Chapter 6

Multi-label Complex Action Recognition

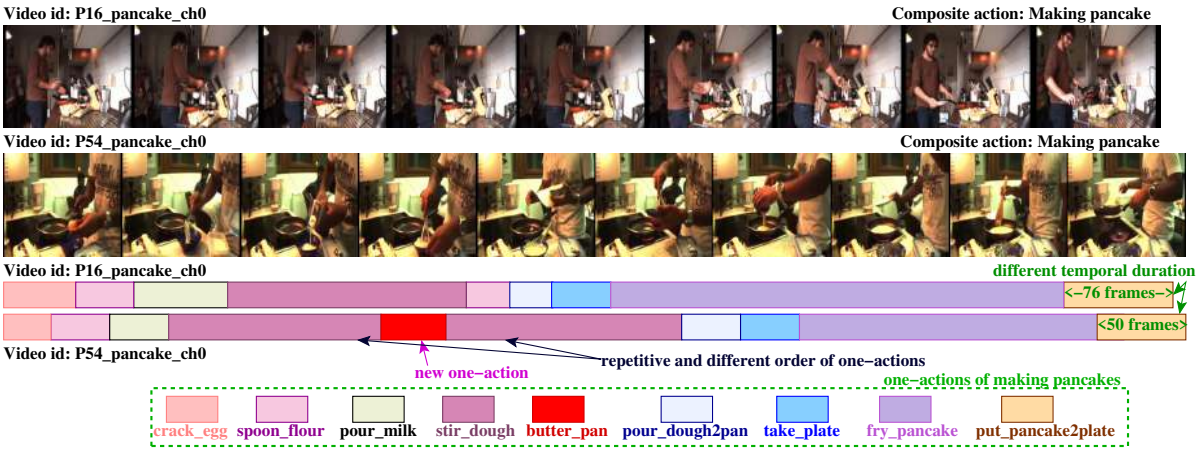

Figure 6.1: Extracted the RGB frames from the video segments present in BreakFastAction dataset. The composite actions 'making pancake' with several one-actions consists of two major issues: a) different temporal duration and b) different temporal order. Due to the wide variations in temporal duration and order, most of deep neural networks are unable to generalize to new instances of videos.

In recent years, most of the industries have achieved high profit by incorporating strategy of human-robot collaboration. Humans understand the instruction-level language, whereas robots understand programmed and schedule actions which are performed in constrained environment [245, 246]. A collaborative robot can reactively

alter its mobility to protect the safety of operators in a complex assembly by observing changes in a shared workspace [247]. However, recent human-robot collaboration assembly (HRCA) systems only collaborate in a predefined manner using fixed robot arms, lacking the intelligence and flexibility required for collaboration. To address this, a new paradigm of proactive HRCA is being considered, where fixed collaborative robots can anticipate the next actions of workers and assist them proactively with subsequent planning.

To achieve this goal, recognizing human actions is essential as it helps in providing foundation for dynamic robotic decision making. One of the prominent areas of research is recognizing human actions from RGB short video data [162]. However, most of the human activities are frequently referred to as composite action that finishes a meaningful task rather than an ephemeral atomic action in day-to-day life. For instance, 'cooking a meal' is a more familiar action than 'chopping vegetables', 'sautéing meat', 'stirring sauces', and 'washing plates'. The former one is a *composite action*, while the later ones are known as *one-actions* [121]. As we know, humans are quite effective in understanding the invariant sequence of one-actions that leads to composite action.

One-actions are characterized by their distinctive visual pattern, coherence in motion, and potential for repetition. In contrast, composite actions consist of multiple one-actions with significant variations in their timing and progression over time [117], as illustrated in Fig.6.1. Due to the presence of several one-actions, composite actions take longer to unfold and are compositionally in-homogeneous. As shown in Fig.6.1, the process of making pancakes involves a sequence of individual actions such as cracking an egg, spooning flour, pouring milk, stirring the batter, buttering the pan, pouring the batter into the pan, frying the pancake, taking a plate, and placing the pancake on the plate. To accurately comprehend the pancake-making process, it is essential for a machine to process all the video frames in order and with the correct duration of each action. Any permutation or alteration in the order or duration of these actions

could result in the misclassification of the overall action, and the AI-based system may struggle to recognize the correct sequence of actions. The issue of temporal permutation of duration and order is crucial to address for reducing the misclassification rate. Aforementioned problem may be particularly severe when dealing with new instances of video with minor changes in the order and duration of one-actions that the model has not been trained on before. Therefore, it is elementary to develop robust algorithms that can accurately identify and analyze the correct sequence of actions, regardless of the order and duration of individual actions.

