Abstract

End-to-end learning has taken hold of many computer vision tasks, in particular, related to still images, with task-specific optimization yielding very strong performance. Nevertheless, human-centric action recognition is still largely dominated by hand-crafted pipelines, and only individual components are replaced by neural networks that typically operate on individual frames. In this work, we show that when appropriately labeled data is available, end-to-end learning on raw pixels from video can compete with, and in many cases outperform, state-of-the-art action recognition pipelines based on pose estimation. We present a new dataset as a real-world benchmark to compare transfer learning capabilities. This dataset consists of 40 granular video-level classes, 9 frame-level classes, spanning four fitness exercises, recorded in a home workout setting. Additionally, we train our models for real-time repetition counting using a temporally-annotated subset.

1 Introduction

Action recognition in videos has slowly been transitioning to real-world applications following extensive advancements in feature representation and deep learning-based architectures. In many applications, models need to extract detailed information of the underlying spatio-temporal dynamics. Towards this, end-to-end learning has recently had a lot of success on generic action recognition datasets comprised of varied everyday activities (3; 9; 23). However, pose-based pipelines seem to remain the preferred solution when the task is strongly related to analyzing body motions (1; 31; 20; 22; 40), such as in the rapidly growing application domain of virtual fitness, where an AI system can be used to deliver real-time form feedback and count exercise repetitions.

In this paper, we present a new fitness action recognition dataset with granular intra-exercise labels and compare few-shot learning abilities of pose estimation-based pipelines with end-to-end learning from raw pixels. We also compare the influence of using different pre-training datasets on the chosen models and additionally train them for repetition counting.

Common approaches to generic video understanding based on end-to-end learning include combinations of 2D-CNNs for spatial feature extraction followed by an LSTM module for learning temporal dynamics (5; 26), directly learning spatio-temporal dynamics with a 3D-CNN (10), or combining a 3D-CNN with an LSTM (24). The temporal understanding can be further improved in a two-stream approach with a second CNN-based stream trained on optical flow (3; 8; 35). The large parameter space of 3D-CNNs can be prohibitive and efforts to reduce this include dual-pathway approaches to low/high frame-rate (7) and resolution (6), temporally shifting frames in a 2D-CNN (19), and non-uniformly aggregating features temporally (17). Using a multi-task approach, an end-to-end model jointly trained for pose estimation and subsequent action classification was shown to improve performance of individual components (15) – but pose information is still needed for training.

For a pose-based pipeline, action classifiers typically use either hand-crafted features (27; 39; 38) or, increasingly, deep learning-based modules. Recent approaches have employed methods such as
Table 1: Side-by-side comparison of QAR-EVD (ours) versus common video datasets including NTU RGBD+D (20), FineGym (32), Jester (23), Something-something (9), Charades (34), Kinetics (12) and Moments (25) based on five criteria: a) focus on body motion, b) fine-grained label taxonomy (e.g. presence of intra-activity variations), c) controlled environment (e.g. fixed camera angle in a home environment), d) “in the wild” (as opposed to e.g. recorded in a lab), and e) dataset size sufficient for stand-alone pre-training.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>QAR-EVD</th>
<th>NTU RGBD</th>
<th>FineGym</th>
<th>Jester</th>
<th>Something-something</th>
<th>Charades</th>
<th>Kinetics</th>
<th>Moments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus on body motions</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Fine-grained labels</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Controlled environment</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>“In the wild”</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Large-scale</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

CNNs [13,14,16], LSTMs [21,22,31,41] or Graph CNNs [40,37,33]. Although these approaches have been popular in human-centric recognition tasks, due to strict modularization of pose estimation and action classification, they do not permit end-to-end training.

In addition to an appropriate model architecture, a dataset with a fine-grained action taxonomy is crucial to learning robust action representations. Existing RGB-based video datasets such as Kinetics (12), Moments in Time (25) and Sports-1M (11) are based on a high-level taxonomy and further, possess correlated scene-action pairings resulting in pronounced representation bias (4,18). These concerns can be mitigated through crowd-sourced collections of predefined labels where the same action can be collected from multiple workers such as in the Something-Something (9), and Charades (34) datasets. However, the "everyday general human actions" within these datasets are loosely specified and left to the worker’s interpretation resulting in a high inter-worker action variance. On the other hand, FineGym (31) focuses on specific fine-grained body motions but includes variability in camera position resulting in lower overall action salience. In contrast, gesture recognition datasets such as Jester (23) control camera and worker positioning and additionally, constrain human motion to appropriately specified hand gestures. A similarly constrained dataset for exact human body movement, that also controls camera motion, does not exist and we believe home fitness is the perfect domain in which to create one as workers can be instructed to move in very specific ways to perform exercises.

