DIY Human Action dataset Generation

Mehran Khodabandeh *  Hamid Reza Vaezi Joze *  Ilya Zharkov  Vivek Pradeep

mkhodaba@sfu.ca  {hava,zharkov,vpradeep}@microsoft.com

Simón Fraser University  Microsoft

Abstract

The recent successes in applying deep learning techniques to solve standard computer vision problems has inspired researchers to propose new computer vision problems in different domains. As previously established in the field, training data itself plays a significant role in the machine learning process, especially deep learning approaches which are data hungry. In order to solve each new problem and get a decent performance, a large amount of data needs to be captured which may in many cases pose logistical difficulties. Therefore, the ability to generate de novo data or expand an existing dataset, however small, in order to satisfy data requirement of current networks may be invaluable. Herein, we introduce a novel way to partition an action video clip into action, subject and context. Each part is manipulated separately and reassembled with our proposed video generation technique. Furthermore, our novel human skeleton trajectory generation along with our proposed video generation technique, enables us to generate unlimited action recognition training data. These techniques enables us to generate video action clips from a small set without costly and time-consuming data acquisition. Lastly, we prove through extensive set of experiments on two small human action recognition datasets, that this new data generation technique can improve the performance of current action recognition neural nets.

1. Introduction

After significant successes in face detection, face recognition and object detection commonly used in our daily life, computer vision researchers are now aiming at understanding video which is one dimension more difficult. These successes rely on advanced machine learning techniques and training data which require computational power, mainly deep networks. Hence, the process of data acquisition may be as vital as the technique used. Large datasets, such as a million object and animal photos [26], hundreds of thousands of faces [22] or millions of scenes [29], enables complex neural networks to train successfully. However, similar results can never be achieved through small datasets manually captured by researchers themselves. Video datasets or specifically human action datasets are more difficult to compile. A few common scenarios to generate a human action dataset are as follows: (1) asking subjects to do a series of actions in front of a camera (2) collecting and labeling existing videos from the internet or crowd sourcing [49] (3) 3D video synthesizing [58, 5]. The first scenario is not scalable considering the number of subjects and the limitations imposed by the capturing environment. These types of datasets are not common anymore due to their small size. Some examples of the second scenario are UCF 101 [51] containing 101 actions of thousands of online clips, Hollywood2 [32] containing 12 actions in ~3000 clip extracted from movies and the kinetics [21] including 400 actions from hundreds of thousands of YouTube videos. Although

*Equal contribution.
these datasets are very useful to benchmark the accuracy of different algorithms, the clips or actions are not necessarily useful for real world action recognition tasks such as security surveillance cameras, sport analysis, smart home devices, health monitoring etc, as each scenario has different settings and sets of actions. The drawback of the last scenario, which is more recent and seems more promising requires MoCap data.

In this paper, we’ve introduced a novel way to partition an action video clip into action, subject and context. We showed that we can manipulate each part separately and assemble them with our proposed video generation model into new clips. The actions are represented by a series of skeletons, the context is an still image or a video clip, and the subject is represented by random images of the same person. We can change an action by extracting it from an arbitrary video clip, generate it through our proposed skeleton trajectory model, or by applying perspective transform on existing skeleton. Additionally, we can change the subject and the context using arbitrary video clips, enabling us to arbitrarily generate action clips. This is particularly useful for action recognition models which require large datasets to increase their accuracy. With the use of a large unlabeled data and a small set of labeled data, we can synthesize a realistic set of training data for training a deep model.

We called it DIY (do it yourself) because we can eventually build our own dataset from a small one. Similar to actual data collection, not only can we add a new person or action to the dataset, but also internally expand the dataset or capture the same data from different angles with very little time and effort.

Lastly, to quantitatively evaluate our data generation technique, we applied it to UT Kinects [65] a human action dataset comprised of 10 actions in 200 video clips. We generated new video clip types by adding new subjects or actions or by expanding current action and subjects. It is shown that generated data along with the existing data, can improve the performance of well-performed video representation networks: 13D [4] and C3D [54] on action recognition task. For further investigation, we applied our method and action recognition task to actions with two persons in SUB interact [69] datasets. The outline of this paper is as follows. In §2 we’ve described related works in action recognition, data augmentation and video generative model. Section 3 introduces our video generation methods as well as skeleton trajectory generation methods with samples and use cases. In §4, we’ve discussed the datasets and action recognition methods used to evaluate our work. In §5 we’ve presented the extensive experimental data backing our claims. Our paper is concluded in §6.

