International Journal of Engineering & Technology, 7 (4.11) (2018) 202-205

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper



Convolutional Neural Network (CNN) based Gait Recognition System using Microsoft Kinect Skeleton Features

Mohd Shahrum Md Guntor[1], Rohilak Sahak[1], Azlee Zabidi[1], Nooritawati Md Tahir[1], Ihsan Mohd Yassin[1]*, Zairi Ismael Rizman[2], Rahimi Baharom[1], Noorfishah Abdul Wahab[1]

[1]Faculty of Electrical Engineering, Universiti Teknologi MARA (UiTM), Shah Alam, Selangor, Malaysia
2Faculty of Electrical Engineering, University of Technology MARA, 23000 Dungun, Terengganu, Malaysia
**Corresponding author E-mail: ihsan.yassin@gmail.com*

Abstract

Biometric identification systems have recently made exponential advancements in term of complexity and accuracy in recognition for security purposes and a variety of other application. In this paper, a Convolutional Neural Network (CNN) based gait recognition system using Microsoft Kinect skeletal joint data points is proposed for human identification. A total of 23 subjects were used for the experiments. The subjects were positioned 45 degrees (oblique view) from Kinect. A CNN based on the modified AlexNet structure was used to fit the different input data size. The results indicate that the training and testing accuracies were 100% and 69.6% respectively.

Keywords: *Convolution Neural Network, biometrics, human gait recognition, Kinect, skeletal joints.*

1. Introduction

The ability to determine the identity of a person with precise accuracy is an irreplaceable tool in many fields such as healthcare, security, social media, gaming and online marketing. Gait analysis is a method for human identification through biometric characteristics in human locomotion. This method has been around only recently and used mostly for clinical and laboratory identification on the variation with a normal gait. In a simple gait cycle, it comprised of a swing and stance phases. The stance phase is a part of gait cycle that resulted when the foot is in contact to the ground and accounts 60% of the gait cycle. Moreover, the stance phases are further divided into sections of movements namely initial contact, loading response, mid stance, terminal stance and toe off. In other hand, the swing phase consist of initial swing, mid swing and terminal swing that is another part of a gait cycle when the foot is swinging in the air and not is contacted to any ground. Furthermore, the swing phase accounts for 40% of the gait cycle [2].

Artificial Intelligence (AI) has been used extensively in the field of gait analysis due to the complexity of identification of this biometric feature. Using traditional neural network approach in gait recognition tend to have its own disadvantage such as requiring multiple parameter tuning and difficulty in data training. There are multiple researches on human gait recognition based skeletal data that improve the computational recognition speed and accuracy human identity verification. A Convolutional Neural Network (CNN) approach are proven to have more accurate results compared to the traditional ones [4].

The CNN method is a part of a new paradigm in AI called Deep Learning (DL). DL networks have complex structures previously unachievable by fourth generation neural networks. For example, CNN contains a cascade of convolution layers as feature extractors (together with feature reduction layers), followed by multilayer perceptron neural network at the final layers to perform classification. This structure allows the CNN to learn and classify very complex patterns, such as in images.

The Microsoft Kinect hardware is a technology developed by Microsoft to control its line of Xbox video game consoles without using a remote. The hardware consists of three sensors, which are the color video camera, depth sensor and multi-array microphone. Of interest is the depth sensor which can detect and track multiple points in the human body and maps them into a digital representation that corresponds to the body shape and the skeletal structure [3].

This paper applies a CNN-based architecture for human recognition using skeletal data captured by a Kinect sensor. The CNN was evaluated in terms of accuracy in feature classification of human recognition using the collected skeletal data. The CNN based architecture was chosen because the pattern recognition for traditional neural network algorithm are manually designed. The CNN uses weights of the convolutional layer for feature extraction and a fully connected layer for classification during the training process. The architecture of the CNN was implemented in the MATLAB software which is a superior language for technical computing. The methodology is described in Section 2, while results and discussions are presented in Section 3. Finally concluding remarks are presented in Section 4.

A real-time gait recognition system was proposed in [5] based on dynamic (angles of joints) and static (bone length) features based on Kinect skeleton information. The static and dynamic feature templates were stored into a database after preprocessing. Feature extraction was performed by calculating the distance and coordinates of joints using Euclidean distance for static features and Dynamic Time Warping (DTW) for dynamic features. Both distance of the features was then fused into a fusion feature. Finally, classification was performed using a Near Neighbor (NN) classifier. The result indicates that the Correct Classification Rate (CCR) is higher after the fusion feature, which reached 82% accuracy and showed the advantage of using this featured method compared to other two methods.

