An Empirical Study on Bidirectional Recurrent Neural Networks for Human Motion Recognition

Pattreeya Tanisaro
Institute of Cognitive Science, University of Osnabrück, Germany
pattanisaro@uni-osnabrueck.de

Gunther Heidemann
Institute of Cognitive Science, University of Osnabrück, Germany
gheidema@uni-osnabrueck.de

Abstract

The deep recurrent neural networks (RNNs) and their associated gated neurons, such as Long Short–Term Memory (LSTM) have demonstrated a continued and growing success rates with researches in various sequential data processing applications, especially when applied to speech recognition and language modeling. Despite this, amongst current researches, there are limited studies on the deep RNNs architectures and their effects being applied to other application domains. In this paper, we evaluated the different strategies available to construct bidirectional recurrent neural networks (BRNNs) applying Gated Recurrent Units (GRUs), as well as investigating a reservoir computing RNNs, i.e., Echo state networks (ESN) and a few other conventional machine learning techniques for skeleton-based human motion recognition. The evaluation of tasks focuses on the generalization of different approaches by employing arbitrary untrained viewpoints, combined together with previously unseen subjects. Moreover, we extended the test by lowering the subsampling frame rates to examine the robustness of the algorithms being employed against the varying of movement speed.

2012 ACM Subject Classification Mathematics of computing → Time series analysis

Keywords and phrases Recurrent Neural Networks, Human Motion Classification, Echo State Networks, Motion Capture, Bidirectional Recurrent Neural Networks

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.21

1 Introduction

The recurrent neural networks, whose structures are similar to those of multilayer perceptrons (MLPs) have been widely used for sequential data processing. Nonetheless, they are different from the MLPs by allowing connections among hidden units, therefore the networks can retain information of past inputs as a vector of activation for each time step which makes RNNs exceedingly deep. Their depth, however, makes them difficult to train because the update of the weight matrices with a gradient-based approach such as Backpropagation Through Time (BPTT) leads to exploding and vanishing gradient problems [1]. Many techniques have been introduced in order to solve these two issues, especially for the vanishing gradient problem, but training RNNs was still a very difficult task and the applications were limited.

Since the emergence of special architecture for gradient-based methods called LSTM [17], training RNNs has become easier and more successful in numerous tasks such as speech recognition [12], acoustic modeling [24], sequence labeling in speech recognition [13], handwriting recognition [15], language modeling and machine translation [30, 31, 4], prediction of successful shooting in basketball [28], analyzing motion patterns in autonomous driving [11], image caption [36] and learning of video representation [29]. The LSTM is

© Pattreeya Tanisaro and Gunther Heidemann;
licensed under Creative Commons License CC-BY

designed to solve the vanishing gradient problem whereas truncating the gradient is harmless to the networks because an LSTM can enforce a constant error flow within special units to bridge time lags. Unlike the traditional recurrent unit which calculates the weighted sum of inputs and directly applies the activation function, the LSTM unit contains a memory cell. Furthermore, there are several studies such as [26] and [18] that reported their achievement of an improvement of the output performance by introducing the depth to the RNNs. Typically, any RNNs when unfolded in time might be considered deep themselves, because the input to output in a given time span has crossed several nonlinear layers as computational paths [26]. Nonetheless, the *depth* of neural networks is usually defined by the number of feedforward neural layers. Most studies in deep RNNs concentrate on sequence-to-sequence modeling, particularly for language modeling for instance [30, 16, 18, 26, 6].

In our study, we focused on solving a classification problem using a special type of deep RNNs, called bidirectional RNNs (BRNNs) which were first introduced by [27] in the late 1990s. Nevertheless, they started to attract attention many years later after a groundbreaking achievement of sequence labeling in speech recognition by [13]. The BRNN is an RNN which contains a separation of a forward and backward pass for positive and negative time direction. Therefore, it is able to store the past and future context, whereas a conventional RNN can only partially achieve this by delaying the output by certain time steps. Our study is set up by employing more than 300 configurations for deep BRNNs after the preliminary tests, of which we inspect how the classification performance is affected by changing the ***width*** and the ***depth*** of the hidden layers. The designed networks are set up in a generic sense by simply stacking multilayer RNNs to have the required depth. The evaluation is based on three Motion Capture datasets for comparing BRNNs with ESNs [19, 20] and traditional machine learning techniques. Motion Capture (MoCap) is a marker-based system which by its high-dimensional nature, nonlinearities and long-range dependencies make it ideal for studying the limitations of time series models [35]. Although the focus of our study is on this particular domain, the design of BRNNs is not just solely specific for MoCap datasets. We are convinced that the study is also applicable to other high dimensional time series data.

2 Related Work

Many studies have demonstrated a superior functionality of applying deep RNNs when compared to shallow networks, for instance, [30] introduced a new architecture called multiplicative RNNs by using multiplicative connections to allow the current input character for the character-level in language modeling to determine the hidden-to-hidden weights. However, this model was trained with Hessian-Free optimizer (HF) instead of gradient descent. The work of [16] focused on a hierarchy of RNNs for character-level language modeling using stochastic gradient descent. It proposed two alternative architectures which are deep MLPs with three hidden layers stacked from one layer on top of each other with temporal feedback loops. One architecture uses feedback loops from output but with the last hidden layer contributing to the output layer, while another architecture allows all the connections from each hidden layer contributing to the output layer. Four other different models to construct deep RNNs have been proposed by [26] for three language models. Quite recently, a hierarchical multiscale RNN model has been presented by [6]. It shows that the proposed network architecture can learn the latent hierarchical multiscale structure from temporal data for character-level language modeling. A similar study to our work which employs bidirectional RNNs for classification tasks has been discussed in [14]. It used five hidden layers of BRNNs with LSTM to classify 61 phoneme outputs in which each layer

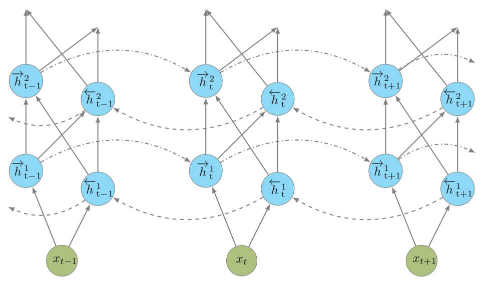
$\overrightarrow{h}^2_{t-1}$
$\overleftarrow{h}^2_{t-1}$
$\overrightarrow{h}^2_{t}$
$\overleftarrow{h}^2_{t}$
$\overrightarrow{h}^2_{t+1}$
$\overleftarrow{h}^2_{t+1}$
$\overrightarrow{h}^1_{t-1}$
$\overleftarrow{h}^1_{t-1}$
$\overrightarrow{h}^1_{t}$
$\overleftarrow{h}^1_{t}$
$\overrightarrow{h}^1_{t+1}$
$\overleftarrow{h}^1_{t+1}$
x_{t-1}
x_t
x_{t+1}

Figure 1 A deep BRNN with two hidden layers following [14]. The *dashed-dotted lines* indicate the **forward** direction depicted by $\overrightarrow{h}_t$ and the *dashed lines* indicate the **backward** direction $\overleftarrow{h}_t$.

