# Repetitive Action Counting with Motion Feature Learning

Xinjie Li, Huijuan Xu
Pennsylvania State University
University Park, USA
xql5497@psu.edu, hkx5063@psu.edu

## Abstract

*Repetitive action counting aims to count the number of repetitive actions in a video. The critical challenge of this task is to uncover the periodic pattern between repetitive actions by computing feature similarity between frames. However, existing methods only rely on the RGB feature of each frame to compute the feature similarity while neglecting the background change of repetitive actions. The abrupt background change may cause feature discrepancies of the same action moment and lead to errors in counting. To this end, we propose a two-branch framework,* i.e.*, RGB and motion branches, with the motion branch complementing the RGB branch to enhance the foreground motion feature learning. Specifically, foreground motion features are highlighted with flow-guided attention on frame features. In addition, to alleviate the noise from moving background distractors and reinforce the periodic pattern, we propose a temporal self-similarity matrix reconstruction loss to improve the temporal correspondence between the same motion feature from different frames. Lastly, to make the motion feature effectively supplement the RGB feature, we present a novel variance-prompted loss weights generation technique to automatically generate dynamic loss weights for two branches in collaborative training. Extensive experiments are conducted on the RepCount and UCFRep datasets to verify our proposed method with state-of-the-art performance. Our method also achieves the best performance on the cross-dataset generalization experiment.*

## 1. Introduction

In recent years, significant advancements have been achieved in the field of video understanding [2, 9, 21, 29]. Among the various tasks in video understanding, the repetitive action counting task has recently regained attention [6, 11, 42, 43]. This task entails the accurate counting of periodic activities in a video (Fig 1 (a)), with applications in fitness planning, physical education exams, etc.

Capturing the underlying periodic pattern between repetitive actions is the main challenge in this task. Early approaches [4, 23, 31] employ frequency analysis on compressed video signals but have limitations with complex actions and backgrounds. Recent advancements in large-scale benchmarks and deep learning methods have addressed these limitations [6, 11, 42, 43]. Zhang *et al.* [42] propose a context-aware framework to handle diverse cycle lengths. RepNet [6] uses a temporal self-similarity matrix to represent the periodic pattern and addresses the cross-dataset generalization. Hu *et al.* [11] introduce a transformer-based network for videos with varying lengths or frequencies. These developments enable the analysis of complex video sequences with diverse actions and backgrounds.

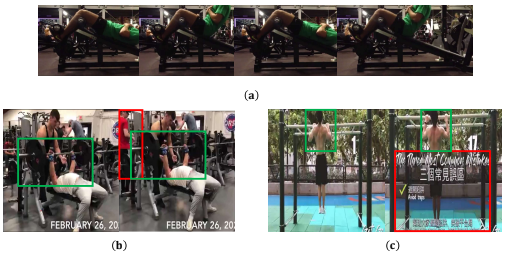


Figure 1. (a) Task Definition: Repetitive action counting requires accurately counting repetitive actions in a video, such as counting the number of sit-ups performed (e.g., 2 sit-ups). (b) &(c) Two examples with abrupt background changes (red rectangle). Despite belonging to the same action moment, frames exhibit distinct RGB features, highlighting the importance of capturing consistent motion features (green rectangle). The illustration videos are from RepCount dataset [11].

Despite the effectiveness of the aforementioned methods, they primarily focus on learning RGB features at the frame level, overlooking the diverse representation of each repetitive action. This limitation poses a significant challenge in uncovering the periodic pattern, particularly when frames indicating the same action moment exhibit significant variations in feature representation. As illustrated by the red rectangles in Fig.1 (b) and (c), frames can undergo drastic changes due to abrupt background alterations. In such scenarios, relying solely on RGB features of individual frames may not yield reliable results. Consequently, it becomes crucial to enhance the discriminative features that

contribute to determining the periodic pattern, namely the motion features depicted by the green rectangles in Fig.1 (b) and (c). By focusing on these motion features, we can improve the discriminative power of the model and address the challenges posed by varying backgrounds and visual variations within repetitive actions.

To address this challenge, we present a novel approach for learning motion features in the repetitive action counting task. Our method entails a two-branch framework, comprising an RGB branch and a dedicated motion branch, with the primary focus on complementing the RGB branch with motion-related information. To achieve this, we introduce a motion attention module, which utilizes flow features as queries, and RGB features as keys and values. This attention mechanism enables the integration of flow information, resulting in improved motion feature learning and more accurate counting of repetitive actions.

Despite the potential benefits of utilizing flow information for motion feature learning, it is crucial to address the issue of unexpected noise introduced by moving background distractors, as depicted by the red rectangle in Fig.1 (b). To mitigate this challenge, we turn to temporal correspondence learning, as demonstrated in prior works [15, 16, 19]. Temporal correspondence learning aims to establish connections between corresponding regions across different frames. By enhancing the temporal correspondence, we can effectively reduce the influence of moving background distractors, which typically appear sporadically in only a few frames. In contrast, the motion features associated with the repetitive actions persist across multiple frames. Strengthening the temporal correspondence not only helps alleviate the noise introduced by moving background distractors but also reinforces the connection between frames depicting the same action moment, thereby facilitating the identification of the underlying periodic pattern, as depicted in Fig. 2.

In this work, we propose a novel temporal self-similarity matrix reconstruction loss to facilitate the learning of temporal correspondence. Specifically, the temporal self-similarity matrix captures the similarities between different frames within a video. In the repetitive action counting task, one frame should be able to be represented by other frames with higher similarities, *i.e.*, the frames belonging to the same action moment. Thus, multiplying the frame features with the temporal self-similarity matrix should be able to output highly similar features, and our matrix reconstruction loss is designed to guarantee this effect by updating the temporal self-similarity matrix, thereby increasing the similarities between frames that exhibit periodic patterns.