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In [6], a person identification system using skeleton information from Kinect was presented. The proposed method uses Microsoft Kinect Software Development Kit (SDK) to record the skeleton data from the depth image. The half-gait cycle gait features were then extracted by computing its feature vectors using Euclidean distance. An Artificial Neural Network (ANN)-based connectionist framework was used for feature selection and a Naive Bayes (NB) classifier was used for classification. Both ANN and NB are supervised learning algorithm implemented to determine features for identification. The results indicate that the static features get higher importance compared to the dynamic features and hybrid feature selection and classification have better recognition accuracy compared to other methods.

2. Methodology

The experiment overview is shown in Figure 1. The major steps involved consisted of data collection, data acquisition, classifier design, training and testing. The CNN was implemented using MATLAB R2017b. Training was performed on a laptop with a NVIDIA GTX 960M Graphics Processing Unit (GPU) with 2 GB of memory.

2.1. Data Collection

The data were obtained from [7]. It contains three-dimensional skeletal joint points of 23 subjects recorded by a Microsoft Kinect depth camera. Samples were collected under indoor environment where each subject was required to wear standardized outfits. Ten walking sequences per subject was captured in oblique view, where the subjects were required to walk in a 45˚ angle from the upper side of the Kinect camera. There were ten walking sequence that consisted of 15,600 recorded joint points for each sample.

2.2. Data Acquisition

The gait data of each walking sequence was set to the corresponding subjects and transposed to 60 by 26 three-dimensional array to fit the input requirement of the CNN. The column represents all the joint points at a time frame, while the row represents the joint data across time. Once the input was reshaped, the gait data was then split and distributed into training and testing data sets in 50:50 ratio. The input was distributed randomly to improve generalization.

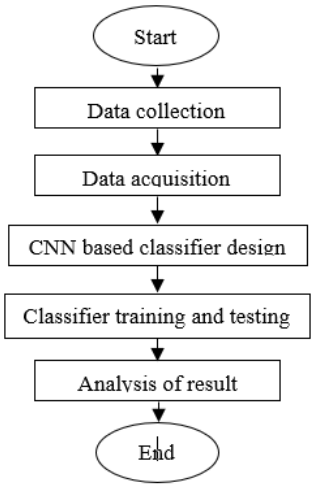

Fig. 1: Methodology flowchart

2.3. CNN Classifier Design

The CNN network consisted of several convolutional layers placed alternately between pooling and normalization layers. Each of the convolutional layer calculates the convolution between the inputs and sets of filters. Training was performed in a supervised manner. The structure of CNN consists of a Convolutional Layer, Batch Normalization Layer, Rectifier Linear Units (ReLU), Pooling Layer and Fully Connected Layer. The CNN architecture model is shown in Figure 2.

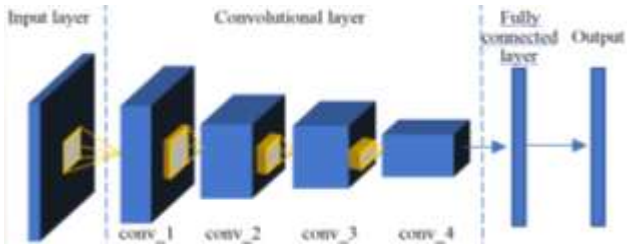

Fig. 1: CNN architectural model

This CNN structure is like AlexNet, a pretrained CNN that consists of five convolutional layers and three fully connected layers. However, this network implements only four convolutional layers and a single fully connected layer. Moreover, the batch normalization, ReLU and pooling layers were placed between the convolutional layers. The hyperparameters and output size of the layers are shown in Table 1-2 respectively.

