Multistream architecture for repetetive action counting

Sanchit Tanwar
Maryland Applied Graduate Engineering
University of Maryland
College Park
stanwar@umd.edu

Abstract

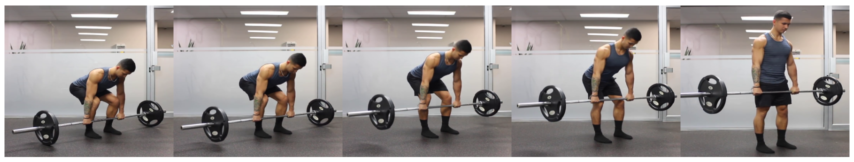
We observe repetition in many aspects of our daily lives, from human activities like exercise to tasks such as assembling an object. Accurate counting of repetitions is important for tracking progress and ensuring proper form in fitness and physical therapy. In this work, we developed a multi-stream model using RGB videos and human pose for the frames to count repetitions. We demonstrate the effectiveness of the multiple streams through experiments and compare our model to existing benchmarks. Our model consistently achieves state-of-the-art results while being computationally efficient. We also added a classification branch that identifies the exercise being performed in videos. This additional branch adds a single fully-connected layer and increase in computation is minimal.

1 Introduction

Repetitive actions are a common occurrence in various domains, including exercise, seasonal changes, and manufacturing processes. To facilitate the identification and quantification of these actions, we propose a system that utilizes computer vision and robust pose estimation algorithms to analyze video footage and automatically count the number of times a specific action is performed. This system has the potential to be utilized in various applications, such as quality control in manufacturing lines and performance analysis in sports. In the context of sports, this system can be used to track an individual's performance over time and inform workout decisions in order to improve fitness levels. Additionally, by incorporating posture analysis capabilities, the system can assist individuals in improving the effectiveness of their exercises by ensuring proper form is maintained. One potential avenue for future work is the integration of additional sensors, such as gyroscopes and accelerometers, which are commonly available on smartwatches, to enhance the accuracy and robustness of the system. The "mmfit" dataset [SHR20] provides a useful resource for exercise recognition using multiple sensor streams, including accelerometers, gyroscopes, and heart rate sensors.

In this study, we focus on utilizing RGB videos to automatically count repetitions in exercises. Our approach builds upon the work presented in [Hu+22] and uses the accompanying dataset for evaluation. The task of counting repetitions in exercises is particularly challenging due to the vast range of actions that can be performed and the similarity between these actions. For instance, the beginning stages of weight lifting and deadlift exercises may be visually similar, as shown in Figure 1, which can lead to confusion for algorithms attempting to count repetitions. Additionally, it is common for exercises to be interrupted by breaks, and an effective repetition counting algorithm must be able to distinguish between exercise actions and random actions that may occur during these breaks. Our proposed solution aims to address these challenges and provide a reliable method for counting repetitions in untrimmed, end-to-end videos.

In this study, we propose a multistream approach for accurately counting repetitions in videos, which outperforms the method presented in [Hu+22] while also significantly reducing computational demands. Our approach includes a classification head that can predict the class of the repeating


(a) This figure shows the motion performed during deadlifting.

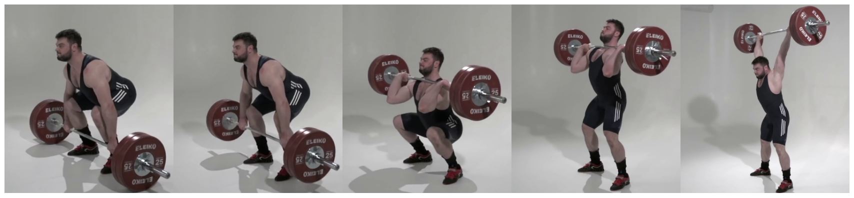

(b) This figure shows the motion performed during weightlifting

Figure 1: The figure a and b show deadlifting and weightlifting motion respectively. As shown in the first few frames, they have the same starting motion.

action, which is particularly useful for exercise counting applications where tracking metrics such as calories burned may be of interest. The inclusion of a classification branch in our network not only offers computational efficiency, but may also have the potential to improve the performance of repetition counting. Further investigation and access to a larger dataset with a greater number of unique exercise classes would be necessary to fully explore this possibility. We also demonstrate that, while a multi-scale approach has benefits, it incurs a high computational cost that may be infeasible for mobile device applications. As an alternative, we utilize a single scale and apply augmentation techniques to address this issue. Our approach utilizes two input streams: one from RGB video and one from human poses. The pose stream is particularly effective for actions such as exercises and can often provide sufficient information for repetition counting. Our pose stream is based on the work presented in [Dua+22] and uses the "Movinets" model [Kon+21] with a reduced number of blocks for feature extraction. The input to this stream is a heatmap generated by a pose estimation model.

