ACTION Recognition

Snehashis MAJHI

Email: snehashis.majhi@inria.fr

Ph.D. Candidate @STARS Team INRIA

Collaboration with TOYOTA Motor Europe

TABLE OF CONTENT

- 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)

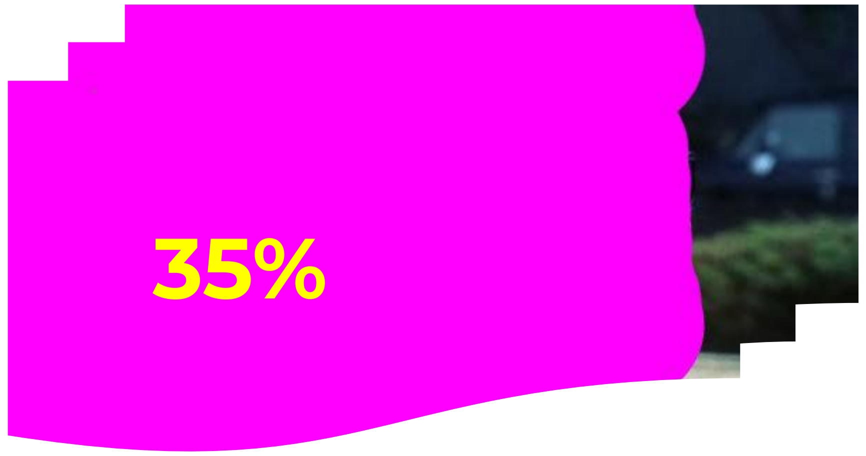
Why Human Action ?

- How many person-pixels are in the video?

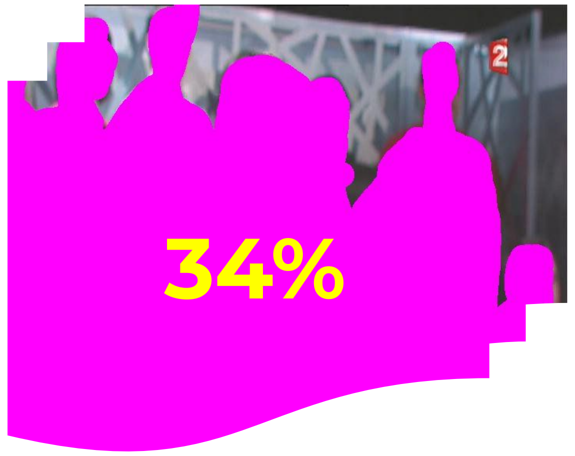
MOVIE

TV

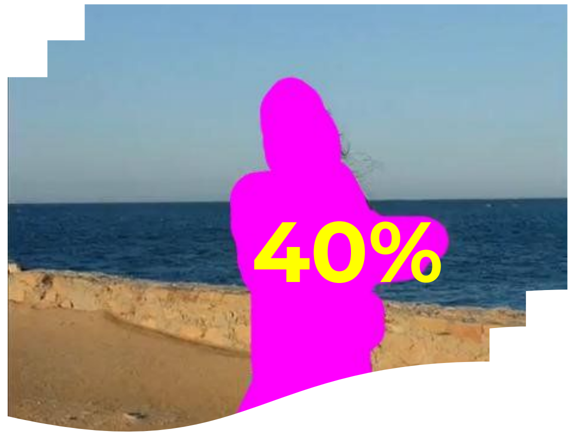
YouTube



MOVIE



TV



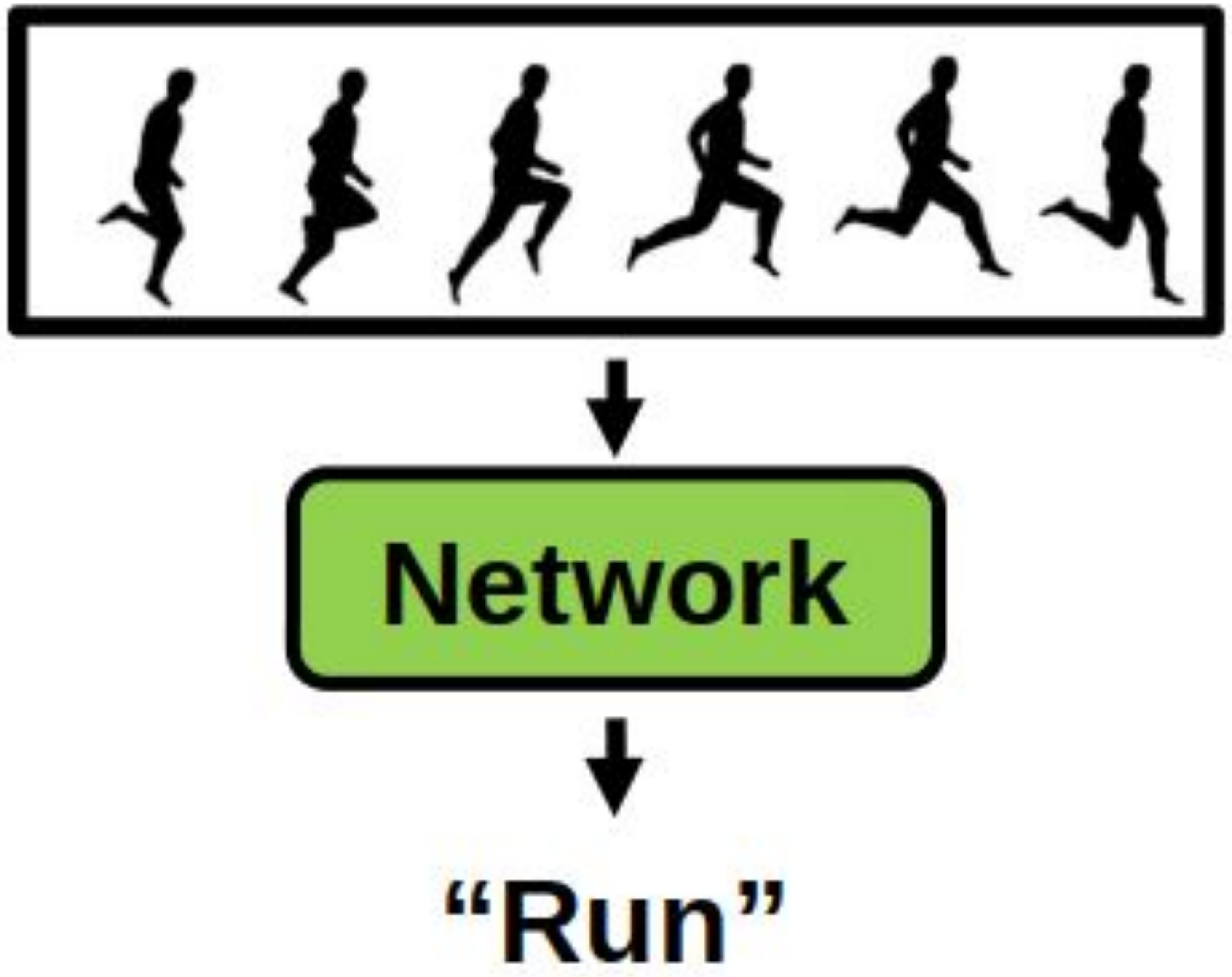
YouTube

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

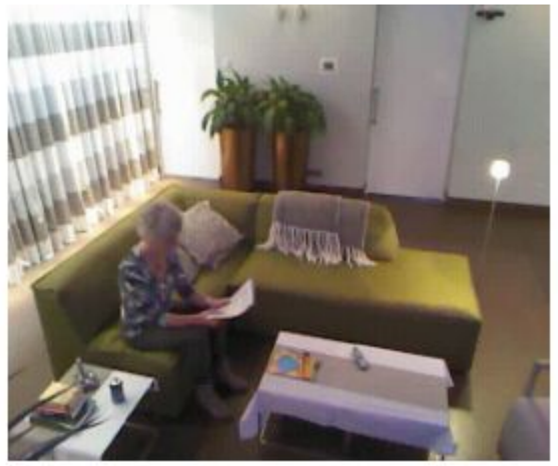


Typical Human Actions::

Daily Living

Drink

UseLaptop

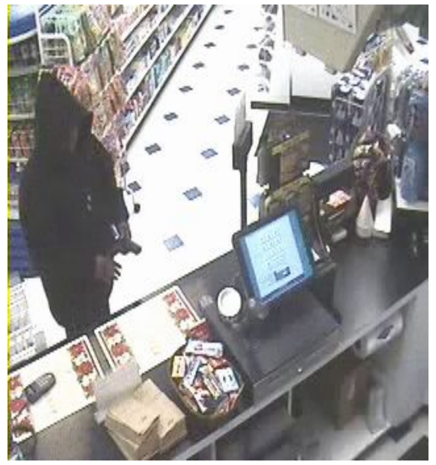


Read

Wild Abnormality

Burglary

Shoplift



Robbery

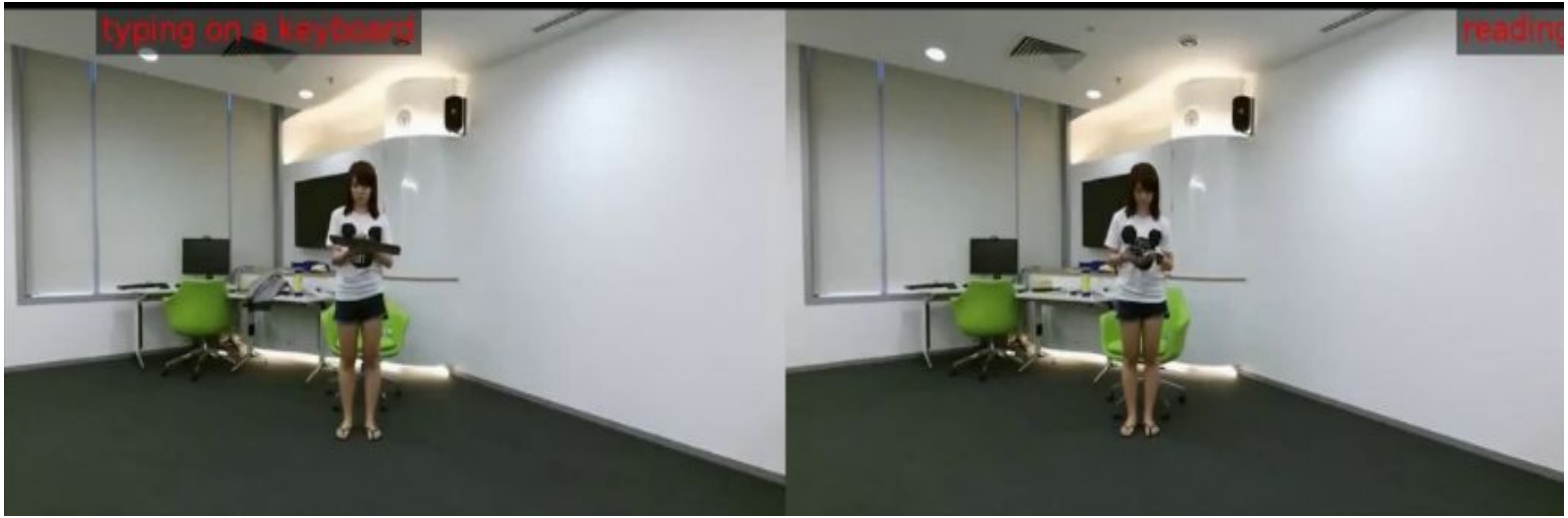
- **Key Challenges:**
  - Subtle Motion
  - High-**Intra**-class Variance
  - Low-**Inter**-class Variance

Challenges::

Subtle Motion:-

Typing Keyboard

Reading

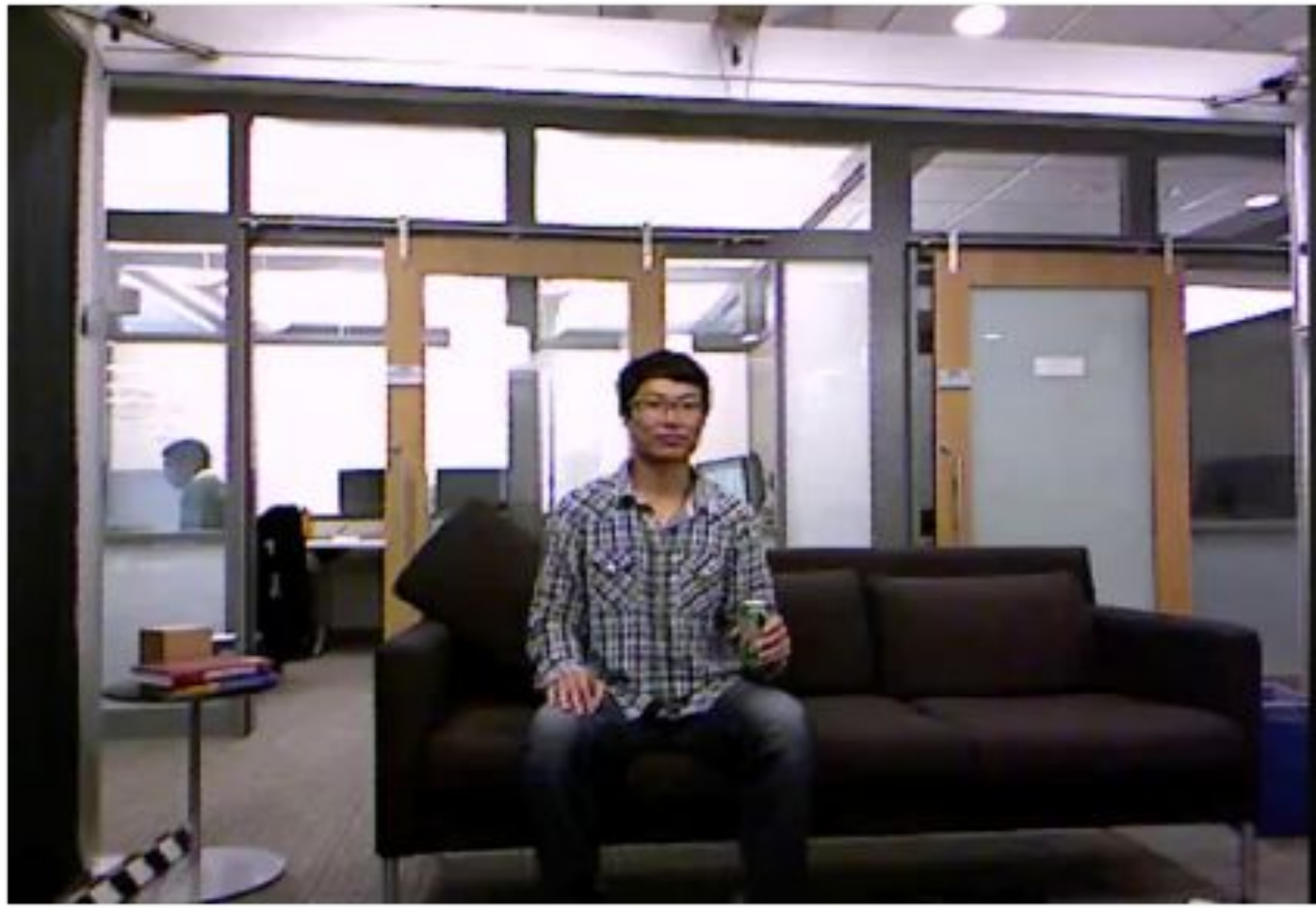


- Same Background
- Almost Similar Posture
- <u>**Different Actions**</u>

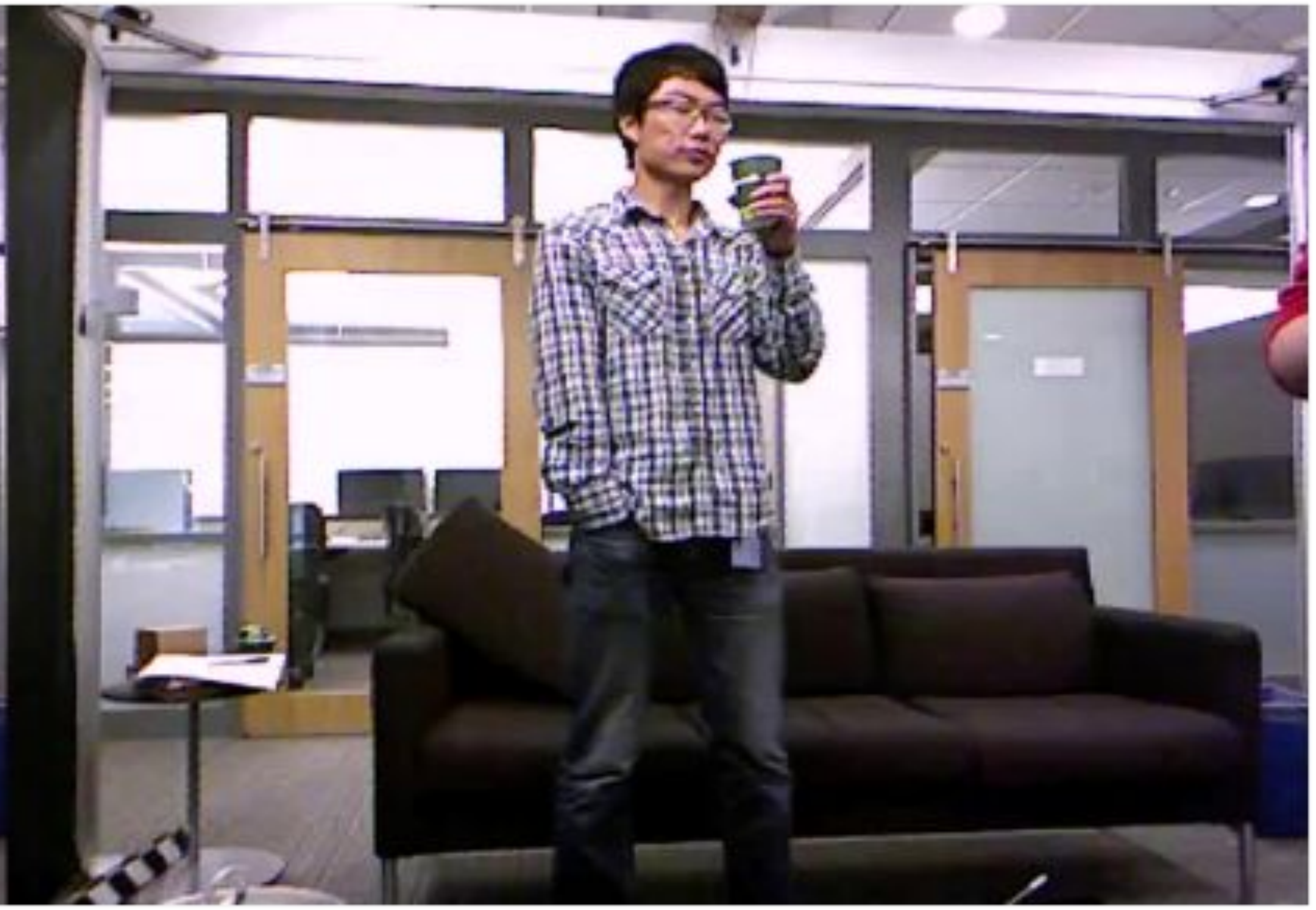
Challenges::

High-Intra-class Variance:-

Drinking



Drinking



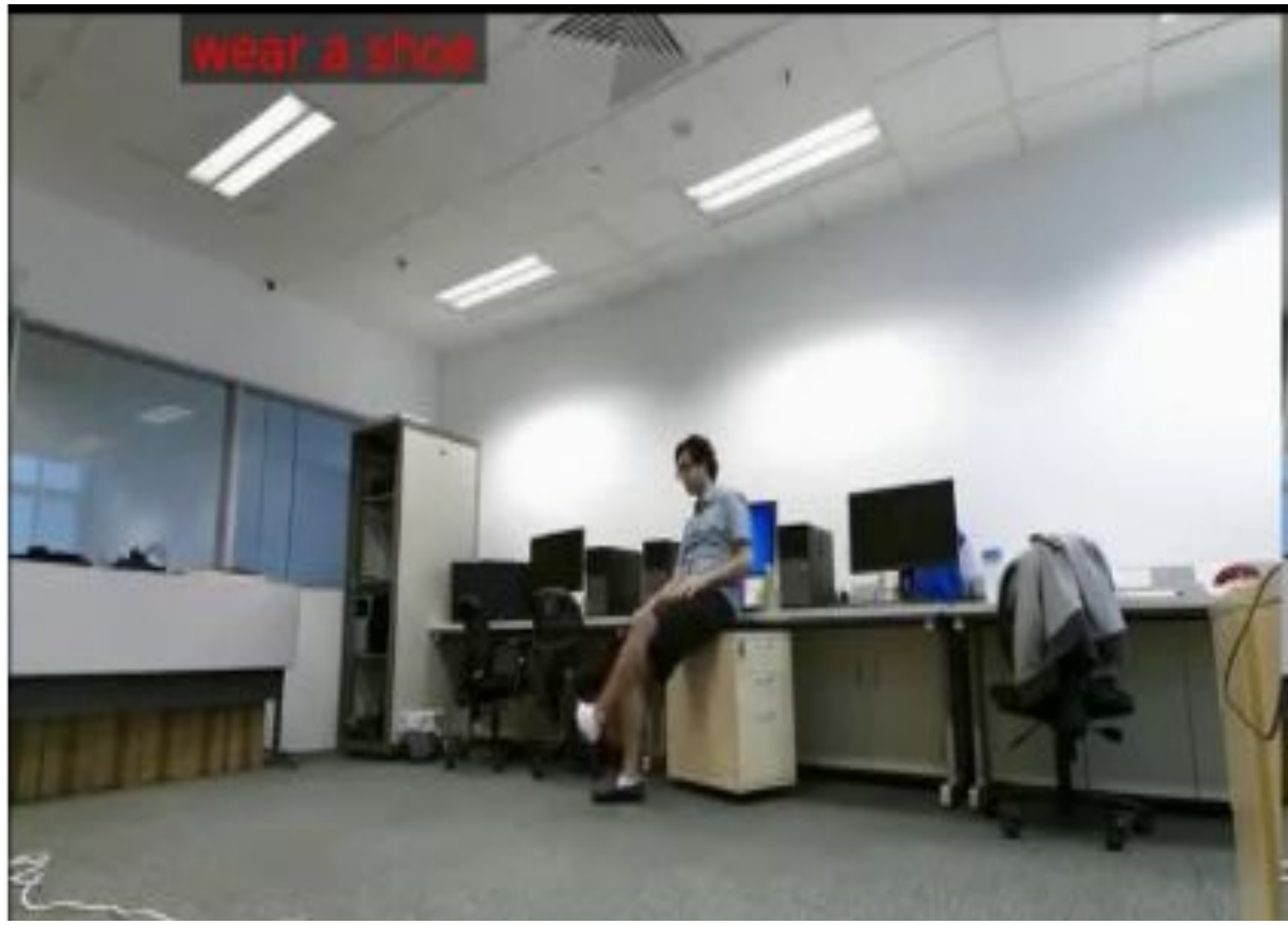
- Same Background
- Same Actions

- Different Posture (sit, stand)

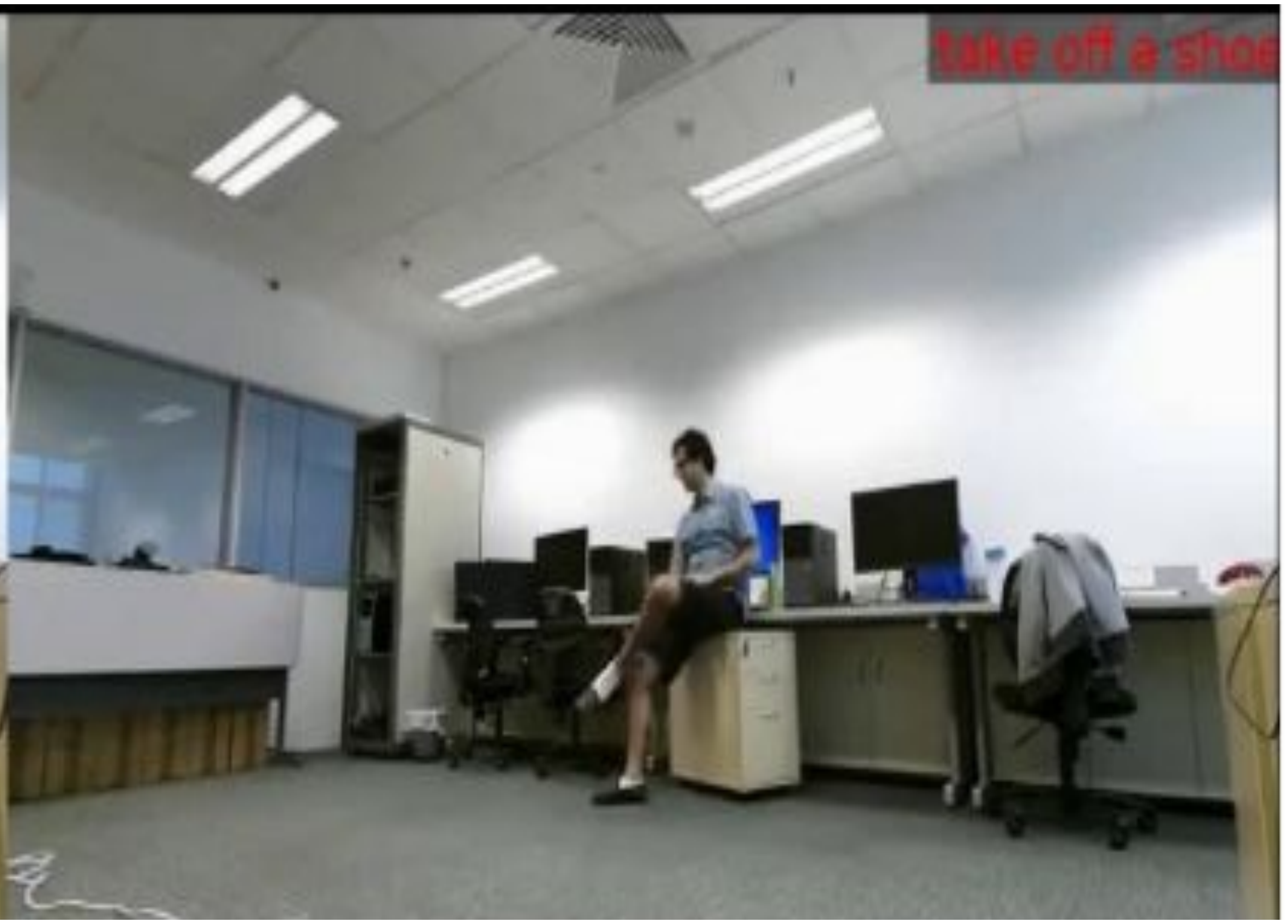
Challenges::