Pose-specific datasets contain an additional layer of annotated skeletal joints obtained either through annotation of scraped video datasets (either manually or using a pose estimation model (22)) or a sensor-derived approach in constrained lab settings (31,20). We present a new crowd-sourced benchmark dataset to fill a gap in the dataset landscape (see Table 1): videos of fitness exercises in a home setting are recorded in the wild providing challenging scene variety while also following a fine-grained label taxonomy. We compare end-to-end action classification models with state-of-the-art pose estimation-based action classifiers and show that the end-to-end approaches can outperform the pose estimation-based alternatives, if the end-to-end models are pre-trained on a large and granular labelled video corpus. We also show that the pose estimation models themselves can greatly benefit from pre-training on the large labelled dataset.

2 The Qualcomm AI Research - Exercise Videos Dataset – a new benchmark dataset

Fitness activities are defined by a well-constrained set of body movements outside of which an individual risks injury or ineffectiveness. There is an opportunity for AI systems to detect mistakes and provide real-time form-correcting feedback. To this end, we present the Qualcomm AI Research - Exercise Videos Dataset, referred later in this paper as QAR-EVD, comprised of granular video-level activity classes capturing subtle variations, including common mistakes. The dataset spans four fitness exercises recorded in a home environment by crowd workers:

- **Dead bug**: The user lies on the back with arms and legs raised and moves them back and forth asynchronously.
- **Inchworm**: From a standing position, the user touches the floor with both hands, walks them forwards, and then back again.
Figure 1: Samples from the QAR-EVD dataset for all four exercises. From top to bottom: Dead bug, inchworm, alternating lateral lunges, spiderman pushups. Best viewed on a screen.

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Validation</th>
<th>Test</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of videos</td>
<td>4000</td>
<td>711</td>
<td>800</td>
<td>5511</td>
</tr>
<tr>
<td>Number of unique workers</td>
<td>129</td>
<td>20</td>
<td>165</td>
<td>314</td>
</tr>
</tbody>
</table>

Table 2: Dataset statistics: Number of videos and unique crowd workers in each split

- **Alternating lateral lunges**: The user performs a lunge step in sideways direction, alternating in both directions.
- **Spiderman pushups**: A pushup variation where one leg is moving to touch knee and elbow.

Example frames from the dataset for each exercise can be seen in figure 1. Each exercise was recorded with deliberate variations such as increased pace or incorrect execution of different aspects of each exercise, some which are visible from a static frame (foot touching the floor), and others which are only apparent across multiple consecutive frames (being too fast or too slow). In total, a fine-grained taxonomy of 40 video-level classes is available to trigger direct feedback.

Each of the 40 classes contains between 130 and 140 videos, with each video lasting between 5 and 8 seconds. The dataset is split into train, validation and test sets with no worker overlap between them. All videos are provided in MP4 format at a frame rate of 30 fps. The dataset contains 5511 videos in total across all splits (see Table 2 for details on the data split). For few-shot experiments, we prepared different versions of the train splits, containing fewer examples per class. We release splits that contain 5, 10, 20, 50 and 100 samples per class.

In addition to delivering form feedback in real-time, another challenging task for fitness AI applications is repetition counting. It relies on precisely parsing the temporal extent of an action segment within an activity and, as such, benefits greatly from the availability of temporal annotations. To this end, in addition to providing video-level labels, we tagged a subset of each exercise within QAR-EVD with frame-level classes, thus making it possible to benchmark models on repetition counting. More details will be provided in Section 3.5.

The data has been collected in the wild by individual crowd workers who performed the actions following instructions from an example video. To match the desired viewing angle of a phone placed on the floor (fitness app scenario), the workers recorded themselves using a camera at a low position. All recorded videos were reviewed to confirm the execution was performed correctly. Because of the distributed nature of the data collection, the recorded samples show a large variety of scene settings, backgrounds and illumination (see figure 2). Each worker recorded videos for multiple action classes, so that the performed action cannot be learned by the visible video setting, but only by learning feature representations of the actual body motion.