2 Related Work

The repetition of actions is a common occurrence in a wide range of tasks, and as such, numerous approaches have been developed to address the problem of counting these repetitions. Temporal auto-correlation methods such as those presented in [Hu+22; Che+17; KO12] have been used in motion recognition to extract repetitive actions from the auto-correlation in time series, which is often represented by the vector inner product due to the presence of periodic information. Temporal segmentation methods, such as those presented in [WS13; GM15], have also been utilized to count the number of repetitions in actions. These methods employ temporal characteristics such as duration and frequency to divide the video into different segments. Some solutions recover the structure of repetition from 1-dimensional signals obtained by converting the field of motion [ABQ08; Lap+05; PKA18], and many of these approaches utilize Fourier Analysis to calculate the frequency of repetition [BA07; CD00].

[Dwi+20] proposes a deep learning model for counting repetitive actions using 2D CNNs for feature extraction in videos, while [Hu+22] utilizes the model [Liu+22] for feature extraction and generates a temporal correlation matrix similar to [Dwi+20] for period prediction. The output of the model is a density map that can be summed to obtain the number of repetitions in the video and can also be used to localize the repeating actions, generated through convolution of the plot maps with a Gaussian kernel, a technique commonly used in crowd counting applications [LSF18; RLH18; WC19]. If temporal locations for the start and end of the activities are provided, temporal action segmentation or localization techniques can be employed [Ish+21; Qin+21; Wan+20]. However, as noted in [Hu+22], using such methods may present challenges in handling high frequency repetitive actions that are segmented into single temporal bounds.

Major part of our model is pose stream and we use skeleton action recognition models for feature extraction from the human poses in several frames. Multiple approaches have been explored for skeleton action recognition. Some works convert the coordinates of a skeleton sequence into a pseudo image with transformations[[LPT18], [Ke+17], [Cae+19]], resulting in a 2D input that is not well-suited to the locality nature of convolution networks and leads to less competitive performance compared to graph convolutional networks (GCNs) [Che+21; YXL18]. We use [Dua+22] which uses a simple approach of using heatmaps generated by top down pose estimation models [Cao+18; Sun+19; NYD16] and pass it through a 3D CNN based network, this approach is simple and can take advantages from all the development in 3D CNN architectures for action recognition.

3 Methods

In this study, we improve upon the approach presented in [Hu+22], which uses a multi-scale feature extraction method to improve performance on long and short videos with high and low action frequencies. However, this approach is computationally demanding, requiring the extraction of features from the same video at different scales, resulting in a processing time of 1.11 seconds for a batch of 64 frames. To address this issue, we propose replacing the computationally intensive feature extraction backbone of [Liu+21] with a more efficient alternative, specifically MoviNets model [Kon+21], which offers improved computational efficiency compared to [Liu+21] and other action recognition models such as [CZ17], [Fei+19], and [Fei20]. Our final model is explained in more detail in 3.2

3.1 Dataset:

In this study, we utilized the RepCount A dataset proposed in [Hu+22] for training our proposed models. This dataset consists of 1041 videos belonging to 10 classes, including workout activities such as squatting, pulling up, and jumping jacks, as well as clips from sports including rowing, pommel horse, and soccer juggling. The dataset statistics are shown in Table 1. The videos in this dataset have a wide range of lengths, from 4 seconds to 88 seconds.

