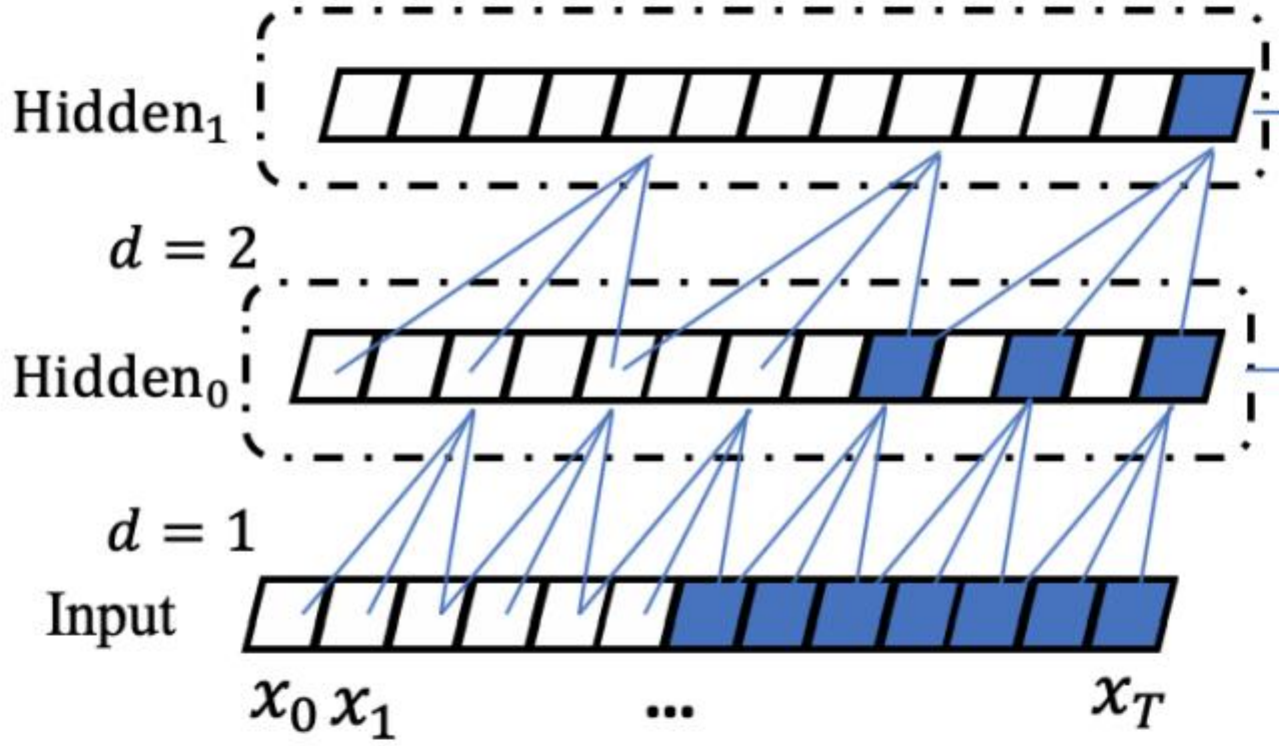


3. Drawback of RNN/LSTM::

- 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).

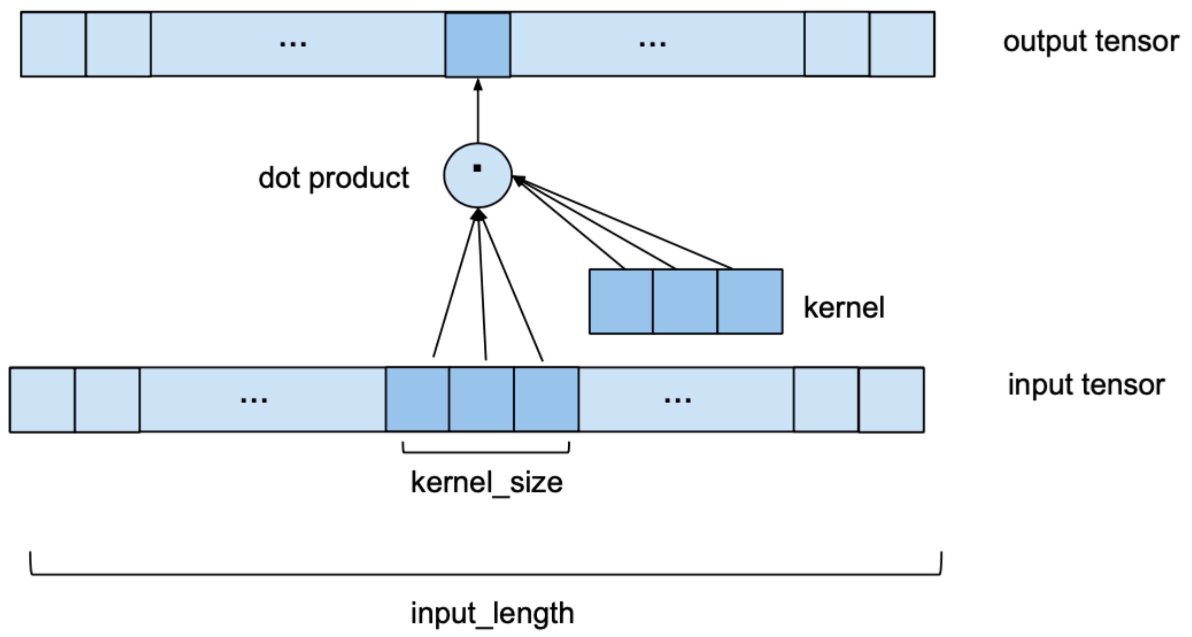
3. Temporal Convolution Network (TCN)::

- 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**).

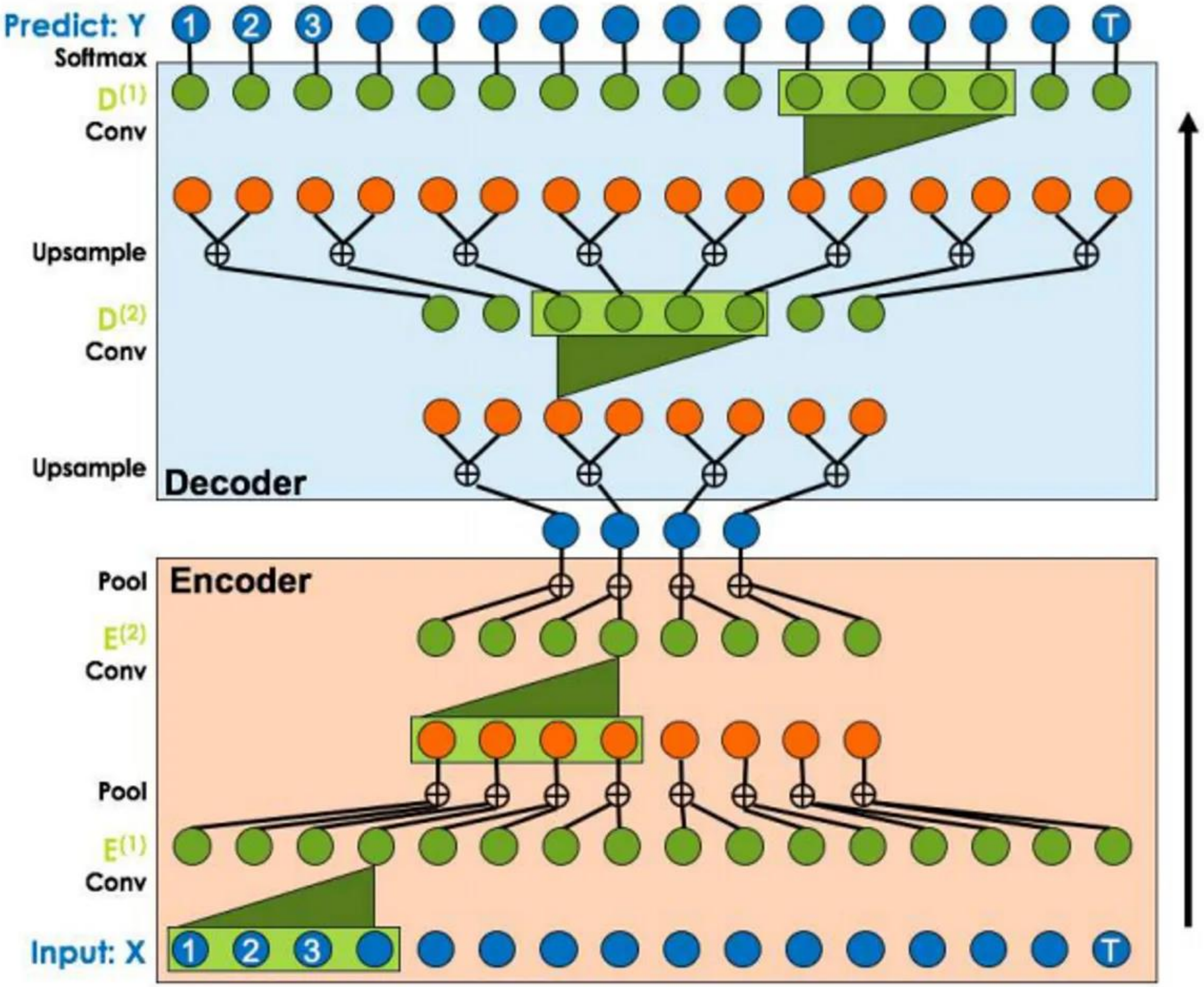
- **TCN can be causal (no information leakage from the future to the past)**



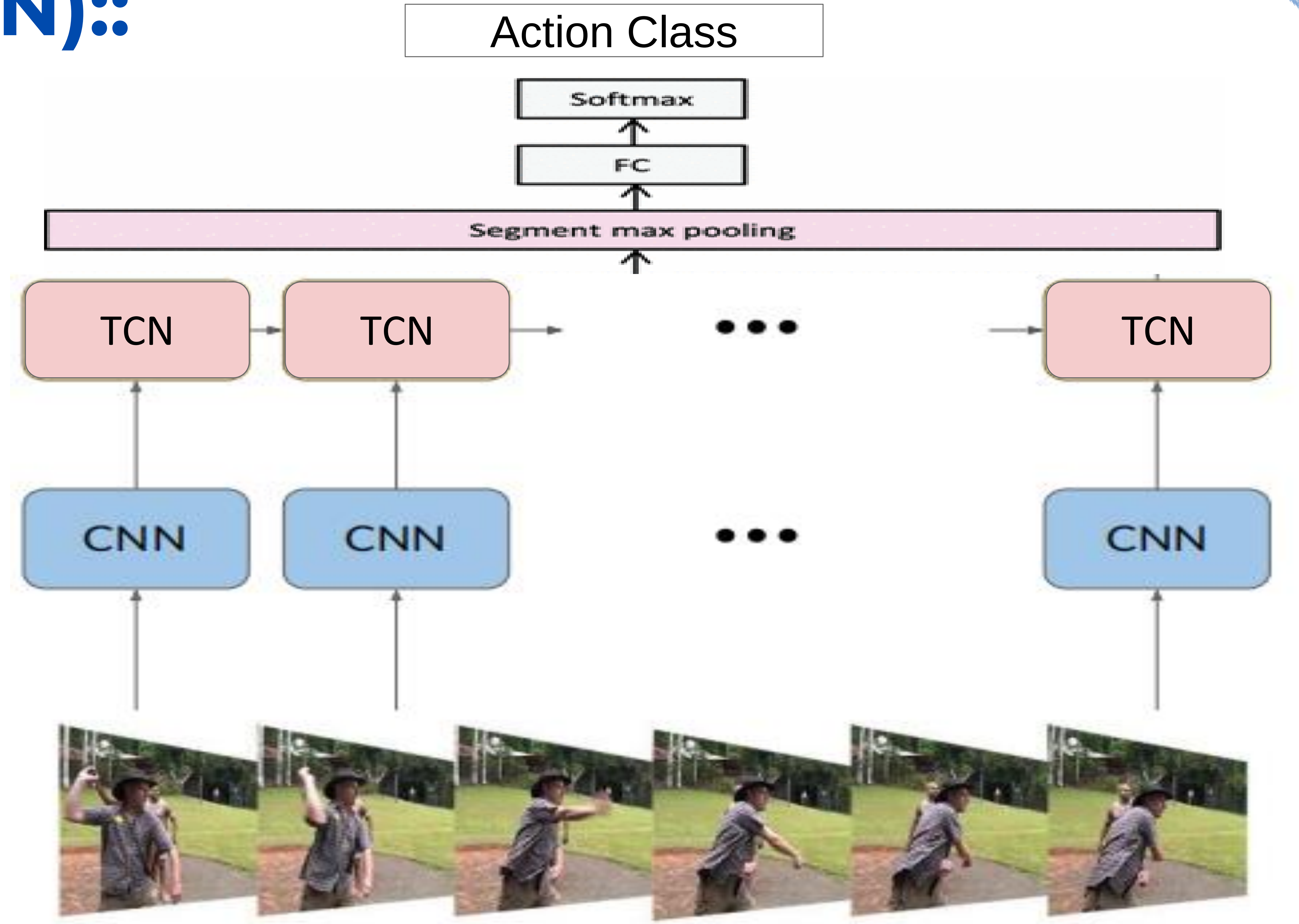
- **TCN can use a very-deep network with the help of residual connections, and it can look very far into the past to predict with the help of dilated convolutions**

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

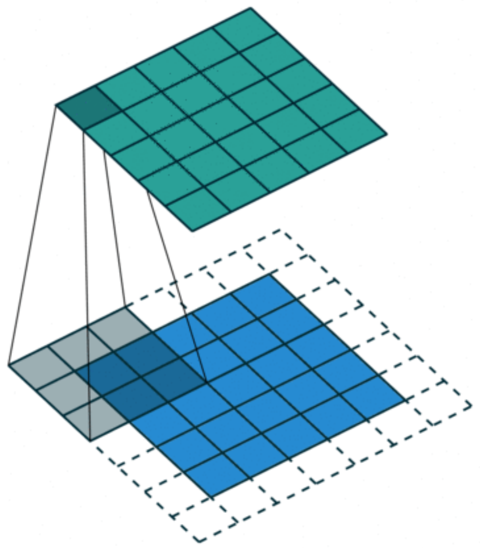
- NO Parallelism
- Fixed Receptive Field Size
- Vanishing Gradient Problem
- High Memory Requirement as it maintain Hidden State
- Not Possible to Knowledge Transfer between Domain(Pre-training LSTM is not a good Idea)

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

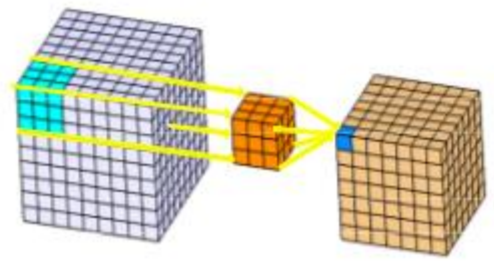


$$H_{out} = \frac{H_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$

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

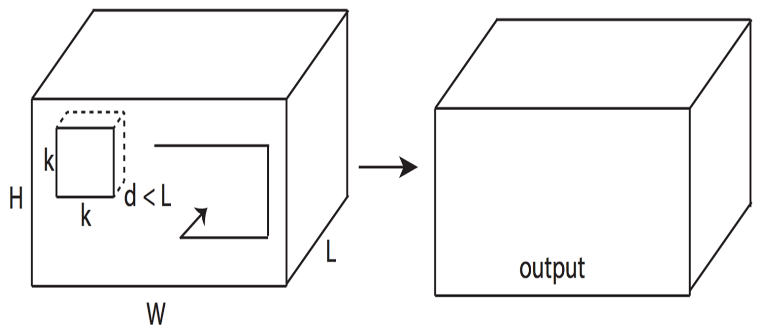
3D Convolution (XYT)

Input: [,,,]
Output: [, , , *#Kernel*]

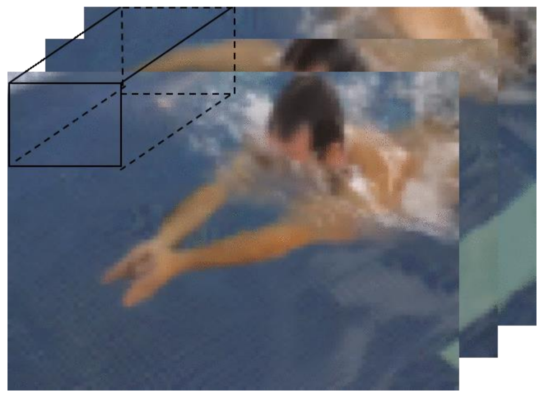


$$H_{out} = \frac{H_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$

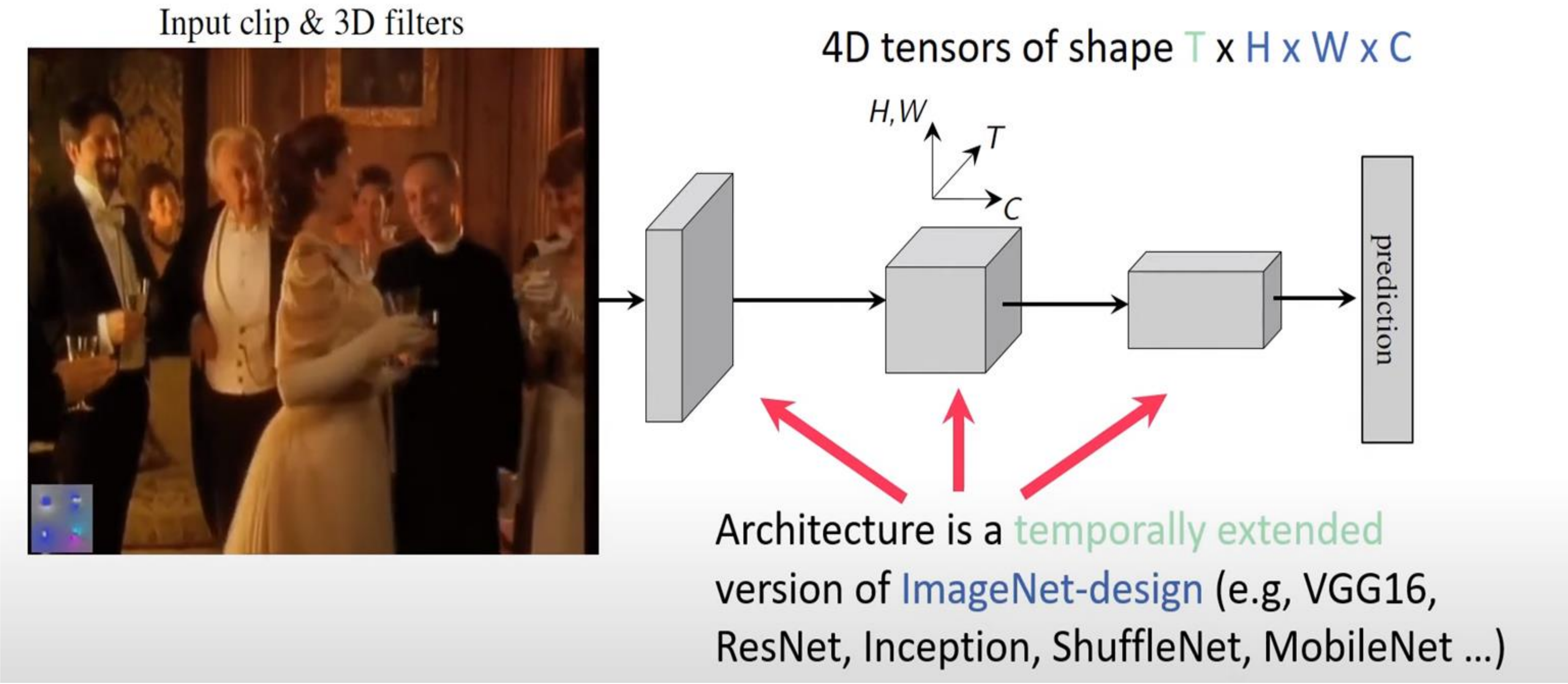
$$W_{out} = \frac{W_{in} + 2 \times padding - dilation \times (kernel\_size - 1) - 1}{stride} + 1$$



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

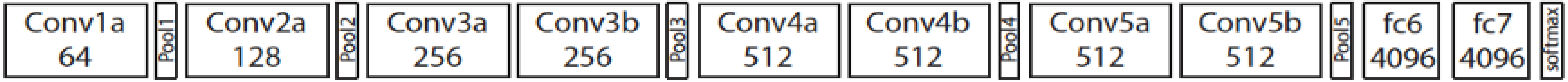


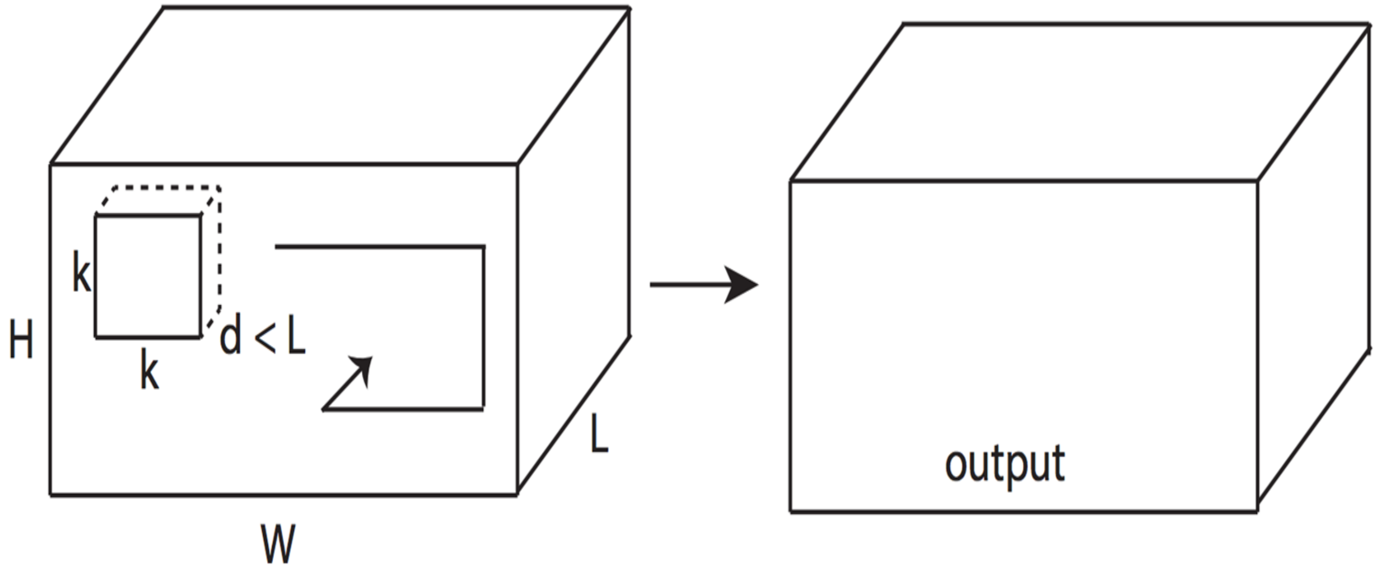
4. 3D Convolutional Neural Networks::

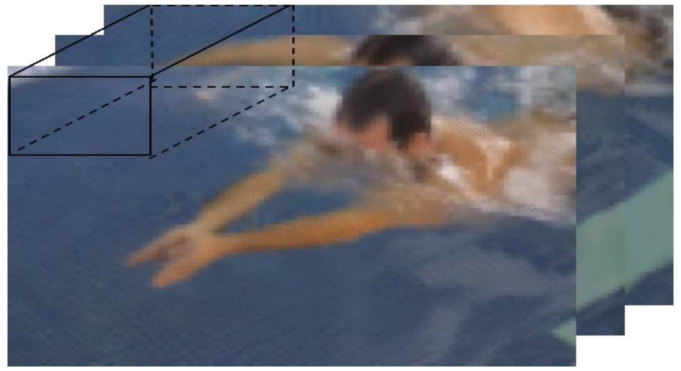


4. C3D Architecture::

- 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.







