Approximated Bilinear Modules for Temporal Modeling

By Xinqi Zhu

ICCV 2019

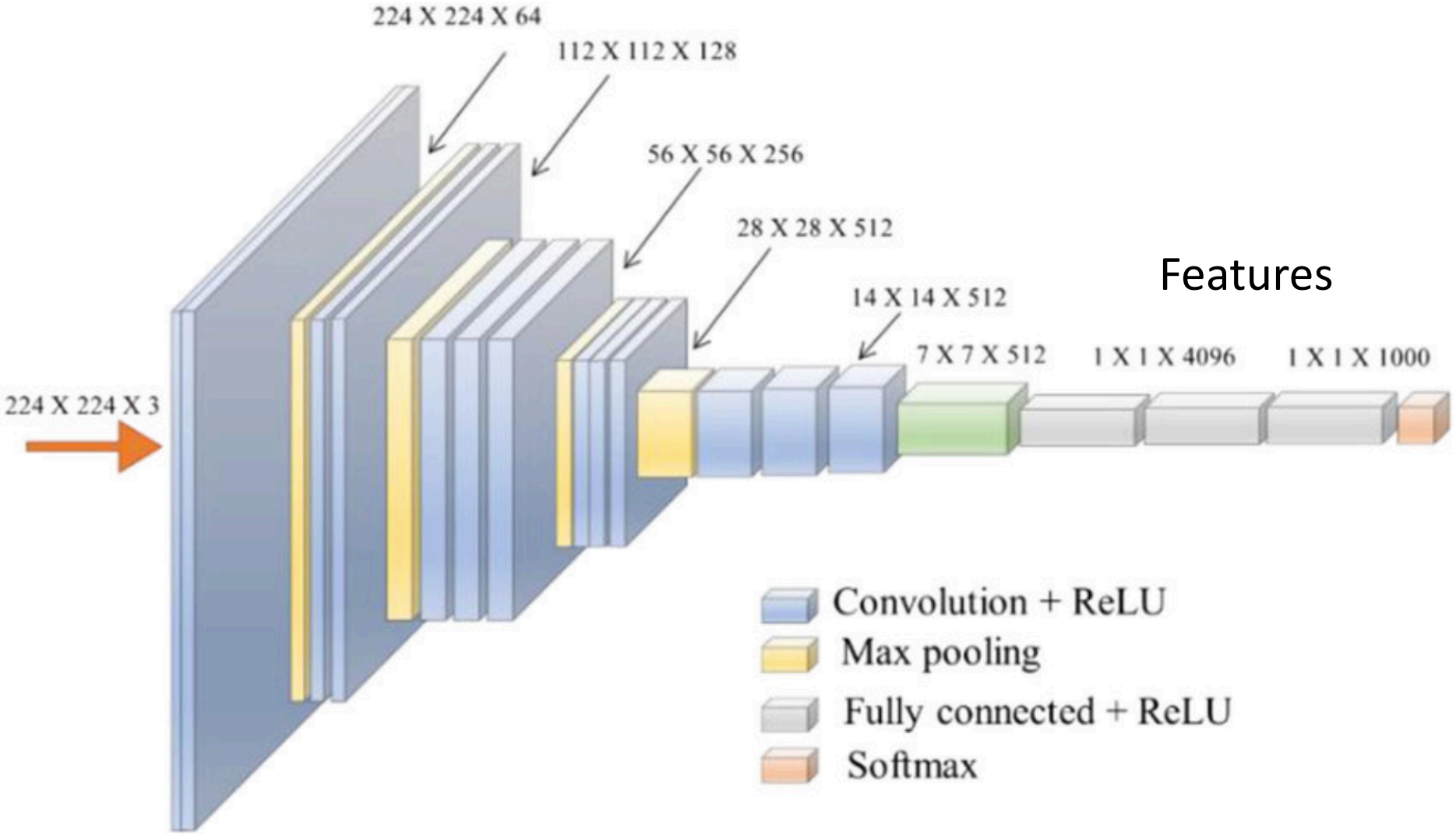
Background: Temporal Modeling in Video Recognition

Ground Truth: Riding a bike

Ground Truth: Putting something on the edge of something so it is not supported and falls down

