Chapter 3

Action Recognition in Presence of Representation Bias

The ability to recognize human actions in a video is challenging due to the complex nature of video data and the subtlety of human actions. Human activities often get associated with surrounding objects and occur in specific scene contexts. The video-based recognition of human actions involves not only the individual actions themselves but also the interaction between the person and their surroundings. Human actions are like an Italian salad, with various components such as gestures, background scenes, surrounding objects, camera movement, lighting conditions, facial expressions, and body movements, all contributing to the overall action. Some components may have little effect on action classification, while others serve as contextual cues that frequently accompany a specific action class. For example, a workout exercise is typically performed in a gym, making the underlying gym scene a relevant contextual cue, as illustrated in Figure 3.1. A noise factor like a cat chasing a mouse in a gym might not greatly impact the classification of a workout activity. However, co-occurring contextual elements can create confusion for the recognition system. For example, if the action being performed is actually eating or drinking in the gym, there may be a lack of focus on the actual activity. Noise and context have different effects on the recognition process, with noise hindering performance while context can provide crucial cues. Representation bias can arise from underlying scenes, background noise, or surrounding objects and can lead to inaccuracies in recognition. Objects associated with action also play a crucial role in recognition. For instance, if a person is sitting on a boat with oars nearby, they might be mistakenly recognized as rowing the boat due to similarities in the background if the relevance of the moving oars is not taken into consideration, as illustrated in Figure 3.1 (third and fourth rows).

In this study, we tackle the challenge of recognizing subtle human actions amidst noise, background context, and surrounding objects. The traditional approach overlooks the separation of representation biases from the action and trains a classifier using supervised learning, which can only distinguish between positive and negative examples. This method may reduce noise but fails to address the impact of irrelevant objects and background context. The dominance of co-occurring objects and other contextual elements in a video makes it difficult to perform accurate classification of similar action classes that share similar representation biases. As a result, the classifier struggles to be applied in real-world situations.

In this work, we aim to tackle the challenge of separating human actions from associated objects and contextual elements. A straightforward solution is to train separate classifiers for each element, including recognizing the human action, identifying surrounding objects, and considering contextual components. However, this approach requires a large amount of annotated training data to differentiate the human action, objects, context, and unrelated noise. The manual annotation process for this level of detail can be time-consuming, if not unfeasible. Additionally, this method is not scalable for large systems recognizing multiple video-based human actions.

The existence of representation bias is due to the interaction between the performer and objects in specific scene contexts. As seen in the first two rows of Figure 3.1, even

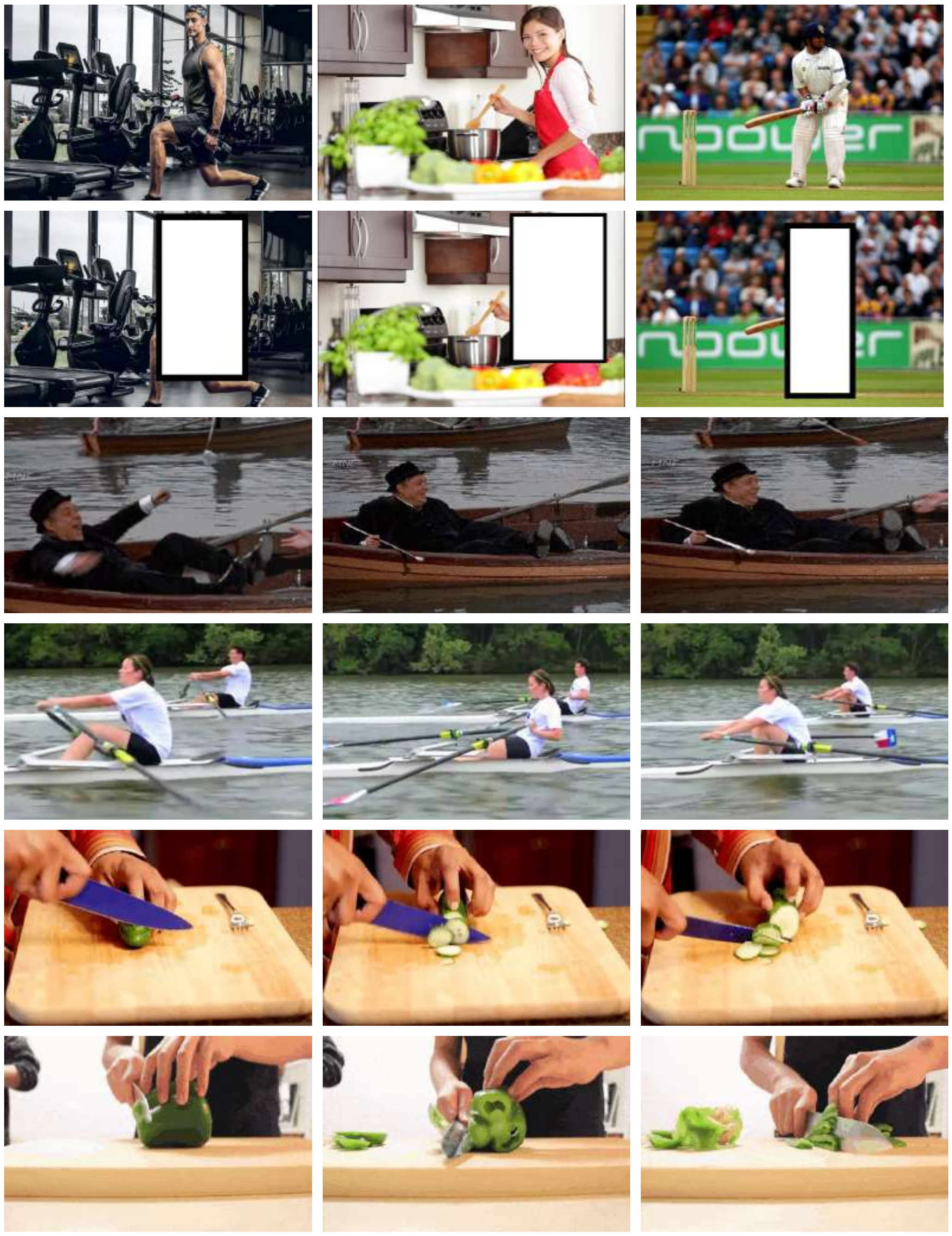


Figure 3.1: The video frames from various video clips of the FactNet dataset are shown in the first row, with the second row showing actor-masked frames. Despite this, actions like working out, cooking, and batting can still be identified, indicating the significance of the background context in action classification. The third and fourth rows depict the actions of sitting on a boat with oars nearby and rowing a boat, respectively. These frames showcase a pair of ambiguous actions that are different in terms of objects and subtle action patterns but share similar contextual elements. The last two rows show the chopping and slicing activities, which can only be distinguished through the long-term temporal context.

when actors are masked in the videos, the actions can still be easily inferred. This suggests that deep neural network models can pick up biases during action recognition training, leading to a lack of generalization in the classification process. Moreover, some actions, such as “chop with a knife” and “slice with a knife” shown in the last two rows of Figure 3.1, are similar in terms of motion patterns, objects, or scenes, making it necessary to capture the action over a longer time frame to differentiate them.

The aforementioned analysis has motivated the development of an efficient architecture for human action recognition that separates actions from co-occurring objects and contextual elements. This work introduces FactorNet, a deep network that factorizes actions into three components: the performer, associated objects, and scene cues.

3.1 Our Contribution

To summarize, the key contributions of this chapter are as follows:

- First, we explore the problem of extracting dense, discriminative and semantically enrich spatiotemporal features from videos. To solve this problem, we design a multi-scale deformable backbone network that consists of eight 3D convolution layers with top-down pathway and lateral connections to merge low-resolution, semantically strong features with high resolution. Finally, we utilize 3D deformable convolution layers to capture the actors and objects of different sizes.
- Second, we introduce an actor-object-scene attention network that separates an the activity performed by an actor, co-occuring objects, and underlying background. This helps to tackle the ambiguity that occurs in action recognition task due to co-occurring objects and scene in a video.
- Third, we capture long-range temporal dependencies for actor-object and scene branches using ConvLSTM. We also incorporate spatiotemporal attention in actor-object and temporal attention in scene branch respectively for improving the recognition accuracy.

- Next, we utilize the concept of attention mechanism in temporal feature pooling to extract semantic information that changes over time in a video clip. We also design a regularized objective function to accelerate the joint training mechanism in training the overall architecture in end-to-end fashion.
- Finally, we analyze the results affected by different parameters in ablation study section on six benchmark datasets. Existing datasets may capture and leverages representation bias during training, which may mitigate the impact of supervised representation for new action classes. Hence, we design a new dataset, denoted by *FactNet*, with 38 classes that composed of activity-object-scene related actions occur in daily life. None of the benchmark datasets has ambiguous conjugate action pairs, considering co-occurring objects.

3.2 Organization of the Chapter

The rest of this work is organized as follows. The next section summarizes state-of-the-art literature for action recognition architectures for videos. In Section 3.4, we describe the proposed architecture for action recognition. We present the experimental results and the ablation study in Section 3.5. Finally, we conclude the chapter with related publication in Sections 3.6 and 3.7, respectively.

3.3 Literature Survey

We describe the literature of the action recognition related to presence of representation bias are as follows:

Attention-based Action Recognition: Motivated by the incredible success of attention mechanism in deep networks, several action recognition architectures [114–116] have included attention networks to improve the performance. Attention pooling [114] mechanism has used the low-rank approximations of bilinear pooling models based on the derivation from top-down and bottom-up attention. It however only considers the spatial locations of adjacent frames in a video. It has focused on only short-range temporal relationships among consecutive frames. To obtain a low-cost I3D network, a gating based attention mechanism is incorporated to improvise the balance between accuracy and speed [115]. In [116], the authors have proposed a local feature integration framework based on attention clusters. They also introduced shifting operation to achieve more diverse action classes. PIC [117] has focused on modeling and recognizing the action performed by a human for a long duration by incorporating self-attention, convolution aggregation respectively.

Human-Object Interaction based Action Recognition: Recently, impressive progress has been made by utilizing pre-fixed graph structures to extract contextual relationship between human and objects. In [118], the authors utilized end-to-end trainable graph neural network (GNN) to detect and recognize activity-object interactions in images and videos. However, the method requires additional annotations of interactions to recognize the action and unable to distinguish ambiguous action occurred due to contextual information. In [119], the network learns 'interaction hotspot' maps, *i.e.,* "how humans interact with different objects" directly from videos of people naturally interacting with objects, with weaker modes of supervision. However, it ignores the contextual cues for action anticipation.

Action Recognition with Representation Bias:

•**Motivation of this Chapter:** The aforementioned methods, however, are still not able to address the ambiguities in action where the actor performs an action in a similar scene, but with a different object, shown in Figure 3.1 (third and fourth rows). Thus, the factorization of action with co-occurring scene without considering associated objects is incomplete. Furthermore, consideration of long duration temporal context is also essential. Therefore, we are motivated to design a deep network that can address the issue of factorization of action into actor with co-occurring objects and underlying scene and also span over a long temporal duration.