**QAR-EVD** has been collected for the purpose of discerning fine variations of exercise execution performed by a worker. In order to create the label taxonomy and recording instructions, the domain
knowledge of several fitness experts had been consulted to collect a list of common mistakes and frequent variations of the individual exercises. Some examples of subtle variations are:

- **Dead bug**: A foot touches the floor; arms are not moving; the wrong leg is moving; execution is too fast
- **Inchworm**: Feet are too narrow or too wide; hands are too far from the body in the initial position; hands are stepping too far forward with each step
- **Alternating lateral lunges**: Bending the wrong leg; low range of motion; execution is too fast
- **Spiderman pushups**: Execution is too fast or too slow; leg movement is not in sync with pushup (three different error variations are labeled); pushup is too shallow

<table>
<thead>
<tr>
<th><strong>QAR-EVD exercises</strong></th>
<th>Closest <strong>BigFitness classes</strong></th>
<th>Closest <strong>Kinetics classes</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>spiderman pushups</td>
<td>pushups - sloppy</td>
<td>push up</td>
</tr>
<tr>
<td></td>
<td>burpee - no upright position</td>
<td>crawling baby</td>
</tr>
<tr>
<td></td>
<td>burpee - no jump</td>
<td>headbanging</td>
</tr>
<tr>
<td>dead bug</td>
<td>bicycle crunches - small torso rotation</td>
<td>lunge</td>
</tr>
<tr>
<td></td>
<td>bicycle crunches - medium torso rotation</td>
<td>side kick</td>
</tr>
<tr>
<td></td>
<td>bicycle crunches - head down</td>
<td>squat</td>
</tr>
<tr>
<td>alternating lateral lunges</td>
<td>skaters - single jump (right to left)</td>
<td>situp</td>
</tr>
<tr>
<td></td>
<td>grabbing an off-screen towel</td>
<td>knitting</td>
</tr>
<tr>
<td></td>
<td>skaters - slow</td>
<td>unboxing</td>
</tr>
<tr>
<td>inchworm</td>
<td>burpee (no pushup) - stepping feet forward</td>
<td>dribbling basketball</td>
</tr>
<tr>
<td></td>
<td>burpee (no pushup) - stepping feet back</td>
<td>deadlifting</td>
</tr>
<tr>
<td></td>
<td>roll down</td>
<td>push up</td>
</tr>
</tbody>
</table>

Table 3: Comparing dataset similarity: for each **QAR-EVD exercise** (column 1), we compute a prototypical feature vector and show its 3 closest class centroids in feature space within **BigFitness** (column 2) and **Kinetics** (column 3).
We plan to release the dataset under a non-commercial license, which permits non-profit research only. The dataset will be hosted under https://developer.qualcomm.com/. For an example of a similar, existing dataset and license, see: https://developer.qualcomm.com/software/ai-datasets/jester

2.1 Datasets used for pre-training

In addition to the dataset we are releasing along with this paper, we use a larger internal video dataset, which we refer to as BigFitness, for pre-training in some experiments. This dataset consists of around 300,000 videos of fitness exercises with a fine-grained label taxonomy across 1,536 classes that are disjoint from the data in QAR-EVD. Similar to QAR-EVD, the videos were recorded and curated by crowd-workers. This dataset will not be released.

In addition to this internal dataset, we also made use of Kinetics (12) and ImageNet (30) for pre-training, as will be described in the results section.

To elucidate the relationship between the pre-training datasets used in our experiments and QAR-EVD, we visualize examples that are the closest in feature space to each QAR-EVD exercise in Table 3. It shows that BigFitness has multiple labels that are conceptually similar, which is to be expected as it contains fitness actions with a disjoint, but also fine-grained, label taxonomy. More general action recognition datasets like Kinetics have some fitness actions, such as push up and lunge, which resemble the exercises from QAR-EVD. However, because of the more coarse label taxonomy, the next nearest neighbors can be very different (such as labels “head banging” or “unboxing”).

3 Experiments

All models were trained on subsets of the QAR-EVD training split, with 5, 10, 20, 50, and 100 samples per class, to evaluate few-shot behavior. Different initialization approaches were tested for each model, including training from scratch, starting from a pre-trained model and fine-tuning the final classification layer, all layers, or a subset of the layers. The approaches are described in more detail in section 3.3.