To train the RGB and motion branches collaboratively, we propose a novel method for generating dynamic loss weights. Instead of manually assigning weights, our approach automatically determines the weights based on the variance of the video. When a video exhibits higher variance, indicating significant background changes, we assign larger weights to the motion branch. This allows the model to prioritize the motion branch in capturing the periodic pattern. By adapting the loss weights to the specific characteristics of each video, our method enhances the model's ability to learn discriminative features and accurately count repetitive actions.

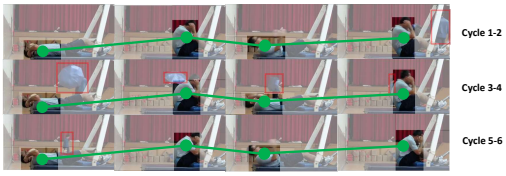


Figure 2. Enhancing temporal correspondence helps the model focus on consistent motion regions (green dot), disregarding background distractors (red rectangle). It also aids in uncovering the periodic pattern for accurate counting (green line). The illustration video is from RepCount dataset [11].

In summary, we present the following contributions in our work:

- We introduce a novel approach to enhance motion feature learning in the repetitive action counting task by incorporating flow information into a two-branch framework with a motion attention module.
- We propose a temporal self-similarity matrix reconstruction loss to improve the temporal correspondence between the same motion features across frames. This helps to uncover the periodic pattern by strengthening the connection between frames of the same action moment.
- We present a variance-integrated loss weights generation method to dynamically adjust loss weights based on video variance, enabling adaptive prioritization of the motion branch in the training process.
- Extensive experiments are conducted on the RepCount dataset and UCFRep dataset to show the effectiveness of our proposed method with state-of-the-art performance.

## 2. Related Work

In this section, we review prior works on repetitive action counting and temporal correspondence learning.

### 2.1. Repetitive Action Counting

Repetitive action counting has been a prominent focus of video analysis research for numerous years [1, 14, 22, 26, 27, 32, 33, 38–41]. Early approaches involve transforming videos into one-dimensional signals, from which frequency information used for counting is extracted through waveform analysis [23], singular value decomposition [4], or peak detection [31]. However, these methods assume strict

periodicity and a stationary context for the repetitive actions. To address non-stationary scenarios, Runia et al. [25] employ wavelet transform on the flow representation, leading to the design of a novel inference pipeline for detecting non-stationary repetitions. Nonetheless, hand-crafted features are still utilized in their approach. In another contribution, Levy and Wolf [17] introduce a CNN-based method, training the model on synthetic data.

Recently, Zhang et al. [42] propose a context-aware regression network with a coarse-to-fine refinement strategy to handle the challenge of diverse cycle lengths in repetitive action counting. Notably, they introduce the UCFRep dataset, which serves as the first large-scale real-life benchmark for the task. Concurrently, RepNet [6] address the cross-dataset generalization issue by incorporating a temporal self-similarity matrix into the repetitive action counting process. Additionally, they introduce the Countix dataset, which consists of real-world videos. Building upon the Countix dataset, Zhang et al. [43] incorporate corresponding soundtracks into the videos and proposed a two-stream framework to tackle the repetitive action counting task. More recently, Hu et al. [11] present a transformer-based multi-scale temporal correlation mechanism to handle both high-frequency and low-frequency actions in repetitive action counting. Furthermore, they introduce another large-scale benchmark called RepCount.

Although the aforementioned works have significantly advanced research in repetitive action counting, they primarily focus on learning RGB features for individual frames, despite the fact that frames belonging to the same action moment can exhibit dramatic changes. In contrast, we are the first to address motion feature learning in the repetitive action counting. To facilitate collaborative learning of motion and RGB features, we propose a variance-integrated loss weights generation technique. Diverging from traditional multi-task learning methods [3, 28], which determine fixed loss weights for the entire training process, our method dynamically generates loss weights in different iterations, enabling more adaptive and effective training.

### 2.2. Temporal Correspondence Learning

Temporal correspondence learning plays a pivotal role in various video tasks, such as video object segmentation [19, 20, 37], video temporal alignment [5, 10, 24], and video object tracking [35, 37]. Recent research in self-supervised learning has explored correspondence learning through two main directions: reconstruction-based methods [15, 16, 19] and cycle-consistency-based methods [13, 34, 35]. In the reconstruction-based approach, leveraging colorization as a pretext task, a query point is reconstructed from points in neighboring frames based on the temporal correspondence between frames. On the other hand, cycle-consistency-based methods typically employ bidirectional patch tracking between adjacent frames to enhance temporal correspondence by minimizing cycle consistency.

Motivated by the effectiveness of the aforementioned methods, we propose to learn temporal correspondence specifically in the context of repetitive action counting, where strong correspondence exists due to the periodic nature of the actions. Drawing inspiration from reconstruction-based methods, we introduce the idea of reconstructing the temporal self-similarity matrix to enhance the temporal correspondence between identical motion features across frames. This method aims to capture the underlying temporal structure and promote consistent representation of the same action throughout the video sequence.

## 3. Method

In this section, we present the details of our proposed motion feature learning method for the task of repetitive action counting. We formulate the repetitive action counting task as follows: the input video data is represented as $\mathbf{V} \in \mathbb{R}^{B\times 3\times F\times H\times W}$, where $B$ is the batch size, $F$ denotes the number of frames in each video, and $H$ and $W$ indicate the height and width, respectively. The corresponding label set is denoted as $\mathbf{Y_c} \in \mathbb{R}^{B\times 1}$, where each element in $\mathbf{Y_c}$ represents the repetitive action count for each video. During the training phase, with the annotations of the starting and ending points of each repetitive action, we utilize the ground truth density map $\mathbf{Y} \in \mathbb{R}^{B\times F}$.

