A Qualitative Comparison between Convolutional Neural Network and Large Scale Memory Storage and Retrieval Neural Network for Human Activity Recognition

Daniel Graupe, *Fellow, IEEE*, Arindam Bose, *Student Member, IEEE*

***Abstract*—Human activity understanding has received increasing attention in multimedia processing and interactions. Learning techniques using different neural networks often employed to recognize different activities. In this paper, we compare two methods for recognizing human activities using information sensed by an RGB-D camera, namely the Microsoft Kinect. This report focuses on developing and comparing the performance of Deep Convolution Neural model (CNN), Large Scale Memory Storage and Retrieval Neural model (LAMSTAR) and LAMSTAR II for automatic activity recognition. Experiments were performed on two public datasets: MSRDailyActivity3D and CAD-60. Experimental results indicate that the LAMSTAR networks outperform CNN and other approaches. The performance we achieved was the accuracy of 95.33% and 99.67% for LAMSTAR and LAMSTAR II respectively.**

***Index Terms*—Activity Recognition, Convolutional Neural Network, LAMSTAR, Microsoft Kinect Camera**

I. Introduction

HUMAN action recognition has been an active research topic in computer vision due to its wide range of applications such as human-machine interaction, choreography, sports, security surveillance, content-based retrieval, etc. Depending on the environment and circumstances, human activity may have different forms ranging from simple hand gestures to complex dancing activities. In the past decades, research on action recognition mainly focused on recognizing actions from conventional RGB videos or wearable sensors like Gyroscopes, Accelerometers etc. [1-4, 8-10, 15, 18, 22]. But the problem with wearable sensors are people often forget to wear the sensors. Also in public places it is always not possible to detect general activities by making people wear sensors in their body.

D. Graupe is Professor (emeritus) of Electrical & Computer Engineering Department and Adjunct Professor (emeritus) of Neurology & Rehabilitation Medicine, University of Illinois at Chicago, Chicago, IL 60607, USA. (e-mail: dangraupe@gmail.com).

A. Bose is a PhD student of Electrical & Computer Engineering Department, University of Illinois at Chicago, Chicago, IL 60607, USA (e-mail: abose4@uic.edu).

Recently, the release of the Microsoft Kinect brings up new opportunities in this field. The Kinect device can provide both depth maps and RGB images in real-time at low cost. Depth maps have several advantages compared to traditional color images. For example, depth maps reflect pure geometry and shape cues, which can often be more discriminative than color and texture in many problems including object segmentation and detection. Moreover, depth maps are insensitive to changes in lighting conditions. Based on depth data, many works [12-13, 19, 24-26, 28, 30-42] have been reported with respect to specific feature descriptors to take advantage of the properties of depth maps.

In this paper, we focus on reliably detecting daily activities that a person performs in their house, such as walking, sitting, sleeping, eating etc. based on angular poses of the body parts using two popular Deep Learning method such as Convolutional Neural Network (CNN) and LArge Memory STorage And Retrieval (LAMSTAR). Deep Convolutional Neural Networks have been demonstrated as an effective class of models for understanding image content, offering state-of-the-art results on image recognition, segmentation, detection and retrieval [6, 11-14, 17-18]. The key enabling factors behind these successes are techniques for scaling up the networks to millions of parameters and massive labelled datasets that can support the learning process. This is a huge problem to gather such an amount of training sets for every other problem. Also CNN takes a lot of time to train such enormous amount of parameters.

In this work, we propose to apply LAMSTAR to depth map sequences for action recognition. LAMSTAR attempts to imitate, in a gross manner, processes of the human central nervous system (CNS), concerning storage and retrieval of patterns, impressions and sensed observations, including processes of forgetting and of recall. It attempts to achieve this without contradicting findings from physiological and psychological observations, at least in an input/output manner. Furthermore, the LAMSTAR model considered attempts to do so in a computationally efficient manner, using tools of neural networks, especially SOM (Self Organizing Map)-based network modules combined with statistical decision tools. The LAMSTAR network is therefore not a specific network but a system of networks for storage, recognition, comparison and

decision that facilitates such storage and retrieval to be accomplished. It combines Kohonen's WTA (Winner-Take-All) principle as in [43], with Link Weights in the sense of Hebb's principle [44] and of the Pavlovian dog [45] example. These Link Weights thus allow efficient integration of many SOM layers in the LAMSTAR network.

We evaluate our method on the MSRDailyActivity3D and CAD-60 datasets individually and achieve results which are better than CNN or other state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 reviews the related works on deep learning for action recognition using depth sequences. Section 3 is a brief introduction of Section 4, 5 and 6 where CNN, LAMSTAR and LAMSTAR II has been described respectively in detail. In Section 7, experimental setup, various experimental results and analysis are presented. Conclusion and future work are made in Section 8.

II. Related Works

With the recent resurgence of neural networks, deep neural architectures have been used as an effective solution for extracting high level features from data. There are a number of attempts to apply deep architectures for 2D video recognition.

In [11], a 3D CNN model is used to extract features from both spatial and temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. In [17], a deep convolutional neural network is trained to classify images of humans based on the activity the person is performing. In [21] a position and velocity path characteristic is developed for the pedestrians using a Kalman filter.

For depth-based action recognition, many works have been reported in the past few years. In [31] differences between projected depth maps are stacked as DMM and then use HOG to extract the features on the DMM. This method transforms the problem of action recognition from 3D space to 2D space. In [33], HON4D is proposed, in which surface normal is extended to 4D space and quantized by regular polychorons. Following this method, Yang and Tian [34] cluster hypersurface normals and form the polynormal which can be used to jointly capture the local motion and geometry information. Super Normal Vector (SNV) is generated by aggregating the lowlevel polynormals. However, all of these methods are based on carefully hand designed features, which are restricted to specific datasets and applications. In [13] a framework called Hierarchical Depth Motion Maps (HDMM) + 3 Channel Deep Convolutional Neural Networks (3ConvNets) is proposed.

On the other hand [24, 25] are based on a hierarchical maximum entropy Markov model (MEMM). It considers a person's activity as composed of a set of sub-activities, and infers the two-layered graph structure using a dynamic programming approach. In [35], a discriminative representation of structure-motion features based on skeleton joints is proposed to detect abnormal activities. In [36], Yang and Tian design an action feature descriptor for action recognition based on differences of skeleton joints, i.e., EigenJoints which combine action information including static posture, motion property, and overall dynamics. Accumulated Motion Energy (AME) is then proposed to perform informative frame selection, which is able to remove noisy frames and reduce computational cost. They employ non-parametric Naïve-Bayes-Nearest-Neighbor (NBNN) to classify multiple actions. In [37], a 3-D spatial and temporal contexts among human subjects or objects are extracted by integrating information from both grayscale and depth images and a latent structural model is developed to integrate the information from multiple levels of video processing for an activity detection. In [40] given an unknown action, Shan et al. extract and classify all its constituent atomic actions and then assign the action label via an equal voting scheme. In [42] the model consists of self-organizing Growing-When-Required (GWR) networks that obtain progressively generalized representations of sensory inputs and learn inherent spatio-temporal dependencies. During the training, the GWR networks dynamically change their topological structure to better match the input space. First pose and motion features are extracted from video sequences and then cluster actions in terms of prototypical pose-motion trajectories. Multi-cue trajectories from matching action frames are subsequently combined to provide action dynamics in the joint feature space.