consists of $2 \cdot 250$ cells. A comprehensive comparative study of deep RNNs has been revealed in [12]. It demonstrated the results of using from one to five hidden layers while fixing the number of neurons for all hidden layers. The results from this experiment exhibited that: i) LSTM works better than the typical *tanh* neuron, ii) bidirectional RNNs with LSTM also give better output performances than typical unidirectional RNNs with LSTM units, and iii) the depth size is more important than the width size. In addition, by fixing the number of neurons of each hidden layer, the networks with three hidden layers work as well as those with five layers, while the number of weights of five hidden layers is almost twice their number for three layers. Furthermore, the evaluation in [18] also confirms that shallow BRNNs outperform shallow unidirectional RNNs on extracting sentence-level opinion expression. It concludes that, for a large network, three hidden layers provide the best output performance for their tasks. In case of a small network, two, three and four hidden layers show equally good performance for certain sizes. By adding more layers, its performance decreases. Further, the study suggests that in conventional stacked deep learners, every hidden layer conceptually lies in a different representation space, and establishes a more abstract and higher-level representation of the input. By taking these findings as our guidelines, we then hypothesized that the activities at each layer could represent some forms of the action descriptors.

3 Classification Approaches

3.1 Deep Bidirectional RNNs

Generally, the BRNN has been applied for sequence-to-sequence learning, particularly for Natural-Language Processing (NLP) tasks. The structure of the BRNN consists of two RNNs, one to compute the forward hidden sequences $\vec{h}_t$ and the second is to compute backward hidden sequences $\overleftarrow{h}_t$. Figure 1 shows an architecture of the BRNN for two hidden layers. Let $x = (x_1, .., x_T)$ be an input sequence for T time steps. The final output y_t is accumulated across the T frames at the last layer and is classified by the probability of human action classes using the Softmax function. We computed the output sequence at time t of y according to [14] as:

$$y_t = g(W_{\vec{h}ny}\vec{h}_t + W_{\overleftarrow{h}ny}\overleftarrow{h}_t + b_y) \tag{1}$$

$g(\cdot)$ is an activation function, $W_{\vec{h}y}$ and $W_{\overleftarrow{h}y}$ are weight matrices at the output layer and b_y is an output bias. The $\overleftarrow{h}_t$ can be interpreted as a summary of the past from $t = T$ to 1, whereas $\vec{h}_t$ is the summary of the future from $t = 1$ to T. The activities in forward and

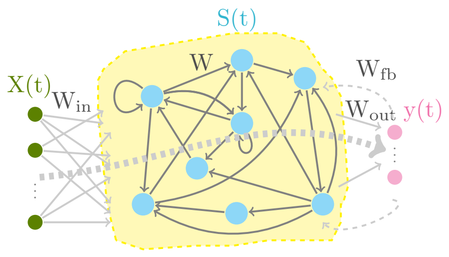

Figure 2 Architecture of an ESN. The *dashed lines* denote the connections which are not compulsory.

backward direction can be written by:

$$\vec{h}_t^n = f(W_{\vec{h}^{n-1}\vec{h}^n}\vec{h}_t^{n-1} + W_{\vec{h}^n\vec{h}^n}\vec{h}_{t-1}^n + b_{\vec{h}}^n) \tag{2}$$

$$\overleftarrow{h}_t^n = f(W_{\overleftarrow{h}^{n-1}\overleftarrow{h}^n}\overleftarrow{h}_t^{n-1} + W_{\overleftarrow{h}^n\overleftarrow{h}^n}\overleftarrow{h}_{t+1}^n + b_{\overleftarrow{h}}^n) \tag{3}$$

where h^0 is the input sequence.

3.2 Echo State Networks

An ESN shown in figure 2 is a type of RNN, whose design does not depend on updating weights by gradient computation, but, instead it creates a random *dynamical reservoir RNN*. The reservoir is then driven by the training data and leaves the weights untrained. The output weights are computed at the readout connection using a linear regression of $y(t)$. The internal unit activities $\vec{S}$ in figure 2 can be updated by:

$$\vec{S}(t) = f(W_{in}\vec{x}(t) + W\vec{S}(t-1) + W_{fb}\vec{y}(t)) \tag{4}$$

$f(\cdot)$ is an activation function of the neurons, a common choice is $tanh(\cdot)$ applied element-wise. By employing the time warping of the input signals, the *leaky integration rate* [21] $\alpha \in (0, 1]$ is adopted to determine the speed of the reservoir update dynamics. The update rule for the internal units is extended to:

$$\vec{S}_{leaky}(t) = (1-\alpha)\vec{S}(t-1) + \alpha\vec{S}(t). \tag{5}$$

Applying a simple linear regression at the readout layer leads to output $\vec{y}(t)$:

$$\vec{y}(t) = W_{out}[\vec{x}(t); \vec{S}(t)] \tag{6}$$

3.3 Traditional Machine Learning Methods

Most work on time series classification for example, the well-known UCR archive, of which all datasets are of one dimensional feature, focuses on an adaptation of distance measures using 1-Nearest Neighbor (1-NN) with Euclidean Distance (ED) and Dynamic Time Warping (DTW). Both techniques have proven to perform very well on the UCR archive, especially DTW. Nevertheless, DTW has limited its applications only on the fixed length data (that is one must extrapolate the shorter sequence in order to have an equal length between two sequences) and gives rise to some issues for the case of multi-dimensional data e.g., the computational complexity and the selection of the dependent or independent warping distance function [32].

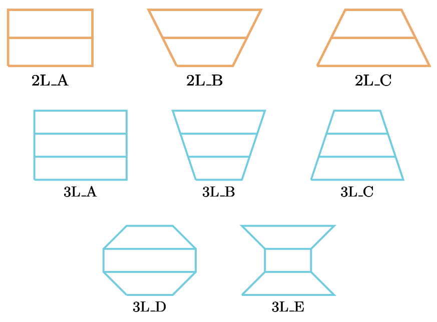

Figure 3 Different geometries representing eight models used in our experiment with varying width of two and three hidden layers.

We adopted two transformation approaches to extract the feature vectors in our experiments. They are **i) a naïve method** by stacking each frame on top of one another, and using majority voting to decide the best course of action, and **ii) a dimensionality reduction technique** of the zero-mean skeleton configuration for feature vectors demonstrated in [34, 23]. The two best classifiers for MoCap classification, cited in [34], k-NN and Random Forest (RF), were chosen for both transformation approaches.