The overall architecture of our model is depicted in Fig.3. It consists of two branches: the motion branch and the RGB branch, with the motion branch complementing the RGB branch. On the RGB branch, we adopt the pipeline proposed in a previous work [11] (Sec. 3.1). On the flow branch, we introduce a motion attention module (MAM) to enhance the learning of motion features by leveraging the flow information (Sec. 3.2). In both branches, we propose a matrix reconstruction loss to encourage temporal correspondence between the same motion features across different frames (Sec. 3.3). Additionally, we propose a variance-integrated technique for generating loss weights to collaboratively train the two branches (Sec. 3.4).

### 3.1. Transformer-based Repetitive Action Counting

In our framework, the RGB branch serves as a fundamental component, employing a transformer-based pipeline for repetitive action counting, as described in [11].

Firstly, we utilize a pre-trained RGB encoder, denoted as $\mathbf{Enc}_V$, to extract features from the input videos $\mathbf{V} \in \mathcal{R}^{B\times 3\times F\times H\times W}$. The output feature tensor $\mathbf{X} \in \mathcal{R}^{B\times C\times F\times H_1\times W_1}$, where $C$, $H_1$, and $W_1$ represent the number of output channels, height, and width, respectively, is obtained through the following operation:

$$\mathbf{X} = \mathbf{Enc}_V(\mathbf{V}) \tag{1}$$

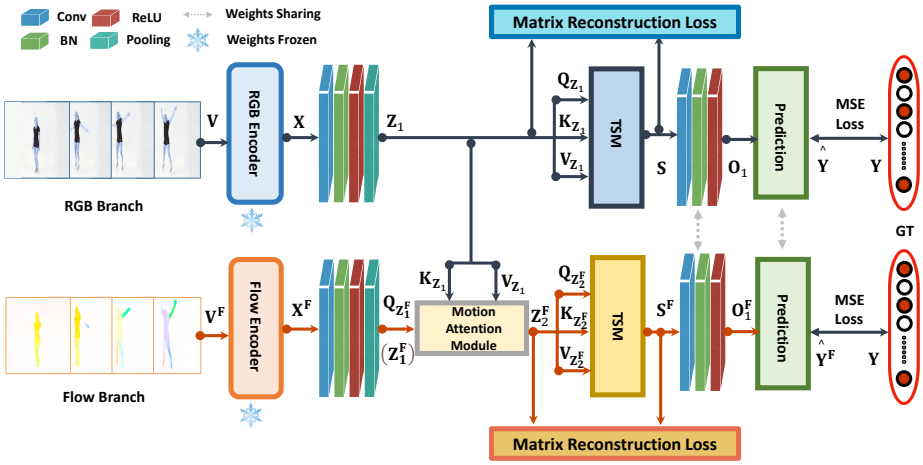


Figure 3. Overview of our Motion Feature Learning Framework. The framework consists of two branches: the RGB branch and the flow branch. In the RGB branch (Sec. 3.1), we begin by employing a pre-trained RGB encoder and a series of operators to extract features from the input data. Next, a Temporal Self-Similarity Module (TSM) is applied to generate the temporal self-similarity matrix. Finally, a sequence of operators and a prediction module are used to make the final prediction based on the temporal self-similarity matrix. In the flow branch (Sec. 3.2), we introduce motion information and fuse it with the RGB feature using a Motion Attention Module (MAM) to enhance the learning of motion features. Both branches incorporate two novel Matrix Reconstruction Losses (Sec. 3.3) to improve the temporal correspondence between the same motion features from different frames. These losses encourage better alignment and consistency of motion features throughout the video. During the training stage, we propose a Variance-Integrated Loss Weights Generation Technique (Sec. 3.4) to dynamically generate loss weights for the two branches. This technique allows the model to adaptively assign higher importance to specific branches based on the variance of the training samples, leading to better performance. In the inference stage, counting results are generated from the predicted density maps $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}^{\mathbf{F}}$ obtained from the RGB branch and the motion branch, respectively. The final output is obtained by averaging the results from both branches, providing a more robust repetitive action count.

Next, we apply a sequence of operators, including Conv3D, BN, ReLU, and Pooling layers, to reduce the dimension, height, and width of the features:

$$\mathbf{Z} = \text{Pooling}\left(\text{ReLU}\left(\text{BN}\left(\text{Conv3D}\left(\mathbf{X}\right)\right)\right)\right) \tag{2}$$

Here, $\mathbf{Z} \in \mathcal{R}^{B\times C_1\times F\times 1\times 1}$. Reshape and transpose operations are then applied to obtain the output feature $\mathbf{Z_1} \in \mathcal{R}^{B\times F\times C_1}$, which is subsequently fed into the temporal self-similarity module.

The temporal self-similarity module (TSM) employs a multi-head self-attention layer, where the input $\mathbf{Z_1} \in \mathcal{R}^{B\times F\times C_1}$ is duplicated three times to create query $\mathbf{Q_{Z_1}}$, key $\mathbf{K_{Z_1}}$, and value $\mathbf{V_{Z_1}}$. The TSM generates the temporal self-similarity matrix $\mathbf{S} \in \mathcal{R}^{B\times D\times F\times F}$ through the following operation:

$$\mathbf{S} = \mathbf{TSM}(\mathbf{Q_{Z_1}}, \mathbf{K_{Z_1}}, \mathbf{V_{Z_1}}) \tag{3}$$

Here, $D$ represents the number of heads in the multi-head self-attention layer. The temporal self-similarity matrix $\mathbf{S}$ is then passed through a sequence of Conv3D, BN, and ReLU layers to increase its dimension, resulting in $\mathbf{O} \in \mathcal{R}^{B\times C_2\times F\times F}$:

$$\mathbf{O} = \text{ReLU}\left(\text{BN}\left(\text{Conv3D}\left(\mathbf{S}\right)\right)\right) \tag{4}$$

After applying transpose and reshape operations, we obtain the output feature $\mathbf{O_1} \in \mathcal{R}^{B\times F\times F\times C_2}$.