Table 1: CNN layer parameters

Layer	Filter Size D × H × W	Other Parameters
Convolution 1	16 × 7 × 7	stride = 1, padding = 0
Batch normalization	-	-
ReLU	-	-
Max pooling	16 × 3 × 3	stride = 1, padding = 2
Convolution 2	32 × 5 × 5	stride = 1, padding = 0
Batch normalization	-	-
ReLU	-	-
Max pooling	32 × 3 × 3	stride = 1, padding = 1
Convolution 3	64 × 3 × 3	stride = 1, padding = 0
Batch normalization	-	-
ReLU	-	-
Max pooling	64 × 3 × 3	stride = 1, padding = 1
Convolution 4	128 × 3 × 3	stride = 1, padding = 0
Batch normalization	-	-
ReLU	-	-
Fully connected + SoftMax	23	-

Table 2: Output size of CNN layers.

Layer	Output D × H × W
Convolution 1	16 × 54 × 20
Max pooling	16 × 48 × 22
Convolution 2	32 × 44 × 18
Max pooling	32 × 44 × 18
Convolution 3	64 × 42 × 16
Max pooling	64 × 42 × 16
Convolution 4	128 × 40 × 14

2.3.1. Convolutional Layer

The convolutional layer was used to convolve the input by sliding the filters horizontally and vertically along the input. The dot product of the weights and the input are computed for each filter and then a bias term was added [8]. In this paper, there were four convolutional layers that were implemented with specific hyperparameters shown in Table 1. The formula to calculate the output size of the convolutional layer and the number of weights per filter are shown in (1) and (2) respectively.

$$\text{Output size} = \frac{W_I - F + 2P}{S} + 1 \quad (1)$$

$$\text{Weight} = F \times F \times 3 \quad (2)$$

where W_I is the input size, F is the filter size, P is the padding size and S is the number of strides.

2.3.2. Batch Normalization Layer

The batch normalization layer takes each input channel x_i across a mini-batch and normalizes it by subtracting the mini-batch mean μ_B and dividing by the mini-batch standard deviation σ_B^2. Then, the layer moves the input by a learnable offset β and scales it by a learnable scaling factor γ. This speed up training of CNN and reduces sensitivity to network initialization [9].

$$\hat{x} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \qquad y_i = \gamma\hat{x} + \beta \tag{3}$$

2.3.3. Rectifier Linear Units (ReLU)

The ReLU performs a threshold operation to the elements of the input that transforms negative values to zero and maintaining the positive values. It is also referred as an activation function that determine the condition of a neuron whether it fires or stay dormant [10].

$$f(x) = \max(0, x) \tag{4}$$

2.3.4. Pooling layer

The pooling layer functions as a down-sampling that divides the input into rectangular pooling regions and computed the values of each region. It also smoothens the output and prevent local variances [11]. There are three types of pooling methods which are maximum, minimum and average pooling. In this paper, a maximum pooling layer had been applied to sub-samples the output of the convolutional layer. The maximum pooling layer computes the maximum values of each region. The formula to calculate the output size of a pooling layer are shown in (5).

$$\text{Output size} = \frac{W_I - P_l + 2P}{S} + 1 \tag{5}$$

where W_I is the input size, P_l is the pool size, P is the padding size and S is the number of strides.

2.3.5. Fully Connected Layer

The fully connected layer acts as a classifier layer in the end process of the CNN as it performs classification of previous extracted feature by operating trained weighted connections. It multiplies the input by a weight matrix and adds a bias vector [12]. This layer takes the output of any previous layer and output an N dimensional vector that that determines the number of selectable classes for classification. The output size and the bias of the layer was set to 23 as there were 23 possible subjects to be classified.

2.3.6. Training Options

The Stochastic Gradient Descent with Momentum (SGDM) was used as an optimizer with an initial learning rate of 0.00001. In the batch processing option, the maximum number of epochs and size of mini-batch was set to 150 and 256 respectively for each training iteration.

3. Results and Discussion

3.1. Trained Convolutional Layer Weights

The learned features or output activations of each convolutional layers are visualized using the "deep dream" function in MATLAB. Figure 3 shows the feature maps of the all the convolutional layers with respect to the number of features. In the first layer, the 16 features are displayed together by using the montage option and the number of features is based on the layer parameters. Adding layers such as batch normalization, ReLU and max pooling in the network increases the complexity of the features as shown in the second, third and fourth convolutional layers. These features are then used for classification in the fully connected layer.