4 Configurations

4.1 BRNN architectures

The RNNs with one hidden layer (**1L**) represent shallow networks, while those with more than one hidden layers represent the deep RNNs. According to [14] and [18], three hidden layers are sufficient to achieve the best performance, while adding more hidden layers worsens the output performance. On this account, in our experiment, we concentrated on evaluating geometries of the networks with two and three hidden layers as illustrated in figure 3. RNNs with three stacked hidden layers numbered from bottom to top labeled as ($\mathbf{L_1} \cdot \mathbf{L_2} \cdot \mathbf{L_3}$) will be referred to throughout the experiment indicating the number of units in **one direction**. (We use the term cell or unit instead of neuron to emphasize the use of gated units in RNNs.) Model *2L_A*, as seen in the figure 3 is depicted for the two hidden layers in which the number of cells of the two hidden layers is almost equal. The model *2L_B* is for two hidden layers, of which the number of neurons of L_2 (top) is at least double the size of L_1 (bottom), and vice versa for model *2L_C*. Note that the first hidden layer ($\boldsymbol{L_1}$ connects to the input nodes laying at the bottom. For models with three hidden layers, we extended the geometries to five architectures as illustrated in figure 3. Here, we combined cell numbers in $\{50, 100, 150, 200, 250, 300, 350, 400\}$ to form these geometries which have a limited total amount of cells from $2 \cdot 200$ up to $2 \cdot 600$ units.

4.2 BRNN configurations and hyperparameters

In order to verify the impact of the width and the depth on the network as well as to simplify the experiment, several parameters in the experiment had been previously investigated. They are: **i) Cell type**. In our experiment, instead of a well-known LSTM, we replaced each cell unit at h with a gated recurrent unit called GRU [4]. GRU is a variation of a gating mechanism and is comparable to LSTM and has been primarily used in machine translation as encoder-decoder models. It is similar to LSTM in that the gating units modulate the flow of information inside the unit, but without memory cells. Comparative studies of using traditional recurrent unit *tanh*, LSTM and GRU, found in [7] and [22] have shown that the gated units, both the LSTM and GRU outperform the conventional recurrent unit. To achieve certainty, we had examined these three cells in BRNNs in our preliminary tests. The networks with GRU cells significantly outperformed the other two cells by more than 5% in all experiments. Therefore, our classification results were from applying the GRU units to the networks. **ii) Optimizer**. In contrast to the optimizer benchmark for RNNs in Penn TreeBank language modeling [8] which mentions that Adam and RMSProp do not work well with RNNs, we found that in our case, both of them converged very quickly even with a very small learning rate and gave good output performance. The primary test was carried out for a shallow BRNN. The selected learning rates for each optimizer here were the recommended values in the papers based on MNIST dataset. For the rest of the experiments, we chose RMSProp as our default optimizer which outperformed all other optimizers. **iii) Regularization**. Because of the limited amount of data, we cannot take out some data for validation. Nonetheless, we added a regularization term L^2 to the objective function with a fixed regulation $\lambda = 0.02$. It is interesting to note that increasing the regularization parameter from 0.01 to 0.02 increases the recognition rate by about 3-5% in most models. We applied the norm clipping with a maximum gradient norm limited to one, and no *dropout* was applied.

4.3 ESN configurations

The ESN configurations in the experiments follow the guidelines suggested in [33] which demonstrated the influence of the ESN settings on various datasets on the UCR archive. Several key parameters are: **i) Sparsity of the reservoir**. In corresponding with BRNN networks which have the networks size in $2 \cdot \{200, .., 600\}$, we set the reservoir size using only half of BRNN with connectivity of 10, 30, 50 and 70% respectively. **ii) Spectral radius** which is considered to be big for the tasks that require an extensive history of an input, while one is served as a reference point. We picked 5.0 from [33]. **iii) Leaky rate** which can be regarded as time warping of the input signals was fixed at 0.1 for all configurations. **iv) Input scaling**: was set to 2.0 similar to [33] and **v) regularization coefficient** was fixed at 0.1. Moreover, the network weight was set to have a uniform distribution in the range of [-0.5,0.5] and no feedback connection was considered here.

4.4 Traditional machine learning configurations

For the direct naïve method, we employed two popular classifiers for classification, 1-NN and RF with 75 trees, which yielded best output performance in [34]. To prove the case of a combined manifold learning with classification, we chose the PCA with two and three components associated with RF and 1-NN.

5 Datasets and Experimental Setups

5.1 Datasets

We evaluated the proposed techniques as described in section 3 and 4 using three MoCap datasets, MHAD-27, MHAD-10 and HDM05.

MHAD-27[1] [2] consists of 27 different actions performed by eight subjects. Each subject repeated the same action four times. The dataset contains a total of 861 data sequences, where three sequences were corrupted and removed from the dataset on the official website.

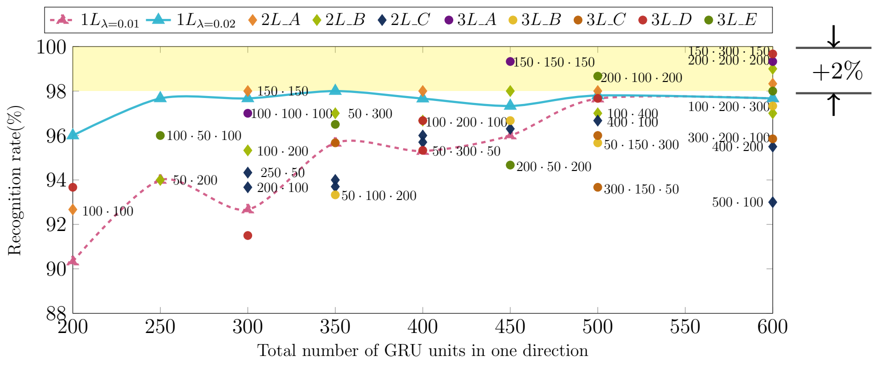

Figure 4 The recognition rates of MHAD-10 from designated network architectures using five-fold cross validation. The setup does not condition on the separation of test subjects from the training set.

The training was performed on six subjects, and two subjects were left out for the test. The recognition rate was reported on an average of 28 combinations. This dataset was recorded using 20 markers. Therefore, we have the feature vector which is captured in 3D space for BRNN of size $(3 \cdot 20) \times 120$, where 120 is the amount that is close to the maximal sequence length of this dataset. Zeros were appended to the shorter sequences in BRNNs. It is important to note that ESN can handle different lengths of the data sequences, so the data is trained with the original length.

MHAD-10 [2] is a 3D MoCap dataset of six subjects performing 10 different hand gestures tracked with 25 markers. The four additional markers comparison with MHAD-27 were put on the left and right hand and a thumb, and one additional marker was put on a spine. Each subject repeats an action 5 times (trials). Therefore, we have a total of 300 videos with various sequence lengths. For BRNN, we chose to fix the number of sequences to 150 which is close to the maximal length of the sequence in the dataset. This makes a feature vector size of $(3 \cdot 25) \times 150$. The training is performed on five subjects from all trials and an unseen subject is left out for testing. Hence the recognition rate is an average of six-fold cross-validation.

HDM05 [25] was originally made up of 130 classes consisting of five subjects performing actions with and without repeating the same cycles separately. This created a total of 2343 sequences. We followed [5, 9, 38] in grouping non-repetitive and repetitive motions together yielding 65 actions. There were about 20 actions which have samples less than 20 i.e., *throwBasketball*, *throwFarR* and *jumpDown* having only 14 trials each, while actions such as *walk*, *elbowToKnee* and *runOnPlaceStart* have 94, 80, 74 trials, respectively. This leads to an unbalancing of data and causes a huge bias towards a particular action. Nonetheless, since we focused on the action recognition of unseen subjects, therefore four subjects were used in the training set and one subject was for the test. We reduced the original number of markers to 19, where some nearby sensors e.g., on the spine as used in MHAD were merged. The average sequence length is 261 with the maximum of 901. With the limitation of our computational capacity, we set the data using the fixed length of 400 for BRNNs. Therefore, the feature vectors of BRNNs have a size of $(3 \cdot 19) \times 400$.

