Multimedia Tools and Applications (2021) 80:12293–12308
https://doi.org/10.1007/s11042-020-10274-0

GPU-based parallel Shadow Features generation at neural system for improving gait human activity recognition

Ricardo Brito[1] · **Robert P. Biuk-Aghai**[1] · **Simon Fong**[1]

Received: 24 March 2020 / Revised: 12 October 2020 / Accepted: 9 December 2020 /
Published online: 9 January 2021

Abstract
In this paper, we propose a new method for improving human activity recognition (HAR) datasets in order to increase their classification accuracy when trained with a certain classifier like a Neural Network. In this paper a novel training/testing process for building/testing a classification model for human activity recognition (HAR) is proposed. Traditionally, HAR is done by a classifier that learns what activities a person is doing by training with skeletal data obtained from a motion sensor such as Microsoft Kinect or accelerometer sensors. These skeletal data are the spatial coordinates (x, y, z) of different parts of the human body. In addition to the spatial features that describe current positions in the skeletal data, new features called Shadow Features are used to improve the supervised learning efficiency and accuracy of Neural Network classifiers. Shadow Features are inferred from the dynamics of body movements, thereby modelling the underlying momentum of the performed activities. They provide extra dimensions of information for characterizing activities in the classification process and thus significantly improving the accuracy. These Shadow Features are generated based on the existing features obtained from sensor datasets. In this paper we show that the accuracy of a neural network classifier can be significantly improved by the addition of Shadow Features and we also show that the generation of Shadow Features can be achieved with little time cost, on the fly, with the NVIDIA GPU technology and the CUDA programming model, this way we can improve the Neural Network accuracy at almost no time cost. GPUs are particularly suitable for generating Shadow Features, since they possess multiple cores which can be taken advantage of, in order to generate Shadow Features for multiple data columns in parallel, therefore reducing a lot of processing time, especially when dealing with huge datasets.

Keywords Neural networks · Shadow features · CUDA · GPU · Human Activity Recognition (HAR)

✉ Ricardo Brito
yb87473@um.edu.mo

Extended author information available on the last page of the article.

1 Introduction

Human activity recognition (HAR) [3, 15, 31, 32] is a branch of machine learning that trains a classification model using historical activity records to recognize unseen activities. HAR has wide application, ranging from abnormal behaviour detection in security surveillance [21] and posture monitoring in tele-rehabilitation [7], to hand gesture recognition in augmented-reality [30] and virtual-reality [18] systems, as well as intelligent home environments [14]. One type of HAR is based on motion sensor data from which the HAR system tries to infer activity patterns for prediction and classification. The sensor data arrives in the form of continuous sequential values, similar to a time series. Time series such as those collected from an accelerometer are usually multivariate, comprising tri-axial information, known in its simplest form as three features (x, y, z). These features plus the timestamp information represent the temporal-spatial displacement of a body part in motion. More complicated time series may contain extra information, such as velocity, acceleration, rotational angle and relative distance to a reference point. In this paper, a new concept of feature is presented, which is called **Shadow Features**, with the goal of improving existing HAR datasets, and we show that these features can be computed and appended to an existing dataset on the fly, with very small time cost by using NVIDIA GPUs and the CUDA programming model to generate such features. Then we use a feed forward Neural Network to perform the activity recognition task. Neural Networks have been used in numerous occasions in literature [10] in many tasks other that HAR very good results, therefore in this paper we decided to use a Neural Network as the final classifier for the HAR task. It is also very common to use the GPU to accelerate data processing tasks due to its parallel computing capabilities, as was shown in [4] in which the authors take advantage of the GPU to reduce their model training time through the simultaneous analysis of multiple inputs. In our proposed work we also take advantage of the parallel processing capabilities of the GPU in order to accelerate the shadow features generation process as well as the Neural Network training and prediction times.

1.1 Related work

The concept of shadow features is inspired by Etienne-Jules Marey (1839-1904), who instigated chronophotography or cinematographic movement recording [25] as a fundamental technique of capturing motion pictures. His lifetime passion to see the invisible motivated him towards a single goal: recording all existing movement produced by humans or anything else, on a recordable surface. In alignment with Etienne-Jules motivation, Shadow Features are derived from the dynamics of human body movements. The Shadow Features' values are subtly approximated by the underlying motion resulting from performing activities. Unlike other feature transformation techniques that have been previously reported in literature, the Shadow Features don't totally replace or transform the existing dataset set features, but it creates a new set of motion dynamic data that augments the original spatial data. Hence the term shadow. These features basically offer extra dimensions of information for characterizing activities in the HAR process. The extra information gives more insight to the classifier during the supervised learning process as the classifier tries to map the dataset to the classification targets. With these Shadow Features, the classifier can better learn to recognize the activities since the classifier is provided with the motion information as well as the spatial coordinate data. This can be seen as a complement to the existing features in order to allow the classifier to better infer the relationship between the dataset and the target class. In HAR, features that can be extracted from raw sensor data are loosely

