Fine-Grained Spatio-temporal Feature Learning Network for Video Understanding

Meng Li, Zongwen Bai*, Meili Zhou
School of Physics and Electronic Information, Yan'an University, Yan'an, Shaanxi, China
**Corresponding Author.*

Abstract: In order to enhance video action recognition accuracy, we introduce an enhanced model built upon the GSF-ResNet-50 to address the challenges in fine-grained recognition tasks: the Fine-Grained Spatio-Temporal Feature Learning Network (FSTFL-Net). This model learns rich and detailed spatio-temporal features while maintaining a low computational overhead to capture local representations between similar actions and subtle differences between actions. FSTFL Net introduces a spatiotemporal correlation (STC) module, which can find the spatial neighborhood represented by each local region, the correlation between adjacent and non adjacent frames, and convert these correlations into relationship values to establish spatial connections within the same frame and temporal connections between different frames. This enables FSTFL Net to improve its discriminative ability for fine-grained images, thereby enhancing recognition ability. In a series of rigorous tests, the suggested approach has demonstrated a notable enhancement in recognition precision across four datasets dedicated to action recognition. Specifically, on the Sth-Sth V1 dataset, the FSTFL-Net has achieved a 2.65% increase in Top-1 accuracy compared to the initial model.

Keywords: Fine-grained Spatio-temporal Feature; Action Recognition; Spatio-temporal Features; Spatio-temporal Correlation; Top-1 Accuracy

1. Introduction

Action recognition requires not only modeling the spatio-temporal information, but also distinguishing the subtle local differences between highly similar actions. Drawing inspiration from the extensive utilization of neural networks within the realm of image processing, CNNs adapt to analyze video data, and find cross-frame spatio-temporal convolutions, enabling the extraction of temporal features alongside spatial information. However, current methods face challenges in discerning the nuanced distinctions among highly similar actions, such as the local leg posture movements, where the subtle variances between walking, jogging, and running are difficult for previous models to detect. Therefore, the established model needs not only to capture the semantic information in video accurately, but also to perform fine-grained action analysis through the overall appearance and motion differences.

Recently, several attempts have been to utilize differences in visual tempo to distinguish similar actions. Initial models such as the Two-Stream[1] and SlowFast[2] networks were foundational in advancing the technology for recognizing actions, SlowFast[2] processes a 64-frame input by sampling at varying temporal frequencies, using a 16-frame interval for its slow pathway and a 4-frame interval for the fast pathway. This results in an input composed of sequences with 4-frame and 16-frame samples. The architecture's backbone subnetworks concurrently integrate information from both fast and slow tempos, enhancing prediction accuracy significantly. However, this method also leads to a substantial rise in computational demands due to the differing sampling rates. To further optimize the visual tempo recognition model, D. Zhang, Ceyuan Yang, et al. have proposed the network architectures such as DTPN[3], TPN[4], etc., where the TPN[4] takes advantage of the hierarchical structure of features formed within the network to process input frames supplied at a single rate, extract features at multiple levels and aggregate them. However, the efficacy of this approach is intricately linked to the network's capacity for modeling, which makes the model less capable

Copyright @ STEMM Institute Press

of capturing fine-grained information.

To enable the network to extract fine-grained information in video frames better, we propose an improved model: Fine-Grained Spatio-Temporal Feature Learning Network (FSTFL-Net), as shown in Figure 1. The network utilizes the GSF-ResNet-50[5] as its backbone, embeds the Spatio-Temporal Correlation (STC) Module subsequent to the res3 layer, and leverages the output features from the res2 and res3 layers as module inputs. It performs end-to-end learning without the need for additional supervision and is capable of extracting pixel-level fine-grained temporal dynamic information in long and short video sequences. Our enhancements bolster the model's effectiveness at capturing minor local differences between similar actions, thereby improving the model's expressiveness and generalization capabilities.

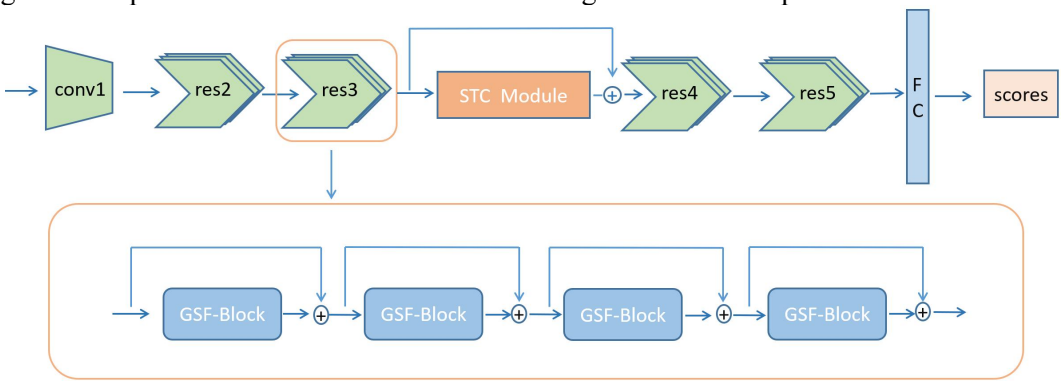

Figure 1. Fine-Grained Spatio-Temporal Feature Learning Network (FSTFL-Net), ⊕ Means Element-Wise Addition