Recently, deep learning models, such as Timeception [121] and PIC [117] resolve the issue of temporal permutation invariance in videos with long-range activities. In [121], the authors have proposed a novel 'Timeception' module that builds the relationship between the one-actions in a video sequence. On the other hand, the authors of [117] have proposed a novel convolutional neural network architecture called 'Permutation Invariant Convolution' that uses an permutation invariant-based attention mechanism to select relevant local features that contribute to overall activity recognition. However, these models are insufficient to captured the fine-grained temporal dynamics of the composite action due to the utilization of pooling operations. As the order and duration of actions may be different in different instances of video, which can lead to inaccuracies in composite action recognition. Moreover, these models require significant computational power to unfold and recognize the long-range activities. The issue of high computational cost and complexity limits the model scalability and practicality in resourced constrained environment. To overcome the aforementioned limitations of existing literature, we propose a resource-constrained architecture for composite action recognition in a video, known as *Temporal Invariant Kronecker-based Deep Network (TIKNet)*. It handles misclassification of composite human actions in videos regardless of temporal permutation invariance of order and duration of one-actions. TIKNet is a lightweight deep network based on the concept of Kronecker product (KP). KP is

a powerful mathematical tool that helps to capture fine-grained temporal dynamics of composite action in a video sequence.

6.1 Our Contribution

The main contribution of our work is outlined as follows:

- First, we explore the problem of heavy-weighted deep networks which are not suitable for resource-constrained devices due to a large number of parameters. We design a light-weight transformer network to capture fine-detail information of actions without introducing extra parameters. We compress the multi-head attention and feed-forward network using the Kronecker decomposition with very low computational overhead. We decompose the learnable matrices in multi-head attention and feed-forward network without affecting the empirical performance of recognizing composite action.
- Next, we propose lemmas to handle the temporal duration and order invariance occurrences in composite actions. We prove that the dot product in multi-head attention is an invariant function to handle variations in temporal duration and order of one-actions. We also propose the duration invariant lemmas through probabilistic approach.
- Finally, we perform abundant experiments on three publicly available datasets to determine the performance of TIKNet. We perform extensive ablation studies to highlight the impact of different modules on TIKNet. To the best of our knowledge, no video dataset exists in the literature that includes ambiguous composite action pairs. We, therefore, create a new dataset, named as *CompositeNet*, that contains 31 composite actions with about 81 videos per class with average of 4 one-actions per composite action.

6.2 Organization of the Chapter

The rest of the chapter is summarized as follows: In the next section, we review the existing literature. Section 6.4 presents our proposed method for recognizing composite actions in videos. The ablation study and qualitative results are described in Section 6.5. The chapter then concludes in Section 6.6. We finally mentioned the publications related to this chapter.

6.3 Literature Survey

Long-term action recognition, also known as composite action, seeks to identify complicated actions in videos that last for several minutes. It has always been a crucial area of research. Video-Graph [248] is a method to handle minutes-long human activities and learn their underlying temporal structure. Timeception [121] layer is designed to learn long-range temporal dependencies and variations in the temporal extents of complex actions. PIC [117] captures long-range temporal permutation invariant features with the help of cascaded layers and shared weights. The literature [121, 248] do not handle temporal order invariance issue. The authors in [117] though address the temporal order invariance problem but are affected by the issue of scene bias and camera motion on composite action recognition tasks in a long-range video. The authors in [249] utilized a two-stream network where one stream captures the spatial and temporal cues and the other stream learns the optical flow features. However, the authors Cao and *et. al* have described a composite two-stream framework that contains feature extraction and reasoning ability in the classifier branch and a 3-channel self-attention in the composite feature branch [250]. Based on the self-attention process, the authors in [251] have presented Bayesian attention belief networks. Gamma distributions, which are roughly approximated by Weibull distributions, are used to model the un-normalized attention scores. Additionally, transfer learning [252] and the unsupervised approach [253] are

investigated in this field.

• **Motivation of this Chapter** There is numerous deep learning model related to obtain spatiotemporal features of a single action such as C3D [12]. It learns short-term motion and appearance features of a video clip with temporal support of 16 consecutive RGB frames. However, the approach can only tackle shorter clips with single action label. More recently, the authors of [140] have integrated 3D convolutions into a state-of-the-art 2D CNN that results in Inflated3D ConvNet (I3D). However, the computation of optical flow is a very time-consuming process in I3D. In [254], the authors explicitly factorize human action out of context to improve performance in one-action recognition. In [255], a weakly supervised attentional mechanism is incorporated that can separate actions from the co-occurring contextual factors spatially and temporally. In ACAM [256], the features related to the actor branch focus on the regions that are pertinent to conditioned actor and pre-trained object detectors. FactorNet [162] performs factorization of human action features into the activity cues related to actor, correlated objects, and underlying scene to reduce the impact of feature representation biases. The aforesaid analysis is for the classification of one-action which is not suitable for composite action since one-actions in composite action examples may have different temporal order and duration. This motivated us to design a deep neural network for the recognition of composite actions and can also handle the issue of scene bias in composite action videos.

6.4 Proposed Approach

In this section, a permutation invariant light-weight deep network is proposed which is known as *TIKNet*, for video-based composite action recognition. The overall network is shown in Figure 6.2. First, we utilize a light-weight pre-trained deep network as a backbone for extracting semantically enriched spatiotemporal features of a given video segment. Secondly, we propose a novel Kronecker-based Permutation Invariant

Transformer (KP-former) Module to compress the parameters of transformer modules and reduce the computation overhead, especially in the case of a processing sequence of frames in a long-range video. In KP-former, we decompose both multi-head attention (MHA) and feed-forward network (FFN) using the Kronecker product decomposition method. Our proposed transformer also holds the property of permutation invariance which models the temporal structure of composite actions. Lastly, we propose duration invariant lemmas that help to make the TIKNet invariant of variations in temporal duration of one-actions in a composite action.