Low-Inter-class Variance:-

Wear shoes



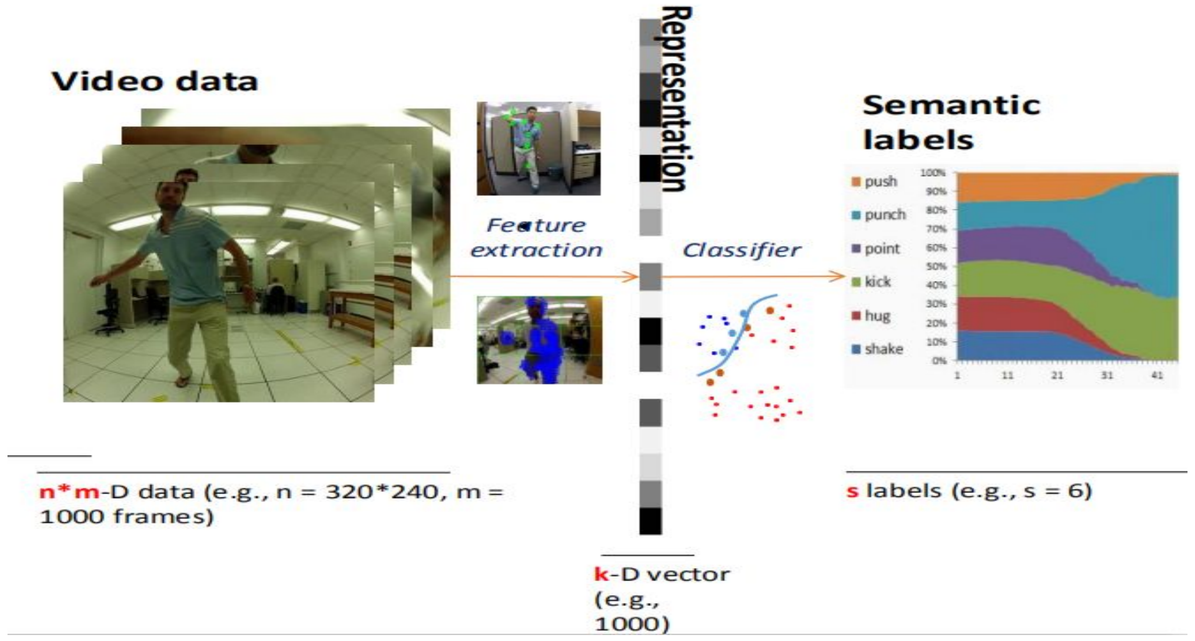
Take off shoes

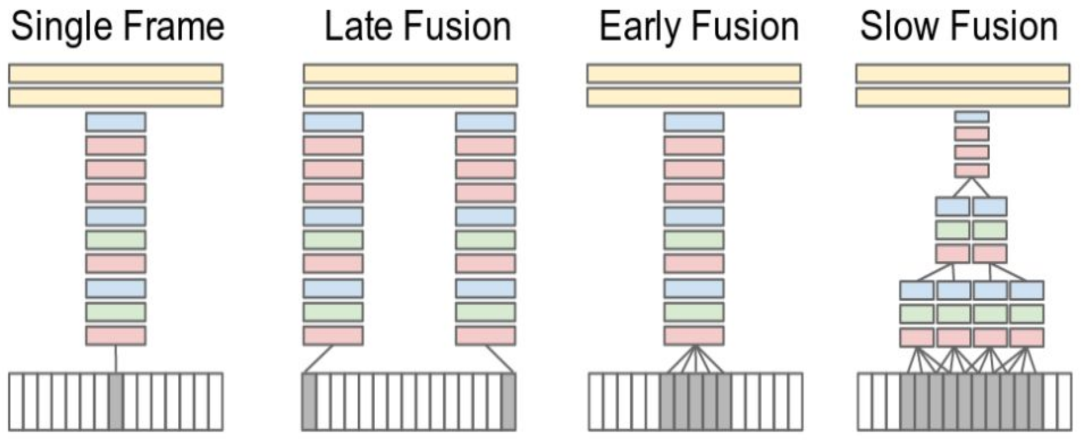


- **Same Background**
- **Almost Similar Posture**
- **Different Actions**

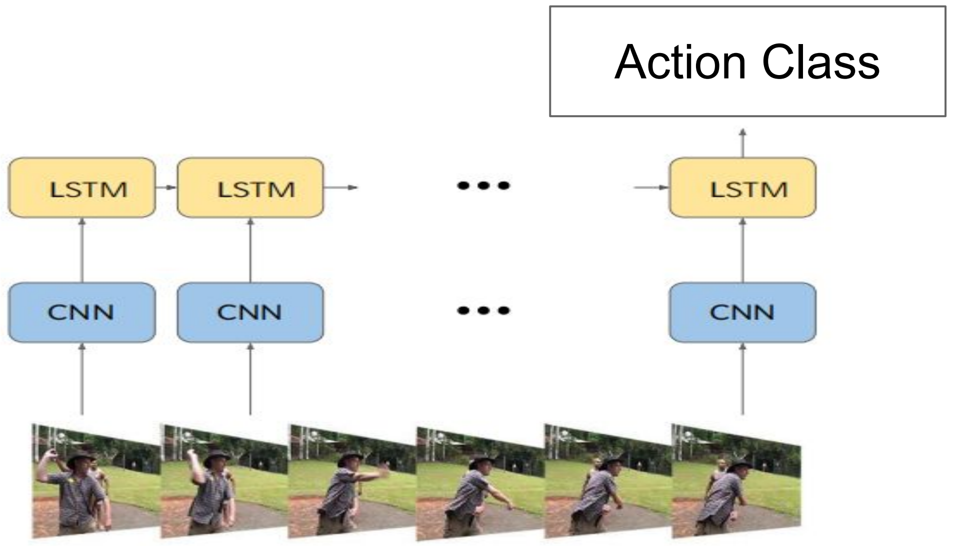
Quick Recap.::

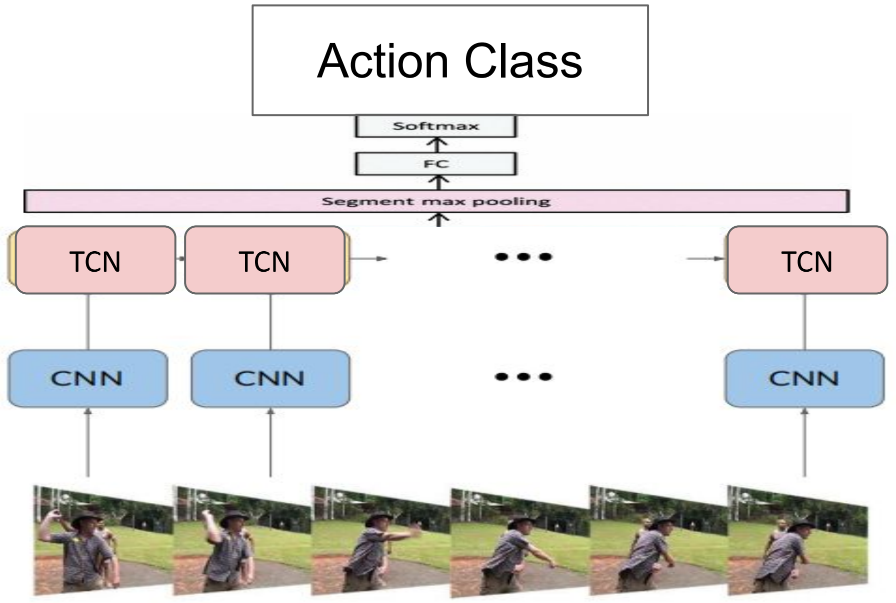
Classical Image Models



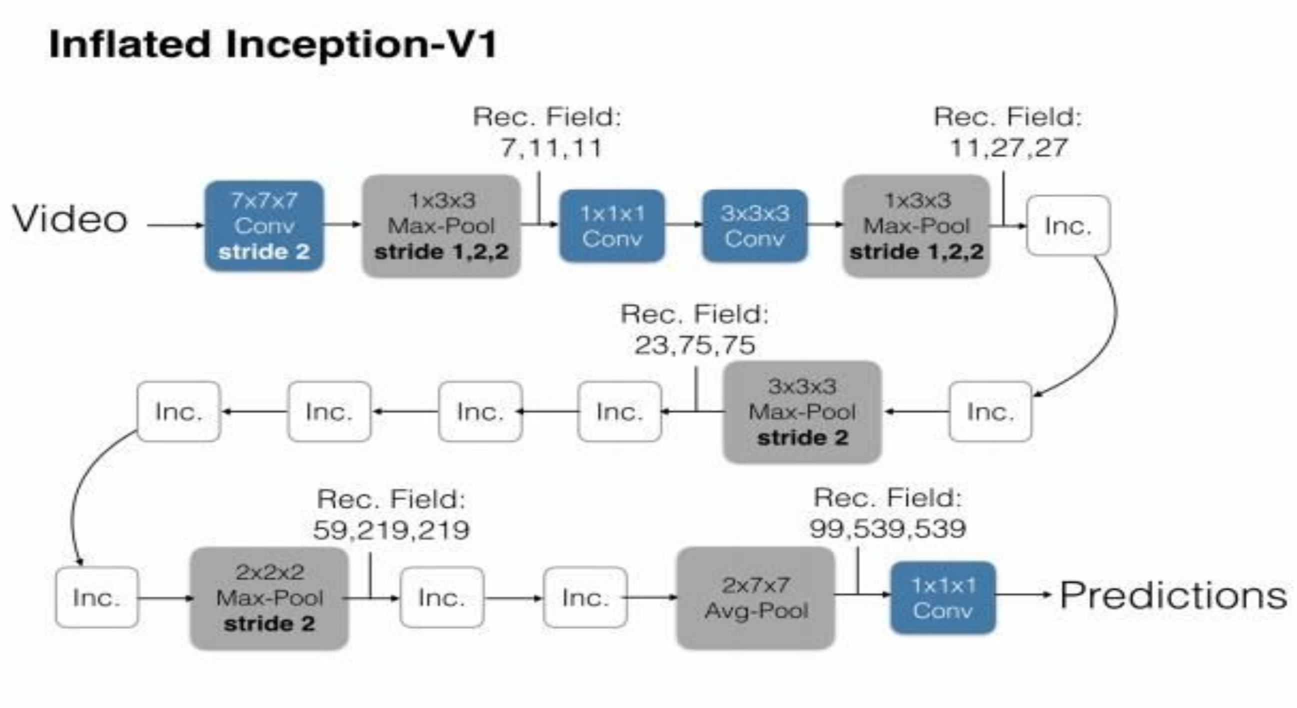


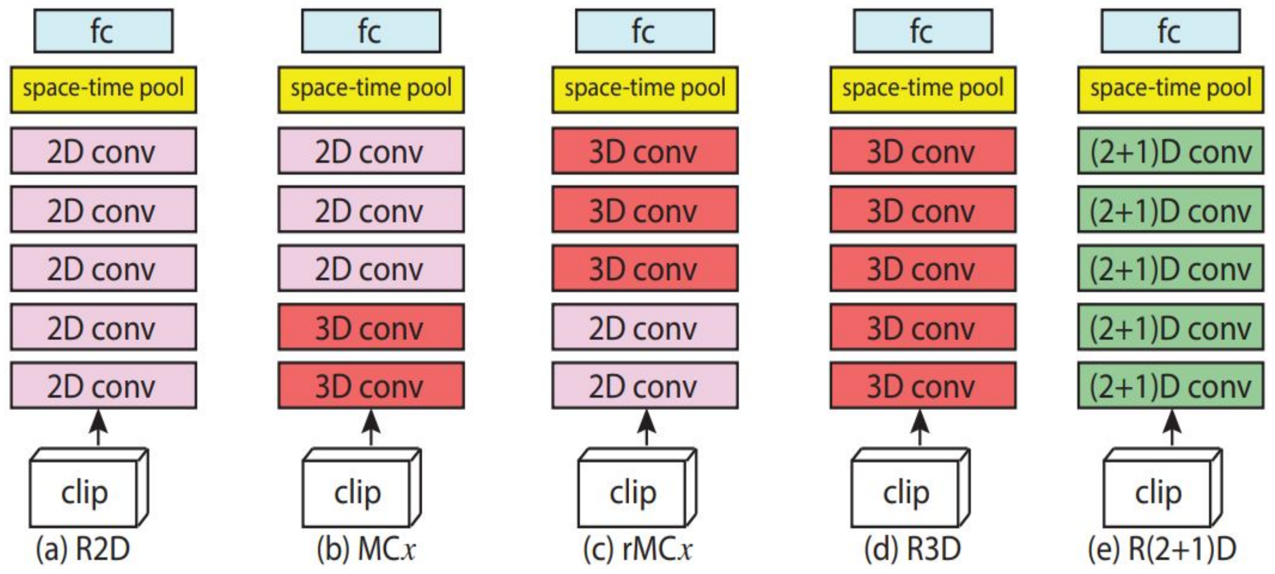
Classical Image Models with Temporal Models





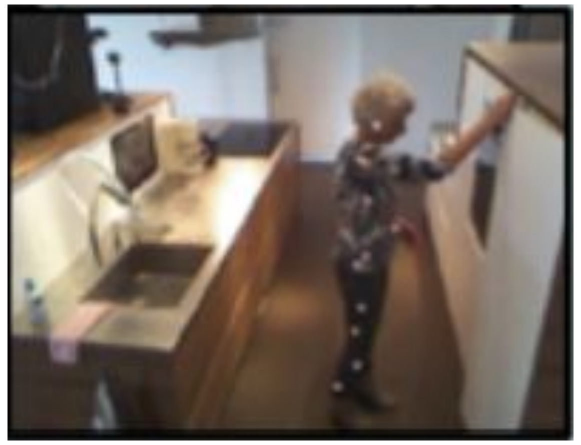
Classical Video Models



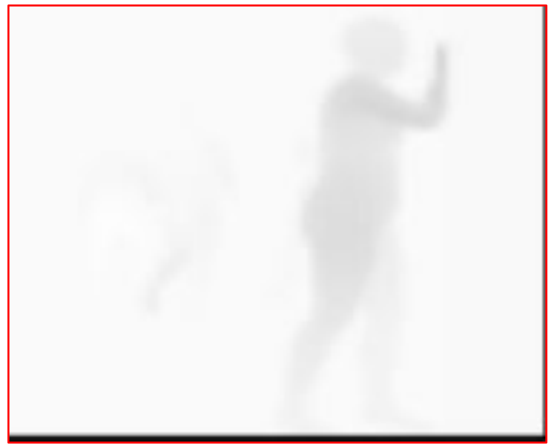


How to Tackle Challenges::

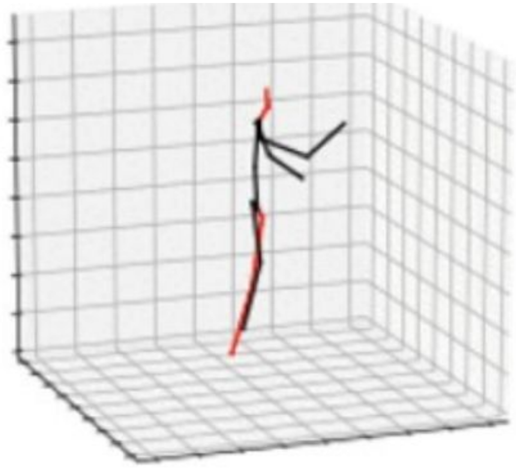
- Usage of Different Modalities to capture unique Cues



Appearance
(RGB)



Motion
(Optical Flow)



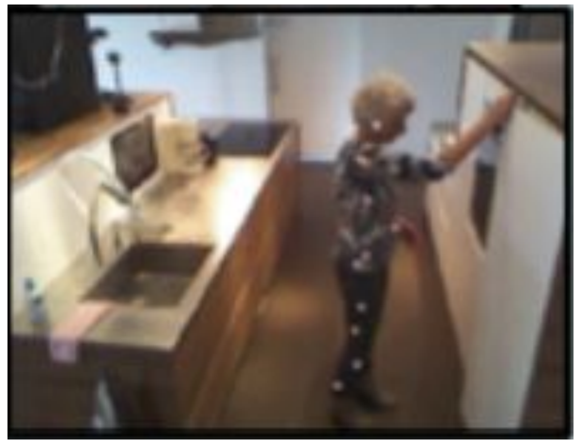
Posture
(3D Poses)

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

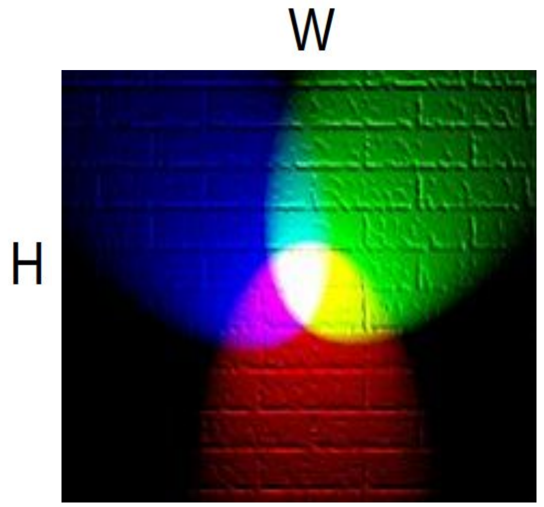
How to Tackle Challenges::

- 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.

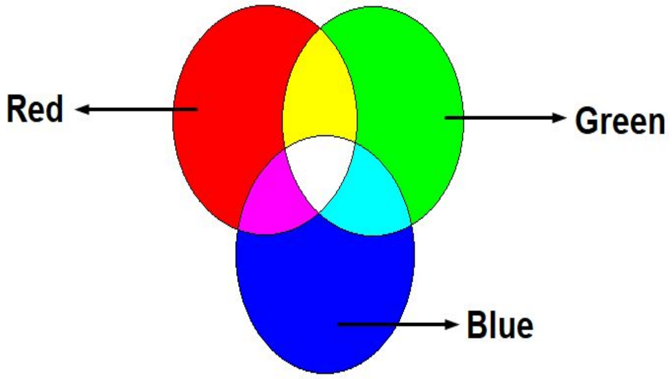
Multiple Modalities::



Appearance
(RGB)



Tensor: [H × W × 3] × T

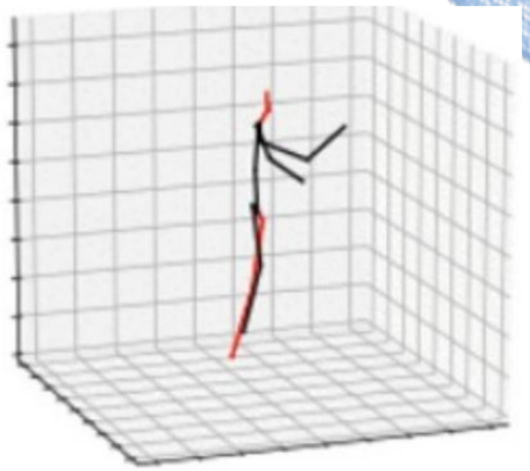


Motion
(Optical Flow)

- 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. (TVF1, FlowNet, PwcNet)

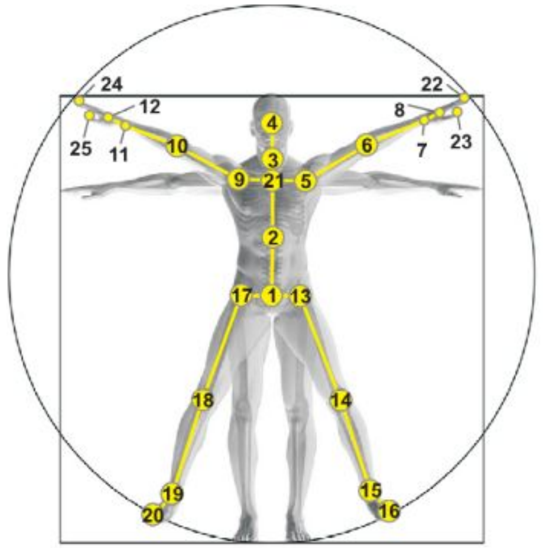


Posture
(3D Poses/Skeletons)

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

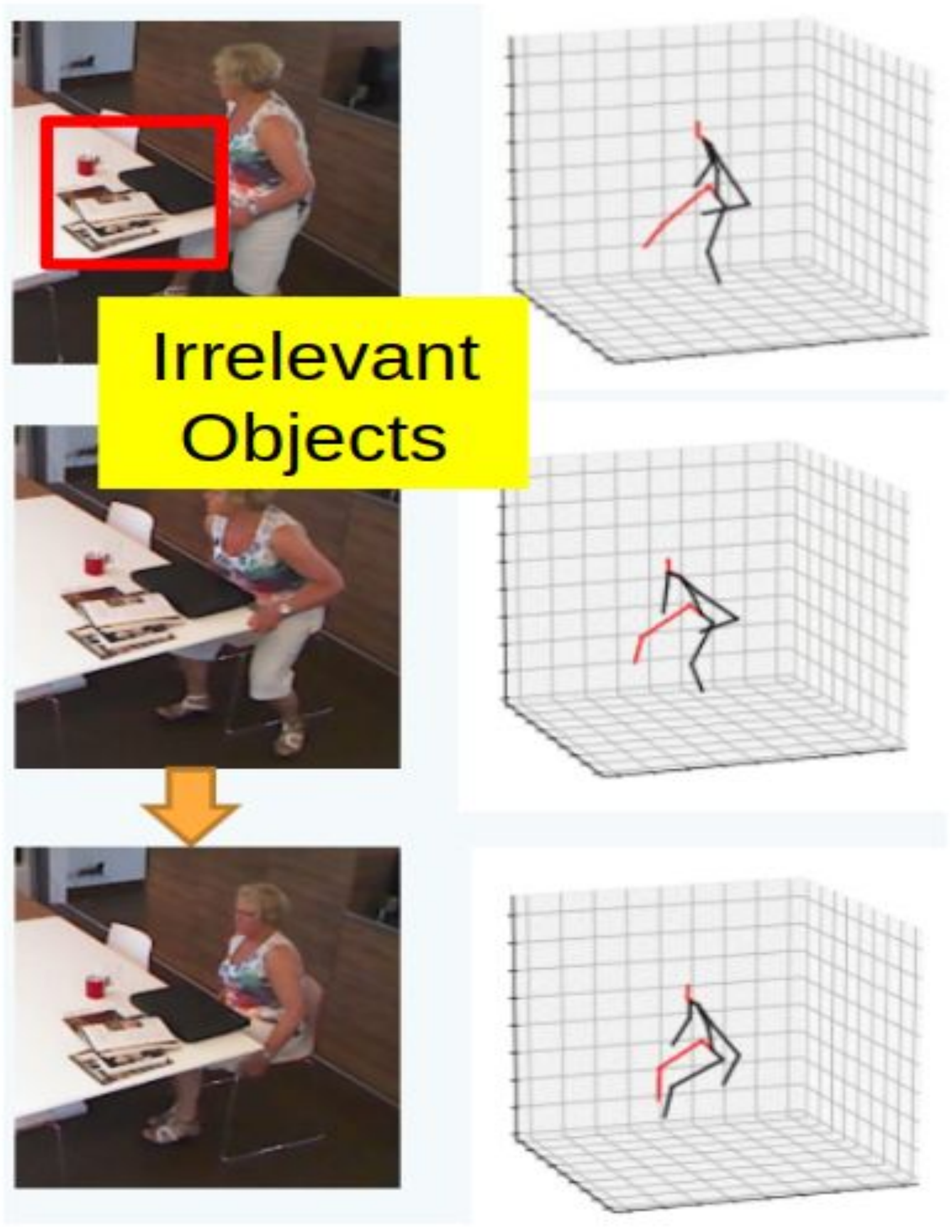
Tensor: [N ×] × T

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



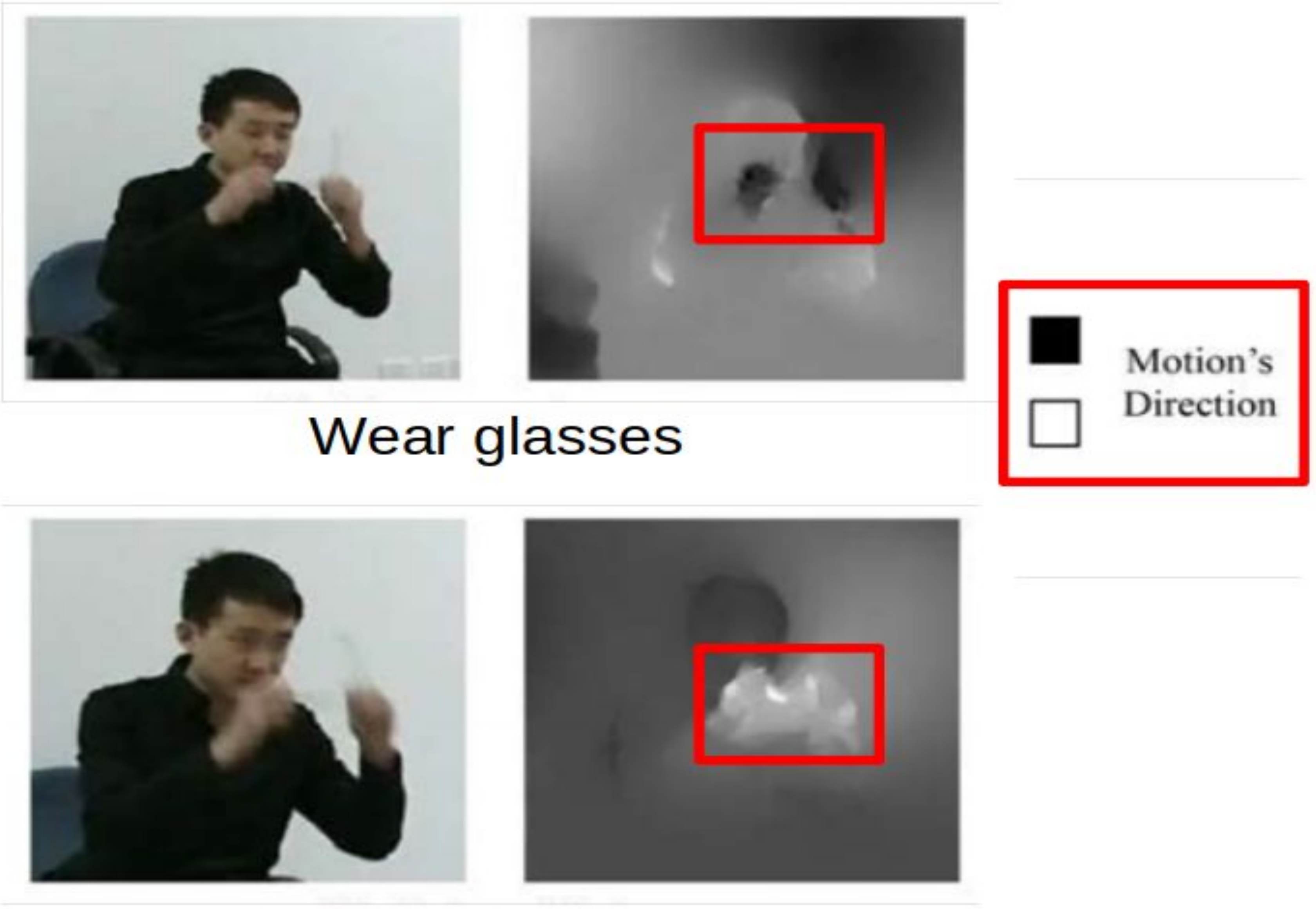
Benefits of Combining Multiple Modalities::

- Provide complementary information.