Our main contribution is the introduction of the STC module (see Figure 2). Given a set of frame sequences, this module establishes the spatial structure relationship within the frame by performing correlation analysis on the spatial neighborhood and temporal neighborhood of each local area, at the same time, it determines the temporal feature pairs of adjacent frames and non-adjacent frames, applies correlation calculations, and constructs correlation tensors for each set of features. Subsequently, through the MLP network, high-level semantic concepts and pixel-level fine-grained temporal dynamic information are extracted from the low-level features of the video frame. Finally, feature fusion transforms the filtered tensor information into motion features. In detail, this module takes the video feature tensor I as input, it transforms it into the correlation tensors S_{spatio} and $S_{temporal}$ by calculating the similarity of the spatial neighborhood, the adjacent frames, and the non-adjacent frames. Then, it extracts the feature tensors F_{spatio} and $F_{temporal}$ from the correlation tensors. Finally, it obtains the final tensor Z, matching the input I dimension, through feature fusion.

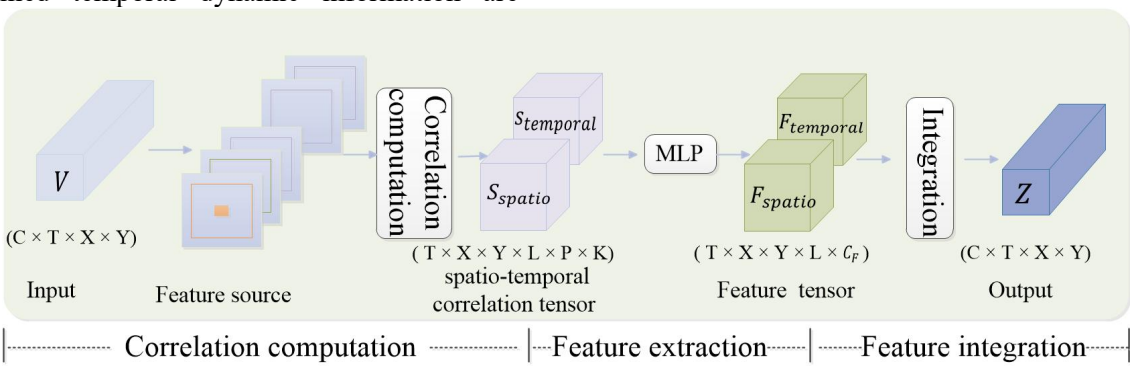

Figure 2. Spatio-Temporal Correlation (STC) Module

2. Related Work

2.1 Based Methods

The recognition of actions is crucial in fields like intelligent surveillance and understanding video sequences and healthcare. The primary objective here is to accurately categorize videos into their respective classes. To enhance the precision of classification, extensive

 Copyright @ STEMM Institute Press

research has focused on leveraging deep neural networks for effectively capturing both the semantic and temporal information in videos. The advent of 2D CNNs and 3D CNNs has significantly propelled the progress in action recognition technology. The two-stream network[1] incorporates two networks to separately process RGB appearance features and optical flow motion data. This model employs an average pooling strategy for spatio-temporal integration, addressing the challenge of short video analysis. Nonetheless, it struggles with analyzing longer video sequences due to its calculation of optical flow, a process that is both time-intensive and incompatible with real-time applications. Another aspect, 3D CNNs introduce the use of 3D convolutional kernels to concurrently get spatial and temporal features, thereby significantly enhancing motion information capture and recognition accuracy, albeit at a high computational cost.

To boost computational efficiency, there has been a shift towards optimizing both 2D and 3D CNNs and exploring hybrid networks. For instance, TSN[6] suggests sampling video clips from uniformly divided segments; TRN[7] introduces an interpretable network that models frame dependencies across various timescales for temporal reasoning; TSM[8] incorporates a shift module to slide certain input feature channels along the temporal axis, simulating 3D convolutions with 2D operations and thus facilitating inter-frame communication; TAM[9] presents a novel two-tier adaptive modeling approach that separates dynamic kernels into position-sensitive mappings and position-invariant aggregation weights, efficiently capturing short-term details and long-term structures. While these hybrid models enable 2D CNNs to reach recognition accuracies comparable to those of 3D CNNs, the issue of fine-motion analysis remains largely unaddressed.

2.2 Action Visual Tempo Modeling

Video actions need to be recognized through frame sequences. However, the diversity and similarity of actions, coupled with the presence of actions with varying visual tempos such as walking, jogging, and running, make action recognition increasingly challenging. People have started studying and exploring visual tempo modeling in recent years. Among these efforts, SlowFast[2] employs a dual-pathway approach, a high-resolution but slower-processing convolutional neural network (CNN) for static elements, and a faster yet lower-resolution CNN for dynamic elements. This method effectively balances detail and speed in video analysis, respectively, and fuses them through lateral connections. While different sampling rates can characterize video samples at various visual speeds, processing frames with two independent networks imposes a significant computational burden. TPN[4] initiates the process by deriving features at multiple levels from the primary network. It then refines these features by modulating both spatial and temporal dimensions. Subsequently, it employs the Information Flow module for advanced processing and integrates all the features to produce the ultimate result. TPN[4] requires only a single network implementation that is compatible with different network architectures. However, the backbone network's temporal modeling ability is limited, so it does not fully exploit the low-level features.