3.1 Architectures

Three end-to-end and two pose estimation-based architectures are compared in our experiments. End-to-end architectures include I3D (3), SI-EN (ours) and SI-BlazePose (ours). The pose estimation pipelines have three components: human detection and localization, pose estimation and graph-CNN-based classification. BlazePose(1) was selected as the pose estimation model for each pipeline with the following state-of-the-art graph-based classifiers: ST-GCN (40) and MS-G3D (22).

3.1.1 End-to-end

I3D. As an end-to-end baseline model for video action recognition, we used the 3D-CNN architecture, I3D-RGB, proposed in (3).

Strided-Inflated EfficientNet (SI-EN). We present SI-EN, which uses EfficientNet-Lite4 (36), a 2D-CNN, as a backbone, with a few modifications to some of the inverted residual blocks. Specifically, we inflate 8 of the blocks in the temporal dimension (blocks 3, 7, 11, 14, 17, 20, 23 and 25), using a temporal kernel of 3, effectively turning them into 3D convolutional modules taking inspiration from (3). More precisely, it is only the first point-wise convolution in the inverted residual block that is inflated. Two of the inflated convolutions (blocks 7 and 14) are implemented with a stride of 2, enabling a lower footprint output of 4 fps from the 16 fps input stream.

SI-BlazePose. As a method to back-propagate through a pose feature bottleneck during an end-to-end classification task, we propose the following architecture which we call SI-BlazePose. It is based on the BlazePose model (1) using inflation to extend it in the temporal dimension. We inflate the last 8 point-wise convolutions with a temporal kernel of size 3, adding a temporal stride of 2 to the 2nd and 4th one. We freeze all layers before the first inflated layer. We use the full image as input, and resize it to 256 × 256 preserving the aspect ratio. We did not crop around the person as a first step, in
contrast to what is done within MediaPipe. Since QAR-EVD is a classification dataset, we replace BlazePose’s body part regression head with a softmax layer.

### 3.1.2 Pose estimation-based

**ST-GCN.** Spatial-temporal graph convolution networks (ST-GCN) use graph convolutions across spatial joint connections and temporal connections from frame to frame (40). Following the original authors’ approach, we included their suggested edge importance weighting method with a spatial partitioning strategy. As our results did not benefit from the enabled dropout regularization, we disabled it.

**MS-G3D.** Multi-scale graph convolutional networks (MS-G3D) adjust the node weighting in the graph for improved multi-scale aggregation and introduce skip connections to the graph for better modeling of spatio-temporal dependencies across longer distances.

As these models are able to work on generic graph layouts, we added support for the BlazePose layout by providing the adjacency matrix of the 33 graph nodes.

### 3.1.3 A note on computational efficiency

A pipeline based on pose-estimation typically consists of 3 components: a detection network producing rough person positions, a pose estimation network producing skeletons for each person (BlazePose in our case), and a classifier mapping a sequence of skeletons to an activity label (ST-GCN or MSG3D in our case). The first two components are image-based while the action classifier is video-based in the sense that it needs a sequence of skeletons. While the detection network can run fairly infrequently (at least, if the person is not moving their position much), the framerate at which the pose estimation component needs to run is determined by the temporal granularity required by the action classifier to obtain high accuracy. An end-to-end neural network on the other hand provides a variety of flexible ways to reduce computational footprint, e.g. by using temporally strided convolutions which reduces the framerate of subsequent layers and outputs. SI-EN specifically exploits this by introducing two 3D convolutions with a temporal stride of 2 early in the architecture. As a result, most of the SI-EN layers run at 4fps rather than the 16fps input framerate. We estimate the computational footprint of the methods using DeepSpeed (28). At an input framerate of 16, SI-EN only requires 4.0 GMACs/s, whereas running BlazePose alone already amounts to 6.7 GMACs/s (i.e. without counting localization and action classification).

![Figure 3: QAR-EVD top-1 accuracy of selected existing architectures, pretrained on various datasets.](https://google.github.io/mediapipe)
3.2 Implementation details

3.2.1 End-to-end

End-to-end models were trained on raw pixels from the QAR-EVD videos. The native resolution was down-scaled to a resolution of 256 × 256 pixels. To keep the original aspect ratio, frames were padded with black pixels to be in a square format before downscaling. Videos were subsampled to 16 fps which showed improved performance over the native 30 fps in preliminary experiments. For training, we took random crops of 63 frames from each video, which corresponds to roughly 4 second long video clips. 63 was chosen because of memory constraints. For evaluation, all frames of a video were passed to the model. As additional augmentation, we applied random color jittering to the 3 input channels. RGB values were scaled to the range from 0 to 1.