III. The Human Activity Recognition

We use several supervised learning approach in which we collected ground-truth labeled data for training our model. Our input is RGBD images from a Kinect sensor, from which we extract certain features that are fed as input to the learning algorithms. We train CNN, LAMSTAR and LAMSTAR II model which will capture different properties of human activities, including their hierarchical nature and the transitions between sub-activities over time. A brief description of these networks are described in following sections.

IV. Convolutional Neural Network

Convolutional neural networks have great potential to identify the various salient patterns of HAR's data signals. Specifically, the processing units in the lower layers obtain the local salience of the signals (to characterize the nature of each basic movement in a human activity). The processing units in the higher layers obtain the salient patterns of signals at high-level representation (to characterize the salience of a combination of several basic movements). Note that each layer may have a number of convolution or pooling operators (specified by different parameters) as described below, so multiple salient patterns learned from different aspects are jointly considered in the CNN.

We start with the notations used in the CNN. Specifically, an instance used by the CNN is an 8x8 two-dimensional matrix containing a single pose for a certain activity. For training data, the true label of the matrix instance is determined by the label of most-closely related activity. For

the jth feature map in the ith layer of the CNN, it is also a matrix, and the value at the xth row for sensor d is denoted as $v_{ij}^{x,d}$ for convenience.

In the convolution layers, the previous layer's feature maps are convolved with several convolutional kernels (to be learned in the training process). The output of the convolution operators added by a bias (to be learned) is put through the activation function to form the feature map for the next layer. Formally, the value $v_{ij}^{x,d}$ is given by

$$v_{ij}^{x,d} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1} w_{ijm}^{p} v_{(i-1)m}^{x+p,d}\right), \quad \forall d = 1, \dots, D \tag{1}$$

where $\tanh(\cdot)$ is the hyperbolic tangent function, b_{ij} is the bias for this feature map, m indexes over the set of feature maps in the $(i-1)$th layer connected to the current feature map, w_{ijm}^{p} is the value at the position p of the convolutional kernel, and P_i is the length of the convolutional kernel.

In the pooling layers, the resolution of feature maps is reduced to increase the invariance of features to distortions on the inputs. Specifically, feature maps in the previous layer are pooled over local temporal neighborhood by either max pooling function

$$v_{ij}^{x,d} = \max_{1\le q\le Q_i}\left(v_{(i-1)j}^{x+p,d}\right), \forall d = 1, \dots, D \tag{2}$$

or a sum pooling function

$$v_{ij}^{x,d} = \frac{1}{Q_i}\sum_{1\le q\le Q_i}\left(v_{(i-1)j}^{x+p,d}\right), \forall d = 1, \dots, D \tag{3}$$

where Q_i is the length of the pooling region.

V. LAMSTAR

The LAMSTAR neural network is specifically designed for application to retrieval, diagnosis, classification, prediction and decision problems which involve a very large number of categories. The resulting LAMSTAR neural network [5] is designed to store and retrieve patterns in a computationally efficient manner, using tools of neural networks, especially Kohonen's SOM (Self Organizing Map)-based network modules [43] combined with statistical decision tools. The basic storage modules of the LAMSTAR network are modified Kohonen networks. SOM modules are Associate-Memory-based Winner-Take-All (WTA). The information is stored and processed via correlation links between individual neurons in separate SOM modules in the LAMSTAR network. Its ability to deal with a large number of categories is partly due to its use of simple calculation of link weights. The link weights are the main engine of the network, connecting many layers of SOM modules such that the emphasis is on correlation of link weights between atoms of memory, not on the memory atoms themselves. Like this, the design becomes closer to knowledge processing in the biological CNS. The forgetting feature is a basic feature of biological networks whose efficiency depends on it, as is the ability to deal with incomplete data sets. Its elementary neural unit or cell (neuron) is the one employed in all neural networks. Accordingly, if the p inputs into a given neuron at the jth SOM layer are denoted as $x_{ij};\ i = 1,2, \dots, p$, and if the (single) output of that neuron is denoted as y, then the neuron's output y satisfies:

$$y = f\left[\sum_{i=1}^{p} w_{ij}x_{ij}\right] \tag{4}$$

where $f[\cdot]$ is a nonlinear function denoted as Activation function.

The Winner-Take-All (WTA) principle, is employed such that an output is produced only at the winning neuron, namely, at the output of the neuron whose storage weights w_{ij} are closest to vector x_j when a best-matching memory is sought at the jth SOM module. By using a link weights structure for its decision and browsing, the LAMSTAR network utilizes not just the stored memory values w_{ij} as in other neural networks, but also the interrelations these memories to the decision module and between the memories themselves.

These relations are fundamental to its operation. By Hebb's Law link weights adjust and serve to establish flow of neuronal signal traffic between groups of neurons, such that when a certain neuron fires very often in close time, then the interconnecting link-weights relating to that traffic, increase as compared to other interconnections. Link weights serve as Hebbian intersynaptic weights and adjust accordingly.

A. Design of the network

The LAMSTAR network has the following components:

1) Input word and its subwords: The input word is divided into a number subwords. The size and the number of subwords will depend on the size of dataset and size of the input sequence. Each subword represents an attribute of the input word.
2) SOM modules for storing input subwords: For every subword there is an associated Self Organizing Map (SOM) module with neurons that are designed to function as Kohonen 'Winner-Take-All' neurons where the winning neuron has an output of 1 while all other neurons in that SOM module have a zero output. In this project, the SOM modules are built dynamically in the sense that instead of setting the number of neurons at some fixed value arbitrarily, the network was built to have neurons depending on the class to which a given input to a particular subword might belong. For example if there are two subwords having same values, then these would fire the same neuron in their SOM layer and hence all they need is one neuron in the place of two neurons. This way the network is designed with lesser number of neurons and the time taken to fire a particular neuron at the classification stage is reduced considerably. A generalized block diagram of LAMSTAR can be found at Fig. 1. Information in the LAMSTAR system is mapped via link