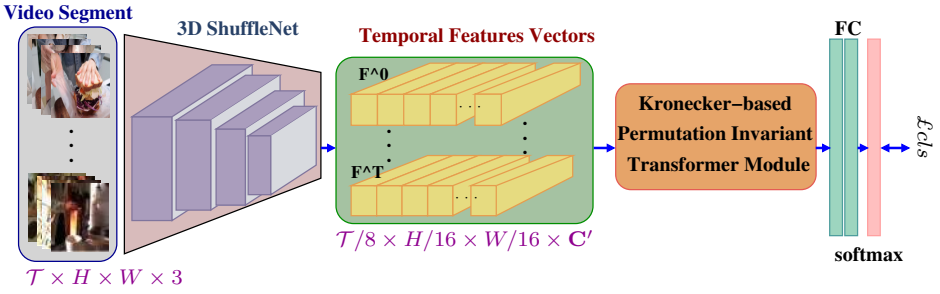

Figure 6.2: Overall architecture of TIKNet. This architecture consists of the 3D ShuffleNet backbone and Kronecker-based Permutation Invariant Transformer (KP-former) modules.

6.4.1 3D ShuffleNet Backbone Network

In this section, we extract spatiotemporal cues from video segments that contain appearance and motion features. These features help in categorizing the video in a particular composite action class. As we know, the magnificent impact of extracting adequate and discriminative spatiotemporal features is obtained from state-of-the-art 3D CNNs, such as C3D, I3D, and R3D [129, 149, 234]. However, the major drawbacks of the aforementioned variants of 3D CNNs are deployment issues on edge devices and heavy computation. Therefore, we opt for light-weight CNNs, like ShuffleNet, MobileNet, and SqueezeNet, which are resource-efficient and resolve the deployment issue on edge devices [235–237]. The classification performance is maintained while the number of model

parameters is reduced by these models. Here, we have chosen ShuffleNet [238] as our backbone module as it outperforms state-of-the-art architecture by a significant margin in terms of parameters, performance, and lower error rate. Mostly, it uses novel point-wise group convolution and channel shuffle that helps to extract discriminate features. It lay out the discriminative appearance and motion features of action performed by a person while reducing the parameters almost to 0.55 million as compared with other CNN models [238].

Formally, we uniformly divide the long-range video $\mathbf{V}$ into a group of non-overlapped $\mathbf{M}$ video segments, which is given as $\mathbf{V} = \{\zeta_1, \cdots, \zeta_{\mathbf{M}}\}$. In each segment, we have club $\mathcal{T}$ successive RGB frames given as input to the backbone network for feature extraction. We have extracted the feature upto last convolutional block of ShuffleNet and output feature vector $\mathbf{F} \in \frac{\mathcal{T}}{8} \times \frac{\mathbf{H}}{16} \times \frac{\mathbf{W}}{16} \times \mathbf{C}'$. Here, $\mathbf{H}, \mathbf{W}$ and $\mathbf{C}'$ are height, width, and output channel, respectively. Further, the extracted spatiotemporal features are given as input to our novel light-weight, permutation invariant transformer module to effectively capture temporal correlated dependencies of one-actions for accurate classification of composite action.

6.4.2 Kronecker-based Permutation Invariant Transformer

The goal of this section is to address the issues related to the recognition of composite actions in a video. The organization of long-range action may have a different permutation of one-actions that leads to the same composite action. The sequence and duration of one-actions can be in different permutations and time duration. These one-actions can also be repeated accordingly by a person in a long-range temporal order. Recently, a video-based transformer network [257] and its variants have shown the powerful capacity and good performance to efficiently learn the motion representation of long-range action videos. However, these effective transformer networks come at the cost of large network parameters and high latency, which makes them unable to run

on resource-constrained edge devices. Therefore, we propose a novel Kronecker-based transformer network, a lightweight and fast transformer that recognizes the composite action in a video. As the structure of the temporal duration of one-actions is unordered, thus we also make our proposed transformer network to be permutation invariant. We named our proposed module as Kronecker-based Permutation Invariant Transformer (KP-former) that handles the temporal order of one-action to accurately predict composite action, as shown in Fig. 6.3.

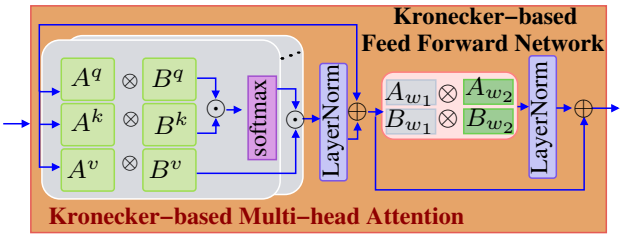

Figure 6.3: The overall architecture of KP-former.