Finally, the previous output feature $\mathbf{O_1}$ is fed into the prediction model, which consists of a multi-head self-attention layer and a sequence of linear layers, to obtain the final predicted density map $\hat{\mathbf{Y}} \in \mathcal{R}^{B\times F}$:

$$\hat{\mathbf{Y}} = \text{Prediction}(\mathbf{O_1}) \tag{5}$$

For the prediction loss, the mean square error (MSE) is utilized, given by:

$$\mathbf{L_P} = \text{MSE}(\hat{\mathbf{Y}}, \mathbf{Y}) \tag{6}$$

In summary, the RGB branch employs an RGB encoder, temporal self-similarity module, and prediction model to extract features, capture temporal relationships, and make predictions for the repetitive action counting task.

### 3.2. Flow-guided Motion Feature Learning

In this section, we present our proposed flow-guided motion feature learning scheme, which aims to capture the critical motion features in repetitive actions.

In repetitive actions, the same action moment in different periodicities can exhibit significant variations. Therefore, relying solely on RGB features may not be reliable. To address this, we introduce motion feature learning to the repetitive action counting task. Recognizing that motion features are crucial in repetitive actions, we propose a motion attention module (MAM) that utilizes flow features to implicitly locate motion features within RGB features.

The **Motion Attention Module (MAM)** takes flow inputs $\mathbf{V^F} \in \mathcal{R}^{B\times 3\times F\times H\times W}$ and follows a similar process as described in Eq.1 and Eq.2 to obtain the processed flow feature $\mathbf{Z^F} \in \mathcal{R}^{B\times C_1\times F\times 1\times 1}$. By reshaping and transposing, we obtain $\mathbf{Z_1^F} \in \mathcal{R}^{B\times F\times C_1}$, which is denoted as the query $\mathbf{Q_{Z_1^F}}$. As shown in Fig. 3, the MAM takes the RGB feature's key $\mathbf{K_{Z_1}}$ and value $\mathbf{V_{Z_1}}$ as additional inputs. These inputs are passed through a multi-head cross-attention layer, yielding the output $\mathbf{Z_2^F} \in \mathcal{R}^{B\times F\times C_1}$. By employing flow information as the query and RGB information as the key and value, we enhance the feature learning of constantly-moving motion features, which are crucial for uncovering the periodic pattern.

Next, we duplicate the processed feature $\mathbf{Z_2^F}$ three times to construct the query $\mathbf{Q_{Z_2^F}}$, key $\mathbf{K_{Z_2^F}}$, and value $\mathbf{V_{Z_2^F}}$. Similarly to Eq. 3, 4, and 5, we obtain the temporal self-similarity matrix $\mathbf{S^F} \in \mathcal{R}^{B\times H\times F\times F}$ and the final predicted density map $\hat{\mathbf{Y}}^\mathbf{F} \in \mathcal{R}^{B\times F}$. The loss function is calculated using the mean square error (MSE) as follows:

$$\mathbf{L_P^F} = \text{MSE}(\hat{\mathbf{Y}}^\mathbf{F}, \mathbf{Y}) \tag{7}$$

In summary, the proposed flow-guided motion feature learning scheme incorporates a motion attention module to capture motion features within repetitive actions. By leveraging flow features, the scheme improves the ability to uncover the periodic pattern and provides a complementary aspect to the RGB branch for accurate action counting.

### 3.3. Matrix Reconstruction for Temporal Correspondence

In Section 3.2, we present a straightforward yet effective approach to augment the learning of motion features. However, it is important to note that not all motion features contribute to repetitive actions, such as unexpected moving background distractors. To mitigate the influence of undesired moving background distractors, it is necessary to enhance the temporal correspondence between identical motion features extracted from different frames. This is because not all frames contain randomly-occurring moving background distractors, but they all encompass consistently-moving motion features. Furthermore, improving temporal correspondence can also reinforce the detection of periodic patterns and yield advantages for the final counting task, as demonstrated in Fig. 2.

Motivated by recent advancements in correspondence learning [15, 16, 19], we propose a novel loss function, termed temporal self-similarity matrix reconstruction loss, to boost the temporal correspondence among identical motion features from diverse frames. Specifically, our loss function relies on the fundamental observation that a single frame shares a high degree of similarity with other frames belonging to the same action moment and can be effectively

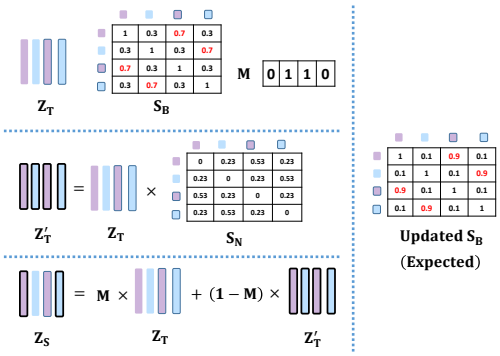


Figure 4. A simple illustration of our proposed matrix reconstruction process. $\mathbf{S_N}$ is normalized from $\mathbf{S_B}$. On the right ($\mathbf{Updated\ S_B}$) is our expected temporal self-similarity matrix after update. (Batch size $B = 1$, frame quantity $F = 4$, and counting number is 2 in this illustration example.)

represented by these frames. Consequently, our primary objective is to update the temporal self-similarity matrix to amplify the similarities between periodic frames, thereby enhancing the temporal correspondence.

Fig. 3 illustrates the architecture of our method. In the RGB branch, the matrix reconstruction module takes as inputs the RGB feature $\mathbf{Z_1}$ and the temporal self-similarity matrix $\mathbf{S}$. Similarly, in the flow branch, the inputs are $\mathbf{Z_2^F}$ and $\mathbf{S^F}$. For the sake of brevity, we describe the details using the RGB branch as an example, and provide a concise depiction in Fig. 4.