2. Related Works

2.1. Action Recognition

Human action recognition has drawn attention for some time. Before deep learning era of computer vision, many researchers tried to inflate successful 2D features or descriptors in order to solve this problem such as 3d SIFT [47], 3d bag of features [27], dense trajectories [62], tracking [41, 40], and automatic target recognition [42]. Please refer to [36] for a comprehensive survey of these types of algorithms.

Deep learning networks significantly outperformed transitional approaches and are therefore the focus of this paper. Unlike image representation network architecture, the video representation networks haven’t had satisfactory advances. There have been different approaches to this problem. Some used the convolution and layers in 2D (image-based) [8, 68] while some used 3D (video-based) kernels [16, 54, 4]. Input to the networks could be just RGB video [54] while optical flow could be used as an additional input [10, 4]. Information could propagate across frames either through LSTMs [8, 68] or feature aggregation [19].

Data Augmentation Using synthetic data or data warping for training classifiers has been proven effective [26, 71, 50]. Sato et al. [45] proposes a method for training a neural network classifier using augmented data. Wong et al. [64] thoroughly investigated the benefits of data augmentation for classification tasks. In action recognition tasks, data is usually very limited [24], since collecting and annotating videos [23] is difficult. Although one can use our algorithm for data augmentation by generating videos varying in background, human appearance, and type of actions, this is not the purpose of our work. Unlike data augmentation that is limited to manipulating data, our method is capable of generating new data with new content and visual features.

2.2. Video Generative Models

Video generation has posed as a challenge for a number of years. The early work in the field focused on generating texture [9, 53, 63]. In recent years with the success of generative models in image generation such as VAEs [12], Plug&Play Generative Networks [34], MoCap data. Please refer to [36] for a comprehensive survey of these types of algorithms.

2.2. Video Generative Models

Video generation has posed as a challenge for a number of years. The early work in the field focused on generating texture [9, 53, 63]. In recent years with the success of generative models in image generation such as VAEs [12], Plug&Play Generative Networks [34], Moment Matching Networks [28], and PixelCNNs [57], a new window of opportunity has opened towards generating videos using generative models. In this paper, we use GANs to generate human skeleton trajectories and realistic video sequences. GAN consists of a discriminator and a generator, trained in a 2-player zero-sum game. Although GANs have shown promising results on image generation [7, 37, 70, 31, 30], they have proven to be difficult to train. To address this issue, Arjovsky et al. [1] proposed Wassertein GAN to combat mode collapse with more stability. Salimans et al. [44] introduced several tricks for training

GANs have previously been used for video generation. There are two lines of work in video generation. First is video prediction where given the first few frames of a video, the goal is to predict the future frames. Several papers focus on producing pixel values conditioned on the past observed frames [67, 52, 35, 33, 18, 66, 59]. Another group of papers aimed at reordering the pixels from the previous frames to generate the new ones [56, 11].

In the second line of work, the goal is to generate a sequence of video frames conditioned on label, single frame, etc. Early attempts assumed video clips to be fixed length and embedded in a latent space [60, 45]. Tulyakov et al. [55] proposed to decompose motion from content and generate videos using a recurrent neural net. Our work is different from [55] where their model learns motion and content in the same network whereas we separated them completely. Furthermore, [55] is not capable of generating complex human motions. Also filling gaps in the background initially blocked by the person in the input video is a difficult task for this method. On the other hand, our method handles these challenges by completely separating appearance, background, and motion. Our work is somewhat similar to [61], which does video forecasting using pose estimation, by modeling the movement of human using a VAE and then using a GAN to predict the pixel value of the future frames.

Our work lies in the "video generation" category where we focus on employing video generation techniques to generate human action videos. In our proposed method we completely separate background, skeleton motion, and appearance, allowing us to model frame generation and skeleton trajectory independently. So, one would require labeled data and the other can benefit from unlimited unlabeled human action videos available on internet, respectively.

