TransRAC: Encoding Multi-scale Temporal Correlation with Transformers for Repetitive Action Counting

Huazhang Hu[1*], Sixun Dong[1*], Yiqun Zhao[1] , Dongze Lian[1,2] , Zhengxin Li[1†], Shenghua Gao[1,3,4†]
[1]ShanghaiTech University [2]National University of Singapore
[3]Shanghai Engineering Research Center of Intelligent Vision and Imaging
[4]Shanghai Engineering Research Center of Energy Efficient and Custom AI IC
{huhzh, dongsx, v-zhaoyq, liandz, lizhx, gaoshh}@shanghaitech.edu.cn

Abstract

Counting repetitive actions are widely seen in human activities such as physical exercise. Existing methods focus on performing repetitive action counting in short videos, which is tough for dealing with longer videos in more realistic scenarios. In the data-driven era, the degradation of such generalization capability is mainly attributed to the lack of long video datasets. To complement this margin, we introduce a new large-scale repetitive action counting dataset covering a wide variety of video lengths, along with more realistic situations where action interruption or action inconsistencies occur in the video. Besides, we also provide a fine-grained annotation of the action cycles instead of just counting annotation along with a numerical value. Such a dataset contains 1,451 videos with about 20,000 annotations, which is more challenging. For repetitive action counting towards more realistic scenarios, we further propose encoding multi-scale temporal correlation with transformers that can take into account both performance and efficiency. Furthermore, with the help of fine-grained annotation of action cycles, we propose a density map regression-based method to predict the action period, which yields better performance with sufficient interpretability. Our proposed method outperforms state-of-the-art methods on all datasets and also achieves better performance on the unseen dataset without fine-tuning. The dataset and code are available [1].

1. Introduction

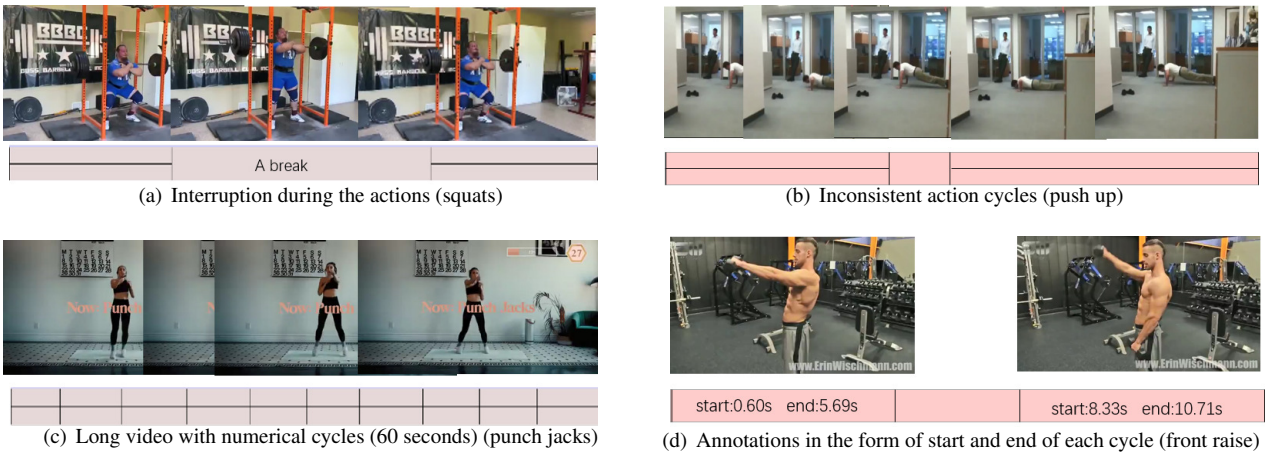
Planetary motion, the change of seasons, and heartbeats, these periodic movements that are everywhere in our lives. They can be modeled by the Newtonian mechanics, or detected with the aid of sensors for the understanding of the world or our bodies. In computer vision, the detection of repetitive/periodic motions also plays an important role, such as in human activity, where the counting of some physical exercise movements benefits people in fitness detection and planning. Although one can use some sensors (*e.g.*, gravity sensors) on the human body, vision-based approaches enable non-invasive and thus make third-view-based video analysis possible and promising. Repetitive action counting in computer vision is also useful as an auxiliary cue for other human-centric video analysis applications, such as pedestrian detection [26] and 3D reconstruction [17, 28].

Despite this importance, repetitive action counting methods in computer vision has rarely been explored. Previous papers [5, 38] tend to count repetitive actions in short videos, such as some simple videos grabbed from the Kinetics dataset [10]. However, these videos lack some realistic scenarios, which limits the application of the method in more realistic scenarios due to the following two points:

- **Restricted video length.** The previous datasets [5, 16, 29, 38] typically contain only short videos (*e.g.*, 0.4-30 s), however, methods are likely to be deployed to long videos in real scenarios. For instance, we count push-ups or jump-jacks with a video length of 60 s. Counting actions in such long videos is more challenging because there might exist various anomalies in real scenarios, such as the action being interrupted with internal or external reasons (Fig. 1(a)), or the inconsistency between action periods (Fig. 1(b)). These anomalies might cause the previous algorithm to fail or obtain sub-optimal performance, affecting the generalization of the algorithm to real scenarios.