categorized into remote features and local features. Remote features are data from a sensor at a fixed position (such as a video camera, Kinect motion sensor and ultrasound sensor) that captures images of a moving subject from some distance away. Local features are data collected from wearable sensors (for example), which 'tag along' with the subject on the go. Traditionally, discrete Fourier transformation (DFT) [23] is applied to a sequence of images to convert the image intensity variation spatially from the temporal domain to the frequency domain. DFT extracts the information contained in the full picture into numeric representations known as wavelets. Therefore, it is prone to quality issues arising from noise, occlusion and variation to viewpoint. To rectify these shortcomings, researchers turned from the full-picture information to cuboids of spatial-temporal shapes in the foreground. Some [5] have even attempted to extract spatial-temporal features, shape structures, shape orientation and salient outlines of the shape by stacking up silhouettes frame by frame, but this was found to be quite inefficient. Researchers improved the shaped-based matching mechanism by using region matching and extracting certain regions [17]. Keeping the appearance of the 3D objects or the region consistent is difficult. Other researchers suggested a flow based feature model [27] and spatial-temporal interest point detector model [9], which only process the flow and point of interest in the local region, respectively. Local features are obtained relatively directly from tracking the pose and coordinates of the body. Different from remote features, local features are more focused on the movements of the interest points without regard to the background or the full picture. To represent local features, spatial information is often refined from the raw tracking data, like GPS coordinates, using spatial techniques such as Boolean features [24], polar coordinate features [6] and geometric relative features [12, 13], to effectively preserve the spatial information that discriminates different body activities. Although these local features are obtained almost directly from the sensing data, many variants derived from different transformation techniques have been proposed by researchers in the hope of better capturing the essence of the spatial information in relation to activity recognition. Refining and abstracting these techniques with the aim of generating better local descriptors include but are not limited to wire-frame or shape-based features [1, 8, 33].

Recent work in HAR consists of extracting images from videos [2, 16], which would then be pre-processed such that the background frame would be detected and then removed (background subtraction), then the human in the video would be detected, features would be extracted and finally a Neural Network would perform the activity recognition. However this method has a few drawbacks: First, this method is intrusive, there is no privacy for the user due to the usage of video images, as has been already shown in [26] privacy is of major importance when dealing with applications of the Internet of Things (IOT), second this method has a lot of pre-processing steps which make it slow. Other works have proposed the usage of more complex Neural Network Architectures such as Recurrent Neutral Networks [28, 29] and Convolutional Neural Networks [19, 20, 22] with the goal of recognising three activities using accelerometer data [20]. However, Convolutional Neural Networks require a lot of data to train and their best result was only 92.71% accuracy. This is due to the limited number of features provided by the smartphone accelerometer (only three features x, y and z). The dataset from most sensors used in HAR is not 2D or 3D, therefore the usage of such complex architectures is overkill for the HAR task. In this paper we argue that image extraction from surveillance videos is not the most efficient way of performing HAR, and it also brings a lot of privacy concerns to the user, therefore we propose an approach using Kinect sensors which do not capture images, but instead capture coordinates which are used as features in the HAR task. In order to further enhance the prediction quality of

the HAR system, the features provided by Kinect Sensors are further augmented with the Shadow Features proposed in this paper. The idea of using Kinect sensor for gesture and activity recognition has also been proposed in [11], but in this paper we take another step forward in the HAR field by proposing a novel algorithm for HAR dataset enhancement, Shadow Features which improve the quality of the HAR dataset and therefore provide more information to the classifier, which in this case is a simple three-layered feed forward Neural Network, much lighter and simpler to train than the Convolutional and Recurrent Neural Networks presented in previous works. The contributions of this paper are the following:

1. A novel algorithm for HAR dataset improvement is proposed, which augments a HAR dataset with extra dimensions of information which contribute greatly to the improvement of the classifier's performance.
2. We demonstrate how the Shadow Features can be generated on the GPU such that HAR datasets can be augmented on the fly, with almost no time delay, making our proposed Shadow Features practical for usage in real life.

This paper is organised as follows: In Section 2, we provide an overview of the proposed classification model, in Section 3 we explain all the details of the proposed Shadow Features generation algorithm, as well as its implementation on the CPU and GPU. In Section 4 we show results which showcase the advantage of using shadow features in HAR, and we also show a speed comparison between the CPU and the GPU. Finally, in Section 5 We conclude this work.

2 Proposed classification model overview

A new classification model is described in this section. It extends the traditional HAR system model in which the raw activity data is pre-processed prior to either training or testing the classifier. Figure 1 presents an overview of the HAR system embracing our new Shadow Feature generation method.

The new HAR model comprises an activity recording module, the Shadow Feature generation module and a classification module. The activity recording model and the classification module are generic. They can be implemented by any suitable software/hardware and algorithms. The recorder can be a kinect sensor or multiple kinect sensors of any kind of sensor, as long as they can capture activities and produce the corresponding activity dataset as time-stamped ordered sequences. The general format of the activity dataset is a matrix with rows of ordered instances (e.g., each row contains information about the video frame per sampling interval) and the columns of attributes or features that describe the information. The training dataset has a labelled column that already contains the type of activity corresponding to each particular row of instance. The training dataset that we use to train or

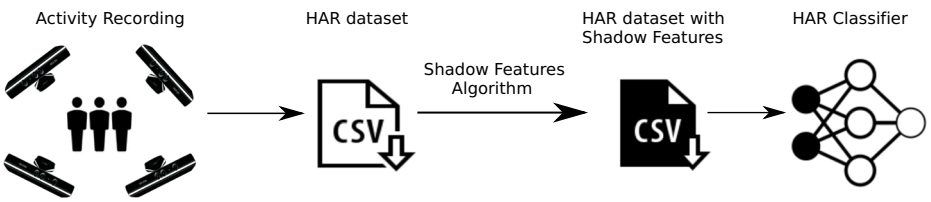

Fig. 1 HAR system with Shadow Features generation method

build a classifier after passing through the Shadow Feature generation process extends the original feature set of the training data with Shadow Features. In the testing phase, an unlabelled dataset us subject to a Shadow Feature generation process configured by the same parameters that controlled the Shadow Feature generation process for the training dataset is generated. Then the extended testing dataset with the corresponding Shadow Features is tested against the built classifier, for activity recognition and the predicted result is outputted from the classifier.