3.4 Proposed Approach

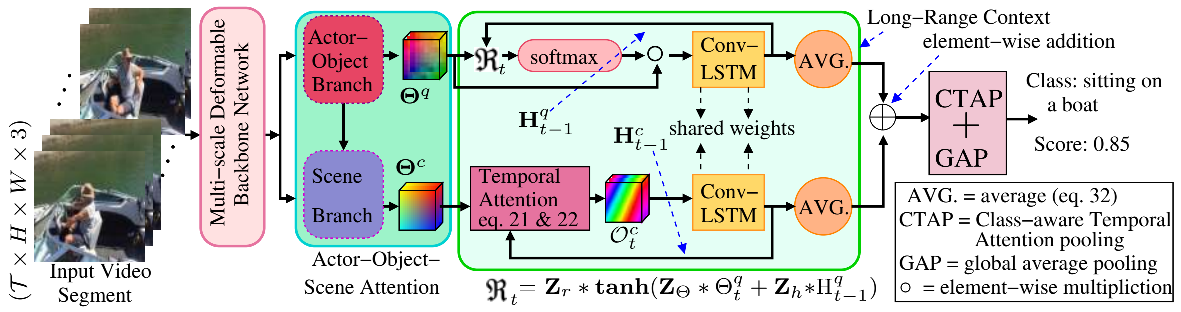

($\mathcal{T} \times H \times W \times 3$)
Input Video Segment
Multi-scale Deformable Backbone Network
Actor–Object Branch
Scene Branch
Θ^q
Θ^c
Actor–Object–Scene Attention
$\Re_t$
softmax
H^q_{t-1}
H^c_{t-1}
Conv–LSTM
shared weights
Conv–LSTM
Temporal Attention eq. 21 & 22
$\mathcal{O}^c_t$
AVG.
AVG.
Long–Range Context
element–wise addition
CTAP + GAP
Class: sitting on a boat
Score: 0.85
$\Re_t = \mathbf{Z}_r * \tanh(\mathbf{Z}_\Theta * \Theta^q_t + \mathbf{Z}_h * H^q_{t-1})$
AVG. = average (eq. 32)
CTAP = Class–aware Temporal Attention pooling
GAP = global average pooling
○ = element–wise multipliction

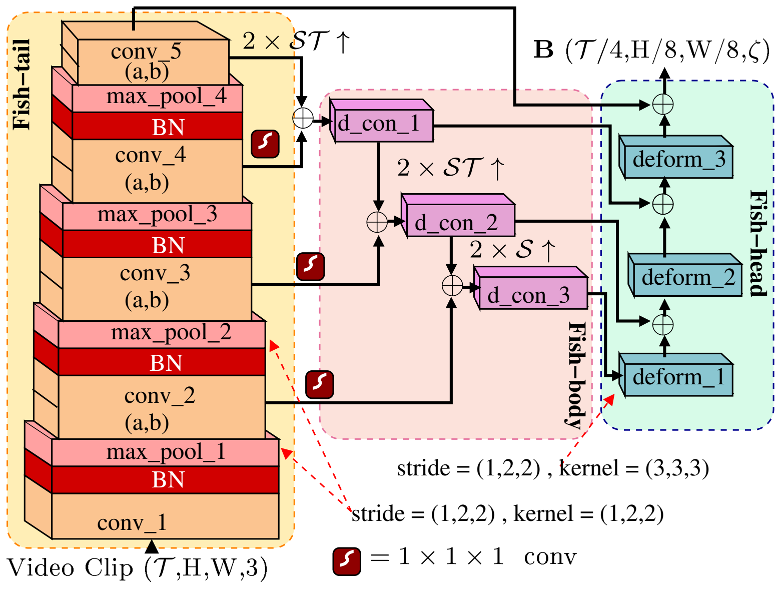
Figure 3.2: Overview of FactorNet. This architecture consists of multi-scale deformable backbone (MDB), actor-object-scene attention (AOSA), long-range context (LRC), and class-aware temporal attention pooling (CTAP) networks. We model factorization of action into actor with co-occurring objects and underlying scene and also span over a long temporal duration in FactorNet.

In video-based action recognition, the architectures [11,54] mostly learn short duration action representations of a human in a video clip that range from 1 to 16 frames. In a real-life scenario, yet, there are ambiguous human actions, like ‘chop with a knife’ and ‘slice with a knife’ that often lasts for several seconds [120,121] and need to capture long temporal duration for precise recognition. Furthermore, the underlying scene plays an important role in action recognition. Figure 3.1 (first two rows) depicts that even if the actor is masked in a video clip the scene bias helps to recognize the action accurately. Moreover, there are still many human actions that are ambiguous when the associated objects are not given proper attention. For instance, ‘rowing a boat’ and ‘sitting on a boat (with oars alongside)’, Figure 3.1 (third and fourth rows) have similar underlying scenes but different human actions due to the movement of oars by the actor to drive the boat. The action performed by an actor with or without an object (or different objects) helps to recognize the actions precisely.

We propose a deep network, known as *FactorNet*, for video-based action recognition. The block diagram of the overall architecture is shown in Figure 3.2. In this work, firstly, we propose a multi-scale deformable backbone network consisting of 3D convolution layers to capture short-range spatiotemporal features of a video clip, which are enriched in both finer-details and semantic knowledge. 3D deformable convolution helps to capture size variations in spatiotemporal extents. Secondly, we design an actor-object-scene attention (AOSA) network that factorizes the action into an actor, co-occurring objects, and scene cues. This focuses on salient parts in visual space to capture spatiotemporal features related to the action performed by an actor and supervise the impact associated object including the underlying scene on classification. Thirdly, we propose a long-range spatial-temporal context (LRC) network that helps to learn the long temporal context of an action for both actor-object and scene branches. We incorporate a layer of attentive ConvLSTM to extract salient features for each branch over time. Finally, we integrate the concept of class-aware temporal feature pooling to reduce the processing time for videos with long duration. Recent literature has utilized 3D deformable convolution for MRI classification [122] and video super resolution [123]. For super-resolution, the authors have incorporated 3D deformable convolution to extract spatiotemporal information from both spatial and temporal dimensions in videos. In addition, 3D deformable convolution also achieves adaptive motion compensation efficiently. Similarly, 3D deformable convolution is implemented for significant improvement in classification performance for unprocessed and skull-stripped brain images. However, it is not explored in the field of action recognition.

3.4.1 Multi-scale Deformable Backbone (MDB) Network

A video is decomposed into spatial and temporal components that capture information about the actor, surrounding objects, and underlying scene over time to identify an action. The spatiotemporal features are learned from a video sequence using a deep network to classify actions into different categories. 3D convolutional neural networks, namely C3D, is a well-suited deep network for encoding spatiotemporal features by convolving 3D kernel over spatial, temporal, and semantic channel subspace. The spatiotemporal feature tensor $\mathbf{B} \in \mathbb{R}^{\mathcal{T}\times H\times W\times \mathcal{C}_{out}}$ is obtained by convolving 3D filter $\mathbf{K} \in \mathbb{R}^{d\times l\times l}$ over the video clip $\mathbf{V} \in \mathbb{R}^{\mathcal{T}\times H\times W\times \mathcal{C}_{in}}$, which is defined by: $\mathbf{B} = \mathbf{K} * \mathbf{V}$, where $*$ denotes a convolution operator, and $\{d, l, l\}$ are the temporal depth ($d < \mathcal{T}$) and spatial resolution of a filter, respectively. $\{\mathcal{T}, H, W, \mathcal{C}_{out}\}$ are the number of frames, height, width, and the number of channels of the output tensor. For an input video clip, the value of $\mathcal{C}_{in}$ is 3, which denotes the RGB channels.



Fish−tail
conv_5 (a,b)
max_pool_4
BN
conv_4 (a,b)
max_pool_3
BN
conv_3 (a,b)
max_pool_2
BN
conv_2 (a,b)
max_pool_1
BN
conv_1
Video Clip ($\mathcal{T}$,H,W,3)
$2 \times \mathcal{ST} \uparrow$
d_con_1
$2 \times \mathcal{ST} \uparrow$
d_con_2
$2 \times \mathcal{S} \uparrow$
d_con_3
Fish−body
B ($\mathcal{T}$/4,H/8,W/8,ζ)
deform_3
deform_2
deform_1
Fish−head
stride = (1,2,2) , kernel = (3,3,3)
stride = (1,2,2) , kernel = (1,2,2)
= $1 \times 1 \times 1$ conv

Figure 3.3: Overview of multi-scale deformable backbone network that consists of Fish-tail, Fish-body, Fish-head block. A video clip of size $\mathcal{T} \times H \times W \times 3$ is given as input to Fish-tail block to obtain spatiotemporal features. conv_1 to conv_5b are the 3D convolution layers with kernel size $3 \times 3 \times 3$. max_pool_1 to max_pool_4 are max-pooling layers and BN is batch normalization layer. The kernel size of the first two max-pool layers are $1 \times 2 \times 2$ and the remaining layers have size 2. d_con_1 to d_con_3 are feature tensors obtained after concatenating the high-, mid-, and low-level feature tensors. $\uparrow$ depicts upsampling operation, which is performed on spatiotemporal feature tensor either spatially or spatial-temporally. deform_1 to deform_3 in Fish-head block is the deformable convolution layers with kernel size $3 \times 3 \times 3$. The stride of deform_1 is $1 \times 2 \times 2$ and the stride of remaining deformable convolution layers are set to 2. **B** is the spatiotemporal feature tensor obtained from MDB.

In this section, we propose a *multi-scale deformable backbone (MDB)* network, which is inspired by C3D [12] and FishNet [124], to extract spatiotemporal high-level semantic features. In addition, we have incorporated deformable convolution [125] layers to extract fine-scale action features that involve an actor and associated objects of different size precisely. Our backbone network is divided into three blocks: fish-tail, fish-body, and fish-head, as shown in the Figure 3.3. The fish-tail block consists of eight 3D convolution layers (conv_1 to conv_5b) and four maxpool layers (max_pool_1 to max_pool_4). The three upsampling mechanism are incorporated in fish body block and three 3D deformable convolutions (deform_1 to deform_3) layers in fish head block, respectively.

Formally, the original video is partitioned into L non-overlapped video clips which is a set of $\mathcal{T}$ RGB frames with spatial size $H \times W$. A video is denoted as $\mathbf{V} = \{V_k | k \in \{1, 2, \cdots, L\}\}$, where $V_k \in \mathbb{R}^{\mathcal{T} \times H \times W \times 3}$ is a video clip. The network is set up to take a video clip as an input to extract spatiotemporal features that help in enhancing the accuracy of recognition. The video V_k is fed into fish-tail block to extract spatiotemporal feature tensor. The size of kernel is $3 \times 3 \times 3$ for all the convolution layers and the value of stride and padding is 1. The number of filters of first two convolutional blocks are 64 and 128, whereas remaining two blocks have 256, 512, and 1024 filters, respectively. The max-pooling layers progressively pools reduced feature tensors in the fish-tail block to reduce the number of training parameters, which is utilized after every convolution blocks. The first one maxpool layers have kernel size $1 \times 2 \times 2$ and the remaining layer has $2 \times 2 \times 2$ size. The intention to merge temporal features gradually is to preserve long-range temporal extent. We have removed the fifth pooling layer of the C3D network. The details of fish-tail block of the backbone network is presented in Table 3.1. We have stacked 3D-batch normalization [126] immediately after each convolutional block to accelerate the learning process.

3D convolutions are insufficient to detect smaller object features with respect to time. Thus, we have combined low-resolution, semantically strong features with high resolution via top-down pathway and lateral connection to our backbone network for capturing smaller objects. Please note that the combined feature tensors are represented as d_con_1, d_con_2, and d_con_3, respectively (as shown in Figure 3.3).

In fish-body block, the high-level features are upsampled and merged with low-level elements to preserve the features from the tail and the previous layer of the body by concatenation. We upsampled the output feature tensor of conv_5b by $2\times$ both spatially and temporally. The upsampled feature tensor is concatenated with the output feature tensor of conv_4b which is ζ-d channel-wise reduced using a $1 \times 1 \times 1$ convolution to produce the feature tensor (d_con_1). Similarly, we have concatenated output of conv_3b layer after applying $1 \times 1 \times 1$ convolution filter with d_con_1, which is $2\times$ upsampled both spatially and temporally. The output from d_con_2 is $2\times$ upsampled spatially using nearest neighbors upsampling and add them element-wise with the ζ-d channel-wise reduced output from conv_2 to obtain the enriched feature tensor at d_con_3 layer. The output produces by fish-body block at every layer is given as an input to fish$-$head block, as shown in Figure 3.3. However, we observe that there is still an unsatisfactory aspect in the extracted feature maps. For example, in Figure 3.4 (first row), the badminton racket is not captured completely using 3D convolution and in Figure 3.4 (second row), the cigar is partially occluded due to presence of fingers, which is not identified wholly in obtained feature map by 3D convolution. Thus, to tackle this tactful problem, we have incorporated deformable convolution in fish-head block to capture the co-occurred objects of different size accurately.

The 3D deformable convolution with input signal $\mathcal{X} \in \mathbb{R}^{\mathcal{T}\times H\times W}$ and convolution kernel $\mathfrak{W} \in \mathbb{R}^{k\times k\times k}$ is defined as:

$$\mathcal{Y}(t,h,w) = \sum_{t\in\mathcal{T}}\sum_{h\in H}\sum_{w\in W} \mathfrak{W}[-k]\mathcal{X}[(t,h,w) + k + \phi_{t,h,w,k}], \tag{3.1}$$

where $\phi_{t,h,w,k} \in \mathbb{R}^{\mathcal{T}\times H\times W\times \mathfrak{W}}$ represents the deformation offsets of deformable convolution. These offsets are learned from another convolution with $\mathcal{X}$, which is given as

$\phi_{t,h,w,k} = \mathcal{H}k * \mathcal{X}(t, h, w, k)$, where $\mathcal{H}$ is a different kernel.