Sit down

3D poses

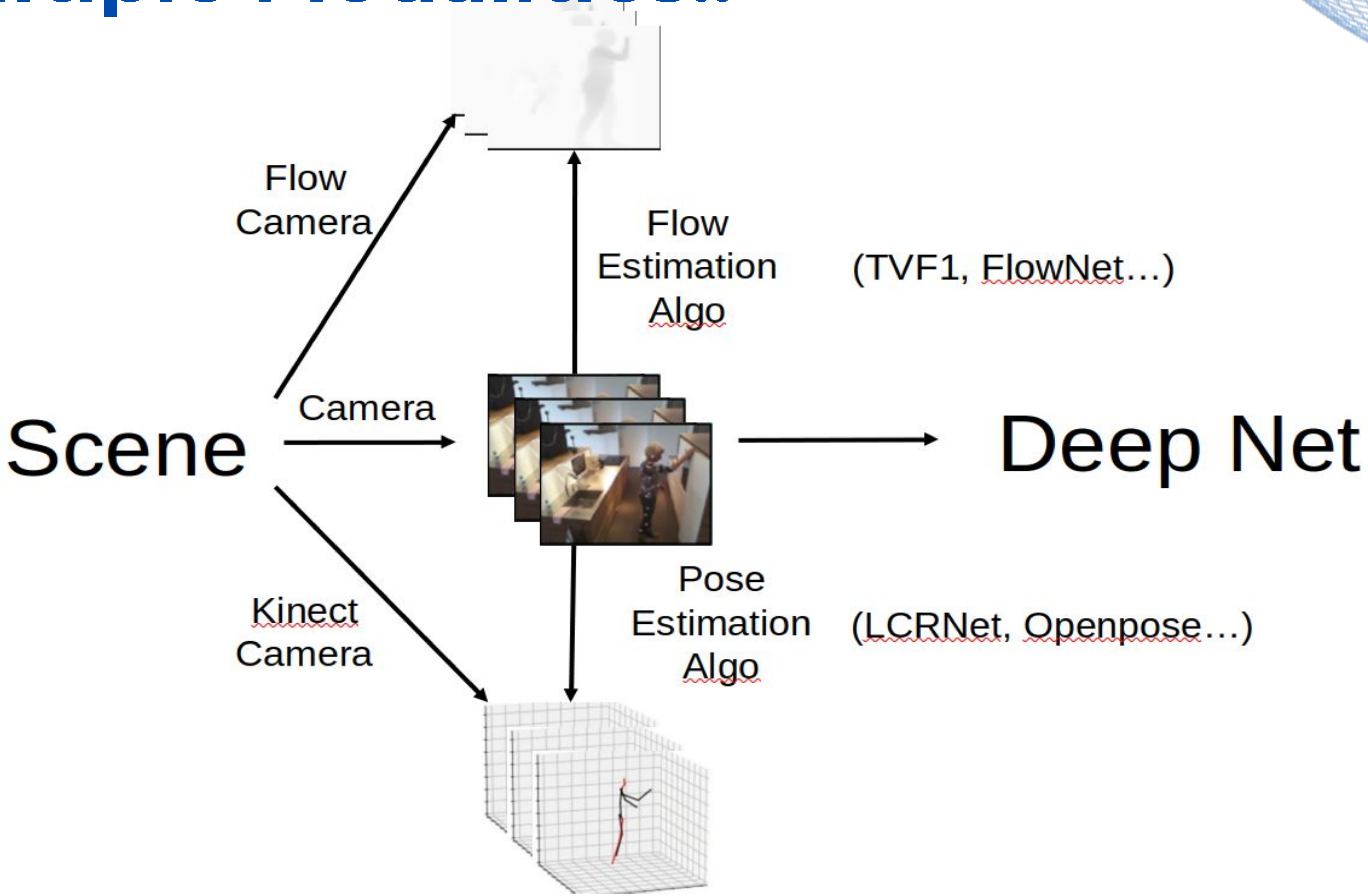


Wear glasses

Take off glasses

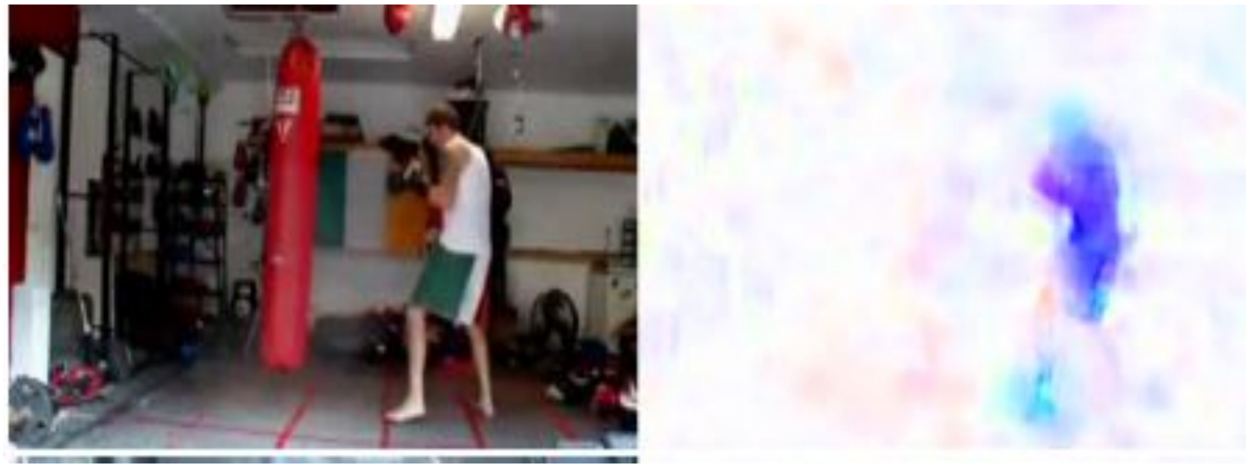
Optical flow

Benefits of Combining Multiple Modalities::

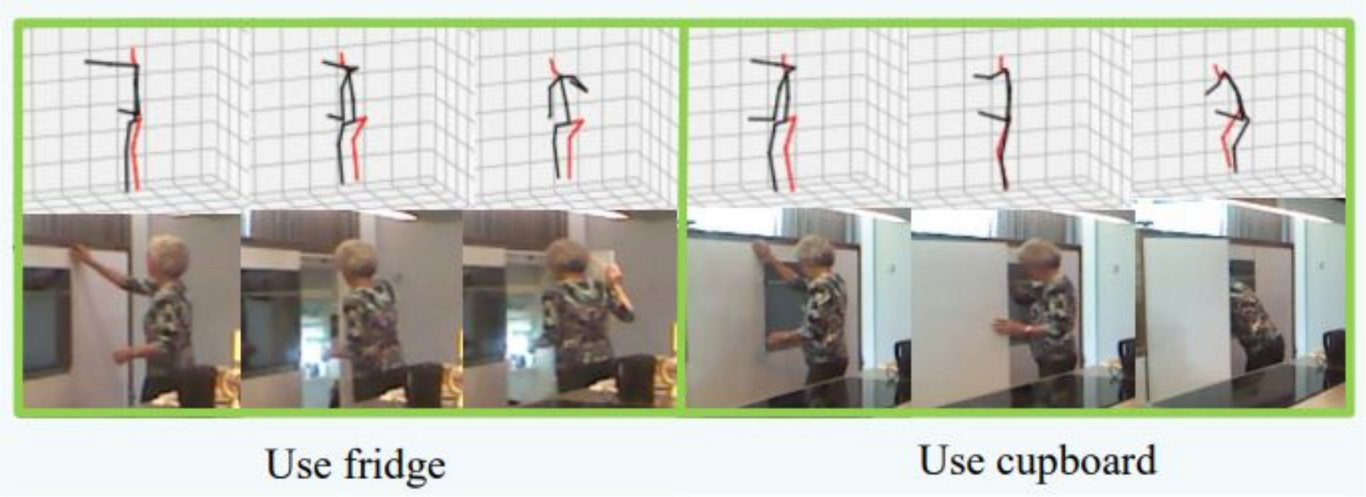


Drawbacks of Different Modalities::

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



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

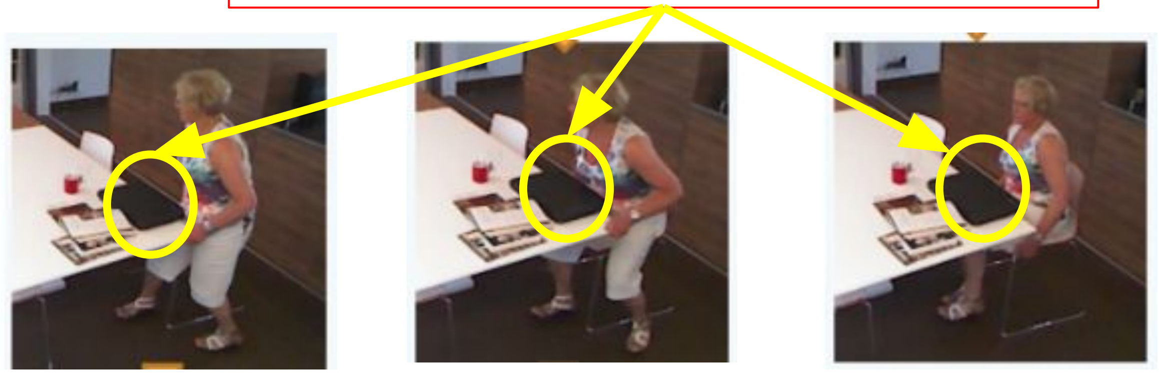


Use fridge

Use cupboard

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

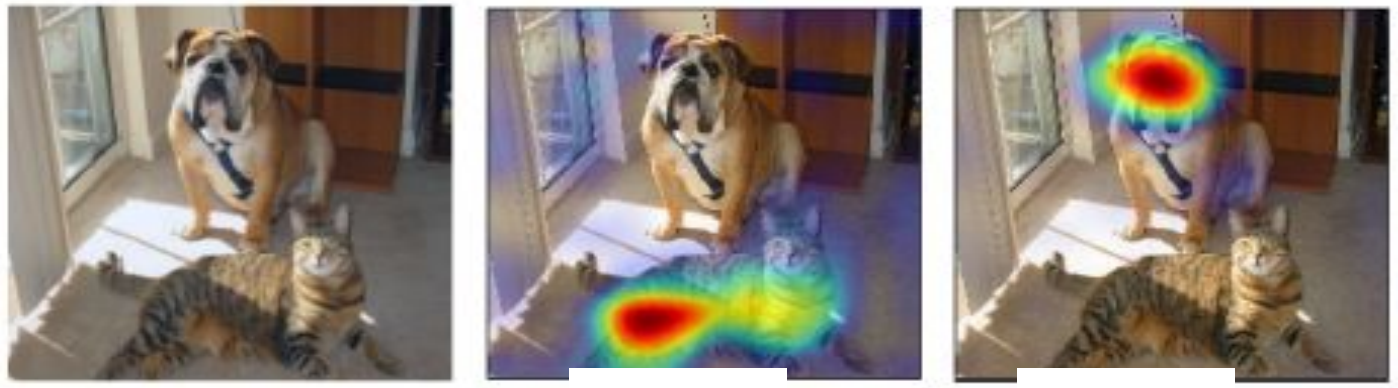
Irrelevant Objects (Laptop, Books) Info.



Action: Sit Down

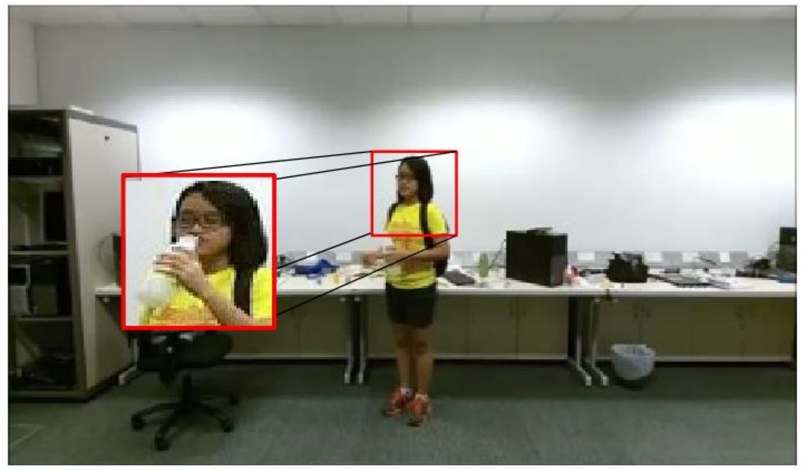
Attention Mechanism::

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

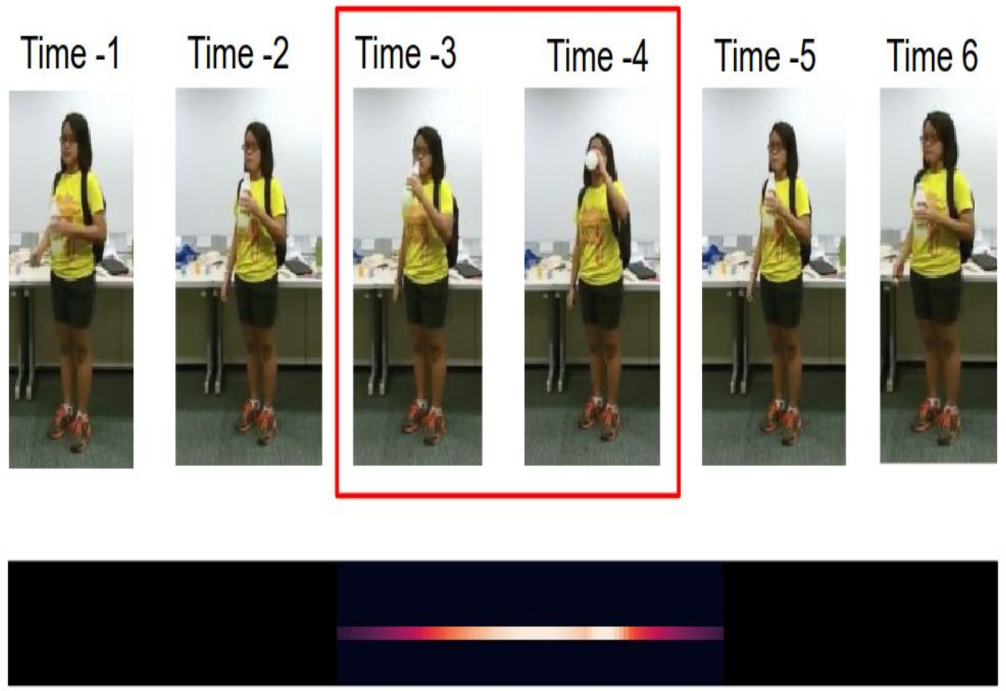

Original Image   Focus on 'Cat'   Focus on 'Dog'

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

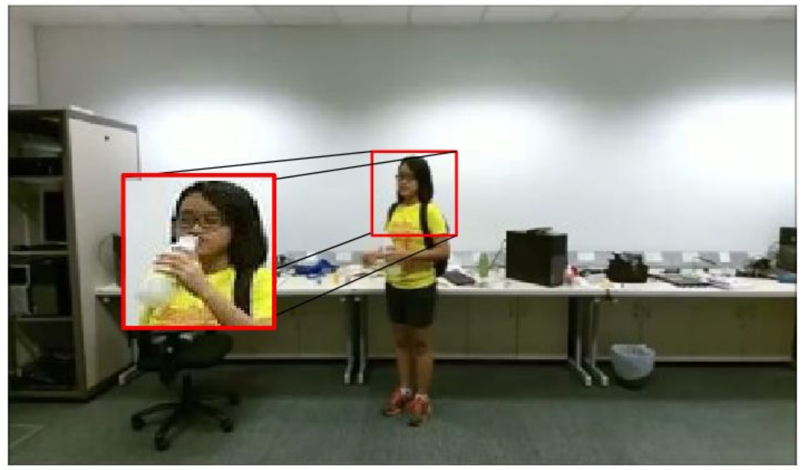
Temporal Attention

Classical Attention Mechanism::

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

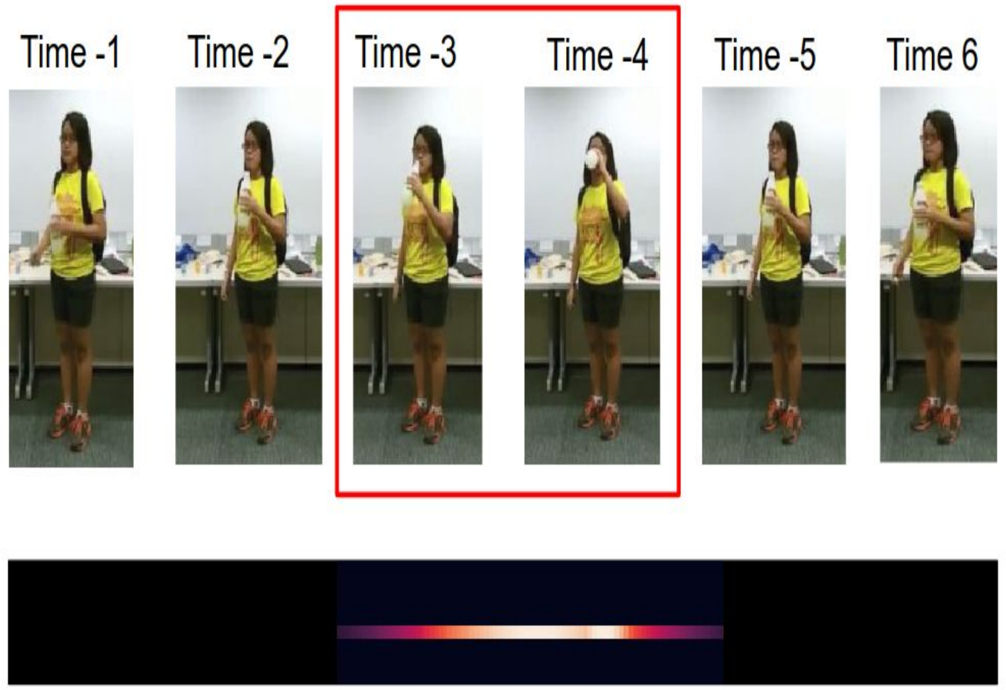
The girl is drinking water from a bottle

Do we really need the whole video to infer that?



➢ Isn't this enough for an inference?

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

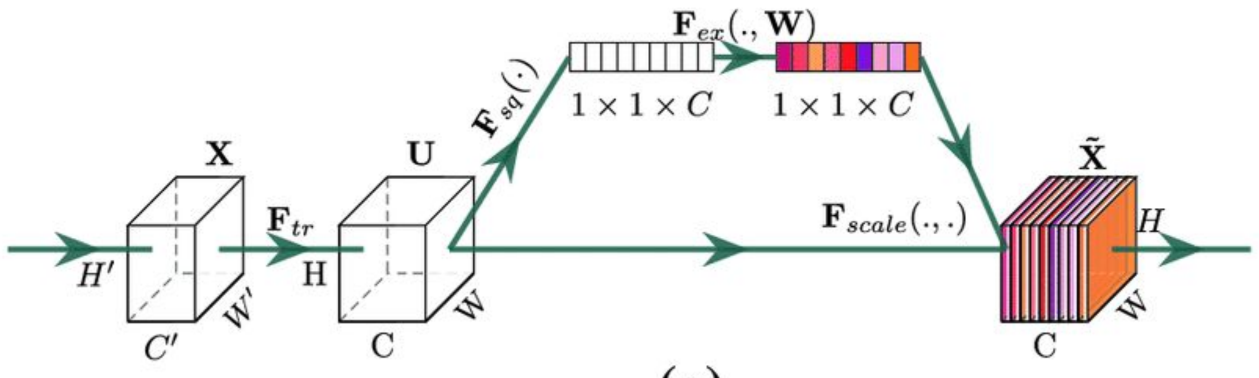


Spatial Attention

Temporal Attention

Squeeze-and-Excitation Attention::

- **Observation in CNN:**
  - Feature Extraction from CNN shrinks the spatial Dimension and expands the channel dimension
  - All <u>channels are weighted equally</u> when considering the output feature map
- **Key Idea:** Assign each channel a different weightage based on how important each channel is .



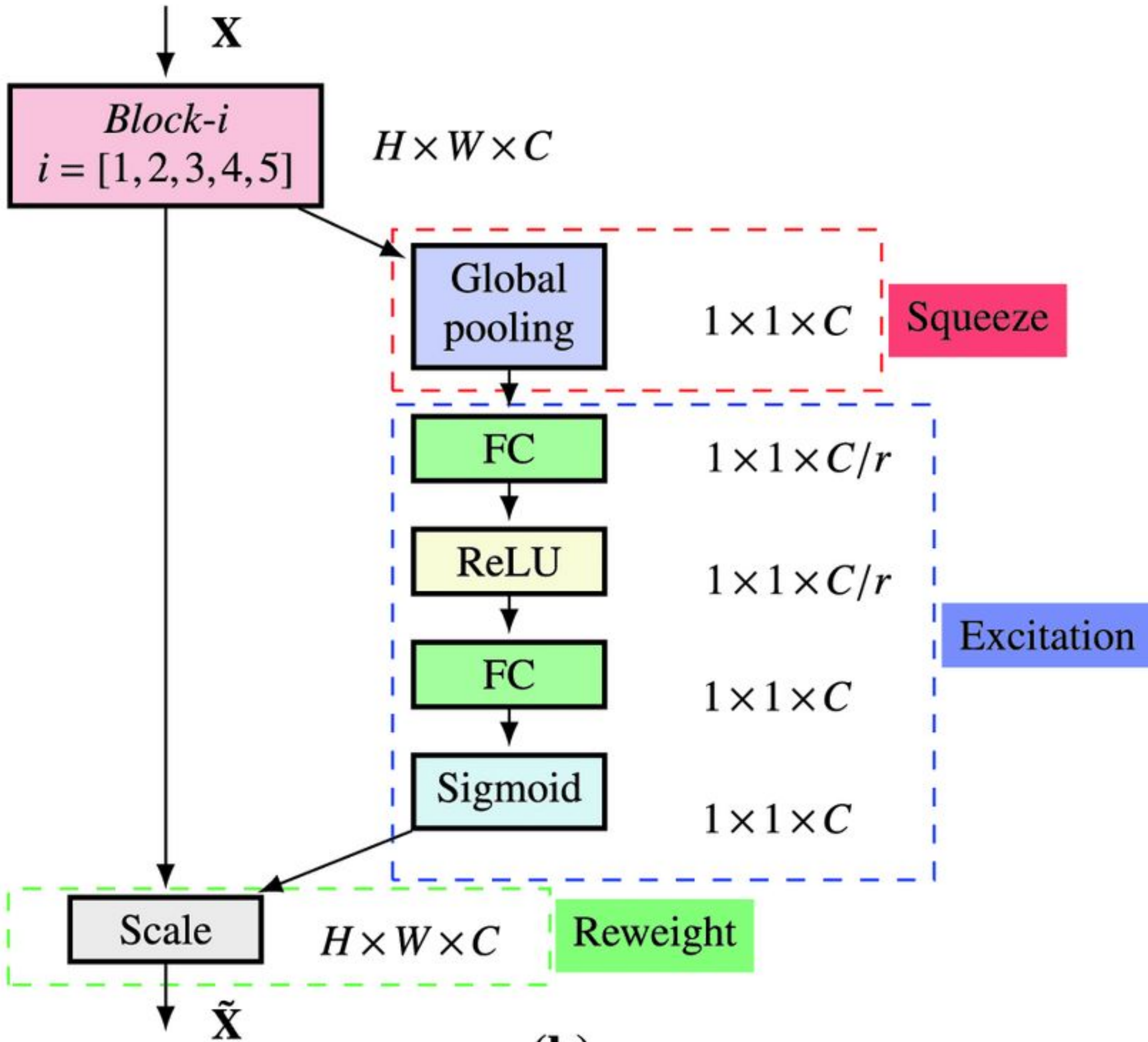
(a)

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).

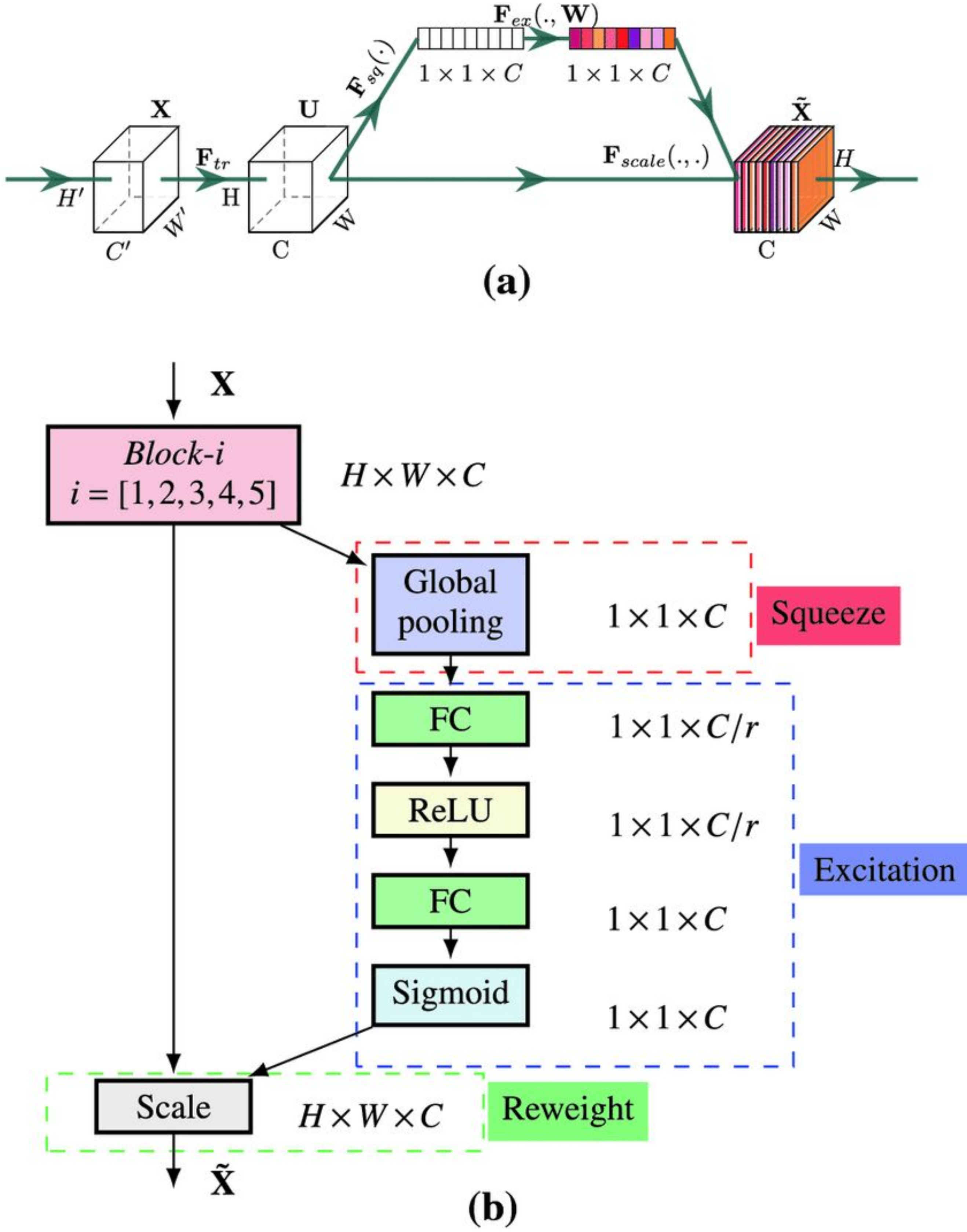
Reweight: Lastly, the output of the computation step is used as a per-channel weight modulation vector. It is simply multiplied with the original input feature map of size (H x W x C). This scales the spatial maps for each channel according to their 'importance'.



(b)

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 .