In incremental learning, which is also known as data stream mining, the training dataset streams continuously into the classifier construction process. The dataset passes through the Shadow Feature generation process as through a pipeline, with new features continuously being generated using a sliding window over the input data stream. Likewise the testing dataset streams into the classifier at any time, the corresponding Shadow Features are added on the fly, and the results are predicted continuously at the end of the pipeline as streaming progresses. What allows the Shadow Features to be added on the fly with no delay is the usage of the GPU in the generation process. More details are given in the next section.

3 Shadow Features generation

Given a HAR dataset with R rows and C columns, for each column, the Shadow Features are generated in the following way:

1. Choose a sliding window of size Q. This sliding window will slide through each column's Q elements with a stride value of 1, and it will slide though every column R times (R is the number of rows, which represents the length of the columns).

2. Every time the window slides through a certain column, it takes the average of Q + 1 elements (actually it can be any statistical operation, as it will be shown later in the experiments section) in its current location and places the result in the entry of the new features column (shadow features column), but for the first Q positions in the Shadow Features column, the values are all zeroes. This can be seen in Fig. 2, where we have a column with data and we choose a sliding window of size Q = 3.

This process is repeated until the number of generated Shadow Features matches the length of the column. So for each column of size R, R Shadow Features are generated.

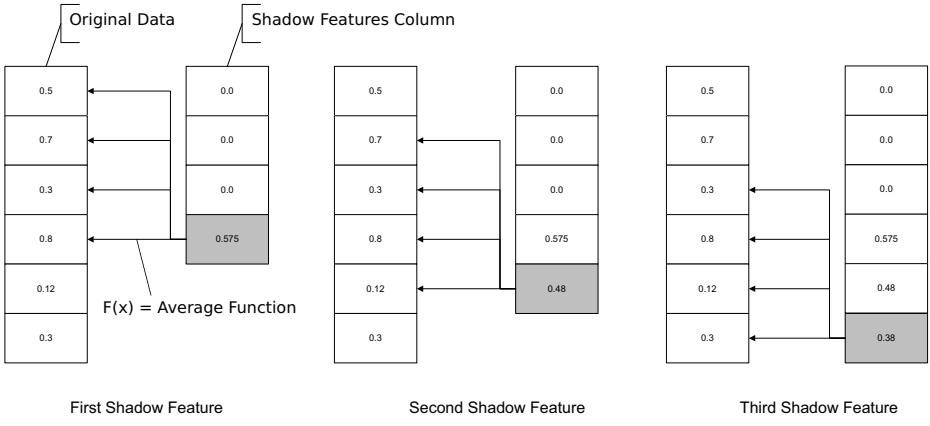

Fig. 2 Shadow Features Generation Process

3. The same process is repeated for every column until all the Shadow Features have been calculated for the entire dataset. This means that the Number of columns C, will double with the addition of the shadow features, thus becoming $2 \times C$.

4. Now, as a last step we combine the Shadow Features with the original data set in an alternate fashion, such that for each column in the original dataset, its corresponding Shadow Features column is right next to it, as demonstrated in the Fig. 3, in which the highlighted columns represent the Shadow Features calculated from their respective preceding columns.

3.1 Shadow Features generation (CPU versus GPU)

In this section we will compare the Shadow Feature generation process on the CPU and the GPU as well as explain how we get a real time gain by generating the Shadow Features on the GPU.

3.1.1 Shadow Features CPU generation

The Shadow Feature generation on the CPU is a serial process, where for each column we take the average of the elements on every Q window as we slide through the column and for each column we slide Q times, doing Q average operations. So assuming that there are R rows and C columns in total, then since this is a serial process we would have a running time of O(C * R * (Q+1)). The pseudo code for the shadow features generation are shown in Algorithm 1.

Algorithm 1 Shadow Features Generation on the CPU.

```
sourceData = transpose(sourceData)
shadowFeatures = emptyList
for row = 1 to maximum number of rows in dataset do
    for i = 1 to maximum in sourceData[row][i:q+i+1] do
        j = q + 1
        if if j ¡ length of sourceData[row] then
            mean = average(sourceData[row][i:q+i+1])
            shadowFeatures[row][q+1] = mean
        end if
    end for
end for
sourceData = transpose(sourceData)
combine the shadow features with the source data
```

As we can see from the pseudo code above, for the Shadow Features generation process on the CPU we have 3 loops. One loop to go column wise, one to go row wise and another one for the average which is omitted from the pseudo code for simplicity, thus we have the running time of O(C * R * (Q+1)) as stated above.

3.1.2 Shadow Features GPU generation

The Shadow Features generation on the GPU is a complete parallel process, and it requires some understanding on how the GPU works in order to get a clear view of it. The GPU contains an array of streaming multiprocessors that allows thousands and even millions of