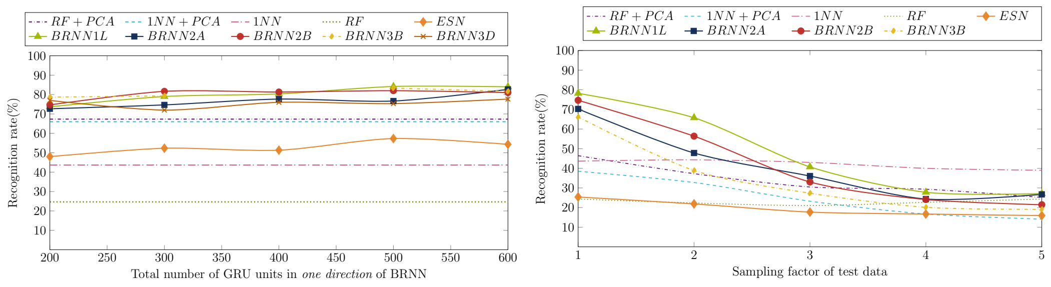

Figure 5 The recognition rates of MHAD-10 by excluding test subjects from the training set. **Left)** Changing the total number of GRU units in one direction of BRNN or the reservoir size of ESN. **Right)** Extending the test on the left by changing subsampling factors of test data of untrained viewpoints.

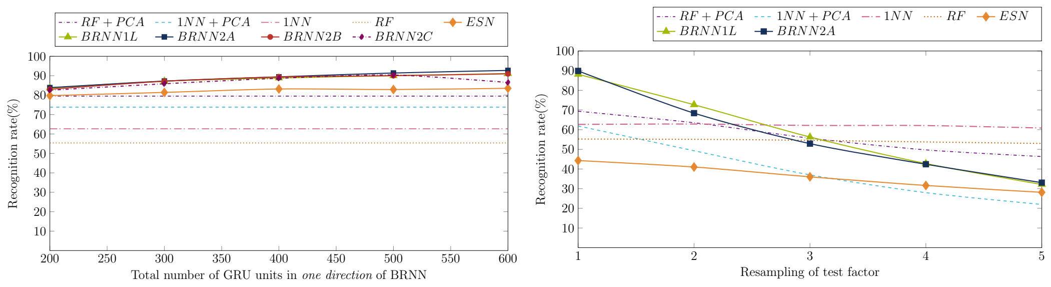

Figure 6 The recognition rates of MHAD-27 by excluding test subjects from the training set. **Left)** Changing the total number of GRU units in one direction of BRNN or the reservoir size of ESN. **Right)** Extending the test on the left by changing subsampling factors of test data of untrained viewpoints.

5.2 Experimental Setups

We normalized each sequence with respect to its in-frame reference of that dataset, where the reference is the joint that laid at the center of the skeletal torso. The evaluations were composed of three experiments: **i) Having some insights of deep networks strategies by varying the width and height of the network.** In this experiment, we did not impose conditions on separation of test subjects from the training set. **ii) Finding a few good models by varying the number of cells in the networks using unknown subjects**. The experiment was set up in a way to find a few good models of each dataset by varying the number of cells in the networks in 200-600 units (one direction in BRNN and the total units in a reservoir for ESN), where the test subjects were excluded from the training set. More than 300 configurations for BRNN were constructed to obtain the best output and created some insights of construction strategies for deep networks. **iii) Extending the test using untrained viewpoints with the variations of speed.** We enhanced the experiment by extending the training set to have five camera angles $\{-90, 45, 0, 45, 90\}$, whereas test angles are in $\{-80, -70, ..., 70, 80\}$. Moreover, we subsampled the original test data using a subsampling factor of 1, 2, 3, 4 and 5, while the training data still remained the same. The subsampling factor of 2 means that every 2^{nd} frame of the data will be taken instead of each single frame (factor of 1).

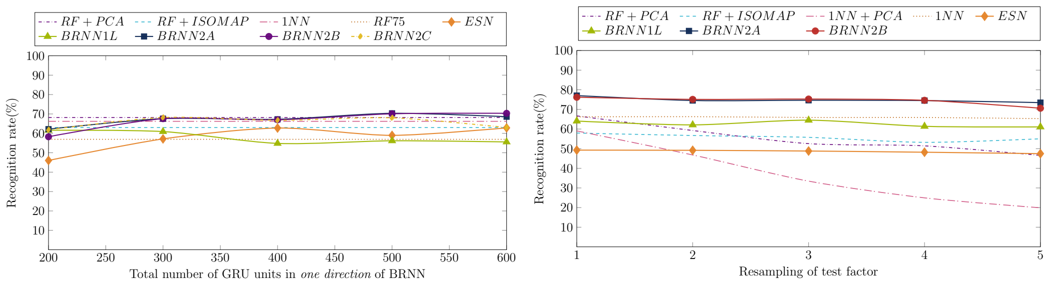

Figure 7 The recognition rates of HDM05 by excluding test subjects from the training set. **Left)** Changing the total number of GRU units in one direction of BRNN or the reservoir size of ESN. **Right)** Extending the test on the left by changing subsampling factors of test data of untrained viewpoints.

6 Experimental Results

6.1 Discussion of results

Firstly, we investigated the effects of geometries of BRNNs following the **construction strategies** depicted in Figure 3. The recognition rates of MHAD-10 performing on average of five-fold cross-validation using shallow and deep BRNNs are shown in Figure 4. Moreover, the recognition rates of each strategy are the average of two runs with the standard deviation of $\pm 3\%$. These test results gain very high recognition rates because they are not based on the separation of test subjects from the training set. It is obvious that the recognition rates which are better than 98% (the yellow shaded area in Figure 4) can only be achieved when the networks are relatively large i.e., the number of cells is greater than $2 \cdot 400$. Furthermore, the models which have any layer containing 50 cells yield output worse than others. This might be because the input feature of MHAD-10 has a size of 75 and any form of dimensionality reduction at any hidden layer in RNNs by shrinking the network's width is not suitable. Hence, by the experimental results, we conclude that the width of a hidden layer next to the input layer in one direction should be larger than the size of the input features.

We enhanced the experiment by **excluding test subjects from the training set to examine the generalization of the models**. The recognition rates of three datasets for two experimental setups of MHAD-10, MHAD-27 and HDM05 can be found in Figure 5, 6 and 7 on the left and right of the figures respectively. In these figures, only a few best models of BRNN were chosen from various configurations based on five geometries in Figure 3. Furthermore, the results also demonstrate the output of the shallow networks, the output of ESNs in section 3.2 and the output of machine learning approaches from section 3.3.