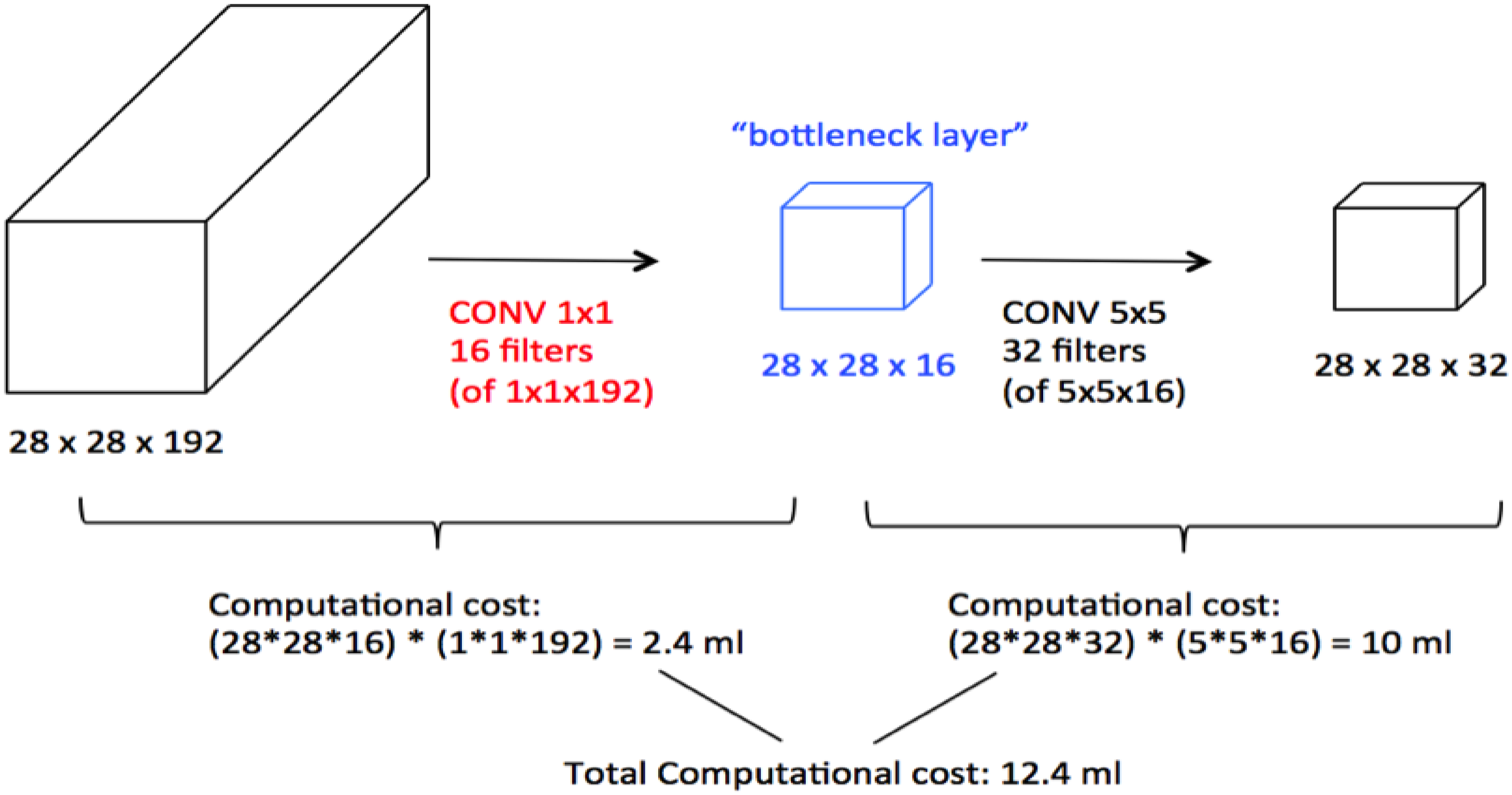
C3D is Temporally extended version of VGG16

4. I3D Architecture::

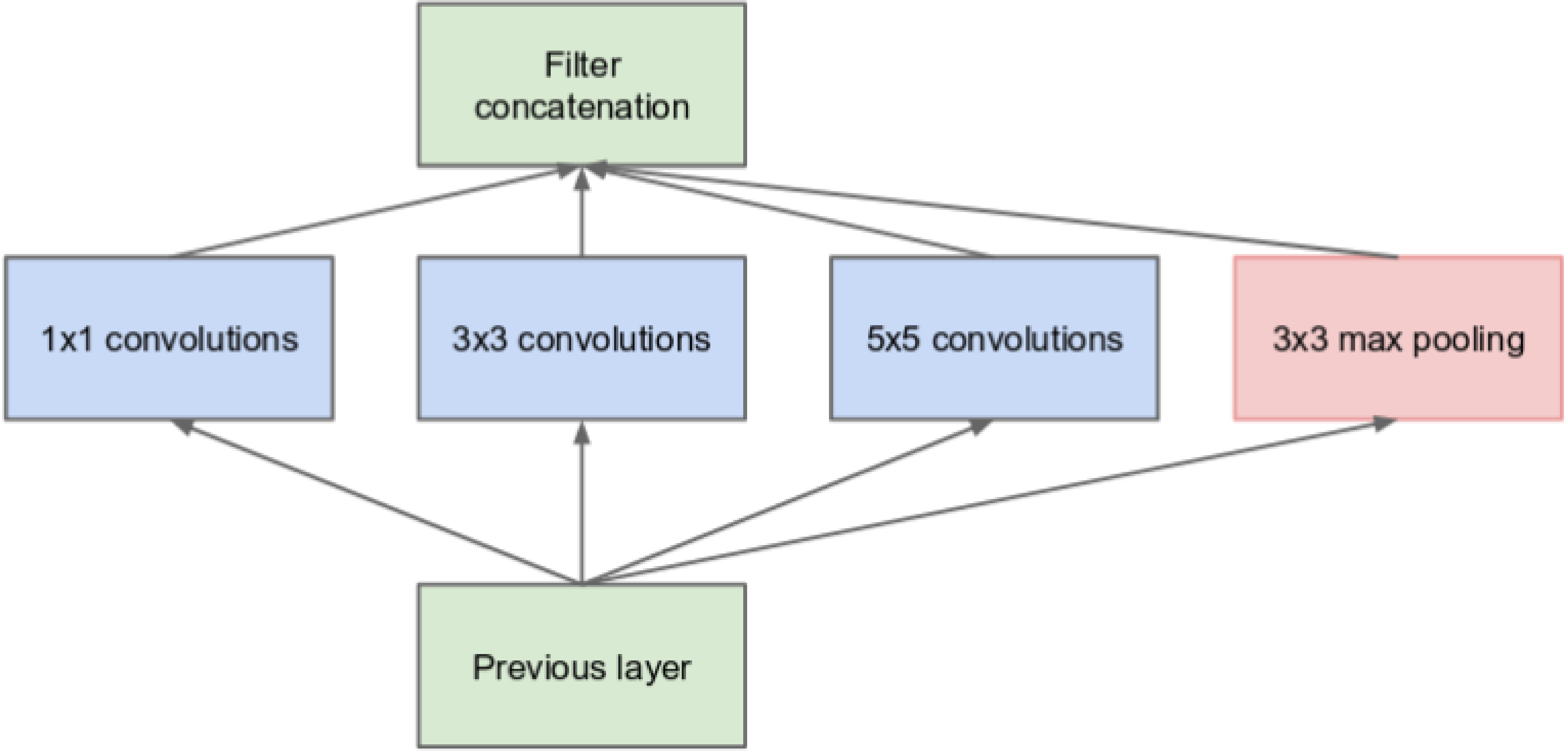
- I3D is designed by replacing the 2D kernels of GoogleNet by 3D kernels.
- It is extended by inflation from the spatial domain.
- Unlike C3D it allows branching in the network architecture.
- Two major component of I3D:
    - **Bottleneck Block**
    - **Inception Block**
- It considers 16/ 64-frames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

I3D is a 3DCNN version of GoogleNet (InceptionV1)

4. Bottleneck Block ::

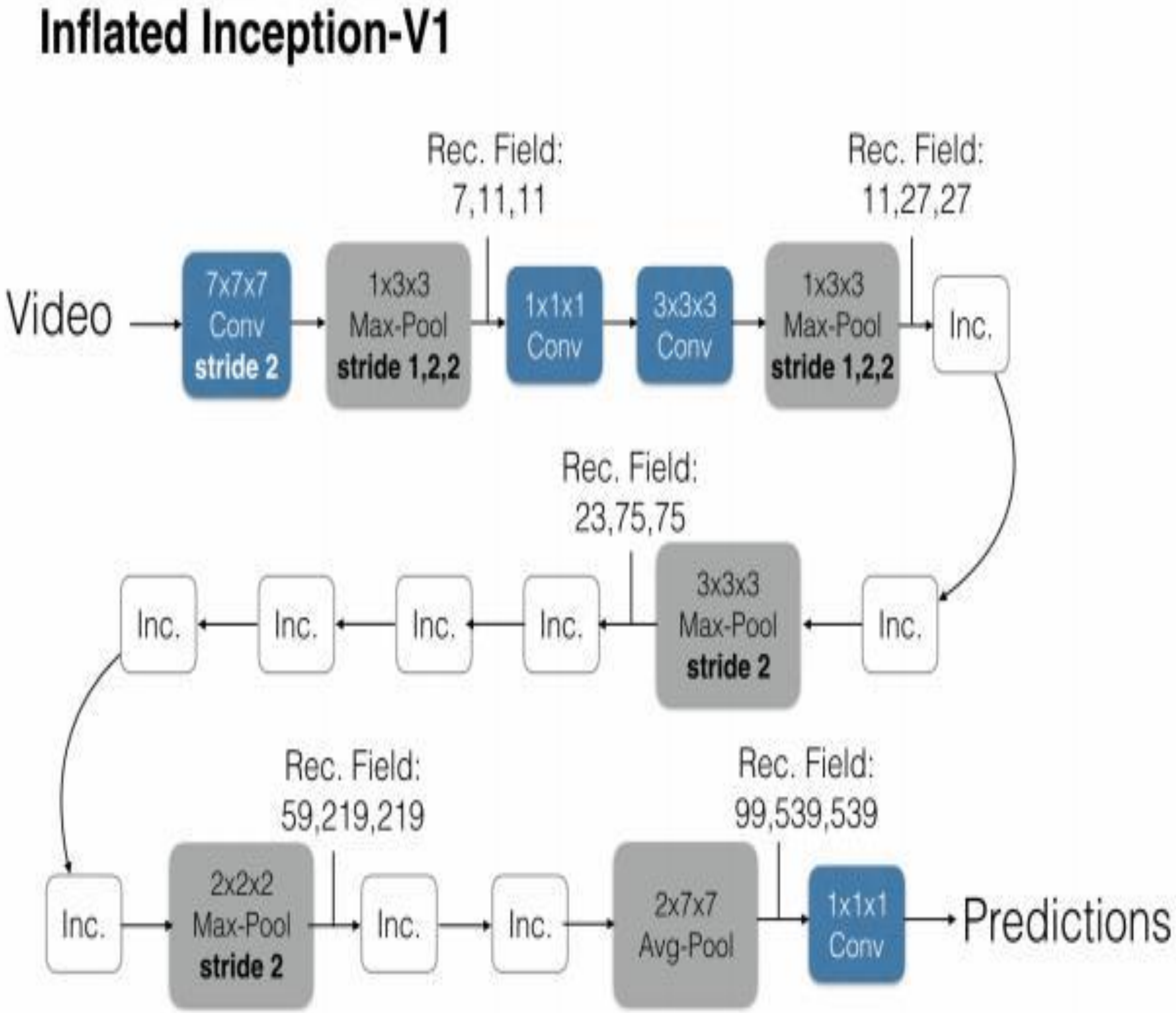


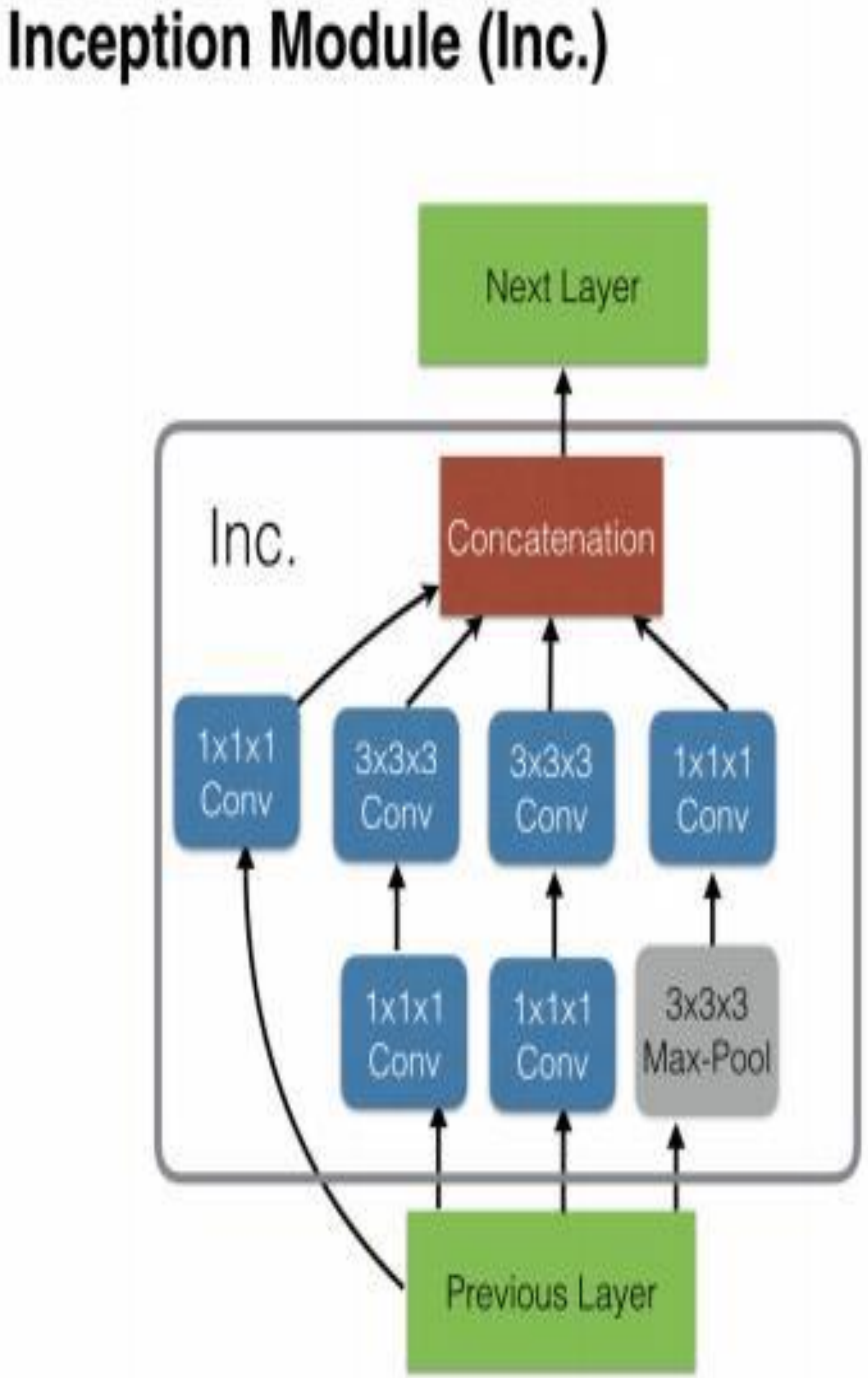
4. Inception Block ::



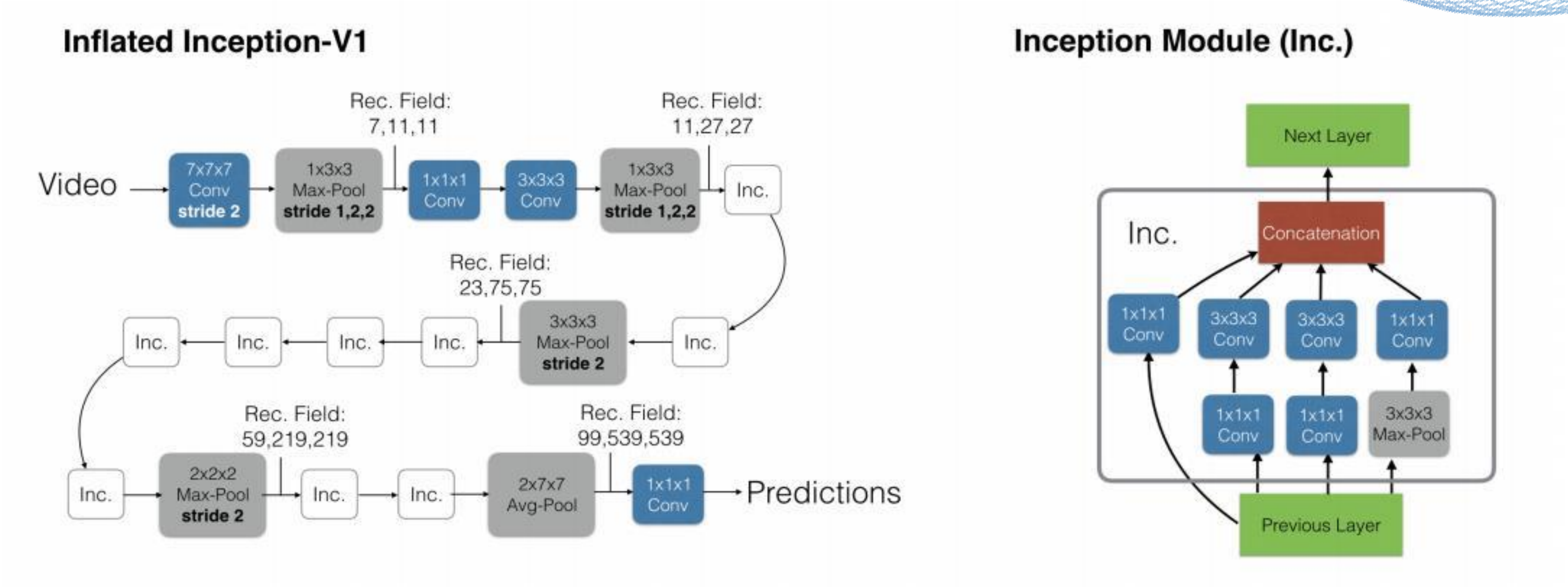
(a) Inception module, naïve version

4. I3D Network ::





4. I3D Network ::

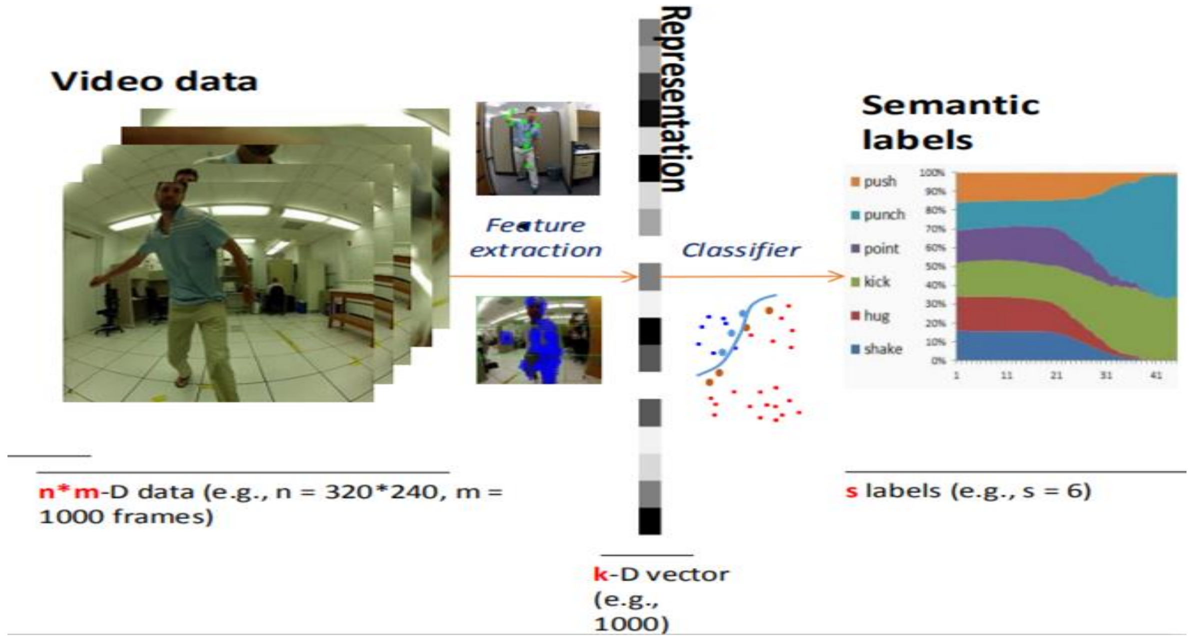


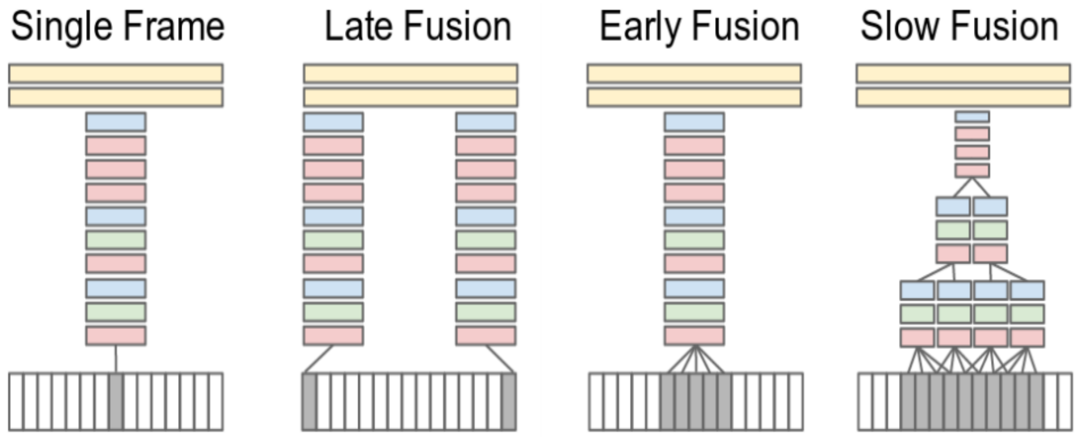
Limitations of 3DCNN

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

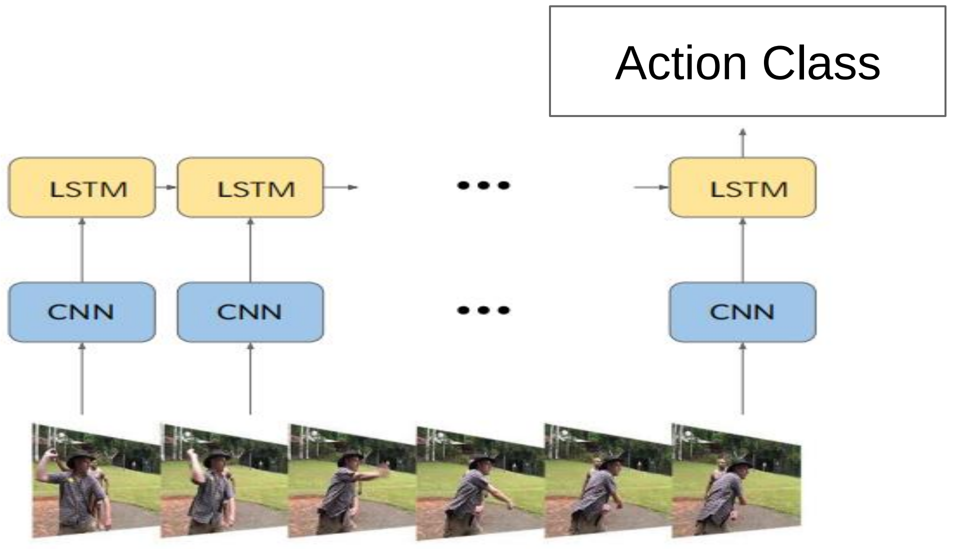
Summary ::

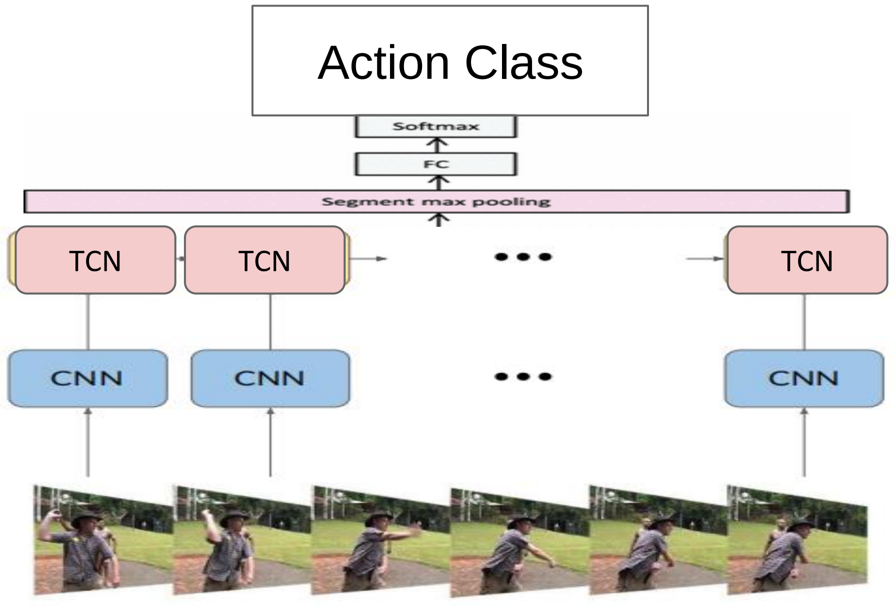
Classical Image Models



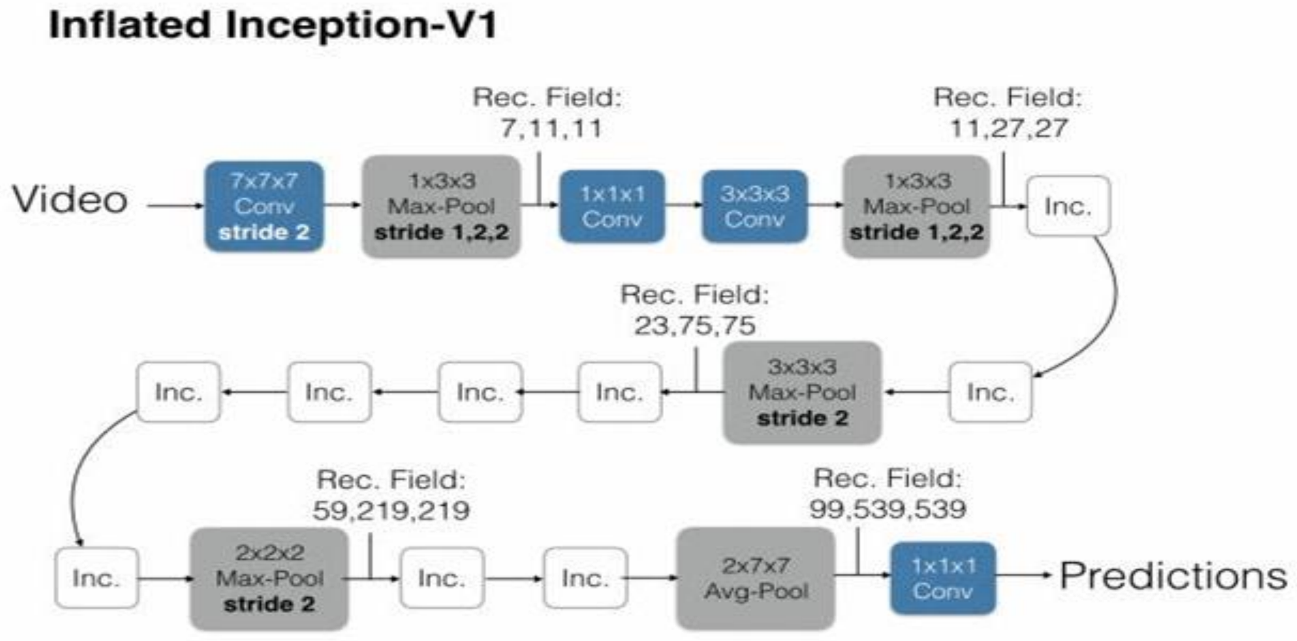


Classical Image Models with Temporal Models





Classical Video Models



After 15 Min. Break

- Introduction to HAR: Human Action Recognition and Challenges
- 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)
  - Vision and Language Models (CLIP)

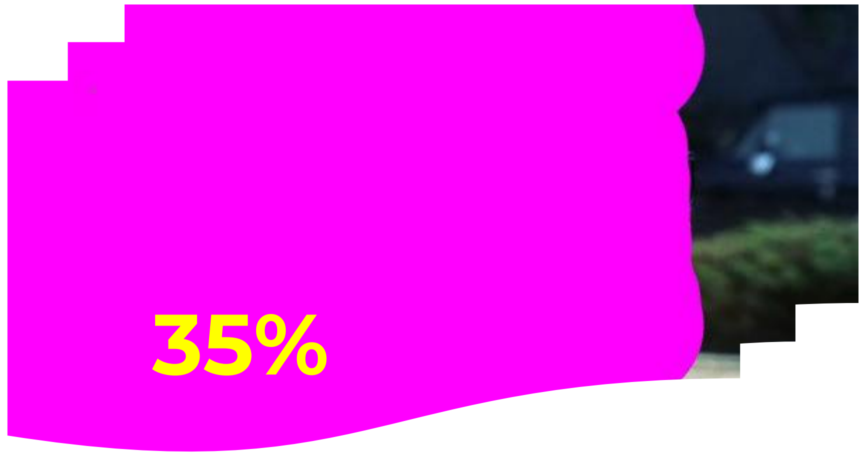
Why Human Action ?