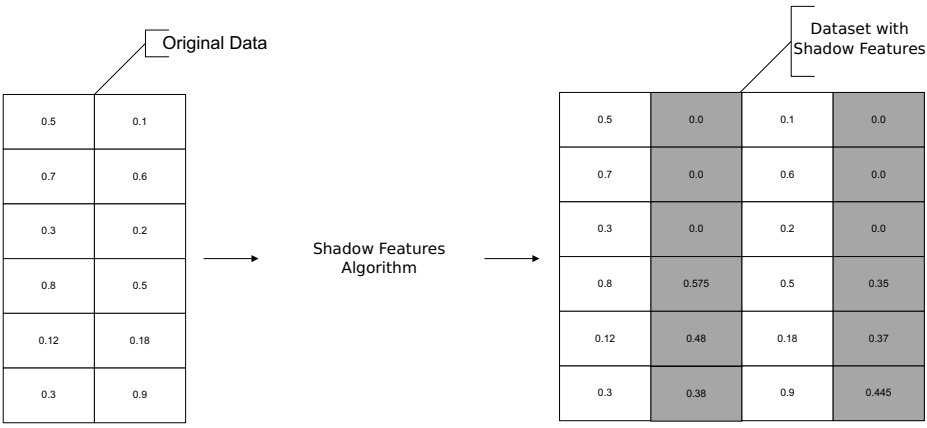

Fig. 3 Resulting dataset with shadow features

threads to be executed in parallel. The GPU organizes threads in blocks, where threads inside the same block can communicate and share resources such as registers, shared memory and others. Blocks of threads are scheduled for executing in the GPU's streaming multiprocessors and these blocks are executed in parallel. Generating the shadow features on the GPU makes intensive use of the thread/block model, allowing all the Shadow Features to be generated in parallel, at the same time, drastically reducing the time cost.

The process to generate the Shadow Features on the GPU is shown in Fig. 4. Suppose that once again the Q value is 3.

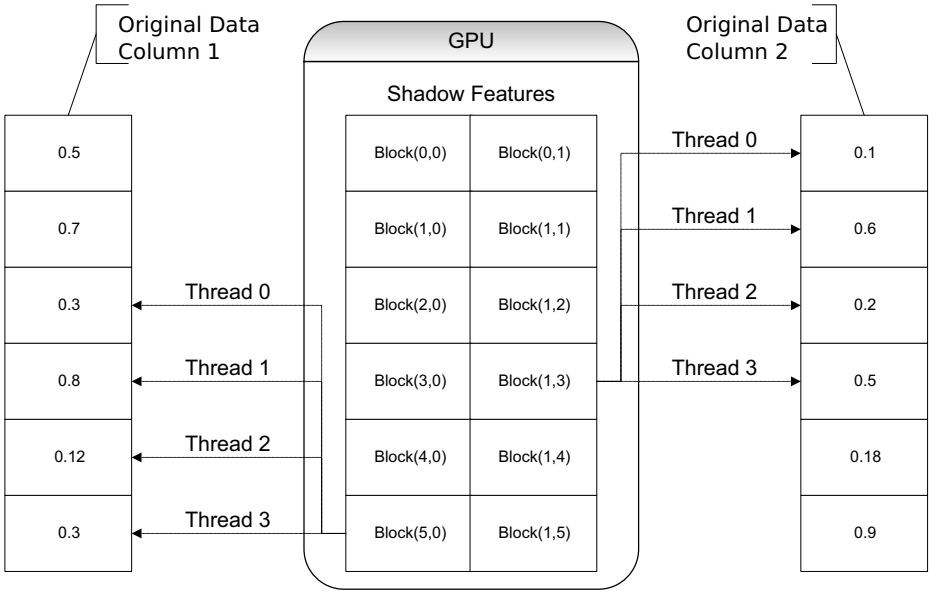

Fig. 4 Shadow Features generation on the GPU

Each block will calculate exactly one feature, so the number of blocks launched must be equal to the number of columns times the number of rows and inside each block there will be exactly Q + 1 threads such that each thread can contribute to calculating the sum of the Q + 1 elements in the sliding window in parallel and then divide this sum by Q + 1 in order to get the average. The average is calculated by a technique called parallel reduction where in each iteration, the threads inside a block work in parallel in order to sum the Q + 1 elements in the sliding window. This process is illustrated in Fig. 5 below, which illustrates a case where we use parallel reduction to sum an array of 16 elements. The threads work together to calculate the sum in parallel and store the result inside the array itself in the first entry. After the sum is calculated, it will be divided by Q + 1 in order to get the average. This same process happens at the same time inside every block which is calculating every Shadow Feature in parallel. So since all the columns are calculated in parallel by the blocks of threads and inside each block we have to do a parallel reduction, then the running time of this algorithm is equal to the running time of the parallel reduction algorithm which is O(log(Q+1)) due to the fact that the length of the array to be summed together is cut by half in each iteration.

In the experiments section we show a comparison between the running times of the Shadow Feature generation process in the CPU and the GPU. The source code for the GPU Shadow Features generation process in the listing below

The LENGTH_OF_COLUMNS refers to the number of instances in the dataset, it is used in order to caluclate the Shadow Features column-wise. The blockIdx.x/y is used to index GPU blocks of threads and the threadIdx.x is used to index GPU threads inside blocks of threads. Line 21 - 32 show the parallel reduction sum that is illustrated in Fig. 5. For the Shadow Feature generation process we launch a grid of 2D blocks of threads in which each block of threads calculates the Shadow Features for a number of rows (specified by q_size) in a column, this way we can have each block of threads calculating all Shadow Features in parallel.

4 Experiments

Two sets of experiments were designed and conducted to verify the efficacy of the proposed Shadow Features algorithm and HAR classification model. In the first set of experiments we compare the performance of the HAR process with and without shadow features, and also compare the performance of different statistical methods to generate Shadow Features. In the second set of experiments we show the efficiency of using the GPU to generate the Shadow Features and compare the running time with the CPU, with the goal of proving that by using the GPU our proposed algorithm can be used on the fly in real world applications. The classifier used in these experiments is a light and simple Feed Forward Neural Network composed of three layers: input, hidden and output. The experiments were run on a Linux Computer running UBUNTU with 2 NVIDIA GEFORCE RTX 2080 Ti GPUs, 8 CPUs, 32 GB of RAM memory and 4 TB of disk space.