The results of the first experiment displayed on the left of Figure 5, 6 and 7 show that the shallow and deep BRNNs significantly outperform other methods in the MHAD-10 and MHAD-27 dataset. The deep BRNNs are slightly better than the other methods on large networks for HDM05. In addition, there are some considerable differences in recognition rates among both datasets and algorithms, especially for MHAD-10. Even though MHAD-10 consists of only 10 actions, nine out of these actions are the movements of solely the right hand. On the contrary, even though MHAD-27 consists of more actions than MHAD-10, the actions also involve a variety of movements of hands and legs which leads to overall better recognition rates of all methods. The dimensionality reduction techniques in combination with RF outperform other machine learning techniques and are close to the winner of HDM05 using deep networks, *BRNN2C* using the total of $2 \cdot 600$ cells. The confusion matrix of

HDM05 using *BRNN2B* model on Figure 7-left is depicted in Figure 8. Actions with one hundred percent recognition rate (recall) from all runs (filled with dark brown in Figure 9) are for instance, *clapAboveHead*, *jogLeftCircle*, *rotateArmsBothBackward*, *standUpKneelToStand* and so forth. The most common misclassified actions in all classifiers are between *deposit* and *grab* as the trajectories of actions drawn in Figure 9. It is difficult to track the trajectories in 3D of a stationary image by eye; hence, we projected a 3D image onto a 2D plane and as we can see, the trajectories from these two groups cannot be distinguished. Therefore, when we allow classification using top three correctness, the recognition rates increase by 12-15% in all methods. For HDM05, there are no significant differences among different classification methods. When applying subsampling factors to **simulate the changes of movement speed** for the test data of MHAD-10 and MHAD-27 using untrained viewpoints, then increasing the subsampling factor decreases the recognition rate. Interestingly, however, this effect does not apply to HDM05. This might be because only HDM05 consists of non-repetitive and repetitive sequences in one action which allows the networks to easily capture the changes of patterns of the action as varying of movement speeds, while MHAD-10 and MHAD-27 only consist of one periodic movement in each action. Besides, by increasing the number of viewpoints in training deep BRNNs, the recognition rates of HDM05 have been increasing by approximately 10% on arbitrary untrained viewpoints.

The results also reveal that deep BRNNs using two layers for the total number of cells greater than $2 \cdot 500$ units such as *BRNN2A* and *BRNN2B* surpass all other models for all three datatsets, including three layers of BRNNs. The shallow BRNNs work equally well or even better than the deep BRNN for MHAD-10 and MHAD-27 but not for HDM05. It is important to note that models with three hidden layers do not perform better than models with two hidden layers, while training such gradient-based approaches requires a large amount of computation time on GPU. The computation time and cost of training and testing BRNNs is much greater than training ESNs, which the training is only performed for the output weights at the readout where there is no cyclic dependency. Training and testing using dimensionality reduction methods demand the least time and computational power. Considering the time complexity for the gradient-based learning by BPTT, it must be analyzed in terms of space for the number of values stored and the time complexity in terms of the number of arithmetic operations required [37]. Therefore, measuring architecture complexity of RNNs is not a trivial task. Nonetheless, for our configurations when the network is fully connected and all weights are adaptable, if the shallow network requires time $\mathcal{T}$ to complete the task, the deep network can be expected to complete the task in about $\mathcal{L} \cdot \mathcal{T}$, where $\mathcal{L}$ is the number of hidden layers.

A Comparison. Other studies which resemble our first experiment use only one default view and do not exclude test subjects from training data. Furthermore, some approaches apply some prior filters before passing data to the networks, for instance, [3] proposed a hybrid MLPs which reported the recognition rate with an accuracy of 95% on HDM05 on 10-fold evaluation. Next, [9] introduced deep BRNN by stacking BRNNs using LSTM units on each skeleton part yielding the best result of 96.92% for HDM05. Followed by [38] which use deep BRNNs with LSTM units on body parts similar to [9] but added a so-called co-occurrence matrix and dropout to a three-LSTM layer with two feedback layers. It accounted for the recognition rate of 97.25%.

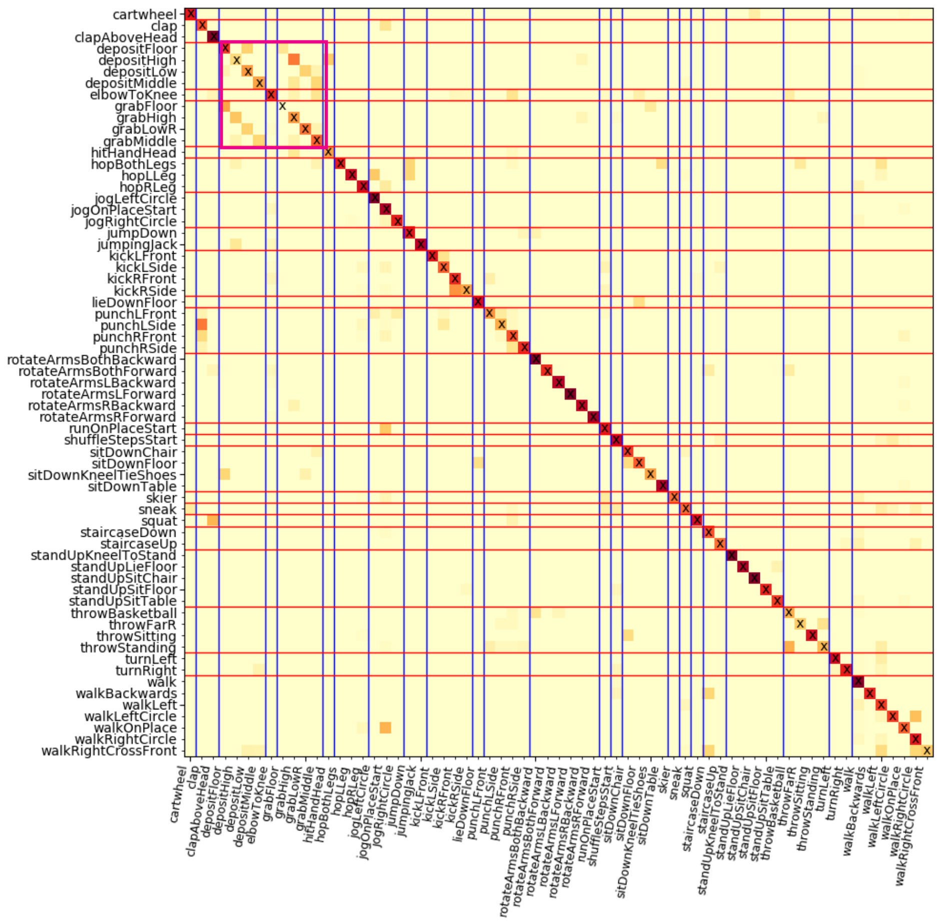

Figure 8 Confusion matrix of HDM05 with 65 actions on average of 2-folds using best model of *BRNN2B* $(2 \cdot [100 \cdot 400])$. The weighted colors are computed from the percentage of the total number of that action. The thick pink rectangle at the left corner shows a group of actions which significantly misclassified in all methods. The red horizontal and vertical blue lines are drawn to highlight groups of the actions.