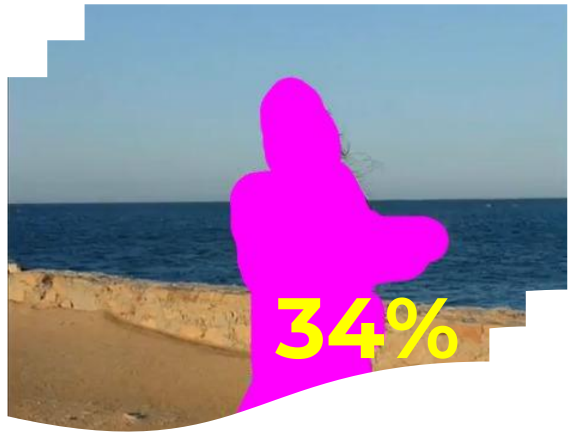
- **How many person-pixels are in the video?**

MOVIE

TV

YouTube





40%

MOVIE

TV

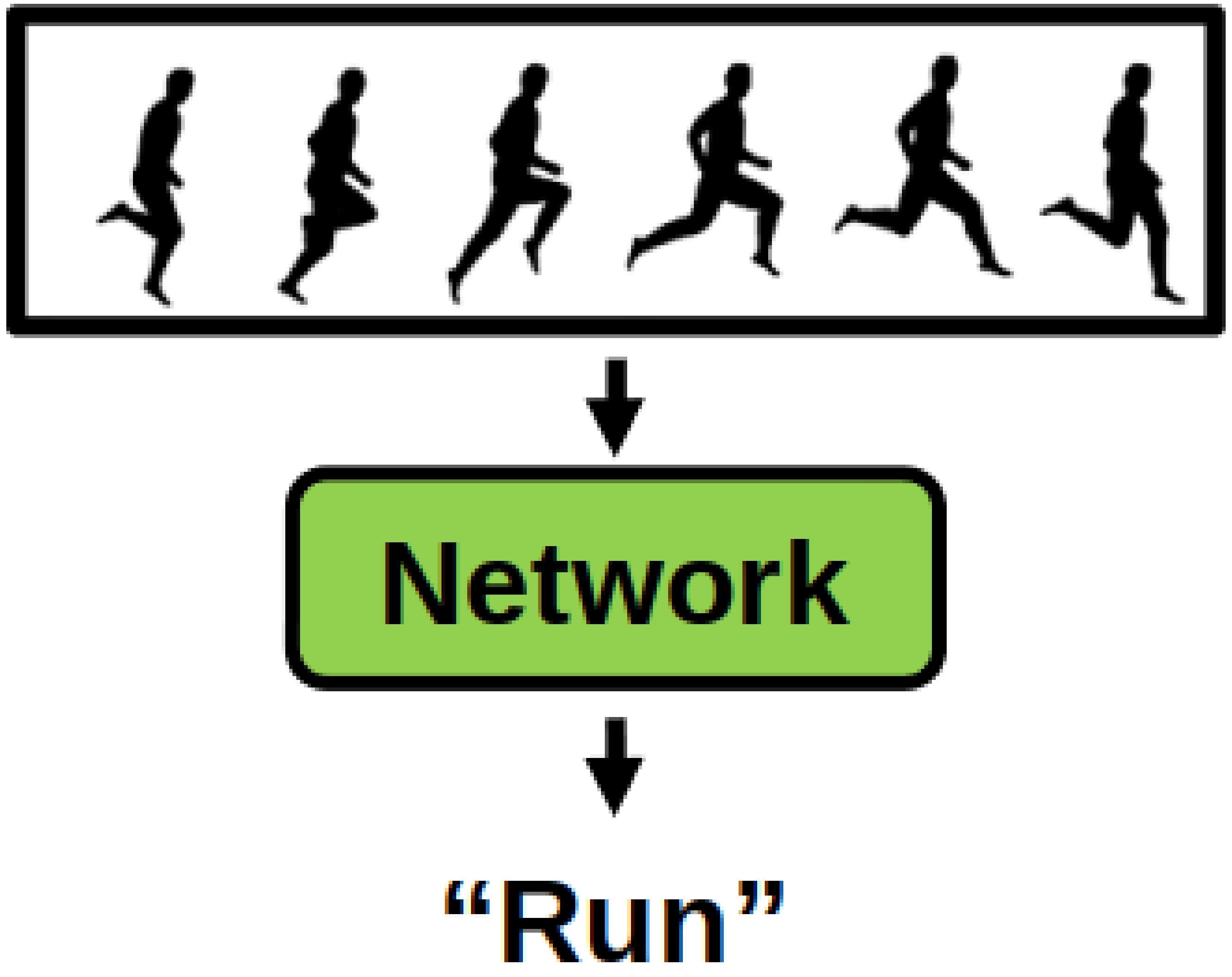
YouTube

Many Videos are Relevant to the HUMANs

Human Action Recognition (HAR) ::

- It can be formulated as a **VIDEO Classification** task and it requires **Holistic human behavior modeling.**

- **Input:** A clipped/trimmed Video **(sequence of Images/Frames)**
- **Output:** An Action Label



Typical Human Actions::

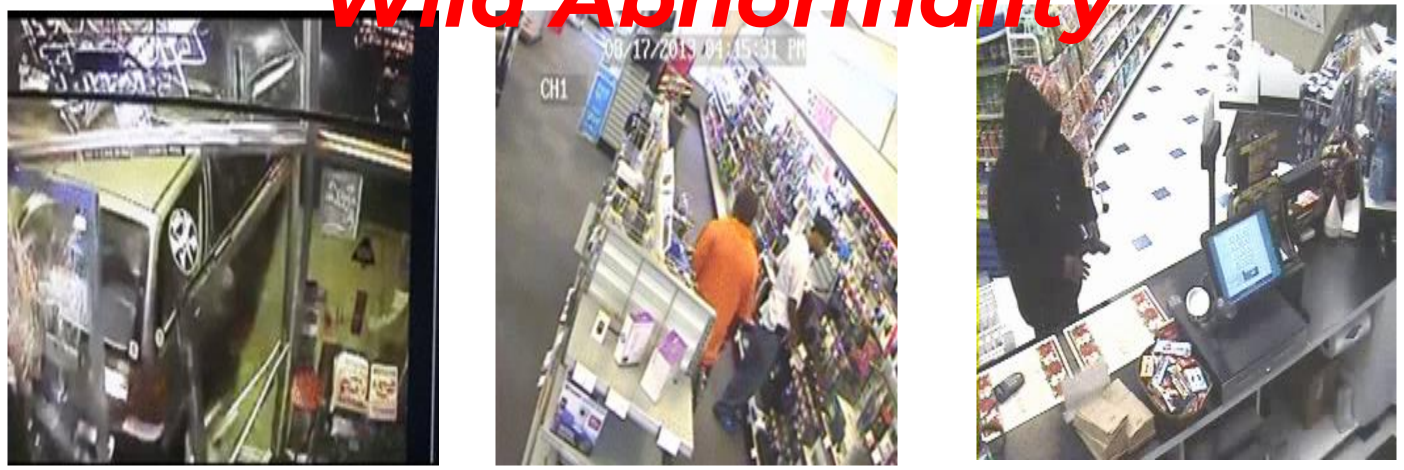
Daily Living

Drink

UseLaptop

Read

Wild Abnormality



Burglary

Shoplift

Robbery

- **Key Challenges:**
  - Subtle Motion
  - High-**Intra**-class Variance
  - Low-**Inter**-class Variance

Challenges::

Subtle Motion:-

Typing Keyboard

Reading

- **Same Background**
- **Almost Similar Posture**
- **<u>Different Actions</u>**

Challenges::

High-Intra-class Variance:-

Drinking

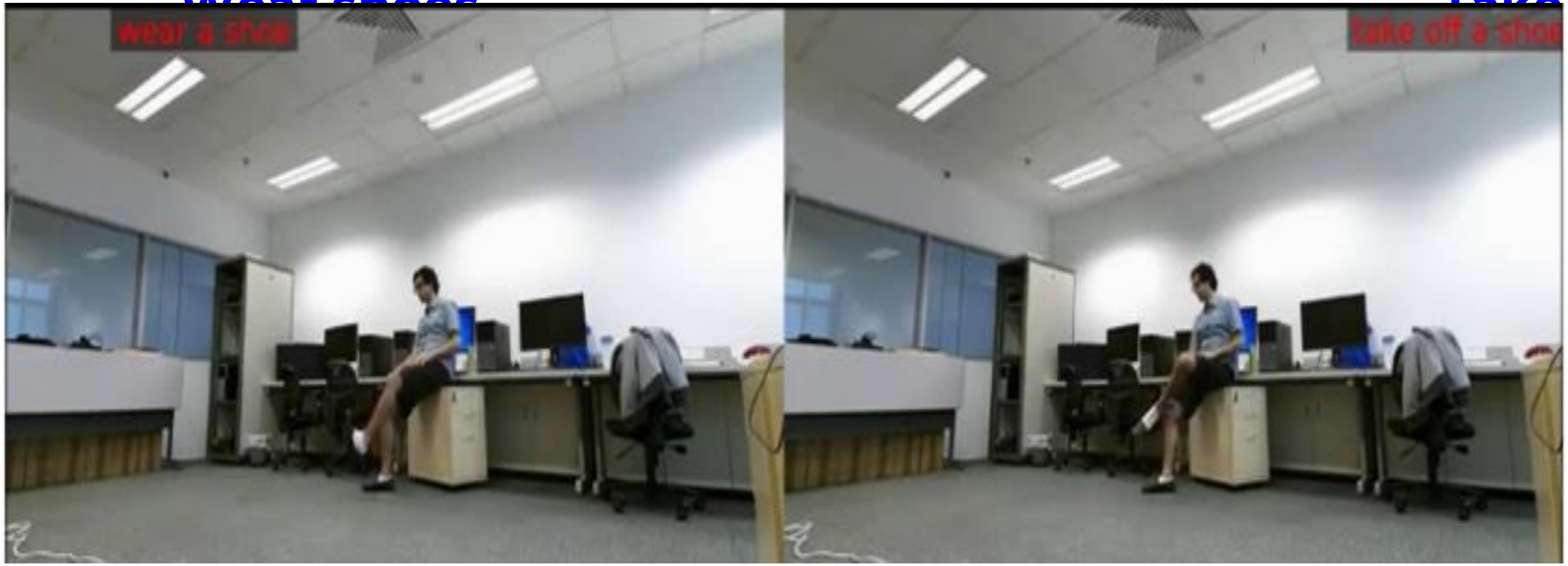
- **Same Background**
- **Same Actions**
- **Different Posture (sit, stand)**

Challenges::

Low-Inter-class Variance:-

Wear shoes

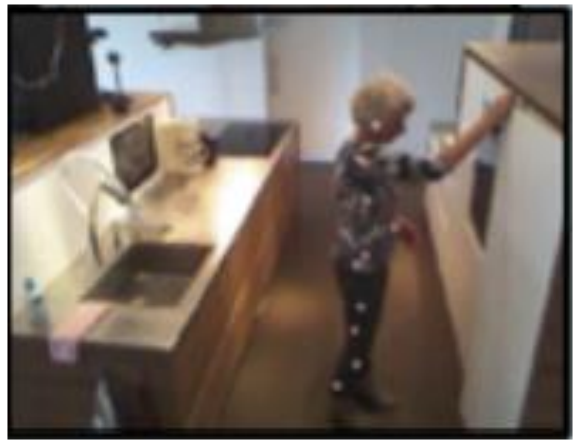
Take off



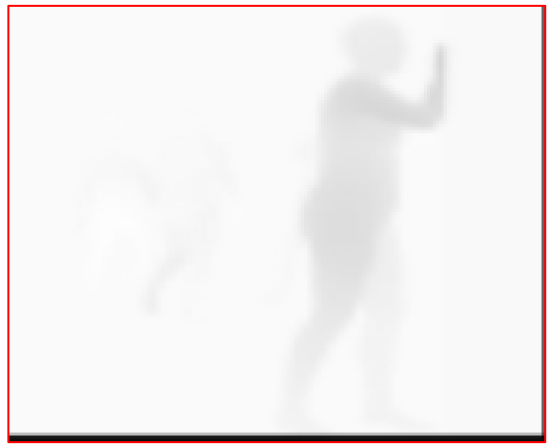
- **Same Background**
- **Almost Similar Posture**
- **Different Actions**

How to Tackle Challenges::

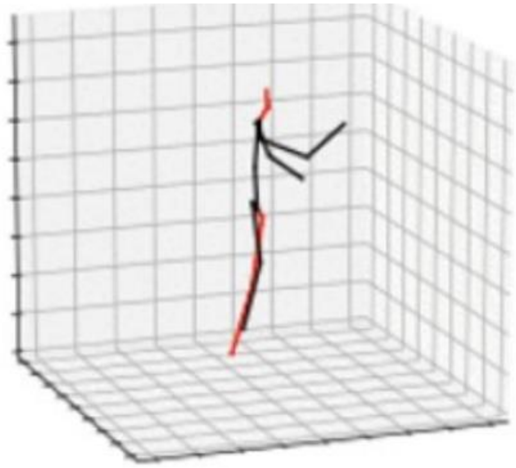
- Usage of Different Modalities to capture unique Cues



Appearance (RGB)



Motion (Optical Flow)



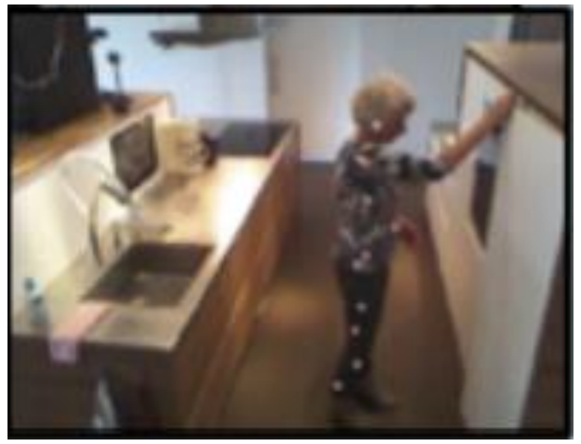
Posture (3D Poses)

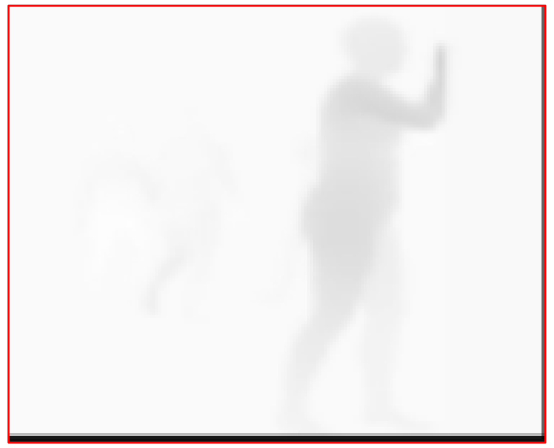
- Discriminative Temporal Modeling (with Attention Mechanism and Transformer Models)

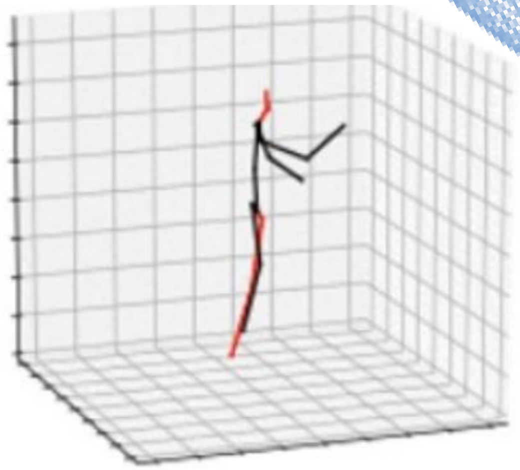
How to Tackle Challenges::

- Usage of Multiple Modalities in IMAGE and VIDEO models to capture category specific unique Cues.
- Salient Feature Learning with Attention Mechanism (Spatial, Temporal, Spatio-Temporal)
- Robust spatio-temporal Feature Correlation Learning with powerful Transformer Models.
- Pre-training Large self-supervised, vision-language model to obtain discriminative human and object centric cues for HAR.

Multiple Modalities::



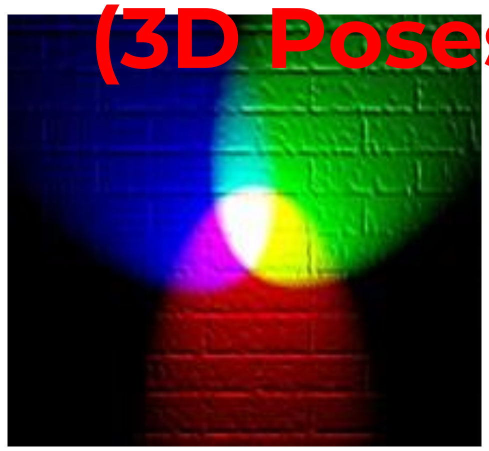




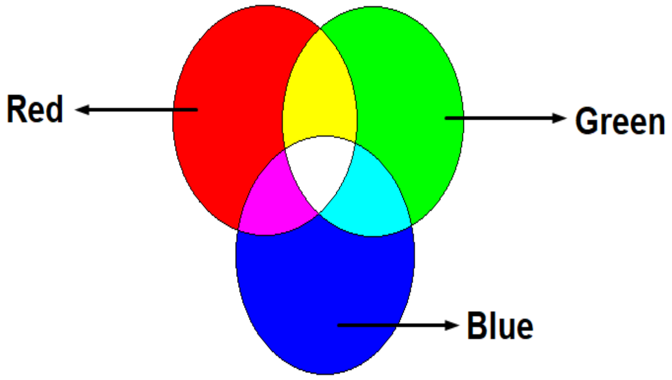
Appearance (RGB)

W

H



Tensor: [H × W × 3] × T



Motion (Optical Flow)

- Computes displacement of each pixel w.r.t. previous Frames
- Represented by you Displacement Vectors: (i) along X-axis, (ii) along y-axis

Tensor: [H × W × 2] × T

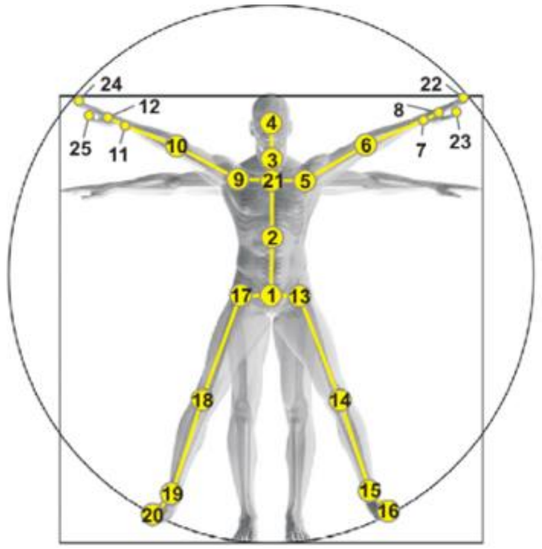
- Acquisition
  - Flow Camera
  - Flow Estimation Algo. (RAFT, TVF1, FlowNet, PwcNet)

Posture (3D Poses/Skeletons)

- 3D Coordinates of 'N' key joints on Human body Tensor.

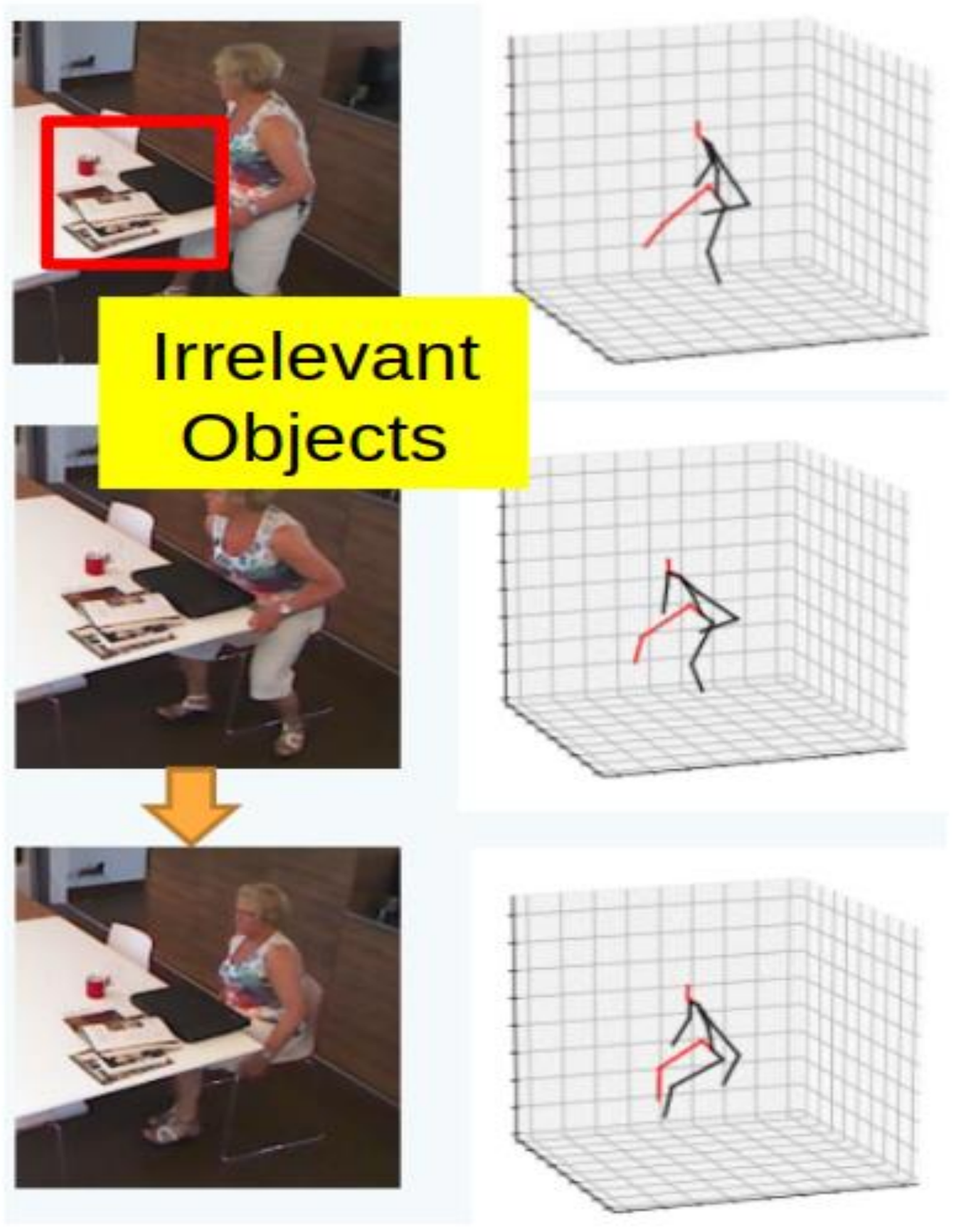
Tensor: [N ×] × T

- Acquisition
  - Kinect Camera
  - Pose Estimation Algo. from RGB images (LCNet, OpenPose, YOLO-V7 Pose)



Benefits of Combining Multiple Modalities::

- Provide complementary information.



Sit down

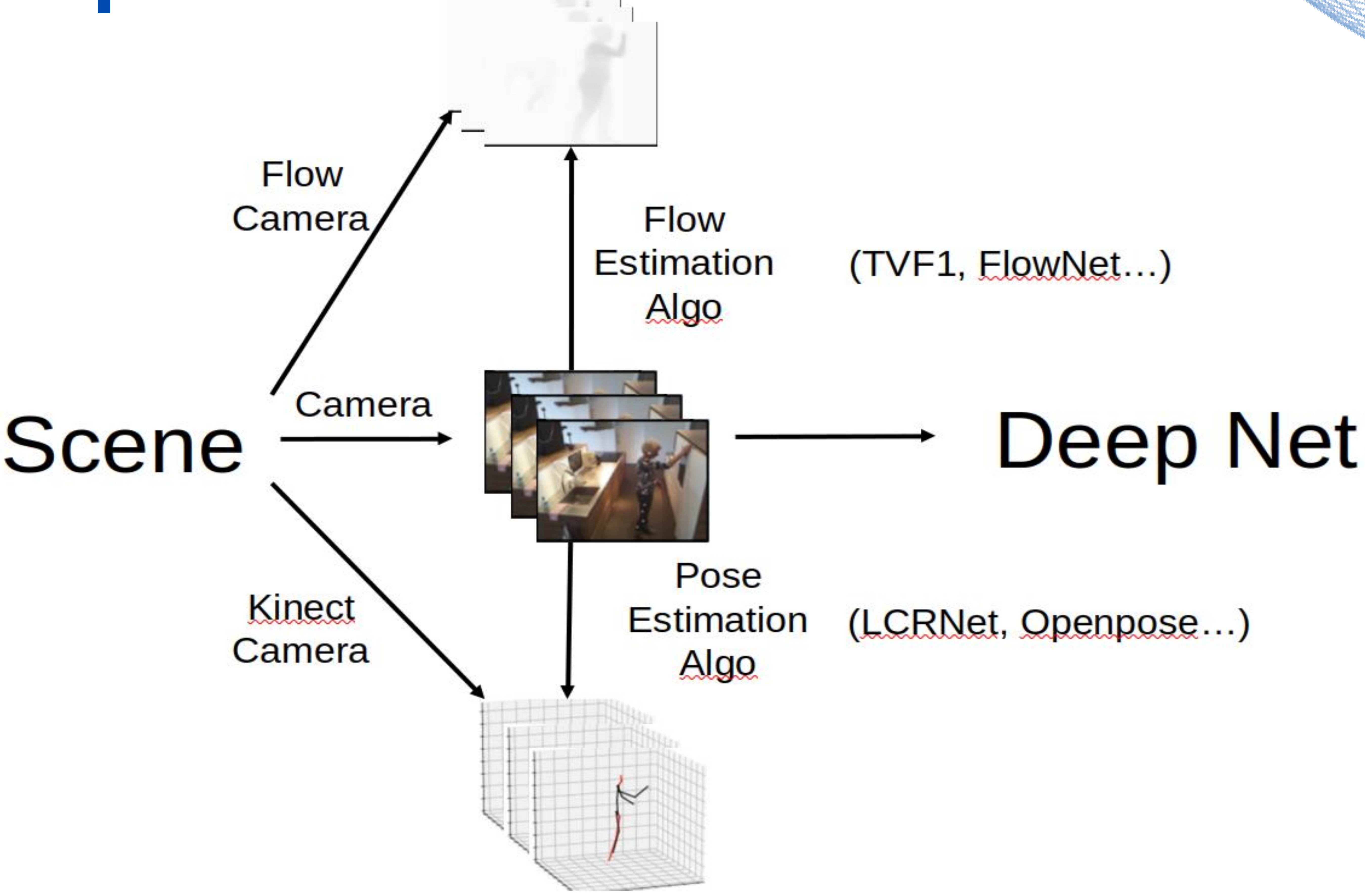
3D poses

Wear glasses

Take off glasses

Optical flow

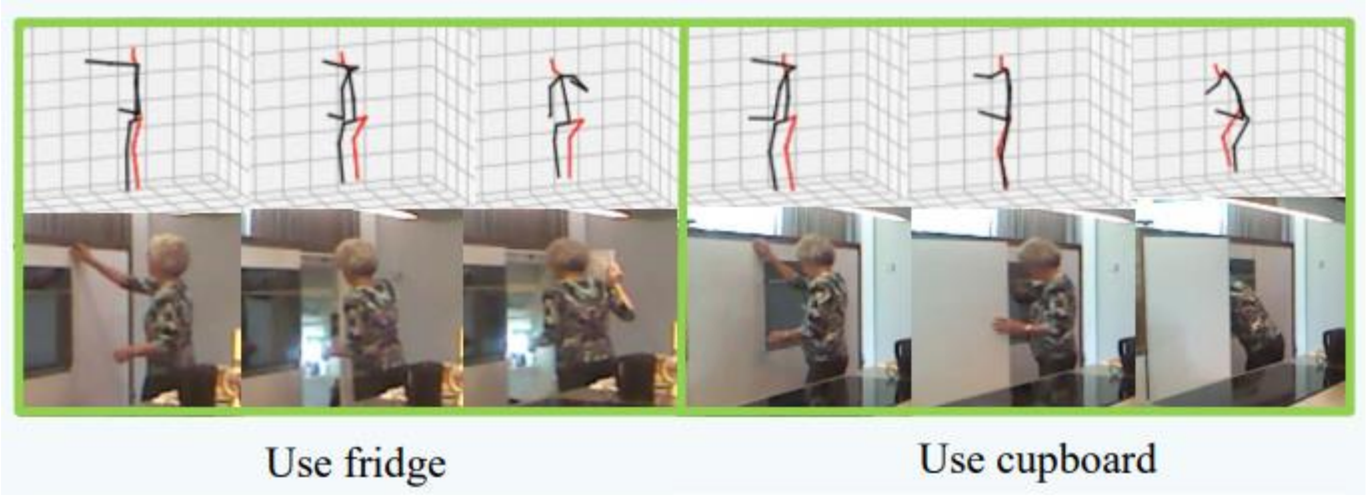
Benefits of Combining Multiple Modalities::



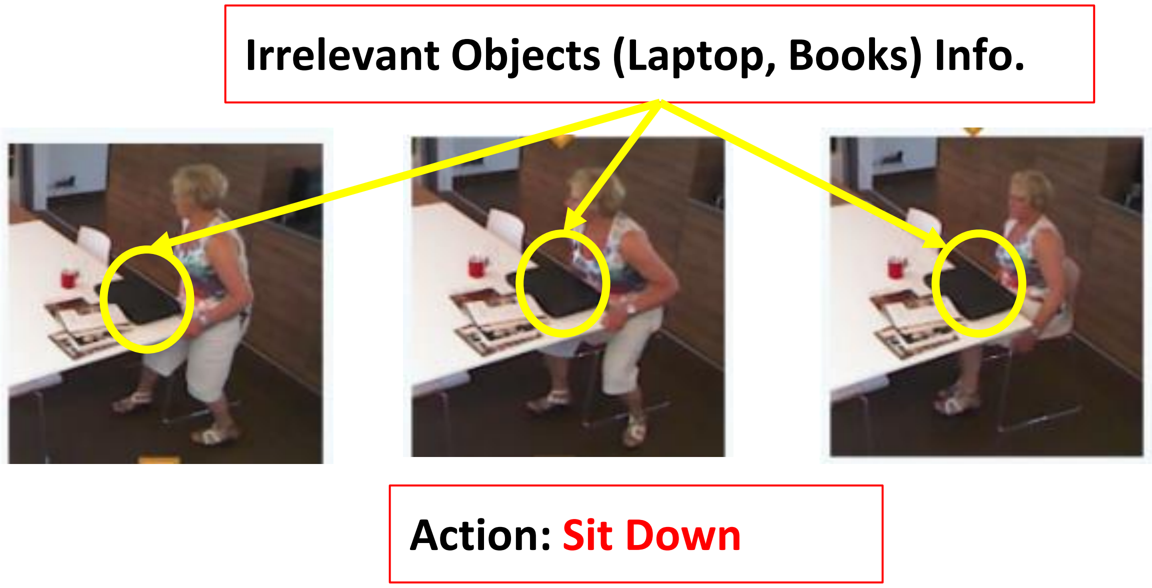
Drawbacks of Different Modalities::

- **Optical Flow:**
  - Time consuming in extracting Flow from RGB at Inference
  - Scenario information is missing

- **3D Poses:**
  - Object information is missing



- **RGB:**
  - Contains Most Information but can be Noisy as well.