2.3 Motion Feature Learning

While leveraging external optical flow for motion information extraction significantly enhances the accuracy of action recognition, employing a dual-stream (RGB and optical flow) framework incurs substantial computational expenses. Consequently, researchers have shifted towards developing methods that directly extract motion features from RGB frame sequences within the neural network architecture. For instance, MFNet[10] enables end-to-end training to derive spatio-temporal information from adjacent frames. Similarly, MotionSqueeze[11] identifies similarities between consecutive frames to map out correspondences, which are then converted into motion features. These approaches virtually eliminate the need for extra computational resources. However, they primarily focus on temporal features extracted from immediately successive frames, posing challenges in comprehensively understanding videos with longer sequences.

3. Proposed Method

In this segment, we will delve into the various components that make up the Spatio-Temporal

Copyright @ STEMM Institute Press

Correlation (STC) Module and the intricate details of its implementation. The steps for the module's implementation are illustrated in Figure 2:

(1) Consider a neural network that takes a video feature tensor $I \in R^{C\times T\times X\times Y}$ as input. The network transforms I into spatial correlation tensors S_{spatio} and temporal correlation tensors $S_{temporal}$ by calculating the similarity of spatially adjacent and non-adjacent frames.

(2) The correlation tensor is processed by a Multilayer Perceptron (MLP) to extract both the high-level semantic information and the detailed temporal dynamics at the pixel level from both adjacent and non-adjacent frames, resulting in the generation of the feature tensors F_{spatio} and $F_{temporal}$.

(3) By feature fusion, we obtain the final tensor $Z \in R^{C\times T\times X\times Y}$, which has the same size as the input V.

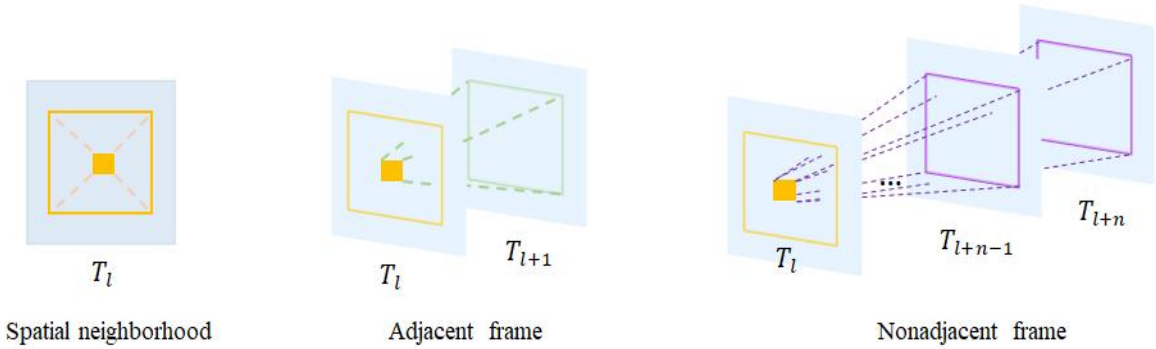

Figure 3. Spatio-Temporal Correlation Calculation Layer

3.1 Correlation Calculation

We utilize the spatio-temporal correlation calculation layer (see Figure 3). The process involves executing a correlation operation to determine the similarity mapping of proximate points within the same frame as well as the similarity mapping across two frames. This method facilitates the acquisition of low-level features for each frame, along with pixel-level detailed temporal dynamic data. The term "correlation calculation layer" denotes a neural network layer designated for assessing the correlation between two feature maps. The underlying implementation principle are outlined as follows:

Given two feature maps I_1 and I_2 both residing in $R^{C\times X\times Y}$, where C represents the channel dimension, X and Y denote the spatial dimensions, according to the definition of the correlation calculation layer, the correlation matrix $R_{(d_1,d_2,i,j)} \in R^{(2D+1)\times(2D+1)\times X\times Y}$, between the two feature maps, where D is the maximum displacement parameter, which determines the maximum distance that can be compared between each region in the two feature maps, The term $R_{(d_1,d_1,i,j)}$ denotes the correlation coefficient between the feature vector at position (i,j) in the first feature map and the feature vector $(i+d_1, j+d_2)$ in the second feature map. This coefficient quantifies the linear association's strength between two variables, with an interval range of (-1,1). A correlation coefficient close to 1 signifies a robust positive correlation, whereas one that approaches -1 reflects a significant negative relationship. This rewording retains the original meaning but presents it with different terminology to reduce redundancy. In cases where the two variables do not exhibit a linear relationship, the correlation coefficient tends towards 0.

The correlation calculation layer employs cosine similarity to serve as the correlation coefficient. Cosine similarity means the cosine between two vectors, indicating their directional similarity. The calculation of cosine similarity is based on the following formula:

$$\text{cosine similarity}(x,y) = \frac{x\cdot y}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad (1)$$

Where x and y represent two n-dimensional vectors. The correlation calculation layer can be implemented using the following steps:

(1) Pad the second feature map with D pixels in four directions so that the correlation can also be calculated at the boundary.