Number of videos	1041
Duration Avg. ± Std.	30.67 ± 17.54
Duration Min./Max	4.0/88.0
Count Avg. ± Std	14.99 ± 14.70
Count Min./ Max	1/141

Table 1: Dataset statistics of RepCount part A

3.2 Model:

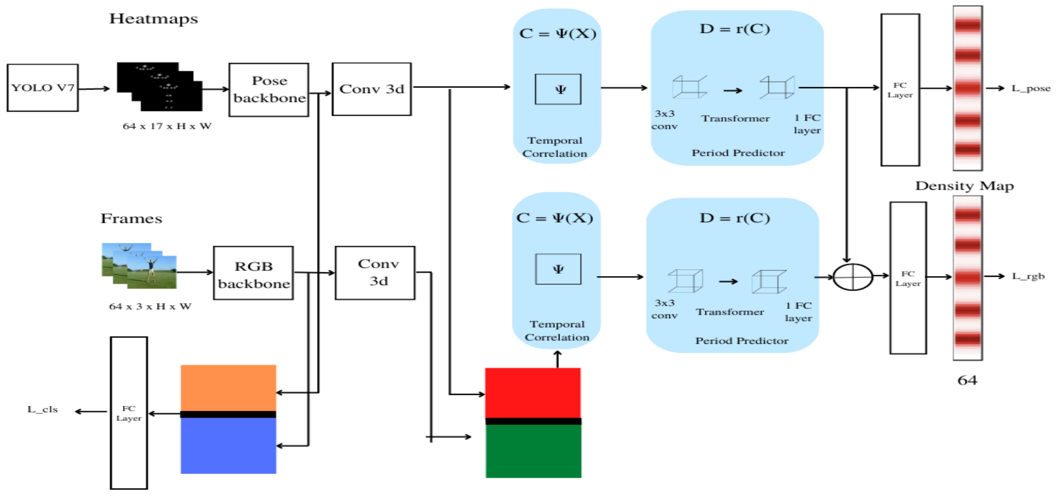


Figure 2: Architecture of the final model showing the feature fusion strategies and multiple streams used.

3.2.1 RGB Stream

- **Backbone:** Our backbones processes the video frames and extracts features that capture the temporal correlations between them. The input to the encoder model are multiple subsequences extracted from $F = [f_1, f_2, ..., f_N]$ by keeping temporal length equal to 8, resulting in $N/8$ segments. The output is a feature map from the last layer of the backbone, the size of this feature map is $N/8 \times 8 \times h_f \times w_f$, where h_f and w_f are the height and width of feature map and 8 depicts temporal length and $N/8$ number of segments. We then retrieve $N \times h_f \times w_f$ dimensional feature map and use a 3D convolution along with 3D Adaptive Max-pooling to extract more temporal context giving us feature embeddings $X = [X^1, X^2, ..., X^N]^T$. We use MoviNets [Kon+21] as our backbone.
- **Temporal correlation and Self-attention:** In this part of the model the correlation c^i of x^i is computed with every other x^j where $j \in [1, 2, ..., N]$ and $j \neq i$. Then, the correlation $C_i = \psi(X_i)$ of all embeddings X is computed by $C = [c^1, c^2, ..c^H]^T$ where H is attention-head. Self-attention mechanism is used to compute the correlation matrix, attention scores are calculated using Key K and Query Q matrices. After this layer, the output shape becomes $[N \times H]$, where N and H are number of frames in the input and number of heads respectively.
- **Period Predictor:** Density map is used to indicate the period. The output of this part is $D = [d_1, d_2, d_3, ..., d_n]$ which represents the distribution of period, this is described in detail in 3.3.

3.2.2 Pose stream

Our pose stream architecture is inspired by [Dua+22]. The poses are extracted using [WBL22] and saved offline. These poses are then transformed into a 3D heatmap volume using same technique suggested in [Dua+22]. Specifically, the 2D pose is represented as a heatmap with dimensions $K \times H \times W$, where total joints is represented by K, H and W respectively indicate height and width the generated heatmap. The Top-Down pose estimator produces a heatmap that can be used directly like a target, but we need to pad it in order to match the size of the original frame within the corresponding bounding box. Alternatively, if only the coordinates (x_k, y_k, c_k) of the skeleton joints are available, a joint heatmap J can be created by combining K Gaussian maps whose centers are at each of the joints. These heatmaps can be visualized in figure: 3, for visualization purpose we sum the heatmap along the K dimension.

$$J_{kij} = \exp - \frac{(i - x_k)^2 + (j - y_k)^2}{2 * \sigma^2} * c_k, \tag{1}$$

Here, variance is controlled by σ. c_k is the confidence score and x_k and y_k are the coordinates of the k-th joint. To create a 3D heatmap volume at the end, we can stack the heatmaps, along the temporal dimension, resulting in a volume with dimensions $K \times T \times H \times W$. The pose stream is similar to the workflow in section: 3.2.1 in most aspects, changes include removal of the last two blocks of the MoviNets backbone since in case of heatmaps, amount of spatial information that needs to be processed is less and thus the model achieves good results even after we truncate the last two blocks. Similar to RGB stream we create subsequences of 8 frames and restack the extracted features.