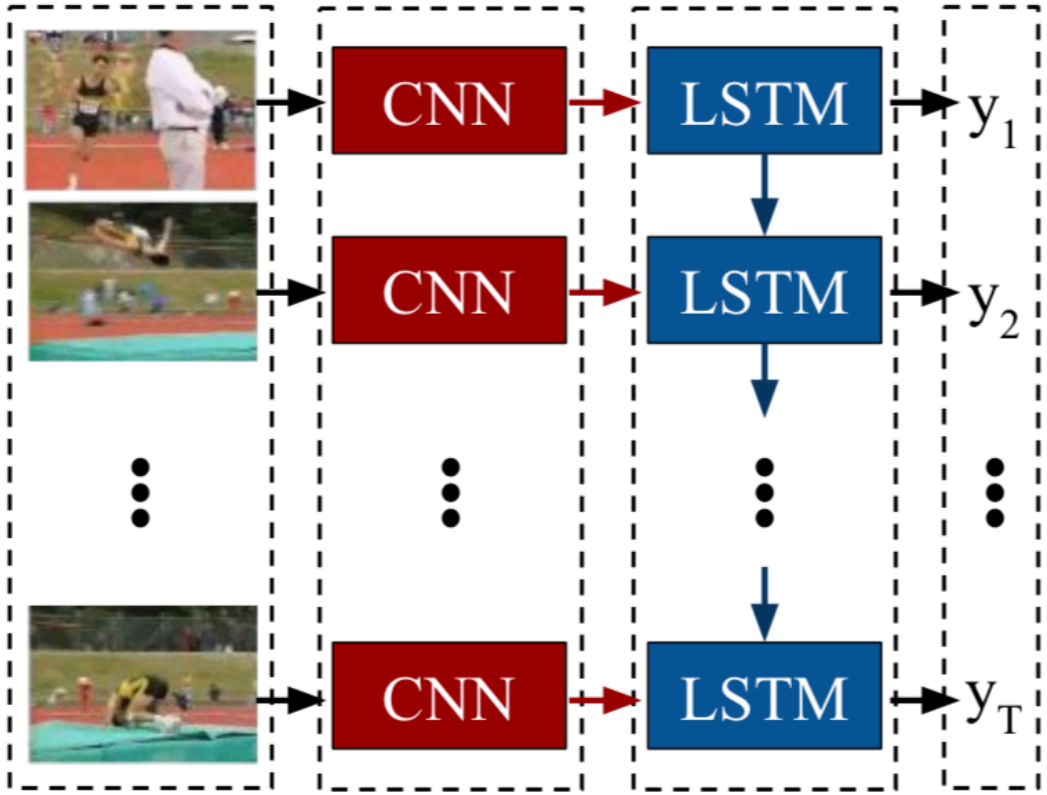
Background: Getting Image Deep Features by CNNs