(a)

(b)

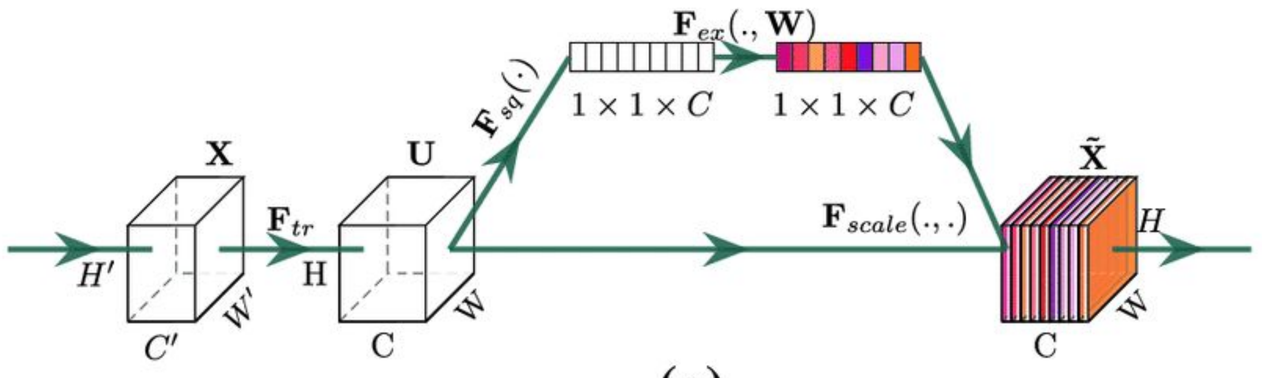
Squeeze-and-Excitation Attention::

3 main Parts of SE:

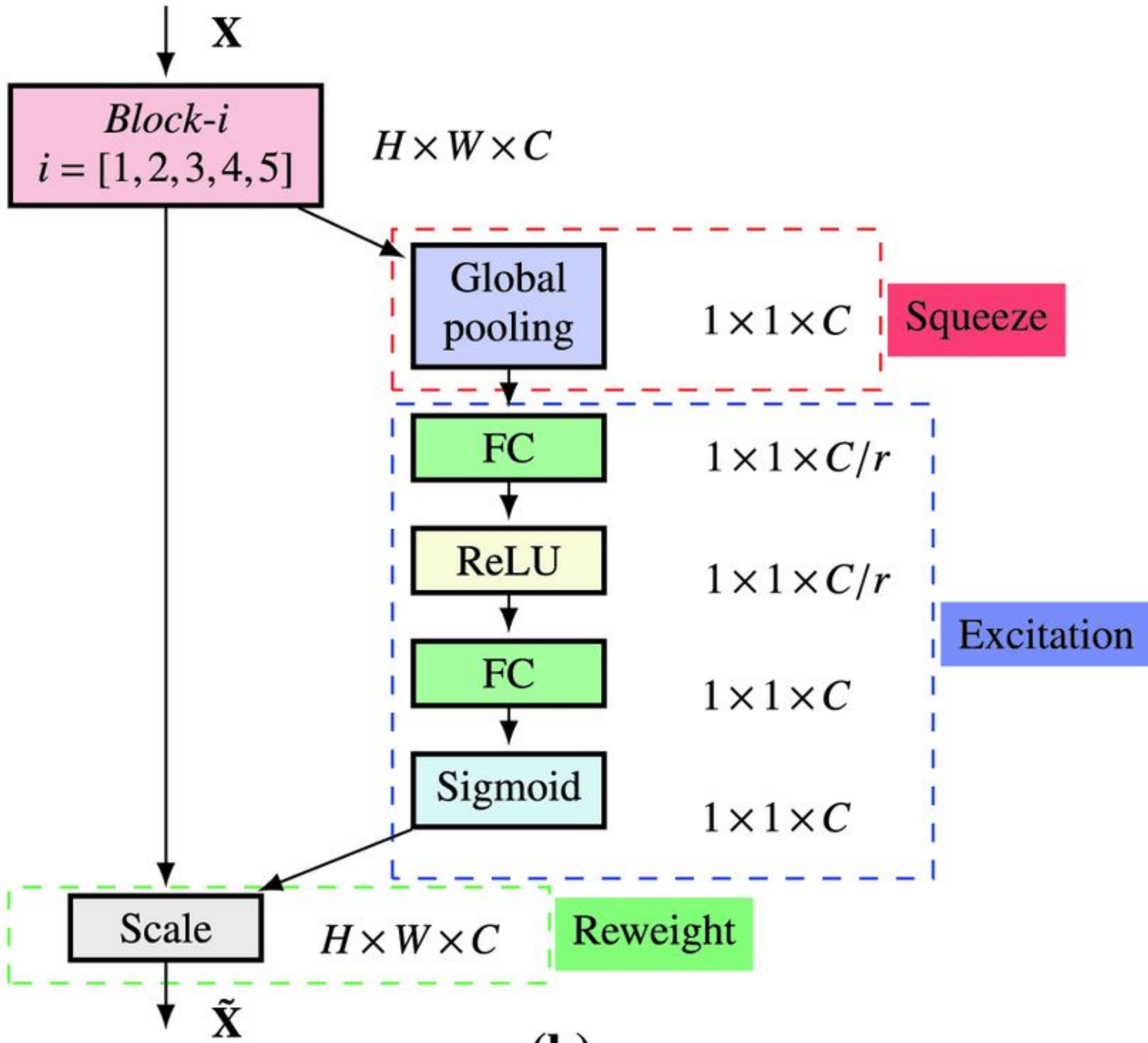
Squeeze: Global Average Pooling is performed on the output feature map of the CNN layer across H and W and the result of output tensor shape is 1 x 1 x C.

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'.



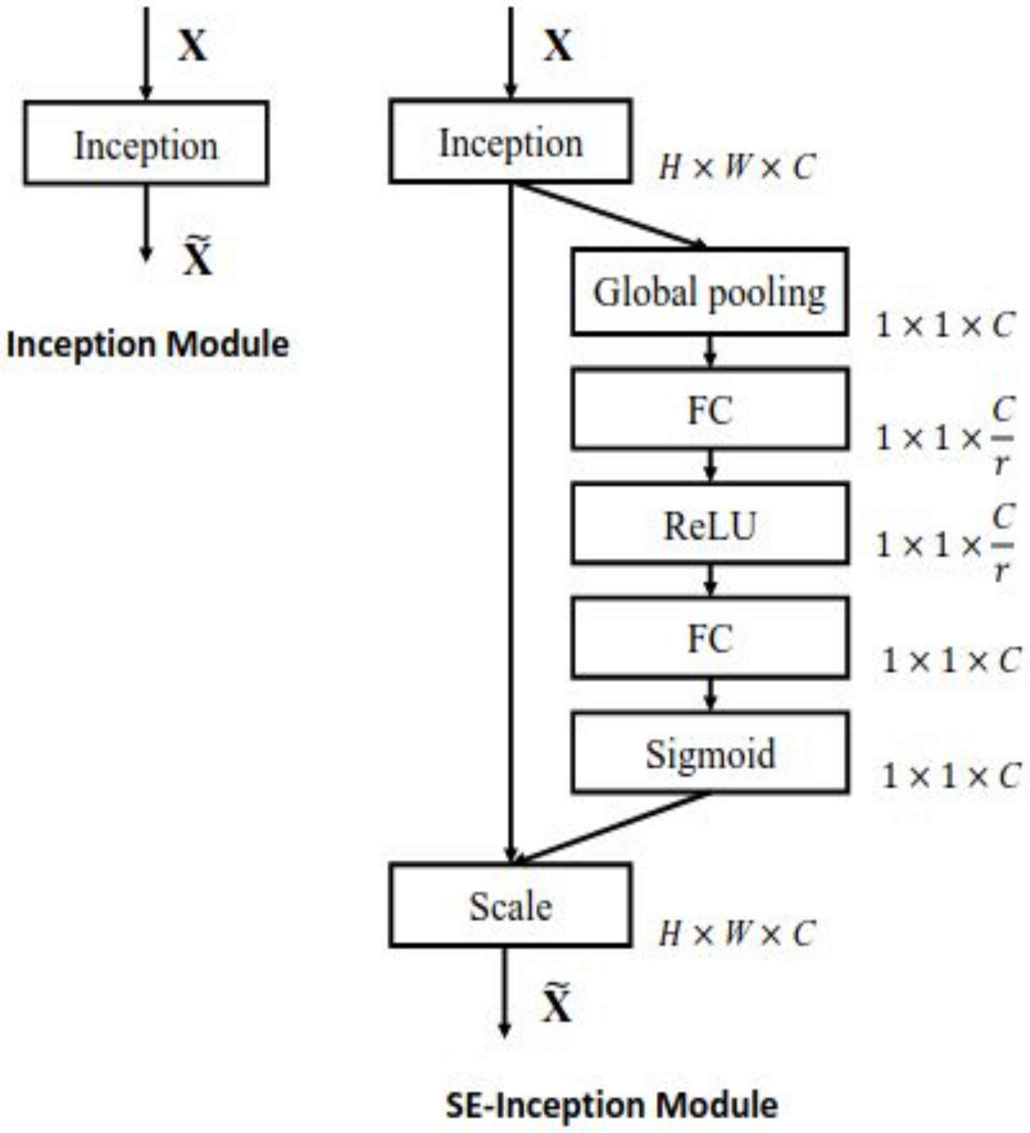
(a)



(b)

Squeeze-and-Excitation Attention::

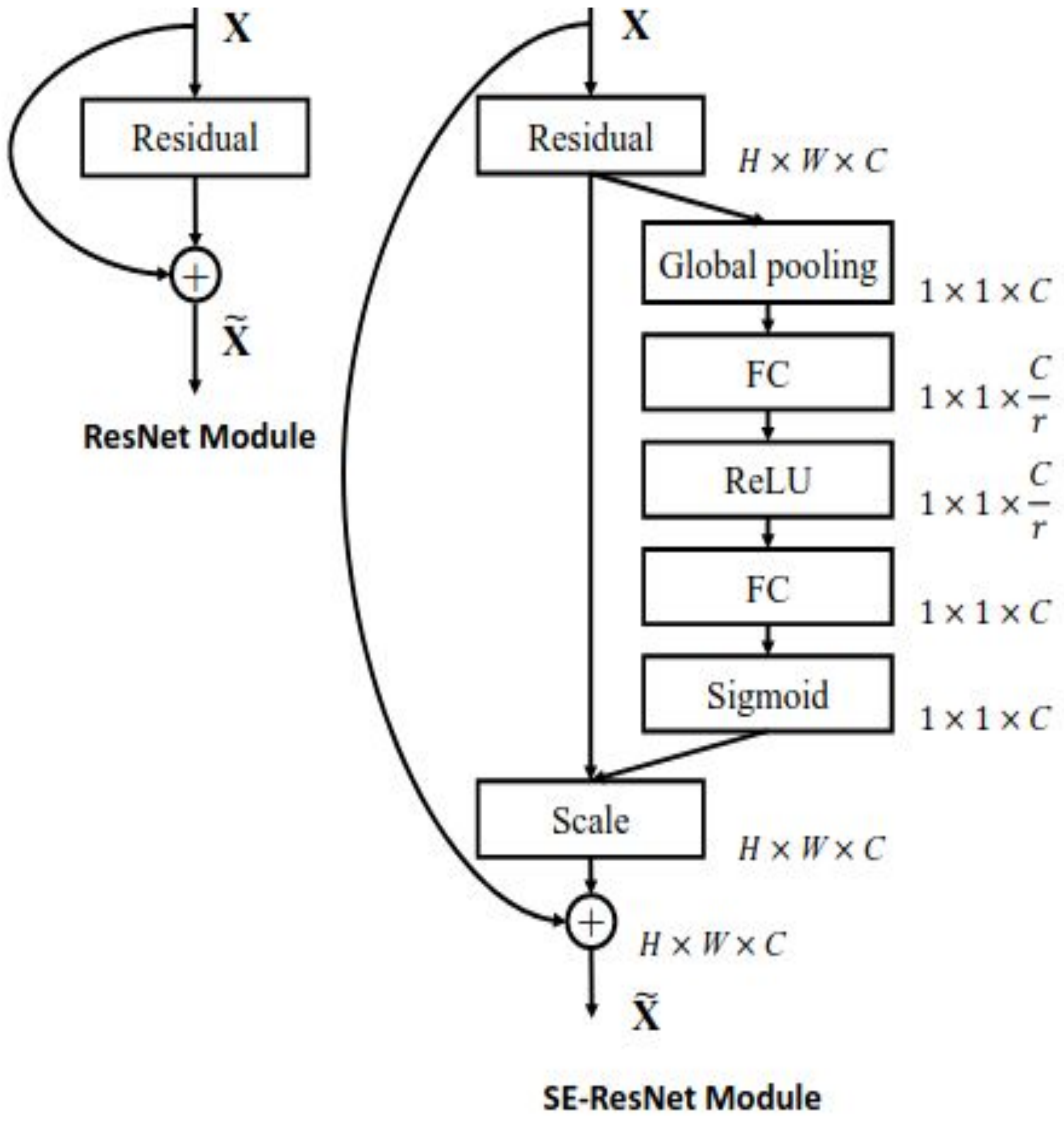
SE Blocks can be easily integrated with many existing CNNs like Inception V1, ResNets, etc.



Inception Module

SE-Inception Module

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



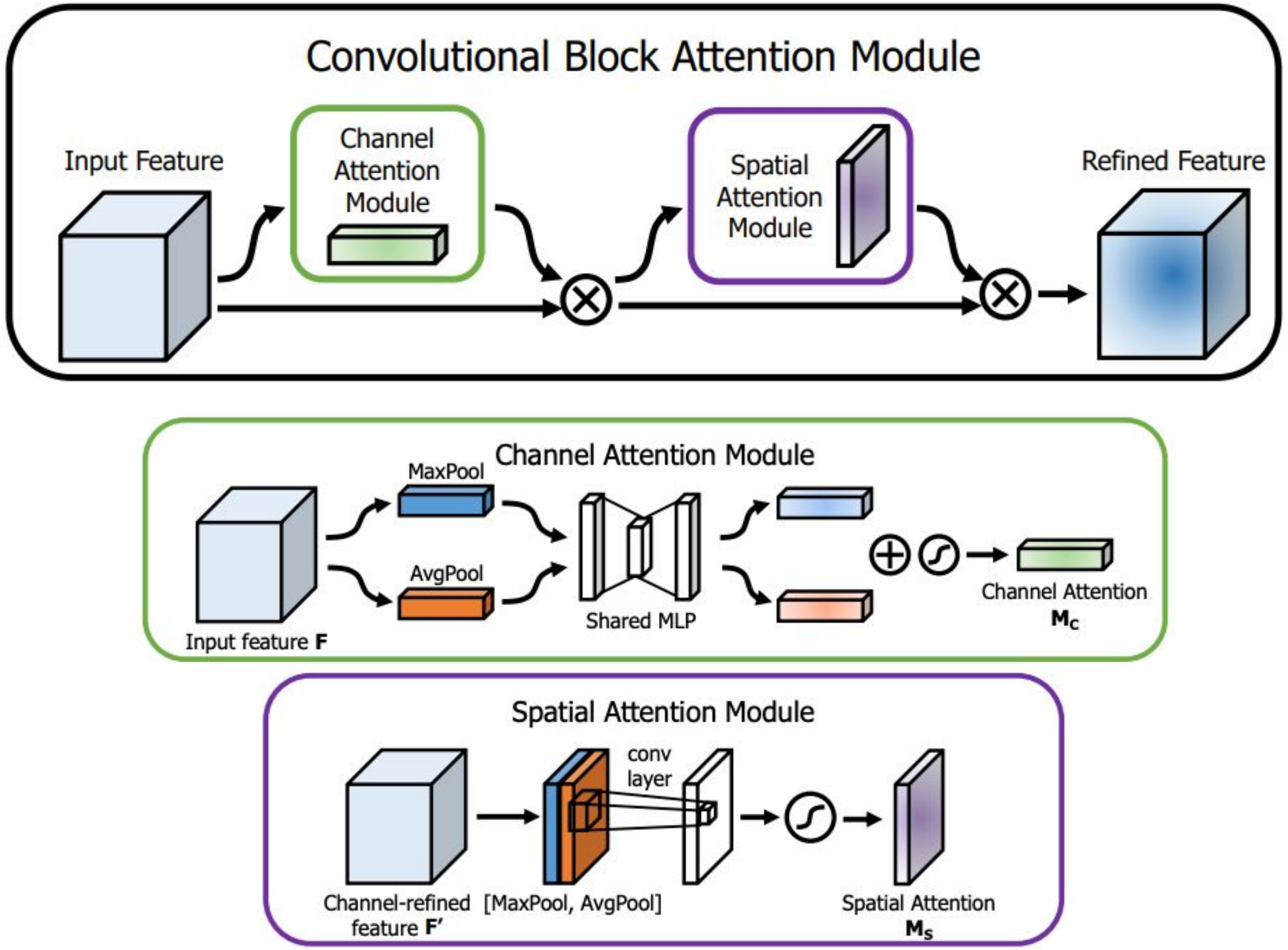
ResNet Module

SE-ResNet Module

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

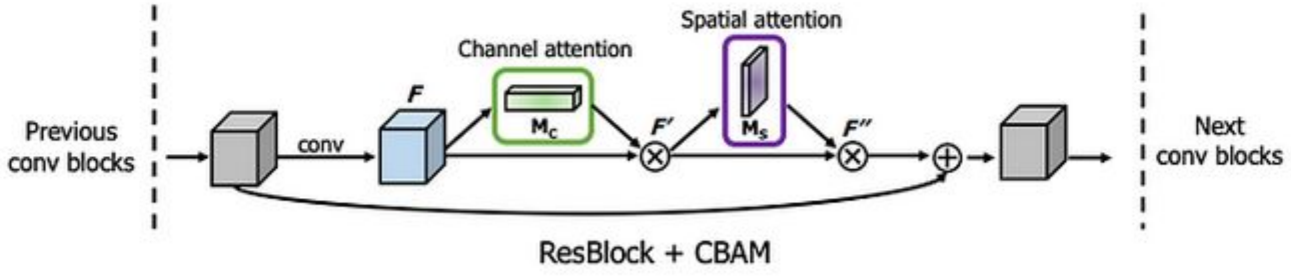
Convolutional Block Attention Module (CBAM)::

- **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**

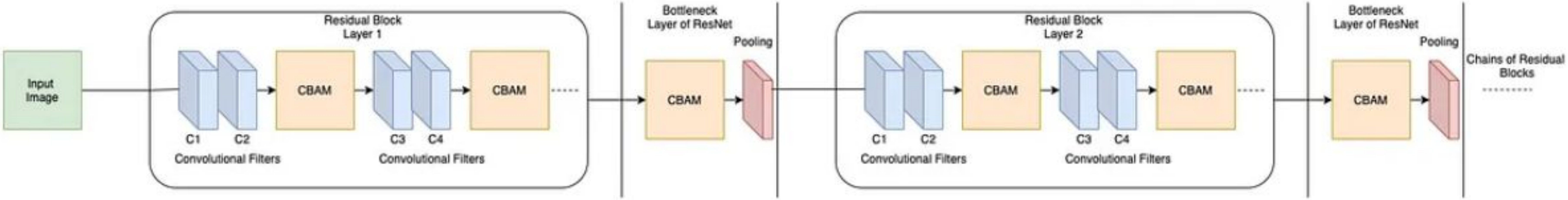


- **CBAM** is applied at every convolutional block in deep networks to get subsequent **"Refined Feature Maps"** from the **"Input Intermediate Feature Maps".**

Convolutional Block Attention Module (CBAM)::



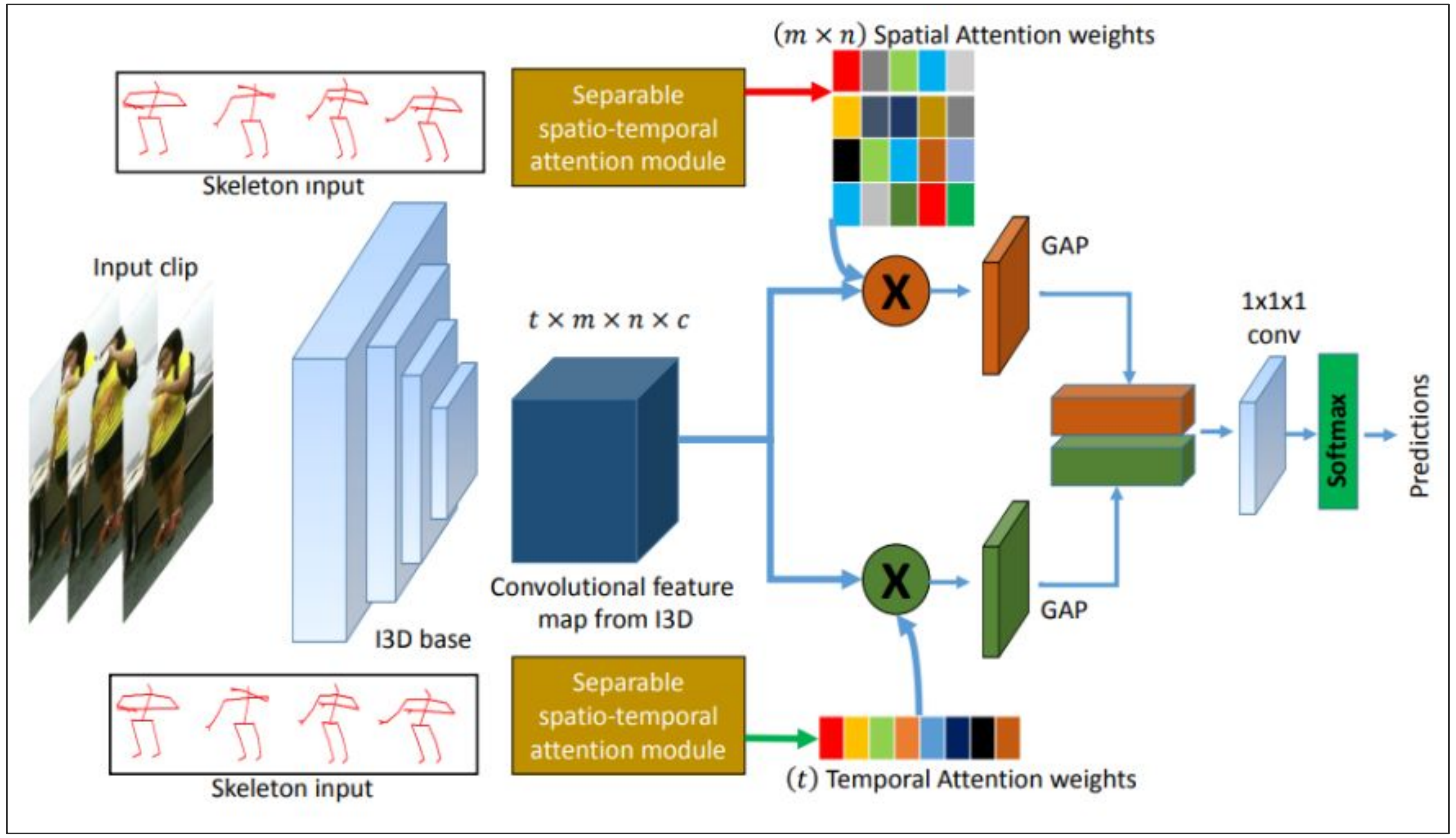
Placement of Spatial and Channel Attention Modules sequentially.



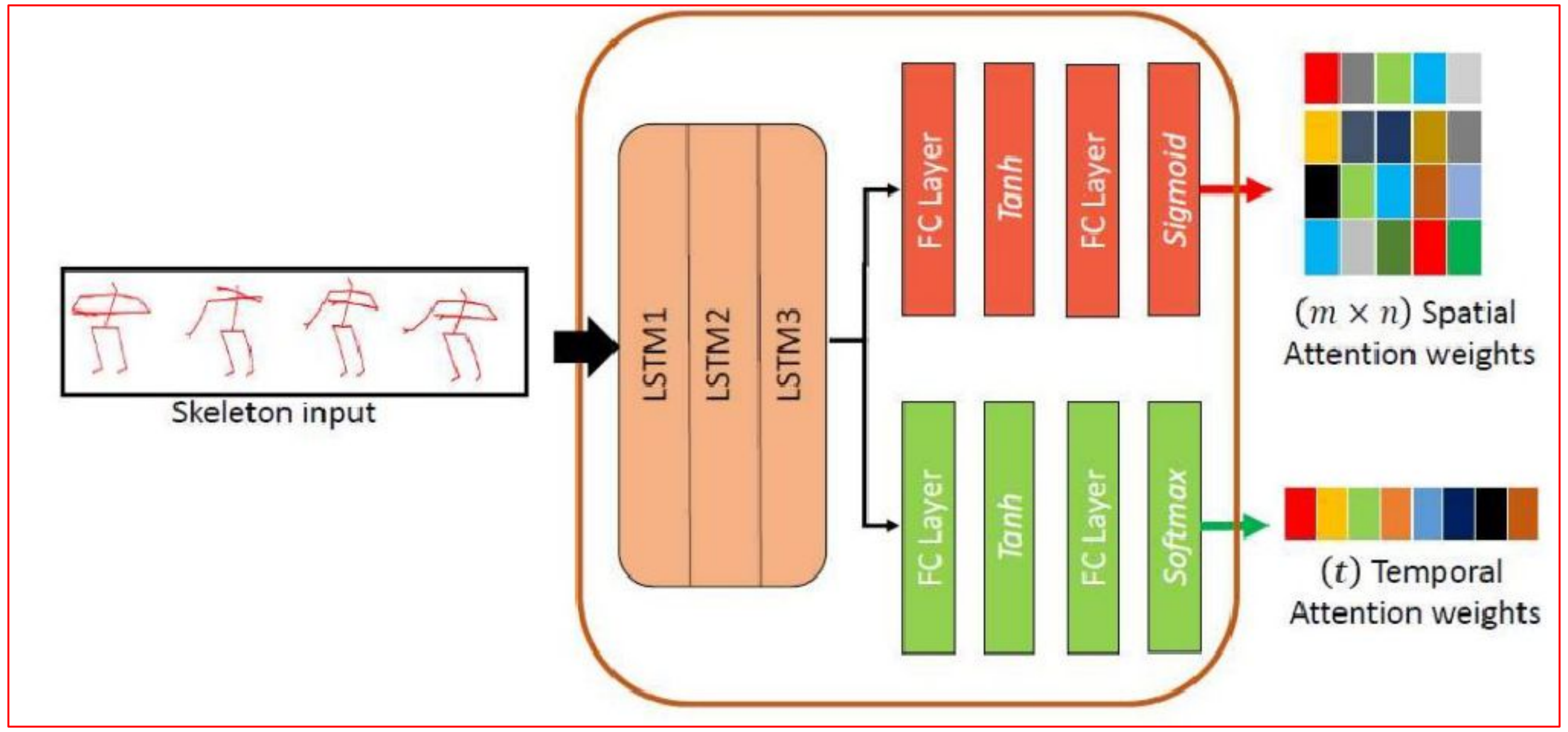
Placement of CBAM module in ResNet architecture.