- **Inadequate annotations.** In previous datasets [5, 16, 29, 38], the number of repetitive actions in a video

*These authors contributed equally to this work.
†Corresponding authors.
[1]https://svip-lab.github.io/dataset/RepCount_dataset.html



(a) Interruption during the actions (squats)

(b) Inconsistent action cycles (push up)

(c) Long video with numerical cycles (60 seconds) (punch jacks)

(d) Annotations in the form of start and end of each cycle (front raise)

Figure 1. The features of our proposed dataset **RepCount**: (a) anomaly case (interruption during actions); (b) anomaly case (inconsistent action cycles); (c) long video that consists of numerical action cycles; (d) the fine-grained labeling protocol.

is simply labeled as a numerical value. Although the count number serves as an ultimate predictive goal, such coarse-grained annotation deprives the interpretability of the algorithm. The model only predicts a numerical value during training or inference, which makes it difficult to evaluate the model more completely. As argued in some crowd counting papers [20, 27, 36, 40], the total number of repetitive actions is correct, but the position of the intermediate cycles might be wrong.

In data-driven deep learning approaches, the dataset is the key to algorithmic innovation. To tackle the above problems, we collect a large-scale human-centric dataset, which is closer to the real one. As shown in Fig. 1(c), there are a large number of variations in video length, while the interruptions or inconsistent action cycles occur in some videos. For more accurate performance evaluation and model interpretability, we provide a more fine-grained annotation of the action cycles, such as Fig. 1(d). Further, we also collect a part containing student activity videos captured in a fully realistic scenario (in local school), which is significantly different from the previous datasets where the videos are crawled from YouTube. Fig. 2 provides an overview of our dataset. Such a dataset is more challenging and has the potential to become a new benchmark for repetitive action counting.

To perform repetitive action counting, previous methods [5] generally take a fixed number of frames for prediction. Such an approach might be reasonable in relatively short videos. For example, TSN [37] extracts three frames for action recognition of trimmed videos, where the information characterizing the action is concentrated on certain keyframes. However, for long videos in the real scenario, extracting fixed frames will result in sub-optimal performance. Since the video duration varies very much (e.g. from 4s to 88s) and if the number of selected frames is too small, high-frequency actions will be neglected. On the contrary, if too many frames are selected, it might cause a waste of computational resources. Another alternative is to sample the video with the same frequency for both long and short videos. However, some actions are very fast (*e.g.*, jumping rope) and some are very slow (*e.g.*, push-ups). Sampling with a fixed frequency would either lead to performance degradation or would not be efficient enough. To balance performance and efficiency, we propose a multi-scale temporal correlation encoding network with transformers that can take care of not only high and low-frequency actions but also long and short videos. This approach allows the model to automatically select its adapted scale to compute the correlation matrix for final count prediction. Furthermore, thanks to the fine-grained annotation of action cycles in our dataset (see Fig. 1(d)), we also propose a density map regression-based method to predict action periods, which not only yields better performance but is also more beneficial for the interpretability of the model.

We summarize our contributions in three-fold:

- We introduce a new dataset, named RepCount, which consists of 1,451 videos and about 20,000 fine-grained annotations. Such a dataset allows for a large number of video length variations and contains anomaly cases, thus is more challenging.
- A new multi-scale temporal correlation encoding network with transformers, which can take care of not only high and low frequency actions, but also long and short videos, is designed for repetitive action counting.
- The proposed method outperforms state-of-the-art methods on our proposed dataset and all other datasets. Fur-

thermore, we also achieve better performance on the unseen dataset without fine-tuning.

2. Related Works

Temporal auto-correlation. The temporal auto-correlation function is widely used in motion recognition [3, 13, 14] and human identification [12]. Auto-correlation in time series contains periodic information [24, 35]. The most common method to represent auto-correlation is the vector inner product. Vaswani et al. [34] obtain the auto-correlation matrices by multiplying *query matrices* and *key matrices*. Panagiotakis et al. [24] implement attention mechanism for video frame series to construct the auto-correlation matrices of video frames. With auto-correlation matrices, the cycle information from time series can be found to count the number of repetitive actions.

Video feature extraction. For a long time, spatial and temporal convolution dominated the area of video feature extraction, such as C3D [33], I3D [2], P3D [25]. However, limited by the small receptive field of the convolution kernel, a convolution-based method is hard to capture the long-range dependencies on the temporal domain. ViT [4] and its variants bring a pioneering change for the Computer Vision field. However, Due to the quadratic computational complexity and complex structure, training of the transformer-based model is costly [11]. Video Swin Transformer is a useful backbone due to its trade-off which has pre-trained on large dateset. Video Swin Transformer [22] is a proper backbone due to its trade-off, which was pre-trained on a large dataset.