To elaborate on the matrix reconstruction process, we outline the steps involved:

Firstly, we compute the mean value of the RGB feature tensor $\mathbf{Z_1} \in \mathbb{R}^{B\times F\times C_1}$ across the channel dimension, resulting in the reconstruction target $\mathbf{Z_T} \in \mathbb{R}^{B\times F}$. Additionally, we calculate the mean value of the temporal self-similarity matrix $\mathbf{S} \in \mathbb{R}^{B\times H\times F\times F}$ along the second dimension, which gives us the reconstruction basis $\mathbf{S_B} \in \mathbb{R}^{B\times F\times F}$. Notably, each row of the $F \times F$ matrix within $\mathbf{S_B}$ captures the similarities between the frame in that row and the other frames.

Secondly, we set the diagonal elements of each $F \times F$ matrix in $\mathbf{S_B}$ to zero, as they represent the self-similarity (which is equal to 1) of the frame in each corresponding row. Preserving these elements in the matrix would render the reconstruction process meaningless. Subsequently, we normalize the remaining elements in each matrix by ensuring that the sum of each row equals 1. This normalization step yields the normalized temporal self-similarity matrix $\mathbf{S_N} \in \mathbb{R}^{B\times F\times F}$.

Thirdly, we perform the reconstruction of the target feature $\mathbf{Z_T}$ based on the normalized temporal self-similarity matrix $\mathbf{S_N}$:

$$\mathbf{Z'_T} = \mathbf{Z_T} \times \mathbf{S_N} \tag{8}$$

Here, $\mathbf{Z'_T} \in \mathbb{R}^{B\times F}$. Since representing all frames could be

computationally challenging, we selectively process a subset of frames. To achieve this, we randomly generate a mask $\mathbf{M} \in \mathbb{R}^{B\times F}$, where each element $\mathbf{m}$ is a shuffled list which contains $i$ zeros and $j$ ones , and $i + j = F$.

Finally, we calculate the reconstruction source $\mathbf{Z_S} \in \mathbb{R}^{B\times F}$ as follows:

$$\mathbf{Z_S} = \mathbf{M} \times \mathbf{Z_T} + (\mathbf{1} - \mathbf{M}) \times \mathbf{Z'_T} \tag{9}$$

The mean square error (MSE) loss is employed as the reconstruction loss:

$$\mathbf{L_R} = \text{MSE}(\mathbf{Z_S}, \mathbf{Z_T}) \tag{10}$$

The same procedure applies to the flow branch, yielding the flow reconstruction loss:

$$\mathbf{L^F_R} = \text{MSE}(\mathbf{Z^F_S}, \mathbf{Z^F_T}) \tag{11}$$

Through the training of the two matrix reconstruction processes, the temporal self-similarity matrices $\mathbf{S}$ and $\mathbf{S^F}$ are updated. Consequently, the similarities between periodic frames, indicating the occurrence of the same action moment, are increased, thereby enhancing the detection of periodic patterns. Simultaneously, the influence of moving background distractors is diminished, as higher similarities consistently arise from the same constantly-moving motion features rather than randomly-appearing background elements. It is important to note that during the training of this process, we freeze the modules preceding the Temporal Self-Similarity Matrix (TSM) modules in Fig. 3. This is necessary as the matrix reconstruction requires fixed target features $\mathbf{Z_T}$ and $\mathbf{Z^F_T}$ as templates. Without fixed targets, the reconstruction process would yield random results and detrimentally affect the temporal self-similarity matrices $\mathbf{S_B}$ and $\mathbf{S^F_B}$.

### 3.4. Variance-Integrated Loss Weights Generation

When considering the importance of the two predictions, $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}^F$, a common approach is to manually assign different loss weights to $\mathbf{L_P}$ and $\mathbf{L^F_P}$. However, this manual assignment of loss weights can be time-consuming and often leads to sub-optimal results. Moreover, in our task, the significance of the motion branch varies depending on the characteristics of the video, specifically, the presence of drastic background changes or high variance.

To address this issue, we propose a variance-integrated method for generating dynamic loss weights for the two branches. The dynamic loss weights capture the importance of each branch, with a higher weight assigned to the motion branch when the video exhibits higher variance. Our approach automatically adjusts the loss weights based on the video characteristics, making it more adaptable and better suited for our task compared to using fixed loss weights throughout the training process.

The process for generating dynamic loss weights is as follows:

Firstly, we calculate the variance $\mathbf{var}$ of each video $\mathbf{v}$: we first transform the video $\mathbf{v} \in \mathbb{R}^{3\times F\times H\times W}$ to $\mathbf{v} \in \mathbb{R}^{F\times 3HW}$. Then, we calculate the variance value along the second dimension, i.e., the variance of $F$ different features.

Next, we concatenate the batched variance $\mathbf{VAR} \in \mathbb{R}^{B\times 1}$ with the prediction $\hat{\mathbf{Y}}^F$ obtained from the motion feature, and concatenate $\mathbf{1} - \mathbf{VAR}$ with the prediction $\hat{\mathbf{Y}}$ derived from the RGB feature. Subsequently, we pass the concatenated tensors through separate linear layers, followed by sigmoid layers, to generate the two loss weights:

For the motion branch:

$$\mathbf{W^F} = \text{Sig}(\text{Linear}(\text{Concat}(\mathbf{VAR}, \hat{\mathbf{Y}}^F))) \tag{12}$$

For the RGB branch:

$$\mathbf{W} = \text{Sig}(\text{Linear}(\text{Concat}(\mathbf{1} - \mathbf{VAR}, \hat{\mathbf{Y}}))) \tag{13}$$

By applying this method, we can dynamically assign loss weights during different iterations of training. This design allows for automatic adaptation to varying video characteristics, making it more generalizable and better suited for our specific task compared to fixed loss weights throughout the entire training process.