Spatio-Temporal Attention::

- **Key Idea:** To 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* × *h* × *w*) dimension input tensor to 2 channels, i.e. (2 × *h* × *w*), where each of the 2 channels represent Max Pooling and Average Pooling across the channels. **(ii) Convolutional Layer, (iii) Batch Norm**

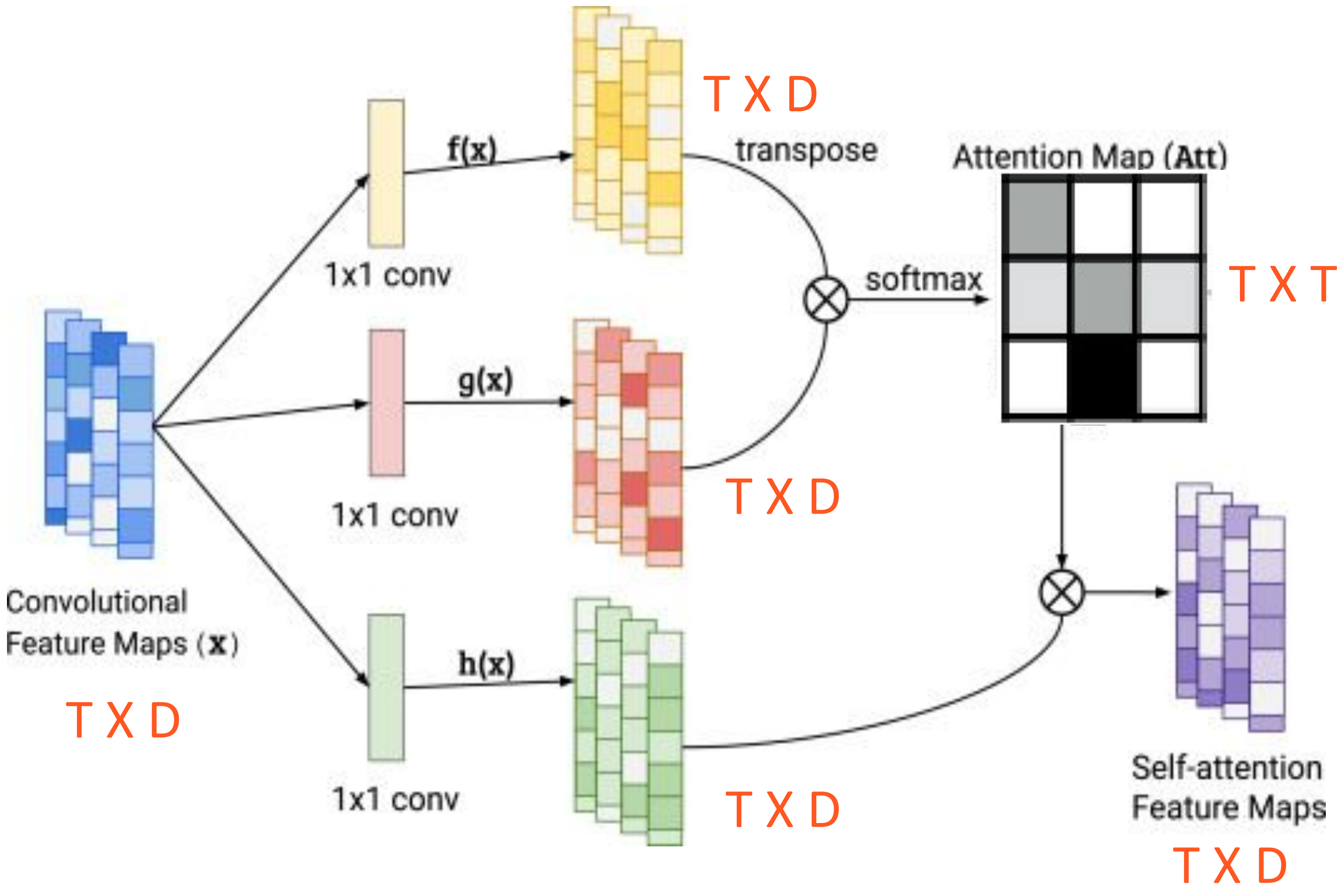


$(m \times n)$ Spatial Attention weights
Skeleton input
Separable spatio-temporal attention module
Input clip
I3D base
$t \times m \times n \times c$
Convolutional feature map from I3D
GAP
GAP
1x1x1 conv
Softmax
Predictions
Skeleton input
Separable spatio-temporal attention module
(t) Temporal Attention weights



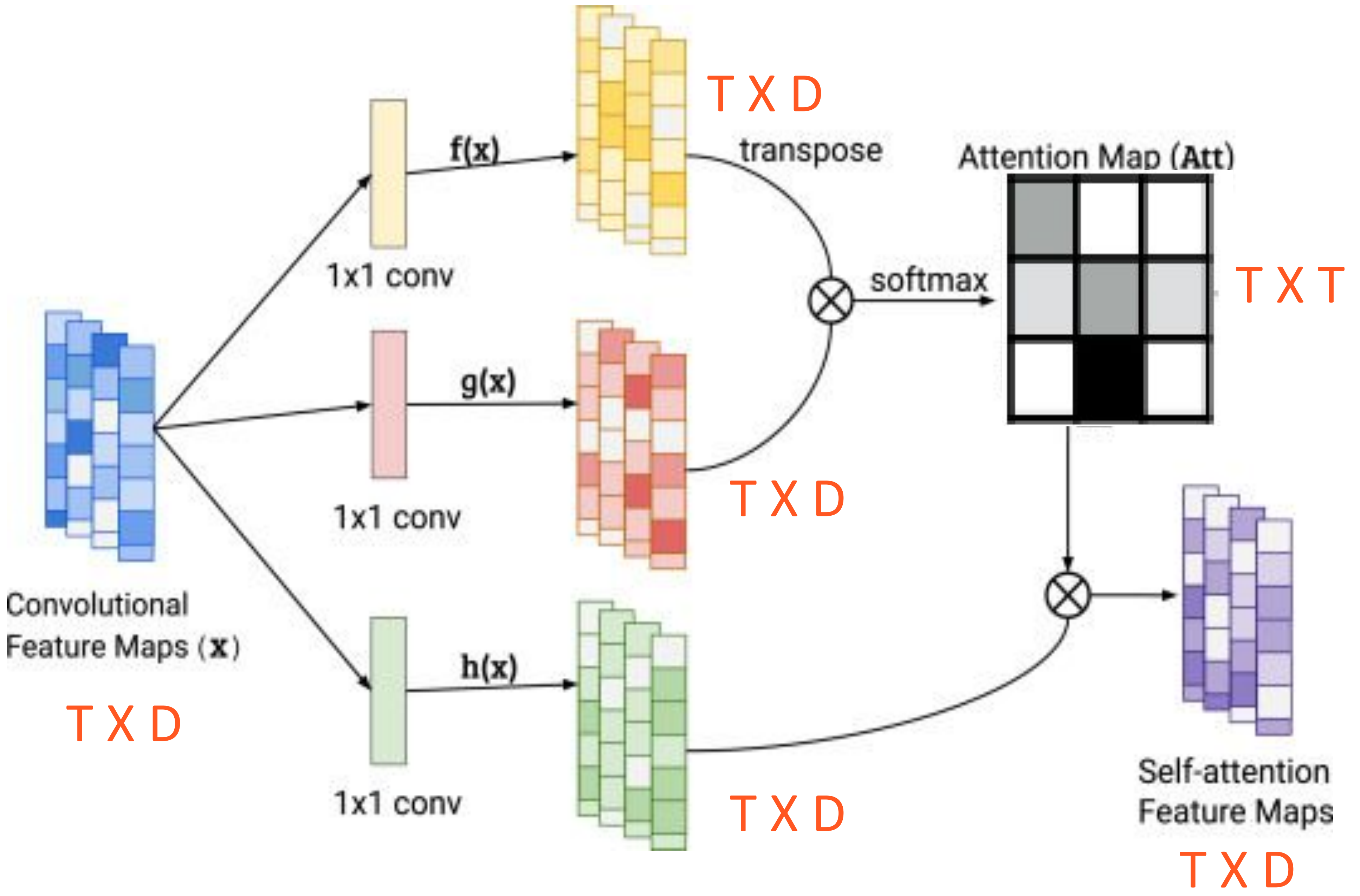
Self Attention::

- Each element attends to every other element. (or) Computes the correlation among the feature vectors in as sequence.
- Each feature vector becomes query, key, and value from the input embeddings by multiplying by a weight matrix.
- Self-attention can enable long-range temporal dependency modeling for action recognition.



Self Attention::

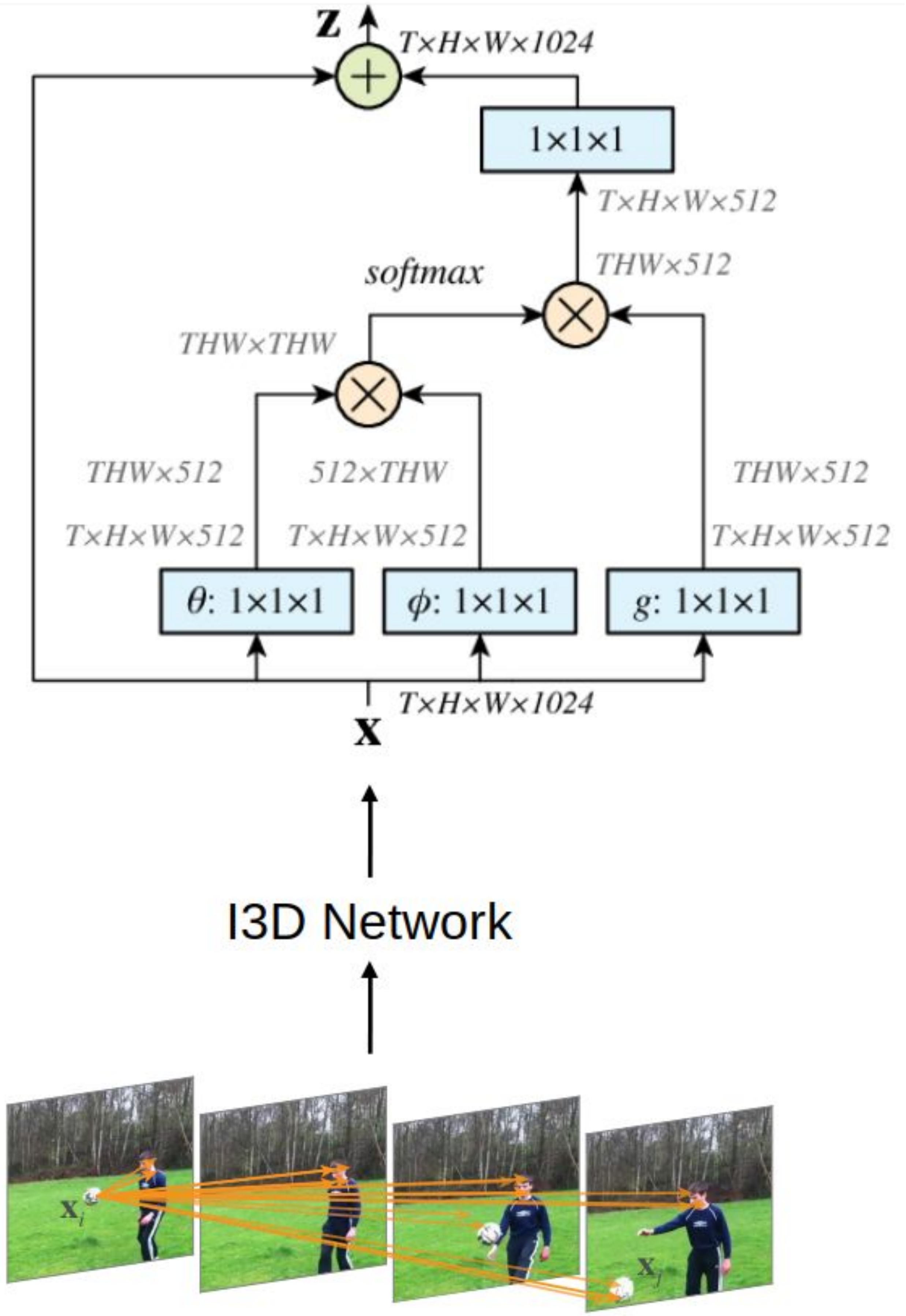
- **Goal:** To capture dependencies and relationships within input sequences.
- Each element attends to every other element. (or) Computes the correlation among the feature vectors in as sequence.
- **How it Works:**
  - It transforms the input sequence into three vectors: **query, key, and value.** 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::

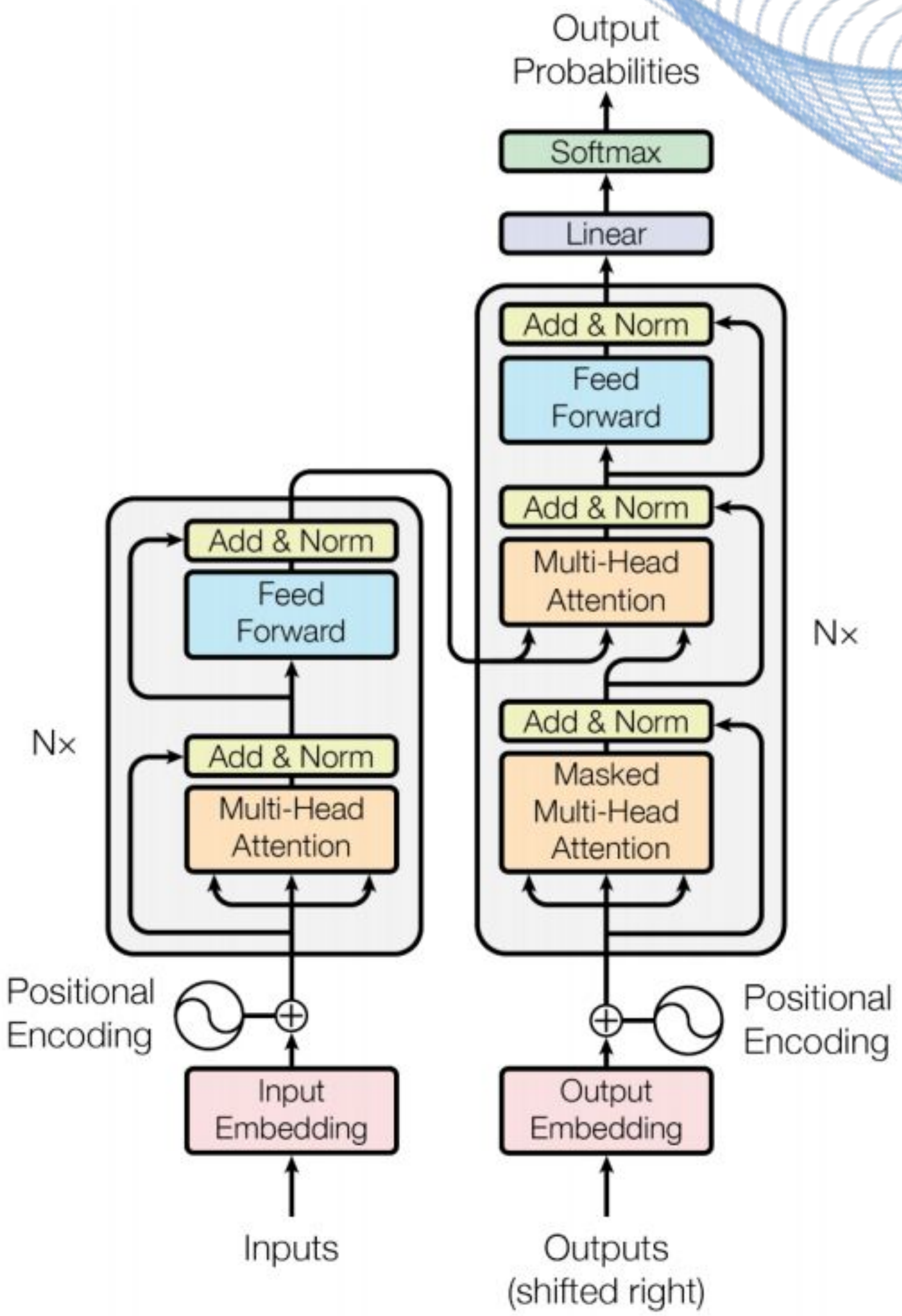
- **Benefits:**
  - **Long-range dependencies:** It allows the model to capture relationships between distant elements in a sequence, enabling it to understand complex patterns and dependencies.
  - **Contextual understanding:** By attending to different parts of the input sequence, self-attention helps 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::



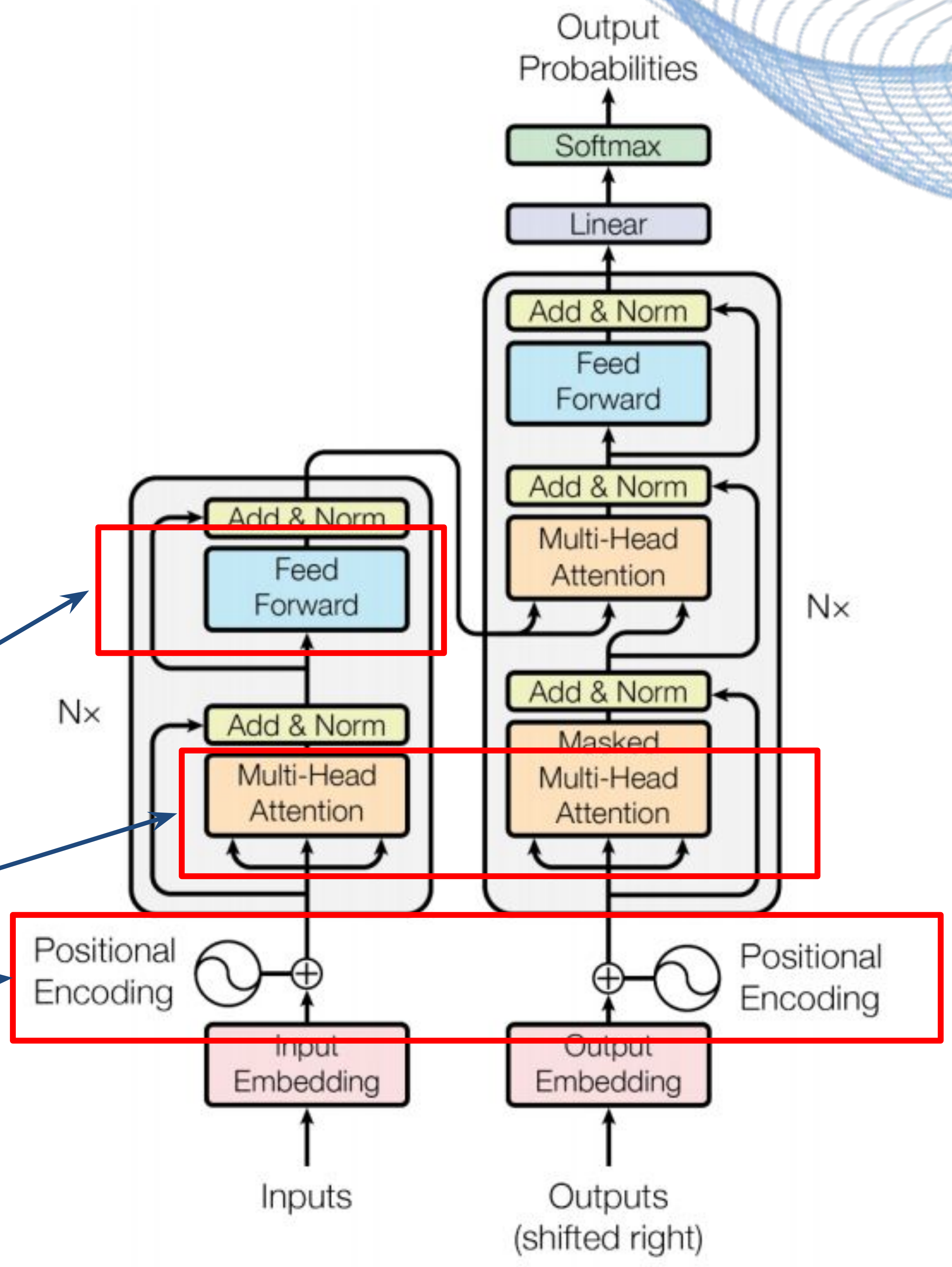
Transformer Models:

- **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



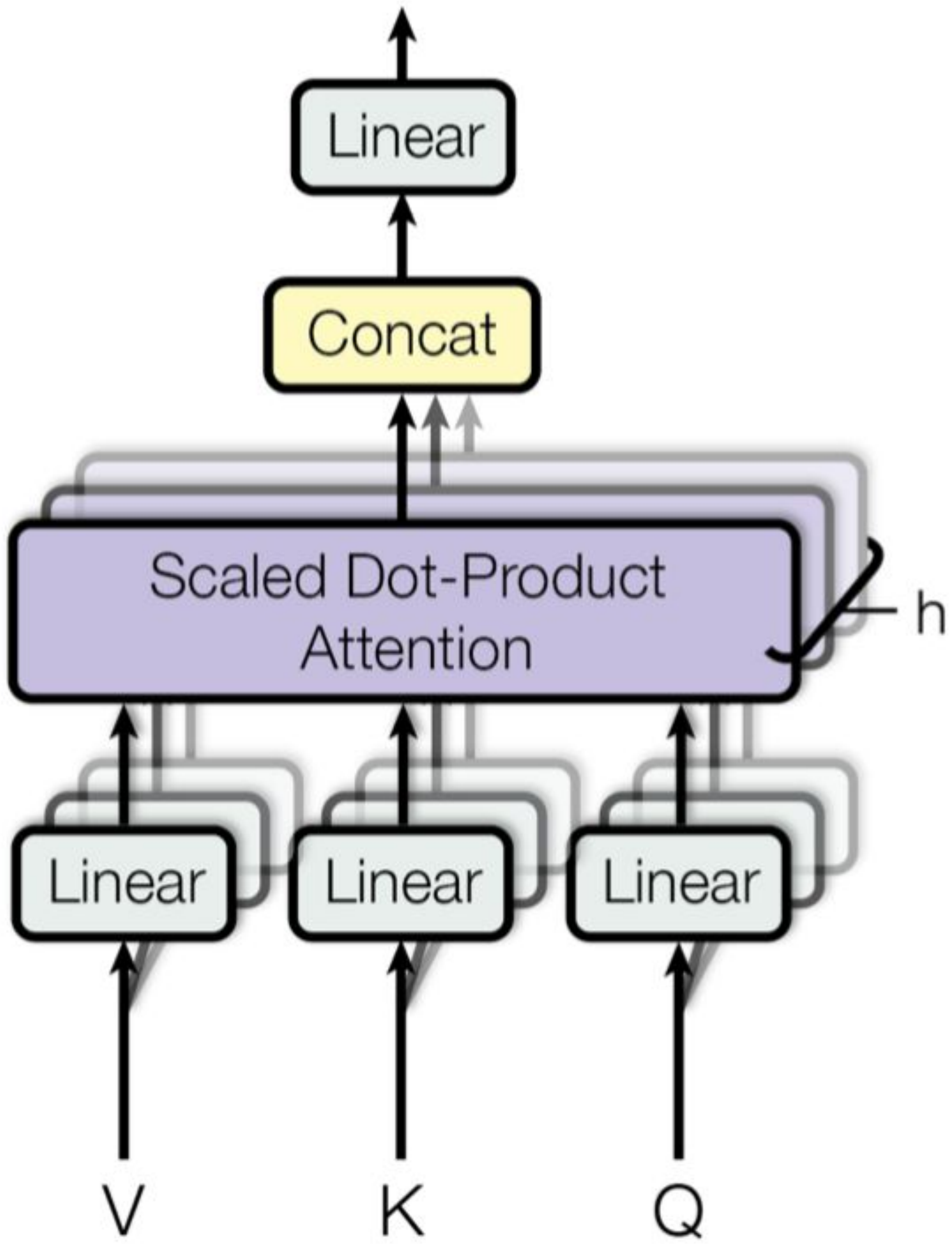
Transformer Models:

- **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

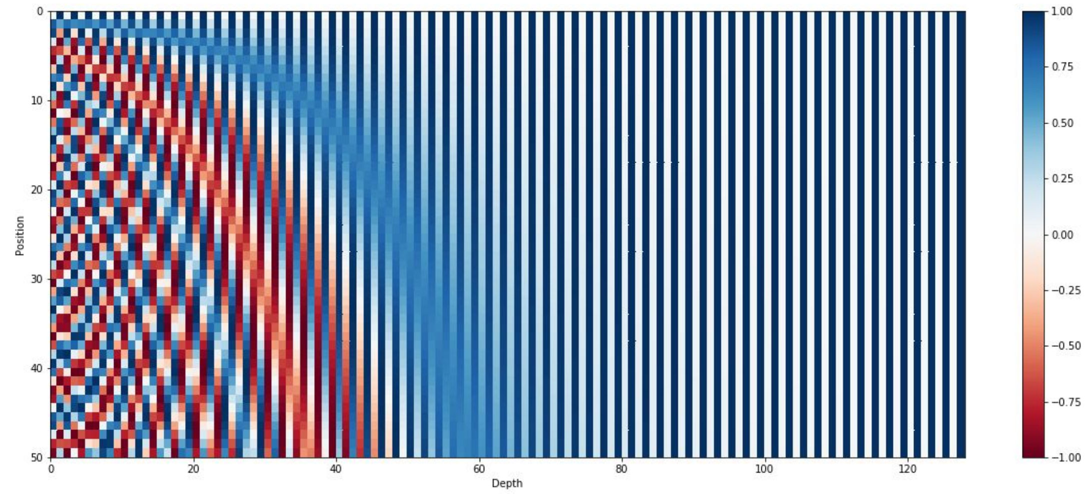


Transformer Models:

Self-Attention or Multi-Head Attention



Position Embedding



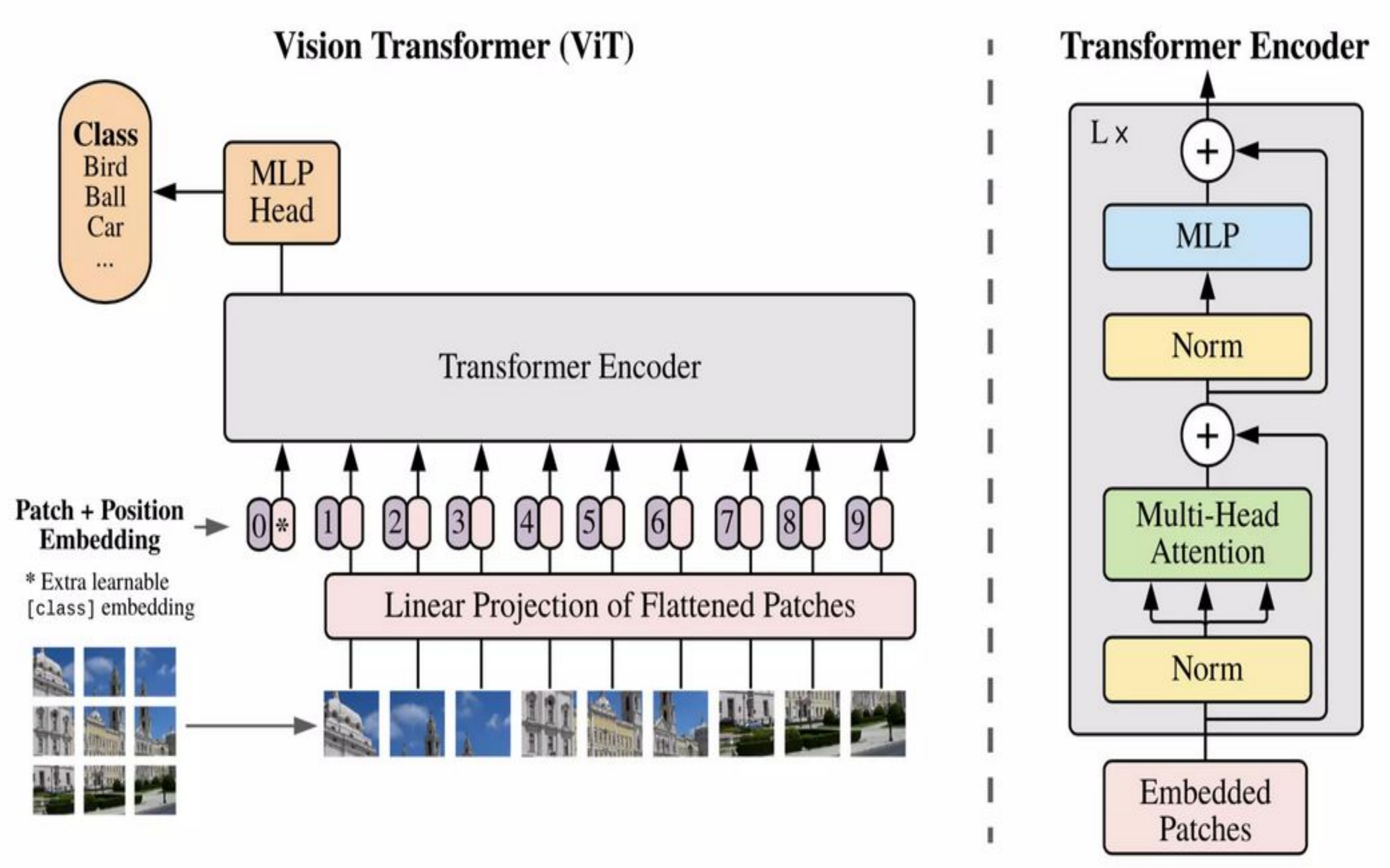
$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

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

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

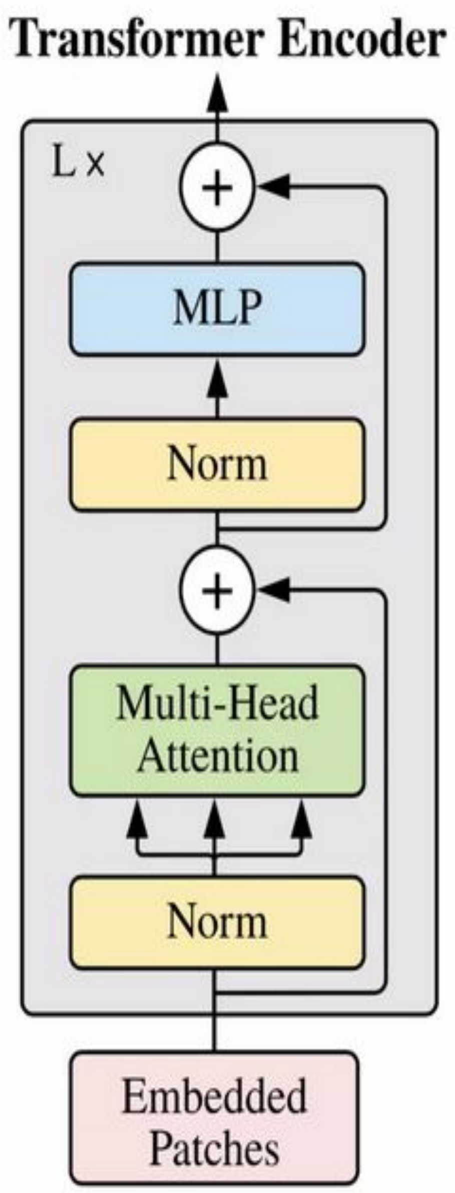
Vision Transformer (ViT):

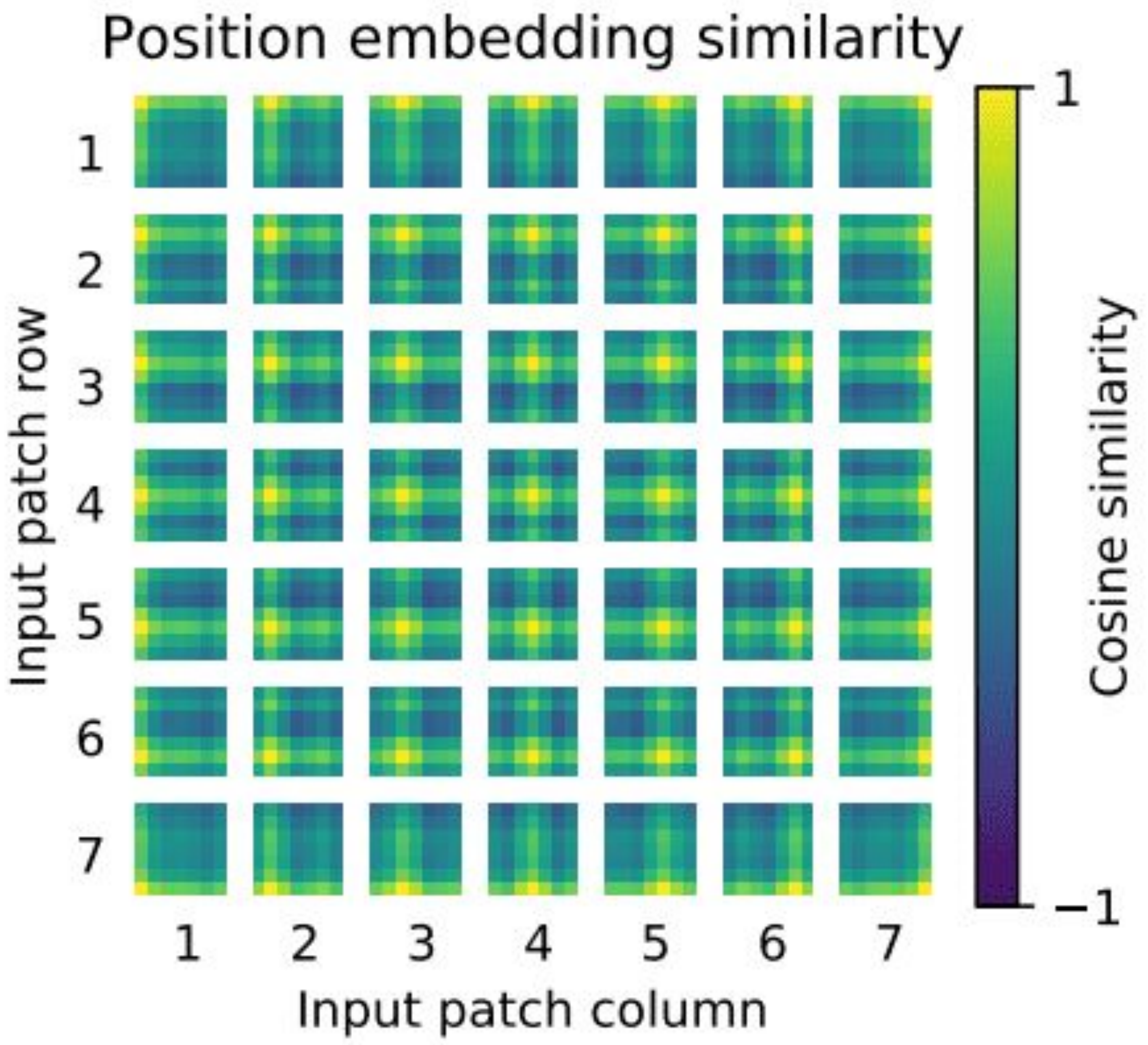
- In ViTs, images are represented as sequences, and class labels for the image are predicted, which allows models to learn image structure independently.
- **How ViT works?**
  - Split an image into patches (Tokenize)
  - Flatten the patches
  - Produce lower-dimensional linear embeddings from the flattened 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.