6.2 Visualization of BRNN

Last hidden layer. For the BRNNs, only half of the output activities of the last hidden layer contribute to the output layer as can be seen in Figure 10. This figure could explain why the number of cells in one direction at the last hidden layer needs to be much greater than the number of the output classes which are required by the Softmax function at the output.

Visualization of input and other hidden layers. Multidimensional time-series data cannot be directly visualized, therefore investigating its behavior is very difficult. One common approach that is normally used in order to get some insight into high dimensional time-series data is by examining their distance matrix. One benefit of using distance matrices, such as Euclidean distance is that we can further analyze the matrix using recurrence plots (RPs) [10] by applying a threshold distance and the Heaviside function. The RPs can tell when the phase space trajectory of the dynamic system re-occur roughly in the same area in the phase

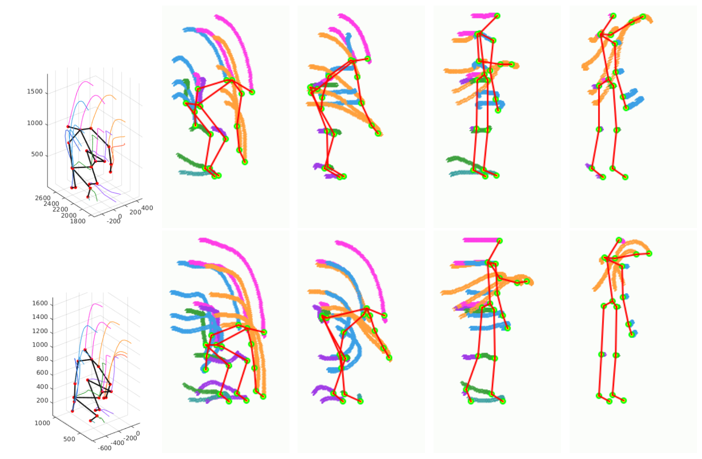

Figure 9 Most common misclassified patterns are the confusion between *"deposit"* and *"grab"*. Top) From left to right: 3D projection of *depositFloor* of default view, and the rotated 2D projections of *depositFloor*, *depositLow*, *depositMiddle* and *depositHigh*, Bottom) *grabFloor* in 3D and 2D projections of *grabFloor*, *grabLowR*, *grabMiddle* and *grabHigh*.

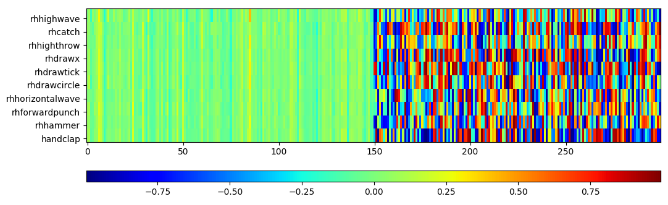

Figure 10 The activities from the last hidden layer with $2 \cdot 150$ cells of MHAD-10 dataset. The left side shows the activities from forward and the right side from the backward direction.

space. Figure 11 shows the Euclidean distance matrices of a subject in HDM05 performing an *elbowToKnee*. The left-most of the figure shows the distance matrix of the input which reveals a few harmonic oscillations that can be observed by the checkerboard structures. The next three figures are the activities from the first hidden layer L_1 for the combination of both directions, for forward and backward direction, respectively. We can infer from the changes of one state of learning to another that the networks opt to differentiate their output activities at each layer. At the upper layer, the scale of the differentiation is larger. The very dark blue corresponds to distance zero and red to the maximum distance between the features in this time span.

7 Conclusion

During the course of conducting our research, we have demonstrated the various influences of various geometries of the deep BRNN's upon human motion recognition. It is crucial to have some empirical insights to amend and influence both the width and depth of the model to suit the research requirements and objectives. The evaluations of the classifications were performed by focusing on the generalization and the robustness of the models by testing on unseen subjects with the variation of movement speeds. The results showed that BRNNs outperformed ESNs and other conventional classification techniques. Correspondingly, we

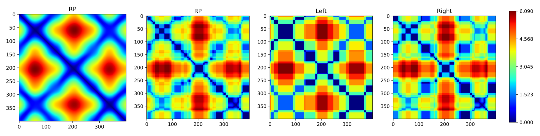

Figure 11 The Euclidean distance matrices of a subject performing *elbowToKnee*. From left to right: i) input ii) the retrieved activities from the first hidden layer of combined directions, iii) forward and iv) backward direction.

discovered that any form of dimensionality reduction, caused by reducing the width of the hidden layers to less than the number of input features or reducing the width of last hidden layer in one direction less than the output units is unsatisfactory. The shallow networks should be included and examined in the experiment as they may not only demonstrate good performance for some datasets, but also provide some insights into the impact of hyper parameters. Nonetheless, to achieve the best outcomes, based on our research, we strongly recommend that deep RNN's as the method of choice for researchers to employ.

References

1 Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, mar 1994. doi:10.1109/72.279181.

2 Chen Chen, Roozbeh Jafari, and Nasser Kehtarnavaz. *UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor*, volume 2015-December, pages 168–172. IEEE Computer Society, 12 2015. doi:10.1109/ICIP.2015.7350781.

3 Kyunghyun Cho and Xi Chen. Classifying and visualizing motion capture sequences using deep neural networks. *CoRR*, abs/1306.3874, 2013. URL: http://arxiv.org/abs/1306.3874.

4 Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*, pages 103–111, 2014. URL: http://aclweb.org/anthology/W/W14/W14-4012.pdf.

5 Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, oct 2014.

6 Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *ICLR 2017 conference*, 2017. URL: https://arxiv.org/abs/1609.01704.

7 Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

8 Timothy Dozat. Incorporating nesterov momentum into adam, 2015.

9 Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

10 J. P. Eckmann, Oliffson S. Kamphorst, and D. Ruelle. Recurrence plots of dynamical systems. *Europhysics Letters*, 4, nov 1987.

11 Mona Fathollahi and Rangachar Kasturi. Autonomous driving challenge: To infer the property of a dynamic object based on its motion pattern using recurrent neural network. *CoRR*, abs/1609.00361, 2016.

12 A. Graves, A. r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, May 2013. `doi:10.1109/ICASSP.2013.6638947`.

13 Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM. `doi:10.1145/1143844.1143891`.

14 Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278. IEEE, 2013. `doi:10.1109/asru.2013.6707742`.

15 Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, 2009. `doi:10.1109/TPAMI.2008.137`.

16 Michiel Hermans and Benjamin Schrauwen. Training and analysing deep recurrent neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 190–198. Curran Associates, Inc., 2013. URL: `http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf`.

17 S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer and Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

18 Ozan İrsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728, 2014. URL: `http://aclweb.org/anthology/D14-1080`.

19 Herbert Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems 15, NIPS 2002*, pages 593–600, 2002.

20 Herbert Jaeger and Harald Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 304(5667):78–80, 2004.

21 Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

22 Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2342–2350, Lille, France, 07–09 Jul 2015. PMLR. URL: `http://proceedings.mlr.press/v37/jozefowicz15.html`.

23 Marco Körner and Joachim Denzler. Analyzing the subspaces obtained by dimensionality reduction for human action recognition from 3d data. In *IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS)*, pages 130–135, 2012.