As we know, a traditional transformer network consecutively piles numerous blocks of the transformer to enhance the ability of a network. The block consists of multi-head attention (MHA) and a feed-forward network (FFN) layers for learning wider and global semantic temporal representations of RGB frames. MHA uses a query, key, and value decomposition whereas FFN has linear layers to maintain the correlation between the RGB frames.

In our KP-former, we decompose both MHA and FFN using Kronecker decomposition [258] to reduce the size of the transformer and increase the latency. KP-former consists of Kronecker-based temporally invariant multi-head attention (KTIMHA) and Kronecker-based temporally invariant feed-forward network (KTIFFN). KTIMHA consists of multiple heads which takes as an input the feature tensor $\mathbf{F}$ to preserve the temporal correlations of one-actions nevertheless of their order, is formally given as:

$$Multihead(\mathbf{F}) = (head_1 \oplus head_2 \oplus \cdots \oplus head_j)\mathfrak{W}_f, \tag{6.1}$$

where $\oplus$ is concatenation operator and $\mathfrak{W}_f$ is learnable parameter of size $\mathbb{R}^{d_v \times d_f}$. Each $head_j$ consists of self-attention function $Att(\cdot)$ which is formally given as:

$$Att(\mathbf{F}) = \mathbf{softmax}\left(\frac{(\mathfrak{W}_q\mathbf{F}) \cdot (\mathfrak{W}_k\mathbf{F})^\top}{\sqrt{d}}\right)(\mathfrak{W}_v\mathbf{F}), \tag{6.2}$$

where $\mathfrak{W}_q, \mathfrak{W}_k, \mathfrak{W}_v$ are weight matrices of query, key, and value $\mathfrak{W}_q, \mathfrak{W}_k, \mathfrak{W}_v$, respectively. The size of $\mathfrak{W}_q \in \mathbb{R}^{d \times d_q}$, $\mathfrak{W}_k \in \mathbb{R}^{d \times d_k}$, and $\mathfrak{W}_v \in \mathbb{R}^{d \times d_v}$. d is the dimensionality of the key vectors, which is utilized for the stability of gradient and $\mathbf{softmax}(\cdot)$ is utilized to extract probability distributions. We have decomposed each weight matrix in $head_j$ using Kronecker-product decomposition to reduce the parameters. Formally, the decompose learnable matrices are formulated as:

$$\mathfrak{W}_\star = \mathbf{A}_\star \circledast \mathbf{B}_\star, \tag{6.3}$$

where $\star$ denotes query, key, and value, respectively and $\circledast$ is Kronecker operator. The dimension of $\mathbf{A}_\star$ and $\mathbf{B}_\star$ is $\mathbb{R}^{d_1 \times d_{1,\star}}$ and $\mathbb{R}^{d_2 \times d_{2,\star}}$, respectively. We have obtained approximate feature representation of $\mathfrak{W}_\star$ by setting the number of Kronecker summations $\mathcal{S}$ equal to $\mathbf{min}(d_1 d_2, d_{1,\star} d_{2,\star})$, formulated as:

$$\mathfrak{W}_\star \approx \sum_{i \in \mathcal{S}} \mathbf{A}_{i,\star} \circledast \mathbf{B}_{i,\star}. \tag{6.4}$$

Similarly, we have decomposed the FFN layer using Kronecker decomposition. In KTIFFN, we have decomposed the learnable parameters of two fully connected layers (FC) which is mathematically given as:

$$\mathfrak{W}_1 = \mathbf{A}_{\mathbf{w_1}} \circledast \mathbf{B}_{\mathbf{w_1}}, \mathfrak{W}_2 = \mathbf{A}_{\mathbf{w_2}} \circledast \mathbf{B}_{\mathbf{w_2}}, \tag{6.5}$$

where $\{\mathfrak{W}_1, \mathfrak{W}_2\}$ are the weight matrices of FC layers and $\{\mathbf{A}_{\mathbf{w_1}}, \mathbf{B}_{\mathbf{w_1}}, \mathbf{A}_{\mathbf{w_2}}, \mathbf{B}_{\mathbf{w_2}}\}$ are decomposed matrices. Mathematically, our proposed feed-forward network is repre-

sented as follows:

$$KTIFFN(\mathcal{X}) = \mathbf{ReLU}(\mathcal{X}\mathfrak{W}_1)\mathfrak{W}_2, \tag{6.6}$$

where $\mathcal{X}$) is obtained from KTIMHA followed by LayerNorm and Add module. $\mathcal{X}$ is calculated as:

$$\mathcal{X} = \mathbf{F} \oplus \mathbf{LayerNorm}(KTIMHA(\mathbf{F})), \tag{6.7}$$

where $\oplus$ is addition operation and **LayerNorm** is Layer Normalization layer [259]. As we have to prove that KP-former is a permutation invariant module, thus we have proposed Lemmas related to permutation invariance property in the following subsections.