3.2.2 Pose estimation

To pre-train pose estimation models in a way that is comparable to the end-to-end models, we extracted pose features from BigFitness using BlazePose (1) as provided by the MediaPipe library (2). The same method was used to extract pose features to train on QAR-EVD. In our experiments, we used all 33 joints and 3 input channels per joint: x position, y position and confidence score. The resulting pose sequences were created at 16 fps, because preliminary experiments showed better results than using the raw 30 fps (just like in the end-to-end experiments). For training, we took random crops of 90 consecutive poses. For evaluation, we passed in the full pose sequence of each sample. Following (40), we used simulated camera movement on top of keypoint coordinates as a data augmentation technique during training.

The Kinetics-Skeleton dataset (40), that we use for pre-training some of the models, uses the OpenPose (2) layout, which has fewer key points than the BlazePose layout (18 instead of 33). In our experiments, we mapped BlazePose keypoints to the OpenPose format with the neck position being defined as the center between the two shoulder joints.

3.3 Results

The performance on QAR-EVD across architectures is reported in Figure 3. For each model, we have tried multiple fine-tuning strategies (e.g. freezing all layers, fine-tuning a subset of the layers, fine-tuning the whole network). Figure 3 only reports the approach that worked the best for each model. Results obtained using the other strategies can be found in Table 4. Regarding pose-based baselines, to the best of our knowledge, there are no versions of MS-G3D and ST-GCN pre-trained on the 33 joints returned by BlazePose and we therefore train the two graph CNNs from scratch in this experiment. We investigate the effect of pre-training MSG-3D and ST-GCN in the next section.

Interesting findings from Figure 3 can be summarized as follows:

- From scratch
- Pretrained on Kinetics
- Pretrained on BigFitness
- ST-EF-BigFitness-10

Figure 4: Effect of pre-training ST-GCN and MS-G3D on Kinetics and BigFitness.
Best performance is obtained by an end-to-end network. SI-EN-BigFitness-10 tops all other approaches with a significant margin, including pose-based solutions that use a graph CNN initialized from random weights. The gap with pose-based pipelines is higher when training data is scarce (45.2% vs 26.7% for ST-GCN-Scratch in the 5-shots case) but shrinks as more training samples are available (66.8% vs 62.1% for MS-G3D-Scratch when the full trainset is used).

Pre-training on a large video dataset is key. Unsurprisingly, the type of data used to pre-train each baseline plays an important role in downstream performance. Best results are obtained by the model that was pre-trained on BigFitness, which is by far the most granular pre-training dataset considered in this experiment. The exact same SI-EN architecture pre-trained on ImageNet performs poorly. The Kinetics baseline, I3D, is roughly on par with pose-based pipelines. On the other hand, the inflated pose 2D CNN, SI-BlazePose-9, obtains decent results when few samples are available but gets significantly outperformed as more samples are available.

MS-G3D seems more prone to overfitting than ST-GCN. While MS-G3D outperforms ST-GCN when more than 50 training samples are available, ST-GCN gets better results in the 5, 10 and 20-shot cases.