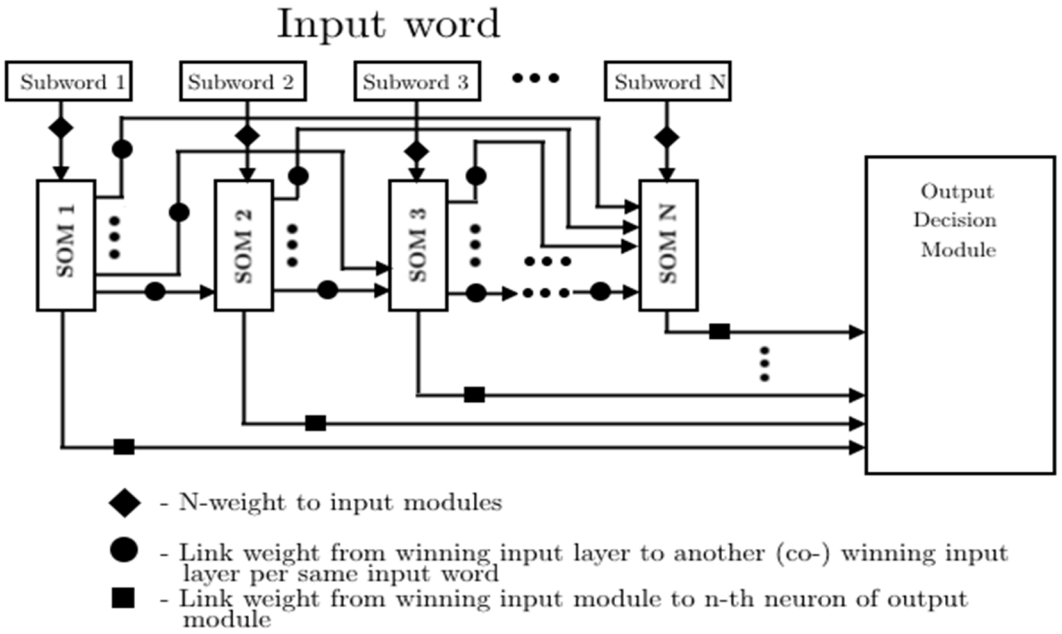

Fig. 1. Block Diagram of LAMSTAR network

weights $L_{i,j}$ (Fig. 1) between individual neurons in different SOM modules. The LAMSTAR system does not create neurons for an entire input word. Instead, only selected subwords are stored in Associative-Memory-like manner in SOM modules (w weights), and correlations between subwords are stored in terms of creating/adjusting L-links that connect neurons in different SOM modules. This allows the LAMSTAR network to be trained with partially incomplete data sets. The L-links are fundamental to allow interpolation and extrapolation of patterns (when a neuron in an SOM model does not correspond to an input subword but is highly linked to other modules serves as an interpolated estimate). We comment that the updating of Link Weights, as considered in this sub-section, applies to both link weights between input-storage SOM modules AND also link-weights from any storage SOM module and an output module (layer). In most applications it is advisable and economical to consider only links to output (decision) modules.

Specifically, link weight values L are updated such that for a given input word, after determining a winning kth neuron in module i and a winning mth neuron in module j, then the link weight $L_{i,j}^{k,m}$ is counted up by a reward increment ΔL, whereas, all other links $L_{i,j}^{s,v}$ may be reduced by a punishment increment ΔM [5]. The values of L-link weights are modified according to:

$$L_{i,j}^{k,m}(t+1) = L_{i,j}^{k,m}(t) + \Delta L \;:\; L_{i,j}^{k,m} \le L_{\max} \quad (5.a)$$

$$L_{i,j}^{s,v}(t+1) = L_{i,j}^{s,v}(t) - \Delta M \quad (5.b)$$

$$L(0) = 0 \quad (5.c)$$

where:

$L_{i,j}^{k,m}$: links between winning neuron i in kth module and winning neuron j in mth module (which may also be the mth output module).

ΔL, ΔM: reward/punishment increment values (pre - determined fixed values). It is sometimes desirable to set ΔM (either for all LAMSTAR decisions or only when the decision is correct) as: $\Delta M = 0$.

$L_{\max}$: maximal links value.

3) Output layer: The output layer is designed to have several neurons, which have the neuron firing patterns which depends on the actual class description of the input sequence. The link-weights from the input SOM modules to the output decision layer are adjusted during training on a reward/punishment principle. Furthermore, they continue being trained during normal operational runs.

VI. LAMSTAR II

The LAMSTAR II [8] (also known as the modified LAMSTAR or normalized LAMSTAR) is functionally almost same as LAMSTAR. The primary difference between the LAMSTAR and LAMSTAR II is realized in the interpretation of the link weights that connect winning neurons to decision modules. The LAMSTAR produces a winning decision v from the J output (decision) neurons in the decision self-organizing-map (SOM) module by considering the sum of link weights that connect the winning neuron w, in each k of the K input-SOM (memory-storage) modules,

$$E(j) = \sum_{k \in K} L_{w,j}^{k} \qquad ,\forall j \in J \quad (6)$$

$$E(v) \ge E(j) \qquad ,\forall j \in J \quad (7)$$

When feedback as to the correct decision becomes available, the LAMSTAR updates the link weight values associated with the decision making process through (5.a) and

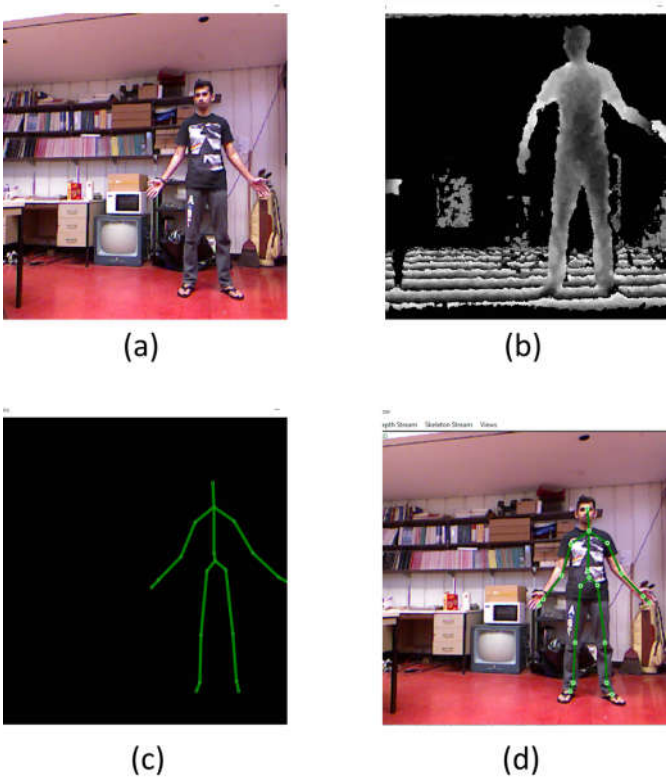

Fig. 2. Output images from Kinect: (a) RGB image, (b) Depth map (c) Skeleton image, (d) Skeleton + RGB image.

(5.b). (5.a) is always performed on the link weights that connect winning neurons to the proper decision. All other link weights connecting the winning neurons to the incorrect decisions receive negative reinforcement according to (5.b).

In contrast, the LAMSTAR II does not consider the link weights directly, but instead considers a function of the link weights as shown in (8) and (9) to choose the winning output neuron (decision) v as in (10).

$$L_{i,j}^{k} = \frac{L_{i,j}^{k}}{\sum_{j \in J} L_{i,j}^{k}} \tag{8}$$

$$E(j) = f\left(l_{w,j}^{1}, l_{w,j}^{2}, \ldots, l_{w,j}^{K}\right), \ \forall j \in J, \quad f: \Re_{+}^{K} \rightarrow \Re_{+} \tag{9}$$

$$E(v) \geq E(j) \quad , \ \forall j \in J \tag{10}$$

To ensure that the normalized link weights found in (8) are non-negative, the un-normalized L-weights have a minimum value of zero as in (5.c). Thus, (9) is a general mapping from a K-tuple of non-negative real numbers to a non-negative real number. The only restriction on this function is that it must be non-decreasing in each member of the K-tuple.

Additionally, when feedback of the correct decision is provided to the network, the LAMSTAR II only provides negative reinforcement according to (6.b) upon the production of an incorrect decision by the network. This seemingly slight change to the LAMSTAR network enables the definition of the Confidence Measure (CM) (11), as well as ensures that the network performs well when additional inputs are added.

$$CM = \max_{j \in J}\left(E(v) - E(j)\right) \tag{11}$$

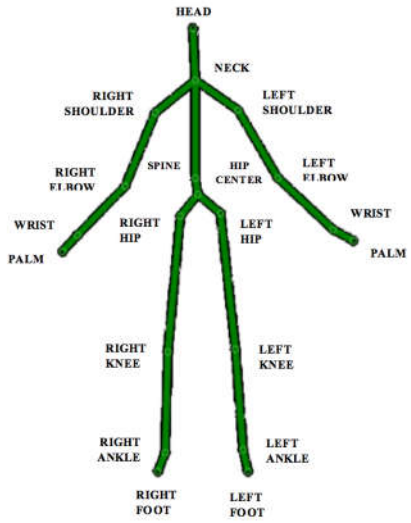

Fig. 3. Skeleton body-joints given by Kinect.