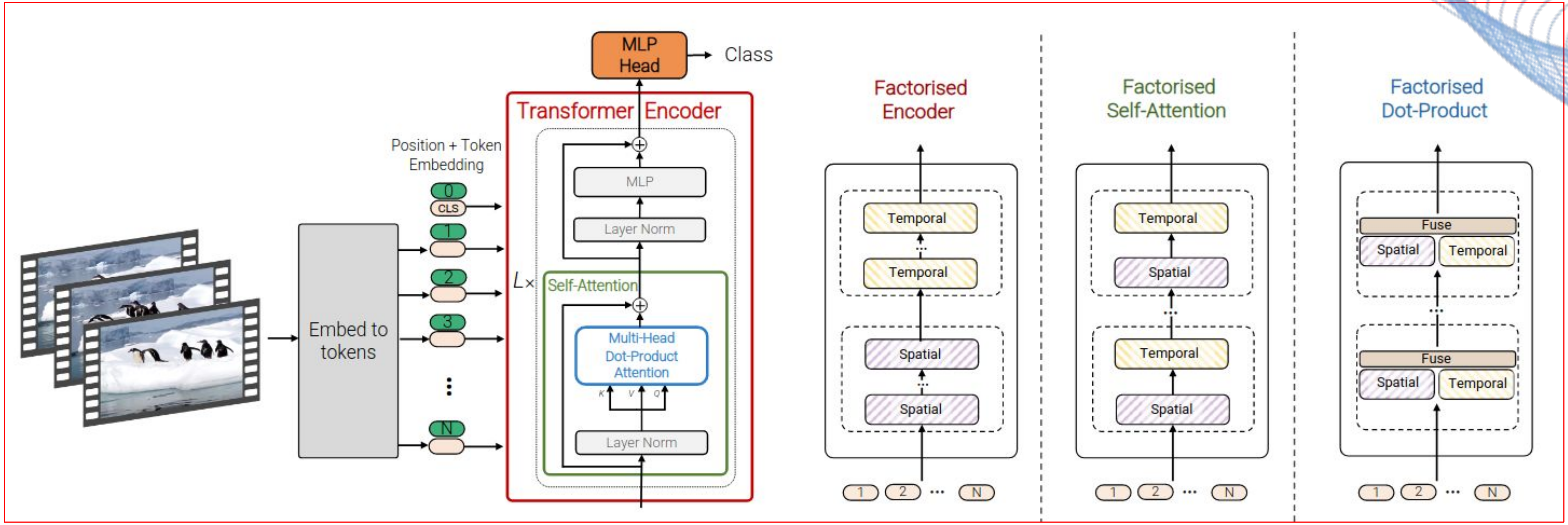
- Skip connections in ViT are even more influential than in CNNs (ResNet) and substantially impact the performance and similarity of representations.

- 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):



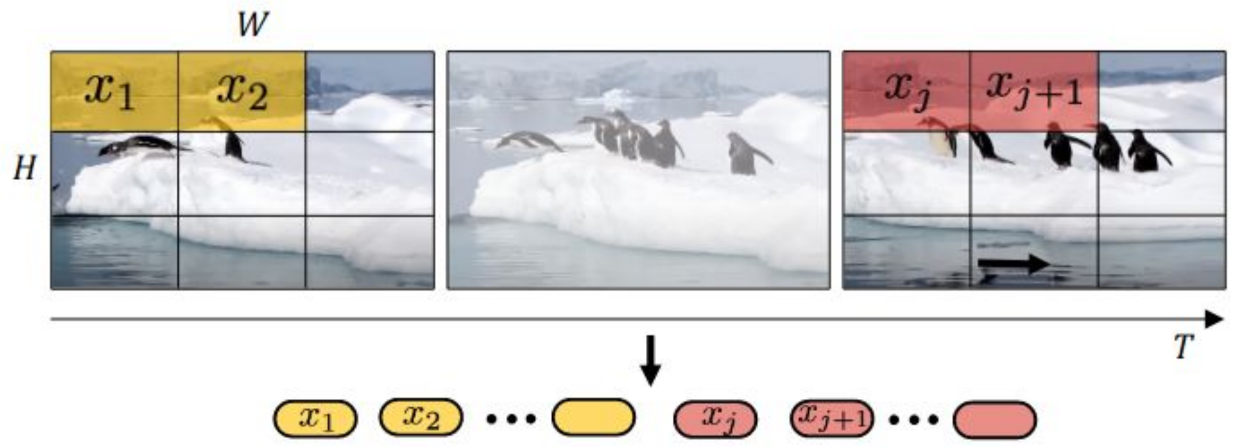


Figure 2: Uniform frame sampling: We simply sample n_t frames, and embed each 2D frame independently following ViT [18].

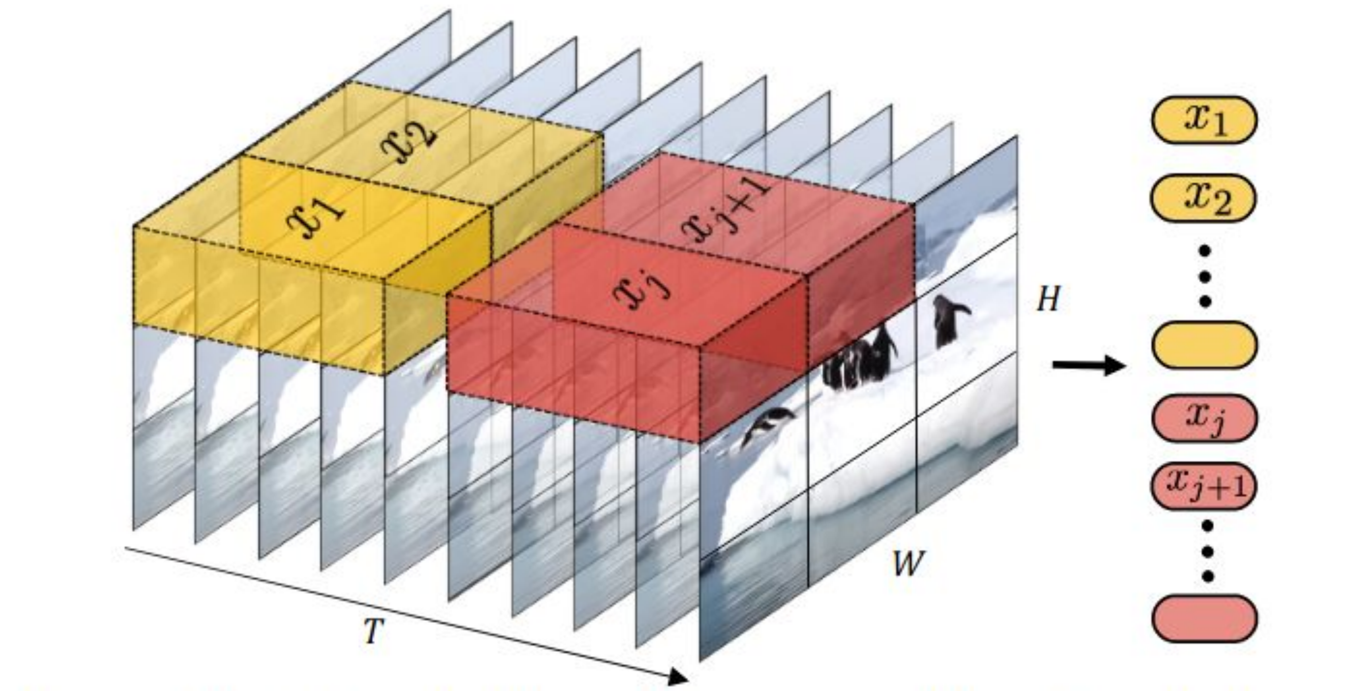
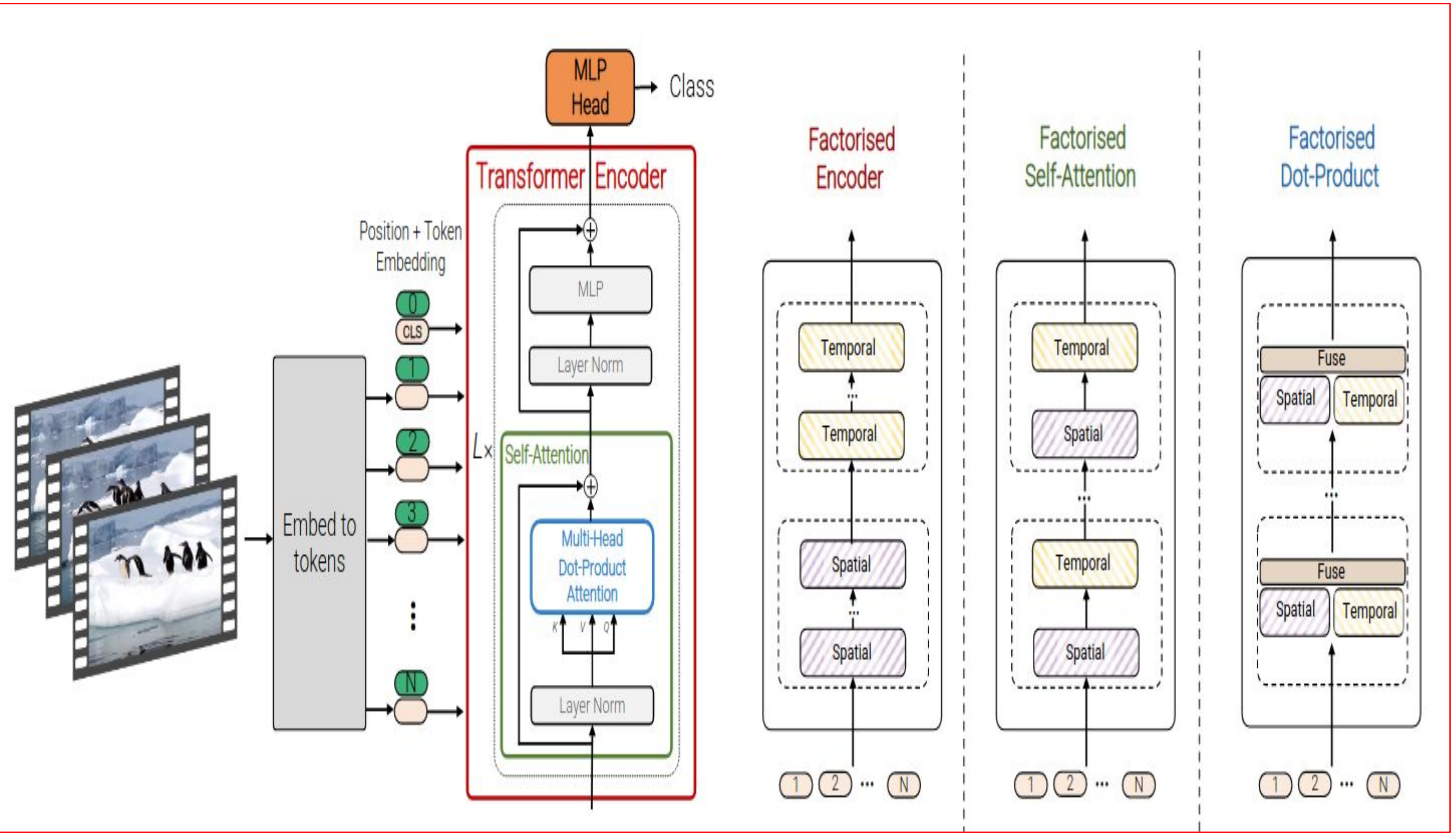


Figure 3: Tubelet embedding. We extract and linearly embed non-overlapping tubelets that span the spatio-temporal input volume.

A Video Vision Transformer (ViViT):



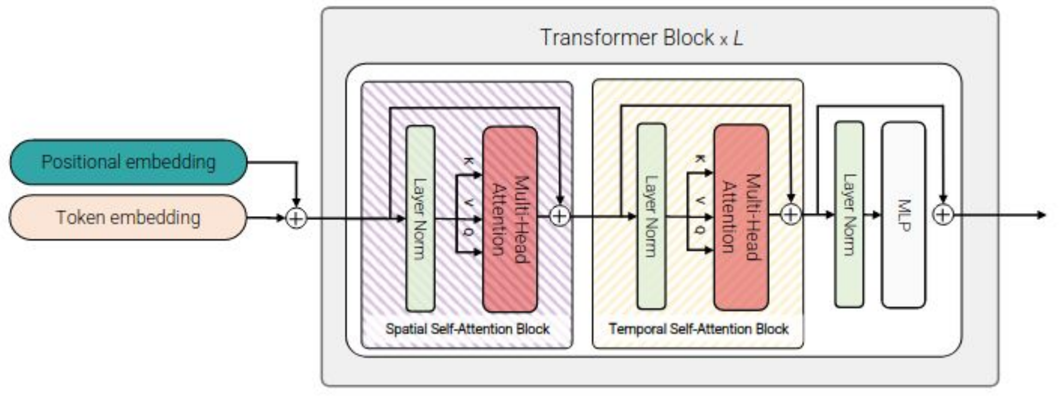


Figure 5: Factorised self-attention (Model 3). Within each transformer block, the multi-headed self-attention operation is factorised into two operations (indicated by striped boxes) that first only compute self-attention spatially, and then temporally.

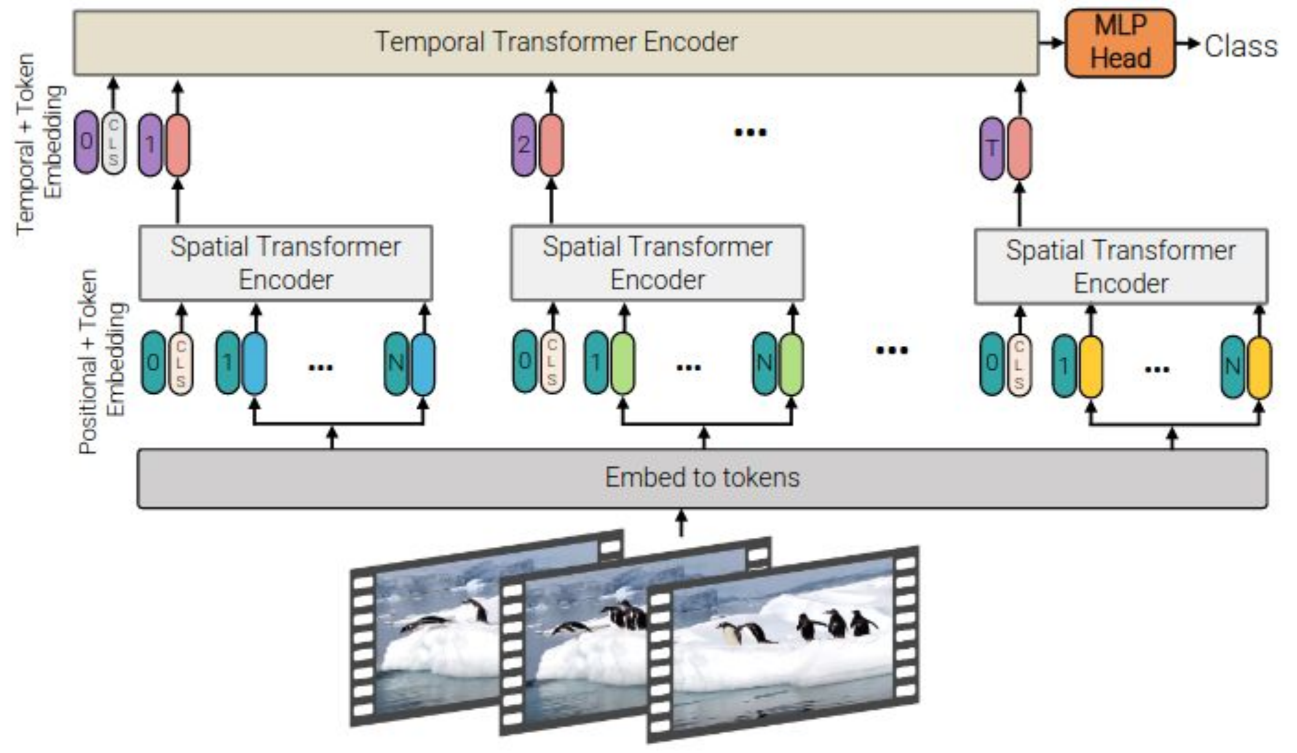
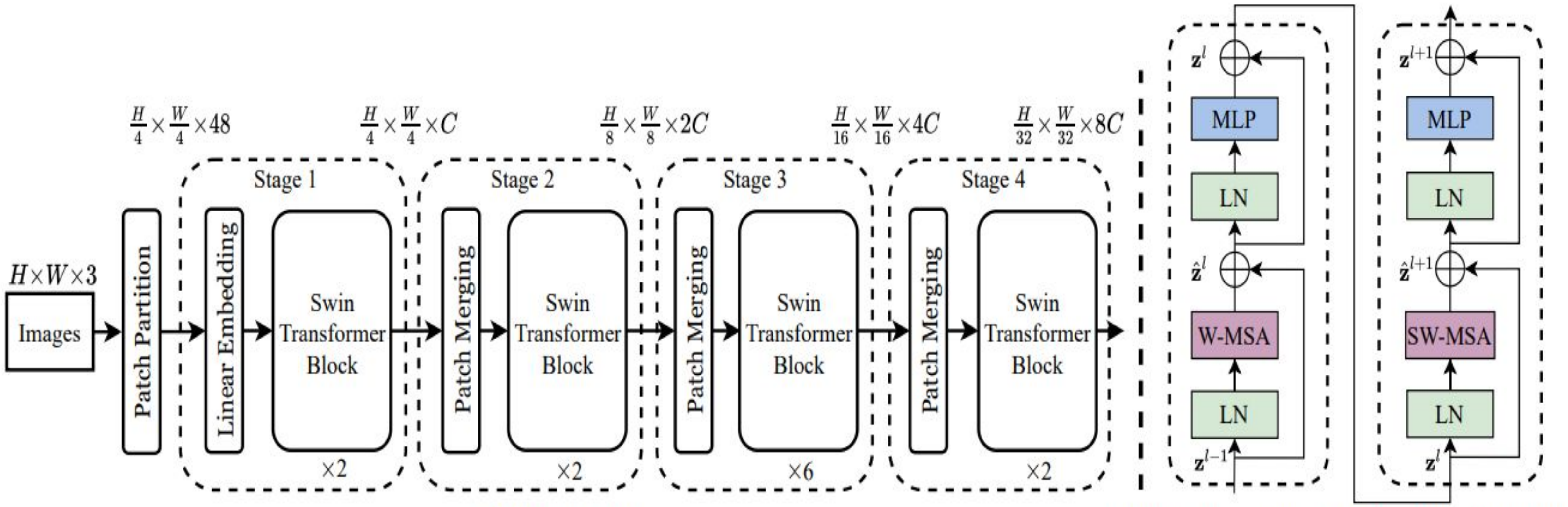

Temporal Transformer Encoder
MLP Head
Class
Temporal + Token Embedding
0
CLS
1
2
...
T
Spatial Transformer Encoder
Spatial Transformer Encoder
Spatial Transformer Encoder
Positional + Token Embedding
0
CLS
1
...
N
Embed to tokens

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.

Swin Transformer :

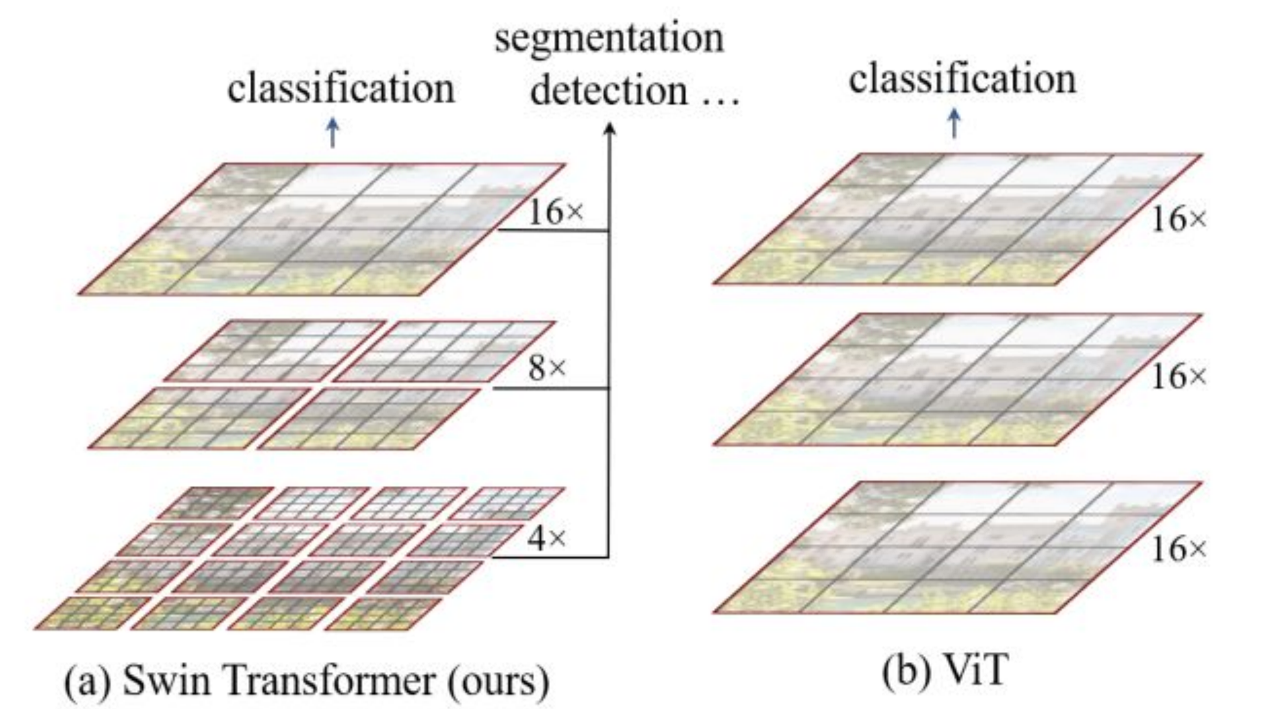
- Swin Transformer **builds hierarchical feature maps by merging image patches** in deeper layers compared to **ViTs that produces feature maps of a single low resolution**.
- It is enabled by **shifted window** to build hierarchical feature maps.



(a) Architecture

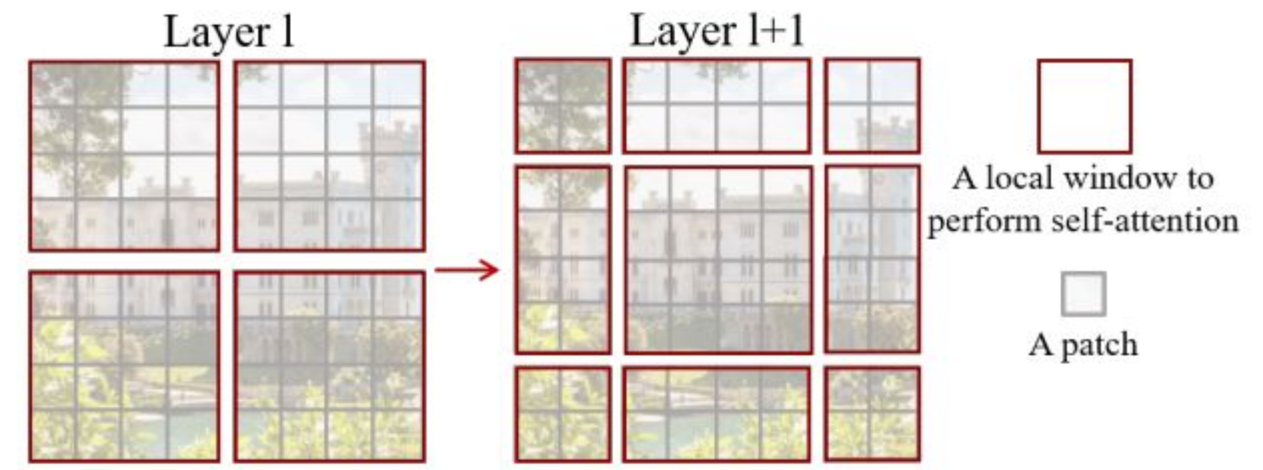
(b) Two Successive Swin Transformer Blocks

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.