6.4.2.1 Permutation Invariance to order of one-actions

In this section, we have explained and discussed the Kronecker-based MHA and FFN modules help to state that our KP-former is invariant to all the permutations of one-actions in the temporal dimensions. We have introduced some terminology and proposed Lemmas to prove whether the module is invariant or not. We have proceeded to formally define the permutation invariance property.

Definition 1. (Permutation Vector). A vector $\mathbf{Z} = (z_1, \cdots, z_n)$, each with n elements is a permutation vector iff permutation operator π moves an element at index i to $\pi(i)$, *i.e.,* $(z_1, \cdots, z_n)$ is equivalent to $(z_{\pi(1)}, \cdots, z_{\pi(n)})$.

Definition 2. (Permutation Matrices). A matrix $\mathbf{M} \in \mathbb{R}^{i \times j}$, with ij elements is a permutation matrix iff permutation operator π moves an element at index i to $\pi(i)$ and j to $\pi(j)$, *i.e.,* $\mathbf{M}_{ij} = \mathbf{M}_{\pi(i)\pi(j)}$.

Definition 3. (Permutation Invariant). Let $\mathcal{S}_n$ be the set of all permutations of a vector or a matrix. A function $f : \mathbf{Z}^n \rightarrow \mathbf{Y}^n$ is permutation invariant iff for any

$\mathcal{S}_\pi \in \mathcal{S}_n$ and for all $\mathbf{Z}^n \in \mathbb{R}^{n\times m}$,

$$f(\mathbf{Z}^n) = f(\mathcal{S}_\pi \mathbf{Z}^n). \tag{6.8}$$

In our case, we have provided our key definition of a function whose result is invariant to permutations of the temporal order of one-actions. To achieve permutation invariance, we have proposed Lemma 1. below that provides an overall characterization of self-attention mechanism used in KTIMHA.

Lemma 1. Let the dot product of query $\mathbf{Q}$ and key $\mathbf{K}$ in the self-attention network remain unchanged after applying any permutation π i.e.,

$$(\mathcal{S}_\pi \mathbf{K})^\top \cdot (\mathcal{S}_\pi \mathbf{Q}) = \mathbf{K}^\top \mathbf{Q}. \tag{6.9}$$

Proof. First, we have considered a matrix product of query and key as follows:

$$\mathbf{K}^\top \mathbf{Q} = \sum_{i\in n} \sum_{j\in m} \mathbf{K}_{ij}^\top \mathbf{Q}_{ij}. \tag{6.10}$$

Next, we have permuted the row of $\mathbf{K}$ and columns of $\mathbf{Q}$, given by:

$$(\mathcal{S}_\pi \mathbf{K})^\top \cdot (\mathcal{S}_\pi \mathbf{Q}) = \sum_{i\in n} \sum_{j\in m} \mathbf{K}_{\pi(i)j}^\top \mathbf{Q}_{i\pi(j)} \tag{6.11}$$

As we know, summation operators have commutative properties. We first reorder the terms of matrices and then apply the commutative property, we formulated the new equation through Eq. 6.10 and Eq. 6.11 as:

$$\sum_{i\in n} \sum_{j\in m} \mathbf{K}_{\pi(i)j}^\top \mathbf{Q}_{i\pi(j)} = \sum_{i\in n} \sum_{j\in m} \mathbf{K}_{ij}^\top \mathbf{Q}_{ij}. \tag{6.12}$$

From the Definition 6.4.2.1 the Eq. 6.12 is further implies to:

$$(\mathcal{S}_\pi \mathbf{K})^\top \cdot (\mathcal{S}_\pi \mathbf{Q}) = \mathbf{K}^\top \mathbf{Q}. \tag{6.13}$$

As the self-attention function of KTIMHA module is satisfying the property of permutation invariance. This implies that our KP-transformer is also permutation invariant.

6.4.2.2 Repetition Invariance to the duration of one-action

As we can observe from the Fig. 6.1 that a composite action may contains a sequence of one-actions which can be repetitive and vary in duration. In video1, a person is cracking egg, spooning flour, and so on whereas in video2, a different person is performing the same action while repeating the sequence of one-action again in a different manner. The duration and repetition of one-action may vary person to person to perform the same composite action. These repeated actions may predict ambiguous action class by a trained deep network due to temporal invariance of one-actions in different instances. Moreover, each event of one-action is independent of other one-action with repeated possibilities. The overall scenario implies to resolve the aforementioned issue through the likelihood approach. We have resolved it through the concept of Bernoulli trails, where the probability p for correct action class on each trail of temporal duration should be same for n independent repetition trails. We have proved the repetition invariance to duration of one-action in a video through the following Theorem 1, describe as follows:

Theorem 1. If the probability of occurrence of a successful event of recognizing one-action from a video in a single trial of a Bernoulli's experiment is p, then the probability that the event occurs exactly r times out of n independent trials is equal to ${}^nC_r(1-p)^{n-r}p^r$, where $(1-p)$ is the probability of failure of recognizing one-action.