VII. Experiments and Results

A. Data Preprocessing

We can recognize a person's activity by looking at his current pose and movement over time, as captured by a set of features. The input sensor model is a RGBD camera (Kinect) that gives us an RGB image as well as depths at each pixel. In order to compute the human pose features, we describe a person by a rigid skeleton that can move at twenty joints (see Fig. 2 and Fig. 3). We extract this skeleton using a tracking system provided by PrimeSense. The skeleton is described by the length of the links and the joint angles. Specifically, we have the three-dimensional Euclidean coordinates of each joint and the orientation matrix of each joint with respect to the sensor. We compute features from this data as follows. In this project we used 3D Euclidean coordinates of twenty joints viz. 1. Hip Center, 2. Spine, 3. Shoulder Center, 4. Head, 5. Shoulder Right, 6. Elbow Right, 7. Wrist Right, 8. Hand Right, 9. Shoulder Left, 10. Elbow Left, 11. Wrist Left, 12. Hand Left, 13. Hip Right, 14. Knee Right, 15. Ankle Right, 16. Foot Right, 17. Hip Left, 18. Knee Left, 19. Ankle Left, 20. Foot Left. These skeleton joints coordinates are measured with respect to the Kinect which makes the whole system to be inconsistent because position and orientation of Kinect camera will play major role in defining the training samples. So we convert the whole coordinate system of the Kinect coordinates with respect to the Spine joint. This process is called Normalization of Kinect coordinates. We use the geometrical transformations for normalizing skeletal data.

We have used 3D transformations throughout our work to transform different body joints, their locations and orientations in the coordinate system of the Kinect to the body-centered coordinate system. The main idea behind computing the body centered 3D orientations and locations of the different body joints is to achieve a view invariant activity recognition system. When changing the point of view in a 3D geometry system, we rotate and translate each point according to the current position and orientation of the person doing the

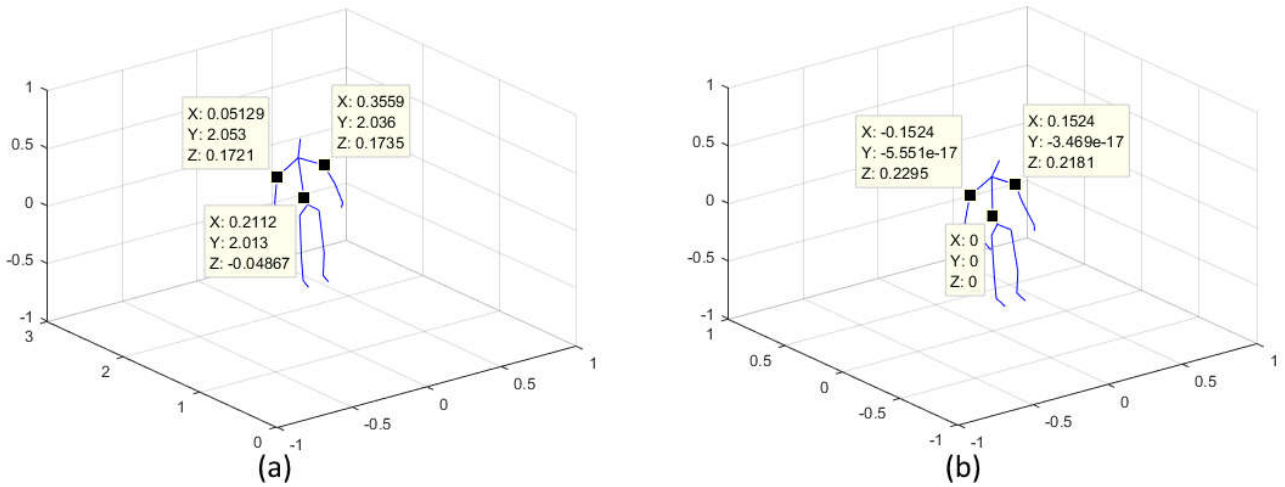

Fig. 4. Normalization of skeleton: (a) Raw skeleton from Kinect, (b) Skeleton after normalization. Note that the spine coordinate of normalized skeleton has been made the origin (0, 0, 0)

viewing or in our case the person performing the activity. This computation is done by multiplying the locations of the different body joints in the Kinect coordinate system by a rotation matrix (R) and a translation function ($\boldsymbol{t}$), which can be mathematically written as:

$$\boldsymbol{x}' = R\boldsymbol{x} + \boldsymbol{t} \quad \text{or} \quad \boldsymbol{x}' = [R \quad \boldsymbol{t}]\overline{\boldsymbol{x}} \tag{12}$$

where $\boldsymbol{x}$ and $\overline{\boldsymbol{x}}$ are the point coordinates of the locations of the different body joints in the Kinect coordinate system represented as homogenous coordinates such that, $\boldsymbol{x} = (x, y, z) \in R^3$, or as an augmented vector $\overline{\boldsymbol{x}} = (x, y, z, 1)$. $\boldsymbol{x}'$ is the set of coordinates of the body joints in three dimensions with respect to the body centered coordinate system.

The rotation matrix R is a matrix, which is used to rotate a vector while preserving its length. This is simply done by multiplying the rotation matrix with the vector. The rotation matrix is a 3×3 orthonormal matrix with $RR^T = I$ and $|R| = 1$. We usually refer the rotation matrix as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{13}$$

Translation is a function used to move every point to a constant distance in a specified direction. Translation can also be interpreted as the addition of a constant vector to every point, or shifting the origin of the coordinate system. To translate a three-dimensional object by a vector $\boldsymbol{V}$, each homogenous vector $\boldsymbol{p}$ of the object can be multiplied by a translational matrix:

$$\boldsymbol{V} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

The multiplication $\boldsymbol{V} \cdot \boldsymbol{p}$, will give us:

$$\boldsymbol{V} \cdot \boldsymbol{p} = \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} p_x + v_x \\ p_y + v_y \\ p_z + v_z \\ 1 \end{bmatrix} \tag{15}$$

Various methods are used to calculate rotations in 3D; some of the most common ones are Euler angles, Axis-Angle, and Quaternions. Each of these transformations has their own advantages and disadvantages. In our normalization approach we used Euler angles. Euler angles are the most intuitive and commonly used transformation of 3D rotation. According to Euler, we can perform any rotation in 3D space as a sequence of 3 individual rotations around 3 orthogonal coordinate axes. We can parameterize rotation in 3D by three angles since rotation around a single axis can be easily measured by a single angle. The Euler angle rotation convention can be performed as follows:

1) Rotate about one body coordinate axis, say z.
2) Rotate about a second body coordinate axis not identical to the first, say x'. The x' axis will be the new x-axis, obtained after the first rotation.
3) Lastly, rotate about another body coordinate axis not identical to the second, say z''. The z'' axis is the new z-axis, obtained after the first and second rotations.

Let the Euler angles with respect to X-axis, Y-axis and Z-axis be denoted by α, β and θ respectively, the corresponding homogenous rotation matrices are given as:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16.a}$$

$$R_y = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16.b}$$

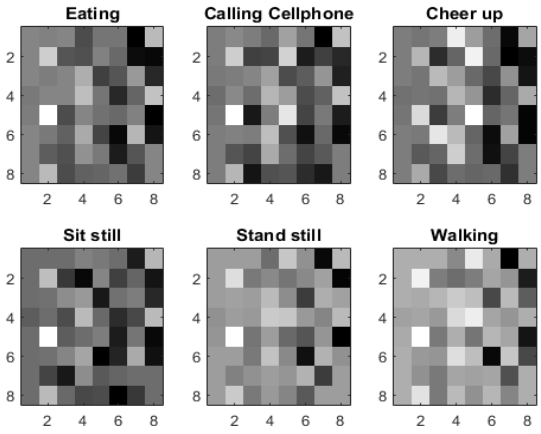

Fig. 5. Image representation of ADLs

$$R_z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16.c}$$

Here, the positive angles are measured clockwise while looking towards the origin.