4.1 Data set and experiments set up

The dataset used in these experiments was collected using a Microsoft Kinect sensor. The dataset contains a target class called “Activity” with information labels which describe what the human activities are with respect to the instances of multi-dimensional data. The collected kinect data has a total of 63 features and 16200 instances that describe 6 different

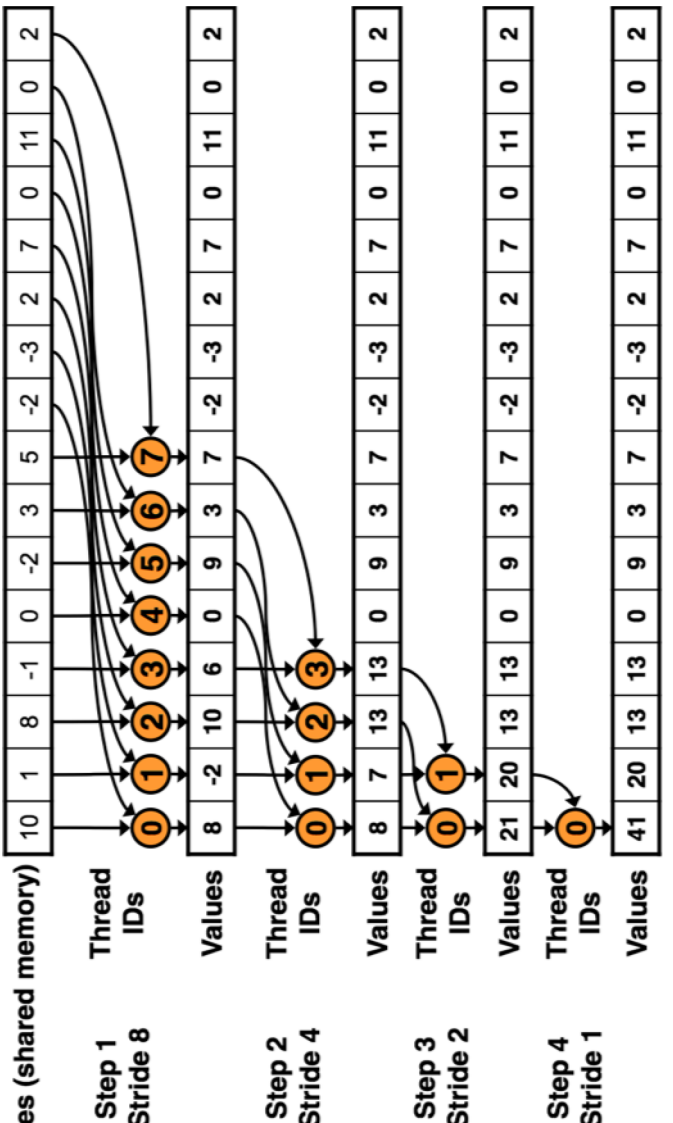

Fig. 5 Parallel reduction sum

```
__global__ void shadow_features(float *source, float *target, int q_size
    ){

  extern __shared__ float temp[];
  int source_index2 = (blockIdx.y*LENGTH_OF_COLUMNS)+(blockIdx.x -
      q_size);

  if(blockIdx.x < q_size){

    target[blockIdx.y * LENGTH_OF_COLUMNS + blockIdx.x] = 0.0;

  }

  else if(blockIdx.x >= q_size && q_size > 1000){

    temp[threadIdx.x] = 0;
    int factor = q_size / 1000;
    for(int i = 0; i < factor; i++)
      temp[threadIdx.x] += source[source_index2 + (i*1000) + threadIdx.x
          ];

    __syncthreads();

    for(int stride = 1; stride < blockDim.x; stride *=2){

      int index = 2 * stride * threadIdx.x;

      if(index < blockDim.x && index + stride < blockDim.x){

        temp[index] += temp[index + stride];

      }

      __syncthreads();
    }

    if(threadIdx.x == 0){
      target[blockIdx.y * LENGTH_OF_COLUMNS + blockIdx.x] = temp[0] / (
          q_size+1);
    }

  }

    else if (q_size <= 1000 && blockIdx.x >= q_size){

      int source_index = (blockIdx.y*LENGTH_OF_COLUMNS)+(blockIdx.x -
          q_size) + threadIdx.x;
      temp[threadIdx.x] = source[source_index];

      __syncthreads();

      for(int stride = 1; stride < blockDim.x; stride *=2){

        int index = 2 * stride * threadIdx.x;

        if(index < blockDim.x && index + stride < blockDim.x){

          temp[index] += temp[index + stride];

        }

        __syncthreads();

      }

      if(threadIdx.x == 0){
        target[blockIdx.y * LENGTH_OF_COLUMNS + blockIdx.x] = temp[0] /
            (q_size+1);

      }
    }

}
```

Listing 1 Shadow Features generation on the GPU (source code)