<table>
<thead>
<tr>
<th>Number of samples per class:</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>End-to-end</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI-EN-ImageNet</td>
<td>8.9</td>
<td>15.5</td>
<td>23.9</td>
<td>33.5</td>
<td>38.1</td>
</tr>
<tr>
<td>SI-BlazePose</td>
<td>25.3</td>
<td>31.1</td>
<td>39.9</td>
<td>47.1</td>
<td>52.9</td>
</tr>
<tr>
<td>I3D-Kinetics-1</td>
<td>12.2</td>
<td>17.1</td>
<td>22.5</td>
<td>25.9</td>
<td>28.4</td>
</tr>
<tr>
<td>I3D-Kinetics-4</td>
<td>18.9</td>
<td>28.6</td>
<td>39.8</td>
<td>51.5</td>
<td>56.4</td>
</tr>
<tr>
<td>I3D-Kinetics-all</td>
<td>19.4</td>
<td>28.7</td>
<td>43.3</td>
<td>53.9</td>
<td>60.9</td>
</tr>
<tr>
<td>SI-EN-BigFitness-1</td>
<td>38.1</td>
<td>44.4</td>
<td>49.5</td>
<td>56.0</td>
<td>58.9</td>
</tr>
<tr>
<td>SI-EN-BigFitness-10</td>
<td>45.2</td>
<td>50.7</td>
<td>56.0</td>
<td>63.5</td>
<td>66.8</td>
</tr>
<tr>
<td>SI-EN-BigFitness-all</td>
<td>36.2</td>
<td>43.6</td>
<td>51.5</td>
<td>60.8</td>
<td>63.6</td>
</tr>
<tr>
<td><strong>Pose-based pipeline</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST-GCN-Scratch</td>
<td>26.7</td>
<td>39.4</td>
<td>45.4</td>
<td>53.7</td>
<td>57.4</td>
</tr>
<tr>
<td>MS-G3D-Scratch</td>
<td>25.5</td>
<td>32.3</td>
<td>44.7</td>
<td>57.6</td>
<td>62.1</td>
</tr>
<tr>
<td>ST-GCN-Kinetics</td>
<td>30.8</td>
<td>39.1</td>
<td>49.9</td>
<td>58.0</td>
<td>60.7</td>
</tr>
<tr>
<td>MS-G3D-Kinetics</td>
<td>38.9</td>
<td>47.2</td>
<td>53.5</td>
<td>62.2</td>
<td>65.6</td>
</tr>
<tr>
<td>ST-GCN-BigFitness</td>
<td>38.7</td>
<td>49.1</td>
<td>53.6</td>
<td>59.7</td>
<td>63.0</td>
</tr>
<tr>
<td>MS-G3D-BigFitness</td>
<td>41.9</td>
<td>51.6</td>
<td>56.3</td>
<td>62.2</td>
<td>65.5</td>
</tr>
</tbody>
</table>

Table 4: Results across all experiments. We report the test set accuracy in percentage on QAR-EVD.

<table>
<thead>
<tr>
<th>Temporal annotations schemes</th>
<th>pushup</th>
<th>dead bug</th>
<th>lateral lunges</th>
<th>inchworm</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI-EN (1) within-repetition vs end-of-repetition</td>
<td>25.9</td>
<td>39.3</td>
<td>33.3</td>
<td>109.0</td>
</tr>
<tr>
<td>SI-EN (2) within vs middle-of vs end-of-repetition</td>
<td>17.1</td>
<td>22.3</td>
<td>13.4</td>
<td>49.2</td>
</tr>
<tr>
<td>SI-EN (3) first half vs second half</td>
<td>4.6</td>
<td>7.2</td>
<td>2.2</td>
<td>21.5</td>
</tr>
<tr>
<td>MSG3D (1) within-repetition vs. end-of-repetition</td>
<td>22.2</td>
<td>40.1</td>
<td>38.4</td>
<td>102.0</td>
</tr>
<tr>
<td>MSG3D (2) within vs middle-of vs end-of-repetition</td>
<td>10.6</td>
<td>27.5</td>
<td>9.0</td>
<td>51.0</td>
</tr>
<tr>
<td>MSG3D (3) first half vs second half</td>
<td>4.9</td>
<td>8.5</td>
<td>4.2</td>
<td>17.2</td>
</tr>
<tr>
<td>STGCN (1) within-repetition vs. end-of-repetition</td>
<td>37.3</td>
<td>81.8</td>
<td>66.3</td>
<td>144.0</td>
</tr>
<tr>
<td>STGCN (2) within vs middle-of vs end-of-repetition</td>
<td>11.9</td>
<td>12.8</td>
<td>7.0</td>
<td>46.5</td>
</tr>
<tr>
<td>STGCN (3) first half vs second half</td>
<td>6.0</td>
<td>13.7</td>
<td>3.6</td>
<td>22.0</td>
</tr>
</tbody>
</table>

Table 5: Repetition counting results across all experiments (mean absolute percentage error).