Proof. As we know, getting exactly r successes of correct action class means getting r successes and (n–r) failures simultaneously. Formally, the probability of getting r

correct action class (successes) and $n - r$ incorrect action (failures) is given by:

$$\mathbf{P}(r \; successes \; and \; (n-r) \; failures) = (1-p)^{n-r}p^r. \tag{6.14}$$

We can choose the r trails of successes through ${}^nC_r$ ways. Once, the successful events are selected, the remaining $(n - m)$ trails should result in the failures. Through the addition theorem [260], the required probability is given by: ${}^nC_r(1-p)^{n-r}p^r$. Finally, we have incorporated the fully-connected layers and a softmax or sigmoid layer after KP-former to classify the one-actions and composite action class, respectively. We have trained our overall architecture using cross-entropy loss and follows the same training strategy as in [117].

6.5 Experimental Results

We empirically evaluate our TIKNet on benchmark datasets for action recognition in videos. A comprehensive study on the influence of different modules of TIKNet is performed. Moreover, we compare TIKNet with the state-of-the-art methods for composite-actions related publicly available datasets (*i.e.*, Charades, Breakfast-Actions, and Multi-THUMOS). We also propose a new composite video dataset, named as *CompositeNet*.

6.5.1 Datasets and Metrics

A set of experiments are conducted on three benchmark composite action datasets, such as Charades, Breakfast-Actions and Multi-THUMOS. We also investigate the performance of our architecture on our CompositeNet dataset.

Charades [261]. Everyday casual activities are covered in this datasets. The actions are performed by human in their houses. It includes 9848 annotated videos with an

average length of 30 seconds. It consists of 157 action classes and each contains 6 one-actions respectively.

Breakfast-Actions [136]. It is an annotated cooking video dataset of people preparing breakfast meals. It includes 1712 breakfast preparation videos for 48 action classes and about 6 action instances for each video. The actions are performed by 52 people in 18 different kitchens. The average duration of the videos is about 2.3 minutes.

Multi-THUMOS [262]. It consists of untrimmed human activities in videos with 65 action classes and 400 videos. It covers total 30 hours of data and each video has a complex action with 11 unit-actions.

Table 6.1: The confusion matrix of CompositeNet dataset performed by TIKNet.

	exercise	**yoga**	**teaching**	**anchoring**
exercise	86.2	13.8	0.0	0.0
yoga	14.1	85.9	0.0	0.0
teaching	0.0	0.0	87.3	12.7
anchoring	0.0	0.0	7.5	92.5

Table 6.2: Performance of current state-of-the-art composite-actions methods on CompositeNet dataset.

Methods	**CompositeNet Acc.**
ResNet152+Timeception [121]	82.8
I3D+ Timeception [121]	85.4
3D ResNet101+Timeception [121]	86.3
ResNet152+Videograph [248]	86.5
I3D+Videograph [248]	86.8
I3D+PIC [117]	87.0
3D ResNet101 + PIC [117]	87.2
3D ResNet101 + NL+PIC [117]	87.5

CompositeNet dataset. In video analysis tasks, action and scene are important to learn long-range spatial-temporal features. The representation bias issue is capture if we train the model with publicly available datasets. We are not able to classify the

new action classes accurately. The conjugate and transpose action pairs that lead to ambiguity are not available in the public datasets. This motivated us to train TIKNet with adequate obscure conjugate and transpose action pairs. We have proposed our dataset, which is known as CompositeNet. CompositeNet has 31 composite classes which consist of 149 one-actions. An average of 81 videos is collected in each class with 3.5 minutes (on average). The videos are collected from movies, YouTube, and web series, which are recorded in the presence of huge disparities in object appearance and pose, object scale, viewpoint, cluttered background, camera motion, and illumination conditions. They are trimmed wherever necessary. The resolution of the videos varies from 640×360 to 1280×1080. We split 50% of the dataset for training, 25% for validation, and 25% for testing. We have shown the confusion matrix of some ambiguous classes such as 'exercise, 'yoga', 'teaching', and 'anchoring in Table 6.1. Moreover, we have shown the performance of current competitive action recognition models in Table 6.2. We validate the effectiveness of individual modules of our architecture on each composite action class of CompositeNet, as shown in Fig. 6.4.

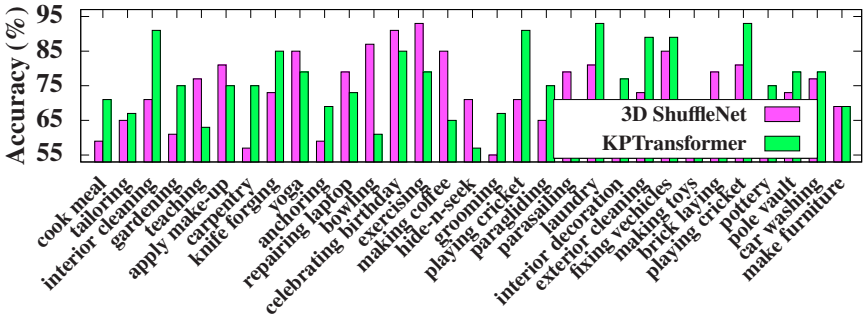

Figure 6.4: Representation of distinct modules of TIKNet on CompositeNet .