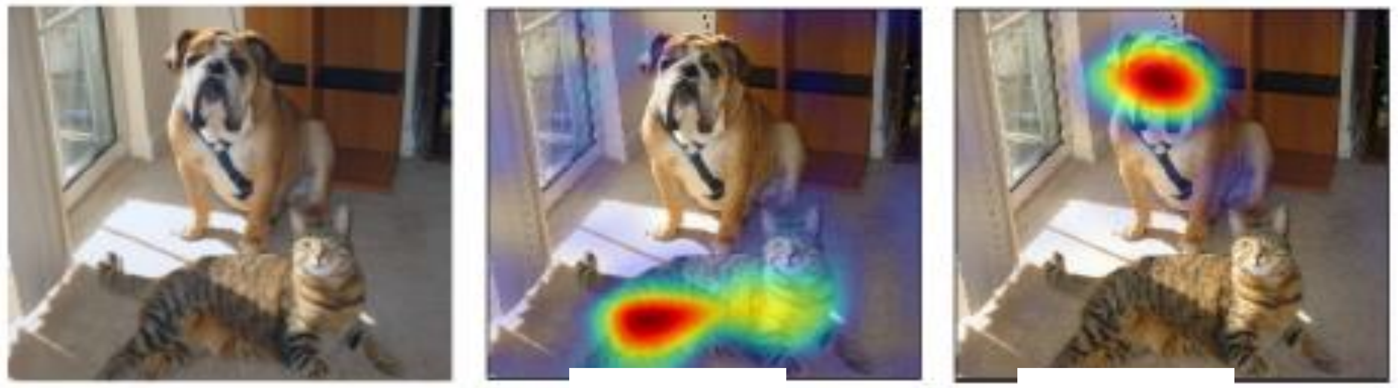


Attention Mechanism::

- **Primary purpose of Attention:** To imitate human visual cognitive systems and focus on essential features. (or) <u>Learn how to pick relevant information from input data.</u>

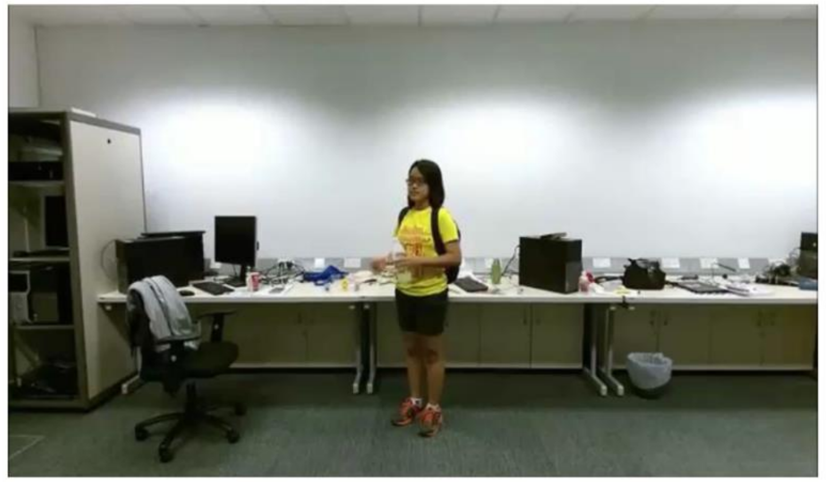
- **Key Idea:** To focus on the significant parts in an image and suppress unnecessary information.

- **CNN with Attention:** are used to make CNN learn and focus more on the important information, rather than learning non-useful background information.

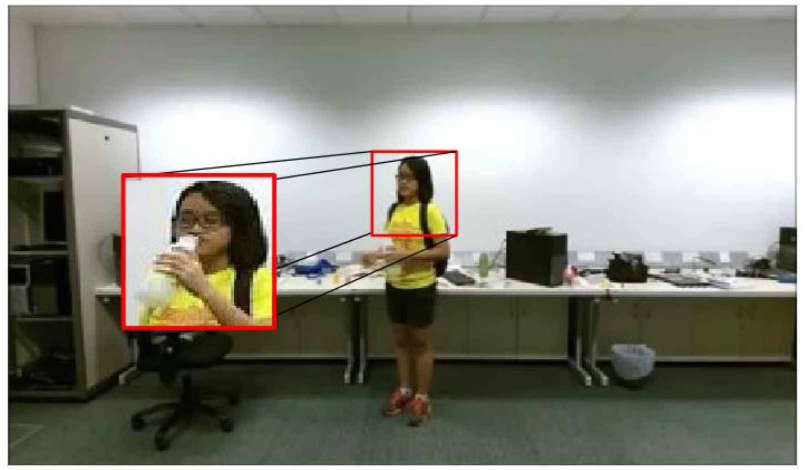


Original Image | Focus on 'Cat' | Focus on 'Dog'

The girl is drinking water from a bottle

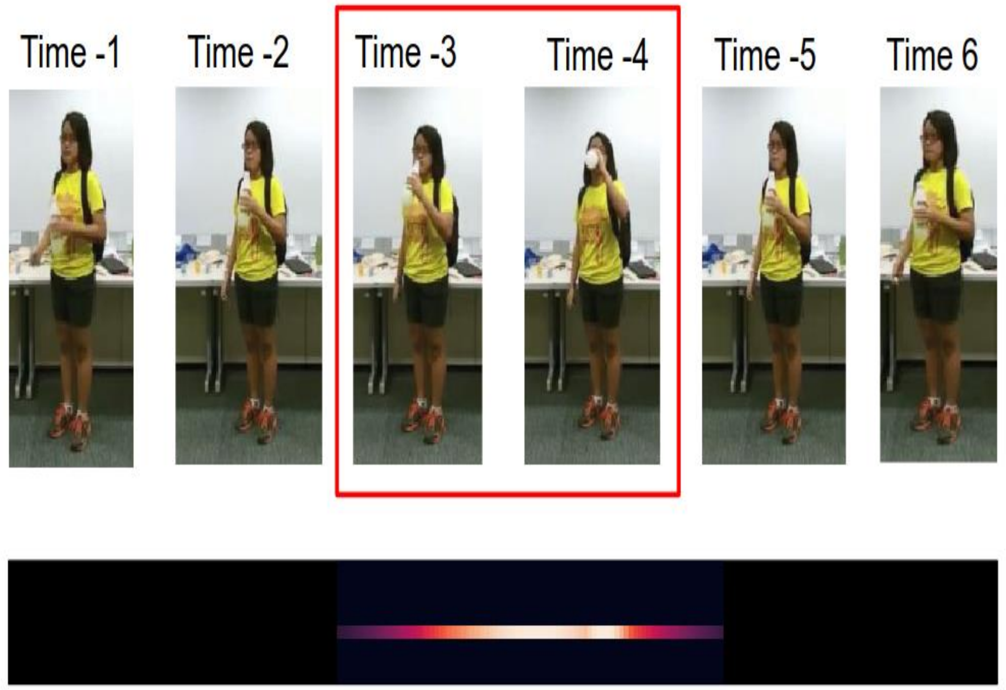


Do we really need the whole video to infer that?



- Isn't this enough for an inference?

Focus in the **Spatial** space is required!



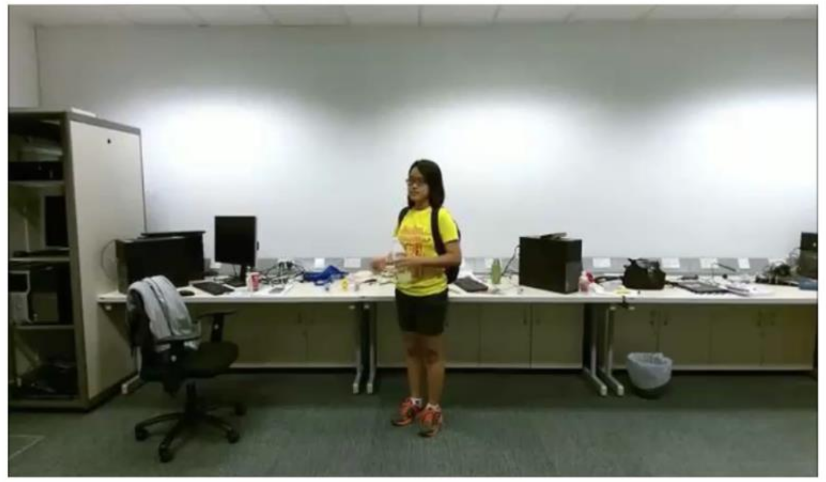
Spatial Attention

Temporal Attention

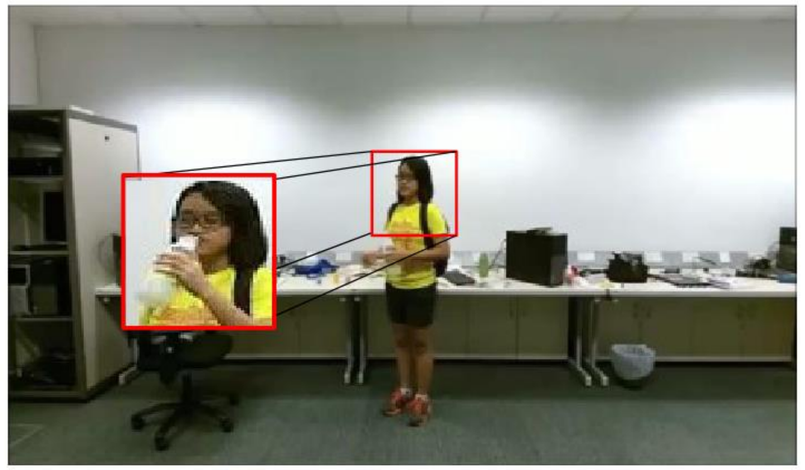
Classical Attention Mechanism::

- **Squeeze-and-Excitation Attention (Channel Attention)**
- **Convolutional Block Attention Module (Channel + Spatial Attention)**
- **Spatial-Temporal Attention**
- **Self-Attention**

The girl is drinking water from a bottle

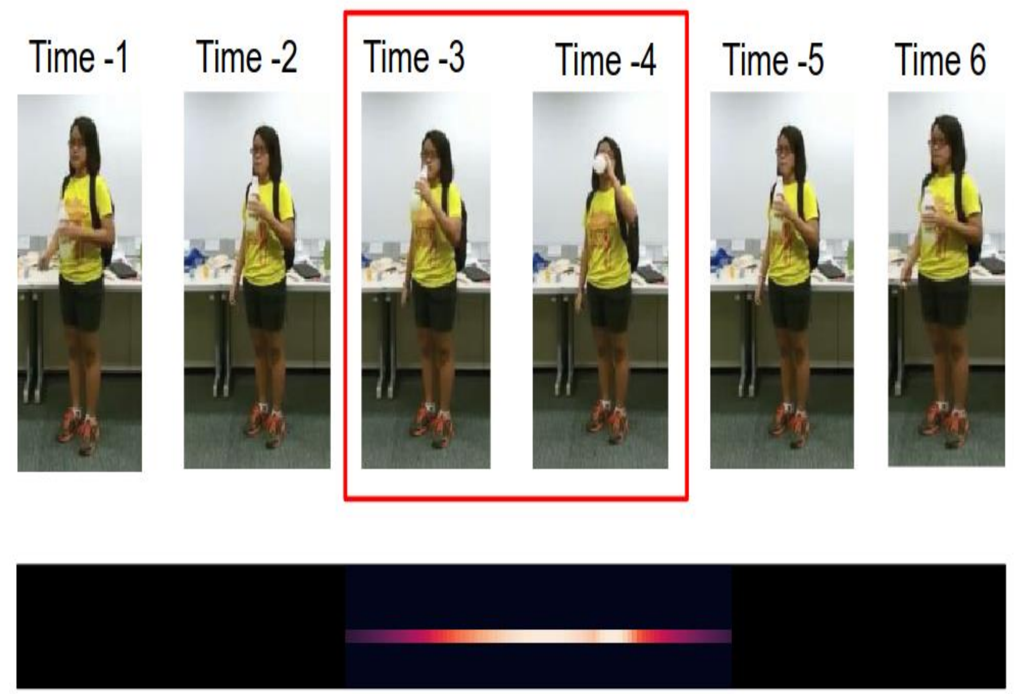


Do we really need the whole video to infer that?



- Isn't this enough for an inference?

Focus in the **Spatial** space is required!



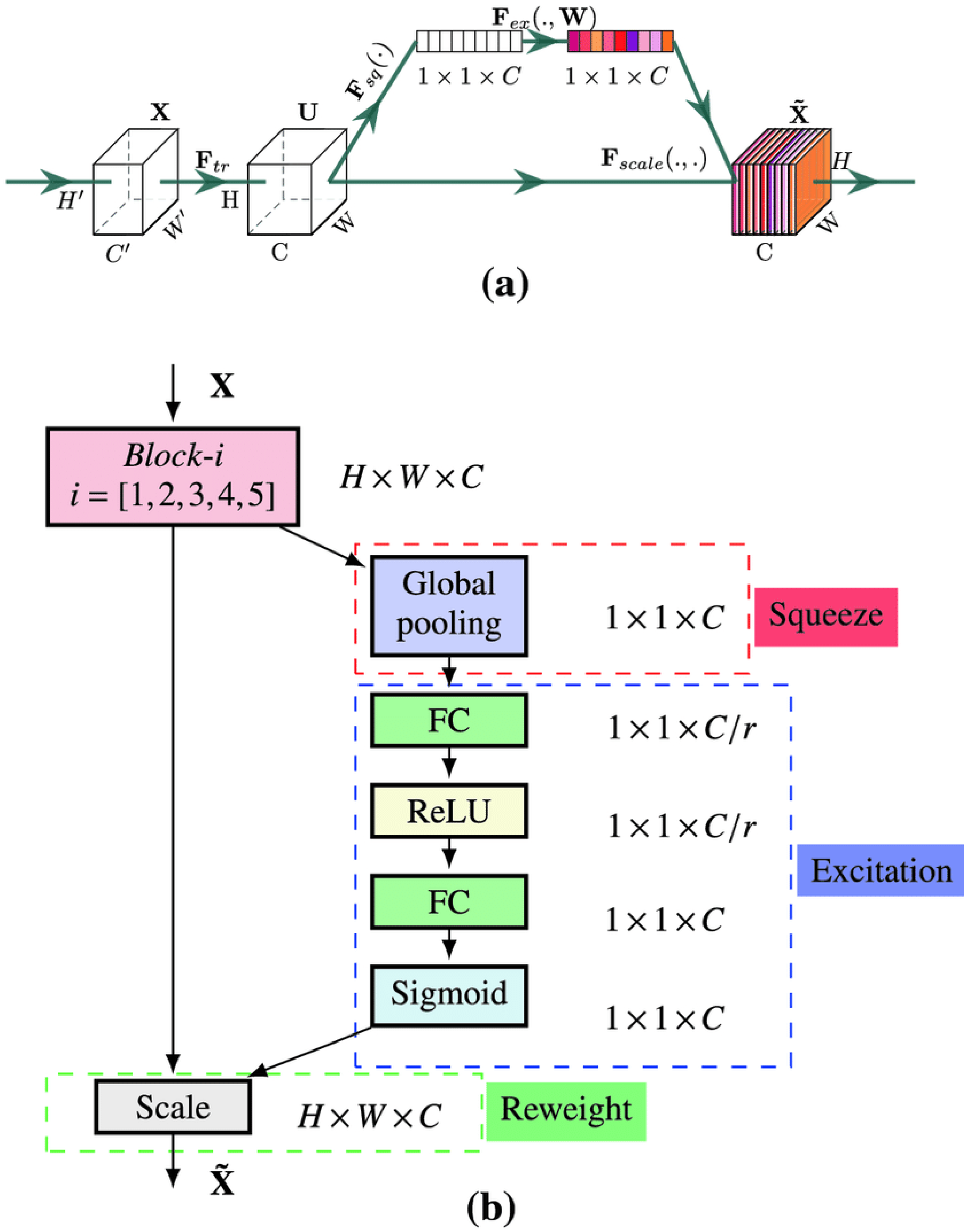
Spatial Attention

Temporal Attention

Squeeze-and-Excitation Attention::

- **Observation in CNN:**
  - Feature Extraction from CNN shrinks the spatial Dimension and expands the channel dimension
  - All channels are weighted equally when considering the output feature map

- **Key Idea:** Assign each channel a different weightage based on how important each channel is .



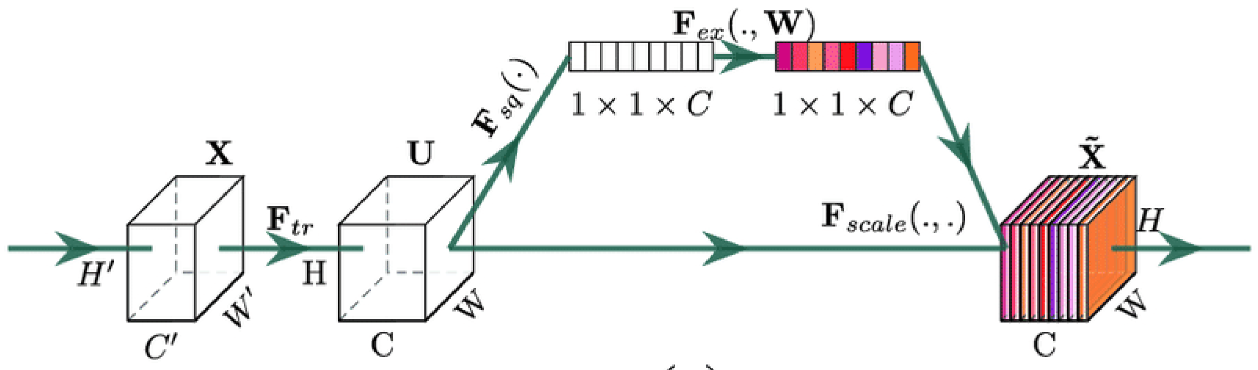
Squeeze-and-Excitation Attention::

3 main Parts of SE:

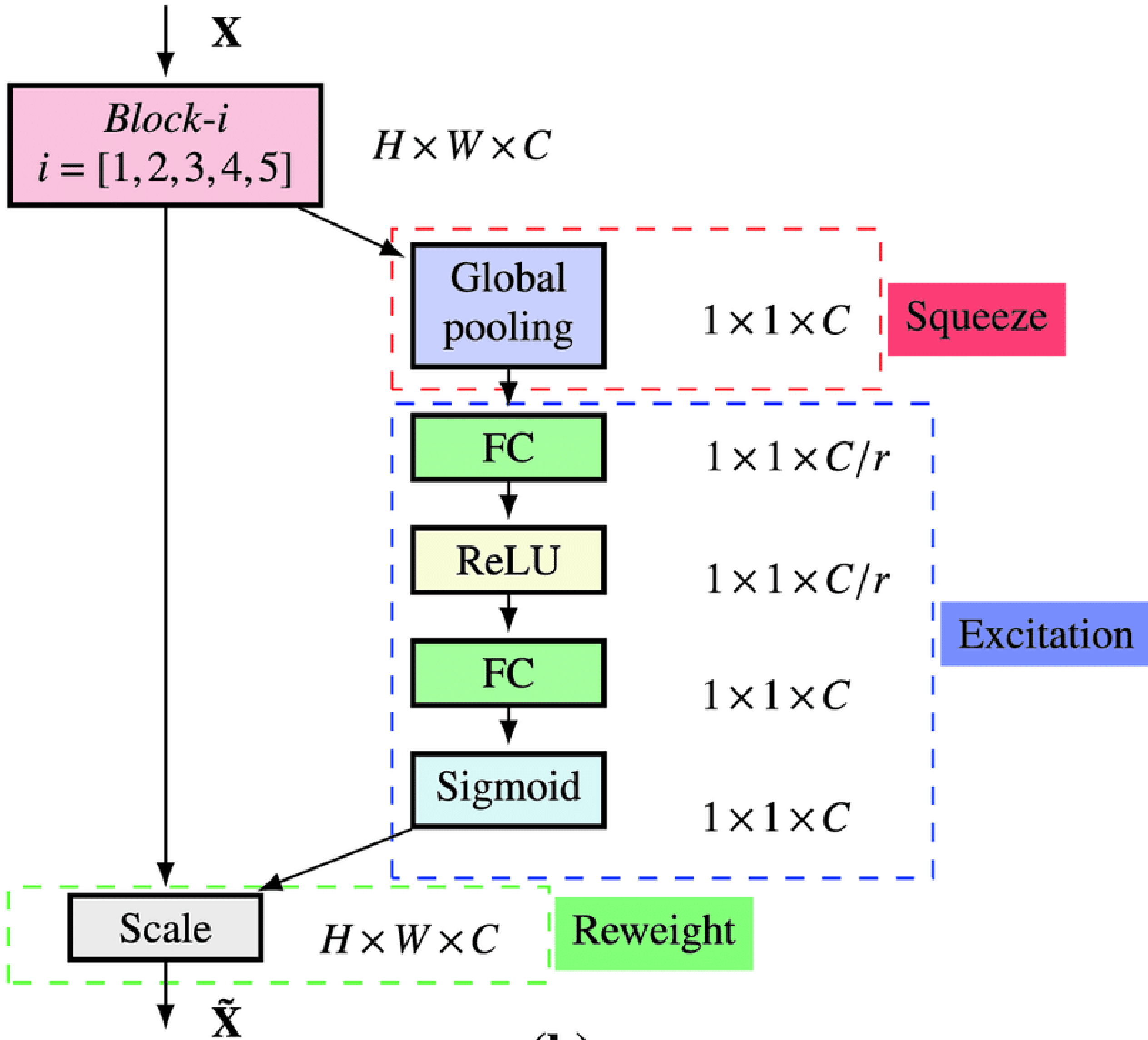
Squeeze: Global Average Pooling is performed on the output feature map of the CNN layer across H and W and the result of output tensor shape is 1 x 1 x C.

Excitation: Vector from the previous operation is passed through two successive Fully-Connected Layers. This serves the purpose of fully capturing channel-wise dependencies that were aggregated from the spatial maps. A ReLU activation is performed after the first FC layer, while the sigmoid activation is used after the second FC layer. In the paper, there is also a reduction ratio such that the intermediate output of the first FC layer is of a smaller dimension. The final output of this step also has a shape (1 x 1 x C).

Reweight: Lastly, the output of the computation step is used as a per-channel weight modulation vector. It is simply multiplied with the original input feature map of size (H x W x C). This scales the spatial maps for each channel according to their 'importance'.



(a)



(b)

Squeeze-and-Excitation Attention::

SE Blocks can be easily integrated with many existing CNNs like Inception V1, ResNets, etc.