3. Method

We define problem as follows; given an action label $l$ a small set of reference images $I = \{I_1, \ldots, I_k\}$ each containing a human subject from which a sequence of video frames is generated featuring a human with the same appearance as the human in the reference image set $I$ performing an action $l$. Modeling the (human/camera) motion and generating photo-realistic video frames may be challenging but knowing the location/motion of human skeletons in each frame would simplify it. Hence, we subdivided the problem into two simpler tasks (inspired by [55, 59]).

- The first task comprised of the reference images $I$, background image $B$, and a sequence of target skeletons $S = [S_1, S_2, \ldots, S_n]$ employed to render photo-realistic video frames of the person in $I$ moving according to $S$ on background.

- The second task produced the target skeleton sequences for the first part. In another words, given action label $l$, a sequence of skeletons of a random person performing action $l$ was generated.

By combining the two tasks, we created a novel algorithm that can generate arbitrary number of human action videos with varying backgrounds, human appearances, actions, and ways each action is performed.

3.1. Video Generation from Skeleton and Reference Appearance

In this section, we explain our algorithm used to generate a video sequence of a person based on given appear-
ance \((I)\) and a series of target skeletons \((S)\) in an arbitrary background \((B)\). In our proposed model, we use GAN conditioned on the appearance, the target skeleton, and the background. Our proposed generator network works in a frame-by-frame fashion, where each frame is generated independently from others. We have tried using LSTMs and RNNs to take into account smoothness of the videos. However, our experiments show frames that are generated separately are sharper as RNNs/LSTMs may introduce blurriness to the generated frames.

**Generator Input.** Our generator network needs a reference image of the person in order to generate images of the same person with arbitrary poses/backgrounds. However, one reference image may not have all the appearance information due to occlusions in some poses (e.g. face is not visible when the person is not facing the camera). To overcome this issue to some extent, we provided multiple reference images of the person to the network. In both training and testing, these images were selected completely at random, so that network would be responsible for choosing the right pieces of appearance features from the set of input images. These images could be selected with a better heuristic to produce better results though this is not in the scope of this work.

The reference images were pre-processed before incorporation into the network. First we extracted the human skeleton from each reference image \(I_i\) (using \([3]\)), then used an offline transform to map the RGB pixel values of each skeleton part from the image to the target skeleton. Also, a binary mask of where the transformed skeleton is located was created. All these images, \(I' = \{I'_1, ..., I'_k\}\), along with the background, \(B\), and the target skeleton, \(S_i\), were stacked.

**Conditional GAN.** Inspired by pix2pix \([15]\), we used a U-net style conditional GAN. The generator \(G(C)\), is conditioned on the set of transformed images and corresponding masks, along with the background and target skeleton. The generator, \(G\), maps \(C = \{I'_1, ..., I'_k, B, S_i\}\) to the target frame \(Y\), such that it fools the discriminator, \(D(C,Y)\). The discriminator, \(D(C,Y)\), on the other hand is trained to discriminate between real images and the fake images generated by \(G\). The architecture of the discriminator is illustrated in Fig. 3. The pipeline and architecture of the generator \(G\) is illustrated in Fig. 2. Fig. 4a illustrates some of the results.

The objective function of GAN is expressed as:

\[
\mathcal{L}_{G A N}(G, D) = \mathbb{E}_{c,y \sim P_{data}(c,y)}[\log D(c,y)] + \mathbb{E}_{c \sim P_{data}(c), z \sim P_{z}(z)}[1 - \log D(c,G(c,z))] + \beta \mathcal{L}_{R}(G) + \lambda \mathcal{L}_{L1}(G)
\]

Following \([15]\) we added an \(L1\) loss to the objective function, which resulted in sharper generated frames.

\[
\mathcal{L}_{L1}(G) = \mathbb{E}_{c,y \sim P_{data}(c,y), z \sim P_{z}(z)}[\|y - G(c,z)\|_1]
\]

In initial experiments, we noticed that using only \(L1\) loss and GAN loss is not enough as the output background would be sharp but the region that the target person is supposed to be was blurry. Subsequently, we introduced a "Regional \(L1\) loss" with a larger weight as following,

\[
\mathcal{L}_{R}(G) = \mathbb{E}_{c,y \sim P_{data}(c,y), z \sim P_{z}(z)}[\|\text{masked}(y) - \text{masked}(G(c,z))\|_1]
\]

where "masked" masks out the region where the person was located. This mask was generated based on the target skeleton, \(S_i\), using morphological functions (erode, etc.).