Counting in computer vision. Counting from images or videos [1, 16, 18, 19, 23, 29, 38, 39] is a very important field in Computer Vision. It has high application value in object detection [30], public transport [15] and physical exercises [31]. Zhang et al. [38] propose a context-aware and scale-insensitive framework for temporal repetition counting. [39] incorporate visual signal with corresponding sight signal to motion counting for the first time.

Density map. The application of a density map enhances the effect of crowd counting [20, 27, 36, 40]. The density map is generated from plot maps by convolving with a Gaussian kernel. The density map applies 2D planar spatial distribution to represent the spatial distribution and the local probability distribution. [40] apply 2D density maps to achieve dense crowd counting. In [32], the extracted features are passed to the density regressor to generate a density map. In many neural network architectures, it could be regarded as an intermediate representation layer. A periodic density map preserves more information and gives the spatial distribution.

Period annotation. Currently, data-driven learning methods have become an essential approach in computer vision. In the scenario of repetition counting, most datasets only label the period cycle count [5, 10, 38]. Researchers have to use generated data synthesized from real data and artificial data for training. [16] first proposed Synthetic data for the training model. However, such data are based on the assumption that the motion is continuous, uninterrupted, uniformly distributed, and with similar periods. [5] naively divided the count of periods by the number of frames to get the period length. This is far from the real motion situation. Therefore, a dataset with periodic fine-grained annotation is invaluable.

3. Our Proposed Dataset

Existing repetition counting datasets, mainly including Countix [5] and UCFRep [38], have been widely consumed for the evaluation of repetition counting models. In these datasets, video clips that are collected from YouTube cover a variety of perspectives, dimension sizes and action categories. Typically, the total number of repetitive actions in a video clip is labeled as its ground truth. While these datasets significantly contributed to modeling the repetition counting problem, there still exist several non-negligible limitations that increase the gap between the scenarios illustrated in the videos and realistic ones, such as i) no interruption to actions, either from internal or external; ii) only containing uniform action frequency in an individual video; iii) the lack of long-range videos; iv) coarse-grained ground truth annotation, *etc.* In particular, the last point hinders the development of more sophisticated models.

To overcome these limitations, we introduce a novel repetition counting dataset called **RepCount** that contains videos with significant variations in length and allows for multiple kinds of anomaly cases, as demonstrated in Fig. 2. These video data collaborate with fine-grained annotations that indicate the beginning and end of each action period. Furthermore, the dataset consists of two subsets namely Part-A and Part-B. The videos in Part-A are fetched from YouTube, while the others in Part-B record simulated physical examinations by junior school students and teachers. Therefore, flexible strategies of data splitting for training and evaluation could be adopted according to the specific demand. Then we introduce the data collection, annotation and statistics in detail.

Dataset collection. According to the source of data, our dataset consists of two parts. For part-A, we collected 1,041 video clips from YouTube. The type of actions includes workout activities (squatting, pulling-up, front-raising, *etc.*), athletic events (rowing, pommel horse, *etc.*) and other repetitive actions (soccer juggling). We select the video that represents at least one integral series of actions in line with human habits. Also, the videos usually contain some irrelevant actions like speaking and relaxation. For the most important, the interruption during an action series is preferable, which may result in difficulties for ac-

(a) Dataset Part-A (b) Dataset Part-B (c) Data analysis

Figure 2. **The summary of proposed benchmark RepCount**: The first two columns represent the part-A and part-B respectively, the right column shows the statistics of video length and repetition count of our dataset.

	UCFRep	Countix	Ours		
			part-A	part-B	part-A + part-B
Sources	Subset of UCF101	Subset of Kinetics	Youtube	Local school	Collected by ourselves
Num. of videos	526	8757	1041	410	1451
Duration Avg. ± Std.	8.15 ± 4.29	6.13 ± 3.08	30.67 ± 17.54	28.53 ± 16.06	**29.359± 16.02**
Duration Min./Max	2.08/33.84	0.2/10.0	4.0/88.0	5.56/79.16	**4.0/88.0**
Count Avg. ± Std.	6.66	6.84± 6.76	14.99 ± 14.70	9.27 ± 4.36	**15.932 ± 15.65**
Count Min./ Max	3/54	2/73	1/141	1/32	**1/141**

Table 1. Dataset statistic of **Countix** [5], **UCFRep** [38] and the proposed **RepCount**. Our dataset is lager than the previous datasets in terms of average duration and average annotations.

curate counting. We use the open-source script *YouTube-download*[2] to download the videos and edit them to keep only useful clips. The length of each video clip is 20-40 seconds in major. For part B, we record the videos of exercises such as sitting- and pulling-up done by volunteers.