### 3.5. Optimization

During the training stage, the loss function is defined as follows:

$$\mathbf{L} = \mathbf{W} \times \mathbf{L_P} + \mathbf{W^F} \times \mathbf{L^F_P} + \mathbf{L_R} + \mathbf{L^F_R} \tag{14}$$

In the inference period, the predictions $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}^F$ are combined to generate the final prediction density map $\bar{\mathbf{Y}}$:

$$\bar{\mathbf{Y}} = (\hat{\mathbf{Y}} + \hat{\mathbf{Y}}^F)/2 \tag{15}$$

Finally, the final counting number $\bar{\mathbf{Y}}_\mathbf{c}$ is obtained by summing the elements of the density map:

$$\bar{\mathbf{Y}}_\mathbf{c} = \text{Sum}(\bar{\mathbf{Y}}) \tag{16}$$

This process allows us to generate an aggregated prediction that combines the outputs of both the RGB and motion branches. The final counting number is then derived by summing the elements of the density map, providing an estimation of the total count.

## 4. Experiments

### 4.1. Datasets

To train and evaluate our proposed method, we utilize two widely adopted large-scale benchmarks, following the approach of prior work [11]. The **RepCount** dataset [11] consists of two parts: Part A, which is collected from

YouTube, and Part B, recorded from a local school environment. Part A encompasses a diverse range of action types, including workout events, athletic activities, and more. On the other hand, Part B focuses on students' daily exercises such as pulling up. Part A comprises a total of 1041 videos, with 757 videos allocated for training, 130 videos for validation, and 151 videos for testing. It's worth noting that in our experiments, we only utilize Part A, as Part B is not publicly available. The **UCFRep** dataset introduced by Zhang et al. [42] is derived from the UCF101 dataset [29]. This dataset consists of 526 videos covering 23 different repetition action classes. Among these, 421 videos are assigned to the training set, while the remaining 105 videos form the test set. In addition to the standard training and testing pipeline, we also employ this dataset for cross-dataset generalization evaluation, following [11].

### 4.2. Experimental Settings

**Implementation Details.** We employ the Adam optimizer with a multi-step decay scheduler to train our model. The training is conducted for 200 epochs using a batch size of 128. The training process is accelerated using four NVIDIA RTX A5000 GPUs with automatic mixed precision enabled. Optical flow maps are generated using the RAFT algorithm [30]. In all our experiments, we consider video sequences of 64 frames as input, following the approach of TransRAC [11]. To ensure a fair comparison, we adopt multi-scale video sequences as input data. The number of heads in the multi-head self-attention layers is set to 4, while the number of heads in the multi-head cross-attention layer is set to 16. For the encoder and input data size, in the experiments conducted on the RepCount dataset [11], we utilize the Video Swin Transformer Tiny model [21] pre-trained on the Kinetics dataset [2]. The input size for this experiment is set to $224 \times 224$. In the experiments performed on the UCFRep dataset [42], we adopt the 3D-ResNext101 model [9, 36] pre-trained on the Kinetics dataset as our encoder. The input size for this experiment is set to $112 \times 112$ to ensure a fair comparison with previous methods [42, 43]. It is important to note that we re-implement the previous methods [42, 43] on the UCFRep dataset using the same amount of input data and the same encoder to facilitate a meaningful comparison.

**Evaluation Metrics.** Following previous works [6, 11, 42, 43], we adopt the Mean Absolute Error (MAE) and the Off-By-One (OBO) accuracy as our evaluation metrics, which are defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\widetilde{c_i} - c_i|}{\widetilde{c_i}} \tag{17}$$

$$\text{OBO} = \frac{1}{N}\sum_{i=1}^{N}[|\widetilde{c_i} - c_i| \leq 1] \tag{18}$$

| Method | MAE↓ | OBO ↑ |
|---|---|---|
| RepNet [6] | 0.9950 | 0.0134 |
| X3D [8] | 0.9105 | 0.1059 |
| Context [42] | 0.8786 | 0.1554 |
| TANet [7] | 0.6624 | 0.0993 |
| Video SwinT [21] | 0.5756 | 0.1324 |
| Huang *et al*. [12] | 0.5267 | 0.1589 |
| TransRAC [11] | 0.4431 | 0.2913 |
| Full [18] | 0.4103 | 0.3267 |
| **Ours** | **0.3841** | **0.3860** |

Table 1. Experimental results on the RepCount dataset [11]. The results of previous methods are copied from TransRAC [11].

| Method | MAE↓ | OBO ↑ |
|---|---|---|
| RepNet [6]∗ | 0.9985 | 0.0090 |
| Context [42]∗ | 0.7620 | 0.4120 |
| TransRAC [11]∗ | 0.6401 | 0.3240 |
| Zhang *et al*. [43]† | 0.4825 | 0.3125 |
| Context [42]† | 0.4689 | 0.4800 |
| Full [18]∗ | 0.4608 | 0.3333 |
| TransRAC [11]† | 0.4409 | 0.4300 |
| **Ours** | **0.3879** | **0.5100** |

Table 2. Experimental results on the UCFRep dataset [42]. ∗: copied from Full [18]. †: re-implemented by our own for fair comparison (see supplementary material for more details).

Precisely, given $N$ videos, the MAE measures the normalized absolute error between prediction count $c$ and the ground truth $\widetilde{c}$, while the OBO estimates the rounding error with 1 as the error margin.

### 4.3. Comparison with State-of-the-art Methods

**Results on RepCount.** In the evaluation on the RepCount dataset [11], we conduct a comprehensive comparison of our method with previous video recognition methods [7, 8, 21] as well as recent approaches in repetitive counting [6, 11, 42]. The results, presented in Table 1, demonstrate that our method outperforms previous techniques by a significant margin. Particularly, our approach surpasses the previous state-of-the-art method TransRAC [11] by $5.90\%$ in MAE and $9.47\%$ in OBO.

**Results on UCFRep.** In the evaluation on the UCFRep dataset [42], we compare our method with recent approaches in repetitive action counting. As presented in Table 2, our method achieves superior performance compared to previous techniques in terms of MAE. Specifically, we outperform the previous state-of-the-art method TransRAC [11] by $5.3\%$ in MAE. Furthermore, in terms of OBO, our method surpasses the previous state-of-the-art method Context [42] by $3\%$.