Table 3.1: Architecture of fish-tail block of backbone network. Convolutional blocks are shown in square brackets, with number of blocks stacked. The filter size of convolutional layers is given as (temporal length× width × height), number of channels. Similar for pooling layers, kernel size is (temporal length× width × height) and output shape is (temporal length× width × height × output channels).

Layers	**Fish-tail block**	
	Blocks	**Output Size**
input	raw input videos: $\mathcal{T} \times 112 \times 112 \times 3$	
conv_1	$[3 \times 3 \times 3, 64]$	$\mathcal{T} \times 112 \times 112 \times 64$
max_pool_1	$1 \times 2 \times 2$	$\mathcal{T} \times 56 \times 56 \times 64$
conv_2	$[3 \times 3 \times 3, 128]$	$\mathcal{T} \times 56 \times 56 \times 128$
max_pool_2	$1 \times 2 \times 2$	$\mathcal{T} \times 28 \times 28 \times 128$
conv_3 (a,b)	$[3 \times 3 \times 3, 256] \times 2$	$\mathcal{T} \times 28 \times 28 \times 256$
max_pool_3	$2 \times 2 \times 2$	$\mathcal{T}/2 \times 14 \times 14 \times 256$
conv_4 (a,b)	$[3 \times 3 \times 3, 512] \times 2$	$\mathcal{T}/2 \times 14 \times 14 \times 512$
max_pool_4	$2 \times 2 \times 2$	$\mathcal{T}/4 \times 7 \times 7 \times 512$
conv_5 (a,b)	$[3 \times 3 \times 3, 1024] \times 2$	$\mathcal{T}/4 \times 7 \times 7 \times 1024$

In fish-head block, we have incorporated deform_1 to deform_3 layers with kernel size $3 \times 3 \times 3$ and ζ number of channels. The stride of deform_1 layer is $\{1, 2, 2\}$ and remaining two are set to $\{2, 2, 2\}$ in all dimension. First, we fed the output of d_con_3 layer into deform_1 to handle the geometric transformations. Next, the output obtained from first deformable convolution layer is element-wise concatenated with the output from d_con_2. Similarly, we fed the concatenated feature tensor to deform_2 layer and then merged using element-wise addition with the output tensor obtained at d_con_1 to produce the output of size $\mathcal{T}/2 \times H/8 \times W/8 \times \zeta$. Finally, a deform_3 is appended on the merged tensor to generate down-sampled feature tensor and merged with output of conv_5b that produces $\mathbf{B} \in \mathbb{R}^{\mathcal{T}/4 \times H/8 \times W/8 \times \zeta}$, where we kept $\zeta = 1024$ to maintain a trade-off between accuracy and training parameters. In this way, the salient features from every layer in backbone network are directly connected to the final layer through concatenation and skip connection.

Our backbone network is used to extract dense, discriminative and semantically

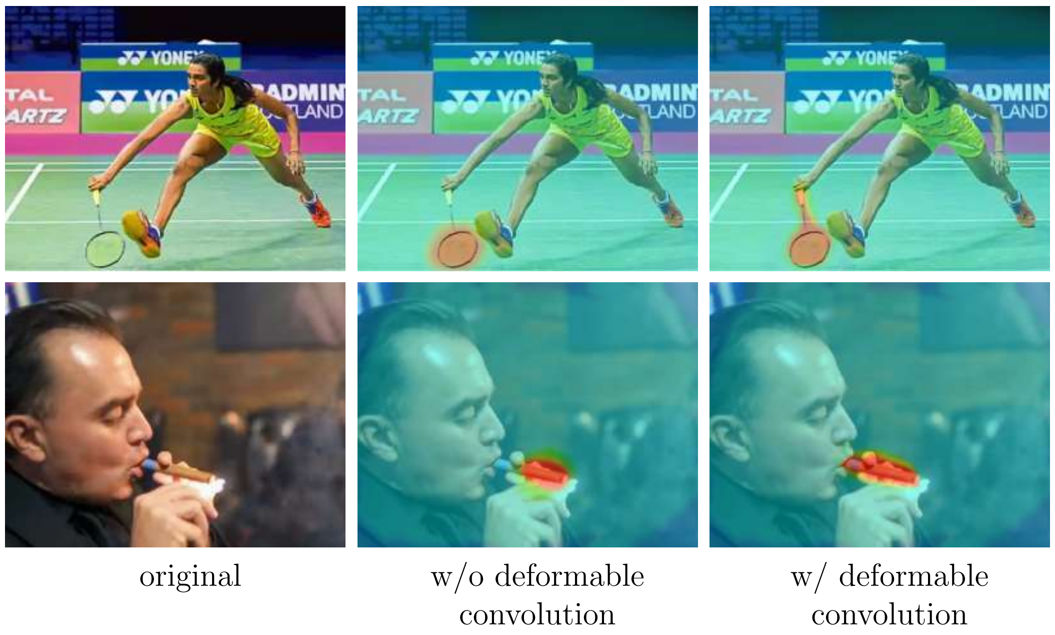


Figure 3.4: Video frames from 'playing badminton' (first row) and 'smoking with cigar' (second row) video clips of FactorNet dataset. These frames show attention map w/ and w/o deformable convolution in backbone Network.

enrich spatiotemporal features from videos. However, it is incapable to highlight the different relevant parts of visual space to acquire contextual features of action and object components in a video clip for recognizing human action more precisely. Therefore, we stacked a spatiotemporal attention network on the backbone network that factorize the actor with co-occurring object and scene information.

3.4.2 Actor-Object-Scene Attention (AOSA) Network

Humans focus on essential parts of visual space to process action relevant regions, instead of exploring an entire scene at a glimpse. This capability of actor, the human performing the action, is referred as visual cognition [127]. The property of visual cognition in action recognition is replicated by incorporating attention mechanism for selecting the pertinent parts in a video sequence. We therefore propose an attention mechanism that extracts associated object and background scene-related information from several parts of the video clip along with capturing the activity performed by an actor.

We incorporate the actor-object-scene attention (AOSA) network that factorizes a video into an action performed by an actor, co-occurring objects, and underlying scene to supervise representation biases. For example, action like ‘playing a saxophone’ will be ambiguous with ‘playing a bagpiper’ due to the similarity in the pattern of action, as shown in Figure 3.5. These two actions belong to two different classes in terms of contextual cues and relevant objects.

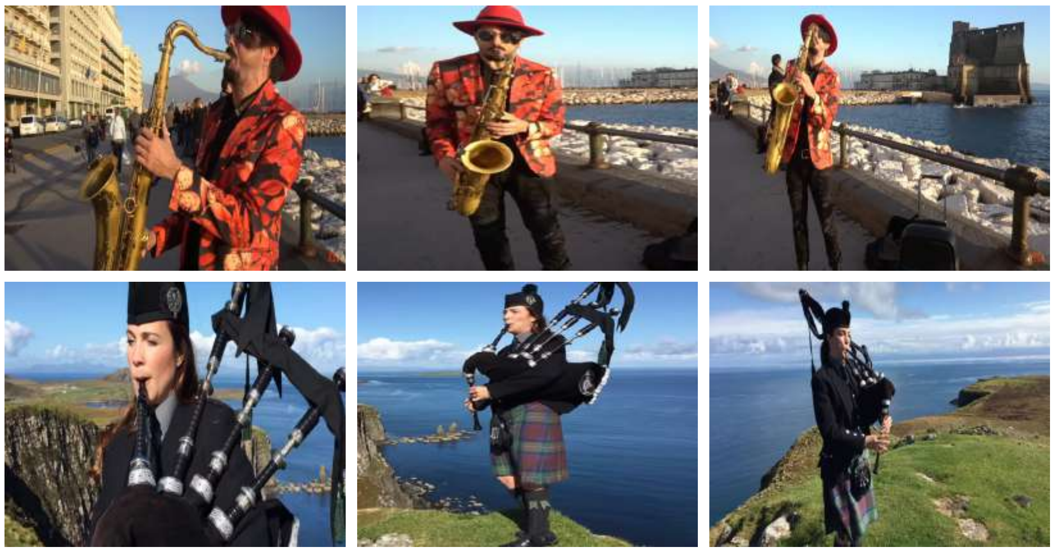


Figure 3.5: Video frames from two different video clips of FactorNet dataset. These frames show a pair of ambiguous actions which are dissimilar in terms of associated object. The first row shows the action of ‘playing a saxophone’ and second row depicts ‘playing a bagpiper’ action. Both have similar action pattern with different contextual elements.

AOSA network is used for encoding the 3D feature maps related to three branches, *i.e.*, actor, object, and scene. It exploits the different degrees of importance of being actor who performs the activity, object, and scene components. Each component within a clip is assigned with class-aware attention weights to adaptively focus on discriminative cues.

Actor-Object branch. In actor branch, we have calculated the score value of being actor component by utilizing sigmoid and softmax separately as shown in Figure 3.6.

First, we have channel-wise pooled the feature from the input feature tensor $\mathbf{B}$ to obtain spatial and temporal features. Formally, the pooled feature map of actor is calculated by:

$$\tilde{\mathbf{B}}^a = \frac{1}{\zeta} \sum_{i \in \zeta} \mathbf{B}(i), \tag{3.2}$$

where $\tilde{\mathbf{B}}^a$ is the output pooled map of size $\frac{\mathcal{T}}{4} \times \frac{H}{8} \times \frac{W}{8} \times 1$. Next, we have used fully connected layer which is followed by softmax and sigmoid functions simultaneously. We have exploited softmax with sigmoid function in separate branch as to eliminate irrelevant actor component present in video clip. The score value of being an actor component is calculated by element-wise multiplication as follows:

$$\mathbf{S}^a = \mathbf{ReLU}(\mathbf{w}_s^\top \tilde{\mathbf{B}}^a + b_s), \tag{3.3}$$

$$\mathcal{A}^a = \delta(\mathbf{w}_m^\top \mathbf{S}^a + b_m) \circ \sigma(\mathbf{w}_n^\top \mathbf{S}^a + b_n), \tag{3.4}$$

where $\mathbf{S}^a$ is a vector with actor components and $\mathcal{A}^a$ is score of being an actor in a video clip. $\{\mathbf{w}_s, \mathbf{w}_m, \mathbf{w}_n, b_s, b_m, b_n\}$ are the learnable parameters and $\{\sigma(\cdot), \delta(\cdot)\}$ are sigmoid and softmax functions to calculate the attention values that ranges between 0 and 1. **ReLU** is the non-linearity function and $\circ$ represents element-wise multiplication. Finally, we have recalibrated the obtained attention map with feature tensor $\mathbf{B}$, which is formulated by:

$$\boldsymbol{\Theta}^a = [\mathfrak{a}_1^a + \mathfrak{a}_1^a \times \mathbf{b}_1, \cdots, \mathfrak{a}_{\mathcal{T} \times H \times W}^a + \mathfrak{a}_{\mathcal{T} \times H \times W}^a \times \mathbf{b}_{\mathcal{T} \times H \times W}], \tag{3.5}$$

where $\boldsymbol{\Theta}^a$ is the actor-attended spatiotemporal feature tensor.

We have observed that the regions where objects are present contain useful information for action recognition. However, small objects create ambiguity and leads to predict wrong action class. For instance, peeler and knife must leads to perform peeling and cutting action respectively. Therefore, we have incorporated the attention mechanism to focus on object-based contextual information. In object branch, we have pooled 3D feature map $\mathbf{B}$ temporally and channel-wise to obtain the spatial information of objects present in video clip. The score value of being an object element is formulated by:

$$\mathcal{A}^o = \sigma(\mathbf{w}_{\mathfrak{a}}^\top \mathbf{ReLU}(\tilde{\mathbf{B}}^o) + b_{\mathfrak{a}}), \tag{3.6}$$

$$\tilde{\mathbf{B}}^o = \frac{1}{\mathcal{T} * \zeta} \sum_{i \in \zeta} \sum_{j \in \mathcal{T}} \mathbf{B}(j, i), \tag{3.7}$$

where $\mathcal{A}^o$ is the confidence score of being object component present in video clip. $\tilde{\mathbf{B}}^o$ is the pooled feature vector for object component. $\{\mathbf{w}_{\mathfrak{a}}^\top, b_{\mathfrak{a}}\}$ are learnable parameters. Finally, we have recalibrated the attended maps $\mathcal{A}^o$ with the original feature tensor to reflect the features of objects, which is given by:

$$\mathbf{\Theta}^o = [\mathfrak{a}_1^o + \mathfrak{a}_1^o \times \mathbf{b}_1, \cdots, \mathfrak{a}_{H\times W}^o + \mathfrak{a}_{H\times W}^o \times \mathbf{b}_{H\times W}], \tag{3.8}$$

where $\mathbf{\Theta}^o$ is the object-attended spatiotemporal feature tensor.