3.2.3 Multi Stream feature fusion

Since our final model is a multistream model, we adopted a feature fusion approach by concatenating the features extracted from the backbones of both streams, resulting in a final feature map of size $[N, 2 \times f]$ as seen in figure 2. The period predictor output was fused before the final fully connected layer by summing the two feature vectors rather than concatenating them. To improve the model performance further, we devised a two-stage training process. In the first stage, the pose stream was trained for repetition count independently. Then, in the second stage, we used a pretrained pose model and froze the backbone of the model, while still incorporating loss from the pose stream during training. However, the pose prediction was discarded during inference.

3.2.4 Classification branch

We use the concatenated feature vector from the two-streams, we take the mean of these features along the time dimensions and it is finally fed as input to a single fully connected layer for classification of

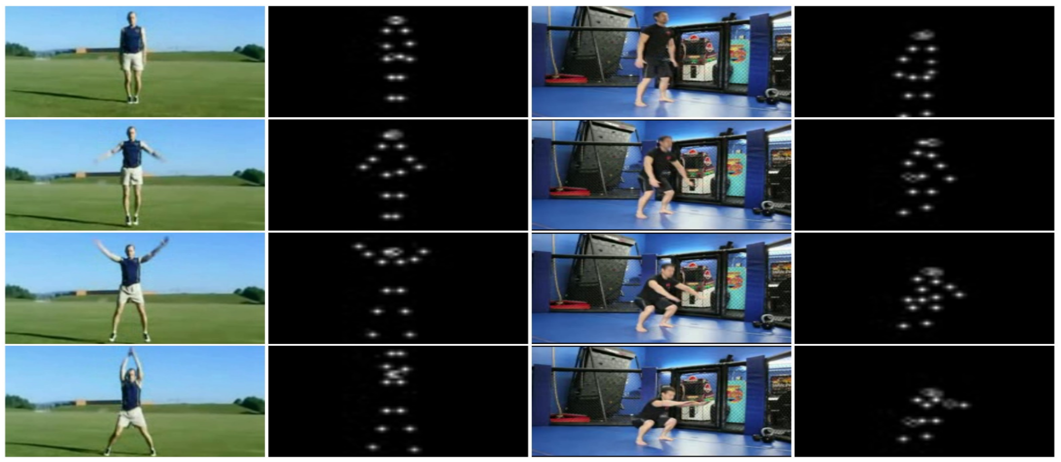


Figure 3: This figure shows frames from 2 exercises: Jumping Jacks and Bodyweight Squats along with their corresponding heatmaps.

actions. Both backbone networks were frozen, thus weights were only updated for the fully connected layer. Our final model achieved satisfactory accuracy. As an alternative to taking the mean across all frames, we could consider using density map predictions for each frame as weights to obtain a weighted mean. While this approach has the potential to improve performance, we were unable to explore it due to time constraints. Further investigation of the impact of this modification on repetition counting, where the classification loss does not contribute to gradients in the case of the mean but would in the case of density map weights, could be a valuable avenue for future work. Since for classification only single fully connected layer was added, it has minimal impact on the computation.

The final loss is given by:

$$L_{final} = L_{rgb} + \lambda_{pose} * L_{pose} + \lambda_{cls} * L_{cls} \tag{2}$$

Here L_{rgb} and L_{pose} are mean squared error for the density map prediction and the ground truth generated using gaussianization process (refer to section 3.3 for more details on it). In all our experiments, wherever valid, we used the values of λ_{pose} and λ_{cls} equal to 0.1. L_{cls} is the cross entropy loss on the classification branch.

3.3 Density map generation

As defined in [Hu+22], to compute the density map, we use Gaussian function [Guo11] and to be able to compute the Gaussian function with high interval of confidence, the values of variance σ and mean μ are needed. If for instance we have label $Y = [y_1, y_2, y_3, ..., y_n]$ where the starting and ending frames of a repetitive action are represented by y_i and y_j where $j = i + 1$. Now using the input $g_\sigma(x)$ along with y_i and y_j, the probability density distribution $G_\sigma(y)$. Then the density map can be calculated using:

$$d_k = \int_{y_k+0.5}^{y_k-0.5} G_\sigma(y)dy, \qquad k \in [i, j] \tag{3}$$

We can get the prediction density map $D = [d_1, d_2, d_3, ..., d_n]$.