**Generalization.** Following [11], we also conduct the cross-dataset generalization experiment, *i.e*., training on the RepCount dataset [11] while testing on the UCFRep

| Method | MAE↓ | OBO ↑ |
|---|---|---|
| RepNet [6] | 0.9985 | 0.0090 |
| TransRAC [11] | 0.6401 | 0.3240 |
| **Ours** | **0.5227** | **0.3500** |

Table 3. Experimental results of the cross-dataset generalization evaluation on the UCFRep dataset [42]. The results of previous methods are copied from TransRAC [11].

| Flow | MAM | R | LW | MAE↓ | OBO ↑ |
|---|---|---|---|---|---|
| | | | | 0.4736 | 0.2450 |
| ✓ | | | | 0.4808 | 0.2715 |
| ✓ | ✓ | | | 0.4159 | 0.3377 |
| ✓ | ✓ | ✓ | | 0.3909 | 0.3576 |
| ✓ | ✓ | ✓ | ✓ | **0.3841** | **0.3860** |

Table 4. Ablation analysis on each component of our method. 'R' indicates reconstruction and 'LW' denotes loss weights.

dataset [42]. As shown in Table 3, our method outperforms the previous state-of-the-art method by $11.74\%$ in MAE and $2.8\%$ in OBO.

### 4.4. Ablation Analysis

In this section, we show ablation results on the RepCount dataset [11]. See supplementary material for more analysis.

**The effectiveness of each component:** As depicted in Table 4, we analyze the impact of each component individually. Initially, simply processing the flow information as RGB information in the global branch results in poor performance. However, after incorporating our motion attention module (MAM), which leverages flow features to enhance motion feature learning implicitly, we observe a significant improvement of $6.49\%$ in MAE and $6.62\%$ in OBO. This highlights the effectiveness of our MAM in capturing motion patterns. Moreover, integrating the matrix reconstruction loss for temporal correspondence further enhances the performance by $2.5\%$ in MAE and $1.99\%$ in OBO. Lastly, our proposed variance-integrated loss weights generation method contributes an additional improvement of $0.68\%$ in MAE and $2.84\%$ in OBO. These results collectively demonstrate the effectiveness of each component in our method.

**Reconstruction on different branches:** To investigate the impact of applying matrix reconstruction loss to different branches, we perform experiments and present the results in Table 5. We observe that applying the reconstruction loss to both branches yields the best performance, indicating that incorporating temporal correspondence through reconstruction in both the RGB and motion branches is advantageous for counting repetitive actions accurately.

**The temporal self-similarity matrix:** We calculate the $L_2$ distance between the ground truth temporal self-similarity matrix and the predicted matrix. The ground truth matrix is constructed based on the annotations of starting point and ending point of each repetitive action. Specifically, the dimensions of the matrix are $64 * 64$ and we put every value indicating the same moment of repetitive actions to 1 and others to 0. We show the mean distance of the test set from the RepCount dataset [11] in Table 6. As we can see from this table, the temporal self-similarity matrix is much more similar to the ground truth with the reconstruction process.

| Flow | RGB | MAE↓ | OBO ↑ |
|---|---|---|---|
| | | 0.4159 | 0.3377 |
| ✓ | | 0.4030 | 0.3444 |
| | ✓ | 0.3929 | 0.3444 |
| ✓ | ✓ | 0.3909 | 0.3576 |

Table 5. Ablation study of applying matrix reconstruction loss to different branches.

| | Distance↓ |
|---|---|
| w/o. Recon | 26.59 |
| w. Recon | 20.32 |

Table 6. The distance between the predicted temporal self-similarity matrix and ground truth matrix with or without the reconstruction process.

| Variance | Prediction | MAE↓ | OBO ↑ |
|---|---|---|---|
| | | 0.3909 | 0.3576 |
| ✓ | | 0.3947 | 0.3642 |
| | ✓ | 0.3910 | 0.3775 |
| ✓ | ✓ | **0.3841** | **0.3860** |

Table 7. The effect of different inputs on the variance-integrated loss weights generation. The first row indicates that no loss weights are used.

**The inputs of variance-integrated loss weights generation.** We analyze the effect of different inputs on the loss weights generation. As shown in Table 7, both the variance and output predictions $\hat{\mathbf{Y}}$ and $\hat{\mathbf{Y}}^{\mathbf{F}}$ are beneficial to our automatic loss weights generation.

## 5. Conclusion

In this paper, we propose a two-branch framework for the repetitive action counting task with the motion branch supplementing the RGB branch. Specifically, we propose a simple motion attention module that uses flow information to implicitly enhance motion feature learning. In addition, to alleviate the noise from moving background distractors, we propose a temporal self-similarity matrix reconstruction loss to improve the temporal correspondence. Lastly, we introduce a novel variance-integrated loss weights generation technique that uses variance as guidance to automatically generate dynamic loss weights for two branches. Extensive experimental results and cross-dataset generalization have proven the effectiveness of our method. In the future, repetitive action counting with multiple repetitive objects should be addressed. In addition, weakly supervised or even self-supervised methods should be considered and the combination with other modalities such as language is promising.