3.4 Closing the gap between pose-based and end-to-end approaches

In this section, we investigate the effect of pre-training the graph CNN component of pose-based pipelines. Pre-training is performed with two datasets: Kinetics and BigFitness. Results can be found in Figure 4. Figure 4 shows that, even for a pose-based pipeline, pre-training on a large video dataset can boost classification accuracy. While an accurate frame-level pose representation alone obtains decent results, the overall solution greatly benefits from pre-training on videos. This suggests that training data that provides some understanding of the temporal aspects of human body motions is highly beneficial, even for pose-based models. While pre-training on Kinetics produces good downstream performance, pre-training on a more granular dataset such as BigFitness works better overall. When it is pre-trained on BigFitness, the MS-G3D-based pipeline is on par with the end-to-end baseline, and the advantage that ST-GCN has over MS-G3D in the lower data regimes vanishes. Additional metrics (e.g. confusion matrices, f-measures) can be found in the supplementary material.
3.5 Learning to count

To explore a more temporally fine-grained recognition task, we also experiment with end-to-end repetition counting ("how many times has a given exercise been performed?"). This is a common task, in particular, in many fitness applications.

Repetition counting is an inherently temporal prediction task. To train the networks on this task, we temporally annotated a subset of videos with frame-by-frame labels describing which phase of the exercise the subject is at any moment in time. We use the same frame-rates as before (16 fps input, 4 fps output) and annotated 100 videos for each of the exercises in the training set. We use the same train/test split as above for evaluation. We experiment with various temporal annotation schemes that can be turned into counts after training: (1) marking frames as within-repetition vs. end-of-repetition, (2) marking frames as within-repetition vs. end-of-repetition vs. middle-of-repetition, (3) using a different encoding of the 3-way annotations in (2), by marking frames as first-half of a repetition (between end-of-repetition and middle-of-repetition) vs. second-half of a repetition (between middle-of-repetition and end-of-repetition).

We train the models by treating these annotations as a simple temporal classification task. For training, we concatenate the videos within a mini-batch along the temporal axis rather than stacking videos on the batch-axis. Since annotation schemes (1) and (2) are highly imbalanced, we weight the classification cost by 0.2 for the under-represented class "within-repetition" during training. For SI-EN, we only train the 10 final layers (as for the best model above).

We turn temporal classifications into counts at inference time by incrementing the count when the end of a repetition is detected. For annotation schemes (1) and (2), we only increment the count if an end-of-repetition event is followed by at least one middle-of-repetition event to avoid over-counting. Table 5 shows the performance of the models in terms of mean absolute percentage error (MAE) (29; 15). It shows that accurate counting performance can be obtained from the relatively small number of annotated videos. While performance is comparable across models, interestingly, even in this setup, the end-to-end approach SI-EN performs roughly on par with or better than the other approaches in most cases. In fact, it shows the best performance in all exercises except for "Inchworm" which unlike the other exercises, has a much smaller number of repetitions per video and yields overall lower accuracy. Note that only SI-EN can make predictions on-line. Overall, while a deeper analysis and comparison with other counting approaches is beyond the scope of this paper, we find that it is possible to obtain very accurate repetition counts entirely end-to-end. We also find that accuracy depends strongly on how temporal annotations are represented during training.

4 Conclusion

In conclusion, we find that action recognition does not need pose estimation. Our experiments show that end-to-end training on large-scale labeled video datasets without any form of frame-by-frame intermediate representation can outperform pose-based approaches, even in the context of a fitness use case where one could assume that an accurate pose representation is all you need. More importantly, regardless of the selected approach, pre-training on a large and granular video dataset is a key ingredient to achieving good downstream performance. In fact, our experiments show that good performance in action recognition tasks is mostly a function of dataset size and label granularity and less of the choice of model.

Limitations and broader impact. The introduced dataset subserves research on end-to-end reasoning about human activities using an RGB camera. It can be used to study and benchmark model architectures and to rethink workflows in the development of end-to-end neural networks. However, the dataset in its current size and form may contain biases. Training on this dataset alone may, for example, lead to models whose behaviors could depend on a subject’s age, gender, ethnic background, etc. As such, the dataset as defined is suitable only for performing the research needs described above.

In addition, model behavior will be a function of camera angle, lighting, and possibly other random aspects of the scene, camera, camera-angle and the subject interacting with the model. As for positive impact, research towards enabling quantitative assessment of health and fitness-related activities with just a camera can democratize access to and can greatly improve individuals’ understanding of such activities and help unlock their benefits.
References


1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see the Limitations and broader impact section.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see the Limitations and broader impact section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The benchmark dataset used to evaluate all models will be released upon paper publication. [No] There is no plan to release the code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Experiments section.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Section 2.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [No]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [No]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [No]