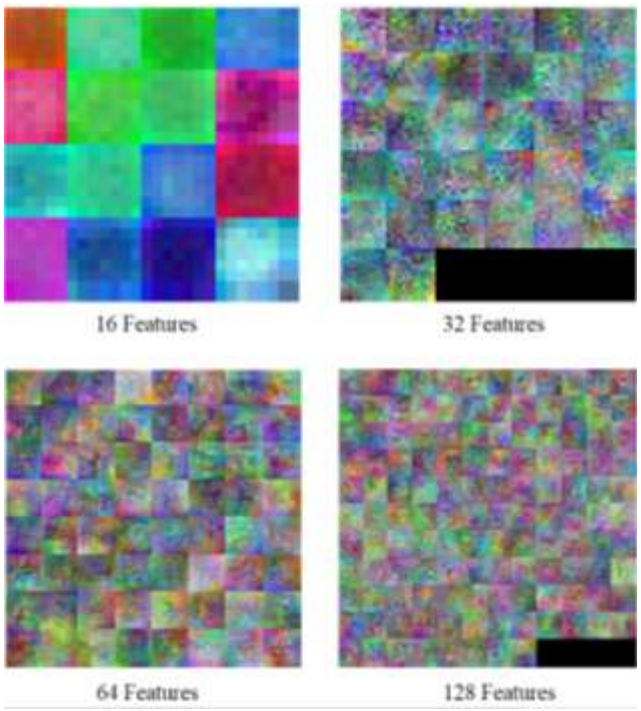

Fig. 2: The visualized feature maps of all the convolutional layers.

3.2. Classification Results

The overall accuracy of this network on training and testing the data sets are 100% and 69.6% respectively. The results are shown from a confusion matrix in Figure 4-5.

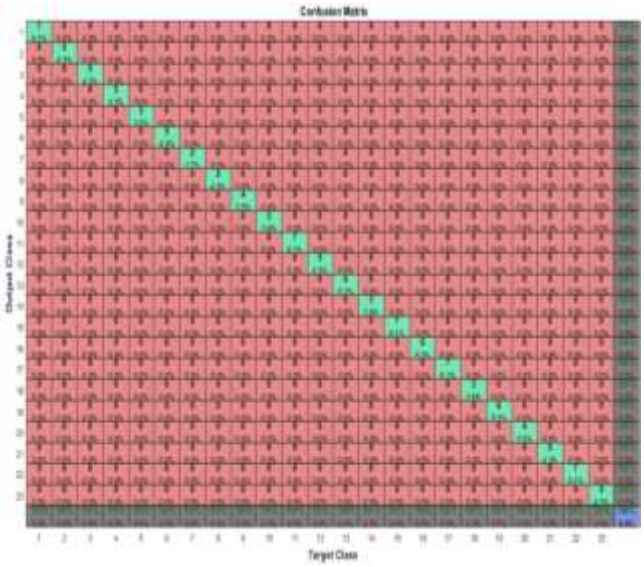

Fig. 3: Confusion matrix comparison on the trained data. The overall accuracy is 100%.

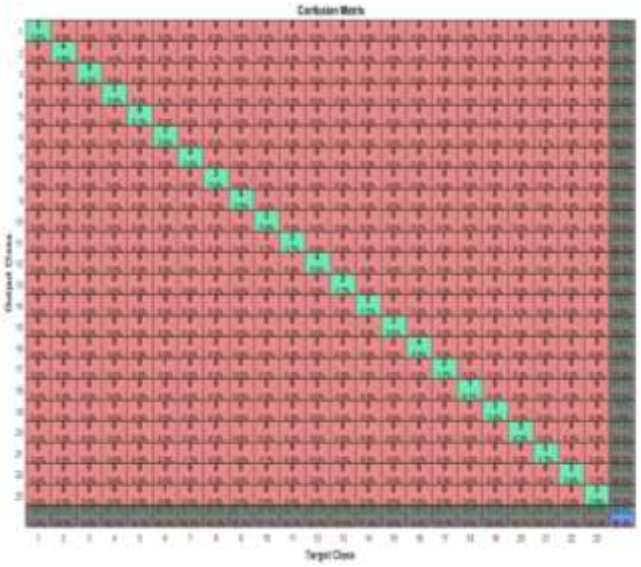

Fig. 4: Confusion matrix comparison on the trained data. The overall accuracy is 69.6%.