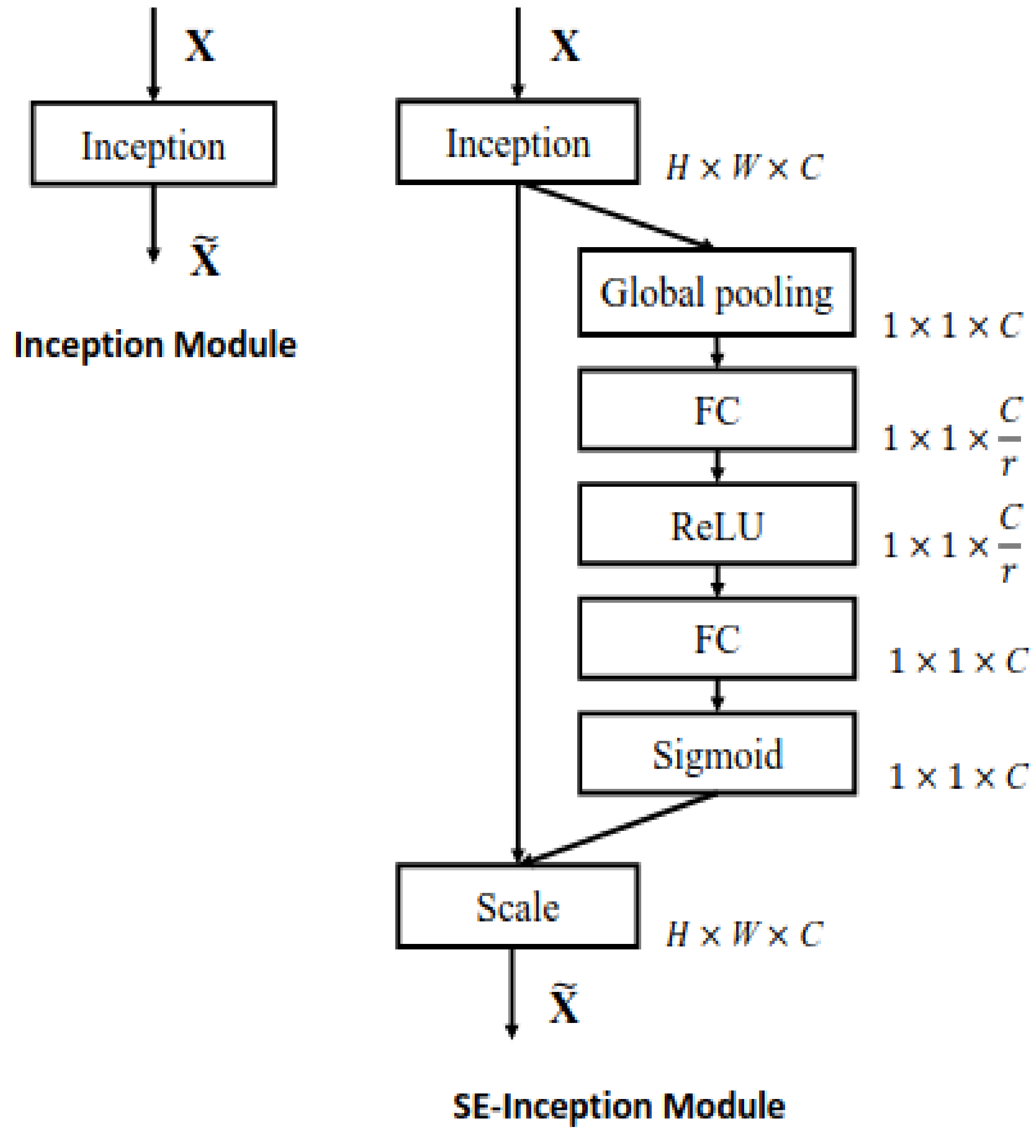


Fig. 2. The schema of the original Inception module (left) and the SE-Inception module (right).

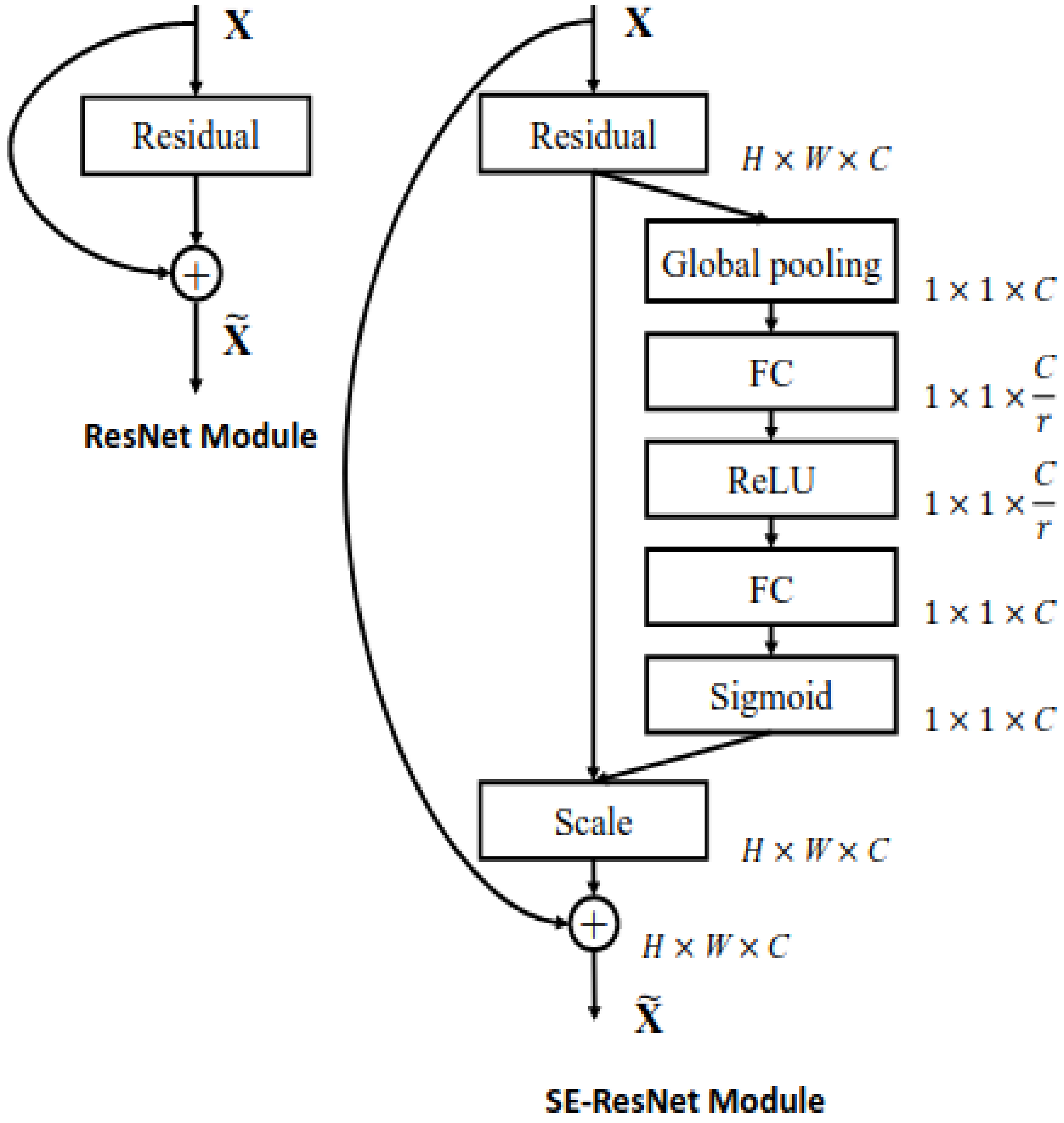
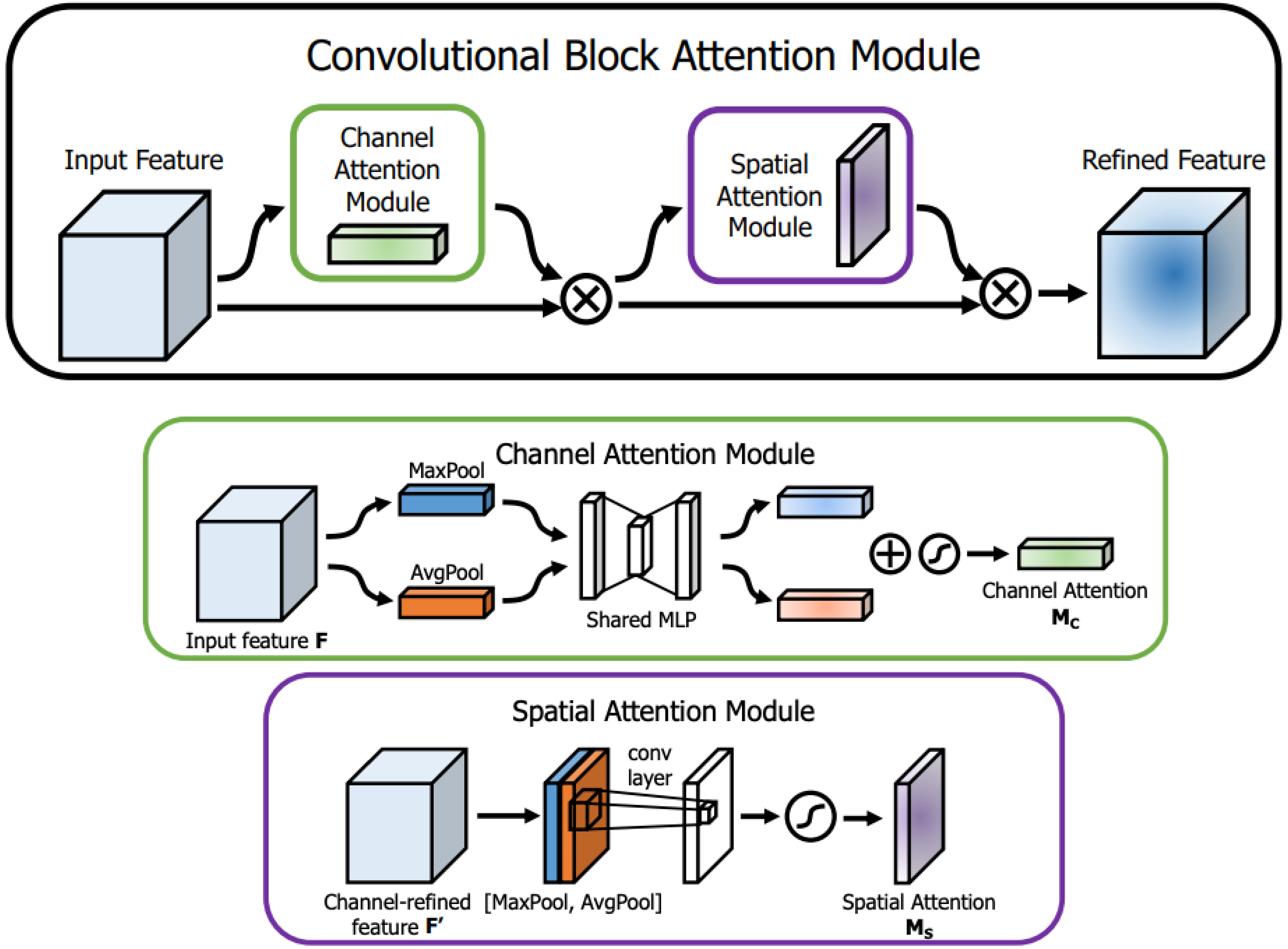


Fig. 3. The schema of the original Residual module (left) and the SE ResNet module (right).

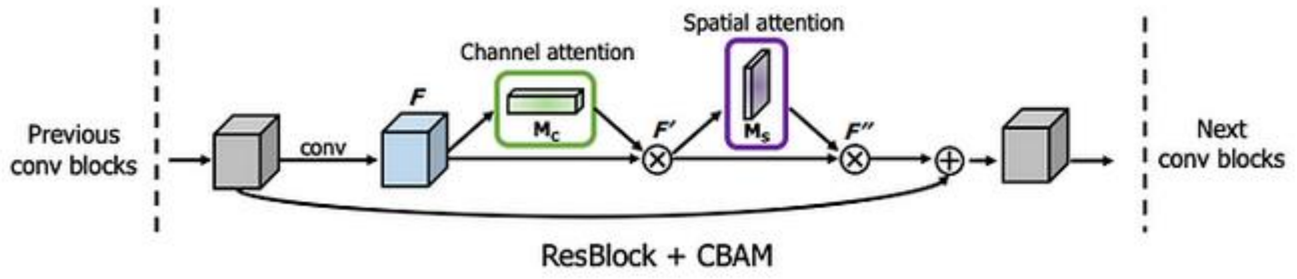
Convolutional Block Attention Module (CBAM)::

- **Key Idea:** To combine both channel and spatial attention, thus CBAM has two sequential sub-modules :
  - **Channel Attention Module (CAM):** Similar to SE attention with a small modification, i.e. instead of single AVERAGE pooling, CAM applies both AVERAGE and MAX pooling to preserves much richer contextual cues.
  - **Spatial Attention Module (SAM):** is three-fold sequential operations, **(i)Channel Pool** that decomposes a ($c \times h \times w$) dimension input tensor to 2 channels, i.e. ($2 \times h \times w$), where each of the 2 channels represent Max Pooling and Average Pooling across the channels. **(ii) Convolutional Layer, (iii) Batch Norm**

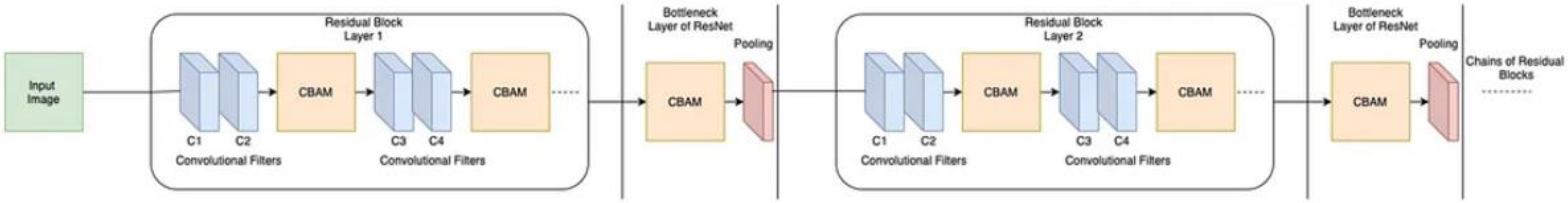


- **CBAM** is applied at every convolutional block in deep networks to get subsequent **"Refined Feature Maps"** from the **"Input Intermediate Feature Maps".**

Convolutional Block Attention Module (CBAM)::



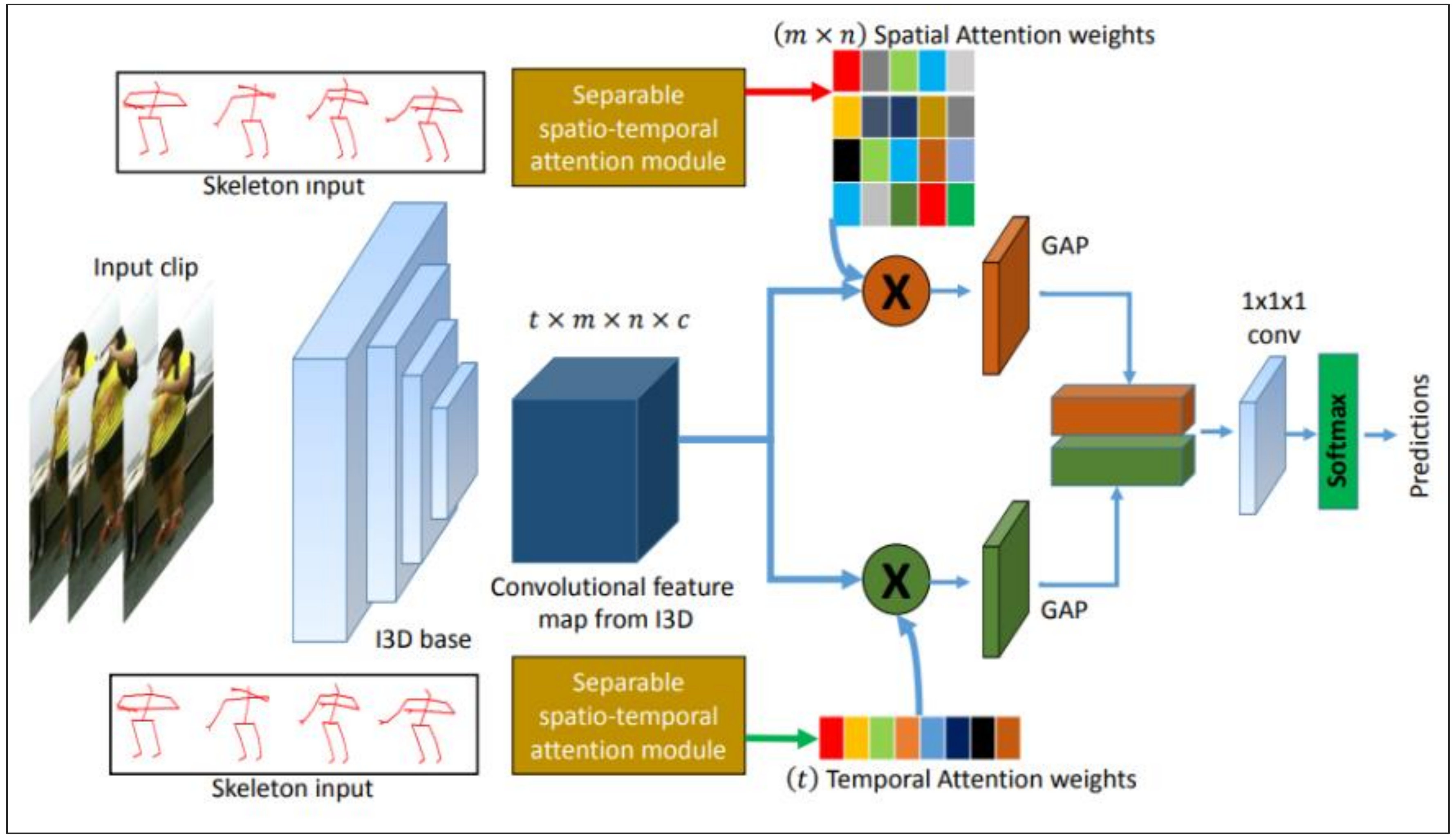
Placement of Spatial and Channel Attention Modules sequentially.

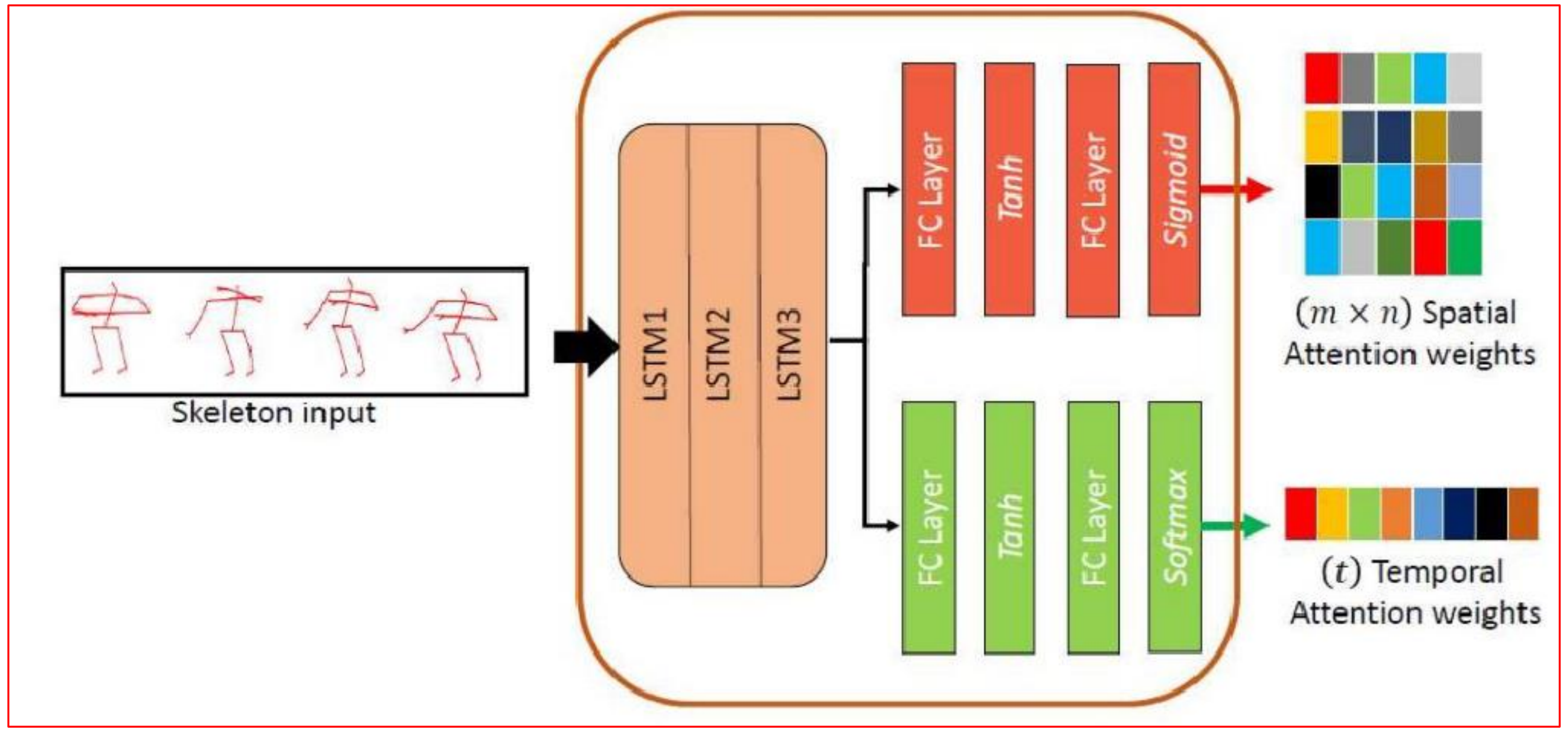


Placement of CBAM module in ResNet architecture.

Spatio-Temporal Attention::

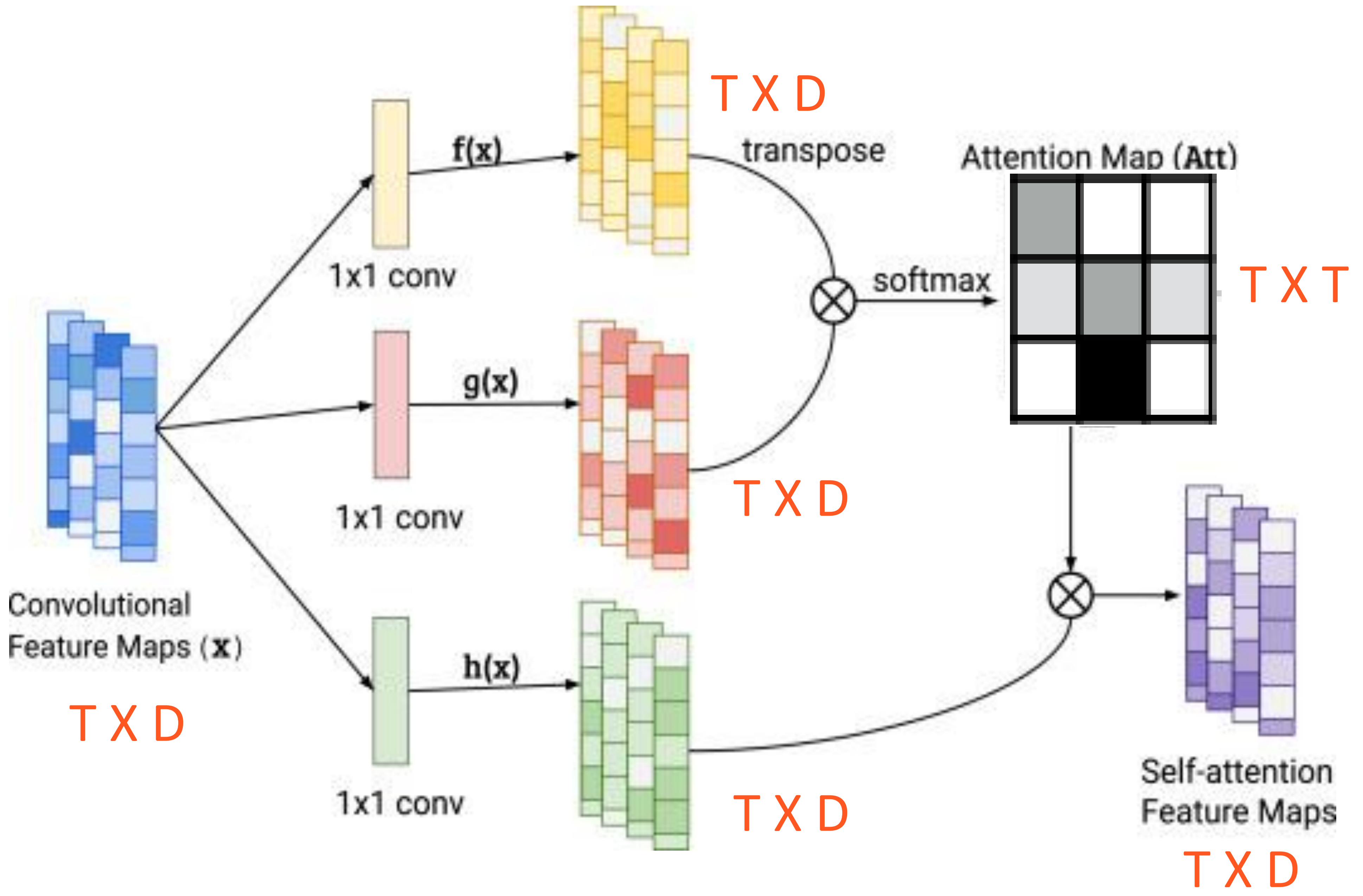
- **Key Idea:** To learn pose driven attention mechanism for highlighting the spatial and temporal saliency of human actions in a dissociative/separate manner.
  - Coupling spatial and temporal attention is difficult for spatio-temporal 3D ConvNet features.
  - spatial attention: focus on the important parts of the image, temporal attention: focus on the pertinent/salient segments of the video.
  - Uses **3D skeleton poses** to compute the spatio-temporal attention weights.
  - It uses **stacked-LSTM to encode the temporal consistency** of 3D skeletons, which is **first pre-trained** and then used for attention map computation.





Self Attention::

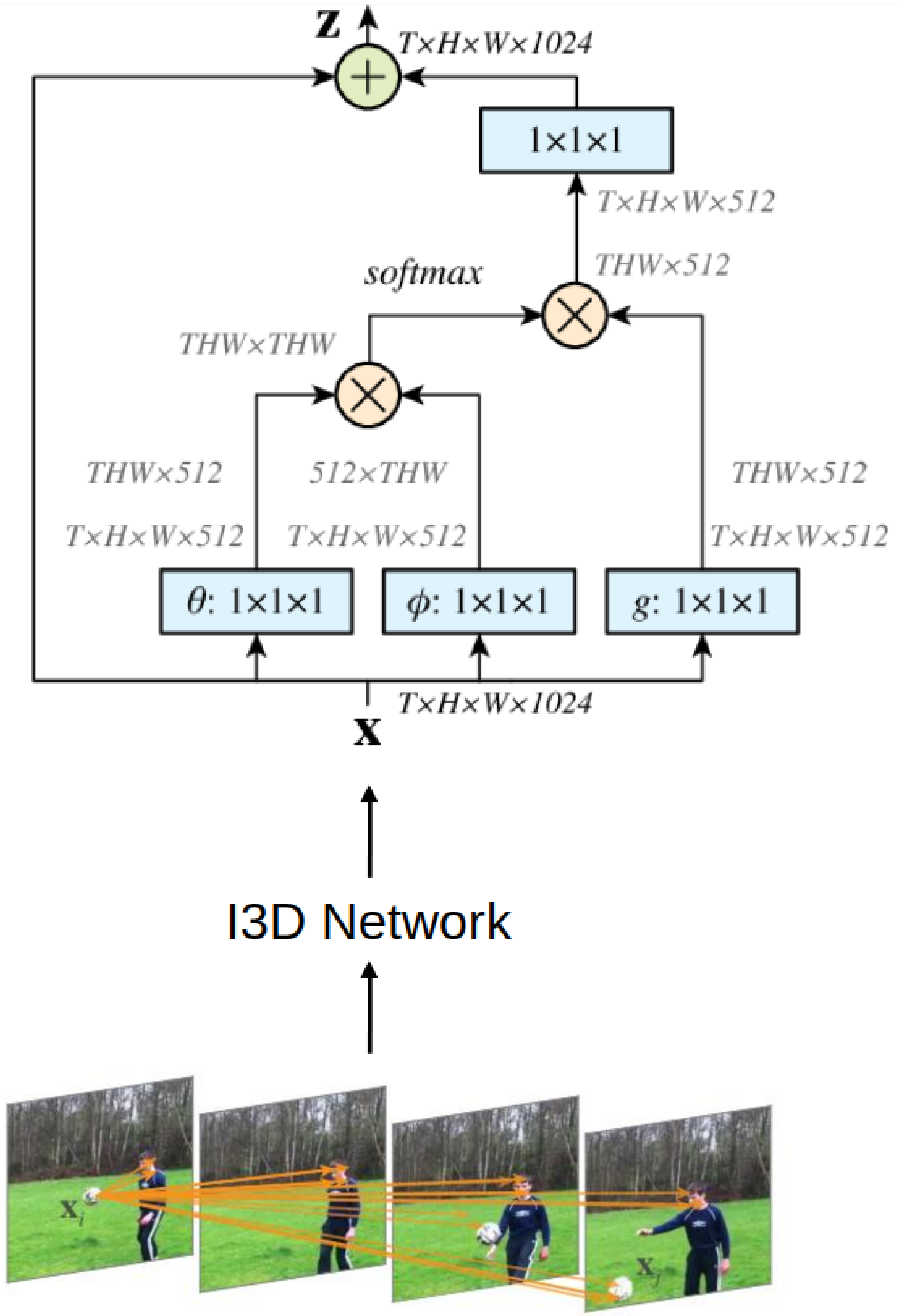
- **Goal:** To capture dependencies and relationships within input sequences.
- Each element attends to every other element. (or) Computes the correlation among the feature vectors in as sequence.
- **How it Works:**
  - It transforms the input sequence into three vectors: **query, key, and value.** These vectors are obtained through **linear transformations of the input.**
  - Second, the attention mechanism **calculates a weighted sum of the values** based on the **similarity between the query and key vectors.**
  - The resulting weighted sum, along with the original input, is then **passed through a feed-forward neural network** to produce the final output.