Dataset annotation. The existing repetition counting datasets simplify the problem by assuming that the actions are periodically uniform and not interrupted by irrelevant situations. Thus they have only coarse-grained annotations in the form of a single-valued total [5] or two timestamps that indicating the start and end time of the whole action [38]. Our protocol for fine-grained annotation is as follows: i) each individual video is assigned to two volunteers; ii) start and end time of every action cycle are labeled; iii) the annotations are cross-validated by comparing that from two volunteers, which should be inspected and revised if they differ more than 1 in total. Following the protocol, each movement cycle is precisely positioned on the time axis, which enables the design and training for models with better interpretability.

Dataset statistics. The summary of our dataset is shown in Tab. 1. In brief, we provide 1,451 videos collaborated with 19,280 annotations. The videos from our dataset have an average length of 39.359 seconds, which is 4-5 times the length of videos from other datasets. Each video clip in our dataset contains 15.932 action cycles in average, while that is 6.66 for *UCFRep* and 6.84 for *Countix*. Furthermore, part-B is constructed for the validation of model generalization. The graphical statistics is shown in the right part of Fig. 2. Our dataset is featured with more realistic scenarios and fine-grained annotations.

[2] https://ytdl-org.github.io/youtube-dl/index.html

4. TransRAC Model

Given a long-duration video that has more than 15 repetitive activities happening in the content, our goal is to count the number of repetitive actions. To achieve this, we propose the model called TransRAC that contains three stages: the encoder, temporal correlation, and period predictor. The video subsequences V are fed into the encoder, then the output X is used to calculate the correlation matrix C by $C = \phi(X)$. At last, using period predictor $D = \tau(C)$ predict the final output density map.

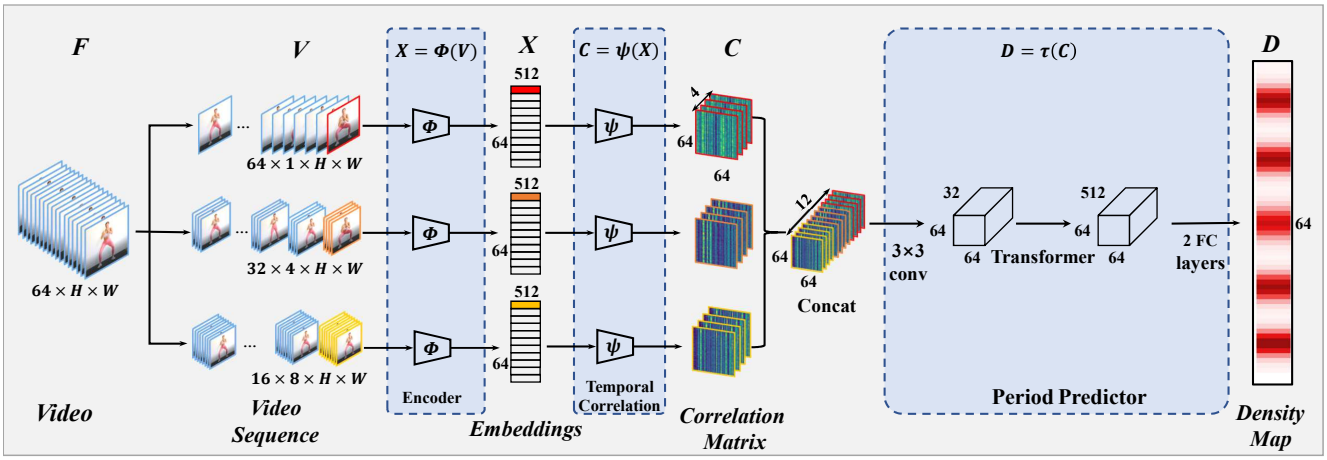


Figure 3. **TransRAC architecture.** We used three sliding windows with step sizes 1, 2, and 4 to generate the video sequence with an overlap: red, orange, and yellow. Then extract features from multi-scale video sequences by the encoder. Calculate the correlation matrix in three scales, respectively. After concatenating three correlation matrices into one, make it throughout the remaining network and output the final density map.

4.1. Encoder

As shown in Fig. 3, the encoder is $X = \phi(V)$. In order to explain the function ϕ, firstly, assuming that we have a sequence of N frames $F = [f_1, f_2, \ldots, f_N]$. we extract the three scale video subsequence V from the original as the input of ϕ. Then we feed the video sequence V into the encoder ϕ to produce multi-scale embeddings $X = [X_1, X_4, X_8]^T$, where $X_1 = [x_1^1, x_1^2, \ldots, x_1^N]^T$. X_4 and X_8 are similar to X_1.