## References


[1] Alexia Briassouli and Narendra Ahuja. Extraction and analysis of multiple periodic motions in video sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007. 2

[2] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4724–4733. IEEE Computer Society, 2017. 1, 7

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 3

[4] Dmitry Chetverikov and Sándor Fazekas. On motion periodicity of dynamic textures. *british machine vision conference*, 2006. 1, 2

[5] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. *computer vision and pattern recognition*, 2019. 3

[6] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Counting out time: Class agnostic video repetition counting in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10387–10396, 2020. 1, 3, 7, 8

[7] Zhaoyang Liu et al. Tam: Temporal adaptive module for video recognition. In *ICCV*, 2021. 7

[8] Christoph Feichtenhofer. X3d: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 7

[9] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet. *computer vision and pattern recognition*, 2017. 1, 7

[10] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram Syed, Andrey Konin, M Zeeshan, Zia Quoc, and Huy Tran. Learning by aligning videos in time. 2022. 3

[11] Huazhang Hu, Sixun Dong, Yiqun Zhao, Dongze Lian, Zhengxin Li, and Shenghua Gao. Transrac: Encoding multi-scale temporal correlation with transformers for repetitive action counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19013–19022, 2022. 1, 2, 3, 6, 7, 8

[12] Yifei Huang, Yusuke Sugano, and Yoichi Sato. Improving action segmentation via graph-based temporal reasoning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14024–14034, 2020. 7

[13] Allan Jabri, Andrew Owens, and Alexei A. Efros. Space-time correspondence as a contrastive random walk. *neural information processing systems*, 2020. 3

[14] Giorgos Karvounas, Iason Oikonomidis, and Antonis A. Argyros. Reactnet: Temporal localization of repetitive activities in real-world videos. *arXiv: Computer Vision and Pattern Recognition*, 2019. 2

[15] Zihang Lai, Erika Lu, and Weidi Xie. Mast: A memory-augmented self-supervised tracker. *computer vision and pattern recognition*, 2020. 2, 3, 5

[16] Zihang Lai and Weidi Xie. Self-supervised learning for video correspondence flow. *arXiv: Computer Vision and Pattern Recognition*, 2019. 2, 3, 5

[17] Ofir Levy and Lior Wolf. Live repetition counting. *international conference on computer vision*, 2015. 3

[18] Jianing Li, Bowen Chen, Zhiyong Wang, and Honghai Liu. Full resolution repetition counting. *arXiv preprint arXiv:2305.13778*, 2023. 7

[19] Liulei Li, Tianfei Zhou, Wenguan Wang, Lu Yang, Jianwu Li, and Yi Yang. Locality-aware inter-and intra-video reconstruction for self-supervised correspondence learning. 2022. 2, 3, 5

[20] Rui Li and Dong Liu. Spatial-then-temporal self-supervised learning for video correspondence. 2022. 3

[21] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv: Computer Vision and Pattern Recognition*, 2021. 1, 7

[22] Nikolay Neshov, Agata Manolova, Krasimir Tonchev, and Ognian Boumbarov. Detection and analysis of periodic actions for context-aware human centric cyber physical system to enable adaptive occupational therapy. *intelligent data acquisition and advanced computing systems technology and applications*, 2019. 2

[23] Erik Pogalin, Arnold W. M. Smeulders, and Andrew Thean. Visual quasi-periodicity. *computer vision and pattern recognition*, 2008. 1, 2

[24] Senthil Purushwalkam, Tian Ye, Saurabh Gupta, and Abhinav Gupta. Aligning videos in space and time. *european conference on computer vision*, 2020. 3

[25] Tom F. H. Runia, Cees G. M. Snoek, and Arnold W. M. Smeulders. Real-world repetition estimation by div, grad and curl. *computer vision and pattern recognition*, 2018. 3

[26] Tom F. H. Runia, Cees G. M. Snoek, and Arnold W. M. Smeulders. Repetition estimation. *arXiv: Computer Vision and Pattern Recognition*, 2018. 2

[27] Bernard Sarel and Michal Irani. Separating transparent layers of repetitive dynamic behaviors. *international conference on computer vision*, 2005. 2

[28] Baifeng Shi, Judy Hoffman, Kate Saenko, Trevor Darrell, and Huijuan Xu. Auxiliary task reweighting for minimum-data learning. *Advances in Neural Information Processing Systems*, 33:7148–7160, 2020. 3

[29] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv: Computer Vision and Pattern Recognition*, 2012. 1, 7

[30] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 7

[31] Ashwin Thangali and Stan Sclaroff. Periodic motion detection and estimation via space-time sampling. *workshop on applications of computer vision*, 2005. 1, 2

[32] Christopher J. Tralie and Matthew Berger. Topological eulerian synthesis of slow motion periodic videos. *international conference on image processing*, 2018. 2

[33] Christopher J. Tralie and Jose A. Perea. (quasi)periodicity quantification in video data, using topology. 2022. 2

[34] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. Learning correspondence from the cycle-consistency of time. *computer vision and pattern recognition*, 2019. 3

[35] Haiping Wu and Xiaolong Wang. Contrastive learning of image representations with cross-video cycle-consistency. *international conference on computer vision*, 2021. 3

[36] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *computer vision and pattern recognition*, 2016. 7

[37] Jiarui Xu and Xiaolong Wang. Rethinking self-supervised correspondence learning: A video frame-level similarity perspective. *arXiv: Computer Vision and Pattern Recognition*, 2021. 3

[38] Ren Yafei, Bo Fan, Weiyao Lin, Xiaokang Yang, Hongxiang Li, Wei Li, and Donghua Liu. An efficient framework for analyzing periodical activities in sports videos. *international congress on image and signal processing*, 2011. 2

[39] Yuzhe Yang, Xin Liu, Jiang Wu, Silviu Borac, Dina Katabi, Ming-Zher Poh, and Daniel McDuff. Simper: Simple self-supervised learning of periodic targets. 2022. 2

[40] Jianqin Yin, Yanchun Wu, Huaping Liu, Yonghao Dang, Liu Zhiyi, and Jun Liu. Energy-based periodicity mining with deep features for action repetition counting in unconstrained videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020. 2

[41] Qingtian Yu, Haopeng Wang, Fedwa Laamarti, and Abdulmotaleb El Saddik. Deep learning-enabled multitask system for exercise recognition and counting. *Multimodal Technologies and Interaction*, 2021. 2

[42] Huaidong Zhang, Xuemiao Xu, Guoqiang Han, and Shengfeng He. Context-aware and scale-insensitive temporal repetition counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 670–678, 2020. 1, 3, 7, 8

[43] Yunhua Zhang, Ling Shao, and Cees GM Snoek. Repetitive activity counting by sight and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14070–14079, 2021. 1, 3, 7