An Image

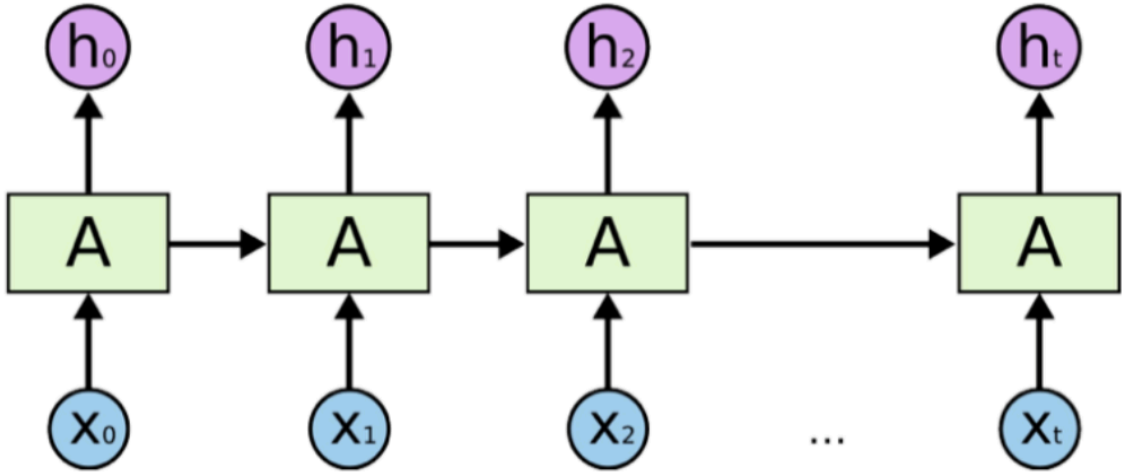


Background: Temporal Modeling in Video Recognition

Method 1: CNN + LSTM



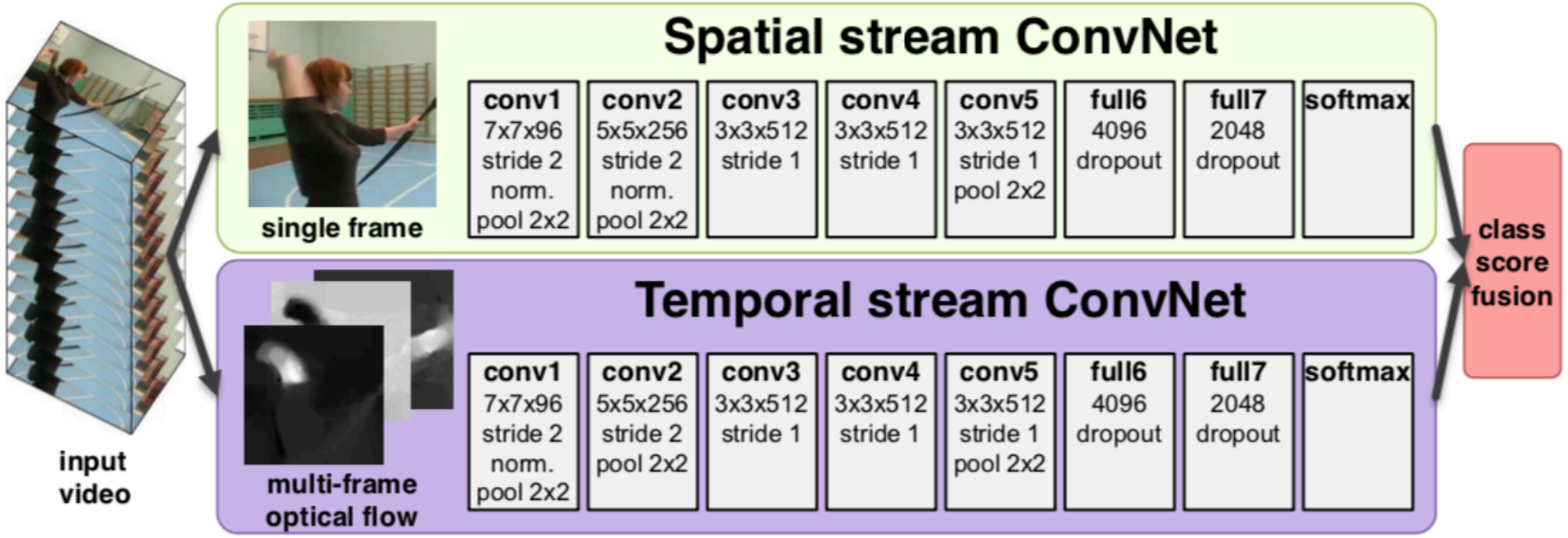
LSTM: Long Short Term Memory Networks



Donahue et. al. UC Berkeley. 2015 CVPR, Long-term recurrent convolutional networks for visual recognition and description

Background: Temporal Modeling in Video Recognition

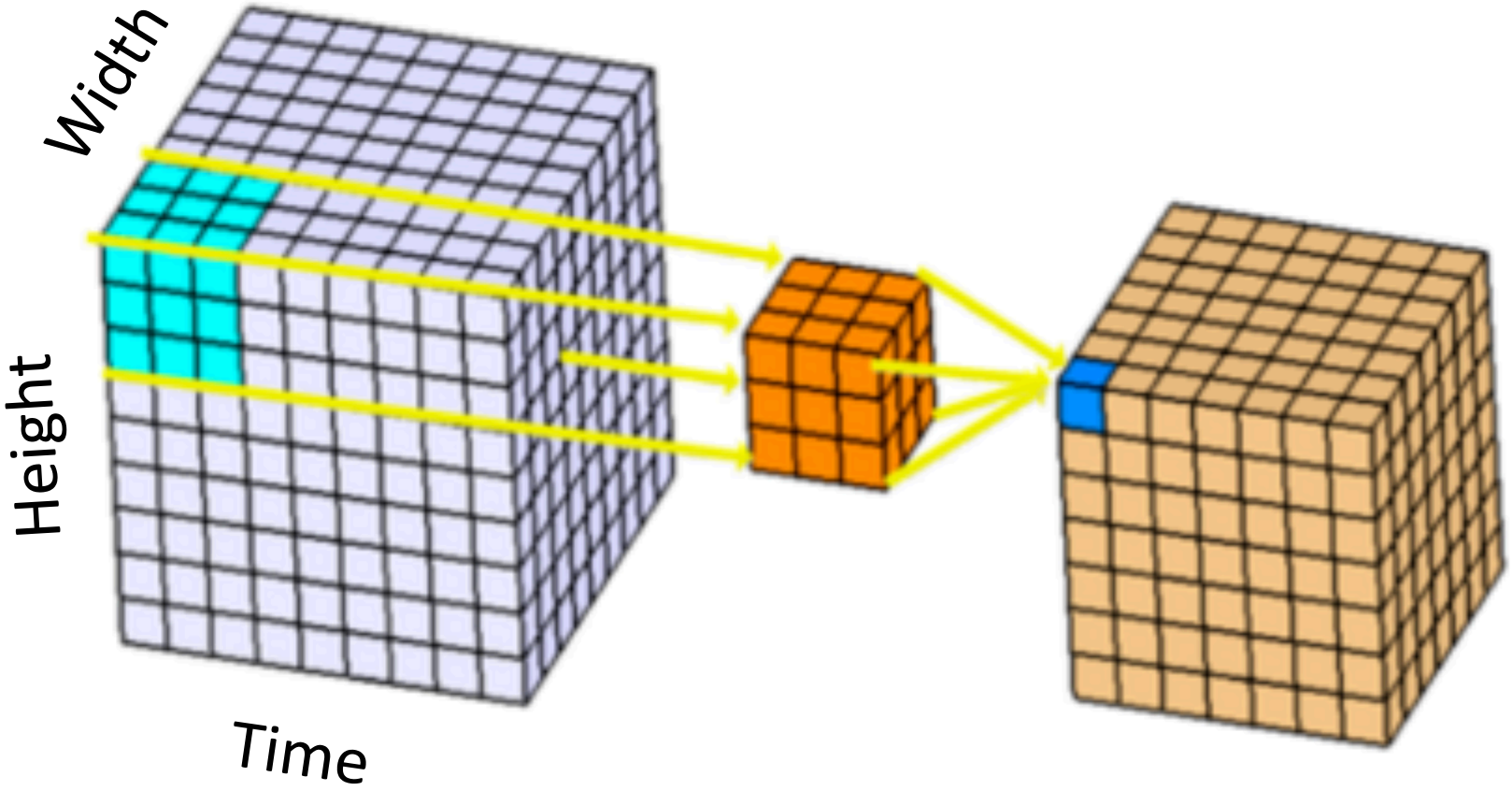
Method 2: Extract Optical Flow before Training