3.4 Implementation details

We employed the [WD21] optimizer with [DJ21] as the base optimizer in our experimental setup. We found the [WD21] optimizer to be effective through our initial experimentation and through independent experimentation on other computer vision tasks. Similar to [Hu+22], we trained our model for 200 epochs with a decreasing learning rate of 8×10^{-6} and a batch size of 32, with the backbone layers of both Pose stream and RGB stream frozen. We used [Kon+21] backbone, specifically MoviNetA0 was used. The model was trained using a number of frames (N = 64). In contrast to the approach of [Hu+22], which utilized a sampling strategy with a consistent stride, we employed a uniform sampling strategy, dividing the total number of frames in the video (F) into bins equal to the number of frames to be sampled (N) and randomly sampling one frame from each bin to obtain N sampled frames. We used frame size of 224×224 for RGB stream and 56×56 for the Pose stream.

3.5 Inference

The performance of video action recognition is affected by the stride used during inference, especially in the case of action counting where actions can occur at varying frequencies. For example, performing a single skipping rope takes much less time than doing squats or bench pressing. If the same stride is used for these two actions, it may lead to inaccurate results. To address this, we propose a dynamic stride-based inference strategy that uses our action classification branch. This strategy is especially useful for untrimmed long sequences that occur in real-world workout scenarios.

Here's how the strategy works: for the first batch, we sample frames at a stride of 4, and obtain the action count and action class for this batch. For subsequent batches, we use a dynamic stride based on the action class predicted in the previous batch. We calculate the dynamic stride using the RepCount A dataset. For each class in the dataset, we calculate the average number of frames required to perform each action. We assume the number of frames per batch to be 64 and the number of actions in each batch to be 5, which was discovered empirically to work best. Using these values, we calculate the stride for each action.

4 Experimentations

In order to optimize computational efficiency, we initiated all our experimentation without the use of multi-scale proposed in [Hu+22]. To determine the optimal temporal length in the single scale, we conducted experiments using temporal lengths of $1, 4, 8, 16$ and found that a temporal length of 8 frames yielded the best results. Unlike [Hu+22], where the stride is half of the temporal length, all of our experiments were performed with a stride equal to the temporal length. We also explored various feature fusion strategies, including the use of sum instead of concatenation before the similarity matrix layer, as described in section 3.2.3. In addition, we present classification accuracy results that can serve as a baseline benchmark for future work.

4.1 Evaluation metrics

Similar to [Dwi+20] and [Hu+22], we use mean absolute error(MAE) and off by one count error (OBO). Mean Absolute Error is normalized absolute error between the ground truth count and the predicted count. Off by one error is If the predicted count is within one count of the ground truth, then we can say that the video has been counted correctly. If it is not within one count, then it is a counting error. This error rate is calculated over the entire dataset.

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i}, \tag{4}$$

$$\text{OBO} = \frac{1}{N} \sum_{i=1}^{N} [|y_i - \hat{y}_i| \leq 1] \tag{5}$$

In the above equations y_i is the ground truth count and $\hat{y}_i$ is the predicted count and N is total number of videos.

4.2 Benchmark and results

The benchmark results for other methods are borrowed from [Hu+22]. To the best of our knowledge our multi-stream model achieves state of the art results on RepCountA test split (table 1) and UCFRep dataset (table 4) and at the same time is computationally less intensive by a big margin. The inference times are calculated on 2070s gpu on a single batch of 64 frames and frame size of 224×224. This time also include inference of Yolov7 model for pose estimation. In most real world fitness applications, human pose can be reused for other tasks like human form detection. Unlike [Hu+22] our model runs at a single scale and thus in table 3 we show the affect on results because of temporal length. We also show the effect of using different feature fusion techniques after the backbone between multiple streams in table 4.The demo video which shows the prediction of our final model for classification, counting and density map plotted as graph which helps in visualizing repetition localization can be viewed **here**. As observed in the video, the model is able to correctly predict classes and count with a good accuracy, though model suffers when the person tries to get up and more investigation is required in the inference strategy, one solution could be to employ a dynamic sampling rate based on the class of action which can be helpful in long videos such as this.

