Indian Journal of Science and Technology, *Vol 11(4), DOI: 10.17485/ijst/2018/v11i4/121090,* January *2018*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Real-Time Human Action Recognition using Stacked Sparse Autoencoders

Adnan Farooq, Emad U Din Mohammad, Abdullah Ahmad Zarir, Amelia Ritahani Ismail* and Suriani Sulaiman

Department of Computer Science, Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, P.O. Box 10, 50728 Kuala Lumpur, Malaysia; amelia@iium.edu.my

Abstract

Objectives: In this paper, an automated real-time human and human-action detection system is developed using Histogram of Oriented Gradients (HOG) and Stacked Sparse Auto-encoders respectively. **Methods:** For human detection, a feature descriptor is trained using SVM classifier and then is used for identification of humans in the frames. Stacked Sparse autoencoders are a category of deep neural networks, and in the proposed work is used for the feature extraction of human actions from the human action video dataset. The extracted features represent a dictionary which is used to map the input and produce a linear combination, following that soft-max classification is applied to train the model. To reduce the computational complexity, input frames has been changed into binary temporal difference images and fed to the neural network. **Analysis:** The proposed model matched the other state of the art models applied for human-action recognition classification problems. **Applications:** The study reveals that using multiple layers can improve the classification performance: 75% with two-layers and 83% with three-layers model.

Keywords: Auto-Encoder, HOG, Soft-Max, SVM

1. Introduction

For the past few years, recognition tasks in computer vision have become as an emerging research interest because of their growing applications in the intelligent technologies. These include the work on image, face, fingerprint or real-time objects detections. Human detection is quite a challenging task due to various feature variables and different pose angles and it has gained much attention as it can be applied in various fields such as surveillance, character animation for games and movies, biomechanical analysis of actions for sports and medicine, advanced intelligent user interfaces and avatars for teleconferencing. Supervised algorithms have been one of the important algorithms in artificial intelligence. In those algorithms, back-propagation is directly performed on initialized weights, which tend to make it slow and get stuck in local minima, especially when implementing on video frames. This results in deep neural networks that take a long time to finish training and ultimately yield bad accuracy. This is further explained by Hilton that if you pre-train each layer of the network in an unsupervised manner to learn a sparsified representation before the classification task begins the learning problem will greatly be reduced.[1] An autoencoder neural network is an unsupervised learning algorithm that applies back-propagation, by setting the target values to be equal to the input. Sparse autoencoders are a type of autoencoder that implements sparsity and generally consists of two steps: a learning algorithm that produces a dictionary that sparsely represents the data, and an encoding algorithm that, given the dictionary, that sparsely represents the data, and an encoding algorithm that, given the dictionary, defines a mapping from a new input vector to a feature vector. The produced diction-

*Author for correspondence

ary also minimizes reconstruction error while restricting the number of code-words required for reconstruction. A simple sparse encoder contains a single hidden layer connected to the input vector, by a weight of matrix that forms the encoding step. Then the output of the hidden layer is held by the reconstruction vector, to form the decoding step.

2. Literature Review

In the current times, there is a growing interest in the research of recognition task with computer vision. Lots of research has been conducted and in this paper, we only highlight some of the work on human and human action detection. In many studies the extraction of crowd features such as velocity and direction is performed by optical flow. In order to find the directions of the crowd flow, Wu used optical flow to detect particle advection trajectories, which was then clustered to find the direction.[2] They model the scene through chaotic dynamics using clustered trajectories. Chen and Huang used similar method to generate different action patterns of the crowd flow.[3] Orientation, position and crowd size were the motions features. Efros correlated optical flow measurements from low-resolution videos to perform action recognition.[4] Furthermore, Zhou and Hoang tracked a human body in a video by subtracting the background and detecting the foreground object.[5] Then used classification to track the object, it was considered human only if it was trackable. Other work has been demonstrated by Wang and Miao, where they modelled the behavior of the scene using the historical information of the motion.[6] There are also work on action recognition that is done using the Convolutional Neural Network (CNN). Cheng-Bin predicted human actions through temporal images and CNN.[7] A CNN model based on temporal images and a hierarchical action structure was developed for real-time human action recognition. How to evaluate any Human Action recognition system is also an important factor to maintain the standard of research and track the progress forward. Tal Hassner provided key critical insight in his paper about action recognition benchmarks.[9] This dissection of available benchmarks for evaluating action recognition makes it easier to pick up appropriate benchmark for particular research in this topic.

In the study by Neibles and Fei-Fei, they used spatial-temporal words to categorize human actions, and achieved a performance of around 70% in classification.[12] In other study, done by Jhuang, they classified human actions using RBF (Radial Basis Functions) kernel with SVM (Support vector machines) algorithm for classification.[13] Their proposed model achieved a classification accuracy of 92%. Similarly, Efros using single clip per action approach evaluated their proposed model on the same dataset, achieving an accuracy of 87%.[4]

3. Methodology

3.1 Dataset

A dataset of 56 videos from the WEIZMANN Action dataset was used for training and testing of the classifier evaluated by two-fold cross-validation.[11] The dataset consisted of 6 different classes: Bend, Run, Walk, Jump, One-hand Wave, Two-hand Wave. All sequences were taken over homogeneous background with static camera.