Our final objective is as follows:

\[
\mathcal{L}(G, D) = \mathcal{L}_{G A N}(G, D) + \lambda \mathcal{L}_{L1}(G) + \beta \mathcal{L}_{R}(G)
\]

where \(\lambda\) and \(\beta\) are weights of \(L1\) and \(R\) regional losses (in our experiments \(\beta > \lambda\)). and the goal is to solve the following optimization problem.

\[
G^* = \arg \min_G \max_D \mathcal{L}(G, D)
\]

**Multi-person Video Generation.** In a nutshell, our algorithm merges transformed images of a person on an arbitrary pose with an arbitrary background in a natural photorealistic way. We managed to go beyond simple one per-
son human action videos and extended our method to multi-
person interaction videos as well. For this purpose, we
trained our model on a two person interaction dataset [69].
The only difference with single frame generation process
is that in the pre-processing phase, for each person in the
input reference image, we needed to know the correspond-
ing skeleton in the target frame, we then transformed each
person’s body parts to his/her own body parts in the target
skeleton. There are some challenges in this task such as oc-
cclusions in certain interactions (e.g. passing by, hugging,
etc.). The dataset that we used contains these occlusions
to some extent. Our method is able to handle relatively
well some simple occlusions that occur in such interactions.
We acknowledge that there is room for improvement in this
area, but that would not fit in the scope of this work. Fig. 4b
illustrates some of the generated videos.

3.2. Skeleton Trajectory Generation

In the previous section, we explained how we designed
a method that enables us to generate videos of an arbitrary
person in any background based on any given sequence of
skeletons. Although number of backgrounds and persons
are unlimited, the number of labeled skeleton sequences are
limited to the ones in the existing datasets. We propose a
novel solution to this problem: using a generative model
to learn the distribution of skeleton sequences conditioned
on the action labels. This allows us to generate as many
skeleton sequences as needed for the actions in the dataset.
Fig. 6 shows a few sample generated skeleton sequences.

We used small datasets for training our model. However,
due to the nature of the problem and the limited amount of
data, generating long sequences of natural looking skele-
tons proved challenging. Thus we aimed at generating rela-
tively short fixed-length sequences. Having said that, train-
ing GAN in such way is still prone to problems such as
mode collapse, divergence, etc. In designing the genera-
tor and discriminator networks, we have taken into account
these problems (e.g. introduced batch diversity in the dis-
criminator, created multiple discriminators, etc.).

Skeleton Trajectory Representation. Each skeleton
consists of 18 joints. We represented each skeleton with a
1 × 36 vector (a flattened version of 18 × 2 matrix of joints
coordinates). We normalized the coordinates by dividing
them by “height” and “width” of the original image.

Generator Network. We used a conditional GAN
model to generate sequences of skeletal positions cor-
responding to different actions. Our generator has a “U”
shape architecture where input consists of action label
and noise, and output is a 8 × 1 × 36 tensor representing a human
skeleton trajectory with 8 time-steps.

Based on our results, providing a vector of random noise
for each time step helps the generator to learn and gener-
alize better. So the input noise, z, is a tensor with size

(b) Trajectory Discriminator Network. The discriminator is the
sum of three discriminators illustrated in this figure: $D = D_f +
D_t + D_j$.

Figure 5: Trajectory GAN network architecture.

8 × 1 × 128; drawn from a uniform distribution. The one-hot
encoding of action label, l, is replicated and concatenated to
the 3rd dimension of the z. The rest is a “U” shaped network
with skip connections that maps the input $(z, l)$ to a skeleton
sequence S. Fig. 5a illustrates the network architecture. We
also used Dense-net [13] blocks in our network.