Video sequences of multi-scale. We extract three scale subsequences from the video: single-frame, 4-frames, and 8 frames indicating the video subsequence of multi-scale (V) in Fig. 3. As Eq. (1), V_1, V_4, V_8 indicates the element length of the V. In the video sampling, we use a sliding window with a step size of 2 to obtain V_4 and a step size of 4 to get V_8. And we also need to pad the video to ensure the same temporal dimension of output three scale sequences.

$$\begin{cases} V_1 = \{\{f_1\}; \{f_2\}; \ldots\} \\ V_4 = \{\{f_1, \ldots, f_4\}; \{f_3, \ldots, f_6\}; \ldots\} \\ V_8 = \{\{f_1, \ldots, f_8\}; \{f_5, \ldots, f_{12}\}; \ldots\} \end{cases} \quad (1)$$

Spatio-temporal features. The video swin transformer [22] is used to extract 3d features from individual video subsequences of different scales. It can easily capture the long-range dependencies using the self-attention mechanism; at the same time, with the hierarchical design, it can also capture the local dependencies, which is more suitable for the image.

Let video subsequence V_i, where $i \in \{1, 4, 8\}$, pass through the feature extraction block to extract the features. Three kinds of video clips with different scales can match different period lengths (*e.g.* jump jacks and squat) better. The resulting features of each scale are of size $7 \times 7 \times t \times 768$, where the t is equal to a 2-fold compression in the temporal dimension. Then all these feature were concatenated in temporal dimension as a feature block .

Temporal context. To take into account more temporal context, we apply a layer of 3D convolution after the feature extractor, which has $3 \times 3 \times 512$ filters with ReLU activation. After that, we use a Global 3D Max-pooling layer over the spatial dimensions to reduce model parameters and obtain the final result X of encoder ϕ as the embeddings in Fig. 3. The above operations are carried on different scale video subsequences to the extent that we can obtain more information on the time domain.

4.2. Temporal correlation and Self-attention

The correlation of embeddings can be expressed as $C_i = \Psi(X_i)$. We need to compute every correlation c_p^i between x_p^i embeddings with other x_p^j, where $j \in \{1, 2, \ldots, N\}$ and $j \neq i$, such that we can use the embedding X_p to obtain the correlation matrix $C_p = [c_p^1, c_p^2, \ldots, c_p^H]^T$, $p \in \{1, 4, 8\}$ and H is the number of attention-head.

Correlation matrix. For the temporal locations of the activities, we use transformers [34] with correlation-matrix and self-attention mechanism to encode multi-scale temporal correlation layers. After encoder the video sequences, we can get embeddings X_i for each scale V_i, where $i \in 1, 4, 8$. And the shape of all embeddings for every scale is 64×512 as Fig. 3. Then We use the self-attention mechanism to calculate the correlation matrix. One scale embeddings X_i is multiplied with two matrices of weights for obtaining keys matrix called K and query matrix called Q. Then we could use K and Q to calculate attention scores,

which should be called correlation in this paper.

Self-attention. We construct the correlation matrix C by $C = f(Q, K)$, where $f(.)$ is known as dot product attention. And as Fig. 3 showed, these are two more important points that We use 4-heads with 512 dimensions (not eight heads which is more usual [34]) and multi-scale embeddings to calculate the correlation. Therefore, after the self-attention layer, concatenating three scales' features into one, we could get the shape of output is $[N, N, M \times H]$. M means how many scales we have. H and N are the numbers of heads and input frames independently. Furthermore in details, N, M and H in TransRAC are 64, 3 and 4, respectively.

4.3. Period Predictor

In Fig. 3, $D = \tau(C)$ shows feed C, where C is concatenated from C_1, C_2 and C_3, to the density map predictor which outputs none element for each video subsequence : the value of density the $D = [d_1, d_2, \ldots, d_N]$ represents the distribution of period. A more detailed version can be seen in Fig. 3.

Density map. The most straightforward advantage of the approach of density map is that it has a strong ability for explanation.. Therefore, We use the density map predicto as our period predictor. The density map contains the global information of the entire video. Each row of the density map indicates the frame's position in the local cycle and the distribution of the frame in the global video. We also compared density map regressor with classifiers in ablation experiments and found density maps performed. And more details for comparisons between the two predictors can be seen in Sec. 5. For more implementation details can be seen in the supplementary material.

4.4. Losses

Our dataset, *RepCount*, is annotated with each position of each motion period in the temporal dimension. Pass those labels through the Gaussian function $G(x)$ to get the ground truth. The process of Gaussianization can be seen in the supplementary material. Therefore, using MSE (Mean Squared Error) as the loss function tends to be a good choice.

4.5. Inference

To compare the network's performance purely from an academic view, we have not taken any measures to improve the prediction accuracy which is used in previous work [5]. The way to infer the counting of repetitions has the following operations:

Sample video. For a video with any length of fewer than two minutes, we directly sample 64 frames. If the input video has less than 64 frames, we will implement padding in the temporal domain.

Calculating. These frames are input into the model to obtain the prediction results of density map $D = [d_1, d_2, \ldots, d_N]$. Applying a linear sum to obtain the predicted value $\hat{p}$ of the number of action periods, where d_i means the value of density map.