24 Yajie Miao and Florian Metze. On speaker adaptation of long short-term memory recurrent neural networks. In *in Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH) (To Appear). ISCA*, 2015.

25 M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn, June 2007.

26 Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

27 M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, nov 1997. `doi:10.1109/78.650093`.

28 Rajiv Shah and Rob Romijnders. Applying deep learning to basketball trajectories. *KDD 2016, Large Scale Sports Analytic Workshop*, 2016.

29 Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 843–852. JMLR Workshop and Conference Proceedings, 2015. URL: `http://jmlr.org/proceedings/papers/v37/srivastava15.pdf`.

30 Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, New York, NY, USA, 2011. ACM. URL: `http://www.icml-2011.org/papers/524_icmlpaper.pdf`.

31 Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL: `http://dl.acm.org/citation.cfm?id=2969033.2969173`.

32 Pattreeya Tanisaro and Gunther Heidemann. Time series classification using time warping invariant echo state networks. In *15th IEEE International Conference on Machine Learning and Applications, (ICMLA)*, 2016.

33 Pattreeya Tanisaro, Constantin Lehman, Leon Sütfeld, Gordon Pripa, and Gunther Heidemann. Classifying bio-inspired model in point-light human motion using echo state network. In *The 26th International Conference on Artificial Neural Networks (ICANN), 2017*, Lecture Notes in Computer Science. Springer, 2017.

34 Pattreeya Tanisaro, Florian Mahner, and Gunther Heidemann. Quasi view-independent human motion recognition in subspaces. In *Proceedings of 9th International Conference on Machine Learning and Computing (ICMLC)*, ICMLC 2017, pages 278–283. ACM, 2017. `doi:10.1145/3055635.3056577`.

35 Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. Two distributed-state models for generating high-dimensional time series. *J. Mach. Learn. Res.*, 12:1025–1068, 2011. URL: `http://dl.acm.org/citation.cfm?id=1953048.2021035`.

36 Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164, 2015.

37 Ronald J. Williams and David Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity, 1995.

38 Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3697–3703, 2016.

A Lists of Complete Actions

Table 1 Ten hand gestures in MHAD-10 from default view at 0°.

Action	#Trials	minLength	maxLength	meanLength
RHHighWave	30	55	161	85
RHCatch	30	38	75	54
RHHighThrow	30	44	78	56
RHDrawX	30	55	81	64
RHDrawTick	30	43	72	57
RHDrawCircle	30	54	89	67
RHHorizontalWave	30	52	139	70
RHForwardPunch	30	42	71	55
RHHammer	30	51	85	65
HandClap	30	47	68	57

B Confusion Matrices from Various Models of Unseen Subjects with Untrained Viewpoints

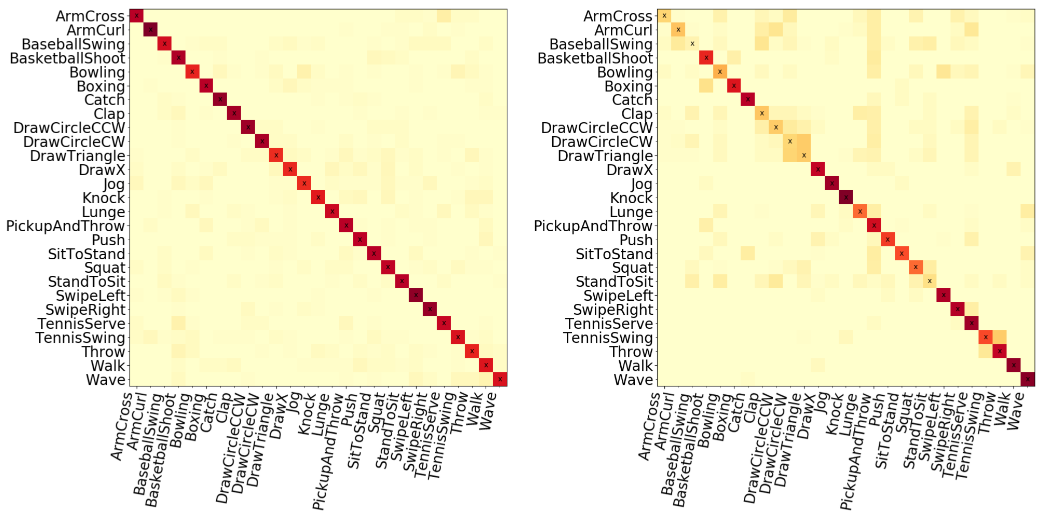

Figure 12 Confusion Matrices of MHAD-27 of the first testing fold from two classification methods. Left) Subspaces employing RF with PCA. Right) Majority vote using 1-NN.

Table 2 27 actions in MHAD-27 from default view at 0°.

Action	#Trials	minLength	maxLength	meanLength
ArmCross	32	50	86	64
ArmCurl,	32	42	86	59
BaseballSwing	32	63	90	74
BasketballShoot	32	46	81	60
Bowling	32	64	101	76
Boxing	32	51	92	68
Catch	32	41	74	59
Clap	32	51	78	61
DrawCircleCCW	32	64	98	74
DrawCircleCW	32	66	95	75
DrawTriangle	32	61	106	77
DrawX	31	55	81	66
Jog	32	51	82	67
Knock	32	53	95	67
Lunge	32	63	103	80
PickupAndThrow	32	68	125	87
Push	32	47	80	62
SitToStand	32	47	69	54
Squat	31	50	116	82
StandToSit	32	46	71	57
SwipeLeft	32	48	76	61
SwipeRight	32	47	75	59
TennisServe	32	52	94	67
TennisSwing	32	44	87	64
Throw	32	44	70	58
Walk	31	60	104	76
Wave	32	49	81	65

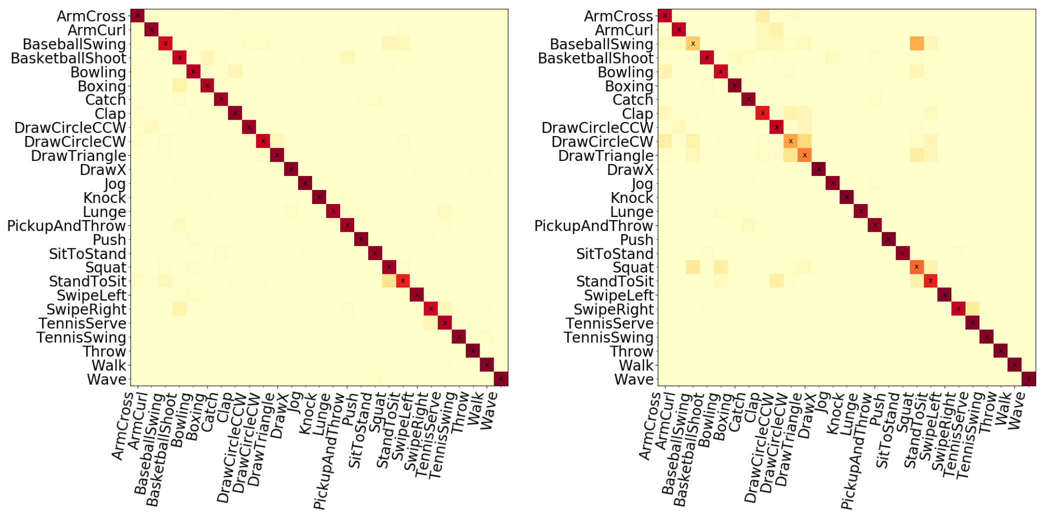

Figure 13 Confusion Matrices of MHAD-27 of 27 folds. Left) BRNN of $2 \cdot [300 \cdot 300]$ cells. Right) ESN with network size of 600 neurons.