(2) Iterate over all possible displacement

 Copyright @ STEMM Institute Press

values (d_1,d_2) corresponding to different positions in the output matrix.

(3) For each displacement value (d_1,d_2), align the first and the second feature map according to the displacement value, and then calculate the dot product between the two feature maps, resulting in a $X \times Y$ submatrix.

(4) Normalization of each submatrix occurs through division by the multiplication of the norms of the related feature maps, yielding an $X \times Y$ submatrix. This submatrix quantifies the cosine similarity between the pair of feature maps.

(5) Concatenate all the submatrices to obtain a $(2D+1) \times (2D+1) \times X \times Y$ output matrix, which represents the correlation matrix between the two feature maps.

The spatio-temporal correlation calculation layer is implemented based on the principle of the correlation calculation layer. For a given set of sequences, use $I^{(t)} \in R^{C\times X\times Y}$ and $I^{(t+l)} \in R^{C\times X\times Y}$ to represent the input feature map pairs on a certain interval L. Then, calculate the similarity score at each position of $F^{(t)}$ and $F^{(t+l)}$. The formula is:

$$S\left(I^{(t)}, I^{(t+l)}\right) = \text{cosine similarity}(I^{(t)}_{(i,j)}, I^{(t+l)}_{(i+d_1,j+d_2,)}) \quad (2)$$
$$S \in R^{T\times X\times Y\times L\times(2D+1)\times(2D+1)}$$

Here $I^{(t)}$ and $I^{(t+l)}$ refer to the tensors of the t-th and t+l-th frames in a sequence, $(i,j)^{(t)}$ is the query coordinate, and $(i+d_1, j+d_2)^{(t+l)}$ is the spatio-temporal offset of the query, we limits its offset to its neighborhood through practice, with $(d_1,d_2) \in [-d_1, d_1] \times [-d_2, d_2$, finally, we specify $d_1 = \text{ceil}(X/2)$, $d_2 = \text{ceil}(Y/2)$, where ceil() is the ceiling function, this process is to convert the C-dimensional feature $I_{(x,y)}$ into the $(2D+1)^2$ -dimensional relation feature $S_{(x,y)}$. Among them, when $l = 0$, the tensor S represents spatial similarity, extracting semantic information in video frames; when $l = 1$, the tensor S represents adjacent-frame similarity, collecting motion information of short video sequences; when $l \neq 0,1$, the tensor S represents non-adjacent frames similarity, capturing motion information from farther away.

3.2 Feature Extraction

To extract motion features from the correlation tensor S, we implement a multilayer perceptron (MLP) approach capable of executing non-linear transformations on the input dataset and learn to map the high-dimensional correlation tensor to the a low-dimensional feature tensor.

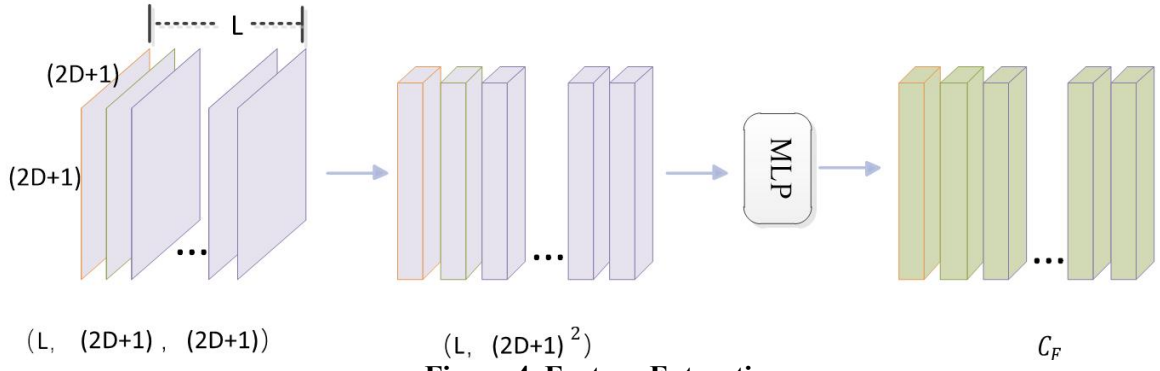

Figure 4. Feature Extraction

We separately perform MLP transformation on the spatial correlation tensor S_{spatio} and the temporal correlation $S_{temporal}$, and obtains the spatial feature tensor F_{spatio} and the temporal feature tensor $F_{temporal}$ (see Figure 4). Since the dimension of the correlation tensor S is high, for the convenience of subsequent processing, we adjust it to $S \in R^{T\times X\times Y\times L\times(2D+1)^2}$, where T represents the temporal dimension, X and Y denote the spatial dimensions, L denote the similarity dimension, and (2D+1) is the feature map dimension. Then, we use a method called n-mode tensor product, which can perform matrix multiplication on the specified dimension, and broadcast on other dimensions. Broadcasting automatically matches tensors of different shapes, which can copy and expand tensors without increasing memory overhead.