5. Experiments

There are five central parts in this section. First of all, we explain some existing benchmarks and the evaluation matrices used in popular repetition counting. Secondly, We illustrate the advantages and capabilities of fine-grained annotations in detail. By visualizing and comparing the predictions for different sports, we propose conjectures and solutions. Then we evaluate our model performance and compare it to other methods, which were trained on our dataset *RepCount*, on the existing benchmarks. At last, we make an ablation study to justify our model design.

5.1. Benchmarks and Evaluation Matrices

We evaluate our method on the four video datasets: our test set of *RepCount part-A*, ours *RepCount part-B* and *UCF Rep* [38]. As illustrated in Tab. 1, *Ours (part-A+part-B)* contains videos with more count and longer duration than all existing datasets. The previous work [5, 38] mainly uses two matrices for evaluating repetition counting in videos:

Off-By-One (OBO) count error. If the predicted count is within one count of the ground truth, we can consider this video are counted correctly. Otherwise, it is a situation of counting error. It represents the error rate of repetition count over the entire dataset.

Mean Absolute Error. This metric means normalized absolute error between the ground truth count and the predicted count. OBO and MAE are defined as follows:

$$\text{OBO} = \frac{1}{N}\sum_{i=1}^{N}[|\widetilde{c_i} - c_i| \leq 1], \tag{2}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\widetilde{c_i} - c_i|}{\widetilde{c_i}}, \tag{3}$$

where $\widetilde{c}$ is the ground truth repetition counts. N is the number of given videos.

5.2. Implementation Details

We implement our method with PyTorch. The encoder, Video Swin Transformer tiny [22], was pre-trained on the *Kinetics*. We use three columns to obtain input video sequences and feed them into the encoder. The hidden layer dimension of transformer-based period predictor is 512. Limited by the memory of GPU, the parameters of the pre-trained encoder were frozen during the training process. We train our remaining layers of the model for 16K steps with a

declining learning rate of 8×10^{-6} and optimized by the Adam optimizer using a batch size of 16 (each with 64 frames). Additional details are provided on the code.

5.3. Fine-grained Annotation

Observing from the Fig. 4, it is easy to find accurate periodic location information on the ground truth, which is essential for accurately counting. As each kind of action has different characteristics, some activities, such as bench press, will be completed speed rate highly, due to the excellent energy of people at the beginning. However, at the end of the action, the rate will slow down. On the other hand, the period length of specific actions can be more uniform. As the examples of Fig. 4 shown, "front-raise" is easier for the man to finish in a statable period. It is because we annotated the data in a more fine-grained way, by which we can get the locations of all kinds of actions from our dataset, we have the opportunity to fine-tune the structure of the model for different needs. Of course, there is no chance to set the density map as a predictor of our model without fine-grained annotations. Overall, a more fine-grained annotation is necessary to precisely help the model count the number of periods.

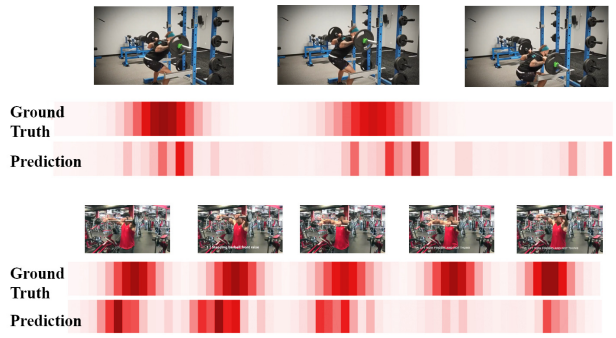


Figure 4. **Visualization of density map.** Here are comparisons between the ground truth and prediction result from our model. We can see from the first pair that the duration of videos in our dataset varies.

5.4. Evaluation and Comparison

We evaluate the effectiveness of the model from multiple aspects. When we compare the TransRAC proposed with RepNet on *RepCount (Part-A and Part-B)* and *UCFRep* datasets, for a fair comparison, we modify the last fully-connection layer of RepNet [5] to make it capable of handling those videos containing more than 32 action periods. Unless otherwise specified, we train the networks on *RepCount Part-A* and validate them on the test set of *Part-A*, obtaining the results shown in Tab. 2. In addition, we compare some SOTA action recognition methods [6, 7, 22] and change output layers accordingly to adapt to our task. Further more, we also compare the SOTA method [9] in the action segmentation field. More detail can be seen in the supplementary materials. One can observe that TransRAC, our model outperforms them by a notable margin on all the considered datasets.

Generalization. From Tab. 3, it also can be seen that the TransRAC model generalizes well on multiple datasets.

	RepCount A	
Method	MAE↓	OBO ↑
X3D [7]	0.9105	0.1059
TANet [6]	0.6624	0.0993
Video SwinT [22]	0.5756	0.1324
Huang et al. [9]	0.5267	0.1589
RepNet [5]	0.9950	0.0134
Zhang et al. [38]	0.8786	0.1554
Ours	**0.4431**	**0.2913**

Table 2. Performance of different method on *RepCount part-A test* when trained on the same train set of *RepCount*.