Simonyan et.al. UniofOxford , NIPS 2014, Two-Stream Convolutional Networks for Action Recognition in Videos

Background: Temporal Modeling in Video Recognition

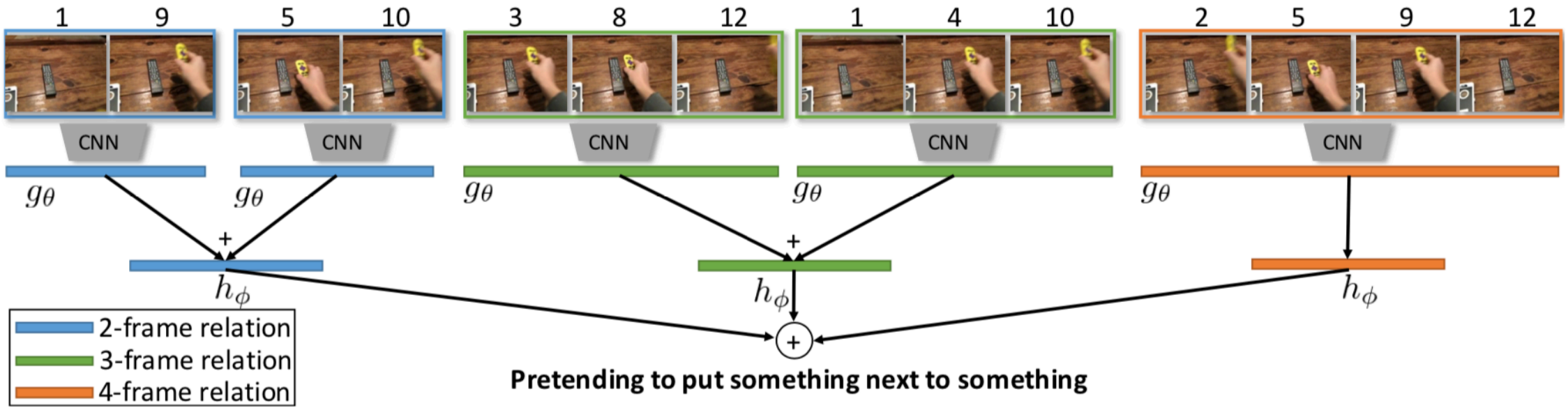
Method 3: Using 3D CNNs



Carreira et.al. DeepMind, CVPR 2017, Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset

Background: Temporal Modeling in Video Recognition

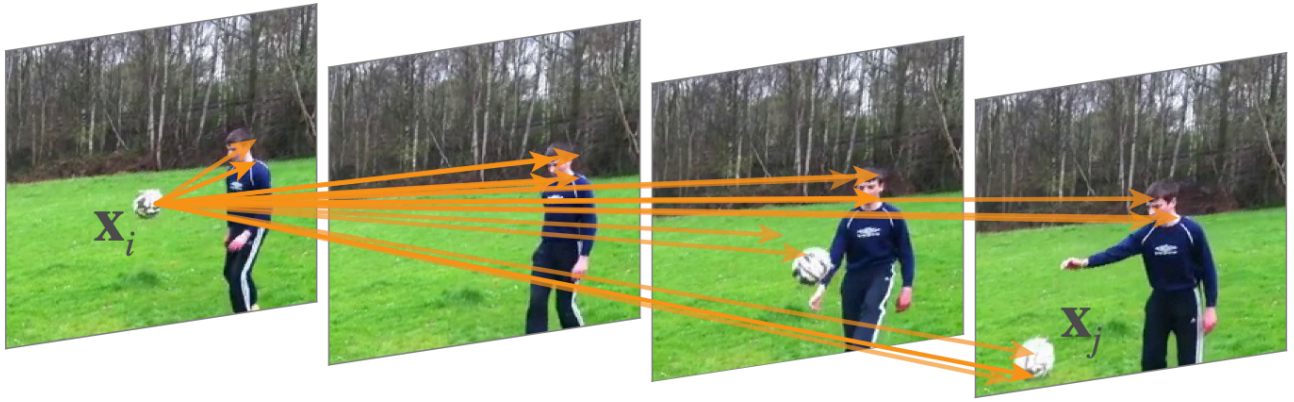
Method 4: Using Multiple MLPs

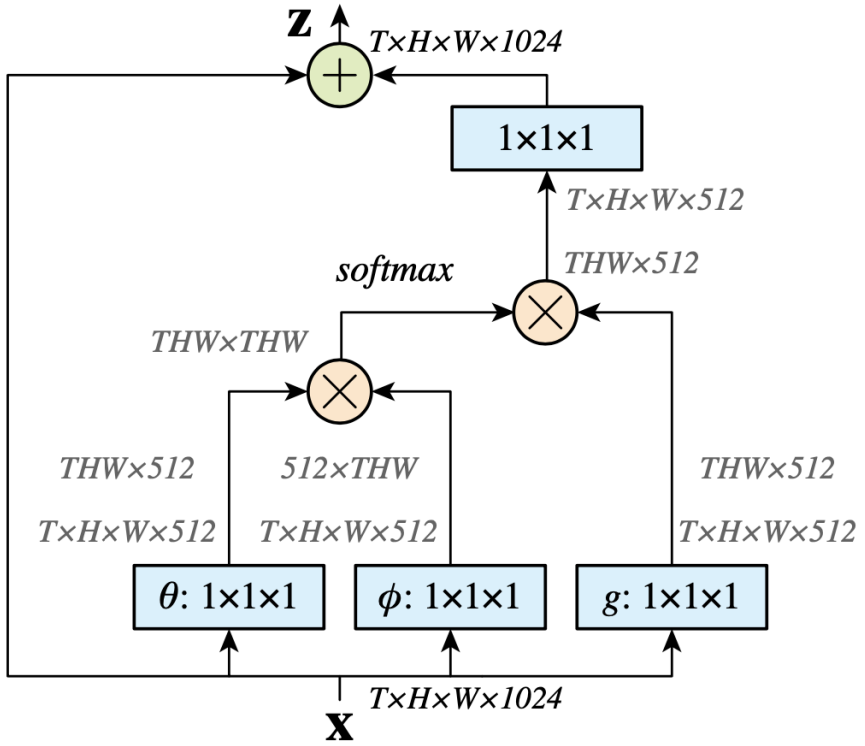


Zhou et.al. MIT, ECCV 2018, Temporal Relational Reasoning in Videos

Background: Temporal Modeling in Video Recognition

Method 5: Using Attention Networks (Non-Local Network)

