Detailed Explanation of the Experimental Settings in TransRAC

Huazhang Hu*, Sixun Dong*, Yiqun Zhao, Dongze Lian, Zhengxin Li, Shenghua Gao†
huhzh@shanghaitech.edu.cn, sixundong@asu.edu, goatzyq@gmail.com
lizhx@shanghaitech.edu.cn,dzlianx@gmail.com, gaosh@hku.hk

Abstract

To clarify the confusion raised by the authors of RepNet in the note [2] and address their concerns regarding RepNet's performance, we provide a detailed description of the experimental settings used in TransRAC [5] and discuss how to evaluate RepNet in the context of [5].

The primary difference in our evaluation is that we retrained RepNet from scratch using a pre-trained ResNet, rather than loading the released checkpoint, which was trained on a large synthetic dataset. Additionally, we did not employ the multi-speed evaluation for inference, as used in [1], which could potentially enhance the results. In our experiments, we trained and evaluated RepNet, other baselines, and TransRAC based on these settings. We believe we fairly evaluate the different neural network frameworks by training them on RepCount-A under our experimental settings.

1. Introduction

Referring to [2], the authors suggest that [5] may not be a fair comparison between RepNet and TransRAC. They argue that modifications to the last FNN layer could significantly reduce RepNet's performance. Furthermore, they believe the original RepNet is capable of handling actions with varying periods by sampling videos at different speeds. As a result, the authors propose that these issues in [5] might lead to RepNet's poor performance.

After thoroughly reviewing the note[2] by the authors of [1, 2], we summarize their concerns about how to fairly evaluate RepNet [1] into three main points, followed by our discussion:

- How is RepNet[1] trained and evaluated in [5]?
- Why is *Multi-speed Evaluation* not utilized?
- Why and how is the last full connection layer of RepNet[1] modified?
- Discussion: Is it a fair comparison? Why low performance?

2. How to train and evaluate

Firstly, in [1], the authors train RepNet[1] on **Synthetic Repetition Videos** and Countix dataset. Tab. 1 shows that training the model on Synthetic dataset leads to a good performance compared to training the model on Countix dataset alone. Moreover, as Tab. 2 shows, when using real-world datasets for training, Countix and RepCount-A differ significantly in scale, containing 4588 and 758 videos, respectively. The result indicates the importance of the dataset used for training. Additionally, we could not find any statistic information regarding Synthetic dataset in [1, 2].

TSM	Training Data Source	QUVA		Countix (Val)	
		MAE ↓	OBO ↓	MAE ↓	OBO ↓
	Synthetic	1.2853	0.64	1.1671	0.5510
✓	Synthetic	0.1035	0.17	0.3100	0.2903
	Countix	0.7584	0.72	0.6483	0.5448
✓	Countix	0.3225	0.34	0.3468	0.2949
	Synthetic + Countix	0.6388	0.57	0.8889	0.4848
✓	Synthetic + Countix	0.1315	0.15	0.3280	0.2752

Table 1. Ablation of RepNet's architecture with or without the temporal self-similarity matrix (TSM) with different training data sources. **This table is adapted from Table 2 of [1].**

However, the experiments of [5] try to analyze different methods' performance based on the same training dataset. Hence, we trained every method based on Tab. 3. At the same time, all methods are trained from scratch but loaded with pre-trained backbones, such as ResNet[4], Swin Transformer[9], and Video Swin Transformer[10], which are trained on ImageNet or Kinetics.

When evaluating, we validate all methods' performance in the same setting. Hence, we evaluate ResNet[4] without *Multi-speed Evaluation* technique. Refer to the next section to explain why we do not use it.

3. Why not utilize *Multi-speed Evaluation*

Although we know *multi-speed evaluation* could help the model handle different period actions from [1]. We didn't find an ablation study for *multi-speed evaluation* in [1]. Other baselines might also benefit from *multi-speed evaluation*. Following the insight from [1, 7], we believe

Sources	UCFRep (Test)	Countix (Train/Val/Test)	RepCount-A (Train/Val/Test)
Num. of videos	526	4588 / 1450 / 2719	758 / 131 / 152
Duration Avg. ± Std.	8.15 ± 4.29	6.13 ± 3.08	30.67 ± 17.54
Duration Min./Max	2.08/33.84	0.2/10.0	4.0/88.0
Count Avg. ± Std.	6.66 ± 4.29	6.84 ± 6.76	14.99 ± 14.70
Count Min./Max	3/54	2/73	1/141

Table 2. Dataset statistics of UCFRep, Countix, and RepCount Part-A.

Paper	Training Datasets	Evaluation Datasets
[5]	RepCount-A	RepCount-B, UCFRep
[1, 2]	Synthetic, Countix	Countix, UCFRep, RepCount-B, and Others

Table 3. Datasets Setting in [5]

that if there are two videos where two people do the same action with different periods/speeds, we could adjust these actions to a similar period by speeding up or speeding down the original videos. This operation could simplify the data. However, our study[5] does not focus on how the video speed influences the different baselines' performance.