Self Attention::

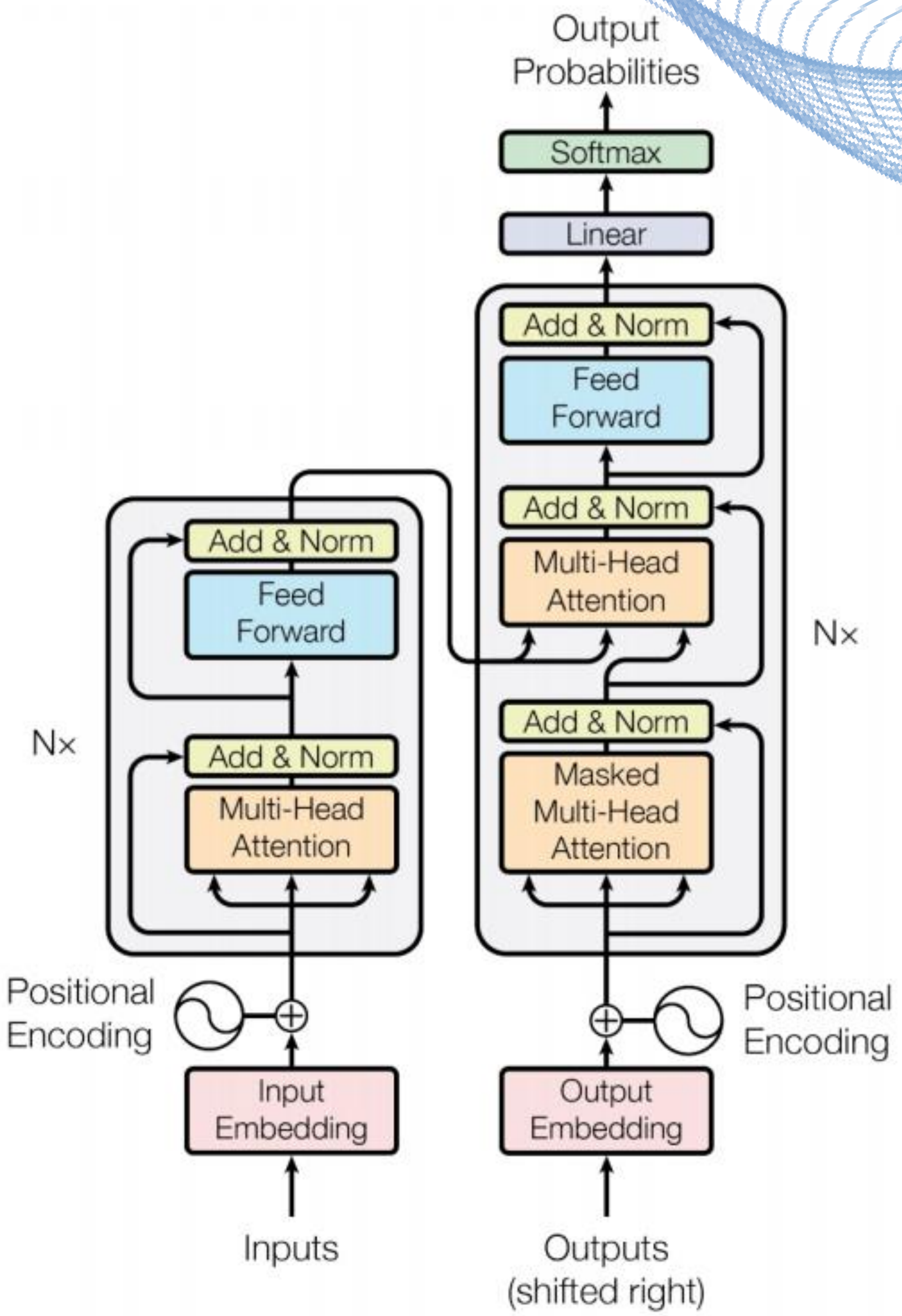
- **Benefits:**
    - **Long-range dependencies:** It allows the model to capture relationships between distant elements in a sequence, enabling it to understand complex patterns and dependencies.
    - **Contextual understanding:** By attending to different parts of the input sequence, self-attention helps the model understand the context and assign appropriate weights to each element based on its relevance.
    - **Parallel computation:** Itcan be computed in parallel for each element in the sequence, making it computationally efficient and scalable for large datasets.

Self Attention in Non-Local Network::



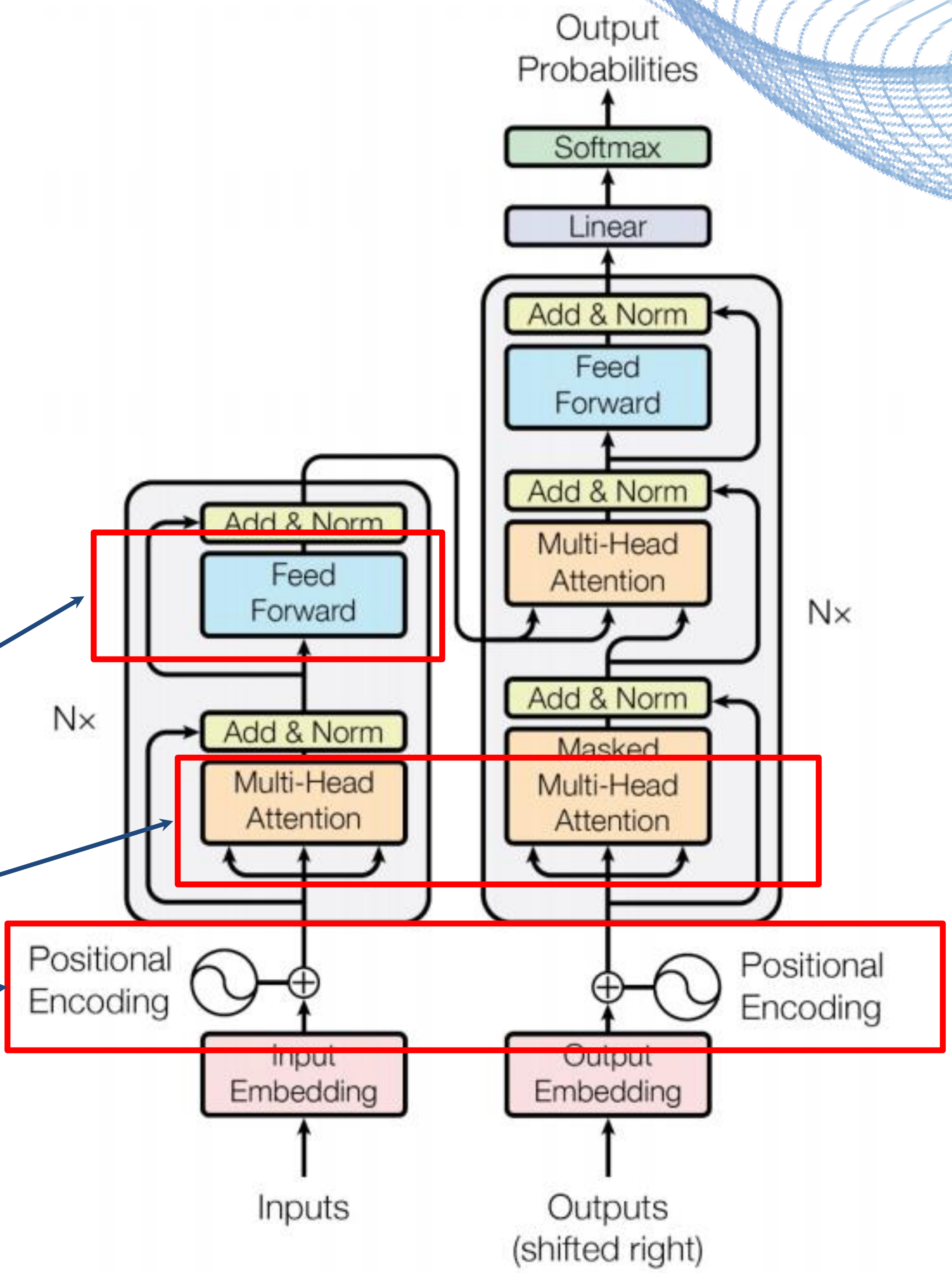
Transformer Models:

- **Transformer are standard architecture for sequence modeling in Natural Language Processing.**
- **A Pure Transformer:**
  - Performs excellent on standard computer vision tasks (like image classification) when applied directly to sequence of image patches or tokens.
  - Achieves State-of-the art results on benchmark problems and can learned representations are transferable to other problem domains
- **Key Components:**
  - Self-Attention or Multi-Head Attention
  - Position Embedding
  - Feed Forward



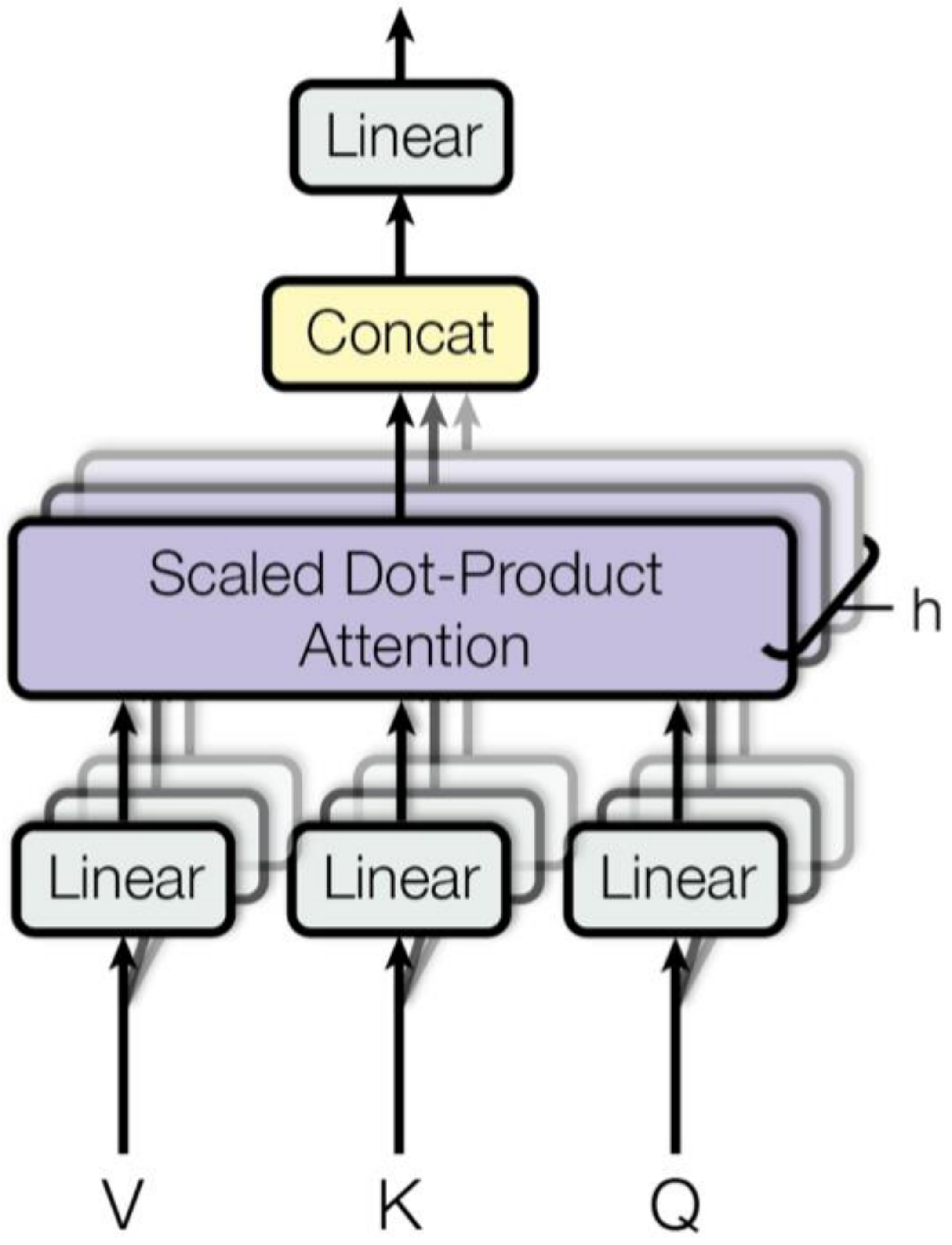
Transformer Models:

- **Transformer are standard architecture for sequence modeling in Natural Language Processing.**
- **A Pure Transformer:**
  - Performs excellent on standard computer vision tasks (like image classification) when applied directly to sequence of image patches or tokens.
  - Achieves State-of-the art results on benchmark problems and can learned representations are transferable to other problem domains
- **Key Components:**
  - Self-Attention or Multi-Head Attention
  - Position Embedding
  - Feed Forward

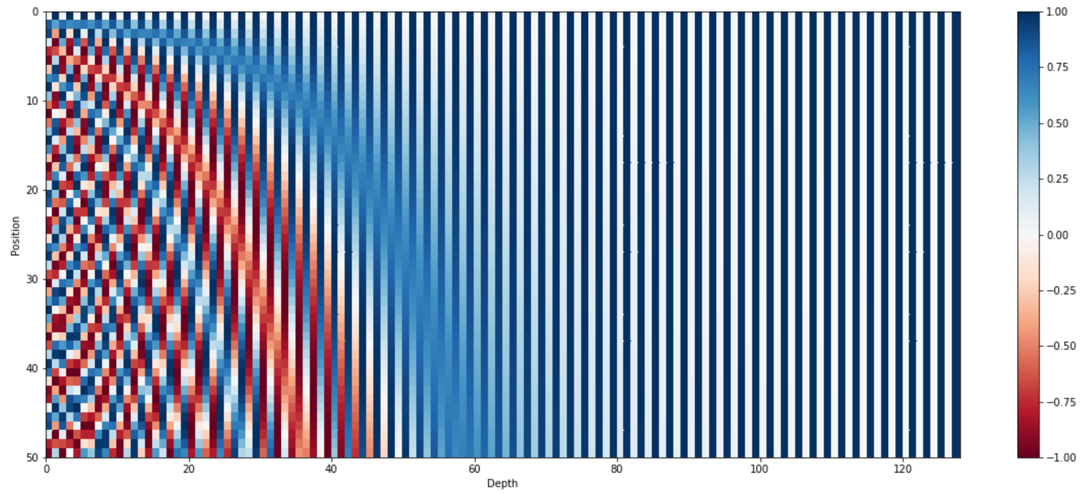


Transformer Models:

Self-Attention or Multi-Head Attention



Position Embedding



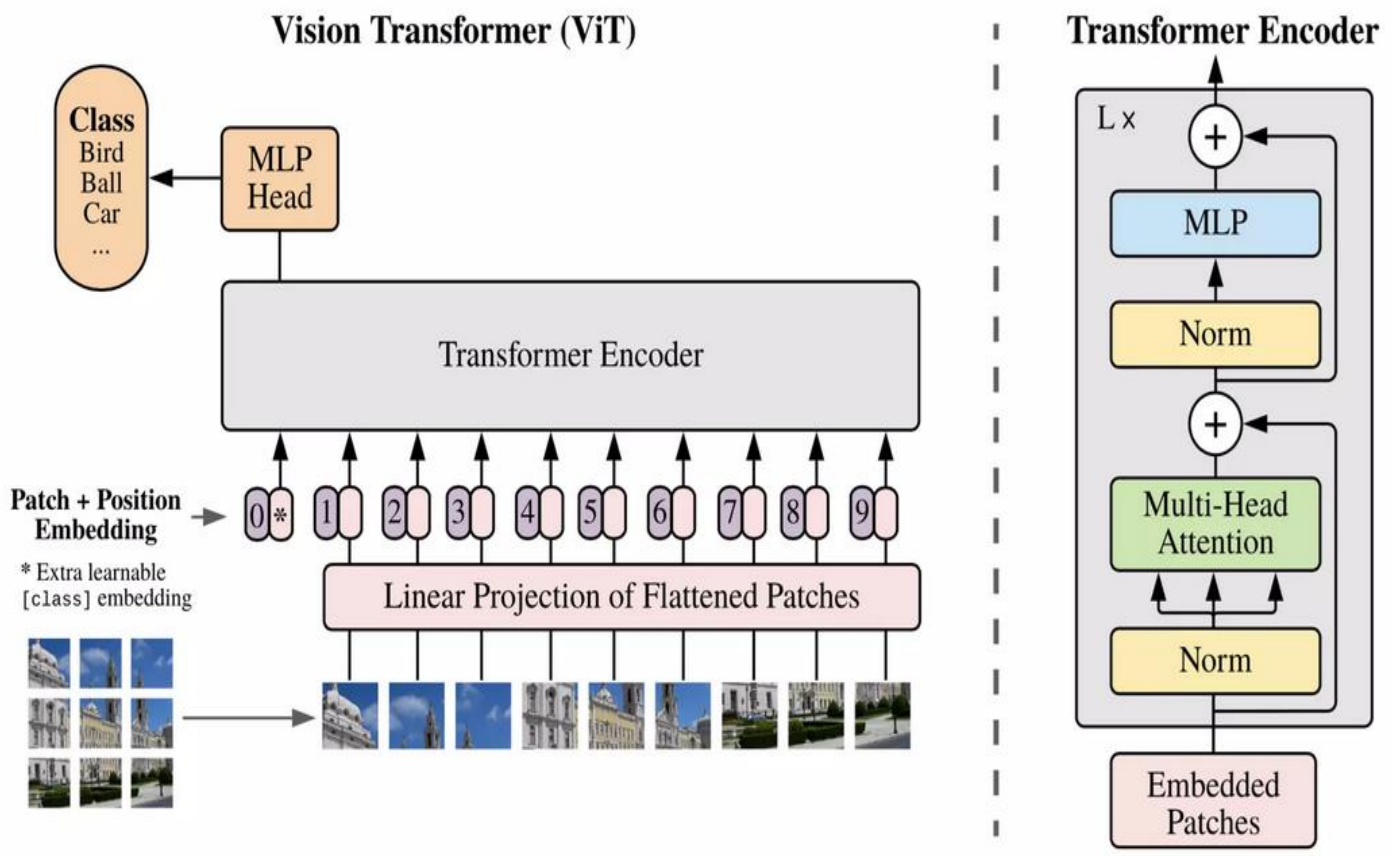
$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

$$\vec{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} \sin(\omega_k . t), & \text{if } i = 2k \\ \cos(\omega_k . t), & \text{if } i = 2k + 1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

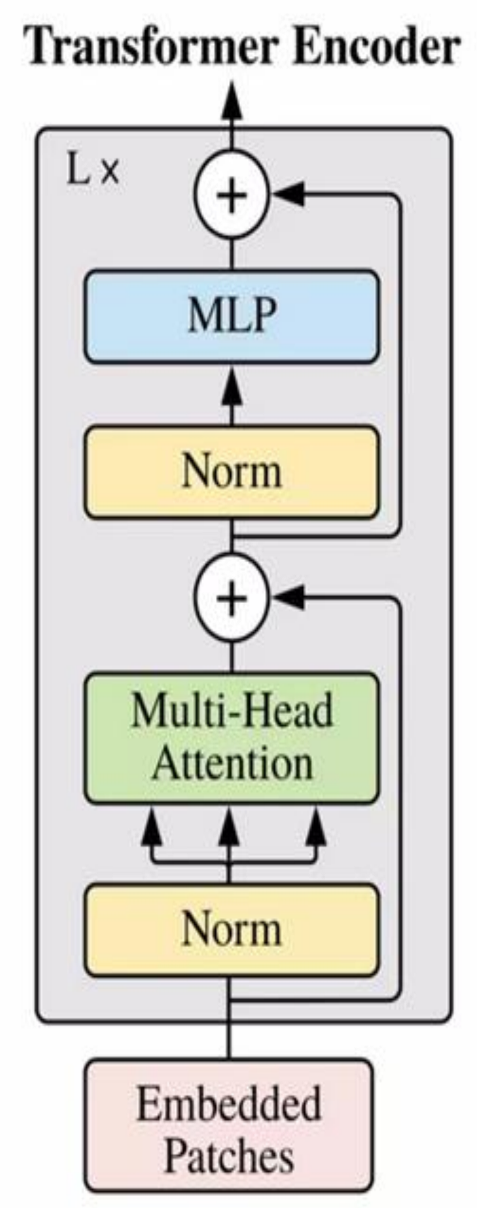
Vision Transformer (ViT):

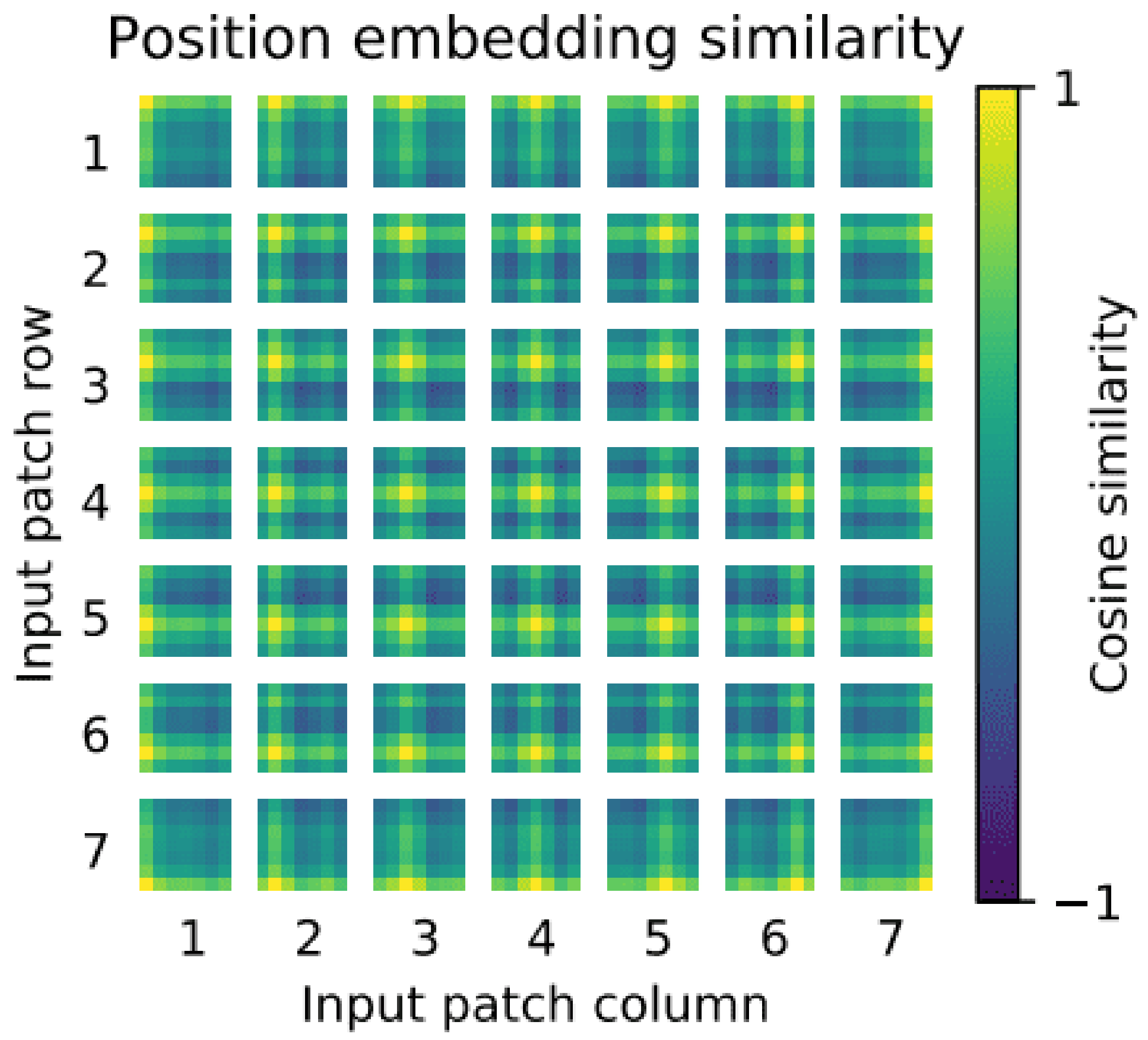
- In ViTs, images are represented as sequences, and class labels for the image are predicted, which allows models to learn image structure independently.
- **How ViT works?**
  - Split an image into patches (Tokenize)
  - Flatten the patches
  - Produce lower-dimensional linear embeddings from the flattened patches
  - Add positional embeddings
  - Feed the sequence as an input to a standard transformer encoder (for interaction among tokens)
  - Pretrain the model with image labels (fully supervised on a huge dataset)
  - Finetune on the downstream dataset for image classification



Vision Transformer (ViT):

- Multiple blocks in the ViT encoder, and each block consists of three major processing elements:
  - Layer Norm: It keeps the training process on track and lets the model adapt to the variations among the training images.
  - Multi-Head Attention Network: Generating attention maps from the given embedded visual tokens. These attention maps help the network focus on the most critical regions in the image, such as object(s).
  - Multi-Layer Perceptrons (MLP): MLP is a two-layer classification network with GELU (Gaussian Error Linear Unit) at the end. The final MLP block also called the MLP head, is used as an output of the transformer.





ViT vs. CNN:

- ViT has more similarity between the representations obtained in shallow and deep layers compared to CNNs.

- Unlike CNNs, ViT obtains the global representation from the shallow layers, but the local representation obtained from the shallow layers is also important.