Discriminator Network. Architecture of discrimina-
tor is three-fold. The base for discriminator is 1D convolu-
tional neural net along the time dimension. In order to
allow discriminator to distinguish “human”-looking skele-
tons, we used sigmoid layer on top of fully-convolutional
net. To discriminate “trajectory”, we used set of convolu-
tions along the time with stride 2, shrinking output to one
1 × 1 × C containing features of the whole sequence. To pre-
vent mode collapse, first we grouped fully convolutional net
outputs across batch dimension,We then used min, max and
mean operations across batch, and provided these statistical
information to the discriminator. This method seems to pro-
vide enough information about distribution of values across
batch and allows to change batch size during training. For
detailed discriminator architecture see Fig. 5b.

Our objective function is:

$$
\mathcal{L}_T (G, D) = \mathbb{E}_{l,s \sim P_{data}(l,s)}[\log D(l, s)] + \mathbb{E}_{l \sim P_{data}(l), z \sim P_z(z)}[1 - \log D(l, G(l, z))]
$$

where l and s are action label and skeleton trajectories,
respectively. We aim to solve the following:

$$
G^* = \arg \min_G \max_D \mathcal{L}_T (G, D)
$$

In this work, we have shown that generative models can
be adopted to learn human skeleton trajectories. We trained
a Conditional GAN on a very small dataset (200 sequences) and managed to generate natural looking skeleton trajectories conditioned on action labels. This can be used to generate a variety of human action sequences that don’t exist in the dataset. However, our work is limited to a fixed number of frames. Thus for future work, we’ll work to improve our method so that it’ll accommodate longer sequences varying in length. We also explained that in addition to the generated skeletons, we can also use real skeleton sequences from other sources (other datasets, current dataset but different subjects) to largely expand existing datasets.

4. Datasets and Action Recognition Methods

4.1. Datasets

In this paper, we’ve claimed to expand small amount of action videos by addition of new generated videos. We targeted smaller action recognition datasets and expanded them to meet the large data load requirements of recent action recognition algorithms such as UCF 101 [51], the kinetics [21] or NTU RGB+D [48]. This eliminates the need for time and cost inefficient data acquisition processes.

UT Kinects [65]: One of the datasets wildly used in our experiments is UT Kinects which includes 10 action labels: Walk, Sit-down, Stand-up, Trow, Push, Pull, Wave-hand, Carry and Clap-hand. There are 10 subjects that perform each of these action twice in front of a rig of RGB camera and Kinect. Therefore in total they are 200 action clips of RGB and depth though depth is ignored. All videos are taken in office environment with similar lighting condition and the position of the camera is fixed.

For the training setup, 2 random subjects were left out (20%, used for testing) and the experiments were carried out using 80% of the subjects. The reported results are the average of six individual runs. The 6 train/test runs are constant throughout our experiment.

SUB Interact [69]: Since our methods work with multiple human subjects in a scene, we picked SUB Interact. It is a kinect captured human activity recognition dataset depicting two person interaction. It contains 294 sequences of 8 classes (Kicking, Punching, Hugging, Shaking-hand, Approaching, departing and Exchanging objects) with subject independent 5-fold cross validation. The original data includes RGB, depth and skeleton but we only use RGB for our purpose. We used a 5-fold cross validation throughout our experiments and reported the average accuracy.

KTH [46]: KTH action recognition dataset was commonly used at the early stage of action recognition. It includes 600 low resolution clips of 6 actions: Walk, Wave-hand, Clap-hand, Jogging, running and boxing which are divided in train, test and validation. The first three action labels are shared with UT dataset while the last three are new. We used this dataset to add new action to UT dataset and for cross dataset evaluation.

4.2. Action Recognition Methods

We used the following deep learning networks which have previously shown decent performance on recent action recognition datasets.

Convolutional 3D (C3D) [54]: is a simple and efficient 3-dimensional ConvNet for spatiotemporal feature which shows decent performance on video processing benchmarks such as action recognition in conjunction with large amount of training data. We used their proposed network with 8 convolutional layers, 5 pooling layers and 2 fully connected layers with 16-frames of $112 \times 112$ RGB input. They released a network pre-trained on UCF Sport [51] which we used for our experiments aimed at training from scratch, denoted as C3D(p) vs. C3D(s). Unfortunately we can not couldn’t converge the C3D when we trained from scratch on UT dataset but it converged successfully on SUB.