Further, we have infer that which object is co-occur with actor and leads to perform the action by utilizing activity-object interaction mechanism. The obtained feature tensor from actor and object branches are given as the inputs to the interaction function for extracting associated object information that co-occur with actor. An interaction between $j - th$ object and $i - th$ actor is obtained through the relation functions as follows:

$$\mathbf{\Theta}^q = \widehat{\mathbf{\Theta}^q} + \mathbf{\Theta}^a, \tag{3.9}$$

$$\widehat{\mathbf{\Theta}^q} = \mathbf{\Phi}[\sum_i \sum_j \mathbf{\Gamma}[\mathbf{\Theta}^a(i), \mathbf{\Theta}^o(j), H]], \tag{3.10}$$

where $\mathbf{\Theta}^q$ is the actor-object attended feature tensor and $\widehat{\mathbf{\Theta}^q}$ is feature representation of activity-object interaction. $\{\mathbf{\Phi}[\cdot], \mathbf{\Gamma}[\cdot]\}$ are MLP layers to obtain two 1D feature representations. This represents whether an object and an actor have maximum interaction value given the knowledge of action class. H is feature representation of the prior likelihood of an activity-object interaction class that appear in a video, which is obtained from pretrained model on ImageNet, similar as in [128].

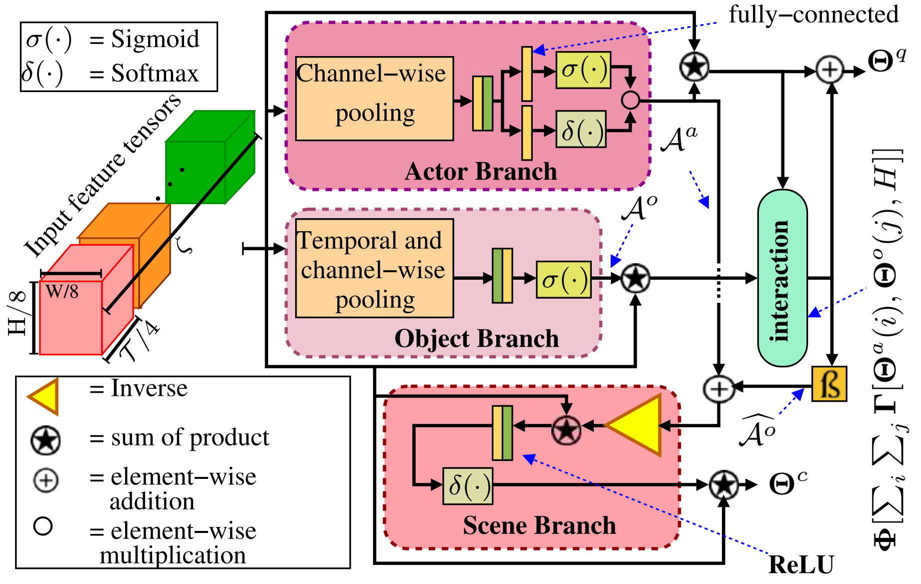


Figure 3.6: Architecture overview of AOSA network. Feature tensor $\mathbf{B}$ is given as an input to actor-object and scene branch. Channel-wise pooling is performed in actor branch to extract spatiaotemporal features. Temporal and channel-wise pooling is performed in object branch to extract visual saliency information. Green and yellow vertical rectangle depicts ReLU activation function and fully-connected layer respectively. **ß** is thresholdedReLU layer followed by point-wise convolution operation.

Scene Branch. Contextual cues also play a vital role to recognize the action in a video clip. However, the classifier may get confused to draw a decision boundary if an action and scene cues are different. For example, a person is smoking in a cricket field and due to large contextual information an action classifier will ignore the actual action (smoking). Thus, we have focused on the underlying background irrespective of the actor and object. In scene branch, first we have added attention maps obtained from actor and activity-object interaction branches to obtain actor-object attention map, which is calculated as follows:

$$\mathcal{A}^z(i) = \mathcal{A}^a(i) + \widehat{\mathcal{A}^o}(i), \tag{3.11}$$

where $\mathbf{A}^z$ is an output attention map with the features of actor-object and $i \in \{1, \cdots, \mathcal{T}HW/256\}$. We obtained an attention map of activity-object interaction through thresholdedReLU and point-wise convolution operation:

$$\widehat{\mathcal{A}^o} = ß(\widehat{\boldsymbol{\Theta}^q}), \tag{3.12}$$

$ß(\cdot)$ represents thresholdedReLU function with point-wise convolution operation to one channel.

The attention map in the scene branch is formulated by:

$$\mathbf{A}^c = \mathbf{1} - \mathcal{A}^z, \tag{3.13}$$

where $\mathbf{A}^c \in [0, 1]$ represents the probability of each location being a contextual component in a video clip and $\mathbf{1} \in [1]^{1 \times HW/256}$ is the vector with all ones. The score map $\mathbf{A}^c$ is recalibrated with original feature tensor $\mathbf{B}$ to reflect the scene information, which is given by:

$$\widehat{\mathbf{A}^c} = [\mathfrak{a}_1^c + \mathfrak{a}_1^c \times \mathbf{b}_1, \cdots, \mathfrak{a}_{H\times W}^c + \mathfrak{a}_{H\times W}^c \times \mathbf{b}_{H\times W}] \tag{3.14}$$

where $\widehat{\mathbf{A}^c}$ is the scene attended feature tensor. Further, we have calculated the score values of class-aware attention map to assign a higher weight to the relevant regions that possess high discriminant contextual-based information. The class-aware attention map is calculated by:

$$\boldsymbol{\Upsilon}^c = \delta(\mathbf{w}_\gamma^\top \mathbf{ReLU}(\widehat{\mathbf{A}^c}) + b_\gamma), \tag{3.15}$$

where $\boldsymbol{\Upsilon}^c$ is class-aware attention map and $\{\mathbf{w}_\gamma, b_\gamma\}$ are learnable parameters. Finally, we have calculated the scene feature representation of a video clip, which is formulated by:

$$\boldsymbol{\Theta}^c = [\boldsymbol{\gamma}_1^c + \boldsymbol{\gamma}_1^c \times \mathbf{b}_1, \cdots, \boldsymbol{\gamma}_{H\times W}^c + \boldsymbol{\gamma}_{H\times W}^c \times \mathbf{b}_{H\times W}], \tag{3.16}$$

where $\boldsymbol{\Theta}^c$ is feature representation of scene cues and $\boldsymbol{\gamma}_i^c$ is the $i-th$ element of attention map $\boldsymbol{\Upsilon}^c$.

We obtained spatiotemporal features with detailed information of the action, object, and contextual cues from our backbone network and attention network. However, the long temporal context that lasts for more than a second is not captured by our network. For example, 'bating in cricket' and 'bating in baseball' can only be distinguished by considering long-range temporal context since the action of the actor and the underlying background including the associated object 'bat' appear almost similar in a short duration video. To include the knowledge of interpreting actions for a long temporal duration including valuable context information depending on 'what has happened earlier', we need to incorporate more sophisticated schemes, like LSTM.

3.4.3 Long-Range Context (LRC) Network

In previous works [116,129], fully-connected LSTM (FC-LSTM) is incorporated to learn the long-range temporal relationships within the frames of the videos. The vectorized features are given as an input to learn temporal features. However, it depletes the spatial information over time. Therefore, the long-range temporal and spatial cues of human actions in videos is preserved by utilizing ConvLSTMs in the proposed deep network. ConvLSTM [130] is an extension of FC-LSTM that captures the relationship between the features spatially and temporally. The inherent convolutional trait helps in obtaining the features in both the input-to-state and state-to-state transition. Formally, it consists of the hidden states $\{\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_t\}$, the inputs $\{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_t\}$, and the cell outputs $\{\mathbf{C}_1, \mathbf{C}_2, \cdots, \mathbf{C}_t\}$ with four gates. The forget, input, output, and memory cells are represented by $\mathbf{f}_t$, $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{g}_t$, respectively. Each input, cell output, hidden state, and gate are 3D tensors whose last two dimensions are spatial information. The input tensor $\mathbf{X}_t$ and hidden state $\mathbf{H}_{t-1}$ at time t and $t-1$, respectively, are given as input to ConvLSTM, which determines the future state of a cell $\mathbf{C}_t$.

In LRC network, we have reformulated inputs of ConvLSTM to estimate the salient features from different video clips and preserve long temporal extent of actions. The actor-object $\boldsymbol{\Theta}^q$ and context $\boldsymbol{\Theta}^c$ feature tensors, which are obtained from AOSA network are given as an input to modified ConvLSTM, as shown in Figure 3.2. The feature tensor $\boldsymbol{\Theta}^q_t$ and $\boldsymbol{\Theta}^c_t$ are used for encoding the actor-object and context cues of the video clips at time t, respectively, is given by:

$$\boldsymbol{\Theta}^{\varphi}_t = \{\boldsymbol{\Theta}^{\varphi}_t(1), \boldsymbol{\Theta}^{\varphi}_t(2), \cdots, \boldsymbol{\Theta}^{\varphi}_t(H/8 \times W/8 \times \zeta)\}, \tag{3.17}$$

where $\boldsymbol{\Theta}^{\varphi}_t$ is feature tensor and φ represents either q or c, at each time steps t. $t = 1, 2, \cdots, \mathcal{T}'$ and $\mathcal{T}' = \mathcal{T}/4$ is the recurrent time step of modified ConvLSTM. In case of learning long-range temporal extents of $\boldsymbol{\Theta}^q_t$ feature tensor, we have reinforced ConvLSTM with spatiotemporal attention mechanism. Similarly, we have reformulated the equations of ConvLSTM for context feature tensors $\boldsymbol{\Theta}^c_t$ by including temporal attention mechanism. Note that ConvLSTM share the similar weights in both actor-object and scene branches.

ConvLSTM remembers the past sequence at time t as it accumulate the previous information in memory cell. However, the motion of objects are not taken into consideration that happens from time t to $t+1$, as it may leads to noisy prediction in spatiotemporal action recognition. For instance, 'chopping' and 'slicing' can be recognized only after tracking consecutive timesteps of a knife. Therefore, we reformulated inputs of ConvLSTM spatially and temporally to capture motion of object and actor simultaneously.

In actor-object branch, first we estimate the relevant locations of $\boldsymbol{\Theta}_t^q$ in space and time by incorporating a spatiotemporal attention mechanism with the guidance of $\mathbf{H}_{t-1}^q$, which is given by:

$$\mathfrak{R}_t = \mathbf{Z}_r * \mathbf{tanh}(\mathbf{Z}_\Theta * \boldsymbol{\Theta}_t^q + \mathbf{Z}_h * \mathbf{H}_{t-1}^q + b_r), \tag{3.18}$$

where $\mathfrak{R}_t$ is the un-normalized spatiotemporal attention weight map of $\boldsymbol{\Theta}_t^q$ at t steps. $\{\mathbf{Z}_r, \mathbf{Z}_\Theta, \mathbf{Z}_h, b_r\}$ are spatiotemporal attention parameters. Second, we normalize $\mathfrak{R}_t$ by using standard softmax function, which is formulated by:

$$\mathfrak{A}_t^q(i,j) = \frac{\mathbf{exp}(\mathfrak{R}_t(i,j))}{\sum_t^{\mathcal{T}/4} \sum_i^{H/8} \sum_j^{W/8} \mathbf{exp}(\mathfrak{R}_t(i,j))}, \tag{3.19}$$

where $\mathfrak{A}_t^q$ is normalize attention weight map that reflects the spatial and temporal importance. Next, we apply $\mathfrak{A}_t^q$ on feature map $\boldsymbol{\Theta}_t^q$ to obtain spatiotemporal attended video-level features is estimated by:

$$\widehat{\boldsymbol{\Theta}_t^q} = \mathfrak{A}_t^q \circ \boldsymbol{\Theta}_t^q, \tag{3.20}$$

where $\widehat{\boldsymbol{\Theta}_t^q}$ is the feature tensor with attention in space-time dimension and $\circ$ is element-wise multiplication.