The confusion matrix is used to describe the performance of the CNN classification model. The formula to calculate the overall accuracy, recall and specificity are shown in (6)-(8) respectively.

$$\text{Accuracy } (\%) = \frac{TP+TN}{TP+TN+FP+FN} \quad (6)$$

$$\text{Recall } (\%) = \frac{TP}{TP+FN} \quad (7)$$

$$\text{Specificity } (\%) = \frac{TP}{TN+FP} \quad (8)$$

where TP is the true positives, TN is the true negatives, FP is the false positives and FN is the false negatives.

Although the training set accuracy was 100%, there was a notable performance decline of the CNN for the testing set. Overfitting of data in the training process was probably the main reason to cause the poor accuracy in testing data. Overfitting is a common problem in neural network that occurs when the network effectively memorized a data rather than trying to predict new unseen data based on their features. Moreover, this problem becomes more significant in deep learning because of the larger number of layers that consists of many neurons. Therefore, network assessment techniques such as k-fold or holdout are recommended to minimize overfitting issues. Other than that, the accuracy of the CNN to perform complex classification is generally constrained by the quality of the training data. The performance of a CNN and other modern nonlinear machine learning techniques increases with the available of large quantity and quality data. This is also another area which can be improved.

In terms of network structure, the accuracy of the network increases as more layers that are fine-tuned because this creates higher detailed of features for classification. However, training and fine-tuning the algorithm is time consuming and requires a Graphics Processing Unit (GPU) with high computational power. Moreover, the accuracy can be further increased by adjusting the training parameters such as the maximum number of epochs and initial learning rate.

4. Conclusion

In this paper, recognition of subjects based on their Kinect gait patterns using CNN architecture was presented. A modified AlexNet structure was used to construct the CNN. From the results, the CNN appears to perform well on the training data. However, there was an approximately 30% decrease in accuracy for the testing set. This could be attributed to overfitting, which would suggest that future works in improving network structure, training parameters and additional data sets is necessary.

Acknowledgement

Authors gratefully acknowledge the financial support from Ministries of Higher Education Malaysia and Institute of Research Management and Innovation (IRMI) Universiti Teknologi MARA Grant No: 600-RMI/NRGS 5/3 (3/2013).

References

[1] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527-1554.

[2] Arun, P. S. Normal gait cycle. http://boneandspine.com/normal-gait-cycle/.

[3] Stephanie, C. The Kinect sensor. https://electronics.howstuffworks.com/microsoft-kinect2.htm.

[4] Lawrence, S., Giles, C. L., Tsoi, A. C., & Back, A. D. (1997). Face recognition: A convolutional neural-network approach. IEEE Transactions on Neural Networks, 8(1), 98-113.

[5] Wang, Y., See, J., Phan, R. C. W., & Oh, Y. H. (2014). Lbp with six intersection points: Reducing redundant information in lbp-top for micro-expression recognition. Proceedings of the Asian Conference on Computer Vision, pp. 525-537.

[6] Sinha, A., Chakravarty, K., & Bhowmick, B. (2013). Person identification using skeleton information from kinect. Proceedings of the International Conference on Advances in Computer-Human Interactions, pp. 101-108.

[7] Sahak, R., Yassin, I., Tahir, N. M., Zaman, F. H. K., & Zabidi, A. (2017). Recognition of human gait in oblique and frontal views using Kinect. Journal of Fundamental and Applied Sciences, 9(4S), 1-18.

[8] MathWorks Inc. 2-D convolutional layer - MATLAB. https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.convolution2dlayer.html.

[9] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. https://arxiv.org/pdf/1502.03167.pdf%E7%9A%84paper%E9%80%82%E5%90%88%E6%83%B3%E6%B7%B1%E5%85%A5%E4%BA%86%E8%A7%A3%E5%8E%9F%E7%90%86%EF%BC%8C%E8%BF%99%E4%B8%AA%E8%A7%86%E9%A2%91%E5%BE%88%E6%B8%85%E6%A5%9A%E7%9A%84%E8%AE%B2%E4%BA%86bn%E8%B5%B7%E5%88%B0%E7%9A%84%E4%BD%9C%E7%94%A8%E3%80%82.

[10] MathWorks Inc. Rectified Linear Unit (ReLU) layer - MATLAB. https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.relulayer.html.

[11] MathWorks Inc. Max pooling layer - MATLAB. https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.maxpooling2dlayer.html?searchHighlight=max pooling&s_tid=doc_srchtitle.

[12] MathWorks Inc. Fully connected layer - MATLAB. https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.fullyconnectedlayer.html.