For the tensor $S \in R^{T\times X\times Y\times L\times(2D+1)^2}$, the perceptron $f(\cdot)$ can be represented as:

$$f(s) = \text{ReLU}\,(S\times_5 W_\phi) \quad (3)$$

Here, the symbol $\times_5$ represents the n-mode tensor product along the fifth dimension, $W_\phi \in R^{C' \times (2D+1)^2}$ represents the perceptron parameter, and ReLU[12] a non-linear activation function. The tensor $f(s) \in R^{T \times X \times Y \times C'}$ is a feature tensor, with its last dimension representing the motion feature dimension.

To extract higher-level motion features, we employ multiple perceptrons to perform function composition, which means taking the output of one function as the input to another function. For instance, if there are n perceptrons $(f_1, f_2, \ldots, f_n)$, their composition can be expressed as:

$$F = (f_n \circ f_{n-1} \circ \ldots \circ f_1)(x) = f_n(f_{n-1}(\ldots(f_1(x))\ldots)) \quad (4)$$

Where $\circ$ denotes the compound operation. The output $F \in R^{T \times X \times Y \times C_F}$ is a feature tensor, whose last dimension is the final motion feature dimension. This approach encodes displacement information and enables direct access to similarity values, offering advantages in learning motion distributions.

3.3 Feature Fusion

We aggregate the feature tensor $F \in R^{T \times X \times Y \times C_F}$, extracted through displacement mapping and similarity mapping, to facilitate its integration into the downstream layer. The tensor F undergoes processing through four depth-wise separable convolution layers: a $1 \times 7 \times 7$ layer followed by three $1 \times 3 \times 3$ layers, resulting in the generation of motion features Z that match the V dimension, as depicted in Figure 5. Compared with two-dimensional convolution, depth-wise separable convolution has lower parameter number and computational cost without sacrificing performance. Batch normalization[13] and ReLU[12] are applied after all depth-wise (DW) and point-wise (PW) layers to enhance the network's non-linear expression ability. The feature fusion process interprets the semantics of displacement mapping and similarity mapping through feature transformation, ultimately aggregates them.

Finally, we utilize the STC module as a residual block, combine the Z generated by this module with the input feature I through element-wise addition, and embed it into the GSF-ResNet-50[5] network model, thereby forming a fine-grained spatio-temporal feature learning network (FSTFL-Net).

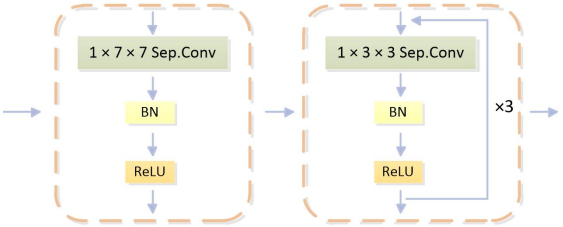

Figure 5. Feature Fusion

4. Experiments

For the sake of fairness, the baseline model is GSF-ResNet-50[5], and it is tested on four different datasets, namely Kinetics-400[14], UCF-101[15], and Sth-Sth V1 & V2[16], to showcase the generality of FSTFL-Net. Meanwhile, we perform numerous ablation study on the Sth-Sth V1 to analyze the best embedding position of the STC module. RGB frames are used in our experiments to reduce the computational cost. These experimental analyses lay the foundation for our work.

4.1 Datasets

We utilize various datasets FSTFL-Net, demonstrating its broad applicability. The datasets are introduced as follows.

(1) Something-Something V1 & V2 (Sth-Sth V1 & V2: These two datasets are large-scale action recognition datasets, both with 174 categories. Sth-Sth V1 comprises 108499 videos, while its extended version Sth-Sth V2 contains 220847 videos. The biggest difference from general datasets is that this dataset focuses especially on temporal relationships, such as "put sth from left or right".

(2) Kinetics-400 and UCF-101: Kinetics-400 contains a wide range of subject-as-human behaviors with 400 categories each with at least 400 video clips, covering a wide range of categories. The UCF-101 dataset originates from YouTube and encompasses 101 distinct categories, featuring a total of 13320 video clips. The data is relatively less and appeared earlier. Compared with Sth-Sth V1 & V2, these two datasets focus more on understanding scene information, and have weak temporal coherence.

4.2 Training

We initialize the ImageNet pre-training weight with ResNet50[17] (refer to Table 1), using the same training method as GSF-ResNet-50[5].

We utilize a frame sampling approach,

 Copyright @ STEMM Institute Press

selecting either 8 or 16 frames per video for action prediction. The optimization algorithm utilized is SGD, momentum of 0.9, and the batch size of 32. The initial learning rate is established at 0.01, with a weight decay of 10^{-4}. We adopt a cosine annealing learning rate strategy, incorporating a 10-epoch warmup period. Data augmentation techniques proposed by L. Wang et al.[18], such as random scaling, cropping, and flipping, are employed. For Sth-Sth V1 & V2 and Kinetics 400 datasets, dropout of 0.5 is used for training over 60 epochs. For training on the UCF-101, we further refine the pre-trained model on the Kinetics 400, and set dropout to 0.8 to prevent overfitting, training for 40 epochs.