Unlike [1], we only input the original video and do not adjust the speed to fairly evaluate the different frameworks in [5]. All experiments are based on this setting. As stated in Sec.4.5 Inference of [5]: *"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]."*, where [5] refers to RepNet[1].

Overall, although we do not follow the approach in [1] of utilizing *Multi-speed Evaluation* to aid RepNet[1] in inference, we believe that our experimental settings fairly compares the performance of different models, including RepNet[1], under the same settings.

4. Why and how to modify the last FFN layer?

Referring to Sec.3.5 Inference, in [4], the authors state: *"As our model can predict period lengths up to 32, for covering much longer period lengths, we sample input video with different frame rates."*, They address the prediction limitation in their method by *Multi-speed Evaluation*. However, as mentioned in Sec. 3, we do not apply this technique to RepNet[1]. Instead, we adjust ResNet's last FFN layers to 64 to improve its network prediction ability, similar to TransRAC, which uses FFN layers of the same size. To ensure a fair performance comparison between different methods, we believe that leaving RepNet's FFN layer dimensions at 32 instead of increasing them to 64 would not be a fair evaluation. The experimental results are shown in Tab. 4, adapted from Tab. 5 of [5].

ResNet[4] + CLS means RepNet[1], but the dimension of the last layer is modified to 64. When training the CLS-based method, we follow [1] to predict *Per Frame Period Length* and *Per Frame Periodicity Prediction* with classification loss. We also utilize frame-level annotation to generate ground truth for these predictors.

Method	RepCount A	
	MAE↓	OBOA ↑
ResNet [4] + CLS	0.9950	0.0134
ResNet [4] + DM	0.6905	0.0811
SwinT [9] + CLS	0.7027	0.118
SwinT [9] + DM	0.6781	0.138

Table 4. The result of abliation study when models trained on *RepCount part-A*. ResNet+CLS indicates the original structure of **RepNet**[1]. ResNet + DM indicates replacing the last layers with the density map regressor. The same applies to swinT, which indicates swin-transformer. **This table is adapted from Tab. 5 of [5].**

ResNet[4] + DM means changing the predictor from *Period Length Predictor* and *Frame Periodicity Predictor* to predict density map.

The results show that SwinT[9] + CLS performs better than ResNet[4] +CLS. The only difference between these two methods lies in their backbones: one uses a pre-trained ResNet on ImageNet [4], while the other uses a pre-trained Swin Transformer on ImageNet [9].

Hence, we believe that our reimplementation[1] of RepNet[1], trained and tested using the PyTorch framework, could achieve better performance by predicting density maps and replacing the RepNet backbone from ResNet[4] to Swin Transformer[9], instead of following [1] and using a classification loss for training.

[1] After contacting the authors of RepNet [1], they suggested rechecking whether our reimplementation aligns exactly with their open-sourced TensorFlow version. We have extensively verified that our model is aligned well with theirs. However, train scripts of RepNet[1] are not open source, we only rely on our train scripts, which are applied to all experiments Tab. 4. If we find any discrepancies in the future, we will update the experimental results accordingly. Furthermore, as shown in Tab. 4, our modifications to the model structure illustrate the effects of different backbones and period prediction methods on overall performance. We believe that the experiments presented in the original paper [5] remain valid, and we have followed their reported results in this note.

5. Discussion

5.1. Is it a fair comparison?

The authors of [1, 2] presented their results in Tables 1-3 of [1, 2], which show that RepNet achieves better performance. However, these methods may have been trained on different datasets, particularly as shown in Tab. 1, where *Synthetic* data has been shown to improve performance significantly. Furthermore, [1] does not release the training and synthetic video scripts. Therefore, we aim to evaluate different methods under the same conditions.

As described in Secs. 2 to 4, we reimplement and modify RepNet [1] to adapt it to our experimental settings. The baseline method from Huang et al. [6] faces a similar issue, as it cannot handle repetitive actions in a single video. To address this, we reconstruct the annotation of RepCount-A and train Huang et al. [6] in our experimental settings.

Although, as noted in [1, 2], we do not use the *Multi-Speed Evaluation* technique to help RepNet perform better, we believe our experimental design provides a fair comparison of the performance of different neural network frameworks, based on the same training, test datasets, and settings.

5.2. RepNet's performance

Referring to [1, 2] and Tabs. 1 and 2 and Tab. 5, the authors proposed that *Training with Synthetic Repetitions* is a valid and efficient system that generates more data for the model to learn. The proposed *Multi-speed Evaluation* could generalize RepNet[1], trained on Synthetic and Countix, to RepCount-A and RepCount-B.