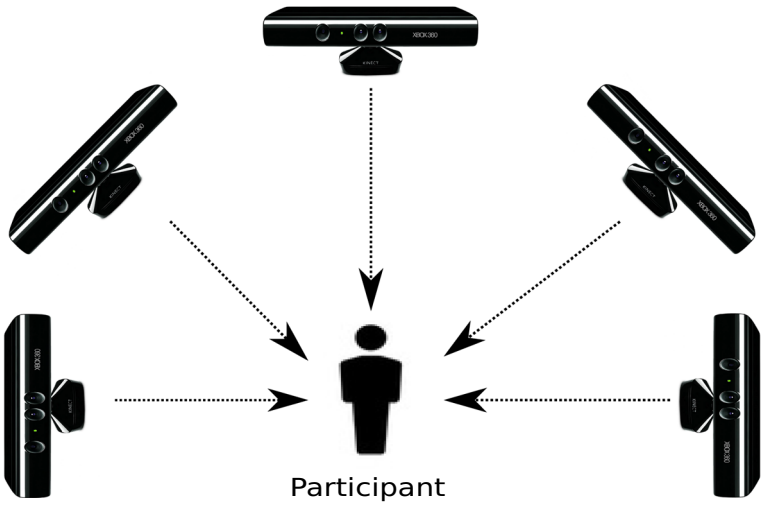

Fig. 6 Placement of Kinect sensors in our experiment

human activities. The features in the dataset are x-y-z spatial information about various parts of the human body such as the head, shoulder, elbow, wrist, hand, hip, knee, and foot. The data collection process consisted of a volunteer performing in front of five kinect sensors placed in different positions doing the 6 different postures. The Kinect sensor was placed at five different positions facing the subject at 0, 45, 90, 135, and 180 degrees. The five positions of the Kinect sensors are shown in Fig. 6. After the application of the Shadow Features algorithm to the dataset obtained from the kinect sensors, the features doubled from 63 to 126. The Shadow Features were applied using different statistical methods (chi-square, mean, root mean, standard deviation, variance and Hurst exponent) with the goal of figuring out which method is the most suitable in the Shadow Features generation process.

The classifier used in our experiments consists of a simple Feed Forward Neural Network with the configuration as shown in Table 1. In our experiments we also show the result of the HAR process without the application of Shadow Features so that we can show the advantages of using Shadow Features in HAR datasets.

The experiments were conducted with different Q sizes. For each Q size, we provide a comparison on the maximum accuracy attained by a neural network training/testing the Kinect dataset with Shadow Features generated by different statistical operations which are

Table 1 Neural network parameters

Number of input Neurons	63 without Shadow Features, 126 with shadow features
Hidden layer size	256
Output Layer size	6
Number of Epochs	128
Learning Rate	0.05
Batch Size	100

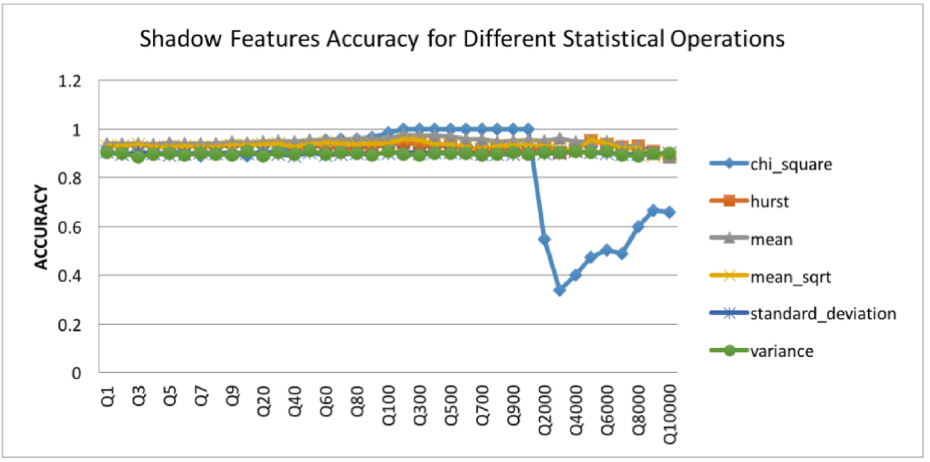

Fig. 7 A comparison between statistical operations with and without Shadow Features

chi-square, mean, root mean, standard deviation, variance and Hurst exponent. The X axis in the bar charts below represent the Q size, where we used values in the following ranges [1,10] with a step of 1, [10,100] with a step of 10, [100,1000] with a step of 100 and [1000, 10000] with a step of 1000. The Y axis represents the accuracy attained by different Q values. The accuracy is in the interval [0.0, 1.0].

As we can see from Fig. 7, by using Shadow Features with different statistical operations we get different results in the improvement of the Neural Network performance. With the chi-square operation obtaining 99% accuracy, outperforming the other operations. Another interesting comparison would be to compare the best result obtained by each statistical operation with the result obtained without applying Shadow Features to the dataset. This comparison in shown in Table 2.

All the statistical operations outperformed the case where no Shadow Features were used, which shows that the shadow features can improve the quality of HAR datasets which leads to the improvement of the performance of the classifier. Table 3 shows a comparison for the measurements of precision, f-measure, recall and specificity between the different statistical operations.

4.2 CPU VS GPU comparison

We also set up experiments to compare the running time of the Shadow Features generation process on the CPU VS the GPU. A graph showing the running time comparison for the

Table 2 Comparison of Shadow Features generated using different statistical operations and not using shadow features

Chi-Square	0.999876543
Husrt Exponent	0.949691358
Mean	0.977530864
Mean Square	0.959259259
Standard Deviation	0.908148148
Variance	0.911049383
Without Shadow Features	0.849814814

Table 3 f-measure, precision, recall and specificity for the activity labels shown for the different statistical operations