In context branch, the temporal attention is incorporated with ConvLSTM to estimate video-level weights of underlying background over time. The un-normalize attended weights of context feature tensors $\boldsymbol{\Theta}_t^c$ under the guidance of previous hidden state $\mathbf{H}_{t-1}^c$ is estimated as:

$$\mathfrak{Z}_t = \mathbf{Z}_z * \mathbf{tanh}(\mathbf{Z}_\Theta * \boldsymbol{\Theta}_t^c + \mathbf{Z}_h * \mathbf{H}_{t-1}^c + b_z), \tag{3.21}$$

where $\mathfrak{Z}_t$ is the un-normalize feature tensor with attended weights and $\{\mathbf{Z}_z, \mathbf{Z}_\Theta, \mathbf{Z}_h, b_z\}$ are temporal attention parameters. Next, we use softmax to normalize $\mathfrak{Z}_t$, which is computed by:

$$\mathcal{O}_t^c(i) = \frac{\mathbf{exp}(\mathfrak{Z}_t(i))}{\sum_i^{\mathcal{T}/4} \mathbf{exp}(\mathfrak{Z}_t(i))}, \tag{3.22}$$

where $\mathcal{O}_t^c$ is normalize attention weight map that reflects the temporal importance of spatiotemporal features. $\mathcal{O}_t^c$ is element-wise multiply with $\mathbf{\Theta}_t^c$ to obtain temporal attended video-level feature tensor, is given by:

$$\widehat{\mathbf{\Theta}_t^c} = \mathcal{O}_t^c \circ \mathbf{\Theta}_t^c, \tag{3.23}$$

where $\widehat{\mathbf{\Theta}_t^c}$ is the video-level feature tensor with attention weights.

Finally, we feed the attended feature maps into ConvLSTM as an attended input at time t, is given by:

$$\mathbf{f}_t^\varphi = \sigma(\mathbf{Z}_{\Theta_f}^\varphi * \widehat{\mathbf{\Theta}_t^\varphi} + \mathbf{Z}_{hf}^\varphi * \mathbf{H}_{t-1}^\varphi + b_f^\varphi), \tag{3.24}$$

$$\mathbf{i}_t^\varphi = \sigma(\mathbf{Z}_{\Theta_i}^\varphi * \widehat{\mathbf{\Theta}_t^\varphi} + \mathbf{Z}_{hi}^\varphi * \mathbf{H}_{t-1}^\varphi + b_i^\varphi), \tag{3.25}$$

$$\mathbf{o}_t^\varphi = \sigma(\mathbf{Z}_{\Theta_o}^\varphi * \widehat{\mathbf{\Theta}_t^\varphi} + \mathbf{Z}_{ho}^q * \mathbf{H}_{t-1}^\varphi + b_o^\varphi), \tag{3.26}$$

$$\mathbf{g}_t^\varphi = \mathbf{tanh}(\mathbf{Z}_{\Theta g}^\varphi * \widehat{\mathbf{\Theta}_t^\varphi} + \mathbf{Z}_{hg}^\varphi * \mathbf{H}_{t-1}^\varphi + b_g^\varphi), \tag{3.27}$$

$$\mathbf{C}_t^\varphi = \mathbf{f}_t^\varphi \circ \mathbf{C}_{t-1}^\varphi + \mathbf{i}_t^\varphi \circ \mathbf{g}_t^\varphi, \tag{3.28}$$

$$\mathbf{H}_t^\varphi = \mathbf{o}_t^\varphi \circ \mathbf{tanh}(\mathbf{C}_t^\varphi), \tag{3.29}$$

where φ is either actor-object or scene branch and $\{*, \circ\}$ denotes convolution operator and Hadamard product. The sets of $\mathbf{Z}$ and b are shared the parameters of ConvLSTM both in actor-object and scene branch to reduce the computational cost including model complexity. $\sigma(\cdot)$ and $\mathbf{tanh}(\cdot)$ are sigmoid and tanh functions, and $\{\mathbf{i}_t^\varphi, \mathbf{o}_t^\varphi, \mathbf{g}_t^\varphi\}$ are input, output gates, and memory cells respectively. $\mathbf{C}_t^\varphi$ is cell output state at $t-th$ time step, $\mathbf{H}_{t-1}^\varphi$ is previous the hidden state, and $\widehat{\mathbf{\Theta}_t^\varphi}$ is current spatiotemporal input feature tensors. Inspired by [131], we have used initialization strategy for faster convergence of hidden state and ConvLSTM cell state. The initial hidden state and cell output of the ConvLSTM are calculated by an average of the feature tensor given by two layer CNNs followed by batch normalization (**BN**), given by:

$$\mathbf{C}_0^{\varphi} = g_c(\sum_{i \in \mathcal{T}'} \frac{\Theta_i^{\varphi}}{\mathcal{T}'}), \tag{3.30}$$

$$\mathbf{H}_0^{\varphi} = g_h(\sum_{i \in \mathcal{T}'} \frac{\Theta_i^{\varphi}}{\mathcal{T}'}), \tag{3.31}$$

where $\mathbf{C}_0^{\varphi}$ and $\mathbf{H}_0^{\varphi}$ are initial cell output and hidden state. $\{g_c, g_h\}$ are represented by **conv** $1 \times 1 \times 1 \rightarrow$ **conv** $1 \times 1 \times 1 \rightarrow$ **BN**. We obtain current hidden state in both branches, *i.e.,* $\mathbf{H}_t^q$ and $\mathbf{H}_t^c$ at each recurrent time step. The output of the hidden state $\mathbf{H}_t^q$ and $\mathbf{H}_t^c$ over the time $\mathcal{T}'$ are summarized to obtain the actor-object and context information simultaneously. These summarized hidden states are formulated by:

$$\mathcal{H}^{\varphi} = \sum_{t=1}^{\mathcal{T}'} \frac{\mathbf{H}_t^{\varphi}}{\mathcal{T}'}, \tag{3.32}$$

where $\mathcal{H}^{\varphi}$ is the summarized hidden states of actor-object or context feature tensor. These hidden states are fused through element-wise addition to obtain $\mathfrak{Q} \in \mathbb{R}^{\mathcal{T}/4 \times H/8 \times W/8 \times \zeta}$ and is fed into temporal pooling layer to reduce the dimension of the output feature tensor.

3.4.4 Class-aware Temporal Attention Pooling (CTAP) Network

The recurrent layers can capture temporal information with hidden states however these temporal cues contain redundant information. For instance, ‘a person is sitting on a boat’, there are only minor differences in a series of consecutive frames of a video. Therefore, the idea of temporal feature pooling has been extensively incorporated to minimize the redundancy and processing time for videos with long time period [37]. In temporal pooling, statistical methods are implemented within various local windows over time to apprehend motion information of videos. In our architecture, we have incorporated temporal pooling on the top of the LRC network over the time steps, as shown in Figure 3.2.

We propose class-aware temporal attention pooling mechanism to extract class-aware spatiotemporal feature tensors and concentrate on relevant information over time. This is useful for discriminating complex human actions in long sequence of a video. We have used the concept of attention pooling to extract semantic information which is given as follows:

Attention pooling: It extracts important semantic information that changes over time on the video clip by calculating attention of feature tensor $\mathfrak{Q}_t$ in $t-th$ time step. The attention feature tensor $\mathbf{\Omega}_t^{(att)}(k)$ of $\mathbf{k}-th$ class is given by:

$$\mathbf{\Omega}_t^{(att)}(k) = [\mathbf{1}^{\mathsf{T}}\mathbf{E_k}]^{\mathsf{T}}, \tag{3.33}$$

$$\mathbf{E_k} = \mathbf{Y_k} \circ \mathbf{U}, \tag{3.34}$$

$$\mathbf{Y_k} = \mathfrak{Q}_t\mu_k, \tag{3.35}$$

$$\mathbf{U} = \mathfrak{Q}_t\varsigma, \tag{3.36}$$

where $\mathbf{k}$ denotes the number of classes and the attention feature tensors $\{\mathbf{Y_k}, \mathbf{U}\} \in \mathbb{R}^{\tau\times 1}$. $\mathbf{1} \in \mathbb{R}^{\tau\times\eta}$ is a tensor of all ones, where $\eta = H/8 \times W/8 \times \zeta$ and $\{\mu_{\mathbf{k}}, \varsigma\} \in \mathbb{R}^{\eta\times 1}$ are class-specific and class-agnostic tensors. The class-specific is deduced via a dense layer with sigmoid classification and recalibration. The class-agnostic tensor is computed by adding non-linearity using ReLU activation. Most deep learning models fed the vectorized feature of the last convolution layer into fully-connected layers followed by a softmax layer. The fully-connected layers, however, are prone to overfit the overall network. Therefore, GAP layer is utilized to minimize overfitting by reducing the total number of parameters in the model. Finally, we have cascaded a softmax classification layer for supervised training. The layer calculates the probability score of predicted action class, which is given as:

$$\hat{\mathbf{y}}_i = \phi(\mathbf{w}\boldsymbol{\Omega} + \mathbf{b}), \tag{3.37}$$

where $\mathbf{w}, \mathbf{b}$ are learnable parameters and $\phi(\cdot)$ is softmax function. $\hat{\mathbf{y}}_i$ is probability distribution that indicate the likelihood of a video belong to each $i - th$ action class.

3.4.5 Joint Training Model

Our FactorNet consist of MDB, AOSA, and LRC networks to recognize action in a video clip. The optimization of overall architecture is difficult, thus we introduce a joint training mechanism in end-to-end fashion. We have formulated the overall objective function of our proposed architecture with categorical cross entropy loss for video clip, given as follows:

$$\mathbf{L}_{\mathbf{overall}} = \mathbf{L}_{cls} + \lambda_1 \mathbf{L}_s + \lambda_2 \mathbf{L}_o, \tag{3.38}$$

where $\{\mathbf{L}_{cls}, \mathbf{L}_s, \mathbf{L}_o\}$ are the loss functions of classification, scene, and object. $\{\lambda_*\}$ are the regularization terms and balance the contribution of two regularization terms. We have used the categorical cross entropy loss function to measure the compatibility between the obtained distribution and ground-truth action class labels for calculating classification loss:

$$\mathbf{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N}[\mathbf{y}_i \cdot \mathbf{log}\ \hat{\mathbf{y}}_i], \tag{3.39}$$

where, $-$ sign ensures that the loss gets smaller when the distributions get closer to each other. N is the number of training videos. $\{\mathbf{y}, \hat{\mathbf{y}}\}$ are ground-truth action class labels and predicted labels. Further, we have calculated scene loss function, which is defined as the log probability of features obtained from backbone network. The function to obtain scene loss is given by:

$$\mathbf{L}_s = -\sum_{i=1}^{S} \{\mathbf{p}_i \cdot \mathbf{log}\ \hat{\mathbf{p}}_{i;\theta_m}\}, \tag{3.40}$$

$$\hat{\mathbf{p}}_{i;\theta_m} = \delta(\mathbf{w}\boldsymbol{\Theta}^c + \mathbf{b}), \tag{3.41}$$

where, $\{\theta_m, \mathbf{w}, \mathbf{b}\}$ are model parameters and S is number of scene classes. $\hat{\mathbf{p}}_{i;\theta_m}$ is the distribution of each $i-th$ class given the parameters (θ_m) of scene branch and $\delta(\cdot)$ is softmax operation. However, scene labels are not present with our given action class labels. Therefore, we have obtained a pseudo scene label $\mathbf{p}_i$ by running Places365 dataset pre-trained ResNet-50 [132] on the Kinetics dataset. Similarly, object loss function is defined as the log probability of features obtained from backbone network, which is given by:

$$\mathbf{L}_o = -\sum_{i=1}^{O} \{\mathbf{p}_i \cdot \mathbf{log}\ \hat{\mathbf{p}}_{i;\theta_m}\}, \tag{3.42}$$

$$\hat{\mathbf{p}}_{i;\theta_m} = \sigma(\mathbf{w}\boldsymbol{\Theta}^o + \mathbf{b}). \tag{3.43}$$

However, object labels are not present with our given action class labels. Therefore, we have obtained a pseudo object label $\mathbf{p}_i$ by running MS-COCO dataset pre-trained ResNet-50. $\sigma(\cdot)$ is sigmoid function and O is number of object classes.