	RepCount B		UCFRep	
Method	MAE ↓	OBO ↑	MAE ↓	OBO ↑
RepNet [5]	0.9994	0.0025	0.9985	0.009
Ours	0.7839	0.091	0.6401	0.324

Table 3. Performance of different method on *RepCount part-B* and *UCFRep* when trained on the same train set of *RepCount part-A*.

Figure 5. **Visualization of bad cases.** Here are results of two bad cases our model predicted. In the first case, another people is moving

While our TransRAC performs well on the major part of the data, there still are some failure cases, as Fig. 5 shown. For the top one in Fig. 5, we achieve a bad predicted result because there is more than one person moving in the video. The failure case on the bottom indicates that the frame extracting strategy could diminish the performance of our model in some extreme situations. It can be seen that

there is an apparent difference between the predicted density map and the ground truth, especially in the left part. As this sample video has a total number of frames of 772, most of the actions concentrate in the first 400 frames, neither the ground truth density map nor the output of our model is capable of handling such imbalance.

5.5. Ablation Studies

We perform several ablations to justify the decisions made while designing TransRAC. We train our model on a train set of *part-A* and then evaluate the model on the test set of *part-A*. More ablation experiments can be seen in the supplementary material.

Correlation matrix. In Tab. 4, We compare the impact of applying different correlation matrix to our model. Temporal self-similarity matrix (TSM) [5] applies squared euclidean distance as the similarity function. But we found using the self-attention mechanism to calculate the correlation matrix is better. Since the experiment illustrates the self-attention mechanism could substantially improve the performance of our model, our model uses the self-attention mechanism.

	RepCount A	
Correlation matrix	MAE ↓	OBO ↑
TSM	0.5678	0.2251
Self-attention (Ours)	0.4431	0.2913

Table 4. Result of our model applying different correlation matrix when trained on training set of *RepCount part-A*.

Density map. We build four models to verify the effectiveness of the density map, as shown in the first four rows of Tab. 5. We could conclude that using density map regressor as the period predictor is significantly better than original classifiers. As shown in the third and the four rows of Tab. 5, if we replace the classifiers with density map regressor, RepNet's performance has been significantly improved. The comparison result indicates that the Density map is more suitable for repetitive action counting.

Multi-scale. In Tab. 5, we compare the impact of applying different scales. We find that the multi-scale model performs better than the single-scale model when the number of frames is equal. The experiment demonstrates that more temporal features at different scales can obtain more period information.

6. Conclusion

In this paper, considering the tough problems of existing methods in dealing with long videos in more realistic scenarios, we propose a new large-scale repetitive action counting dataset. Such a dataset covers a wide variety of video lengths where action interruption or action inconsistencies situations occur in the video, this is more realistic. For model interpretability and more accurate evaluation, we further provide fine-grained annotation. The overall dataset contains 1,451 videos with about 20,000 annotations, which is more challenging and has the potential to be a new benchmark. To balance the performance and efficiency, we propose to encode multi-scale temporal correlation with a transformer to tackle the repetitive action counting problems in realistic scenarios. We also propose a density map regression-based method to predict the action period, which yields better performance with sufficient interpretability. Extensive experiments show that our method achieves state-of-the-art results on all datasets and also achieves better performance on the unseen dataset without fine-tuning.

	RepCount A	
Method	MAE↓	OBO ↑
ResNet [8] + CLS	0.9950	0.0134
ResNet [8] + DM	0.6905	0.0811
SwinT [21] + CLS	0.7027	0.118
SwinT [21] + DM	0.6781	0.138
Ours (Scale-1)	0.6595	0.1854
Ours (Scale-4)	0.5434	0.2649
Ours (Scale-8)	0.6657	0.192
Ours (Multi)	**0.4431**	**0.2913**

Table 5. The result of abliation study when models trained on *RepCount part-A*. ResNet+CLS indicates the original structure of **RepNet** [5]. ResNet + DM indicates replacing the last layers with the density map regressor. The same applies to swinT which indicates swin-transformer. Ours (Scale-X) indicates the single column without multi-scale correlation, where X represent V_1, V_2, and V_3. Ours (Multi) indicates our proposed structure.

Broader Impact and Limitations. The proposed dataset is about the counting of repetitive actions, which means videos from our dataset are human-centric. The abuse of our dataset may cause privacy leaks. The usage of our dataset is limited to academic research. The proposed method predicts results based on the dataset, which may include some negative social impacts. Thus, the results conducted by our method may reflect the bias from the dataset. Other technical limitations are talked about in the Sec. 5.

Acknowledgements. The work was supported by National Key R&D Program of China (2018AAA0100704), NSFC #61932020, #62172279, Science and Technology Commission of Shanghai Municipality (Grant No. 20ZR1436000), and "Shuguang Program" supported by Shanghai Education Development Foundation and Shanghai Municipal Education Commission.