	Chi-square	Hurst	Mean	Mean sqaure	Standard deviation	Variance
Class 0						
f-measure	1	1	1	0.999814781	0.942396313	0.933953128
Precision	1	1	1	1	0.981595092	0.982011447
Recall	1	1	1	1	0.908888889	0.892962963
Specificity	1	1	1	1	0.996666667	0.996740741
Class 1						
f-measure	0.999814849	0.944055944	0.943167647	0.950686862	0.821582486	0.831794872
Precision	1	1	1	1	0.781740371	0.781364637
Recall	1	1	1	0.99962963	0.884074074	0.901111111
Specificity	1	1	1	1	0.955185185	0.955851852
Class 2						
f-measure	1	0.888027896	0.946345257	0.897206916	0.794248255	0.798543689
Precision	1	1	1	1	0.925579103	0.91886608
Recall	1	0.848888889	0.901481481	0.874444444	0.731111111	0.731111111
Specificity	1	1	1	1	0.988814815	0.987703704
Class 3						
f-measure	0.999814849	0.98973607	0.996297668	0.994803267	0.952655266	0.962545455
Precision	1	1	1	1	0.926795096	0.945357143
Recall	1	1	1	1	0.99037037	0.985185185
Specificity	1	1	1	1	0.984518519	0.988666667
Class 4						
f-measure	0.999814849	1	1	1	0.95779163	0.951977401
Precision	1	1	1	1	0.922469983	0.909581646
Recall	1	1	1	1	0.998518519	0.998518519
Specificity	1	1	1	1	0.983259259	0.980148148
Class 5						
f-measure	1	1	1	1	1	1
Precision	1	1	1	1	1	1
Recall	1	1	1	1	1	1
Specificity	1	1	1	1	1	1

case of a Kinect sensor dataset with 63 features (columns) and 16200 instances (rows) is shown in Fig. 8. On the x-axis we have the Q size as described in the previous experiment and on the y-axis we have the time in seconds that it takes to generate the Shadow Features with the respective Q size.

As we can see from Fig. 8 the running time for the GPU is below zero seconds, due to the fact that the operations run in parallel. The maximum running time for the GPU is

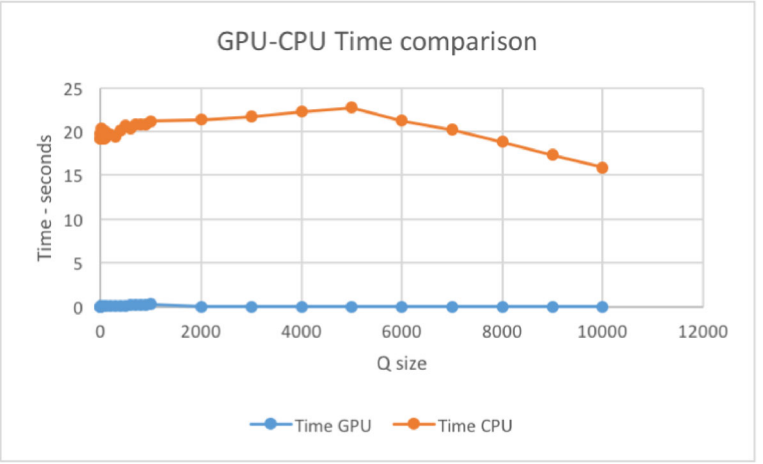

Fig. 8 GPU-CPU running time comparison

0.244879 seconds, whereas the maximum running time for the CPU is 22.74865603, which gives us a gain of 92.90 of the GPU over the CPU. These results show that using the GPU to generate shadow features in real life applications is a paramount in order for the features to be generated on the fly with minimum delay.

5 Conclusion

In this paper we presented a new pre-processing algorithm for Human Activity Recognition (HAR), Shadow Features generation, which adds extra dimensions of information to HAR datasets, with the goal of improving the quality of these datasets which would then result into better classification accuracy of the Neural Network classifier used in a HAR setting. We proposed a new HAR model in which Shadow Features are generated on the fly as soon as data is available from kinect sensors by taking advantage of the parallel processing capabilities of the GPU. We provided experimental analysis that show the importance of using the GPU instead of the CPU, and we also showed experimental results of using different statistical operations in the Shadow Feature Generation process and proved that using Shadow Features with any of the discussed statistical operations has a better performance than not using Shadow Features at all, therefore the inclusion of Shadow Features in HAR is definitely a step forward in the HAR field. In future research We plan on evaluating the performance of the proposed Shadow Features with different Machine Learning classifiers other than Neural Networks.

Acknowledgements The authors are grateful for the financial support from the Research Grants, (1) Nature-Inspired Computing and Meta-heuristics Algorithms for Optimizing Data Mining Performance, Grant no. MYRG2016-00069-FST, offered by the University of Macau, FST, and RDAO; and (2) A Scalable Data Stream Mining Methodology: Stream-based Holistic Analytics and Reasoning in Parallel, Grant no. FDCT/126/2014/A3, offered by FDCT Macau.