Table 1. We Used the Same Training Strategy As GSF-ResNet-50 to Initialize the ImageNet Pre-Training Weights with the ResNet50[17]

Stage	Layer	Output size
raw	-	224×224
$conv_1$	7×7,stride 2,2	112×112
$pool_1$	3×3 max,stride 2,2	56×56
res_2	$\begin{bmatrix} 1\times1,64 \\ 3\times3,64 \\ 1\times1,256 \end{bmatrix}\times3$	56×56
res_3	$\begin{bmatrix} 1\times1,128 \\ 3\times3,128 \\ 1\times1,512 \end{bmatrix}\times4$	28×28
res_4	$\begin{bmatrix} 1\times1,256 \\ 3\times3,256 \\ 1\times1,1024 \end{bmatrix}\times6$	14×14
res_5	$\begin{bmatrix} 1\times1,512 \\ 3\times3,512 \\ 1\times1,2048 \end{bmatrix}\times3$	7×7
Global average pool,fc		1×1

4.3 Testing

For the Sth-Sth V1 & V2 datasets, we adopted two testing schemes. Scheme one involves using an 8 or 16-frame clip as input from the video, followed by cropping the central part (224×224) of the video. This testing scheme was more efficient. Scheme two, three 224×224 cropping methods to test from 10 randomly extracted clips, providing higher prediction accuracy. Our evaluation included both the forecast for an individual clip and the collective average prediction derived from ten randomly selected clips. The evaluation on the Kinetics 400 involved averaging the predictions from 10 clips uniformly sampled from each video. We evaluated two clips uniformly extracted from each video on the UCF-101.

4.4 Experimental Analysis

4.4.1 The comparative results with the baseline models

To assess the performance of the proposed FSTFL-Net, we applied identical training and testing methodologies as the baseline GSF-ResNet-50 network. As shown in Table 2, on the large-scale Kinetics-400 dataset that focuses on scene information, FSTFL-Net improved the Top-1 accuracy slightly by 0.74% compared to the GSF-ResNet-50 network. However, on the Sth-Sth V1 & V2, which focus on temporal information, the FSTFL-Net improved the Top-1 accuracy by 2.65% and 1.07%. This demonstrates that that FSTFL-Net significantly enhanced the capability to capture fine-grained information in both short and long video sequences.

Table 2. The Comparison between the GSF and FSTFL-Net on Different Datasets When Inputting 8 Frames, the Symbol "-" Means no Results are Given

Dataset	Model	Top-1(%)	Top-5(%)	ΔTop-1(%)
Sth-Sth V1	GSF-ResNet-50	48.36	-	+2.65
	Ours	**51.01**	**79.56**	
Sth-Sth V2	GSF-ResNet-50	61.46	666	+1.07
	Ours	**62.53**	**88.04**	
Kinetics-400	GSF-ResNet-50	74.74	-	+0.74
	Ours	**75.48**	**92.12**	

4.4.2 Comparison with domestic and international advanced level

To verify the generality of the temporal modeling capability of FSTFL-Net, we compared its performance with others on the Sth-Sth V1 & V2, Kinetics-400, and UCF-101. As illustrated in Tables 3, 4 and 5.

Table 3 compares the recognition performance of FSTFL-Net and other methods on the Sth-Sth V1 & V2. To ensure a fair comparison, most experiments utilized ResNet50 as the backbone. This table is divided into two sections: the first part lists the results of some methods employing 2D CNN or 3D CNN to

achieve video classification, and the second part shows the performance of FSTFL-Net. As shown in table 3, when the number of input frames is consistent, our method demonstrates slightly higher accuracy compared to 2D CNN models like TEA and TANet, as well as 3D CNN models like I3D on the Sth-Sth V1. While Martinez et al. achieved superior performance using I3D ResNet-152 compared to our method, we maintained a slight advantage when utilizing ResNet50 as the backbone. Overall, the temporal modeling capabilities of FSTFL-Net have been significantly enhanced.

Table 4 showcases a performance evaluation between FSTFL-Net and various other models on the Kinetics-400, highlighting differences in results. The table indicates that when FSTFL-Net utilizes 8 frames as input, it achieves a Top-1 accuracy of 75.48%, surpassing the R(2+1)D model by 1.18%, which uses 32 frames as input. When all models employ 16 frames as input, FSTFL-Net outperforms models like TEA, TSM, and other models slightly. In comparison to the STM model, the accuracy is enhanced by 2.51% with only a 0.05% increase in GFlops. Even when using 8 frames as input, FSTFL-Net's accuracy is still better than the STM model that uses 16 frames as input. This shows that FSTFL-Net has excellent feature extraction ability, which can effectively extract and express the semantic content and dynamic features of video sequences.

Table 5 provides compares of FSTFL-Net with various methods on the UCF-101 dataset. Following the same processing method as TSM and TEA, the table displays the average TOP-1 accuracy of three different cropping methods. Compared with the TEA method, our method still has room for improvement. It holds a slight advantage over the 3D CNN methods (like I3D, P3D) and the 2D CNN methods (like TSM) and among others.