(a) Swin Transformer (ours)

(b) ViT



Shifted Window

Video Swin Transformer :

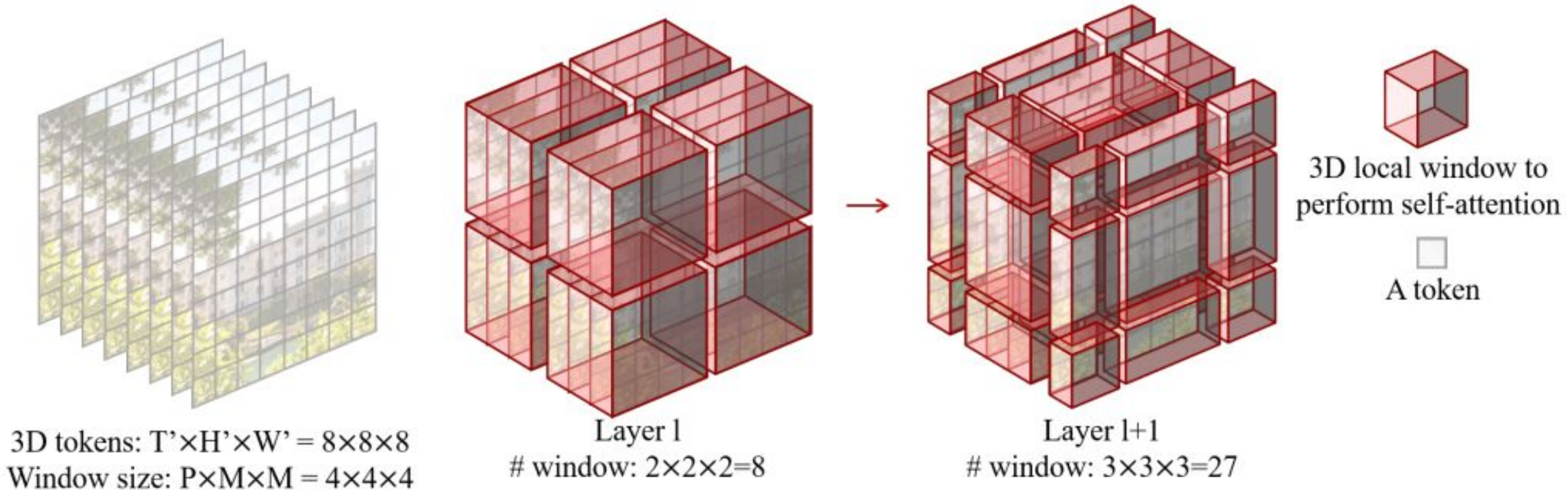

3D local window to perform self-attention
A token
3D tokens: T'×H'×W' = 8×8×8
Window size: P×M×M = 4×4×4
Layer l
# window: 2×2×2=8
Layer l+1
# window: 3×3×3=27

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.

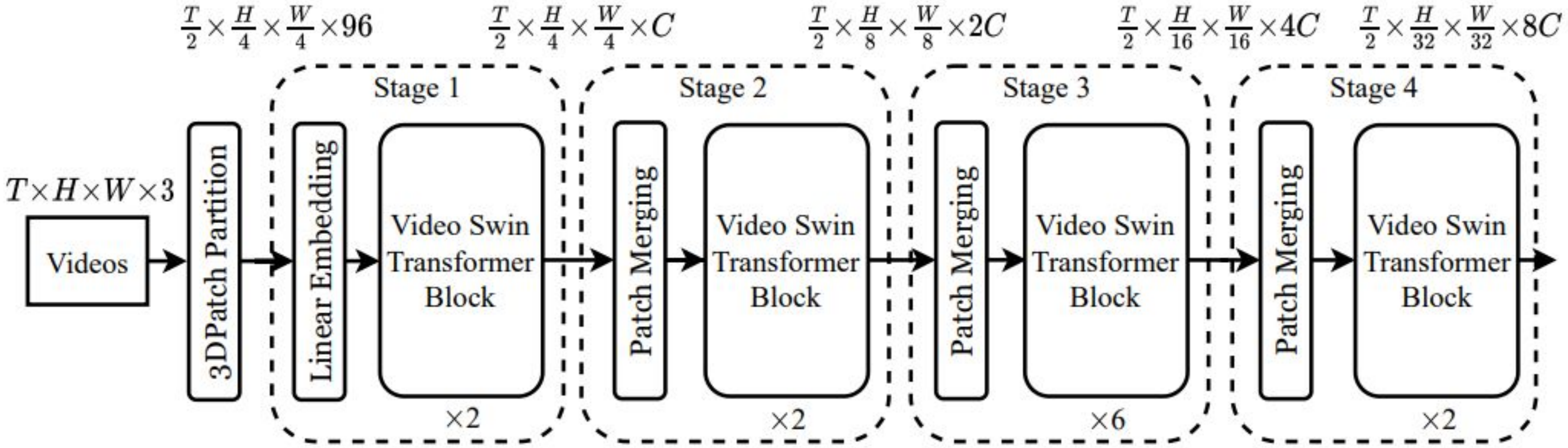

$\frac{T}{2} \times \frac{H}{4} \times \frac{W}{4} \times 96$
$\frac{T}{2} \times \frac{H}{4} \times \frac{W}{4} \times C$
$\frac{T}{2} \times \frac{H}{8} \times \frac{W}{8} \times 2C$
$\frac{T}{2} \times \frac{H}{16} \times \frac{W}{16} \times 4C$
$\frac{T}{2} \times \frac{H}{32} \times \frac{W}{32} \times 8C$
$T \times H \times W \times 3$
Videos
3DPatch Partition
Stage 1
Linear Embedding
Video Swin Transformer Block
×2
Stage 2
Patch Merging
Video Swin Transformer Block
×2
Stage 3
Patch Merging
Video Swin Transformer Block
×6
Stage 4
Patch Merging
Video Swin Transformer Block
×2

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

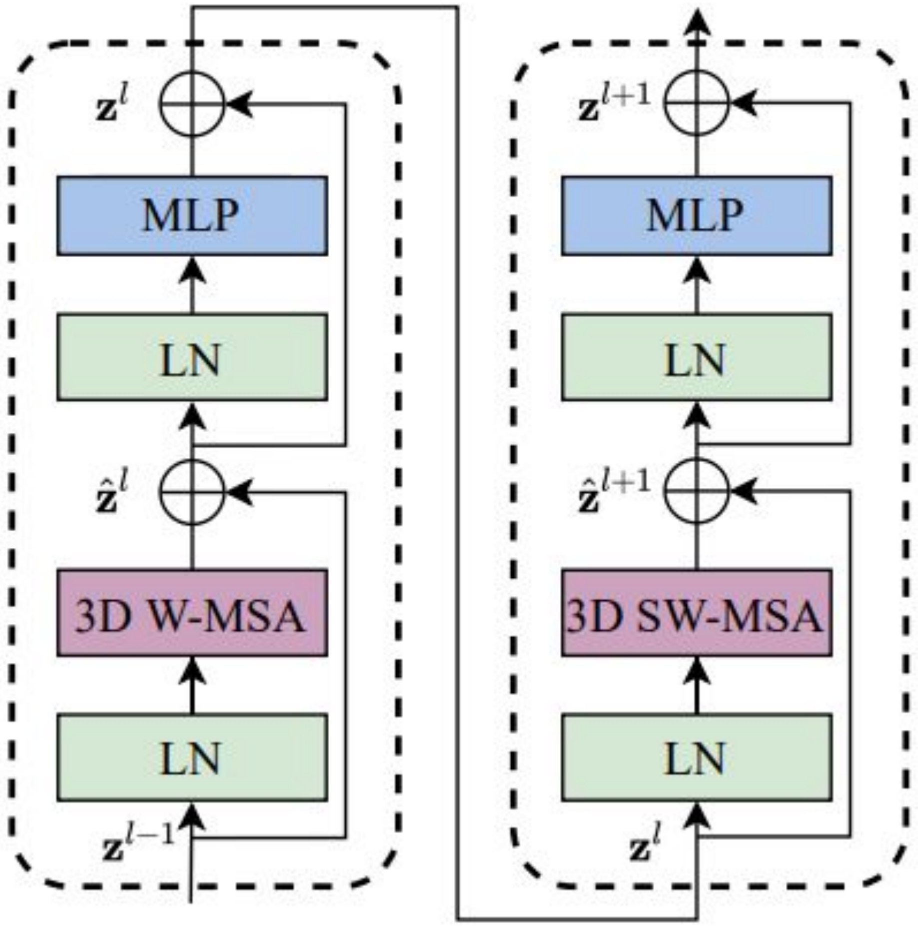
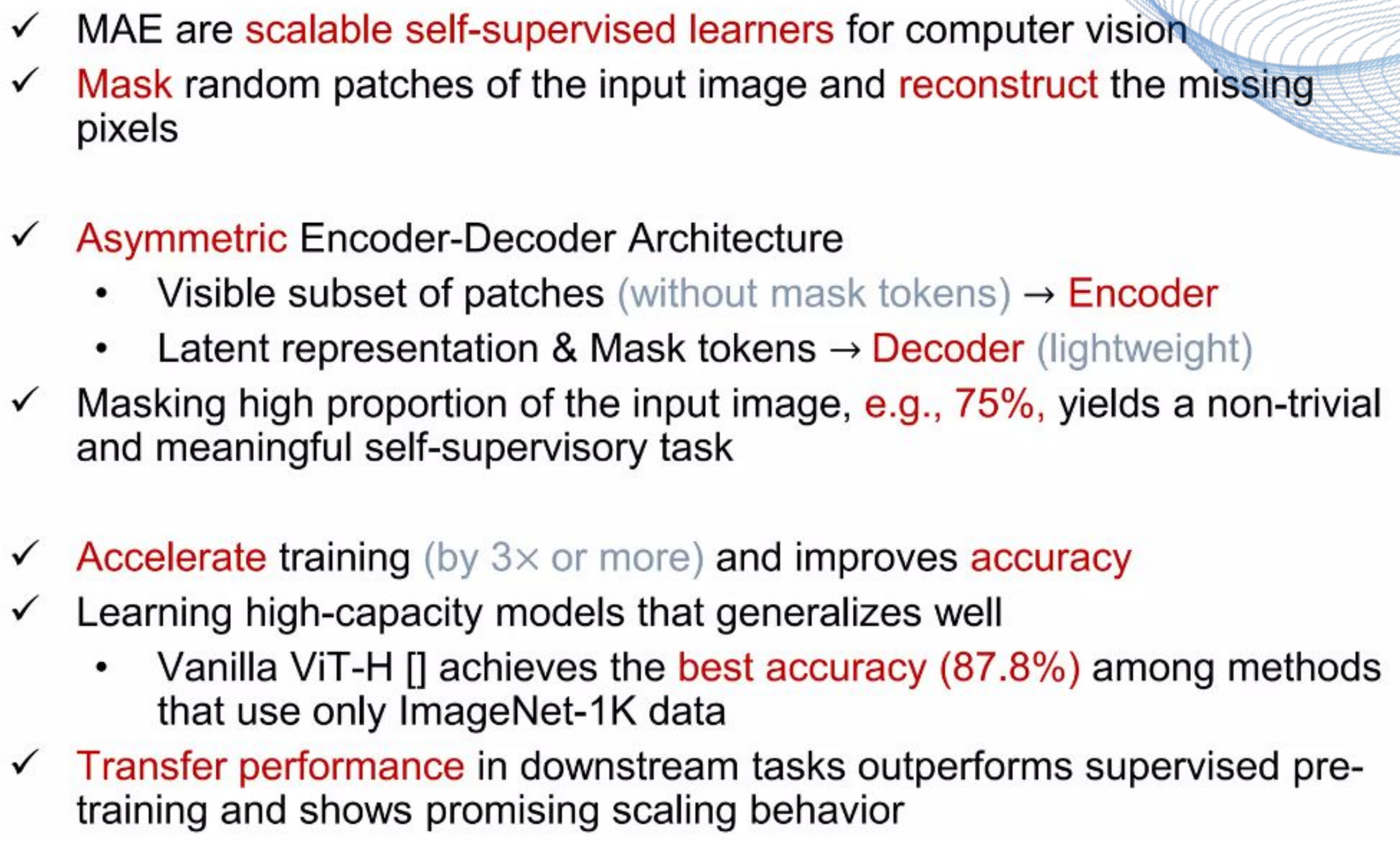


Figure 2: An illustration of two successive Video Swin Transformer blocks.

Masked Auto Encoder (MAE):

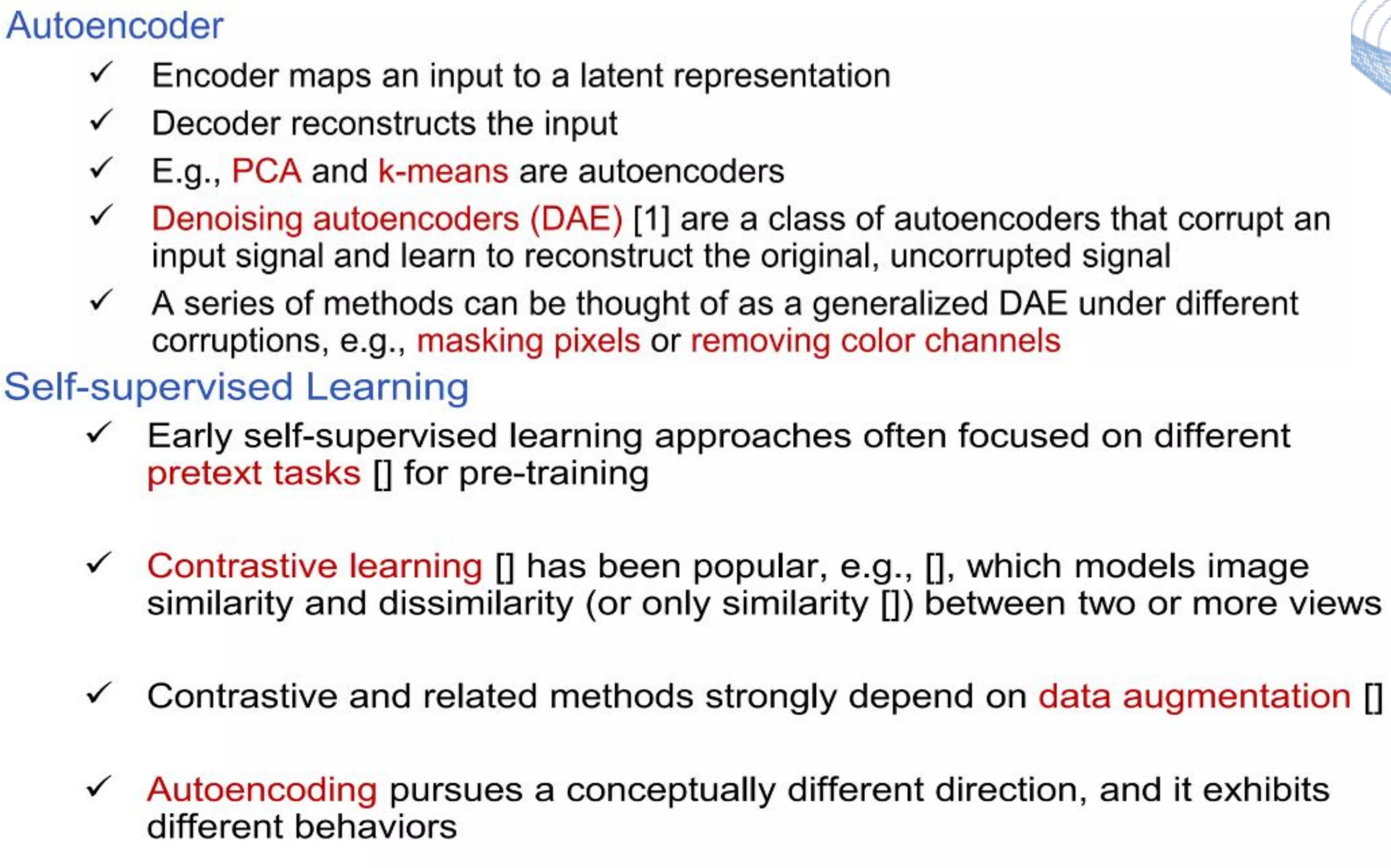


- ✓ MAE are scalable self-supervised learners for computer vision
- ✓ Mask random patches of the input image and reconstruct the missing pixels

- ✓ Asymmetric Encoder-Decoder Architecture
  - Visible subset of patches (without mask tokens) → Encoder
  - Latent representation & Mask tokens → Decoder (lightweight)
- ✓ Masking high proportion of the input image, e.g., 75%, yields a non-trivial and meaningful self-supervisory task

- ✓ Accelerate training (by 3× or more) and improves accuracy
- ✓ Learning high-capacity models that generalizes well
  - Vanilla ViT-H [] achieves the best accuracy (87.8%) among methods that use only ImageNet-1K data
- ✓ Transfer performance in downstream tasks outperforms supervised pre-training and shows promising scaling behavior

Background of MAE::



Autoencoder

- ✓ Encoder maps an input to a latent representation
- ✓ Decoder reconstructs the input
- ✓ E.g., PCA and k-means are autoencoders
- ✓ Denoising autoencoders (DAE) [1] are a class of autoencoders that corrupt an input signal and learn to reconstruct the original, uncorrupted signal
- ✓ A series of methods can be thought of as a generalized DAE under different corruptions, e.g., masking pixels or removing color channels

Self-supervised Learning

- ✓ Early self-supervised learning approaches often focused on different pretext tasks [] for pre-training
- ✓ Contrastive learning [] has been popular, e.g., [], which models image similarity and dissimilarity (or only similarity []) between two or more views
- ✓ Contrastive and related methods strongly depend on data augmentation []
- ✓ Autoencoding pursues a conceptually different direction, and it exhibits different behaviors

MAE::

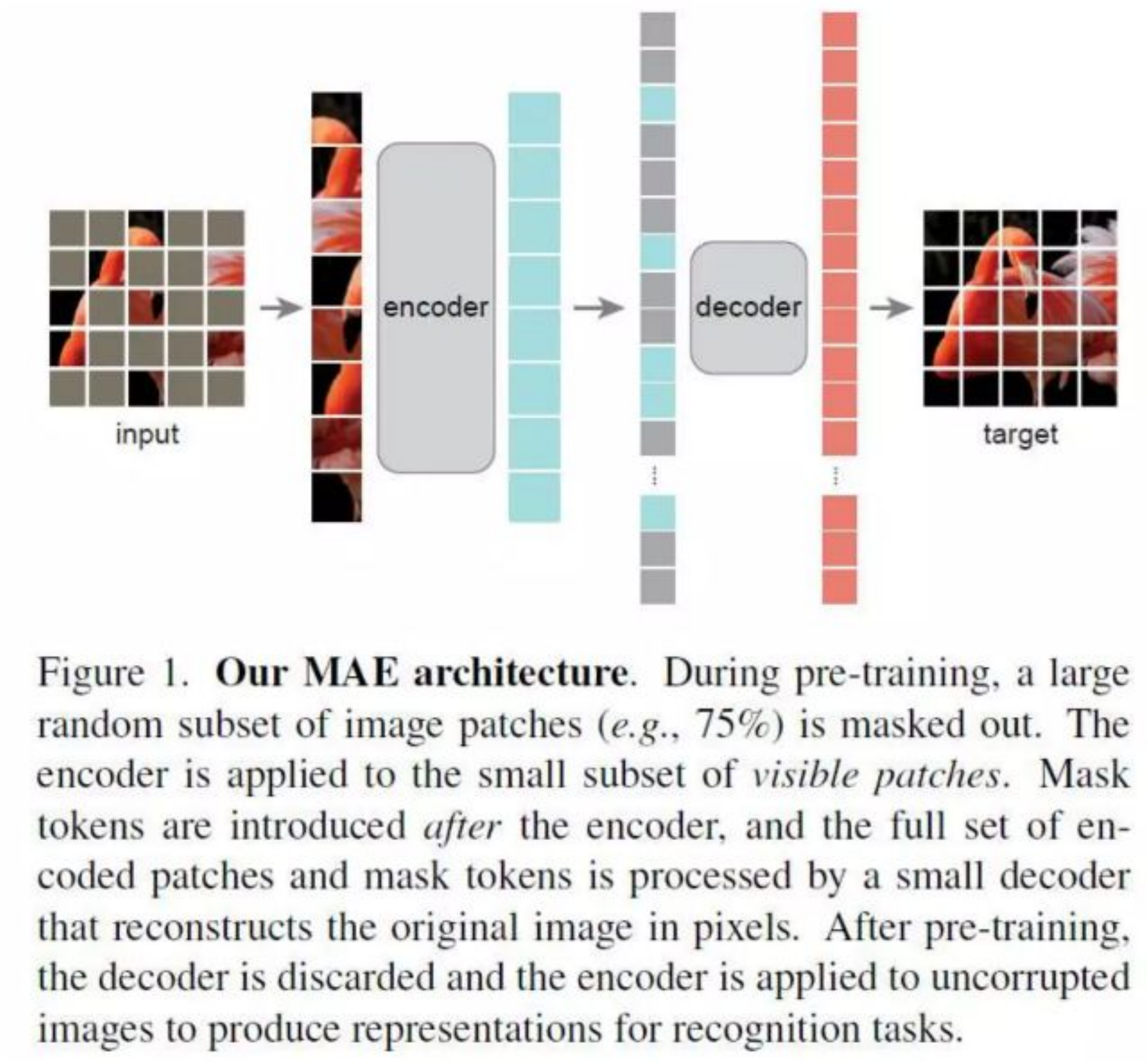


Figure 1. **Our MAE architecture**. During pre-training, a large random subset of image patches (*e.g.*, 75%) is masked out. The encoder is applied to the small subset of *visible patches*. Mask tokens are introduced *after* the encoder, and the full set of encoded patches and mask tokens is processed by a small decoder that reconstructs the original image in pixels. After pre-training, the decoder is discarded and the encoder is applied to uncorrupted images to produce representations for recognition tasks.

VideoMAE::

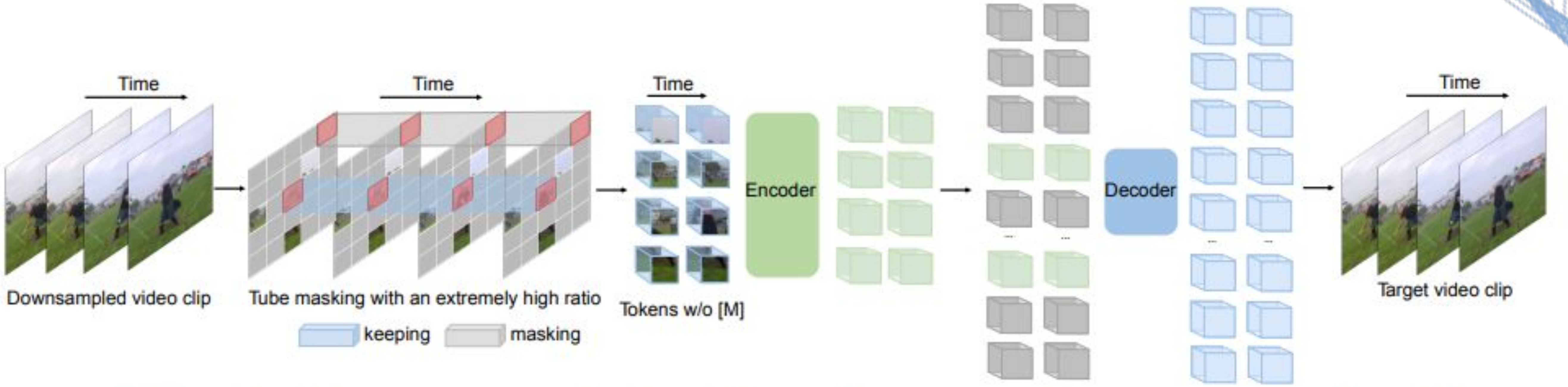
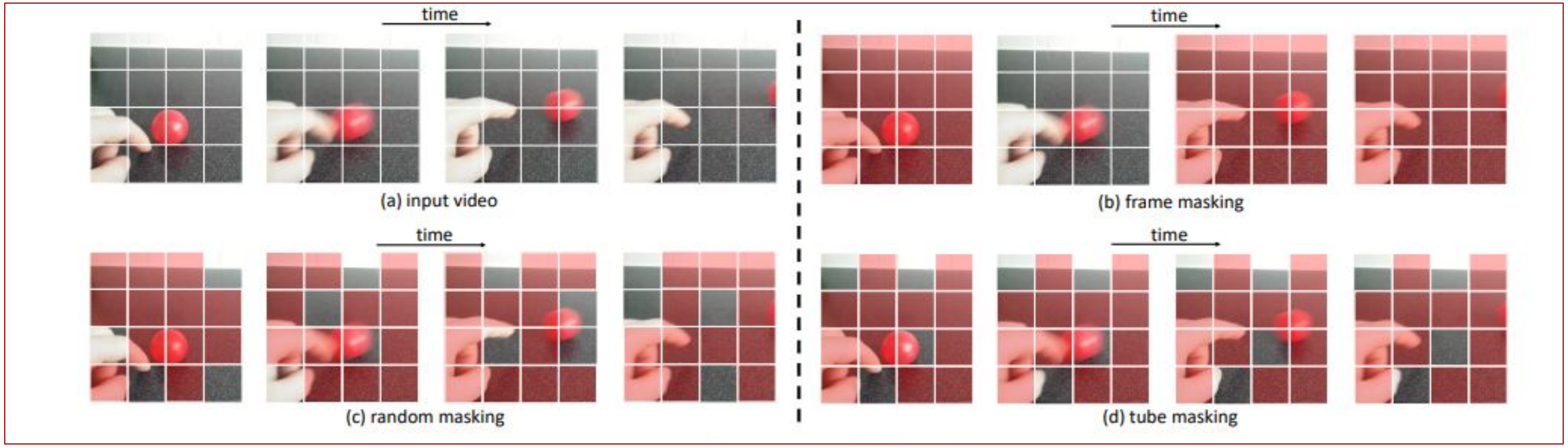


Figure 1: **VideoMAE** performs the task of masking random cubes and reconstructing the missing ones with an asymmetric encoder-decoder architecture. Due to high redundancy and temporal correlation in videos, we present the customized design of tube masking with an extremely high ratio (90% to 95%). This simple design enables us to create a more challenging and meaningful self-supervised task to make the learned representations capture more useful spatiotemporal structures.



(a) input video

(b) frame masking

(c) random masking

(d) tube masking

CLIP:: Contrastive Language-Image Pre-training

Background of Image-Text Pair

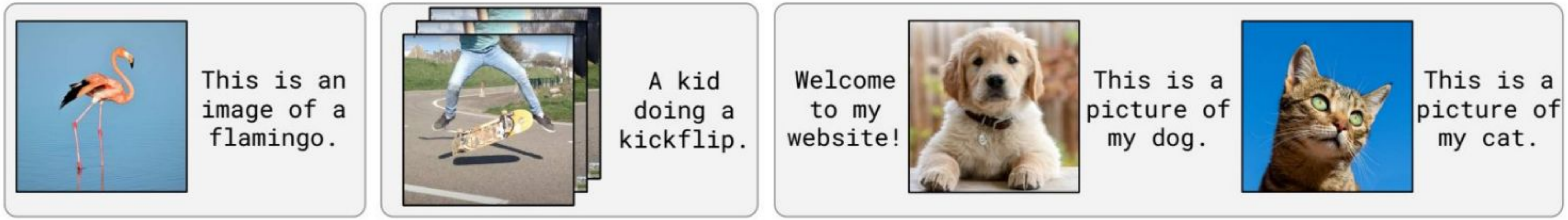

This is an image of a flamingo.
A kid doing a kickflip.
Welcome to my website!
This is a picture of my dog.
This is a picture of my cat.