Euler angles are intuitive to use and have a compact representation than others. An important advantage of Euler angles parameterization is that the user is sure to get a valid rotation for any specified value of angles. However, they suffer from a situation called Gimbal lock, which occurs when one of the Euler angles approaches 90°. Two of the rotational frames combine together as a result, hence losing one degree of rotation. In the worst-case scenarios, all three rotational frames combine into one resulting in only one degree of rotation. Another problem with Euler angles is that they cannot be interpolated in a meaningful way. Euler angles do not provide independent controls for controlling a rotation, if one the angles is changed the meaning of subsequent angles is altered. Fig. 4 depicts the raw skeleton and the skeleton after normalization.

After normalization we get 60 coordinates from 20 body joints for each frame. We add two zeros in the front and two zeros in the end to make it a 64 length vector so that we can make it 8x8 image like matrices. Fig. 5 depicts input data for different activities. We now apply this data to different neural networks.

B. CNN Network Structure and Process Description

In this section we discuss the training and testing process associated with CNN network we used in our work.

Every layer of a CNN transforms one volume of activations to another through a differentiable function. We use three main types of layers to build CNN architectures: Convolutional Layer (CONV), Pooling Layer (POOL), and Fully-Connected Layer (FC). Following are some details of different layer. In our work the CNN consists of simple architecture with 4 layers [INPUT - CONV - ReLU - POOL - FC] as in Fig. 6.

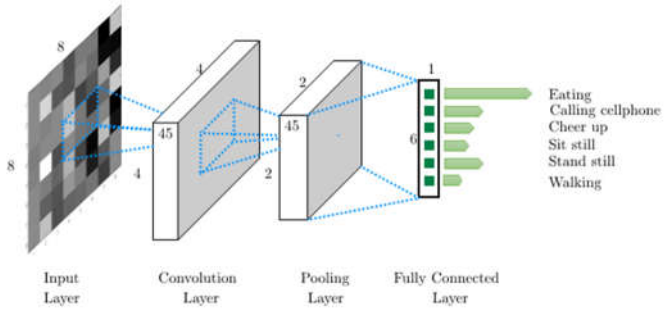

Fig. 6. Illustration of CNN layers

INPUT will hold the raw values (after normalization and resizing to 8x8) of the coordinate image, in our case an image of width 8, height 8, and with one gray channel.

CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and the region they are connected to in the input volume.

ReLU (rectified linear unit) layer will apply an element-wise non-saturating activation function, such as the $f(x) = \max(0, x)$ thresholding at zero. This leaves the size of the volume unchanged. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. Other functions are also used to increase nonlinearity. For example the saturating hyperbolic tangent, $f(x) = \tanh(x)$, $f(x) = |\tanh(x)|$, and the sigmoid function $f(x) = 1/((1 + e^{-x}))$. In our work we used $f(x) = \max(0, x)$ thresholding at zero.

POOL layer will perform a down-sampling operation along the spatial dimensions (width, height), resulting in 3rd dimension as volume. In order to reduce variance, pooling layers compute the max or average value of a particular feature over a region of the image. This will ensure that the same result will be obtained, even when image features have small translations. This is an important operation for object classification and detection. In our work we used max pooling.

FC (Fully-Connected) layer will compute the class scores, resulting in volume of size where each of the class numbers correspond to a class score. This layer is same as a standard multilayer perceptron neural network that maps the latent features into the output classes. The output of this layer is governed by the softmax function

$$v_{ij} = \frac{\exp\left(v_{(i-1)j}\right)}{\sum_{j=1}^{C} \exp\left(v_{(i-1)j}\right)} \tag{17}$$

where C is the number of output classes. This softmax function provides the posterior probability of the classification results. Then, an entropy cost function can be constituted based on the true labels of training instances and probabilistic outputs of softmax function.

In this way, CNN transform the original image layer by layer from the original pixel values to the final class scores. Some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons).

TABLE I
FIRING ORDER OF THE OUTPUT NEURONS

Activity Class	N_1[a]	N_2[a]	N_3[a]	N_4[a]	N_5[a]	N_6[a]
Class 1	Fired	Not Fired	Not Fired	Not Fired	Not Fired	Not Fired
Class 2	Not Fired	Fired	Not Fired	Not Fired	Not Fired	Not Fired
Class 3	Not Fired	Not Fired	Fired	Not Fired	Not Fired	Not Fired
Class 4	Not Fired	Not Fired	Not Fired	Fired	Not Fired	Not Fired
Class 5	Not Fired	Not Fired	Not Fired	Not Fired	Fired	Not Fired
Class 6	Not Fired	Not Fired	Not Fired	Not Fired	Not Fired	Fired

[a]N_i = ith output neuron

On the other hand, the ReLU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the CNN computes are consistent with the labels in the training set for each image.

C. LAMSTAR and LAMSTAR II Network Structure and Process Description

In this section we discuss the training and testing process associated with LAMSTAR networks. We also describe the structure of the LAMSTAR network used in our work.

1) Training Process

The training of the LAMSTAR network if performed as follows:

1.1) Subword Formation: The input patterns are to be divided into subwords before training/testing the LAMSTAR network. In order to perform this, the every row of the input 8x8 sequence is read to make 8 subwords followed by every column to make another 8 subwords resulting in a total of 16 subwords.

1.2) Input Normalization: Each subwords of every input pattern is normalized as follows:

$$x_i' = \frac{x_i}{\sqrt{\sum x_j^2}} \tag{18}$$

where, x — subword of an input pattern. During the process, those subwords, which are all zeros, are identified and their normalized values are manually set to zero.

1.3) Dynamic Neuron formation in the SOM modules: The first neuron in all the SOM modules are constructed as Kohonen neurons as follows. As the first pattern is input to the system, one neuron is built with 8 inputs and random weights to start with initially and they are also normalized just like the input subwords. Then the weights are adjusted such that the output of this neuron is made equal to 1 (with a tolerance of 10^{-5} according to the formula:

$$w(n+1) = w(n) + \alpha * (x - w(n)) \tag{19}$$

where,

α — learning constant = 0.8

w — weight at the input of the neuron

x — subword

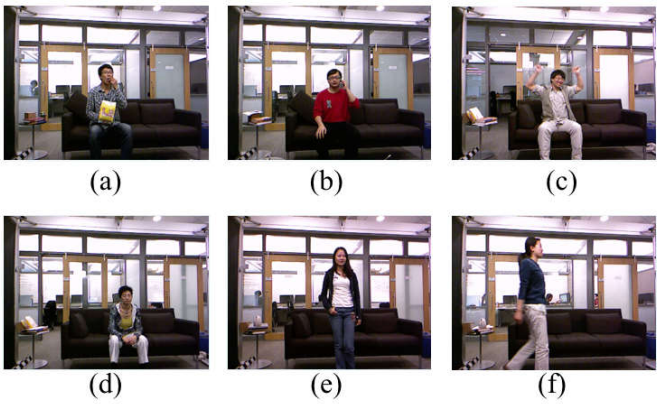

Fig. 7. Activity poses from MSRDailyActivity3D dataset: (a) Eating, (b) Calling cellphone, (c) Cheering up, (d) Sitting still, (e) Standing still, (f) Walking.

$$z = w * x$$

where, z — output of the neuron (in the case of the first neuron it is made equal to 1).