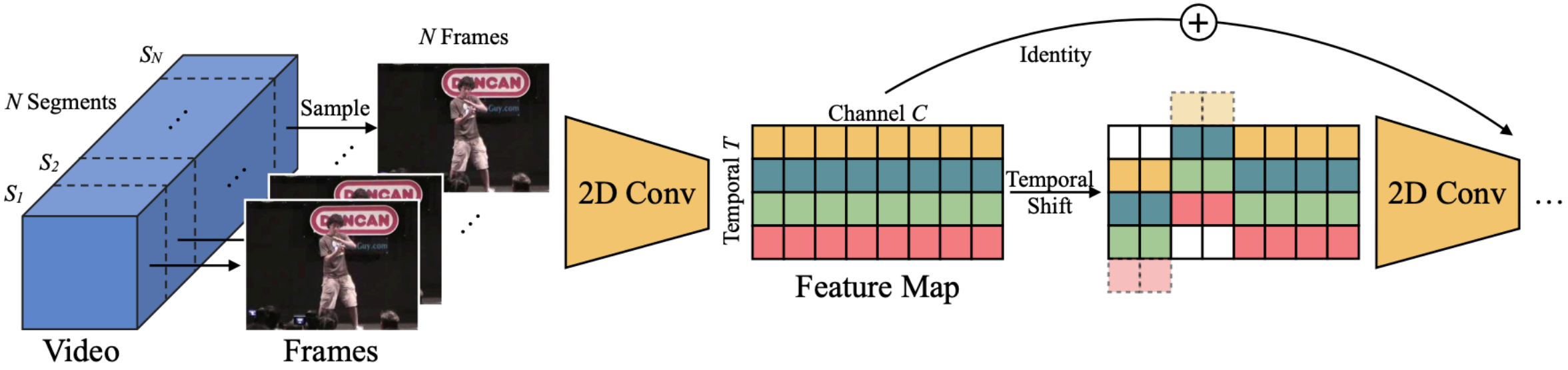




Wang et.al. CMU, CVPR 2018, Non-local Neural Networks

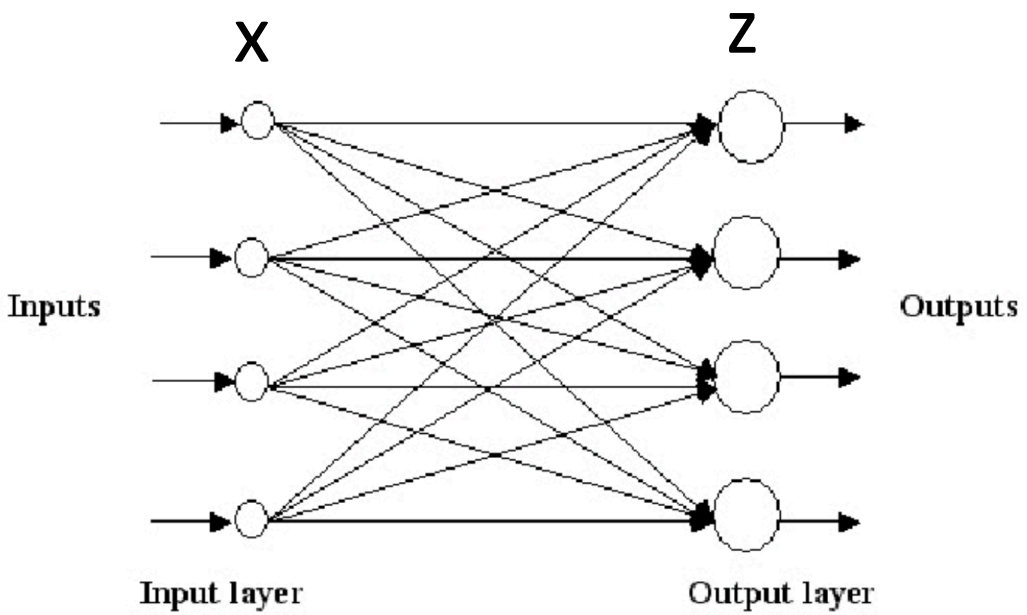
Background: Temporal Modeling in Video Recognition

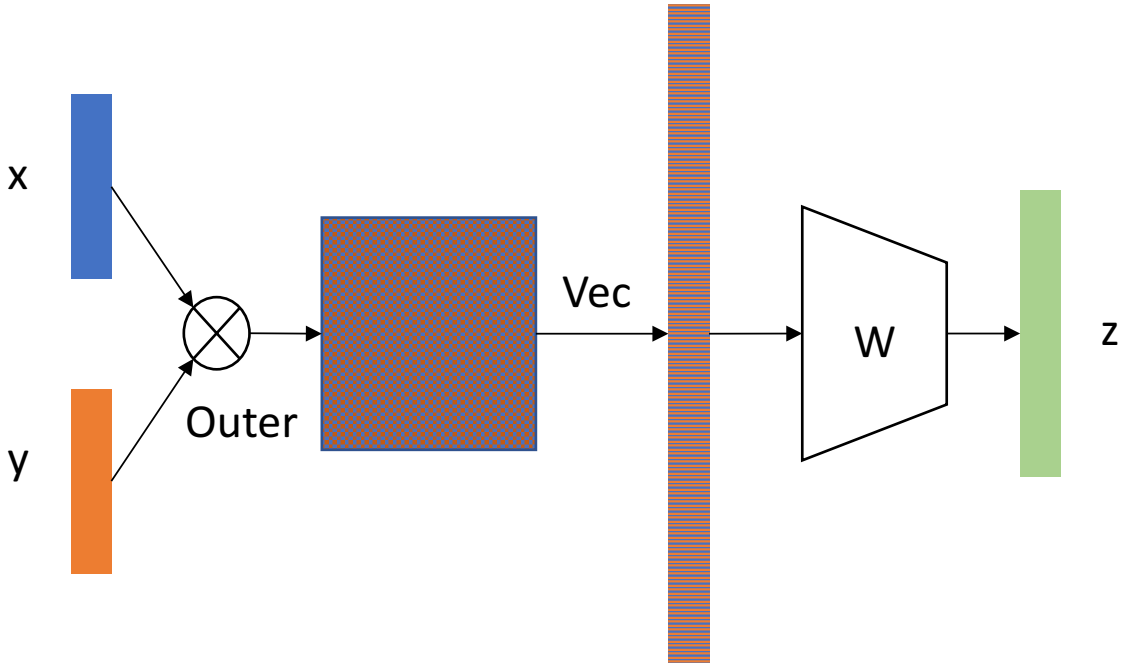
Method 6: By Exchanging Features Between Frames



Lin et.al. MIT, ICCV 2019, Temporal Shift Module for Efficient Video Understanding