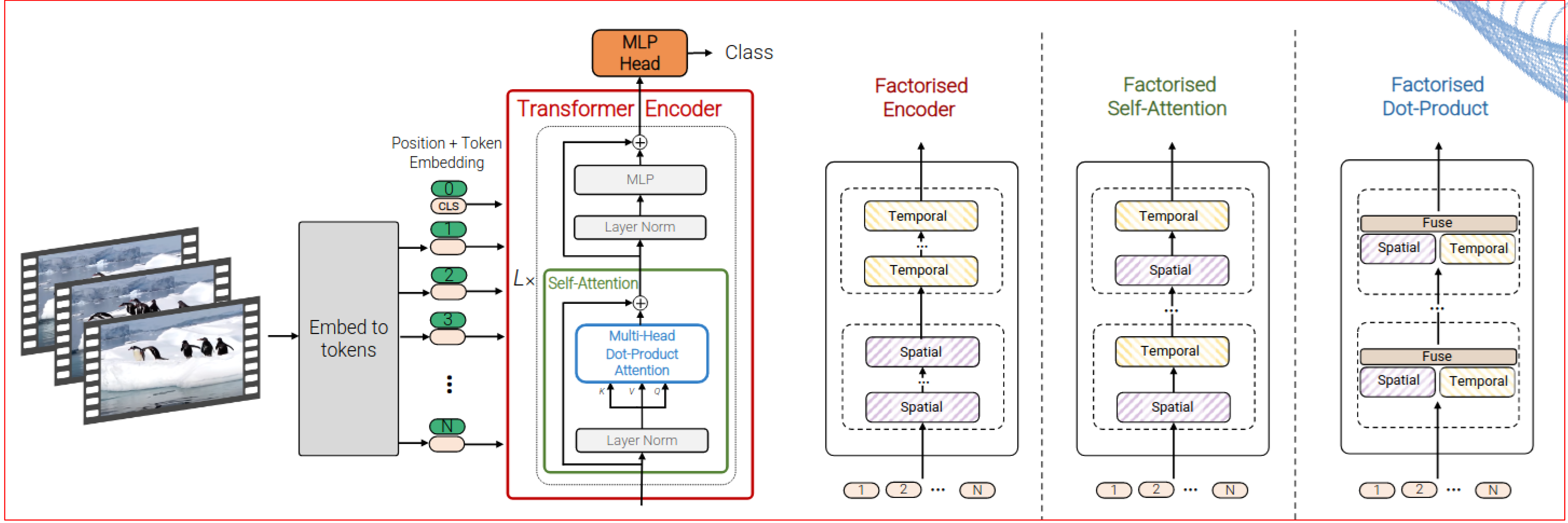
- Skip connections in ViT are even more influential than in CNNs (ResNet) and substantially impact the performance and similarity of representations.

- ViT retains more spatial information than CNN.

- ViT can learn high-quality intermediate representations with large amounts of data.

- ViT is more Scalable and Efficient compared to CNN

A Video Vision Transformer (ViViT):



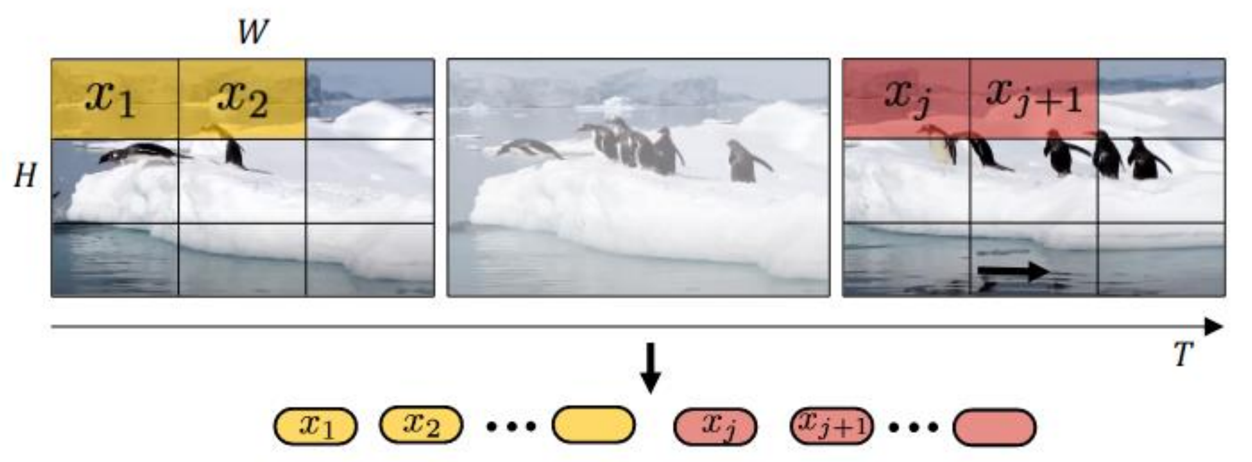


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [18].

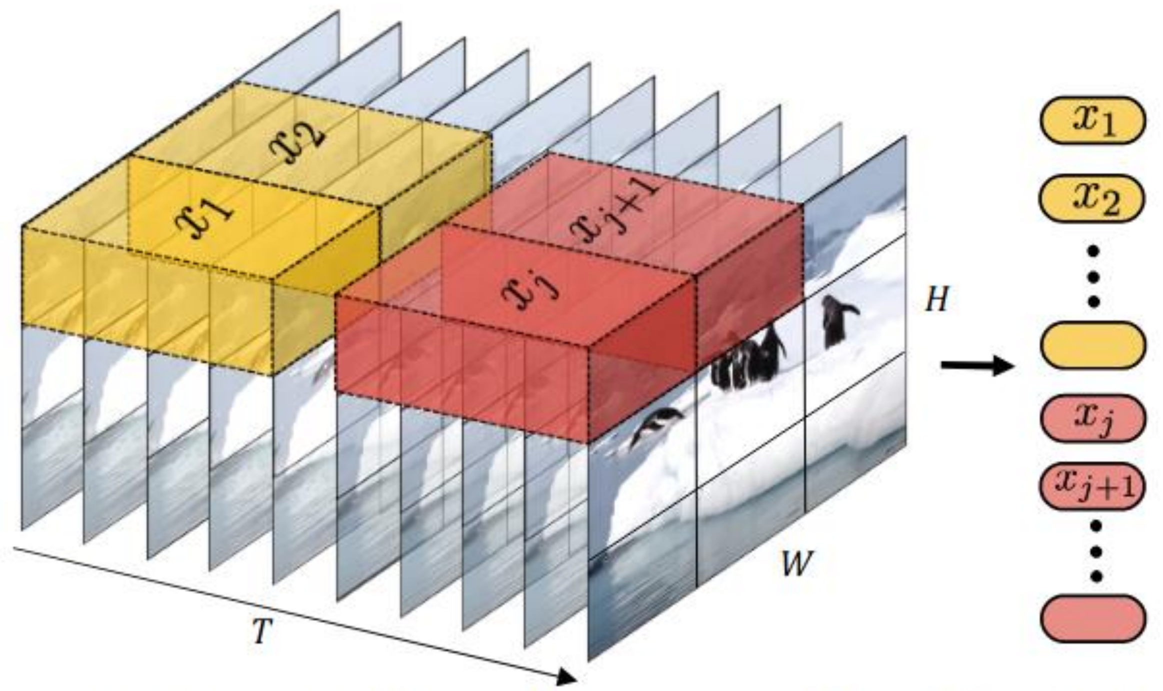
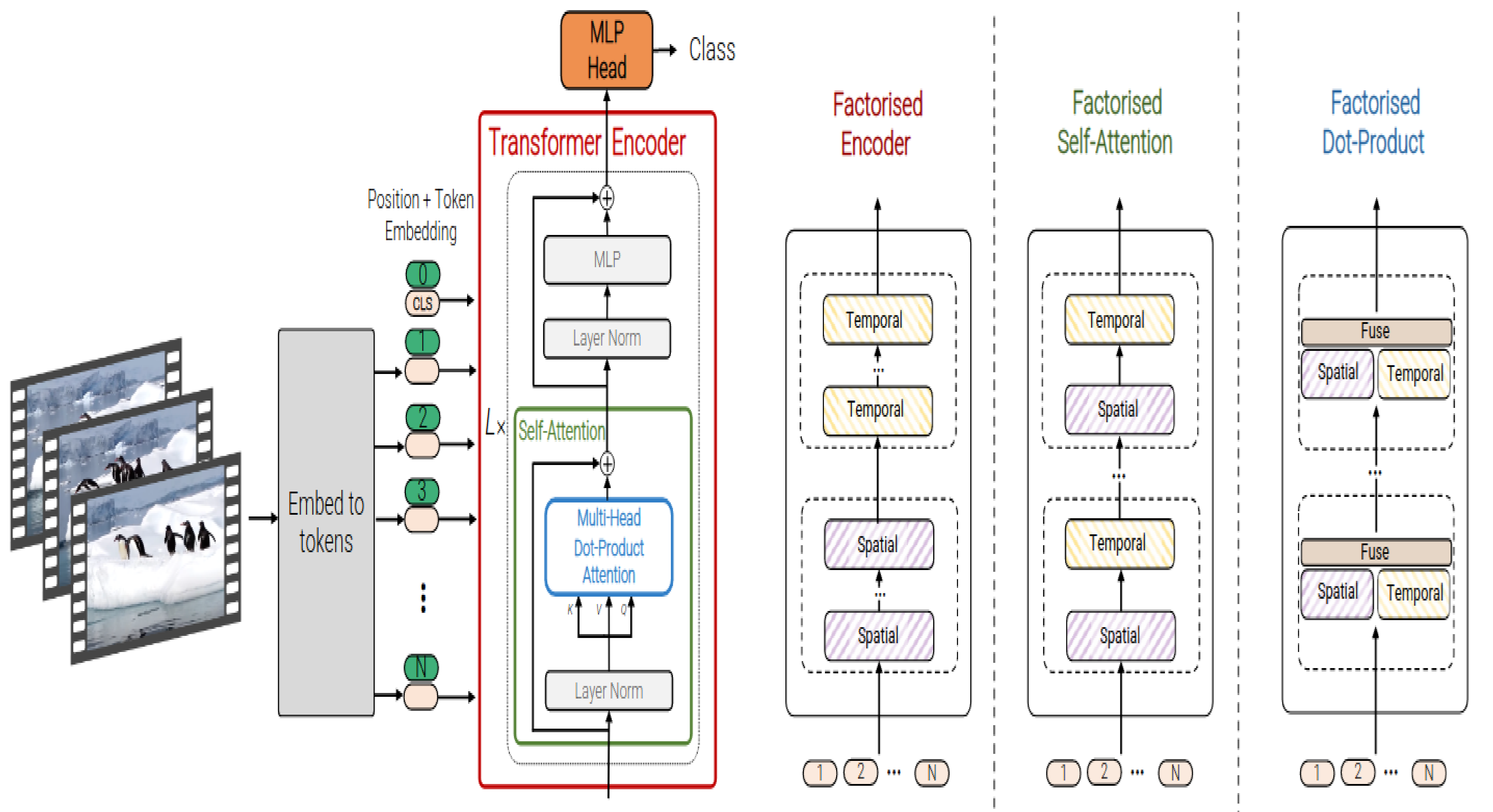


Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

A Video Vision Transformer (ViViT):



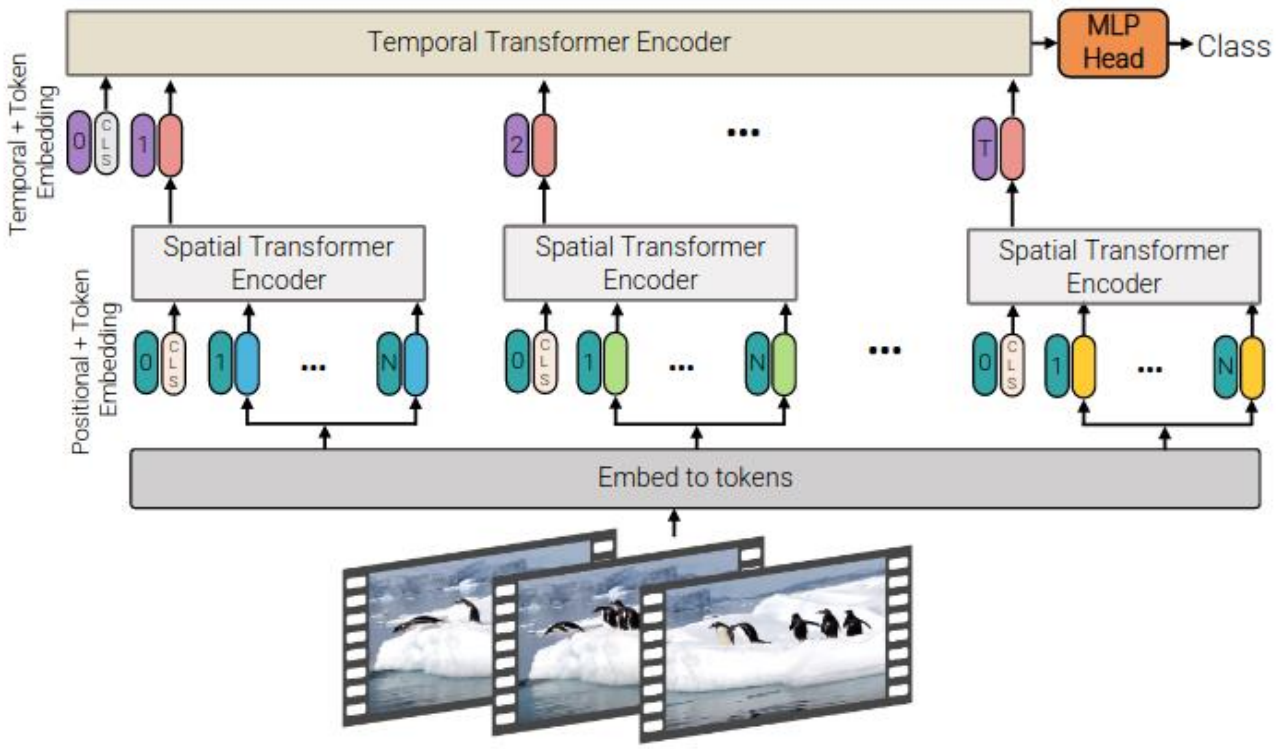


Figure 4: Factorised encoder (Model 2). This model consists of two transformer encoders in series: the first models interactions between tokens extracted from the same temporal index to produce a latent representation per time-index. The second transformer models interactions between time steps. It thus corresponds to a "late fusion" of spatial- and temporal information.

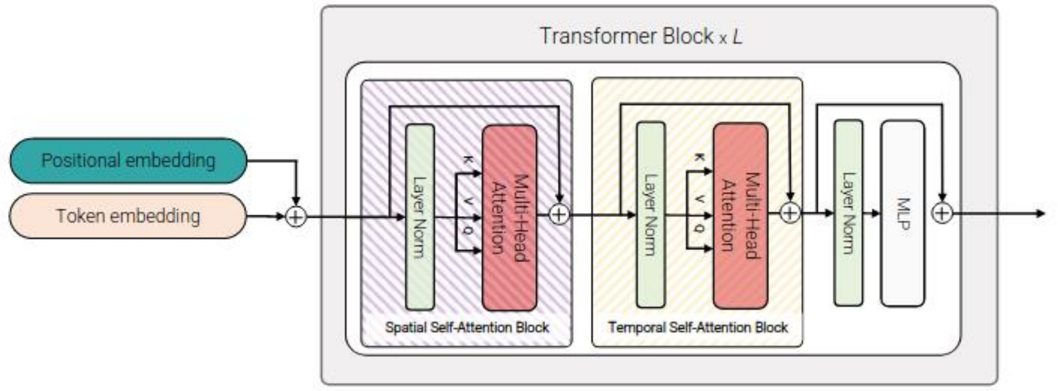
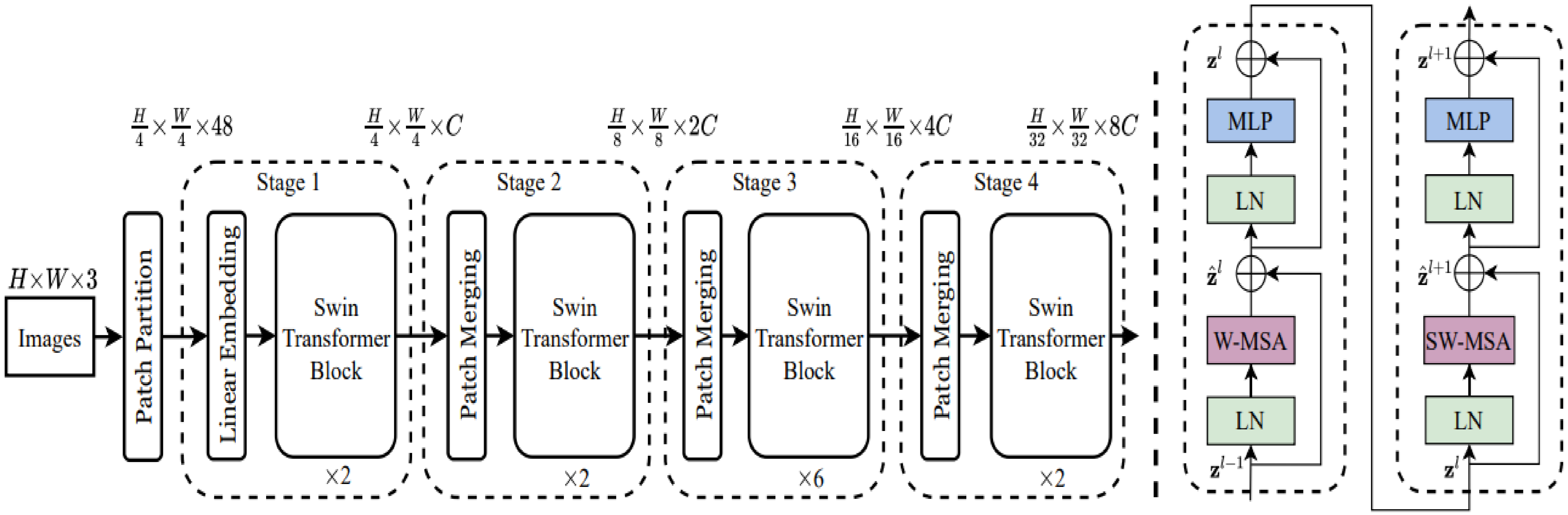


Figure 5: Factorised self-attention (Model 3). Within each transformer block, the multi-headed self-attention operation is factorised into two operations (indicated by striped boxes) that first only compute self-attention spatially, and then temporally.

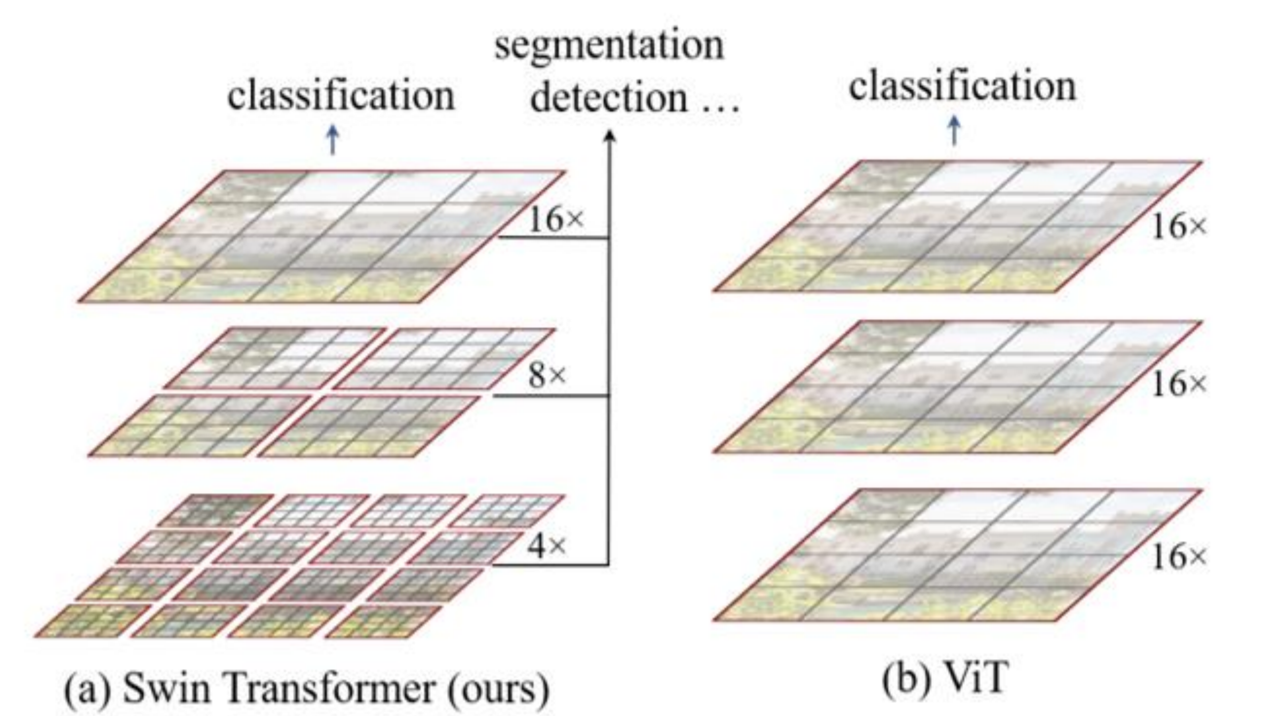
Swin Transformer :

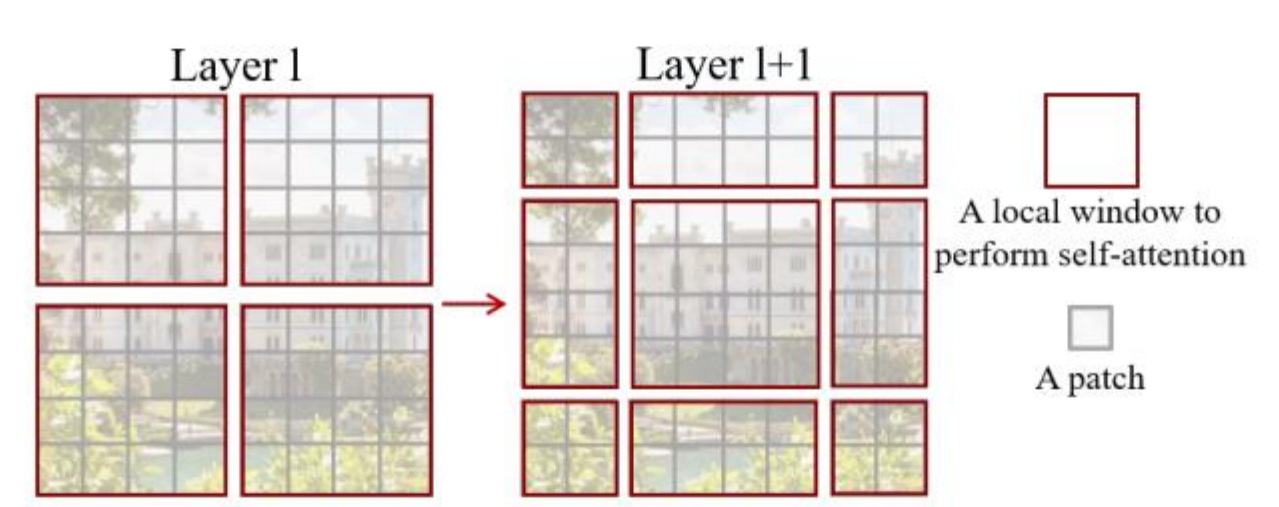
- Swin Transformer **builds hierarchical feature maps by merging image patches** in deeper layers compared to **ViTs that produces feature maps of a single low resolution**.
- It is enabled by **shifted window** to build hierarchical feature maps



(a) Architecture (b) Two Successive Swin Transformer Blocks

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.





Shifted Window

Video Swin Transformer :

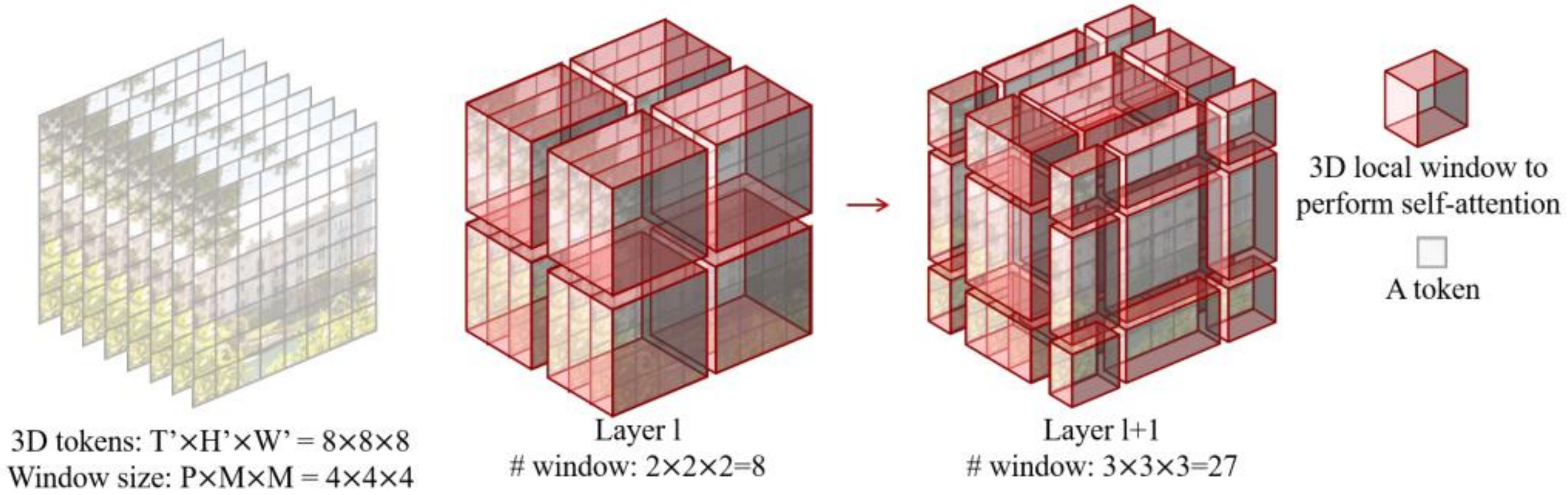


Figure 3: An illustrated example of 3D shifted windows. The input size $T' \times H' \times W'$ is 8×8×8, and the 3D window size $P \times M \times M$ is 4×4×4. As layer l adopts regular window partitioning, the number of windows in layer l is 2×2×2=8. For layer l+1, as the windows are shifted by $(\frac{P}{2}, \frac{M}{2}, \frac{M}{2})$=(2, 2, 2) tokens, the number of windows becomes 3×3×3=27. Though the number of windows is increased, the efficient batch computation in [28] for the shifted configuration can be followed, such that the final number of windows for computation is still 8.