3.2 Pre-Processing

The goal of the preprocessing was to remove the background noise and normalize the data, to create a compact representation of the human action. As a preprocessing task, we scale down the image dimensions from (180 x 144) to (96 x 71), and convert each frame to grayscale, and then to its binary image, keeping track of the temporal differences of the human motion as shown in Figure 1.

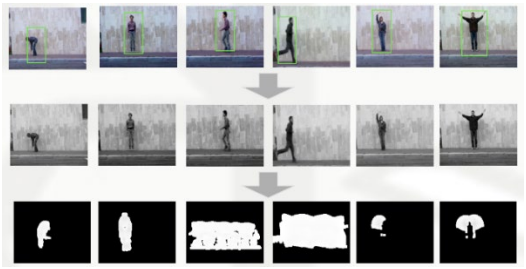

Figure 1. Sample frames from the WEIZMANN Dataset and the preprocessing.

3.3 Feature Extraction

After the preprocessing task, the temporal differences of the images are fed to the auto-encoder.

3.3.1 Sparse Auto-Encoder

An Auto-encoder neural network is an unsupervised learning algorithm, that implements back-propagation to produce the output that is identical to the input, i.e. . $y^{(i)} = x^{(i)}$

The network has less units in the hidden layer compared to the input layer. Having comparatively less units in the hidden layer is a way of constructing a compressed representation of the input. By this compression, an interesting structure of the features of the input can be discovered at each hidden layer. The vectorized features are then passed on to other layers to learn higher-order features, constructing a deep network. This way the stacked auto-encoder tends to learn a good representation of its input.

3.3.2 Feed-Forward Algorithm

The auto-encoder is trained with the input, $x^{(i)}$ to get the activation for each neuron units in the hidden layer. The "neuron" is a computational unit that takes the input and produces the output $f(W^Tx)$, which is called the activation function, in which "W" is the weight over the edges and "x" is the input. The activation for each node in each layer is depicted by α_j^l, where "j" is the neuron number and "l" is the layer number. For our experiment we used the sigmoid activation function (ranges between 0 and 1), which is assumed to be active if its value is close to 1, otherwise inactive if its value is close to 0, and the weights for our network are initialized randomly, the function is described as,

$$f(x) = \frac{1}{(1+e^{(-z)})} \tag{1}$$

where. $z=W^Tx = W_ix_i + b$. This function will return the element wise sigmoid output of input vector. For each node in each layer the values of the function are fed to the succeeding layers, to get the output, $h_{W,b}(x)$ in the last layer. The output is then compared to the input x. The algorithm can be summarized as follows:

Algorithm 1

Require: bxias, hidden size, visible size, input

Ensure: hidden layer activation

1. Initializethe limits
2. Initialize the weights and biases from the limits
3. Compute hidden layer activations

3.3.3 Cost Function Calculation

In this method, the output from the network is compared with the input and squared-error difference is calculated, which forms our cost function and is shown below,

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{(W,b)(x)} - y\right\|^2 \tag{2}$$

For m training set examples, the cost function becomes,

$$J(W,b) = \frac{1}{m}\sum_{(i=1)}^{m}\left\|h_{(W,b)}(x) - y\right\|^2 \tag{3}$$

Following this, we add the regularization term (also called a weight decay term) to the cost function that will decrease the magnitude of weights and will help to prevent over-fitting in case it occurs. The weight decay is a form of penalty for complexity. It penalizes models with extreme parameter values and have an increasing variance in the data error. It often produces sparse models. Therefore, after adding the term, our overall cost function is depicted below,

$$J(W,b) = \frac{1}{m}\sum_{(i=1)}^{m}\left\|h_{(W,b)}(x) - y\right\|^2 + \frac{\lambda}{2}\sum_{(l=1)}^{(n_l-1)}\sum_{(i=1)}^{(s_l)}\sum_{(j=1)}^{(s_l+1)}(W_{ji}^l)^2 \tag{4}$$

Where, = Total number of layers and = number of neurons in each layer. The weight decay parameter λ controls the relative importance of the two terms. The weight decay is not applied to the bias terms , because it does not produce much effect on the output. In auto-encoders, we add an extra term to the cost function which is KL (Kullback-Leibler) divergence term. Let $a_j^{(l)}(x)$ denote the activation of the hidden unit for a specific input x, which forms the average activation of hidden unit.

$$\hat{\rho}_j = \frac{1}{m}\sum_{(i=1)}^{m}[a_j^l(x^i)] \tag{5}$$

To enforce the constraint, we let $\hat{\rho}_j = \rho$ where ρ is a sparsity parameter, typically a value close to zero. Therefore, we aspire that the average activation of hidden unit *j* nears zero. To achieve this, we choose the following penalty term that gives reasonable results.