Inflated 3D ConvNets (I3D) [4]: is a more complex model which has recently been proposed as the state-of-the-art for action recognition task. It builds upon Inception-v1 [14], but inflates their filters and pooling kernels into 3D. It is a two-steam network which uses both RGB and optical flow input with $224 \times 224$ inputs. We only used RGB for simplicity. They released a network pre-trained on ImageNet [6] followed by the Kinetics [21]. We used this for our experiments aimed at training from scratch, denoted as I3D(p) vs. I3D(s).

We use data augmentation by translation and clipping as mentioned in [4] for all experiments. For training, we only used the original clips as test, making sure there was no generated clips with skeletons or subjects (subject pair) from test data in each run.

5. Experiments

So far, we have introduced our video generation method which enable us to generate new action clips for the action
recognition training process. In this section, we show different scenarios for generating new data and running experiments for each to see if adding the generated data to a training process can improve the accuracy of the action recognizer. We applied our proposed video generation models to all the experiments using skeletons. The skeletons were trained using data from UT and SBU datasets as well as 41 un-annotated clips (between 10 to 30 seconds) that we captured from our colleagues. For future works, we will train our model again using a large amount of data from web. But the time being, we are satisfied with the current model as higher resolution for action recognition is currently unnecessary. Our technique for generating new action video clips has the capacity of running experiments with numerous varying settings. Here, we show five experiments which may be quantitatively evaluated.

5.1. Generated Trajectory

The first experiments is a combination of our proposed video generation technique and skeleton trajectory generation. We generated around 200 random skeleton trajectories from action labels in UT dataset using the method mentioned in §3.2. Each of these skeleton trajectories generated a video by proposed video generation applied to a person in UT dataset, meaning our new dataset is doubled with half of it being the generated data. We then trained our model by I3D and C3D using training setting mentioned in §4.1. Table 1 shows about 3% improvement for I3D with and without training data as well as significant improvement (by 15%) for C3D network which is less complex.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I3D(s)</td>
<td>64.58%</td>
<td>67.50%</td>
</tr>
<tr>
<td>I3D(p)</td>
<td>86.25%</td>
<td>89.17%</td>
</tr>
<tr>
<td>C3D(p)</td>
<td>55.83%</td>
<td>70.83%</td>
</tr>
</tbody>
</table>

Table 1: Action recognition on UT dataset using original data compared to generated from scratch data with proposed method in §3.1 and §3.2

5.2. New Subjects

One common way to extend a video dataset is to invite new people to do a series of actions in front of a camera. Diversity [2] in body shape, cloths and behaviour will clearly help with the generalization of the ML methods. In this experiment, we aimed to virtually add new subject to the dataset. Thus, we collected a small unannotated clips from 10 distinct persons and fed them as new subjects into our proposed video generation method. For UT, each subject was replaced by a new one for all of his/her action which is similar to adding 10 new subjects to UT. The same was done with SUB to double the dataset, the only difference being the replacement each pair with a new subject pair. Figure 4b shows a few new subjects with their generated action videos from SBU dataset. The results have been presented in Table 2.

<table>
<thead>
<tr>
<th></th>
<th>UT</th>
<th>SBU</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3D(s)</td>
<td>64.58%</td>
<td>67.08%</td>
</tr>
<tr>
<td>I3D(p)</td>
<td>86.25%</td>
<td>89.17%</td>
</tr>
<tr>
<td>C3D(s)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C3D(p)</td>
<td>55.83%</td>
<td>70.43%</td>
</tr>
</tbody>
</table>

Table 2: Performance comparison of multiple algorithms, trained on original data and additional subjects.