We have used mean accuracy (Acc.) for Breakfast-Actions datasets. We have used mean average precision (mAP) for Charades and multi-THUMOS datasets.

6.5.2 Implementation Details

The implementation details of different modules of CompositeNet are briefly described as follow:

Length of RGB frames. RGB frames are extracted from Charades, Multi-THUMOS dataset and Breakfast-Actions at 20 frames per second. In case of CompositeNet datasets, we choose 15 frames per second. To get complete knowledge to describe motion is the main motive behind the selection of different frames per second. As for the experiments, we have found that 1024 RGB frames are best suitable for TIKNet. Thus, we have used 1024 RGB frames as the value of the temporal dimension.

Backbone Network. When pre-training backbone on a specific video dataset, we uniformly sample each composite video $\mathbf{V}$ into 1024 RGB frames, named as a video segment. A segment is fed into the backbone network.

KP-former Module. In this module, LayerNorm is incorporated after every KTIMHA and KTIFFN to normalize and control the gradient scales. We have initialized the weight matrices using He initializer to get uniform warmup of all matrices.

Training details. TIKNet is implemented based on Pytorch [263] platform. We use 8 GTX 1080TI GPU machine to train our TIKNet for recognizing composite actions. To avoid overfitting, data augmentation is conducted throughout training by cropping a random clip with 112×112 crops from four corners and one center spatial location. We generate 4 groups by sampling from 4 positions with equal temporal spacing and average the scores of all non-overlapping segments. The input size of a composite video segment is $1024 \times 112 \times 112 \times 3$.

The training procedure for TIKNet consists of three stages. In the first step, we pre-train the backbone network using a large dataset (*Charades*) separately in a fully-supervised manner to obtained fine spatiotemporal action features. The final prediction is made by feeding the output embeddings from the KP-former into a two-layer linear classifier. We utilize Adam as an optimizer to recognize composite action. In addition,

we set momentum of 0.9 and a weight decay of 10^{-6}. We initialize the parameters with He initializer. The dropout layer is used in between the convolution layers and the dropout rate is set to 0.6 to mitigate over-fitting. The batch size is set to 32 for pre-training the backbone network over 100 epochs. After pre-training, in second stage we plug KP-former modules on top of the backbone and fine-tune the TIKNet on the same dataset. Finally, we fine-tune our TIKNet in an end-to-end fashion with the target dataset.

Inference. At test time, we resize the sides of frames to 112 and utilize a center crop. We sample 10 video segments randomly from videos to measure recognition scores in the temporal domain. The final prediction is the averaged classification scores of all segments.

6.5.3 Ablation Study

This section examines ablation studies to measure the performance of TIKNet for recognizing composite action in a video segment.

•**Effect of RGB frames length.** The impact of varying temporal length in an input video segment analyzes to capture the temporal effect on all the datasets. As we increase the number of RGB frames by 2^k, we reduce the batch size to $1/2^k$ clips per GPU to fit into memory. In Table 6.3, we examine the various number of RGB frames from 128 to 2048. We draw the following conclusions: (1) The number of parameters increases as the length of RGB frames increases. (2) There is a trade-off between accuracy and computational overhead. We observe that 1024 RGB frames are sufficient to recognize correct composite actions.

•**Impact of different backbone networks.** In addition to analysis of different components, we have experimented with different variants of 3D-based deep networks as a backbone modules. Table 6.4 describes the effect of variation in the backbone network on BreakFast-Actions and CompositeNet in terms of mean accuracy. We have experi-

Table 6.3: Effect of length of RGB frames as an input in our TIKNet on benchmark datasets, *i.e.,* Charades and CompositeNet in terms mAP and mean accuracy.

# frames	Charades	CompositeNet	FLOPs (G)
	mAP	Acc.	
128	57.0	84.8	1.0 ×
256	57.3	85.9	1.4 ×
512	59.6	87.1	1.6 ×
1024	**60.2**	**87.9**	2.0 ×
2048	52.9	83.7	2.2 ×

mented with both 3DResNet, I3D, and C3D models and their variations as a backbone network. We observe that parameters of 3D ShuffleNet is less than other backbone networks with good accuracy.

Table 6.4: Performance of TIKNet with different backbone networks (3DResNet, I3D (RGB only), and C3D netwok on Breakfast-Actions and CompositeNet datasets in terms of mean accuracy.

Backbone Variations	Breakfast-Actions Acc.	CompositeNet Acc.	#params ($\times 10^9$)
3DResNet-18 [15]	69.4	86.0	33.3
3DResNet-34 [15]	70.0	86.8	63.6
I3D [140]	71.4	87.1	12.75
C3D [12]	70.8	85.5	32.1
3D ShuffleNet [238]	**71.6**	**87.9**	**0.55**

•**Effectiveness of KP-former.** In this section, we compare KP-former to other tensor decomposition compression methods. We apply architecture compression technique based on Tucker [264], TensorTrain [265], Tensor-Ring [266] along with our proposed KP-former to depict the effectiveness of Kronecker decomposition. The comparison results are reported in Table 6.5. We decompose the transformer network using different decomposition methods and named as, Tucker-former, TensorTrain-former, and Tensor-Ring-former, respectively. We also compare the paramaters and FLOPS on Charades and BreakFast action datasets to effectively show the performance of our proposed model in Table 6.6. As this table suggests, KP-former outperforms all other

decomposition methods for a similar compression factor. We attribute the performance of KP-former to its higher representation power. This is refected in its ability to better reconstruct weight tensors in a pretrained network in comparison to other decomposition methods.