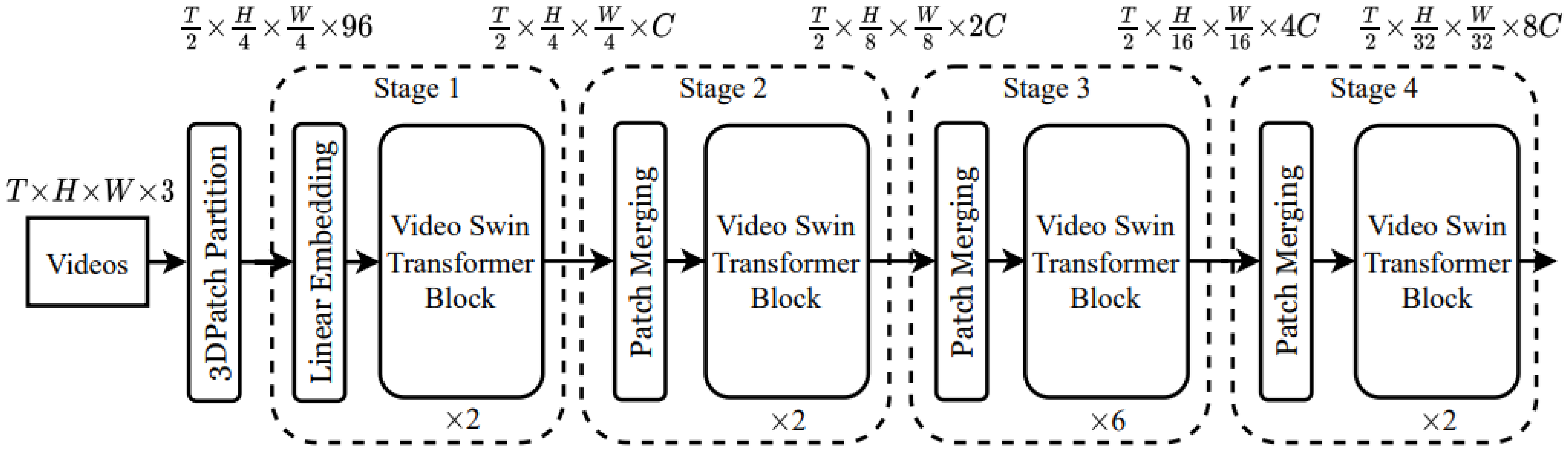


Figure 1: Overall architecture of Video Swin Transformer (tiny version, referred to as Swin-T).

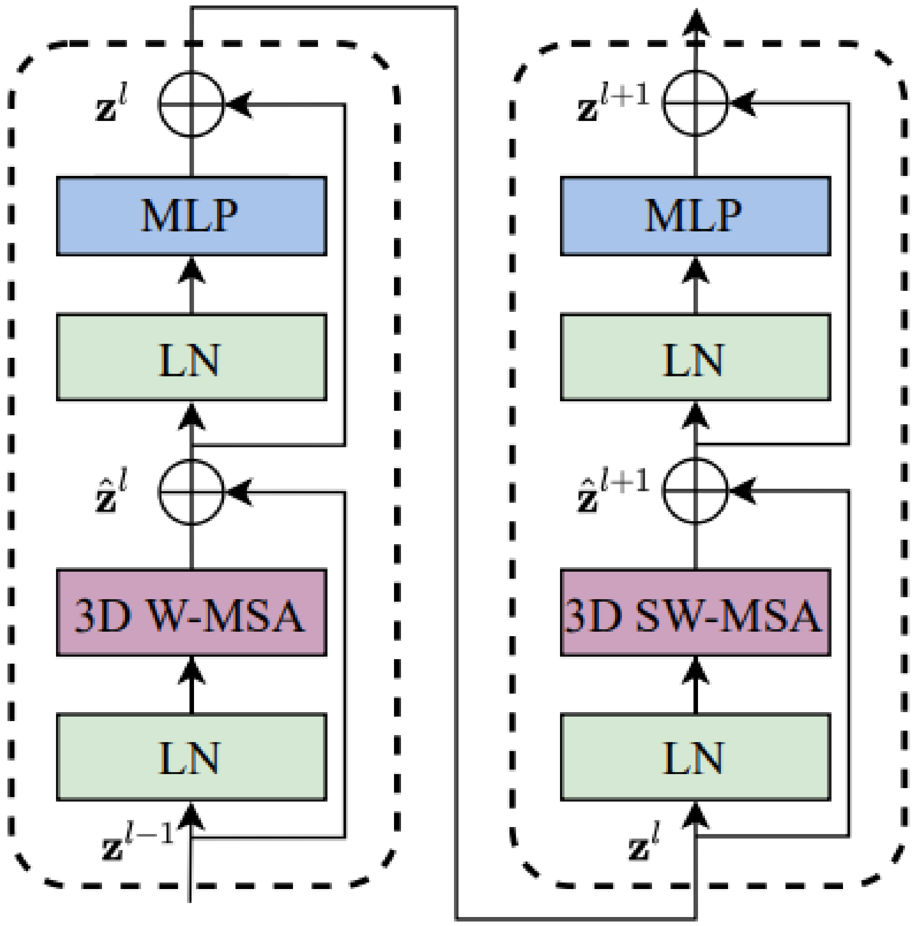


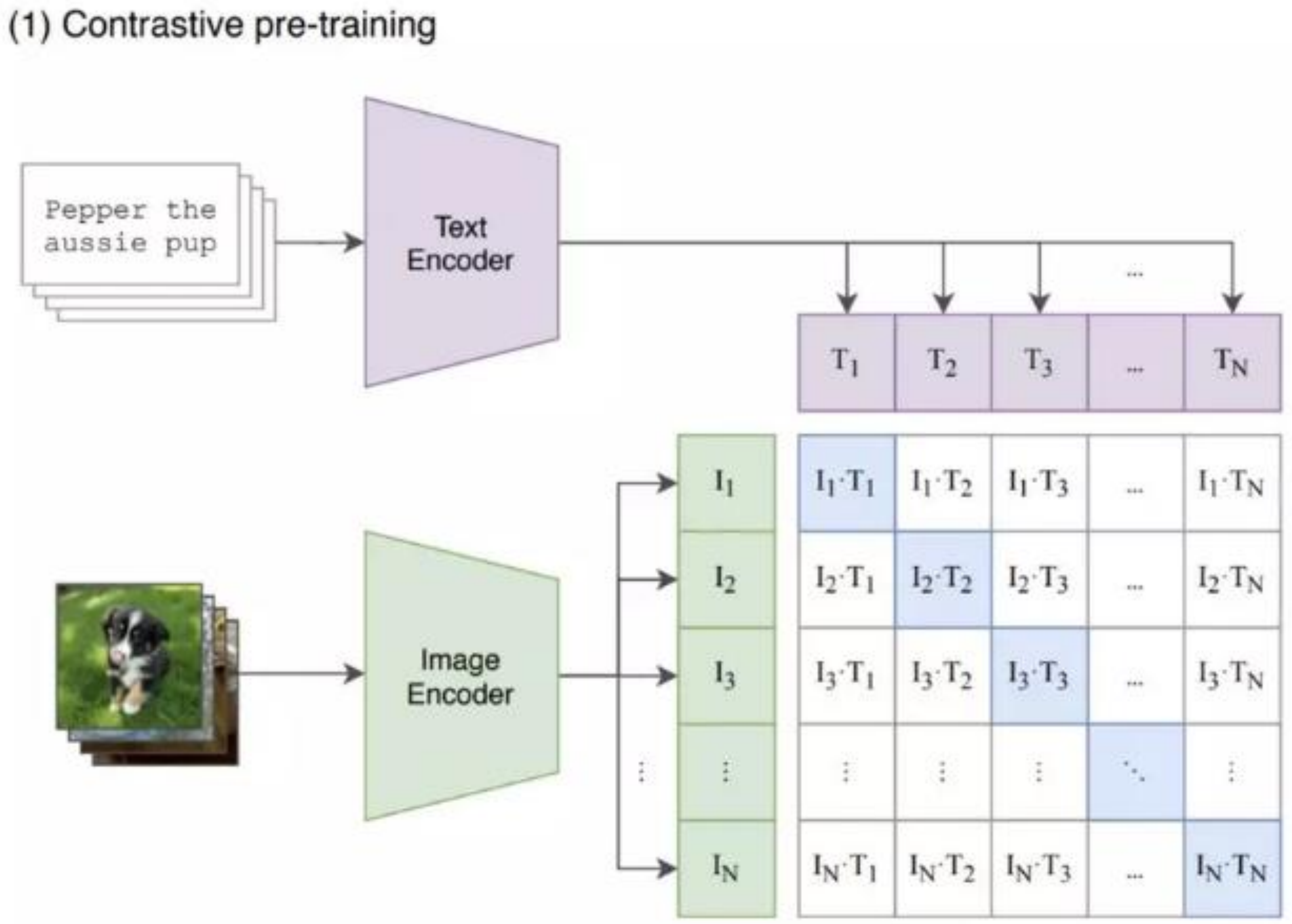
Figure 2: An illustration of two successive Video Swin Transformer blocks.

CLIP:: Contrastive Language-Image Pre-training

Background of Image-Text Pair

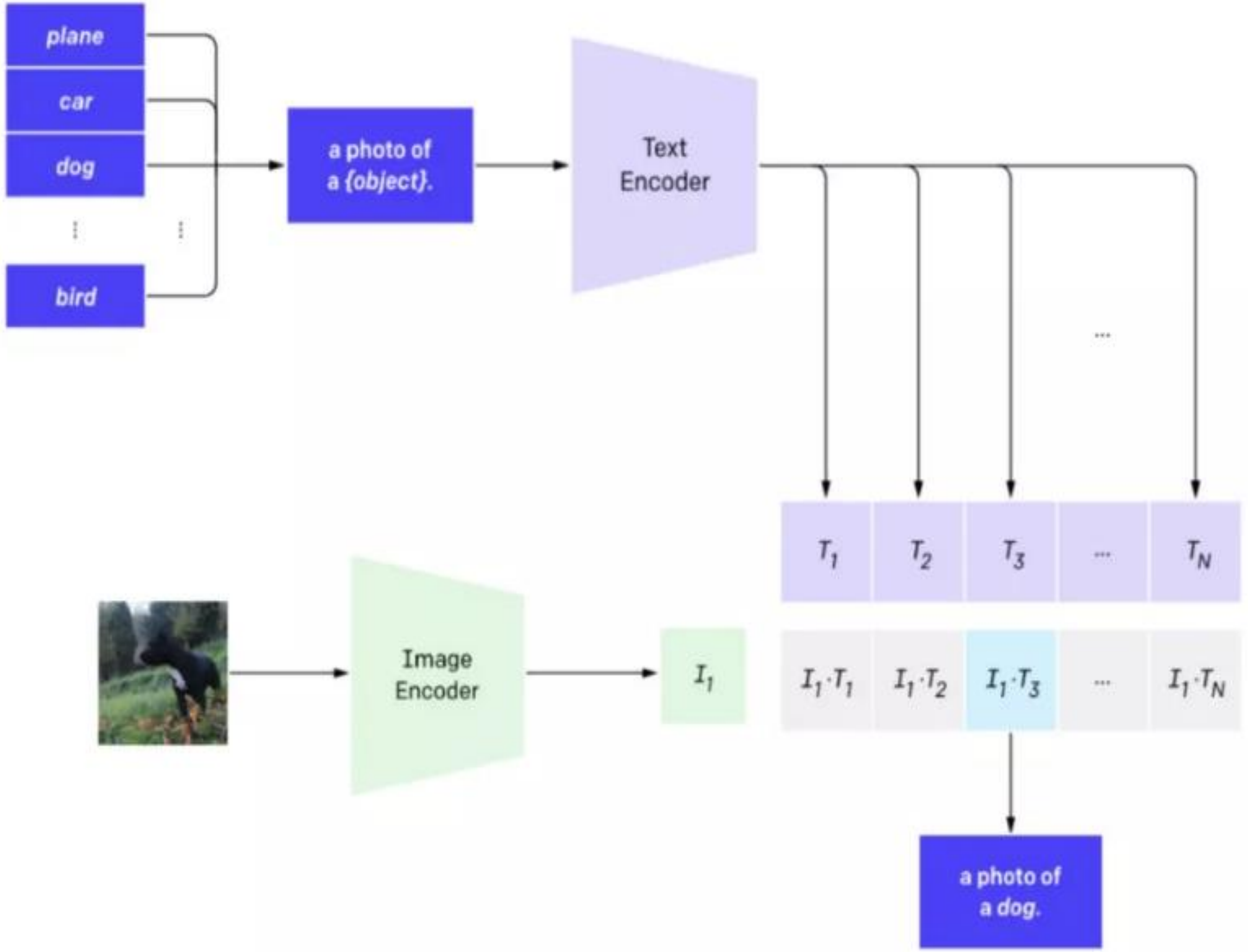
- N: Number of visual inputs for a single example
- T: Number of video frames
- H, W, C: height, width, color channels

CLIP:: Contrastive Language-Image Pre-training



- **400** million (image, text) pairs collected from Internet.
- Trained modifications of **ResNet-50** and **ViT-B**
- Batch size **32 768** for **32** epochs
- **The largest ResNet model, RN50x64, took 18 days to train on 592 V100 GPUs while the largest Vision Transformer took 12 days on 256 V100 GPUs**

CLIP for Zero-shot Classification



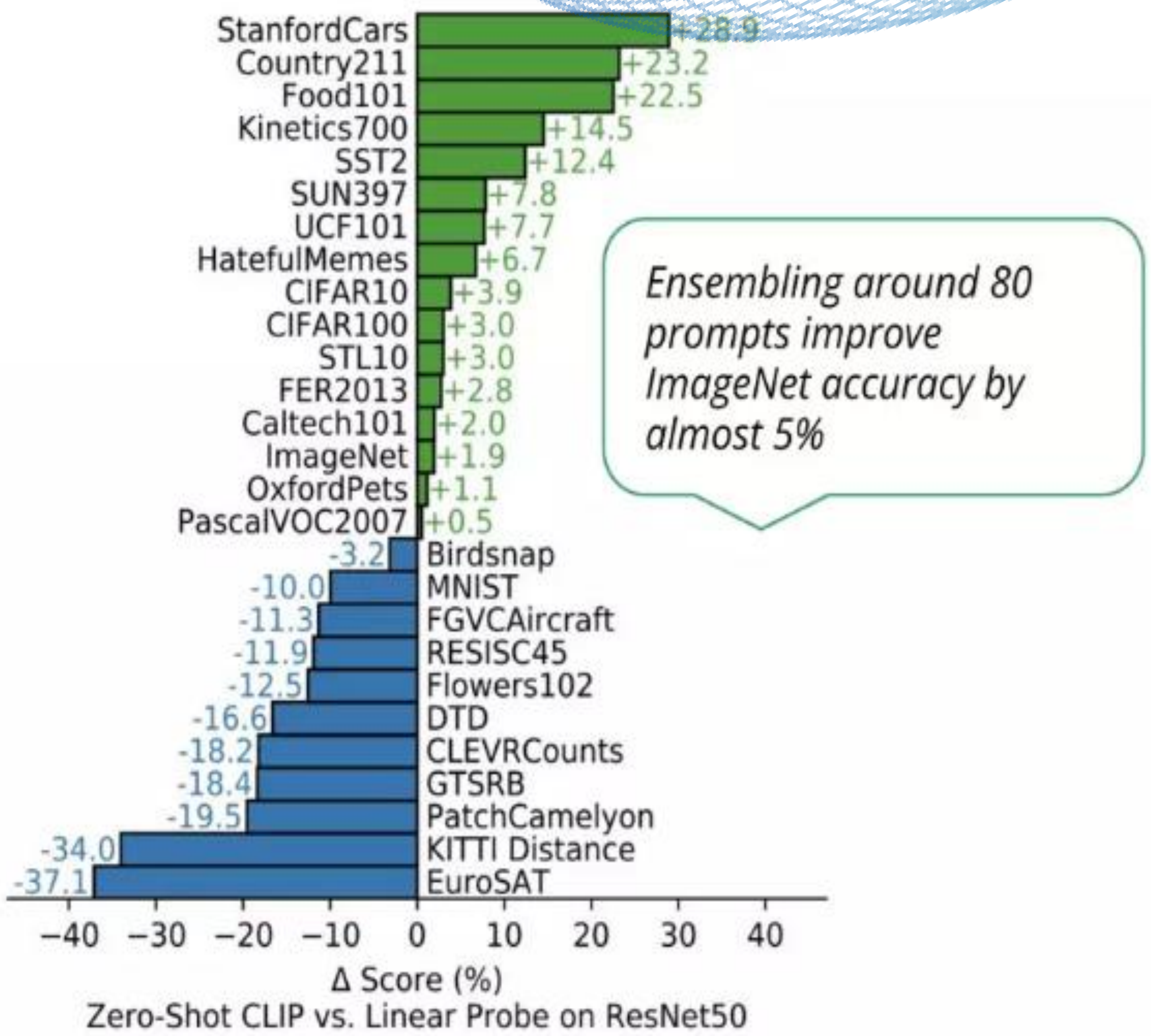


Figure 5. **Zero-shot CLIP is competitive with a fully supervised baseline.** Across a 27 dataset eval suite, a zero-shot CLIP classifier outperforms a fully supervised linear classifier fitted on ResNet-50 features on 16 datasets, including ImageNet.

CLIP Limitations ::

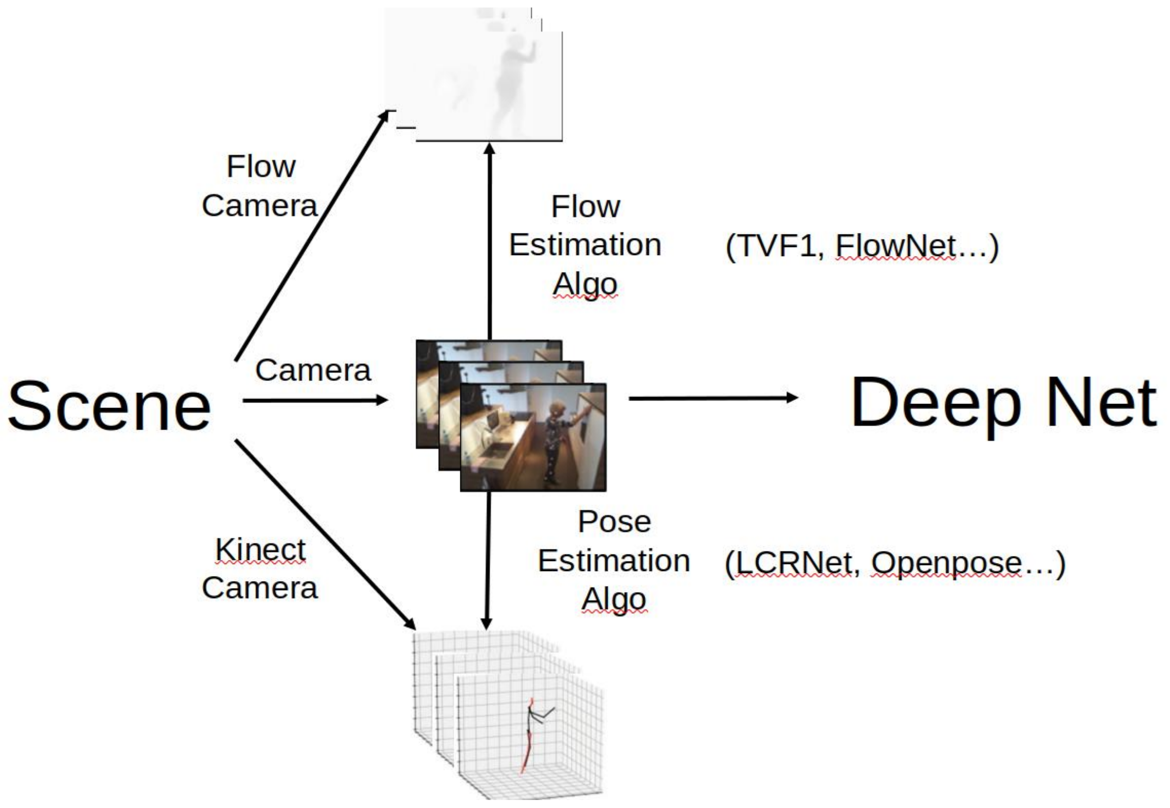
- poor generalization to images not covered in its pre-training dataset (MNIST)
- counting the number of objects in an image
- predicting how close the nearest object is in a photo
- CLIP's zero-shot classifiers can be sensitive to wording or phrasing and sometimes require trial and error "prompt engineering" to perform well.

Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

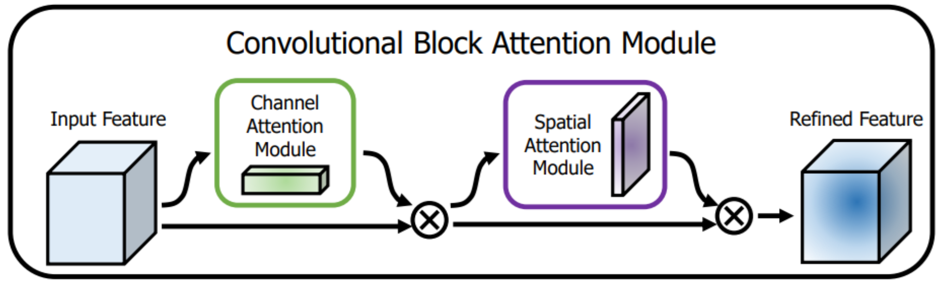
Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

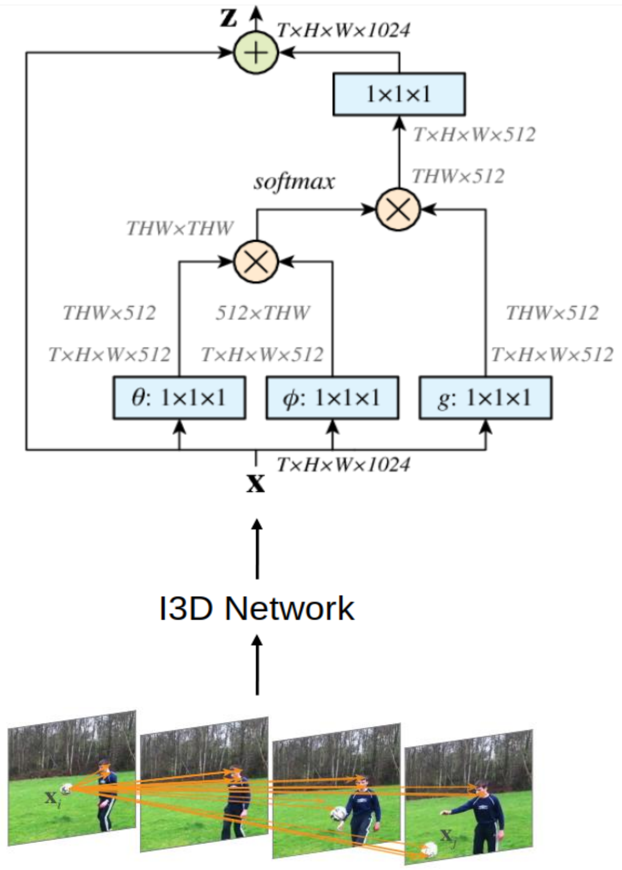
Summary::

Combining Multiple Modalities for HAR

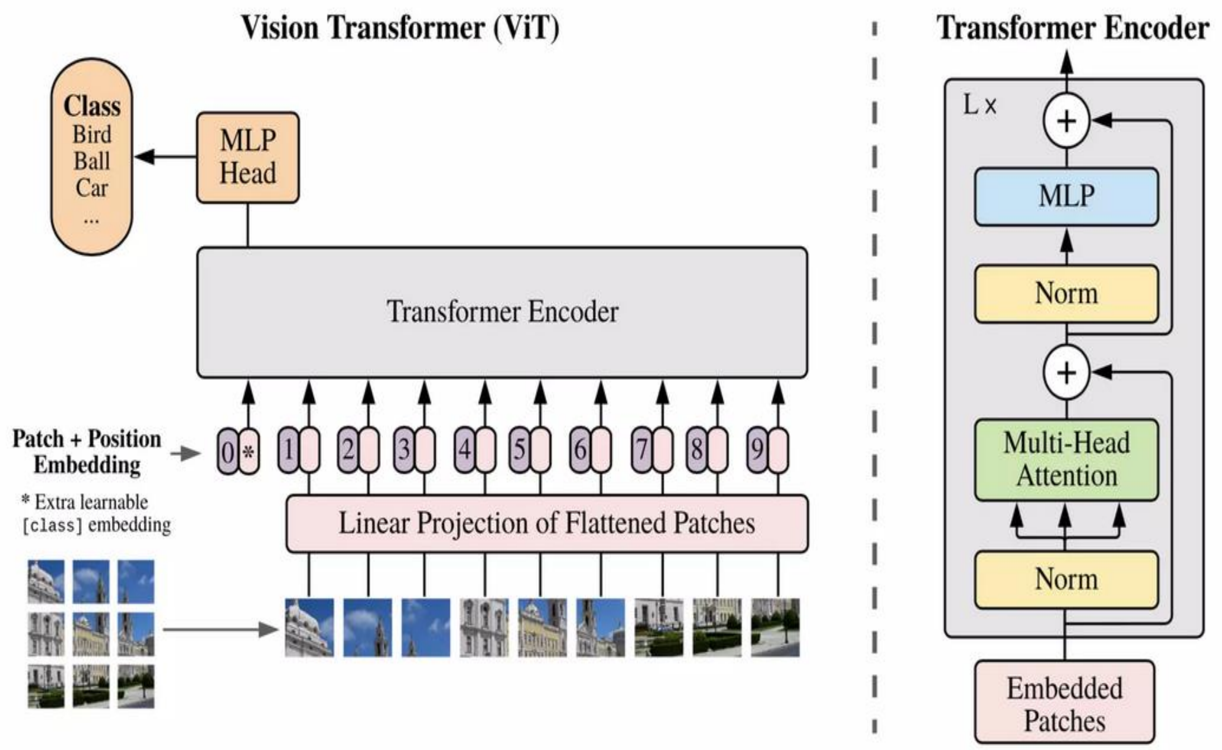


Attention Mechanism

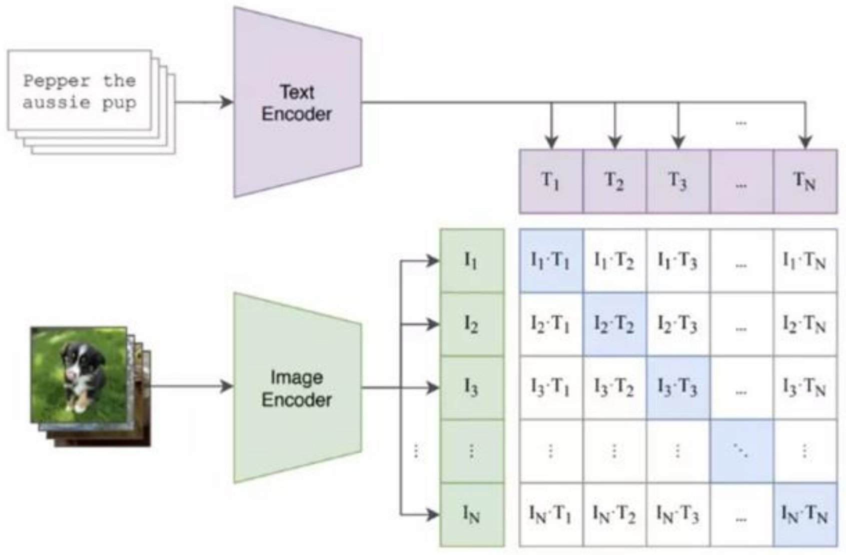




Transformer Models



CLIP:Vision-language



Thank you for your attention!