When the subwords of the subsequent patterns is input to the respective modules, the output at any of the previously built neuron is checked to see if it is close to 1 with a certain threshold. This threshold is chosen carefully. The higher threshold we set, the better the training will be but with the more the training time. This threshold defines how close to 1 we can reach without creating a new neuron. If one of the neurons satisfies the condition, then this is declared as the winning neuron, i.e., a neuron whose weights closely resemble the input pattern. Else another neuron is built with new sets of weights that are normalized and adjusted as above to resemble the input subword.

During this process, if there is a subword with all zeros then this will not contribute to a change in the output and hence the output is made to zero and the process of finding a winning neuron is bypassed for such a case.

1.4) Desired neuron firing pattern: The output neuron firing pattern for each character in the training set has been established as given in Table I.

1.5) Link weights: Link weights are defined as the weights that come from the winning neuron at every module to the 6 output neurons. If in the desired firing, a neuron is to be fired, then its corresponding link weights are rewarded by adding a small positive value of 5. On the other hand, if a neuron should not be fired then its link weights are reduced by 5. This will result in the summed link weights at the output layer being a positive value indicating a fired neuron if the neuron has to be fired for the pattern and high negative value if it should not be fired. The weights at the SOM neuron modules and the link weights are stored.

2) Testing Process

The LAMSTAR network was tested with test patterns are detailed in this section. The patterns are processed to get 16 subwords as before. Normalization is done for the subwords as explained in the training. The stored weights are loaded. The subwords are propagated through the network and the neuron with the maximum output at the Kohonen layer is found and

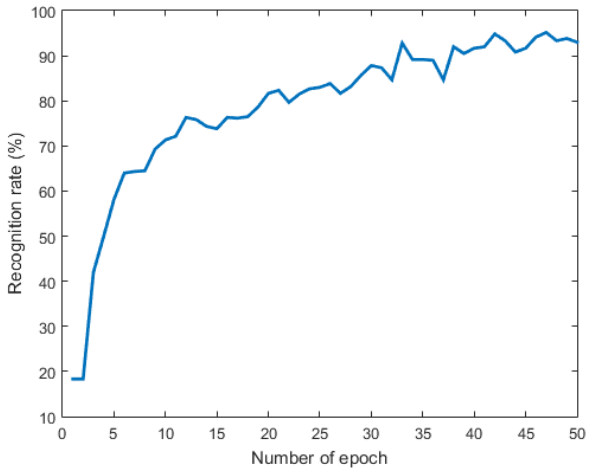

Fig. 8. CNN result: Recognition rate vs. number of epoch graph. The recognition rate increases with number of epochs and gets converged after 45th epoch.

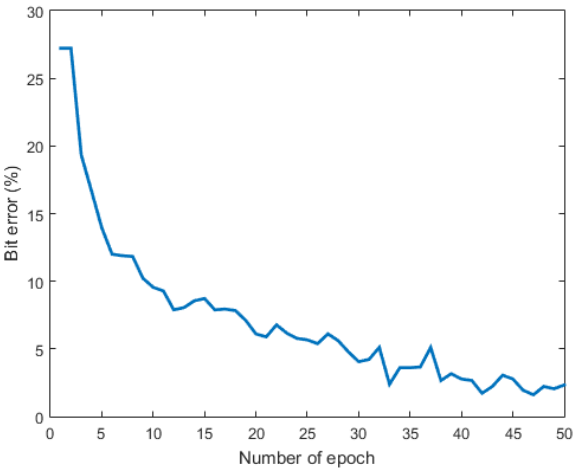

Fig. 9. CNN result: Bit error rate vs. number of epoch graph. The bit error rate decreases with number of epochs and gets converged after 45th epoch.

their link weights are sent to the output neurons. The output is a sum of all the link weights.

D. Dataset

We consider two datasets for human activity recognition: 1) the MSRDailyActivity3D[1] from the Microsoft Research Laboratory and 2) CAD-60[2] from Computer Science Department, Cornell University.

1) The MSRDailyActivity3D ADL dataset is a daily activity dataset captured by a Kinect device in a home environment by 10 subjects. There are 16 activity types: drinking, eating, reading book, calling cellphone, writing on a paper, using laptop, using vacuum cleaner, cheering up, sitting still, tossing paper, playing game, laying down on sofa, walking, playing guitar, standing up, sitting down. If possible, each subject performs an activity in two different poses: "sitting on sofa" and "standing". So the total number of the activity sequences is "320". Also each sequence contains different number poses ranging from 60-300. For our purpose we chose six different activities: (1) Eating, (2) Calling cellphone, (3) Cheering up, (4) Sitting still, (5) Standing still and (6) Walking (see Fig. 7). We chose 7590 different poses from these six activity sets for our training set and 600 different poses for testing set.
2) The CAD-60 dataset is captured in office, kitchen, bedroom, bathroom, and living room environment by four subjects. There are 12 activity types: rinsing mouth, brushing teeth, wearing contact lens, talking on the phone, drinking water, opening pill container, cooking (chopping), cooking (stirring), talking on couch, relaxing on couch, writing on whiteboard, working on computer. For our purpose we selected several poses from 5 ADLs which are: (1) Brushing teeth, (2) talking on the phone, (3) drinking water, (4) cooking (chopping) and (5) working on computer. This dataset was used to compare our results with other state-of-the-art recognition methods used for HAR.

For the Convolution Neural Network I used the DeepLearnToolbox[3] for CNN written in MATLAB. DeepLearnToolbox is a MATLAB/Octave toolbox for deep learning which Includes Deep Belief Nets, Stacked Autoencoders, Convolutional Neural Nets, Convolutional Autoencoders and vanilla Neural Nets.

TABLE II
CONFUSION MATRIX OF THE ACTIVITIES ON THE MSRDAILYACTIVITY3D DATASET USING CNN

	Predicted Class						
Actual Class	ET	CC	CU	SiS	StS	WK	Recall (%)
ET	84	9	0	4	0	3	84
CC	0	96	0	4	0	0	96
CU	0	0	100	0	0	0	100
SiS	0	9	0	91	0	0	91
StS	0	0	0	0	100	0	100
WK	0	0	0	0	13	87	87
Precision (%)	100	84	100	92	88	96	

ET = Eating, CC = Caling cellphone, CU = Cheering up, SiS = Sitting still, StS = Standing still, WK = Walking

E. Experimental Results

In this section we describe the experimental results of our CNN and LAMSTAR networks for MSRDailyActivity3D dataset. Then we compare our result with other state-of-the-art methods for MSRDailyActivity3D and CAD-60 dataset.

1) CNN Results: The results of the proposed CNN method on MSRDailyActivity3D dataset and are shown in Fig. 8 and Fig. 9. Fig. 8 depicts the recognition rate (%) and Fig. 9 shows the bit error rate (%) against the number of epochs. We tested our CNN for 50 epochs for all 600 test poses. After reaching epoch 45 we got convergence. So after epoch 50 our maximum recognition rate was 93% and bit error rate was 2.333%. By bit error rate we mean

[1]Downloaded from: http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/.

[2]Download from: http://pr.cs.cornell.edu/humanactivities/data.php. Copyright (c) Cornell University, 2009.