Table 6.5: Comparison of different decomposition methods on Charades, Breakfast-Actions, and CompositeNet datasets in terms of mAP and mean accuracy.

Compression Methods	**Charades mAP**	**Breakfast-Actions Acc.**	**CompositeNet Acc.**
Transformer	58.2	68.3	82.0
Tucker-former	58.6	69.1	82.8
Tensor-Ring-former	59.7	70.4	87.0
TensorTrain-former	50.1	68.1	79.3
KP-former	**60.2**	**71.6**	**87.9**

Table 6.6: Comparison of paramaters and FLOPs of different decomposition methods on Charades and Breakfast-Actions datasets in terms of mAP and mean accuracy.

Compression Methods	**Param (M)**	**FLOP (B)**	**mAP**	**Acc.**
Transformer	24.5	3.8	58.2	68.3
Tucker-former	18	4.0	58.6	69.1
Tensor-Ring-former	17	4.5	59.7	70.4
TensorTrain-former	12	5.7	50.1	68.1
KP-former	**9**	**6.3**	**60.2**	**71.6**

•**Impact of number of heads in KP-former.** Table 6.7 shows the comparable values on different rates of heads in KP-former. The range of number of heads varies from 2 to 8. Our experiments show that 4 number of heads in multi-head attention achieves better accuracy than others. Initially, we observe that as we increase the number of heads the performance improves, however, the computational complexity also increases. When we append more than 4 heads, the accuracy does not improve further, rather the performance degraded. This is due to the overfitting of parameters.

•**Impact of standalone modules.** The modules present in TIKNet are ShuffleNet, and KP-former. In Table 6.8, we analyze the efficiency of standalone modules to recognize the composite actions. In addition, we explore the combination of all modules on

Table 6.7: Comparison of number of heads on Charades, Breakfast-Actions, and CompositeNet datasets in terms of mAP and mean accuracy.

number of heads	Charades mAP	Breakfast-Actions Acc.	CompositeNet Acc.
1	54.3	69.0	84.6
2	55.4	69.8	85.0
3	58.2	70.1	86.6
4	**60.2**	**71.6**	**87.9**
5	58.8	68.1	85.9
6	57.7	67.5	85.7
7	57.5	66.8	85.0
8	57.4	66.2	84.8

Breakfast-Actions, Charades, and CompositeNet datasets to validate the participation of each module in TIKNet.

Table 6.8: Impact of standalone modules in TIKNet on Charades, Breakfast-Actions, and CompositeNet datasets (mAP and mean accuracy).

Model	Charades mAP	Breakfast-Actions mAP	CompositeNet Acc.
ShuffleNet only	59.1	70.0	85.6
KP-former only	59.8	70.6	86.2
TIKNet	**60.2**	**71.6**	**87.9**

•**Variation of training samples.** This section provides the visualization results of the attention-based saliency. We present two variants of training samples on Charades and Multi-THUMOS datasets for comparing the performance. We estimate the performance of both the variants with the training size as shown in Fig. 6.5. It can be noticed that as we increase the number of videos in each method, mAP also improves. The performance improves significantly when training size is raised from 10^5 to 10^7 on Charades. In contrast, the training size is increased from 10^5 to 10^8 on Multi-THUMOS.

•**Visualization Verification.** Fig. 6.6 shows an example of composite action with Kronecker-based multi-head attention mechanism. By visualizing the composite action cues, we conclude that RGB frames with higher action probabilities will be em-

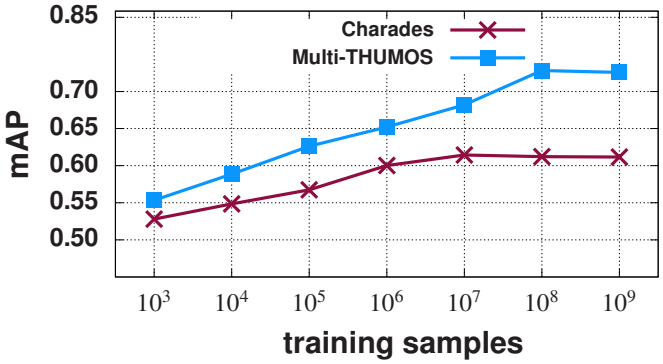

Figure 6.5: mAP vs. training samples on Charades and Multi-THUMOS datasets.

Figure 6.6: Video frames of 'cooking a meal' action from CompositeNet dataset. The one-actions are: 'washing vegetables','chopping vegetables', 'pouring oil in pan', 'sautéing vegetable', and 'serving food'.

.