5.3. New Actions

In real computer vision problems, one might decide to add a new label class after the data collection process has been done. Adding a new label action to a valid dataset could cost the same as gathering a dataset from scratch as all the subjects are needed for re-acting that single action. As mentioned in §4.1, UT consists 10 action labels. In this experiment, we try to introduce new actions (i.e. running, jogging, and boxing) to UT dataset, which do not already exist. We used the skeleton data, which are extracted by OpenPose [3], from the training set of a third dataset, KTH [46]. We randomly picked 5 clips from each of these 3 actions and used all the subjects of UT to generate 150 new video clips. We then trained a new model using a pre-trained I3D network on the union of the original training data of UT and the newly generated data (150 clips). Since the KTH data is grey scaled images, we randomly grey scaled both the original and the generated training clips in the training phase. For each run, we found per class accuracy for UT test set (refer to §4.1 for explaining UT train/test) as well as KTH test sets. Table 3 shows average of the per class accuracy for both test sets. We may consider KTH test results
as a measure of cross dataset accuracy for walk, wave-hand and clap-hand. Our trained network on new action labels boxing, running and jogging achieved 72.14%, 44.44% and 63.20%, respectively. This indicates that the new actions in the dataset performed as good as the data captured by camera.

Table 3: Per class average accuracy for model trained by i3d using original training data from UT plus new action clip generated by our method using skeleton extracted from KTH training set.

<table>
<thead>
<tr>
<th>Action</th>
<th>UTK Test</th>
<th>Label</th>
<th>KTH Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>91.67%</td>
<td>Walk</td>
<td>67.18%</td>
</tr>
<tr>
<td>Wave-hand</td>
<td>100.00%</td>
<td>Wave-hand</td>
<td>58.59%</td>
</tr>
<tr>
<td>Clap-hand</td>
<td>91.67%</td>
<td>Clap-hand</td>
<td>28.90%</td>
</tr>
<tr>
<td>Push</td>
<td>33.33%</td>
<td>Boxing</td>
<td>72.14%</td>
</tr>
<tr>
<td>Pull</td>
<td>58.33%</td>
<td>Running</td>
<td>44.44%</td>
</tr>
<tr>
<td>Pick-up</td>
<td>100.00%</td>
<td>Jogging</td>
<td>63.20%</td>
</tr>
<tr>
<td>Sit-down</td>
<td>87.50%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stand-up</td>
<td>95.83%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Throw</td>
<td>54.17%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carry</td>
<td>79.17%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: The comparison of dataset expansion by original data for UTK and SUB dataset.

<table>
<thead>
<tr>
<th></th>
<th>UTK</th>
<th>SBU</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3d(s)</td>
<td>64.58%</td>
<td>69.58%</td>
</tr>
<tr>
<td>i3d(p)</td>
<td>86.25%</td>
<td>90.42%</td>
</tr>
<tr>
<td>c3d(s)</td>
<td>86.48%</td>
<td>93.54%</td>
</tr>
<tr>
<td>c3d(p)</td>
<td>96.83%</td>
<td>69.58%</td>
</tr>
</tbody>
</table>

5.5. Real World

In this section, we carried out 4 different experiments on 2 datasets for benchmarking. Although in all experiments, the generated data improved the network performance, we believe none of the experiments show the actual strength and convenience of our proposed methods in real world scenarios. In both datasets, as well as other commonly used small datasets, the environmental setup for data acquisition such as distance from camera view [17] and light condition were kept as uniformly as possible for both test and train video clips. This would be unattainable in real life data acquisitions. A way of overcoming this obstacle would be to collect diverse sets of data for strong neural network models. We’ve previously shown that by partitioning the video to action, subject and context allows us to easily manipulate the background or change the camera view. In this experiment, we applied perspective transform on skeleton while using diverse backgrounds. Although the model trained with these data did not outperform our previous experiments, a live demo showed it to be better for unseen cases, qualitatively. Figure 8 illustrates an input skeleton and its perspective transform as well as the generated clip.

6. Conclusion and Future Works

In this paper, we’ve introduced a novel way to partition an action video clip into action, subject and context. We showed that we can manipulate each part separately, reassemble them with our proposed video generation model into new clips and use as an input for action recognition models which require large data. We can change an action by extracting it from an arbitrary video clip, generate it through our proposed skeleton trajectory model or by applying perspective transform on existing skeleton. Additionally, we can change the subject and the context using arbitrary video clips.

For the future work, we will replace our 2d skeleton with 3d skeleton to achieve a 3d transformation and handle occlusions. Additionally, while our video generation technique demonstrated acceptable results for $255 \times 255$ images, we believe it can be extended even further to achieve higher resolution by feeding more unannotated data.
References