Table 3. Performance Comparison with Domestic and International Advanced Level on Sth-Sth V1 & V2, the Symbol "-" Indicates that No Results are Given

Method	Backbone	Frames ×Crops× Clips	Params(M)	GFLOPs	Sth-Sth V1		Sth-Sth V2	
					Top-1(%)	Top-5(%)	Top-1(%)	Top-5(%)
TSN	BN-Inception	16 × 1 × 1	10.45	32.87 × 1 × 1	17.44	-	32.71	-
TSM	ResNet-50	16 × 1 × 1	24.3	65 × 1 × 1	47.3	-	61.2	-
bLVNet-TAM	bLResNet-50	16 × 1 × 2	25	47 × 1 × 2	48.4	78.8	61.7	88.1
CorrNet	ResNet-101	32×1 × 10	NA	224 × 1 × 10	51.1	-	-	-
TPN	ResNet-50	8 × 10 × 1	NA	NA	49.0	-	62	-
I3D	ResNet-50	32 × 1 × 2	28	153 × 1 × 2	41.6	72.2	-	-
NL-I3D	ResNet-50	32 × 1 × 2	35.3	168 × 1 × 2	44.4	76.0	-	-
TEA	ResNet-50	8× 1 × 1	NA	35 × 1 × 1	48.9	78.1	-	-
TEA	ResNet50	8×3×10	NA	35 × 3 × 10	51.7	80.5	-	-
MSNet	ResNet-50	8 × 1 × 1	24.6	34 × 1 × 1	50.9	80.3	63.0	88.4
MSNet	ResNet-50	16 × 1 × 1	24.6	67 × 1 × 1	52.1	82.3	64.7	89.4
GST	ResNet-50	8 × 1 × 1	21	29.5 × 1 × 1	47.0	76.1	61.6	87.2
GST	ResNet-50	16 × 1 × 1	21	59 × 1 × 1	48.6	77.9	62.6	87.9
GSF	ResNet-50	8 × 1 × 1	23.97	33.4 × 1 × 1	48.36	-	61.46	-
GSF	ResNet-50	16 × 1 × 1	23.97	66.8 × 1 × 1	50.37	-	63.41	-
TANet	ResNet50	8 × 1 × 1	NA	33 × 1 × 1	47.3	75.8	60.5	86.2
TANet	ResNet50	16 × 1 × 1	NA	66 × 1 × 1	47.6	77.7	62.5	87.6
Martinez et al.	ResNet-50	NA	NA	NA	50.1	-	-	-
Martinez et al.	ResNet-152	NA	NA	NA	**53.4**	-	-	-
Ours	ResNet-50	8 × 1 × 1	23.99	35.3 × 1 × 1	51.01	79.56	62.53	88.04
Ours	ResNet-50	16 × 1 × 1	23.99	70.6 × 1 × 1	52.45	81.06	64.25	89.13
Ours	ResNet-50	32 × 1 × 1	47.98	105.9 × 1×1	**53.21**	**81.93**	**66.12**	**90.05**

Table 4. Comparisons with Other Methods on the Kinetics-400, "N/A" Indicates That No Results are Given

Method	Pretrain	Frame	FLOPs × Views	Kinetics-400	
				Top-1(%)	Top-5(%)
I3D	Scratch	64	108G × N/A	72.1	90.3
R(2+1)D	Sports-1M	32	152G × 10	74.3	91.4
SlowFast	Scratch	8+32	106G × 30	77.9	93.2
TPN	Scratch	32	N/A	78.9	93.9
TSM	ImageNet	16	65G × 30	74.7	91.4
STM	ImageNet	16	67G × 30	73.7	91.6

 Copyright @ STEMM Institute Press

TEA	ImageNet	16	70G × 30	76.1	92.5
TANet-50	ImageNet	8	43G×30	76.3	92.6
TANet-50	ImageNet	16	86G×12	76.9	92.9
GSF	ImageNet	16	N/A	74.74	-
Ours	ImageNet	8	35.3G× 30	75.48	92.12
Ours	ImageNet	16	70.6G × 30	**76.21**	**92.84**

Table 5. Comparisons with Other Methods on the UCF-101 Dataset

Method	Pretrain	Backbone	UCF-101
P3D	ImageNe	ResNet50	88.6%
I3D	Kinetics	Inception V2	95.6%
TSM	Kinetics	ResNet50	96.0%
TEA	Kinetics	ResNet50	**96.9%**
STM	Kinetics	ResNet50	96.2%
Ours	Kinetics	ResNet50	96.86 %

Table 6. Comparison of Different Inputs on the Final Result after the STC Module is Placed in the Designated Layer

Layer	Frames	Top-1(%)	Top-5(%)
baseline	8	48.36	-
res2	8	48.41	77.35
res3	8	**51.01**	**79.56**
res4	8	49.25	78.19
res5	8	49.13	78.02
res2−3	8	50.86	79.13
res2−4	8	50.80	79.06
res2−5	8	50.77	78.95

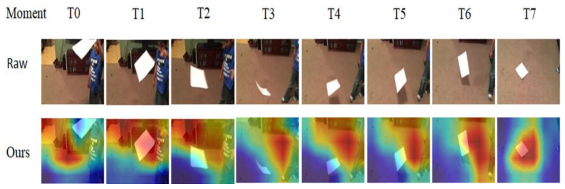

Figure 6. Grad-CAM Visualization Activation Map

4.4.3 Ablation study

To assess the impact of the STC module's embedding position within the baseline on the model's recognition capabilities, we conducted an ablation experiment on the Sth-Sth V1. The experiment involved sampling 8 frames, a batch size: 32, the momentum: 0.9, learning rate: 0.01, and a cosine learning rate schedule with a warmup period of 10 epochs for adjusting the learning rate. Additionally, a dropout rate of 0.5 was applied to prevent overfitting, and the training duration was set to 60 epochs. The results, presented in table 6, indicate that using the outputs of res2 and res3 as inputs to the STC module enables FSTFL-Net to achieve optimal performance. This is attributed to the output of res3 containing richer low-level features and capturing some temporal dynamic information.