Model	Year	MAE ↓	OBOA ↑
Mod. RepNet in[5]	2022	0.9950	0.0134
TransRAC [5]	2022	0.4431	0.2913
MFL* [8]	2024	0.384	0.386
RACNet* [8]	2024	0.4441	0.3933
ESCounts [12]	2024	**0.245**	**0.563**
RepNet[1]	2020	0.3308	0.5329

Table 5. Counting Results on the **RepCount-A** dataset (test split). MAE is reported with $\alpha = 0.1$. * denotes that [2] were not able to verify the value of α. **This table is adapted from Tab. 5 of [2].**

However, according to the trainsets data scale and experimental results shown in Tabs. 2 and 4, RepNet [1] may not perform optimally under our experimental settings.

(1) Firstly, compared with training RepNet on the Countix dataset [1], we hypothesize that the RepCount-A training set, which contains 758 videos, may be insufficient to fully enable RepNet's capabilities.

(2) Secondly, as shown in Tab. 2, if we consider modifying RepNet's *Period Length Prediction Layer* from a 32-classifier head to a 64-classifier head, and assuming the action durations are uniform, the available data for each class can be calculated as follows:

$$\text{RepCount-A: } \frac{758}{74} = 11.8, \quad \text{Countix: } \frac{4588}{32} = 143.4.$$

These results indicate that when training models on RepCount-A, the classifier-based model may not have enough training samples to support effective learning. In addition, if the durations of the action are not uniform, some classes may lack sufficient data for learning.

(3) Thirdly, we need to sample a video to 64 frames in our experiment settings. The original annotations may be very densely compressed. For example, a 30-second video (30 FPS) and its annotations (start frame, end frame) could be compressed:

$$\begin{aligned} &\text{Original Annotation: } [0, 10, 10, 20, 20, 30, 90, 899] \\ &\longrightarrow \text{Compressed Annotation: } [0, 1, 1, 1, 1, 2, 63, 63] \end{aligned}$$

For density map prediction, we could generate a density map as $[1, 2, 1, \ldots, 1]$, which would contain 4 actions. However, period prediction based on classifiers such as [1] cannot handle this situation properly. In our reimplementation, we had to ignore actions that occurred more than once in the same frame.

For example,

$$\begin{aligned} \text{Compressed Annotation:} &\quad [0, 1, 1, 1, 1, 2] \\ \longrightarrow \text{Adjusted Annotation:} &\quad [0, 1, 1, 2] \end{aligned}$$

We need to adjust this compressed annotation to $[1, 2]$ to generate the ground truth for the classifier layers. Thus, the final ground truths for training classifiers will be:

$$\begin{aligned} \text{Period Prediction} &= [1, 1, 0, \ldots, 0, 1] \\ \text{Periodicity Prediction} &= [1, 1, 0, \ldots, 0, 1] \end{aligned}$$

which would only contain 3 actions. As a result, many mistakes occurred when generating ground truth for classification in these situations.

In general, as shown in Tab. 4, and supported by recent works works[3, 8, 11, 12] show, we believe utilizing a density map for period prediction may be a better choice for the repetitive action counting task. Notably, [3], authored by the authors of RepNet[1], also adopts this approach in their latest 2024 work.

6. Conclusion

We detail our experiment settings and analyze and explain why RepNet failed to work in [5]. In [1], the authors propose two valid approaches to enhance RepNet[1] performance. In contrast, [5] conducts experiments to analyze the performance of different neural network frameworks under

the same setting without focusing on using additional synthetic data or designing algorithms to improve inference. In the setting of [5], as shown in Tab. 4, RepNet[1] unfortunately doesn't perform well, but the follow-up works[5, 8, 11, 12] work well.

Consequently, Repetitive Action Counting (RAC) still faces many unexplored challenges, such as model generalization, handling long videos, and training proposed frameworks (e.g., [5], [8], [11], [12]) using synthetic videos like those in [1]. Additionally, dealing with very short and repetitive action periods remains challenging. We hope this paper will address the concerns raised by the authors of [1, 2] and contribute to advancing research in this field.

References

[1] 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

[2] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Short note on evaluating repnet for temporal repetition counting in videos. In *ArXiv*, 2024. 1, 2, 3, 4

[3] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, and Andrew Zisserman. Ovr: A dataset for open vocabulary temporal repetition counting in videos. *arXiv preprint arXiv:2407.17085*, 2024. 3

[4] 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. 1, 2

[5] 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, 4

[6] 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. 3

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

[8] Xinjie Li and Huijuan Xu. Repetitive action counting with motion feature learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 6499–6508, 2024. 3, 4

[9] 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. 1, 2

[10] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021. 1

[11] Yanan Luo, Jinhui Yi, Yazan Abu Farha, Moritz Wolter, and Juergen Gall. Rethinking temporal self-similarity for repetitive action counting. In *2024 IEEE International Conference on Image Processing (ICIP)*, pages 2187–2193. IEEE, 2024. 3, 4

[12] Saptarshi Sinha, Alexandros Stergiou, and Dima Damen. Every shot counts: Using exemplars for repetition counting in videos. *arXiv preprint arXiv:2403.18074*, 2024. 3, 4