Method	**MAE** $\downarrow$	**OBO** $\uparrow$	**Accuracy**	**Inference Time (in sec)**
RepNet [Dwi+20]	0.9950	0.0134	-	0
Zhang et al. [Zha+20]	0.8786	0.1554	-	0
TransRac [Hu+22]	0.4431	0.2913	-	1.11
Multi-stream	**0.4004**	**0.3778**	**83.4%**	0.21
RGB-stream	0.4427	0.3187	-	0.11
Pose-stream	0.7338	0.1542	-	0

Table 2: The table shows comparison of several methods on RepCountA: the test set released by the authors of [Hu+22].

Temporal Length	**MAE** $\downarrow$	**OBO** $\uparrow$
1	0.7512	0.2188
4	0.4903	0.2813
8	**0.4427**	**0.3187**
16	0.5187	0.2562

Table 3: The table shows the effect of different Temporal length for RGB stream model. We got best results with temporal length of 8.

UCFRep dataset	**MAE** $\downarrow$	**OBO** $\uparrow$
TransRac [Hu+22]	0.6401	0.324
Ours (Multi Stream)	**0.514**	**0.394**

Method	**MAE** $\downarrow$	**OBO** $\uparrow$
Sum	0.4717	**0.3778**
Concat	**0.4004**	**0.3778**

Table 4: The table on the left shows the comparison of [Hu+22] and Our multi stream model when the model is trained on RepCountA dataset and tested on the test split of UCFRep dataset. Table on the right shows the effect of using different feature fusion strategy after the backbone.

5 Conclusion

In this work, we propose a multi-stream model for repetitive action counting. The model uses RGB videos and the human pose extracted from the RGB frame as a second stream for feature extraction. To the best of our knowledge, our model improves results on both the RepCountA and UCFRep datasets and achieves state-of-the-art results on both and also reducing te amount of computation required over the previous approaches. The keypoints detected in human poses can be reused in fitness applications for detecting and improving the form of humans in exercises. We also add a classification branch to our network with minimal computational expense, as all features are shared with the repetition counting branch. Our classification branch was disconnected and the classification loss had no impact on the repetition counting. Further investigation can be done along this line, as the class of action can help us in dynamically choosing the frame length and stride from the video, which can lead to further improvement in results. We simulated this by carefully investigating the results on the longest test video, and the results improved by splitting the video into multiple parts.

6 Acknowledgement

The computational resources for this work were provided by the University of Maryland Institute for Advanced Computer Studies. We thank Aman Sharma and Ritesh Chowdhery for their valuable discussions throughout the project. We also appreciate the valuable discussions with Dr. Jia Bin Huang during doubt sessions and the support of the TAs via Piazza.

References

[CD00] Ross Cutler and Larry Davis. "Robust Real-Time Periodic Motion Detection, Analysis, and Applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (May 2000). DOI: 10.1109/34.868681.

[Lap+05] Ivan Laptev et al. "Periodic Motion Detection and Segmentation via Approximate Sequence Alignment." In: vol. 1. Jan. 2005, pp. 816–823. DOI: 10.1109/ICCV.2005.188.

[BA07] Alexia Briassouli and Narendra Ahuja. "Extraction and Analysis of Multiple Periodic Motions in Video Sequences". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.7 (2007), pp. 1244–1261. DOI: 10.1109/TPAMI.2007.1042.

[ABQ08] Alexandra Albu, Robert Bergevin, and Sébastien Quirion. "Generic temporal segmentation of cyclic human motion". In: *Pattern Recognition* 41 (Jan. 2008), pp. 6–21. DOI: 10.1016/j.patcog.2007.03.013.

[Guo11] Hongwei Guo. "A Simple Algorithm for Fitting a Gaussian Function [DSP Tips and Tricks]". In: *IEEE Signal Processing Magazine* 28 (2011), pp. 134–137.

[KO12] Takumi Kobayashi and Nobuyuki Otsu. "Motion recognition using local auto-correlation of space–time gradients". In: *Pattern Recognition Letters* 33 (July 2012), pp. 1188–1195. DOI: 10.1016/j.patrec.2012.01.007.

[WS13] Heng Wang and Cordelia Schmid. "Action Recognition with Improved Trajectories". In: Dec. 2013, pp. 3551–3558. DOI: 10.1109/ICCV.2013.441.

[GM15] Georgia Gkioxari and Jitendra Malik. "Finding action tubes". In: June 2015, pp. 759–768. DOI: 10.1109/CVPR.2015.7298676.