4.4.4 Empirical analysis

To further understand the working principle of the FSTFL-Net model, when the input is 8 frames, we use Grad-CAM for visualization on the Sth-Sth V1. The visualization of the activation maps of FSTFL-Net, as shown in Figure 6, indicate that the model adeptly identifies the regions associated with the action.

5. Conclusion

An essential challenge in the realm of video understanding lies in analyzing fine-grained actions, which are those that look similar but have subtle differences. We introduce an enhanced version of FSTFL-Net, which can effectively utilize spatial information, short video sequence information, and long video sequence information, to achieve fine-grained action understanding. The key innovation is to extract fine-grained temporal dynamic information, which enables the model to distinguish highly similar actions. The findings showcase the effectiveness of FSTFL-Net in the domain of video understanding.

Acknowledgments

This paper receives support from the following projects:

A. The Major Special Project of the Ministry of Science and Technology-Theory and Methods for Security and Reliability of Green Internet of Things Powered by Artificial Intelligence. Project Number.: 2022YFE0138600. Sub-proiect Leader. Proiect Duration: January 2023 to December 2026, Funding Amount: 8.5 million CNY, Ongoing.

B. The National Natural Science Foundation of China project "Research on the Theory and Application of Cross-Media Collaborative Deep Security Situational Awareness in Artificial Intelligence" Project No.: 62266045, Project Duration: January 2023 to December 2026, Funding Amount: 330,000 CNY, Ongoing.

Copyright @ STEMM Institute Press

References

[1] Simonyan K, Zisserman A. Two-Stream Convolutional Networks for Action Recognition in Videos. Advances in neural information processing systems, 2014, 1. DOI: 10.1002/14651858.CD001941.pub3.

[2] Feichtenhofer C, Fan H, Malik J, et al. SlowFast Networks for Video Recognition // 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2019. DOI: 10.1109/ICCV.2019.00630.

[3] D. Zhang, X. Dai, and Y.-F. Wang, "Dynamic temporal pyramid network: A closer look at multi-scale modeling for activity detection," in Asian Conference on Computer Vision, 2018, pp. 712–728.

[4] Yang C, Xu Y, Shi J, et al. Temporal Pyramid Network for Action Recognition//2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2020. DOI: 10.1109/CVPR42600.2020.00067.

[5] Sudhakaran, S., Escalera, S., & Lanz, O. (2022). Gate-Shift-Fuse for Video Action Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*, 10913-10928.

[6] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V an Gool, "Temporal segment networks for action recognition in videos," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 11, pp. 2740–2755, 2019.

[7] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in Proc. of European Conference on Computer Vision, 2018, pp. 803–818.

[8] J. Lin, C. Gan and S. Han, "TSM: Temporal Shift Module for Efficient Video Understanding," 2019 IEEE / CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 7082 - 7092, doi: 10.1109/ICCV.2019.00718.

[9] Z. Liu, L. Wang, W. Wu, C. Qian and T. Lu, "TAM: Temporal Adaptive Module for Video Recognition," 2021 IEEE / CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 2021, pp. 13688-13698, doi: 10.1109/ICCV48922.2021.01345.

[10] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, "Motion feature network: Fixed motion filter for action recognition," in Proc. of European Conference on Computer Vision, 2018, pp. 387–403.

[11] H. Kwon, M. Kim, S. Kwak, and M. Cho, "Motionsqueeze: Neural motion feature learning for video understanding," in Proc. of European Conference on Computer Vision, 2020, pp. 345–362.

[12] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10). Omnipress, Madison, WI, USA, 807–814.

[13] loffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015).

[14] Carreira J, Zisserman A. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. IEEE, 2017. DOI: 10.1109/CVPR.2017.502.

[15] K. Soomro, A. R. Zamir, and M. Shah, "ucf101: A dataset of 101 human actions classes from videos in the wild," arXiv preprint. arXiv:1212.0402, 2012.

[16] R. Goyal, S. Kahou, V. Michalski, J. Materzynska, S. Westphal, H. Kim, V. Haenel, I. Fruend, P. Yianilos, M. Mueller-Freitag, et al. The "Something Something" Video Database for Learning and Evaluating Visual Common Sense. In Proc. Int. Conf. Comput. Vis., pages 5842–5850, 2017.

[17] He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. IEEE, 2016.DOI: 10.1109/CVPR.2016.90.

[18] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In Proc. Eur. Conf. Comput. Vis., pages 20–36, 2016.

 Copyright @ STEMM Institute Press