[3]Source code is available in: https://github.com/rasmusbergpalm/DeepLearnToolbox. Copyright (c) 2012, Rasmus Berg Palm (rasmusbergpalm@gmail.com)

TABLE III
LAMSTAR AND LAMSTAR II RESULTS AFTER TESTING 600 POSES FOR DIFFERENT THRESHOLDS FOR MSRDAILYACTIVITY3D DATASET

Network Type	Threshold	Training Time (sec)	Correct Detection (Out of 600 poses)	Wrong Detection (Out of 600 poses)	Recognition Rate (%)	Bit Error (%)
LAMSTAR	0.95	4.1835	0	600	0	83.33
LAMSTAR II		8.8706	100	500	16.67	27.7778
LAMSTAR	0.99	7.4821	141	459	23.50	61.00
LAMSTAR II		11.8511	100	500	16.67	27.7778
LAMSTAR	0.995	11.8540	159	441	26.50	49.75
LAMSTAR II		15.8227	174	426	29.00	23.6667
LAMSTAR	0.999	75.1410	407	193	67.83	10.9722
LAMSTAR II		77.8930	509	91	84.83	5.0556
LAMSTAR	0.9995	152.8203	490	110	81.67	6.1111
LAMSTAR II		157.2739	580	20	96.67	1.1111
LAMSTAR	0.9999	378.6317	574	26	95.33	1.4444
LAMSTAR II		429.4255	598	2	99.67	0.0111

the Hamming distance between the actual and recognized class label. And also it took 507.3073 seconds to complete the training process for 50 epochs. Table II shows the final confusion matrix for CNN.

2) LAMSTAR and LAMSTAR II results: In case of LAMSTAR and LAMSTAR II we tested our method for different thresholds on MSRDailyActivity3D dataset. The results are shown in Table III. It shows that for increasing threshold the recognition rate increases but the time taken by the network is also increased. Also it is evident that for same threshold LAMSTAR II works better than LAMSTAR. For 0.9999 threshold we got 95.33% and 99.67% recognition rate for LAMSTAR and LAMSTAR II respectively. Also it took 378 and 429 seconds in case of 0.9999 threshold for LAMSTAR and LAMSTAR II respectively. It shows that LAMSTAR networks perform far better than CNN model. Table IV and V show the final confusion matrices for LAMSTAR and LAMSTAR II. Finally the comparison results for best scenario between CNN, LAMSTAR and LAMSTAR II for MSRDailyActivity3D dataset are summarized in Table VII.
3) Comparison with State-of-the-art methods: Finally we compare our CNN and LAMSTAR networks with other state-of-the-art methods and previous approaches applied on MSRDailyActivity3D and CAD-60 datasets. Table VII and Table VIII depict the comparison result for MSRDailyActivity3D and CAD-60 respectively. It is clear from the tables LAMSTAR networks specially LAMSTAR II outperforms all the previous approaches.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we compare the performance of LAMSTAR and LAMSTAR II networks with respect to deep CNN classification model for action recognition using depth map sequences. The LAMSTAR is evaluated on extensive datasets and compared to a number of state-of-the-art approaches which shows it works far better than CNN or other state-of-the-art methods. LAMSTAR can itself achieve state-of-the-art results in individual datasets and it has the capability of maintain the accuracy in more complicated datasets. However the training time of LAMSTAR can be improved using direct storage method. Direct storage requires to define the entries as binary numbers. This can be a tremendous improvement in speed throughout. In our future work, we will use the direct storage method to improve the recognition accuracy and test our result with more complex and extensive datasets.

REFERENCES

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in ESANNCIML, 2013.

[2] M. Zhang, A. A. Sawchuk, "USC-HAD: a daily activity dataset for ubiquitous activity recognition using wearable sensors," in ACM Conference on Ubiquitous Computing, 2012.

[3] M. Shoaib, S. Bosch, O. D. Incel, H. Scholten, P. J. Havinga, "Fusion of smartphone motion sensors for physical activity recognition," Sensors

TABLE IV
CONFUSION MATRIX OF THE ACTIVITIES ON THE MSRDAILYACTIVITY3D DATASET USING LAMSTAR

	Predicted Class						
Actual Class	ET	CC	CU	SiS	StS	WK	Recall (%)
ET	100	0	0	0	0	0	100
CC	0	99	0	1	0	0	99
CU	0	3	96	1	0	0	96
SiS	0	0	0	100	0	0	100
StS	0	0	0	0	100	0	100
WK	0	0	0	3	18	79	79
Precision (%)	100	97	100	95	84	100	

ET = Eating, CC = Caling cellphone, CU = Cheering up, SiS = Sitting still, StS = Standing still, WK = Walking

TABLE V
CONFUSION MATRIX OF THE ACTIVITIES ON THE MSRDAILYACTIVITY3D DATASET USING LAMSTAR II

	Predicted Class						
Actual Class	ET	CC	CU	SiS	StS	WK	Recall (%)
ET	100	0	0	0	0	0	100
CC	0	100	0	0	0	0	100
CU	0	0	100	0	0	0	100
SiS	0	0	0	100	0	0	100
StS	0	0	0	0	100	0	100
WK	0	0	0	0	2	98	98
Precision (%)	100	100	100	100	98	100	

ET = Eating, CC = Caling cellphone, CU = Cheering up, SiS = Sitting still, StS = Standing still, WK = Walking

TABLE VI
SUMMARIZED COMPARISON BETWEEN CNN, LAMSTAR AND LAMSTAR II FOR MSRDAILYACTIVITY3D DATASET

Parameter	CNN	LAMSTAR	LAMSTAR II
Training time (sec)	507.307*	378.631†	429.425†
Training accuracy (%)	94..33‡	98.67‡	100‡
Testing time (sec)	172.365§	151.23§	153.365§
Bit error rate (%)	2.333◊	1.444Δ	0.011Δ
Recognition rate (%)	93◊	95.33Δ	99.67Δ

*Training time of 7590 training samples for 50 epochs
†Training time of 7590 training samples for threshold 0.9999
‡Testing with the same input used as training set
§Testing time of 600 test samples on trained network
◊For a trained CNN for 50 epochs
ΔFor a trained LAMSTAR/LAMSTAR II with threshold 0.9999

TABLE VII
RECOGNITION ACCURACY COMPARISON OF OUR WORK AND PREVIOUS APPROACHES ON MSRDAILYACTIVITY3D DATASET

Method	Accuracy (%)
LOP [28]	42.5
Depth Motion Maps [31]	43.13
Joint Position [28]	68
Moving Pose [32]	73.8
Local HON4D [33]	80
Actionlet Ensemble [28]	85.75
SNV [34]	86.25
HDMM + 3ConvNets [13]	81.88
Our method: CNN	93
Our method: LAMSTAR	95.33
Our method: LAMSTAR II	99.67

TABLE VIII
RECOGNITION PRECISION AND RECALL COMPARISON OF OUR WORK AND PREVIOUS APPROACHES ON CAD-60 DATASET

Method	Precision (%)	Recall (%)
MEMM [24, 25]	67.9	55.5
SSVM [26]	80.8	71.4
Structure-motion Features [35]	86	84
NBNN [36]	71.9	66.6
Image Fusion [37]	75.9	69.5
Spatial-based Clustering [38]	78.1	75.4
SI Point Feature [39]	93.2	84.6
Pose Kinetic Energy [40]	93.8	94.5
K-means Clustering + SVM + HMM [41]	77.3	76.7
S-ONI [42]	91.9	90.2
Our method: CNN	92.33	93
Our method: LAMSTAR	96.67	95.33
Our method: LAMSTAR II	100	100