References

1. Agarwal A, Triggs B (2006) Recovering 3d human pose from monocular images. IEEE Trans Pattern Anal Mach Intell, 44–58
2. Babiker M, Khalifa OO, Htike KK, Hassan A, Zaharadeen M (2017) Automated daily human activity recognition for video surveillance using neural network. In: 2017 IEEE 4th international conference on smart instrumentation, measurement and application (ICSIMA), pp 1–5
3. Bagate A, Shah M (2019) Human activity recognition using rgb-d sensors. In: 2019 international conference on intelligent computing and control systems (ICCS), pp 902–905
4. Bhattacharya S, Somayaji S, Reddy P, Kaluri R, Singh S, Gadekallu T, Alazab M, Tariq U (2020) A novel pca-firefly based xgboost classification model for intrusion detection in networks using gpu. Electronics 9:219
5. Blank M, Gorelick L, Shechtman E, Irani M, Basri R (2005) Actions as space-time shapes. In: Proceedings of the tenth IEEE international conference on computer vision (ICCV), vol 2, pp 1395–1402
6. Campbell LW, Becker DA, Azarbayejani A, Bobick AF, Pentland A (1996) A invariant features for 3-d gesture recognition. In: Proceedings of the second international conference on automatic face and gesture recognition, pp 157–162
7. Chan JH, Visutarrom T, Cho S-B, Engchuan W, Mongolnam P, Fong S (2016) A hybrid approach to human posture classification during tv watching. J Med Imag Health Inform. Accepted for publication
8. Danafar S, Gheissari N (2007) Action recognition for surveillance applications using optic flow and svm. Comput, 457–466
9. Dollár P, Rabaud V, Cottrell G, Belongie S (2005) Behavior recognition via sparse spatio-temporal features. In: Proceedings of the 2nd joint IEEE international workshop on visual surveillance and performance evaluation of tracking and surveillance, pp 65–72
10. Gadekallu T, Khare N, Bhattacharya S, Singh S, Reddy P, Srivastava G (2020) Deep neural networks to predict diabetic retinopathy. J Ambient Intell Humaniz Comput
11. Gavrilova M, Wang Y, Ahmed F, Paul PP (2018) Kinect sensor gesture and activity recognition: New applications for consumer cognitive systems. IEEE Consum Electron Mag 71:1–8
12. Hoang LUT, Ke S, Hwang J, Yoo J, Choi K (2012a) Human activity recognition based on 3d body modeling from monocular videos. In: Proceedings of frontiers of computer vision workshop, pp 6–13
13. Hoang LUT, Tuan PV, Hwang J (2012b) An effective 3d geometric relational feature descriptor for human action recognition. In: Proceedings of IEEE RIVF international conference on computing and communication technologies, research, innovation, and vision for the future (RIVF), pp 1–6
14. Iglesias JA, Angelov P, Ledezema A, Sanchis A (2010a) Human activity recognition based on evolving fuzzy systems. Int J Neural Syst 20:355–364
15. Iglesias JA, Ledezma A, Sanchis A (2010b) Human activity recognition based on evolving fuzzy systems. Int J Neural Syst 20:355–364
16. Käse N, Babaee M, Rigoll G (2017) Multi-view human activity recognition using motion frequency. In: 2017 IEEE international conference on image processing (ICIP), pp 3963–3967
17. Ke Y, Sukthankar R, Hebert M (2007) Spatio-temporal shape and flow correlation for action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 1–8
18. Kim Y, Sim S, Cho S, Lee W, Cho K, Jeong YS, Um K (2014) Intuitive nui for controlling virtual objects based on hand movements. In: Lecture notes in electrical engineering, vol 309, pp 457–461
19. Lee K, Chae S, Park H (2019) Optimal time-window derivation for human-activity recognition based on convolutional neural networks of repeated rehabilitation motions. In: 2019 IEEE 16th international conference on rehabilitation robotics (ICORR), pp 583–586
20. Lee S-M, Yoon SM, Cho H (2017) Human activity recognition from accelerometer data using convolutional neural network. In: 2017 IEEE international conference on big data and smart computing (BigComp), pp 131–134
21. Leo M, D'Orazio T, Spagnolo P (2004) Human activity recognition for automatic visual surveillance of wide areas. ACM, New York, pp 124–130
22. Liu C, Ying J, Han F, Ruan M (2018) Abnormal human activity recognition using bayes classifier and convolutional neural network. In: 2018 IEEE 3rd international conference on signal and image processing (ICSIP), pp 33–37
23. Mitra SK (2011) Human activity recognition using dtf. In: Proceedings of the third IEEE national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG), pp 239–242
24. Müller M, Röder T (2005) Clausen efficient content-based retrieval of motion capture data. In: ACM Trans, pp 677–685

25. Museum TJPG (1990) Photography: discovery and invention
26. Psychoula I, Merdivan E, Singh D, Chen L, Chen F, Hanke S, Kropf J, Holzinger A, Geist M (2018) A deep learning approach for privacy preservation in assisted living. In: 2018 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops), pp 710–715
27. Shecthman E, Irani M (2005) Space-time behavior based correlation. In: IEEE computer science society conference on computer vision and pattern recognition (CVPR), vol 1, pp 405–412
28. Singh D, Merdivan E, Hanke S, Kropf J, Geist M, Holzinger A (2017a) Convolutional and recurrent neural networks for activity recognition in smart environment, 194–205
29. Singh D, Merdivan E, Psychoula I, Kropf J, Hanke S, Geist M, Holzinger A (2017b) Human activity recognition using recurrent neural networks. In: Machine learning and knowledge extraction. Springer International Publishing, pp 267–274
30. Song W, Lu Z, Li J, Li J, lIao J, Cho K, Um K (2014) Hand gesture detection and tracking methods based on background subtraction. In: Lecture notes in electrical engineering, vol 309, pp 485–490
31. Sorkun MC, Danişman AE, İncel D (2018) Human activity recognition with mobile phone sensors: impact of sensors and window size. In: 2018 26th signal processing and communications applications conference (SIU), pp 1–4
32. Tsitsoulis A, Bourbakis N (2013) A first stage comparative survey on human activity recognition methodologies. Int J Artif Intell Tools, 22
33. Veeraraghavan A, Roy-Chowdhury AK, Chellapa R (2005) Matching shape sequences in video with applications in human movement analysis. IEEE Trans Pattern Anal Mach Intell, 1896–1909

Affiliations

Ricardo Brito[1] · Robert P. Biuk-Aghai[1] · Simon Fong[1]

Robert P. Biuk-Aghai
robertb@umac.mo

Simon Fong
ccfong@umac.mo

[1] Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Avenida da Universidade, Taipa, Macau S.A.R., China