3.5 Experimental Results

n this section, we perform an exhaustive set of experimentation and briefly report the implementation details of our FactorNet for training and testing. Then, we comprehensively examine the effect of different networks of our architecture to determine their impact on classification. We also extensively examine the different variation of hyper-parameters in our network. Finally, we compare with the state-of-the-art methods on both temporal-related benchmark datasets (*i.e.,* Something-Something V1, ActivityNet, and Breakfast Action) and scene-related publicly available datasets (*i.e.,* Kinetics400, UCF101, and HMDB51). We also proposed a new dataset, denoted by *FactNet*, which contains 38 action classes.

3.5.1 Datasets and Metrics

In our experiment, we elect six challenging, accessible, and publicly available video-based action recognition benchmark datasets, *i.e.,* UCF101, HMDB51, Kinetics400, Breakfast-Actions, Something-Something V1, and ActivityNet. The statistics of all the dataset is shown in Table 3.2. We also examine the performance of our architecture on FactNet dataset.

Table 3.2: Statistics of the datasets used in experimentation.

dataset		**#classes**	**#videos**	**avg. #clips/ #classes**	**avg. temporal length**	**resolution**	**fps**
scene	UCF101	101	13K	100	7.21s	320×320	25
	HMDB51	51	7K	101	1s	240×240	30
	Kinetics400	400	28K	137	10s	340×340	25
motion	Breakfast-Action	48	1K	10	2.3m	-	15
	SSV1	174	100K	620	4.03s	240×240	15
	ActivityNet	203	27K	137	5m	1280×720	30
Ours	FactNet	38	8K	80	60s	Multi-scale	30

UCF101 [1]. It consists of 101 action classes that include 13k clips and 27 hours of video data. It covers a broad range of actions, such as, sports, human-human interaction, and activity-object interaction. The video instances include variation in the background, viewpoints, and severe camera motion including object appearance, scale, and pose.

HMDB51 [2]. This dataset consists of 51 different action classes with 7K videos collected from movies and Youtube. The actions classes are grouped into 5 sub-categories: general facial actions, facial actions with object manipulation, general body movements, body movements with object interaction, and body movements for human interaction, respectively. HMDB51 contains 4.9K training and 2.1K testing videos.

Kinetics400 [133]. This human action video dataset have trimmed videos of 400 action classes taken from YouTube. Each class consists of at least 400 video clips with an average temporal length of 10 seconds. It covers a broad scope of classes, *i.e.*, activity-object interactions and human-human interactions. It is split into 240K training, 20K validation, and 40K test videos.

Something-Something V1 (SSV1) [134]. It is a large group of 100K densely-labeled video clips that show humans performing predefined necessary actions with everyday objects. It contains 174 action classes with training, validation, and test videos.

ActivityNet [135]. This dataset covered a large range of complex human daily living actions. It includes 203 classes of action with 193 video clips/class on average and 849 hours of video data. The average length of videos is between 5 and 10 minutes. The dataset consists of 27.8K trimmed videos. We split 50% of data into training, 25% for validation, and 25% for testing.

Breakfast-Actions [136]. It is a dataset for unscripted cooking-oriented human activities with 10 different cooking activities and 66.7 hours of video data. These activities are performed by 52 different actors in 18 different kitchen locations. These activities are composed from various actions, which are annotated manually in 48 different action classes. Breakfast-Actions consist of 1712 videos with 1357 for training and 335 for test.

FactNet. Learning long-range spatial-temporal features with respect to objects and scene are critical for many video analysis tasks. To train a model using publicly available benchmark datasets certainly may capture and leverages representation bias. The characterization of supervised learning may not be efficient to new action classes. None of the publicly available benchmark dataset has ambiguous conjugate action pairs, especially taking co-occurring objects in consideration. This motivated us to train our network with sufficient ambiguous conjugate action pairs. We have proposed our dataset, which is known as FactNet. It is dataset with 38 classes and each class has an average of 80 videos. The average length of trimmed videos is 60 seconds, which are collected from YouTube. The spatial resolution of the videos ranges from 480×480 to 1080×1080. It covers a large range of classes consisting of activity-object-scene related actions that occur in daily life. All types of variation are included in our dataset, i.e., variations in camera motion, size of objects, background, illumination, and viewpoints, respectively. We split 50% of the dataset for training, 25% for validation, and 25% for testing. The annotation is done by crowdsourcing. We have provided the statistic of our dataset in Table 3.3. The confusion matrix of the dataset is shown in Table 3.4 for few FactNet action classes, which is performed on our FactorNet.

Table 3.3: Statistics of FactNet dataset.

Specification	**Values**
#action	38
#videos	8K
Average #clips/classes	200
Average temporal length	60s
Min resolution	480
Max resolution	1080
Frame rate	30
Total duration	135 hours
Mean video length	25s
Min video length	10s
Max video length	2 min
Audio	Yes
Background	Yes
Camera Motion	Yes

We have used Top-1 and Top-5 metrics for Kinetics400, Top-1 for Something-Something V1 and Top-3 for ActivityNet datasets. We have used mean accuracy (Acc.) for UCF101, HMDB51, and BreakFast-Actions datasets. We also measure the computational complexity of the proposed network in terms of number of training parameters (in short params) and FLOP counts.

EA	84.3	15.7	0.0	0.0	0.0	0.0
CA	18.2	81.8	0.0	0.0	0.0	0.0
SB	0.0	0.0	82.8	17.2	0.0	0.0
RB	0.0	0.0	20.6	79.4	0.0	0.0
TL	0.0	0.0	0.0	0.0	83.7	16.3
TM	0.0	0.0	0.0	0.0	18.7	81.3
	EA	**CA**	**SB**	**RB**	**TL**	**TM**

Table 3.4: The confusion matrix of FactNet dataset (performed on FactorNet). We have used the abbreviations of FactNet classes. EA- eating an apple, CA-cutting an apple, SB-sitting on a boat, RB-rowing a boat, TL-typing on the laptop, TM-typing on mobile.

3.5.2 Implementation Details

The implementation details of different modules of FactorNet are briefly described as follows:

RGB inputs. We extract frames of UCF101 and Kinetics400 at 25 fps (frames per second). In case of ActivityNet, HMDB51, and FactNet datasets, video frames are fetched at 30 fps and for Breakfast-Action and Something-Something V1 datasets, the frames are extracted at 15 fps. The purpose for taking different fps is to make sure that one video clip contains enough information to interpret motion as different datasets have different motion speeds. We have used 16 consecutive RGB frames in our overall experiment to maintain reasonable training time. In addition, we have also experimented with temporal length of $\{8, 32, 64, 128\}$ RGB frames considering the same data augmentation. We have added padding for videos with shorter length to satisfy the input given to FactorNet architecture.

Multi-scale Deformable Backbone Network. First, we have initialized the parameters of eight 3D convolution layers (conv_1 to conv_5b) using a pre-trained model "C3D" on Sport-1M datasets (similar as in [12]) and fixed them during training. The parameters of deformable layers from deform_1 to deform_3 are learned using stochastic gradient descent with momentum during training.

Actor-Object-Scene Attention Network. We have three loss functions that aims to encourage the action-object and scene branch to learn the respective feature representation. The feature representation of object, scene, and activity-object interaction are learn through pseudo labels. First, we train AOSA separately and then fine-tune with the specific dataset to obtain the features of object, scene, and activity-object interaction respectively. At the time of training, we freeze the MDB network and only train the AOSA.

Long-Range Context Network. We have included a single convLSTM layer with 1024 hidden state dimensions with a sequence length of $\mathcal{T}/4$ in actor-object and scene branch, respectively. At the initial stage of training, only LTC network is trained, while the MDB and AOSA networks are frozen.

Training. We develop our FactorNet architecture in Keras API on the top of TensorFlow API. We have resized the extracted frames for all the datasets. At training, data augmentation is performed to avoid over-fitting by cropping a random clip. We use input spatial resolution of 112×112 and temporal length of $\mathcal{T}$ of 16 during training. The batch size of 30 clips is considered due to the limitation in GPU memory. We computed all the parameters and train our model on same machine with a 8 GTX 1080TI GPU. In our work, we have trained the proposed network in four stages. We first train the MDB and freeze the AOSA, and LRC modules. In second stage, we train the AOSA network in a semi-supervised way and freeze the other modules to learn accurate action information with respect of object and scene. In third stage, we fixed the weights of MDB and AOSA modules and train LRC to learn long-range temporal dependencies. Finally, we freeze the learnable weights of backbone, AOSA, and LRC modules and train the overall architecture respectively. The batch normalization layers are enabled during training only.

Hyper-parameters In case of scene-related datasets, we start training with a learning rate of 0.01 and reduce the rate by a factor of 10 for every $45th$ epochs. For temporal-related datasets and our dataset we start training with learning rate of 0.001. The learning rate is reduced by a factor of 10 at every $30th$ epoch. We use Adam as an optimizer with a momentum of 0.9 and a weight decay of 10^{-4}. The weights are initialized with He initializer. We use the dropout layer between the conv_3b and batch normalization layer in fish-tail block and set the dropout rate to 0.7. Similarly, a dropout layer is added on the top of fish-head block and global pooling layer, which is set to 0.5 to alleviate over-fitting.

Inference. We rescale the shorter dimension of video to 112 pixels, as provided in [137]. In practice, we usually sample 10 clips evenly from the full-length video and compute the softmax scores individually (similar as [138]). The final prediction is the averaged modified softmax scores of all clips.

3.5.3 Ablation Study

In this section, we examine abundant ablation studies to estimate recognition accuracy and computational overhead of the proposed architecture.

Temporal Window Size. The performance of our FactorNet is influenced by temporal range of a video clip. We analyze the impact of the number of RGB frames in an input video clip on UCF101, HMDB51, and FactNet datasets. In the Table 3.5, we mutate our MDB with different number of temporal length from 8 to 128 and depicts the mean accuracy along with the FLOP count. The conclusion drawn after experiments are as follows: I.) with the increase in the number of frames, the FLOP counts increase and II.) there exist a trade-off between mean accuracy and computational complexity. It is clear from the table that the temporal range of 16 frames provides high accuracy with optimal number of parameters.

Effect of different backbone network variants. We have designed a multi-scale deformable backbone network that extracts rich spatial features with optimal number of layers. The network is kept shallow in Fish-tail block to restrict the number of training parameters. Table 3.6 shows the effect of different model, such as 3DResNet, I3D, and

Table 3.5: Effect of temporal range in our proposed architecture for UCF101 and FactNet datasets.

#frames	**UCF101 Acc.**	**FactNet Acc.**	**FLOPs (G)**
8	96.5	75.1	1 ×
16	**98.0**	**87.7**	1.2 ×
32	96.0	80.7	1.6 ×
64	92.0	70.1	2.0 ×
128	89.4	68.2	2.1 ×

C3D models as our backbone in Fish-tail block and keeping rest of the architecture same. The importance of the proposed backbone network and its variants in terms of mean accuracy for BreakFast-Actions and Top-1 for Kinetics400 are reported in Table 3.6. It is clear from the Table 3.6 that our backbone network outperforms other models with training parameters approx. similar to 3DResNet-34 [139]. We also investigate the impact of using top-down skip connection on the performance of the proposed network in Table 3.7. We have further experimented with number of layer required in Fish-body and Fish-head blocks, as shown in Figure 3.7. We can therefore conclude that three layers are adequate for extracting salient feature maps.

Table 3.6: Effect of backbone network variants, like 3DResNet, I3D, and C3D network as a MDB on Kinetics400 and Breakfast-Actions datasets. Please note that we have replace MDB with specific backbone network keeping the rest of architecture same.

Backbone Variations	**Kinetics400 Top-1**	**Breakfast-Actions Acc.**	**#params** ($\times 10^9$)
3DResNet-18 [139]	79.5	87.4	52.2
3DResNet-34 [139]	80.7	88.0	83.5
I3D [140]	81.7	88.3	32.5
C3D [12]	80.1	87.7	99
MDB	**83.7**	**93.5**	83

Impact on deformable convolution. Table 3.8 shows the comparison results for utilizing deformable convolution in MDB network. The result concludes that our baseline model is adequate to extract rich contextual information and enhance performance

Table 3.7: Performance of our multi-scale deformable backbone network with different variations on Something-Something V1 and ActivityNet datasets.