My Method





torch.nn.Linear

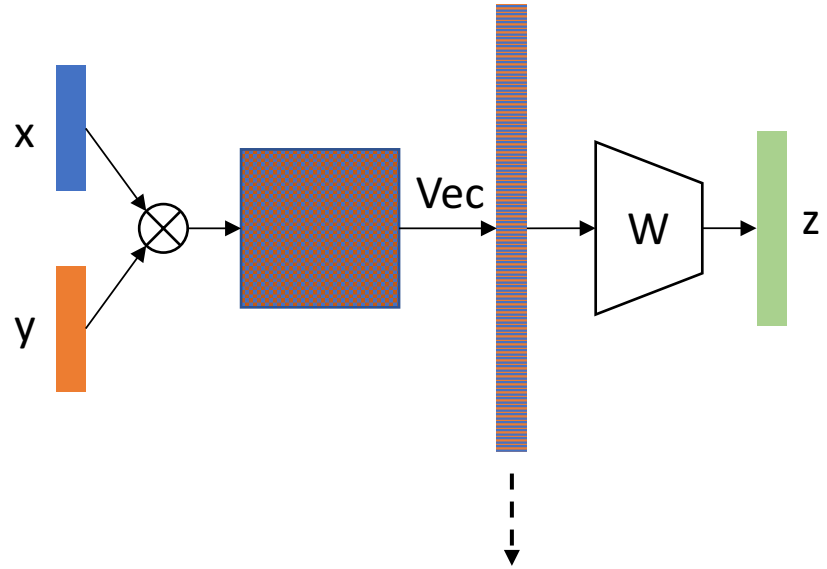
$$z = xW^T + b$$

torch.nn.Bilinear

$$z = xWy + b$$

My Method

Bilinear: $z = xWy + b$



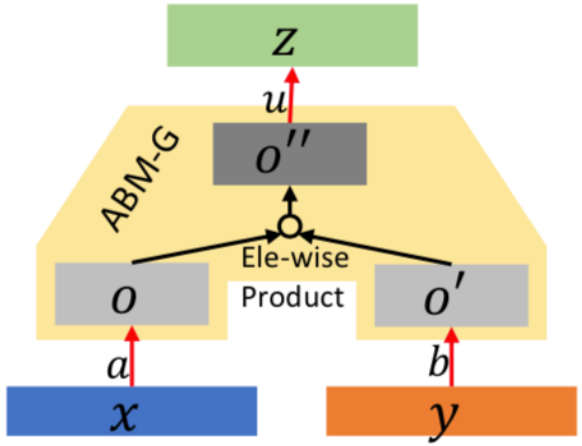
But this vector is too large…

For each w_{kij} in weight W,

$$w_{kij} = \sum_{r=1}^{R} u_{kr} a_{ir} b_{jr},$$

then:

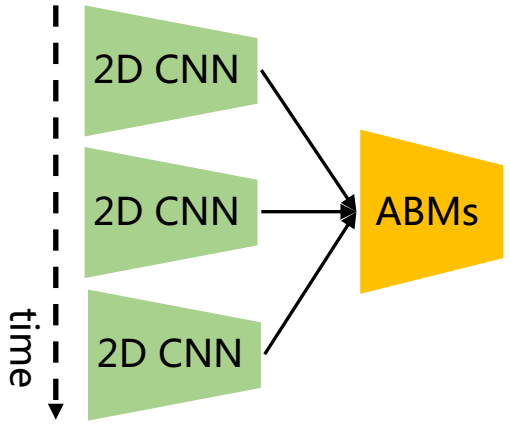
$$\begin{aligned} \boldsymbol{z} &= ABM_g(\boldsymbol{x}, \boldsymbol{y}) \\ &= \boldsymbol{u} \cdot (\boldsymbol{a}^T \boldsymbol{x} \circ \boldsymbol{b}^T \boldsymbol{y}), \end{aligned}$$



My Method

How to plug ABMs into existing deep models?

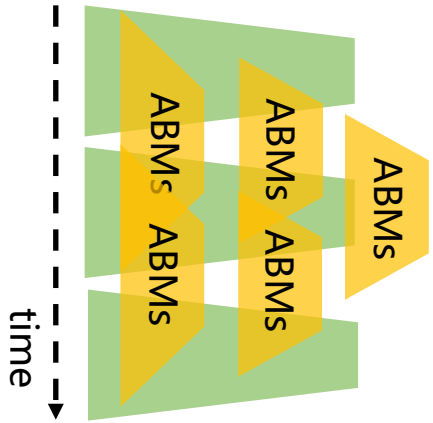
- On Top of CNNs.
  - +) Easy implementation
  - +) Super lightweight
  - +) High speed
  - -) Hard for deep ABMs
  - -) Small temporal receptive field



My Method

How to plug ABMs into existing deep models?

- Implanted into CNNs.
  - +) Better performance
  - +) Large temporal receptive field
  - +) Reuse pretrained parameters
  - -) Heavier



My Method

Details about *Implanted in CNNs* implementation.