References

[1] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. Counting in the wild. In *European Conference on Computer Vision*, 2016. 3

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 3

[3] Chen Chen, Baochang Zhang, Zhenjie Hou, Junjun Jiang, Mengyuan Liu, and Yun Yang. Action recognition from depth sequences using weighted fusion of 2d and 3d auto-correlation of gradients features. *Multimedia Tools and Applications*, 76(3):4651–4669, 2017. 3

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3

[5] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Counting out time: Class agnostic video repetition counting in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3, 4, 6, 7, 8

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

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

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 8

[9] 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

[10] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1, 3

[11] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. 3

[12] Takumi Kobayashi and Nobuyuki Otsu. A three-way auto-correlation based approach to human identification by gait. In *IEEE Workshop on Visual Surveillance*, volume 1, page 4. Citeseer, 2006. 3

[13] Takumi Kobayashi and Nobuyuki Otsu. Three-way auto-correlation approach to motion recognition. *Pattern Recognition Letters*, 30(3):212–221, 2009. 3

[14] Takumi Kobayashi and Nobuyuki Otsu. Motion recognition using local auto-correlation of space–time gradients. *Pattern Recognition Letters*, 33(9):1188–1195, 2012. 3

[15] Paulius Lengvenis, Rimvydas Simutis, Vygandas Vaitkus, and Rytis Maskeliūnas. Application of computer vision systems for passenger counting in public transport. *Elektronika ir Elektrotechnika*, 19(3):69–72, 2013. 3

[16] Ofir Levy and Lior Wolf. Live repetition counting. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015. 1, 3

[17] Xiu Li, Hongdong Li, Hanbyul Joo, Yebin Liu, and Yaser Sheikh. Structure from recurrent motion: From rigidity to recurrency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[18] Dongze Lian, Xianing Chen, Jing Li, Weixin Luo, and Shenghua Gao. Locating and counting heads in crowds with a depth prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 3

[19] Dongze Lian, Jing Li, Jia Zheng, Weixin Luo, and Shenghua Gao. Density map regression guided detection network for rgb-d crowd counting and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1821–1830, 2019. 3

[20] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5099–5108, 2019. 2, 3

[21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021. 8

[22] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 3, 5, 6, 7

[23] Erika Lu, Weidi Xie, and Andrew Zisserman. Class-agnostic counting. In *Asian conference on computer vision*, pages 669–684. Springer, 2018. 3

[24] Costas Panagiotakis, Giorgos Karvounas, and Antonis Argyros. Unsupervised detection of periodic segments in videos. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 923–927. IEEE, 2018. 3

[25] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *proceedings of the IEEE International Conference on Computer Vision*, pages 5533–5541, 2017. 3

[26] Yang Ran, Isaac Weiss, Qinfen Zheng, and Larry S. Davis. Pedestrian detection via periodic motion analysis. *Int. J. Comput. Vis.*, 71(2):143–160, 2007. 1

[27] Viresh Ranjan, Hieu Le, and Minh Hoai. Iterative crowd counting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2, 3

[28] Evan Ribnick, Nikos Papanikolopoulos, Evan Ribnick, and Nikolaos Papanikolopoulos. 3d reconstruction of periodic motion from a single view. 1

[29] Tom F. H. Runia, Cees G. M. Snoek, and Arnold W. M. Smeulders. Real-world repetition estimation by div, grad and curl. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1, 3

[30] Nilakorn Seenouvong, Ukrit Watchareeruetai, Chaiwat Nuthong, Khamphong Khongsomboon, and Noboru Ohnishi. A computer vision based vehicle detection and counting system. In *2016 8th International Conference on*

Knowledge and Smart Technology (KST), pages 224–227. IEEE, 2016. 3

[31] Andrea Soro, Gino Brunner, Simon Tanner, and Roger Wattenhofer. Recognition and repetition counting for complex physical exercises with deep learning. *Sensors*, 19(3):714, 2019. 3

[32] Xin Tan, Chun Tao, Tongwei Ren, Jinhui Tang, and Gangshan Wu. Crowd counting via multi-layer regression. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1907–1915, 2019. 3

[33] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015. 3

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 3, 5, 6

[35] Michail Vlachos, Philip Yu, and Vittorio Castelli. On periodicity detection and structural periodic similarity. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 449–460. SIAM, 2005. 3

[36] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 3

[37] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Val Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016. 2

[38] Huaidong Zhang, Xuemiao Xu, Guoqiang Han, and Shengfeng He. Context-aware and scale-insensitive temporal repetition counting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 3, 4, 6, 7

[39] Yunhua Zhang, Ling Shao, and Cees G. M. Snoek. Repetitive activity counting by sight and sound. *CoRR*, abs/2103.13096, 2021. 3

[40] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 589–597, 2016. 2, 3