MDB Variations	Something-Something V1 Top-1	ActivityNet Top-3
MDB w/o Fish-body block	54.6	86.3
MDB w/o Fish-head block	53.9	85.9
MDB w/o skip connections	52.7	85.1
MDB	**55.3**	**87.9**

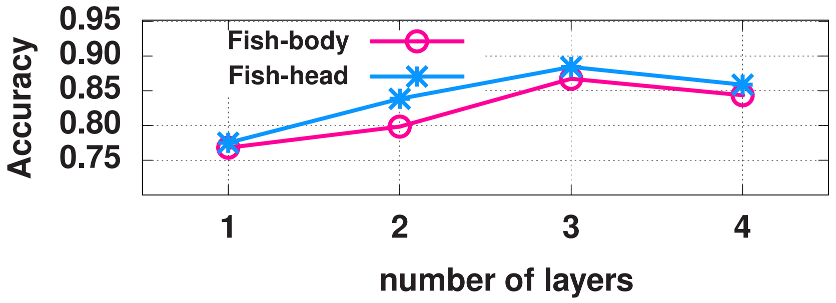


Figure 3.7: Accuracy vs. number of layers in Fish-body and Fish-head blocks.

to recognize the action for ActivityNet and FactNet datasets. We have also shown the results in Figure 3.4.

Table 3.8: Comparison of deformable convolution in backbone network on ActivityNet and FactNet datasets.

MDB Variations	ActivityNet Top-3	FactNet Acc.
w/o deformable convolution	85.5	77.6
w/ deformable convolution	**87.9**	**87.7**

Impact on AOSA network. We have shown the comparison results for pooling mechanism in actor and object branches in Table 3.9. In addition, Table 3.10 shows the comparison results for different sub-networks implementations on actor, object, and scene branch. We compare our FactorNet with scores obtained from actor and object branch. It is clear from the table that channel-wise pooling in actor branch performs better than temporal and channel-wise pooling.

Impact of LRC network. In Table. 3.11, we demonstrate the comparison results for

Table 3.9: Comparison of actor and object branches of AOSA network on ActivityNet and Something-Something V1 datasets.

AOSA Variations	**Pooling mechanism**	**ActivityNet**	**Something-Something V1**
		Top-3	**Top-1**
Actor Branch	Temporal-channel wise pooling	87.75	55.0
	Channel-wise pooling	**87.9**	**55.3**
Object Branch	Temporal-channel wise pooling	**87.9**	**55.3**
	Channel-wise pooling	85.0	54.7

Table 3.10: Comparison of AOSA on ActivityNet, Breakfast-Actions, and FactNet datasets.

AOSA Variations	**ActivityNet**	**Breakfast-Actions**	**FactNet**
	Top-3	**Acc.**	**Acc.**
w/o actor score	79.2	80.1	75.6
w/o object score	79.8	82.2	80.0
w/o scene score	80.8	85.2	80.1
w/o actor-object score	75.8	88.2	79.2
baseline	**87.9**	**93.5**	**87.7**

ConvLSTM on actor-object and scene branches on Breakfast-Actions and Something-Something V1 datasets. Our experiments show that spatiotemporal attention in actor-object branch achieves better accuracy to handle long-range temporal actions. In our experiment, we found that providing spatiotemporal attention to scene branch, leads to increase misclassification. Thus, we incorporate temporal attention with ConvLSTM in scene branch. We also study the usage of different hidden states for learning long-range spatiotemporal information, as shown in Figure 3.8. We can see 1024 hidden states perform better as compared to others.

Table 3.11: Comparison of actor-object and scene branch of LRC network on Breakfast-Actions and Something-Something V1 datasets.

LRC variations	**Attention mechanism**	**Breakfast-Actions Acc.**	**Something-Something V1 Top-1**
Actor Branch	Spatial	86.2	53.9
	Temporal	87.6	53.7
	None	72.6	52.3
	Both	**93.5**	**55.3**
Scene Branch	Spatial	88.1	54.6
	Temporal	**93.5**	**55.3**
	None	75.3	54.2
	Both	70.8	51.3

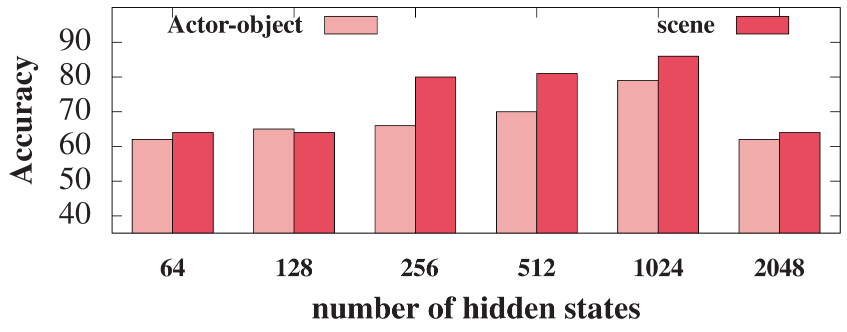


Figure 3.8: Accuracy vs. the number of hidden states.

Study on pooling mechanisms. In temporal attention pooling network, we explore four different temporal pooling mechanisms: max, average, standard deviation, and attention pooling. Table 3.12 shows the comparison results for these four different implementations. Our experiments show that using temporal attention pooling always achieves the best performance even on different datasets *i.e.* UCF101, BreakFast-Actions, and FactNet.

Table 3.12: Comparison of temporal pooling mechanisms on UCF101, BreakFast-Actions, and FactNet datasets.

Temporal Pooling variants	**UCF101 Acc.**	**BreakFast-Actions Acc.**	**FactNet Acc.**
max	93.8	84.9	76.3
average	94.0	85.3	78.7
standard deviation	95.8	89.5	79.2
attention	**98.0**	**93.5**	**87.7**

Impact of individual networks. Our proposed networks in FactorNet are MDB, AOSA, LRC, and CTAP. In Table 3.13, we investigate the accuracy of how well standalone networks can perform on the recognition task. We also examine the combination of our proposed networks on Breakfast-Actions, ActivityNet, and Something-Something V1 datasets to validate the contribution of each network combination in our FactorNet.

Table 3.13: Effect of individual modules in our architecture for ActivityNet, Breakfast-Actions, and Something-Something V1 datasets.

Model Variations	**ActivityNet Top-3**	**Breakfast** -Actions **Acc.**	**Something-** Something V1 **Top-1**
only MDB	79.1	85.3	48.3
only AOSA	79.9	86.1	49.2
only LRC	80.3	86.6	50.7
only CTAP	80.3	87.6	50.6
AOSA + LRC	83.7	88.1	52.7
AOSA + CTAP	84.3	88.2	53.4
LRC + CTAP	85.1	89.4	54.8
MDB+AOSA+LRC+CTAP	**87.9**	**93.5**	**55.3**

Study on different classes of FactNet. In this section, we validate the effectiveness of individual networks of our architecture on each action class of FactNet, as shown in Figure 3.11. It can be seen that different networks have different score values on a particular action. For instance, 'AOSA' works better in case of 'smoking cigar' and 'eating a roll' actions as compared to other networks, since most of the actions are focused on temporal context.

Visualization of AOSA network. In this section, we show the effect of each branch of AOSA network. The heat maps are computed for actor, object, and scene branches and are shown in Figure 3.9. To recognize a human action, it is important to attend to the human subjects, and this explains why the attention maps for action have higher weights on human subjects. However, not all humans in a video frame receive the same attention, and not all body parts of a human receive attention. To recognize co-occurring associated objects, it is necessary to deal with the co-occurring objects and their interaction with the human subject. Only the object which is interacting with human subjects has a higher weight. On the other hand, the weights of the scene maps emphasize the background regions with non-uniform distribution. We have visualized the effect of using softmax, sigmoid, and their combination in Figure 3.9.

3.5.4 Comparison with State-of-the-Art

In this section, we compare our architecture with the state-of-the-art approaches [11, 12, 14, 30, 31, 114–117, 121, 129, 137, 138, 140–152] on six different benchmark datasets. We also compare with two-stream networks considering both RGB and Optical flow modalities for action recognition on UCF101 and HMDB51 datasets.

Results on UCF101, HMDB51, and Kinetics400 datasets. We compare the performance of our FactorNet with state-of-the-art methods on UCF101, HMDB51, and Kinetics400 datasets. Table 3.14 shows the results performed on UCF101 and HMDB51 datasets that consider to have scene-related videos. It is clear from Table 3.14 that our network outperforms the recent approaches on UCF101 dataset with an improvement of

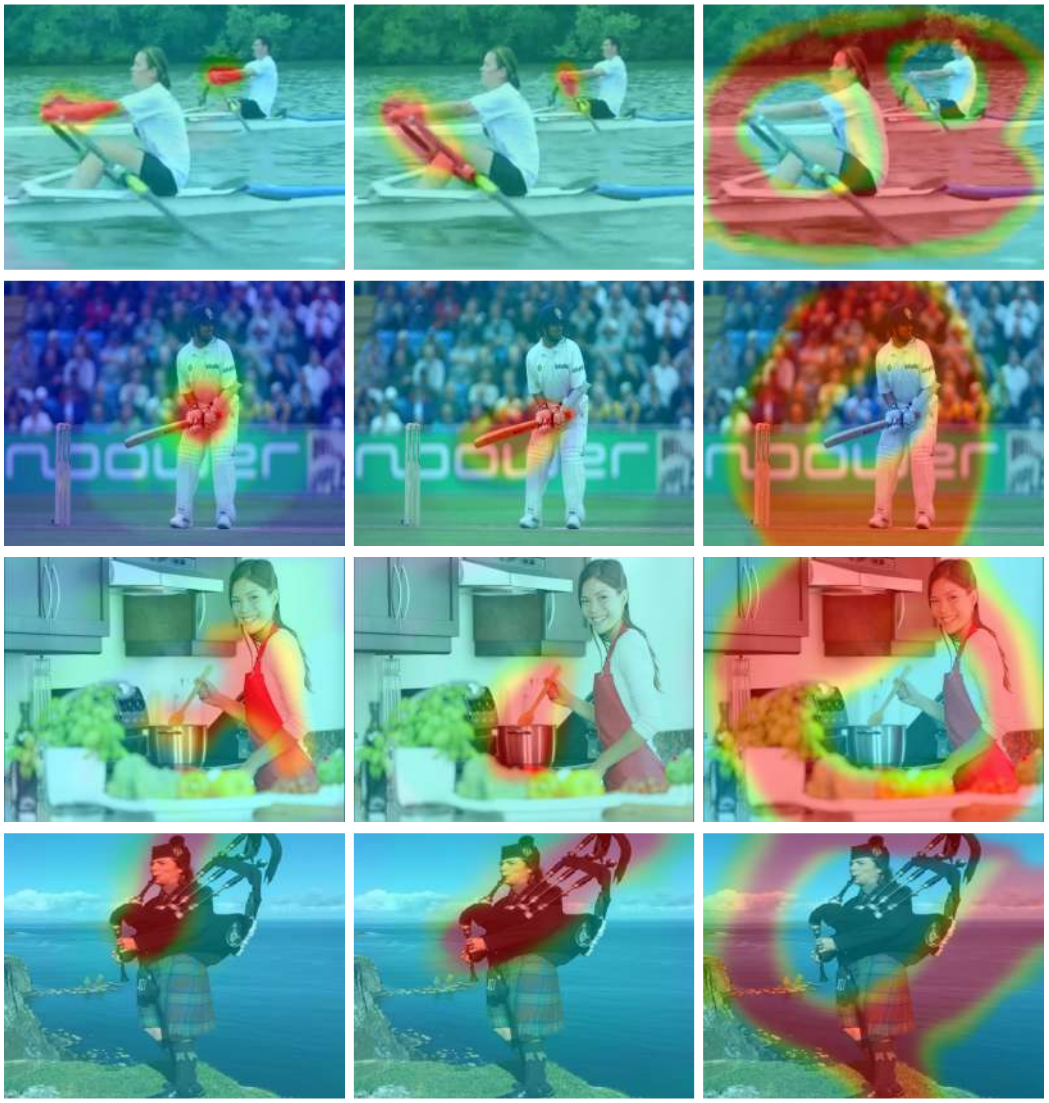


Figure 3.9: Video frames from different video clips of FactNet dataset. Visualizing the actor, object, and scene attention maps of AOSA network. From top to bottom, the actions are: rowing boat, batting cricket, cooking, and playing bagpiper. The action and object maps emphasis attention on the human, object, and their interaction, while the scene scene maps focus more on the underlying background. In the attention map red means important region.