MLP

$w_2 \cdot ReLU(w_1 \cdot x)$

$w_2 \cdot (ReLU(w_1 \cdot x) \circ 1)$

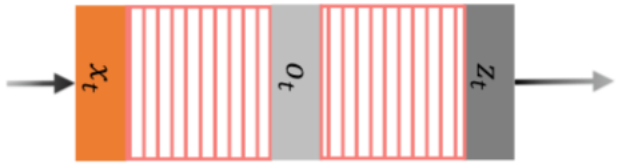
$w_2 \cdot (ReLU(w_1 \cdot x) \circ ReLU(w_1' \cdot x))$

[6]

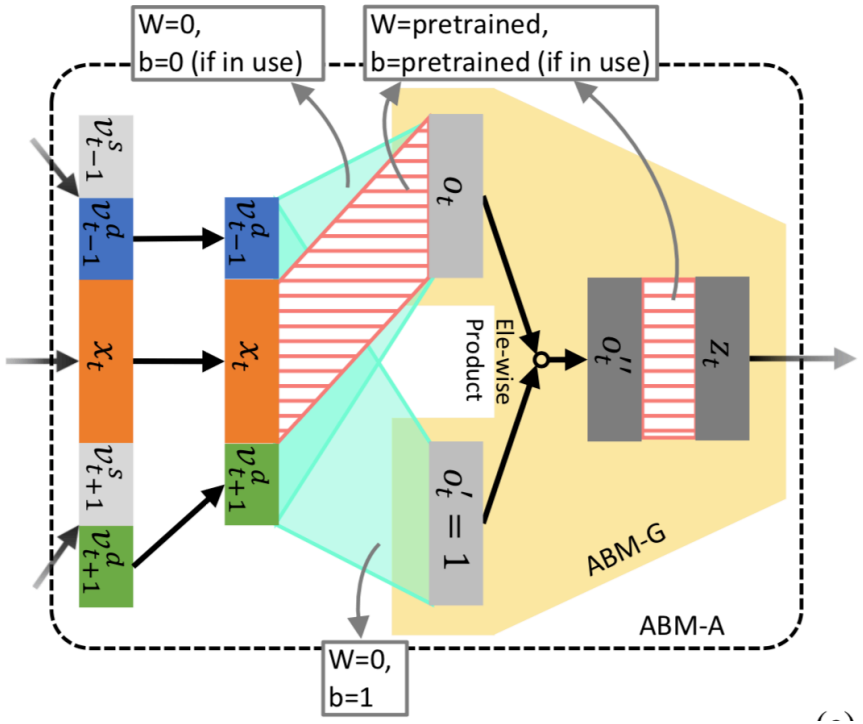
$w_2 \cdot ((w_1 \cdot x) \circ (w_1' \cdot x))$

ABM

MLP



ABM



Experiments

State-of-the-art comparison.

Model	Pretrain	Modality	#Frame	Backbone	v1-Val	v1-Test	v2-Val	v2-Test
Multi-Scale TRN [54]	ImgN	RGB	8	BN-Inception	34.44	33.60	48.80	50.85
3D-VGG-LSTM [31]	ImgN	RGB	48	3D-VGG	-	-	51.96	51.15
ECO-Lite$_{En}$ [55]	Kin	RGB	92	2D-Inc+3D-Res	46.4	42.3	-	-
NL I3D [49]	Kin	RGB	64	3D-ResNet-50	44.4	-	-	-
NL I3D+GCN [49]	Kin	RGB	64	3D-ResNet-50	46.1	**45.0**	-	-
TSM [27]	Kin	RGB	16	2D-ResNet-50	44.8	-	58.7	59.9
TSM$_{En}$ [27]	Kin	RGB	24	2D-ResNet-50	46.8	-	-	-
ABM-C-*in*	ImgN	RGB	16	2D-ResNet-50	47.45	-	-	-
ABM-C-*in*	ImgN	RGB	16×3	2D-ResNet-50	**49.83**	-	-	-
ABM-A-*in* β=1/2	ImgN	RGB	16×3	2D-ResNet-34	46.16	-	-	-
ABM-C-*in*	ImgN	RGB	16×3	2D-ResNet-34	46.81	-	**61.25**	**60.13**
ABM-AC-*in*$_{En}$	ImgN	RGB	32×3	2D-ResNet-34	**49.02**	42.66	-	-
Multi-Scale TRN [54]	ImgN	RGB+Flow	8+8	BN-Inception	42.01	40.71	55.52	56.24
ECO-Lite$_{En}$ [55]	Kin	RGB+Flow	92+92	2D-Inc+3D-Res	49.5	43.9	-	-
TSM [27]	Kin	RGB+Flow	16+8	2D-ResNet-50	49.6	**46.1**	63.5	**63.7**
ABM-C-*in*	ImgN	RGB+Flow	(16+16)×3	2D-ResNet-34	50.09	-	**63.90**	62.18
ABM-AC-*in*$_{En}$	ImgN	RGB+Flow	(32+16)×3	2D-ResNet-34	**51.77**	45.66	-	-

Table 4. State-of-the-art comparison on Something-v1 and v2 datasets. En means an ensemble model, ImgN means pretrained on *ImageNet*, and Kin means pretrained on *Kinetics*. $N \times K$ means snippet sampling (see Sec. 3.4). Grouped by input modalities.

Thank you!
Q&A