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked hourglass networks for human pose estimation". In: *European conference on computer vision*. Springer. 2016, pp. 483–499.

[CZ17] 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*. 2017, pp. 6299–6308.

[Che+17] Chen Chen et al. "Action recognition from depth sequences using weighted fusion of 2D and 3D auto-correlation of gradients features". In: *Multimedia Tools and Applications* 76 (Feb. 2017). DOI: 10.1007/s11042-016-3284-7.

[Ke+17] Qiuhong Ke et al. "A New Representation of Skeleton Sequences for 3D Action Recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 4570–4579.

[Cao+18] Zhe Cao et al. "OpenPose: realtime multi-person 2D pose estimation using part affinity fields. CoRR abs/1812.08008 (2018)". In: *arXiv preprint arXiv:1812.08008* (2018).

[LSF18] Weizhe Liu, Mathieu Salzmann, and Pascal V. Fua. "Context-Aware Crowd Counting". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 5094–5103.

[LPT18] Diogo Carbonera Luvizon, David Picard, and Hedi Tabia. "2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 5137–5146.

[PKA18] Costas Panagiotakis, Giorgos Karvounas, and Antonis A. Argyros. "Unsupervised Detection of Periodic Segments in Videos". In: *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), pp. 923–927.

[RLH18] Viresh Ranjan, Hieu M. Le, and Minh Hoai. "Iterative Crowd Counting". In: *European Conference on Computer Vision*. 2018.

[YXL18] Sijie Yan, Yuanjun Xiong, and Dahua Lin. "Spatial temporal graph convolutional networks for skeleton-based action recognition". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[Cae+19] Carlos Antônio Caetano et al. "SkeleMotion: A New Representation of Skeleton Joint Sequences based on Motion Information for 3D Action Recognition". In: *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (2019), pp. 1–8.

[Fei+19] Christoph Feichtenhofer et al. "Slowfast networks for video recognition". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6202–6211.

[Sun+19] Ke Sun et al. “Deep high-resolution representation learning for human pose estimation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 5693–5703.

[WC19] Jia Wan and Antoni B. Chan. “Adaptive Density Map Generation for Crowd Counting”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 1130–1139.

[Dwi+20] Debidatta Dwibedi et al. “Counting out time: Class agnostic video repetition counting in the wild”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 10387–10396.

[Fei20] Christoph Feichtenhofer. “X3d: Expanding architectures for efficient video recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 203–213.

[SHR20] David Strömbäck, Sangxia Huang, and Valentin Radu. “MM-Fit: Multimodal Deep Learning for Automatic Exercise Logging across Sensing Devices”. In: *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4.4 (Dec. 2020). DOI: `10.1145/3432701`. URL: `https://doi.org/10.1145/3432701`.

[Wan+20] Zhenzhi Wang et al. “Boundary-Aware Cascade Networks for Temporal Action Segmentation”. In: *ECCV (25)*. Vol. 12370. Lecture Notes in Computer Science. Springer, 2020, pp. 34–51.

[Zha+20] Huaidong Zhang et al. “Context-aware and scale-insensitive temporal repetition counting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 670–678.

[Che+21] Yuxin Chen et al. “Channel-wise topology refinement graph convolution for skeleton-based action recognition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13359–13368.

[DJ21] Aaron Defazio and Samy Jelassi. *Adaptivity without Compromise: A Momentumized, Adaptive, Dual Averaged Gradient Method for Stochastic Optimization*. 2021. arXiv: `2101.11075 [cs.LG]`.

[Ish+21] Yuchi Ishikawa et al. “Alleviating over-segmentation errors by detecting action boundaries”. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 2021, pp. 2322–2331.

[Kon+21] Dan Kondratyuk et al. “Movinets: Mobile video networks for efficient video recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 16020–16030.

[Liu+21] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[Qin+21] Zhiwu Qing et al. “Temporal context aggregation network for temporal action proposal refinement”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 485–494.

[WD21] Less Wright and Nestor Demeure. “Ranger21: a synergistic deep learning optimizer”. In: *arXiv preprint arXiv:2106.13731* (2021).

[Dua+22] Haodong Duan et al. “Revisiting skeleton-based action recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2969–2978.

[Hu+22] Huazhang Hu et al. “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*. 2022, pp. 19013–19022.

[Liu+22] Ze Liu et al. “Video swin transformer”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 3202–3211.

[WBL22] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *arXiv preprint arXiv:2207.02696* (2022).