Table 3 65 actions in HDM05 from default view at 0°.

Action	#Trials	minLength	maxLength	meanLength
cartwheel	28	281	701	450
clap	31	32	211	109
clapAboveHead	31	59	657	248
depositFloor	32	181	641	363
depositHigh	28	129	509	245
depositLow	28	196	481	277
depositMiddle	29	178	662	270
elbowToKnee	80	93	574	243
grabFloor	16	186	401	268
grabHigh	29	170	460	258
grabLowR	29	191	544	294
grabMiddle	28	112	352	210
hitHandHead	13	141	281	226
hopBothLegs	55	56	432	151
hopLLeg	64	62	254	119
hopRLeg	65	58	246	116
jogLeftCircle	32	197	400	292
jogOnPlaceStart	70	80	241	147
jogRightCircle	33	190	441	288
jumpDown	13	177	381	288
jumpingJack	65	116	484	201
kickLFront	43	129	841	294
kickLSide	39	131	721	315
kickRFront	45	121	668	296
kickRSide	44	127	740	310
lieDownFloor	20	301	901	655
punchLFront	45	119	761	263
punchLSide	45	90	721	235
punchRFront	45	138	761	286
punchRSide	42	97	662	242
rotateArmsBothBackward	32	62	649	214
rotateArmsBothForward	32	62	739	230
rotateArmsLBackward	32	57	708	215
rotateArmsLForward	32	55	739	215
rotateArmsRBackward	32	54	649	210
rotateArmsRForward	32	54	733	213
runOnPlaceStart	74	58	182	100
shuffleStepsStart	51	161	540	319
sitDownChair	20	154	441	318
sitDownFloor	20	224	601	407
sitDownKneelTieShoes	17	425	825	645
sitDownTable	20	162	401	270
skier	40	123	459	202
sneak	63	164	751	372
squat	65	136	823	271
staircaseDown	15	139	319	222
staircaseUp	27	164	444	292
standUpKneelToStand	17	100	301	182
standUpLieFloor	20	279	703	525
standUpSitChair	20	176	441	295
standUpSitFloor	20	167	641	403
standUpSitTable	20	121	454	250
throwBasketball	14	281	721	407
throwFarR	14	361	600	524
throwSitting	28	188	404	282
throwStanding	28	242	541	353
turnLeft	30	119	281	196
turnRight	30	135	260	196
walk	94	122	369	214
walkBackwards	30	158	433	299
walkLeft	32	277	659	411
walkLeftCircle	37	261	560	397
walkOnPlace	60	121	400	233
walkRightCircle	27	246	542	381
walkRightCrossFront	29	195	701	434

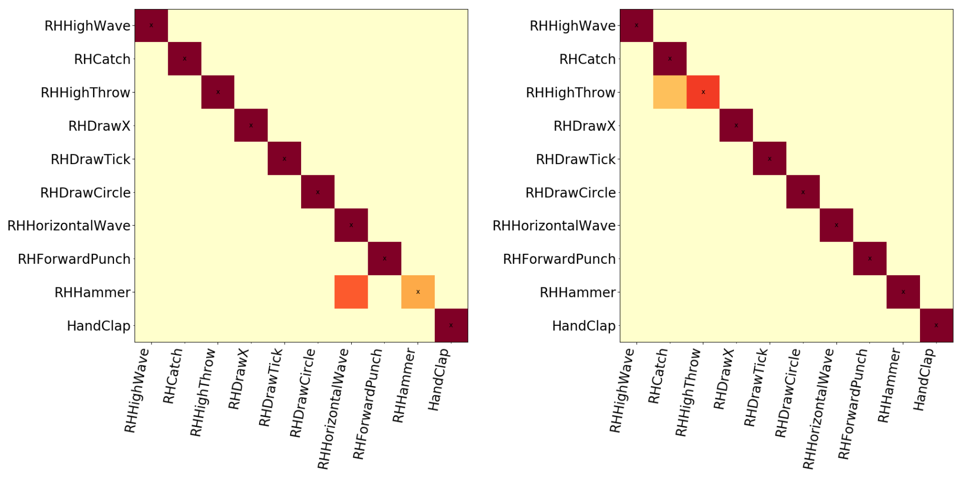

Figure 14 Confusion Matrices of MHAD-10 with the same configurations of BRNN employing $2 \cdot [50 \cdot 150 \cdot 250]$ cells, but testing on different subjects shown on the left and the right figure.

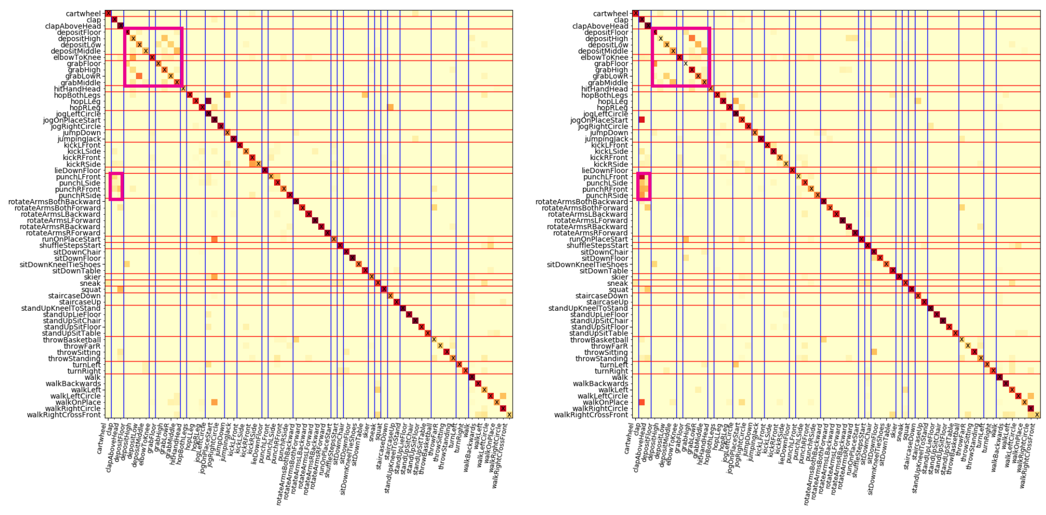

Figure 15 Confusion Matrices of HDM05 of all folds in each configuration. Two common misclassified groups of actions are highlighted in the pink color. Left) BRNN of $2 \cdot [200 \cdot 400]$ cells. Right) BRNN of $2 \cdot [250 \cdot 250]$ cells.