(Basel), June 2014.
[4] P. Casale, O. Pujol, P. Radeva, "Human activity recognition from accelerometer data using a wearable device," in Pattern Recognition and Image Analysis, 2011.
[5] D. Graupe, "Large Scale Memory Storage and Retrieval (LAMSTAR) Network," in Principles of Artificial Neural Networks, 3rd ed. World Scientific, NJ, USA, 2013, pp. 203–274.
[6] W. Jiang, "Human Activity Recognition using Wearable Sensors by Deep Convolutional Neural Networks," in Proc. MM 2015, October 26–30, 2015, Brisbane, Australia.
[7] D. Graupe, H. Kordylewski, "A Large Scale Memory (LAMSTAR) Neural Network for Medical Diagnosis," in Proc. EMBS 1997, October 30-November 2, 1997, Chicago, IL, USA.
[8] D. Graupe, N. C. Schneider, "A Modified LAMSTAR Neural Network and Its Applications," International Journal of Neural Systems, Vol. 18, No. 4 (2008) pp. 331–337.
[9] A. Bayat, M. Pomplun, D. A. Tran, "A Study on Human Activity Recognition Using Accelerometer Data from Smartphones," Int. Conf. on Mobile Systems and Pervasive Computing, 2014.
[10] Ó. D. Lara, M. A. Labrador (2012, Nov.). A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Trans. Communications Surveys & Tutorials*, Vol 15, Issue 3, pp. 1192–1209.
[11] S. Ji, W. Xu, M. Yang, K. Yu, "3D Convolutional Neural Networks for Human Action Recognition," in Proc. ICML, 2010.
[12] K. Wang, X. Wang, L. Lin, M. Wang, W. Zuo, "3D Human Activity Recognition with Reconfigurable Convolutional Neural Networks," MM'14, November 3–7 2014, Orlando, FL, USA.
[13] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, P. Ogunbona, "Deep Convolutional Neural Networks for Action Recognition Using Depth Map Sequences," arXiv, preprint arXiv:1501.04686, 2015.
[14] J. B. Yang, M. N. Nguyen, P. P. San, X. L. Li, S. Krishnaswamy, "Deep Convolutional Neural Networks on Multichannel Time Series for Human Activity Recognition," in Proc. IJCAI 2015, pp. 3995-4001.
[15] A. Sharma, Y. D. Lee, W. Y. Chung, "High Accuracy Human Activity Monitoring using Neural network," in Proc. Int. conf. on Convergence and Hybrid Information Technology, 2008, pp. 430-435.
[16] J. B. Arie, Z. Wang, P. Pandit, S. Rajaram, "Human Activity Recognition Using Multidimensional Indexing," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol 24, no. 8, 2002, pp 1091–1104.
[17] A. Bearman, C. Dong, "Human Pose Estimation and Activity Classification Using Convolutional Neural Networks," Available in http://cs231n.stanford.edu/reports/cdong-paper.pdf
[18] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, J. Zhang, "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors," arXiv:1501.06262, 2015.
[19] S. S. D. Sashikumar, "3D Human Activity Recognition by Indexing and Sequencing (RISq)," MS Thesis, Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL, USA, 2015.
[20] R. Bordor, B. Jackson, N. Papanikolopoulos, "Vision-Based Human Tracking and Activity Recognition," Available in mha.cs.umn.edu/Papers/Vision_Tracking_Recognition.pdf
[21] M. K. Fiaz, B. Ijaz, "Vision based human activity tracking using artificial neural networks," in Proc. ICIAS 2010, Kuala Lumpur, Malaysia, June 2010, pp. 1–5.
[22] B. Delachaux, J. Rebetez, A. P. Uribe, H. F. S. Mejia, "Indoor Activity Recognition by Combining One-vs.-All Neural Network Classifiers Exploiting Wearable and Depth Sensors," in Proc. IWANN 2013, Part II, LNCS 7903, pp. 216–223, 2013.
[23] S. Homayon. (2015, Feb.). Iris Recognition for Personal Identification using LAMSTAR Neural Network. IJCSIT, vol 7, no 1.
[24] J. Sung, C. Ponce, B. Selman, A. Saxena, "Human Activity Detection from RGBD Images," in Proc. AAAI workshop on Pattern, Activity and Intent Recognition (PAIR) 2011.
[25] J. Sung, C. Ponce, B. Selman, A. Saxena, "Unstructured Human Activity Detection from RGBD Images," in Proc. ICRA 2012.
[26] H. S. Koppula, R. Gupta, A. Saxena, "Learning Human Activities and Object Affordances from RGB-D Videos," arXiv:1210.1207v2, May 2013.
[27] G. Yu, Z. Liu, J. Yuan, "Discriminative Orderlet Mining For Real-time Recognition of Human-Object Interaction," in Proc ACCV 2014.
[28] J. Wang, Z. Liu, Y. Wu, J. Yuan, "Mining Actionlet Ensemble for Action Recognition with Depth Cameras," in Proc. CVPR 2012, Providence, Rhode Island, June 16-21, 2012.
[29] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," Master's Thesis, Technical University of Denmark, Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark, 2012.
[30] C. Kapoor, "A comparison of Indexing and Sequencing with Dynamic Time Warping for Recognition of Human Activities," MS Thesis, Department of Electrical and Computer Engineering, University of Illinois, Chicago, IL, USA, 2015.
[31] X. Yang, C. Zhang, and Y. Tian, "Recognizing actions using depth motion maps-based histograms of oriented gradients," in ACMMM, 2012.
[32] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection," in ICCV, 2013.
[33] O. Oreifej and Z. Liu, "Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences," in CVPR, 2013.
[34] X. Yang and Y. Tian, "Super normal vector for activity recognition using depth sequences," in CVPR, 2014.

[35] C. Zhang, Y. Tian, "RGB-D Camera-based Daily Living Activity Recognition," Journal of Computer Vision and Image Processing, Vol. 2, No. 4, December 2012.
[36] X. Yang, Y. Tian, "Effective 3D Action Recognition Using Eigenjoints," Journal of Visual Communication and Image Representation (JVCIR), Special Issue on Visual Understanding and Applications with RGBD Cameras, 2013.
[37] B. Ni, Y. Pei, P. Moulin, S. Yan, Multilevel Depth and Image Fusion for Human Activity Detection, *IEEE Trans*. *Cybernetics*, 2013.
[38] R. Gupta, A. Y. S. Chia, D. Rajan, "Human Activities Recognition using Depth Images, " in Proc. of the 21st ACM international conference on Multimedia, 2013.
[39] Y. Zhu, W. Chen, G. Guo "Evaluating Spatiotemporal Interest Point Features for Depth-based Action Recognition," In Image and Vision Computing, 2014.
[40] J. Shan, S. Akella, "3D Human Action Segmentation and Recognition using Pose Kinetic Energy," In IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO), 2014.
[41] S. Gaglio, G. Lo Re, M. Morana, "Human Activity Recognition Process Using 3-D Posture Data," In IEEE Transactions on Human-Machine Systems, 2014.
[42] G. I. Parisi, C. Weber, S. Wermter, "Self-Organizing Neural Integration of Pose-Motion Features for Human Action Recognition," In Frontier in Neurobotics, 2015.
[43] T. Kohonen, "Self-Organization and Associative Memory," Springer Verlag, Berlin, 1984.
[44] D. Hebb, "The Organization of Behavior," John Wiley, New York, 1949.
[45] I. P. Pavlov, "Conditioned Reflexes," (in Russian), English translation by G. V. Anrep: Oxford University Press, 1927, Dover Press, 1960.