Image-Text Pairs dataset
[N=1, T=1, H, W, C]

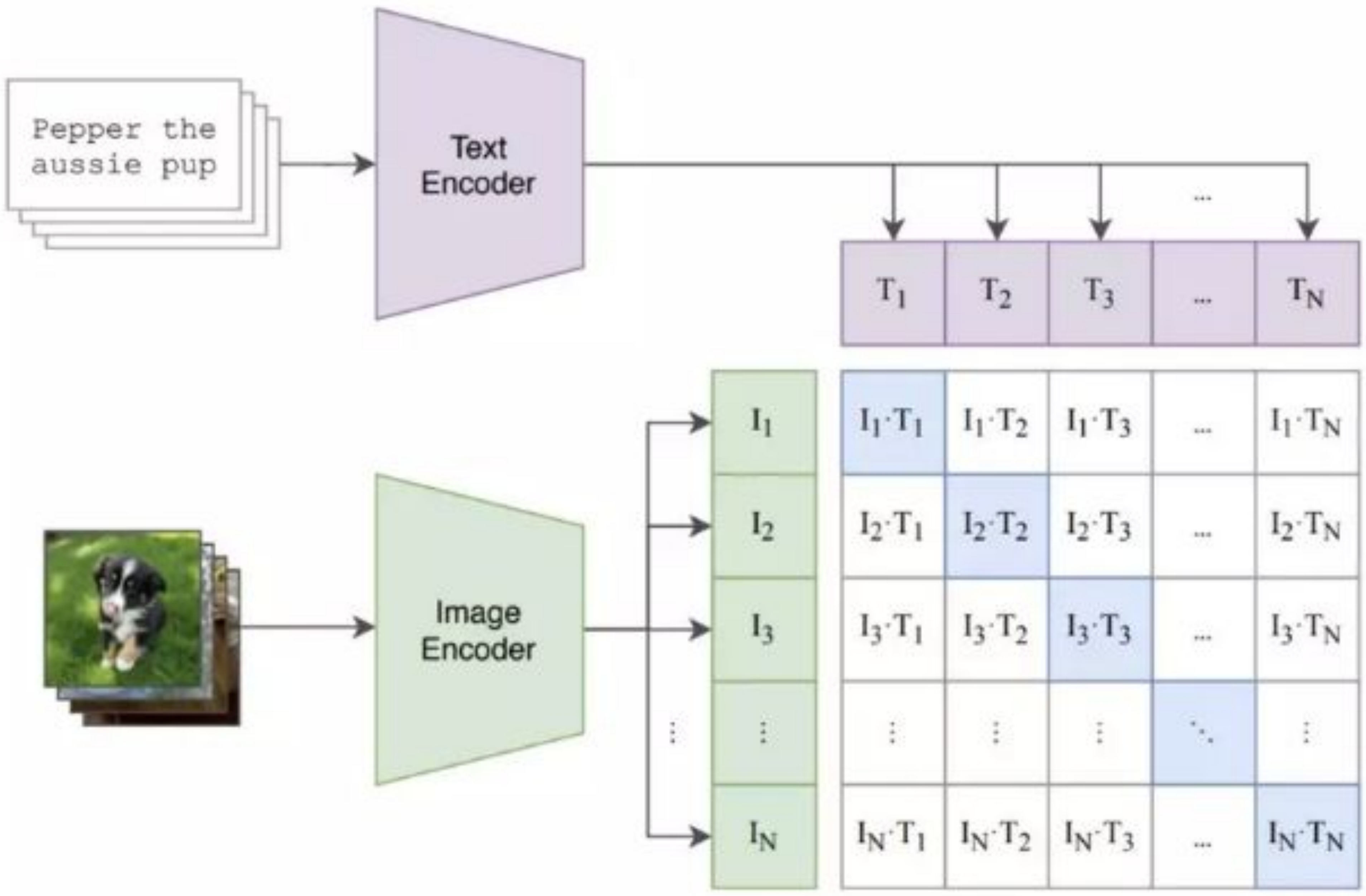
Video-Text Pairs dataset
[N=1, T>1, H, W, C]

Multi-Modal Massive Web (M3W) dataset
[N>1, T=1, H, W, C]

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

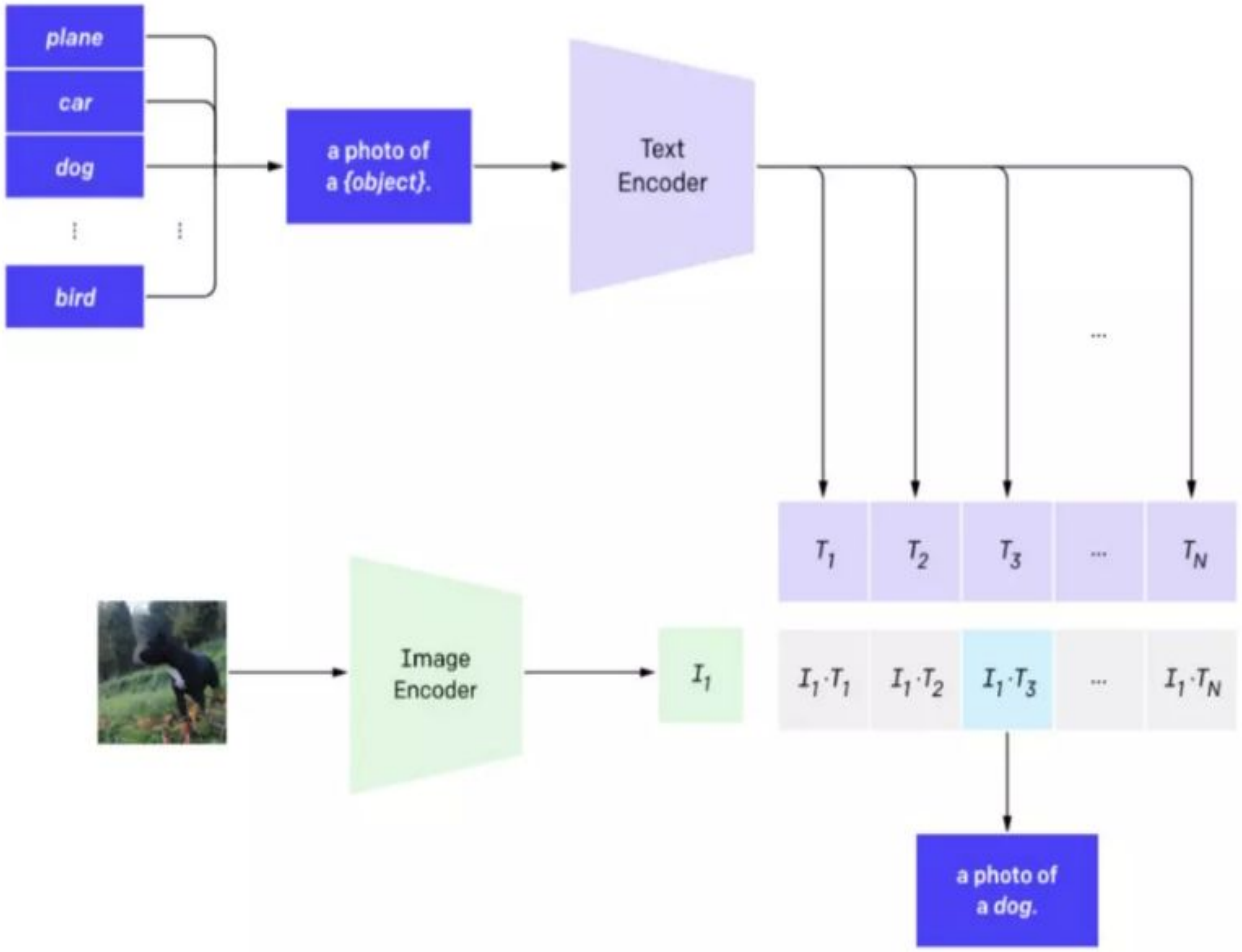
CLIP:: Contrastive Language-Image Pre-training

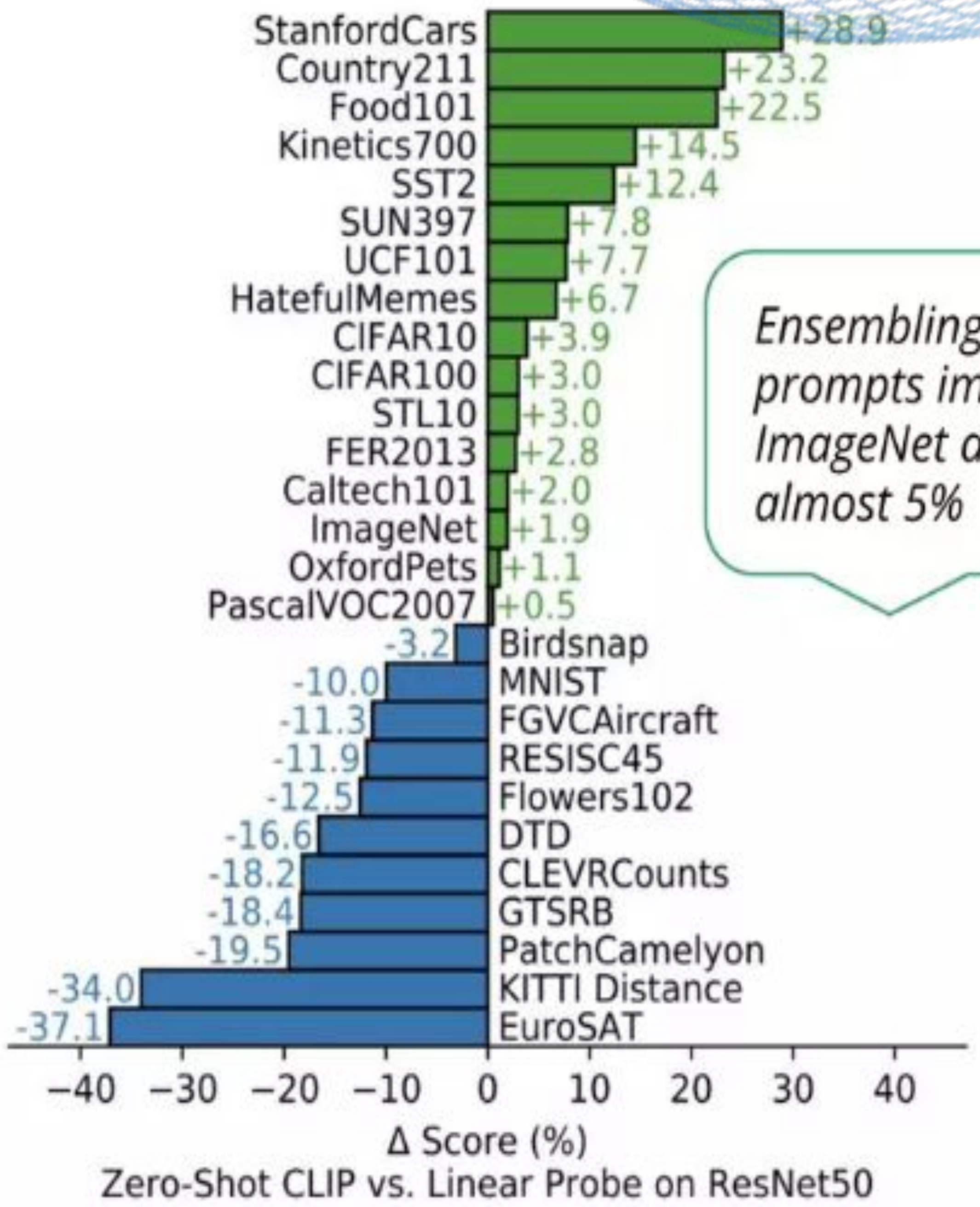
(1) Contrastive pre-training


Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ⋮ IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
⋮ ⋮ ⋮ ⋱ ⋮
IN·T1 IN·T2 IN·T3 ... IN·TN

- **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



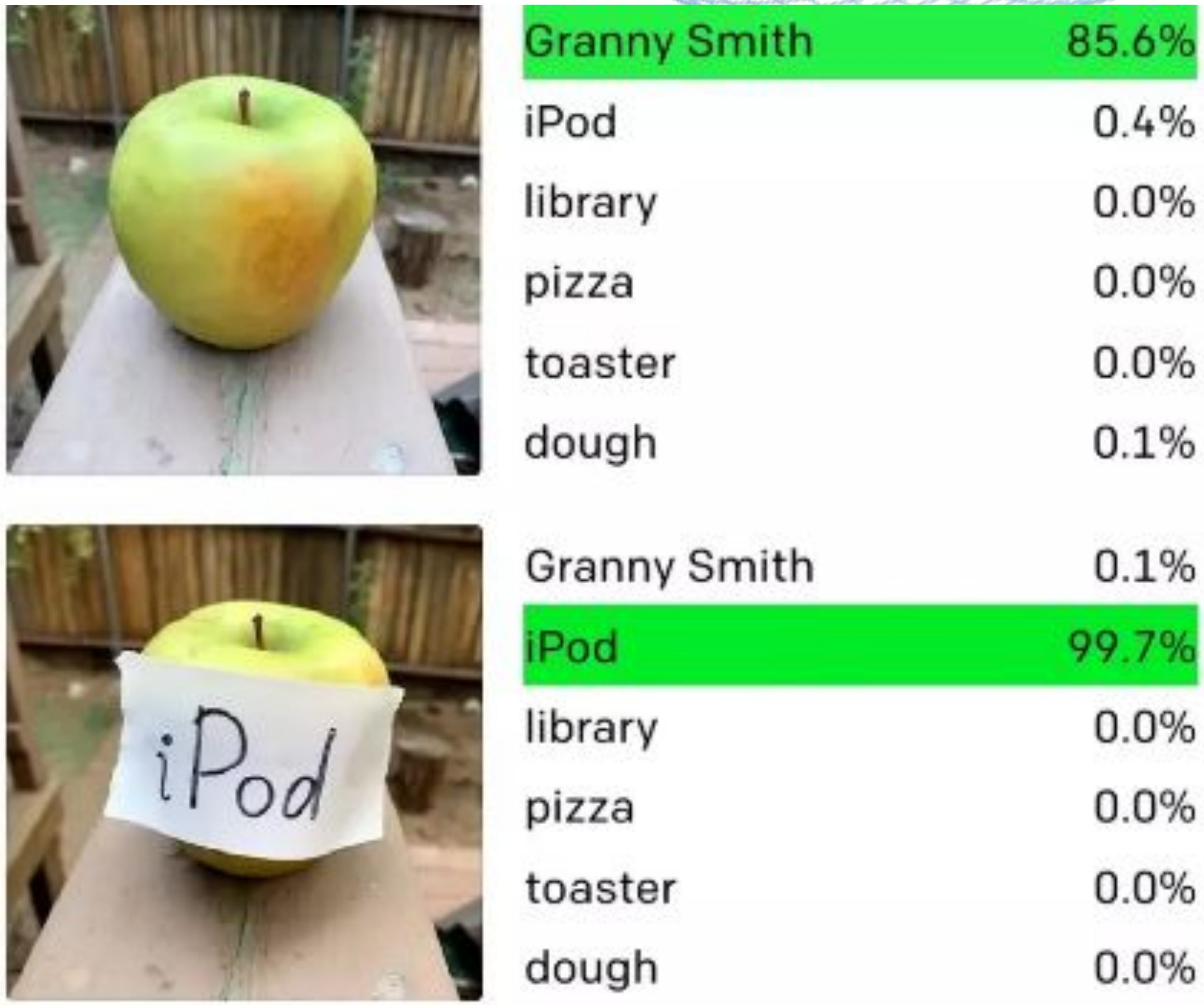


Ensembling around 80 prompts improve ImageNet accuracy by almost 5%

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.

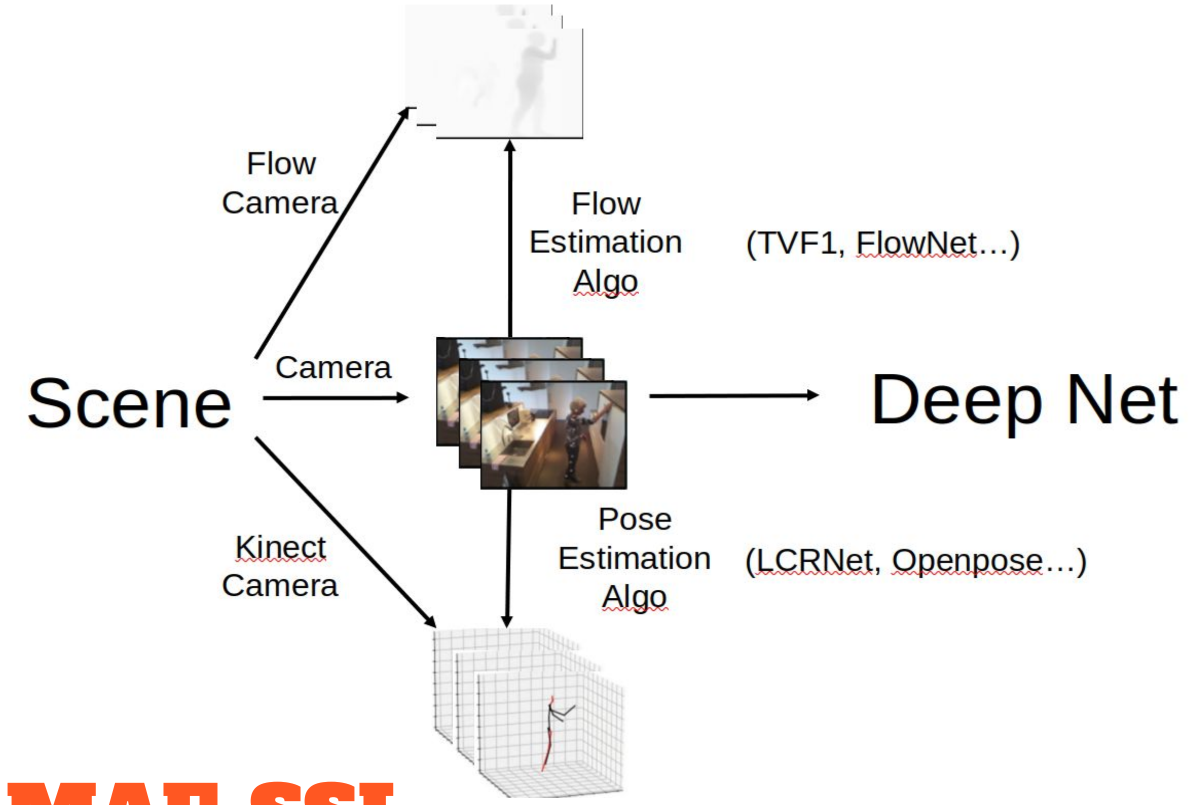


Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

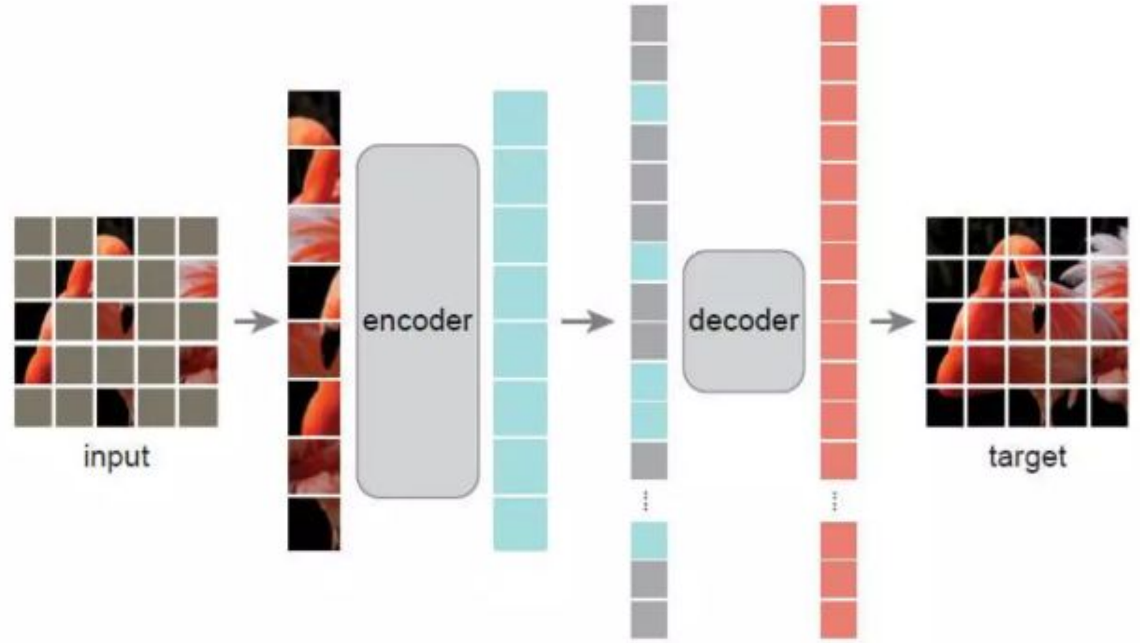
Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

Summary::

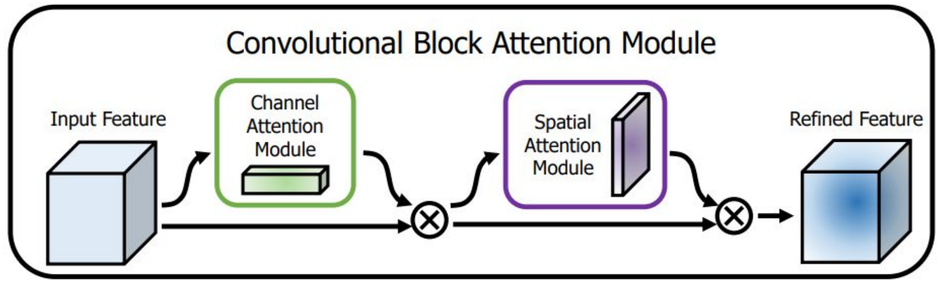
Combining Multiple Modalities for HAR

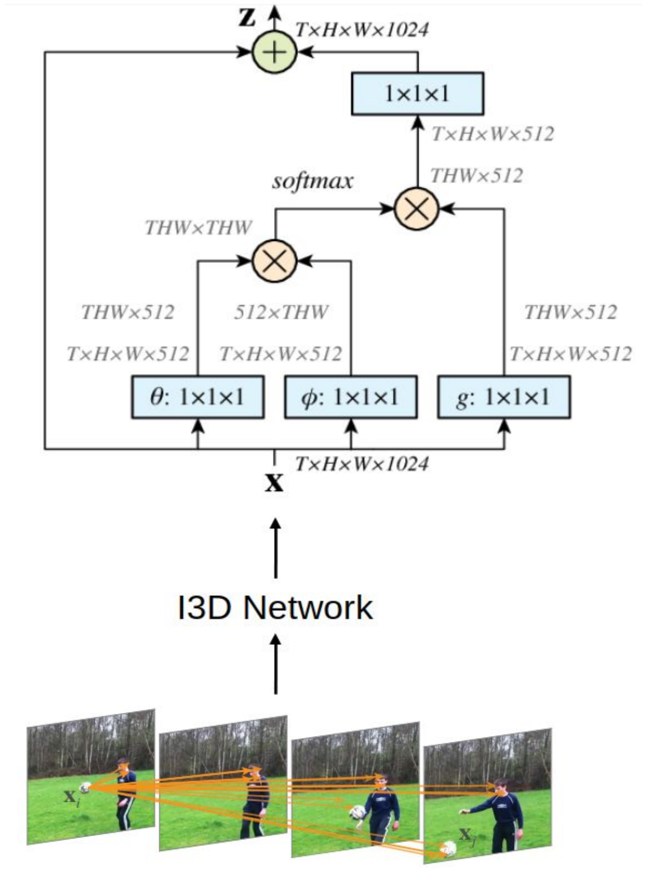


MAE-SSL

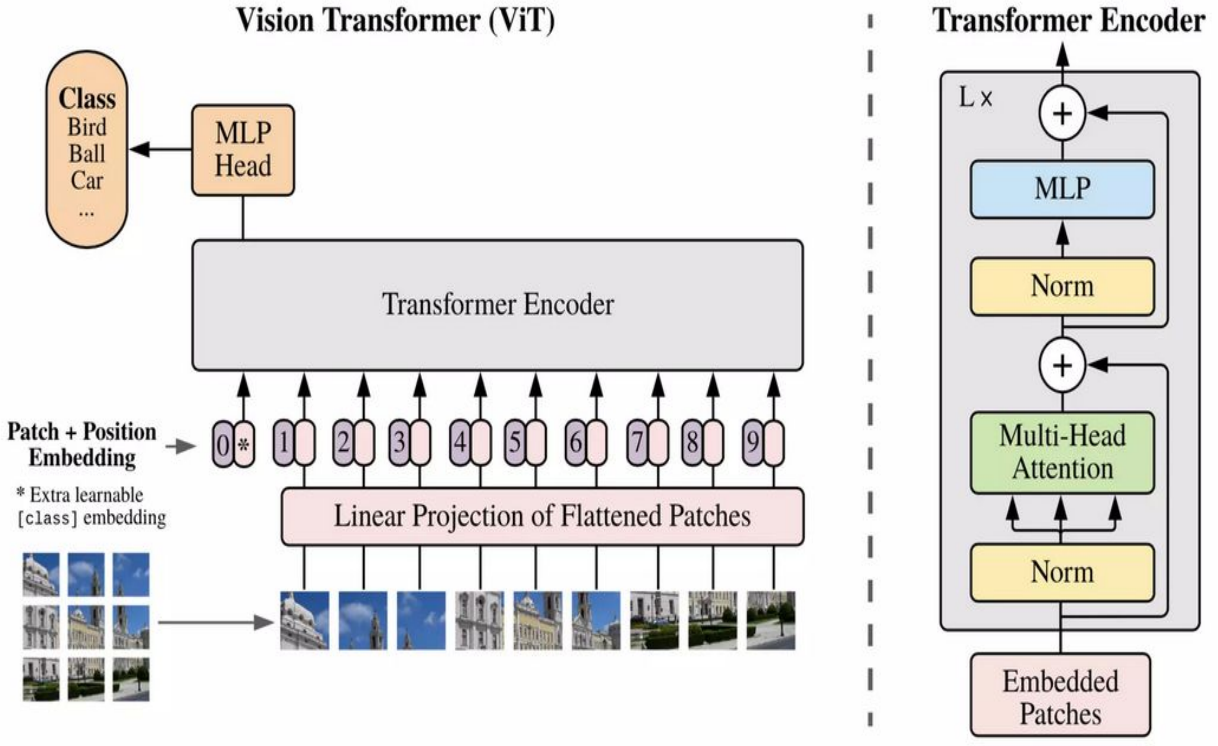


Attention Mechanism

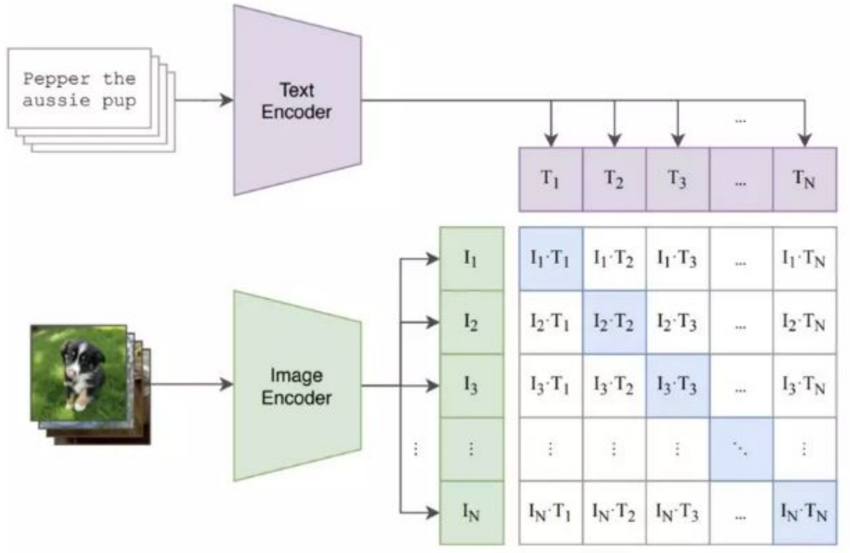




Transformer Models



CLIP:Vision-language



	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$	...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$	...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$	...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	⋱	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$	...	$I_N \cdot T_N$

Thank you for your attention!