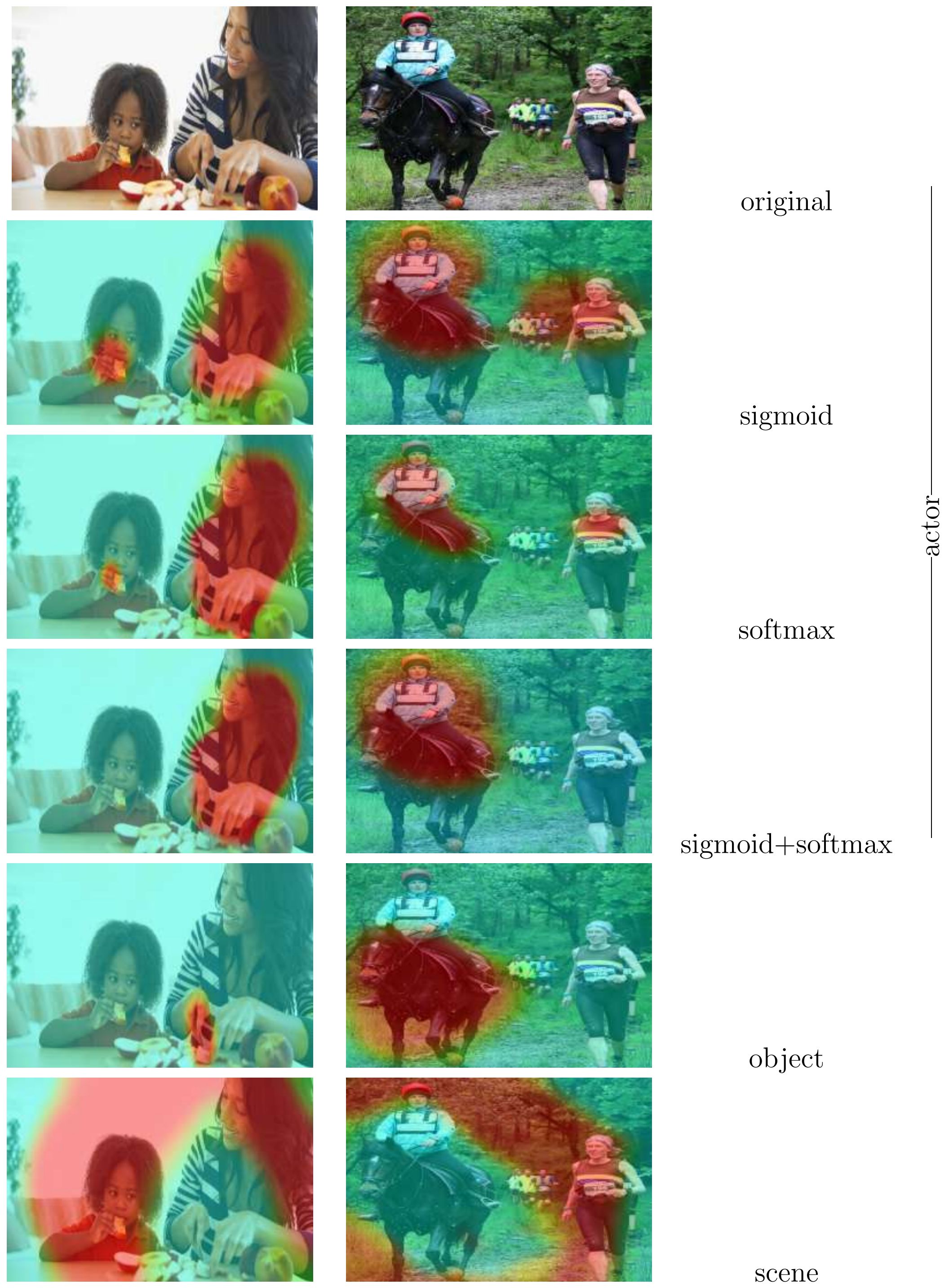


Figure 3.10: Video frames from different video clips of FactNet dataset. Visualizing the sigmoid, softmax, combination, object and scene attention maps of AOSA network for cutting apple and riding horse.

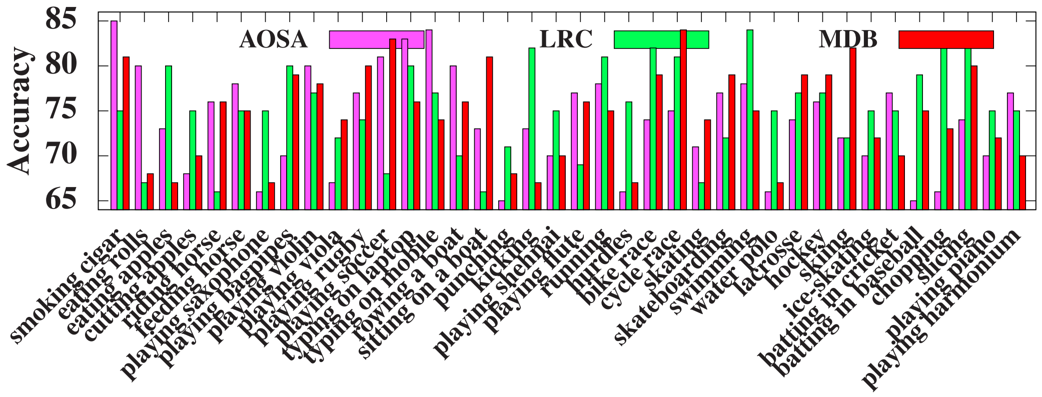


Figure 3.11: Performance of different networks of our architecture on 38 classes of FactNet dataset.

1.1%. The performance of our model outperforms most of the existing methods except for ECO [143].

Table 3.15 depicts Top-1 and Top-5 on Kinetics400 dataset. The accuracy improves 1.1% (Top-1) and 1.6% (Top-5) in Kinetics400 dataset, when our network has only RGB modality. We also report the FLOPs of our method with 16 frames as an input and 112×112 with only 1 crop. The FLOPs of FactorNet is approx. $266G$, which is almost similar to ECO [143]. However, our model surpasses ECO and other model significantly in term of accuracy.

Results on Breakfast-Actions, ActivityNet, and Something-Something V1 datasets. Table 3.16 depicts the accuracy for Breakfast-Actions, Top-3 for ActivityNet, and Top-1 for Something-Something V1 datasets. On Breakfast-Actions, we achieve 1.16% gain. There is only small incremental gain (Top-3) for ActivityNet dataset. We obtain a gain 2.0% (Top-1) from state-of-the-art results on Something-Something V1 dataset, as shown in Table 3.17 and set a new state-of-the-art performance.

Impact of Optical flow modality. We also investigate the effect of optical flow on our proposed architecture. We compare the state-of-the-art results on UCF101 and HMDB51 datasets that includes both RGB and optical flow modalities in Table 3.14. We compute optical flow using the TV-L1 algorithm [153] to calculate motion within the consecutive frames of a video. The optical flow is given as input to temporal stream which however incur significant computational cost [154] and only capture short-range temporal cues of actions.

Table 3.14: Comparison with state-of-the-art methods for scene-related dataset on UCF101 and HMDB51 datasets.

Methods	**UCF101**	**HMDB51**	**Modality**
	Acc.	**Acc.**	
C3D [12]	85.2	-	RGB
C3D + iDT [12]	90.4	-	RGB
Two-stream CNN [11]	88.0	59.4	RGB + Flow
LTC [30]	82.4	-	RGB
LTC [30]	91.7	64.8	RGB + Flow
Factorized CNN [31]	88.1	59.1	RGB + Flow
I3D [140]	84.5	49.8	RGB
Attention pooling [114]	-	52.2	RGB
MiCTNet [141]	88.9	63.8	RGB
MiCTNet [141]	94.7	70.5	RGB + Flow
S3G [115]	96.8	75.9	RGB
S3G + faster RCNN [115]	78.8	-	RGB + Flow
VideoLSTM [144]	79.6	43.3	RGB
VideoLSTM + iDT [144]	92.2	73.7	RGB + Flow
R(2+1)D + Sport1M [149]	93.6	66.6	RGB
STC [145]	93.7	66.8	RGB
ARTNet with TSN [14]	94.3	70.9	RGB
ECO [143]	94.8	72.4	RGB
StNet + ResNet101 [142]	94.3	-	RGB
StNet + IRv2 [142]	95.7	-	RGB
STM [138]	96.2	72.2	RGB
MARS [146]	95.6	73.1	RGB
MARS [146]	98.1	**80.9**	RGB + Flow
I3D-LSTM [129]	95.1	-	RGB
R(2+1)D + Kinetics [149]	96.8	74.5	RGB
Zhu *et al.* [147]	96.9	75.7	RGB
Ours	**98.0**	80.0	RGB
Ours	96.8	80.1	RGB + Flow

Table 3.15: Comparison with state-of-the-art methods on Kinetics400 dataset. "N/A" represents that the authors do not report the inference protocol in their work.

Methods	**Top-1**	**Top-5**	**FLOPs**(G)×**crops**
STC [145]	68.7	88.5	-
S3D [115]	69.4	89.1	71 × N/A
ECO [143]	70.7	89.4	267 × 1
ARTNet with TSN [14]	70.7	89.3	-
R(2+1)D [149]	72.0	90.0	152 × 1
I3D [140]	72.1	90.3	108 × N/A
MARS [146]	72.8	-	-
STM [138]	73.7	91.6	67 × 3
R(2+1)D + Sport1M [149]	74.3	91.4	152 × 1
Attention clusters [116]	75.0	91.9	-
Zhu *et al.* [147]	75.3	-	-
Non-Local Network (I3D) from [148]	77.7	93.3	359 × 30
Slow Fast [150]	78.9	93.5	213 × 30
ip-CSN-152 [151]	79.2	93.8	108.8 × 30
Slow Fast + NL [150]	79.8	93.9	234 × 30
ir-CSN-152 [151]	82.6	95.3	108.8 × 30
Ours	**83.7**	**96.9**	266 × 1

Table 3.16: Comparison with state-of-the-art methods for long-range videos on Breakfast-Actions and ActivityNet datasets.

Methods	**Breakfast-Actions**	**ActivityNet**
	Acc.	**Top-3**
C3D [12]	-	81.16
P3D ResNet [152]	-	87.71
I3D [140]	80.64	-
Non-local [137]	83.79	-
Timeception [121]	86.93	-
PIC [117]	89.84	-
Ours	**93.5**	**87.9**

Table 3.17: Comparison with the state-of-the-art methods on Something-Something V1 dataset.

Method	**Frames**	**Backbone**	**Validation**		**FLOPs**
			Top-1	**Top-5**	(G)×**crops**
I3D from [148]	32	3DResNet50	41.6	72.2	153 × 3
I3D + GCN from [148]	32	3DResNet50	46.1	76.8	303 × 3
ECO [143]	8	BN Inception + 3DResNet18	39.6	-	32 × 1
ECO [143]	16		41.4	-	64 × 3
ECO$_{EN}$ Lite [143]	92		46.4	-	267 × 3
S3D-G [115]	64	Inception	48.2	78.7	-
STM [138]	8	ResNet50	49.2	79.3	33 × 3
STM [138]	16	ResNet50	50.7	80.4	67 × 3
ir-CSN-152 [151]	8	-	52.1	-	-
ip-CSN-152 [151]	8	-	53.3	-	-
Ours	16	-	**55.3**	**85.1**	266 × 1

3.6 Conclusion of the Chapter

In this work, we have presented a simple yet effective deep network that supervises the effect of co-occurring objects and the underlying context for action recognition. Our deep network also captures long-range dependencies of the videos incorporating attention mechanisms in ConvLSTM. We utilize multi-scale lateral-path connections in the backbone network to incorporate finer-detailed enriched spatiotemporal features and prevent the issue of degradation. We also handle the problem of geometric transformations of smaller objects using deformable convolutions. An actor-object-scene branch helps to recognize ambiguous actions by factorizing the spatiotemporal feature tensors with respect to actor, co-occuring object, and underlying background. We explore extensive ablation studies to evaluate the classification accuracy of the proposed FactorNet architecture for different publicly available benchmark datasets with both scene-related and temporal-related videos. FactorNet outperforms state-of-the-art literature in terms of Top-1 Something-Something V1 datasets. We generate a new dataset FactNet, where the annotation is performed based on actor-object-scene interaction.

3.7 Publication related to the Chapter

1. Nitika Nigam, Tanima Dutta and Hari Prabhat Gupta, "FactorNet: Holistic Actor, Object, and Scene Factorization for Action Recognition in Videos," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 3, pp. 976-991, March 2022, doi: 10.1109/TCSVT.2021.3070688.