$$\sum_{(j=1)}^{(s_l)} KL(\rho \,||\, \hat{\rho}_j) = \sum_{(j=1)}^{(s_l)} \rho \log\frac{\rho}{\hat{\rho}} + (1-\rho)\log\frac{1-\rho}{1-\hat{\rho}_j} \tag{6}$$

Hence, our overall cost function becomes,

$$J_{sparse}(W,b) = J(W,b) + \beta\sum_{(j=1)}^{(s_l)} KL(\rho \,||\, \hat{\rho}_j) \tag{7}$$

where β controls the weight of the sparsity penalty term. Therefore, it is the use of regularization and KL divergence that makes the auto-encoders sparse and hence are known as sparse auto-encoders. The pseudo-code for the cost-function calculation is as follows:

Algorithm 2 Sparse Autoencoder Cost

Require: theta, input

Ensure: gradient of theta using back-propagation algorithm

1. Extract weights and biases from the theta
2. Compute output layer by performing a feed-forward pass
3. Estimate the average activation values of hidden layers
4. Compute difference using back-propagation algorithm
5. Compute the gradient values by averaging partial derivatives

3.3.4 Back Propagation Algorithm

In this algorithm, we implement the gradient descent that minimizes the cost function, which is used to update the weights and biases over the defined iterations of training data. One iteration of gradient descent updates the parameters W,b as follows:

$$W_{ij}^{(j)} = W_{ij}^{(j)} - \alpha \frac{\alpha}{\alpha W_{ij}^{(j)}} J_{sparse}(W,b) \tag{8}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\alpha}{\alpha b_i^{(l)}} J_{sparse}(W,b) \tag{9}$$

The KL-divergence term is incorporated into our derivative calculation and the error is calculated as follows,

$$\delta_i^l = \left(\left(\sum_{(j=1)}^{(s_l)} W_{ji}^l \delta_j^{(l+1)} \right) + \beta \sum_{(j=1)}^{(s_l)} KL(\rho \,||\, \hat{\rho}_j f'(z_i^{(l)}) \right) \tag{10}$$

After the error calculation, the partial derivatives can be calculated as follows,

$$\frac{\partial}{\partial W_{ij}^{(i)}} J_{sparse}(W,b;x,y) = a_j^{(l)} \delta_i^{(l+1)} \tag{11}$$

$$\frac{\partial}{\partial b_i^{(l)}} J_{sparse}(W,b;x,y) = \delta_i^{(l+1)} \tag{12}$$

And thus final updates are done. After finishing the training, at each hidden layer, a dictionary of features of various orders is stored, that is referenced while testing especially the dictionary acquired from the last hidden layer. Hence, the feature extraction is done using the unsupervised learning algorithm.

3.3.5 Classification

The output of features acquired from the sparse auto-encoders is fed to the softmax classifier layer, which is trained with the same dataset along with the specified labels of classes for identification. The classification task is a supervised learning. Hence, the output result is provided by the classification layer.

3.3.6 Histogram of Oriented Gradients (HOG)

We have used this HOG for real-time human detection in video frames. It has evolved as the strong tool in image processing for computer vision. It is a kind of feature descriptor that generalizes the object by dividing the object into connected regions in such a way that the same object (person) produces as close as possible to the same feature descriptor when viewed under different conditions. This makes the classification task easier. The histogram of gradient directions for pixel intensities in the specified region is calculated. The descriptor forms the aggregation of those histograms. It uses a "global" feature to describe a person rather than a collection of "local" features. The entire person is represented by a single feature vector. The HOG detector uses a sliding detection window which is moved around the frames. In every part of the detector window, a HOG descriptor is computed for the detection window. The acquired descriptor is then fed to the SVM classifier.

4. Results

We trained and tested our model on the data samples selected from the WEIZMANN Dataset of human action recognition and also applied real-time classification on it. The architecture that we used to train and test the model was a 2.6 GHz Intel Core i5 with 8 GB RAM. Fig. 2 illustrates the comparison of different models on the same Dataset with the presented articles.[12-14] Our model is capable of matching these state of the art methods with 83%, even though trained on a limited amount of dataset.

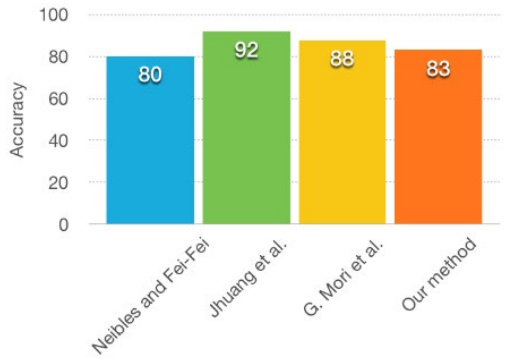

Figure 2. Accuracy comparison with other models.

Our results also showed how the accuracy of the two-layer and three-layer auto-encoders changes with the number of iterations. Fig. 3 illustrates how on increasing iterations; a three-layer auto-encoder gave better classification accuracy. Though the results seem to be comparable, the difference isn't for large number of iterations. Both models achieve same accuracy.

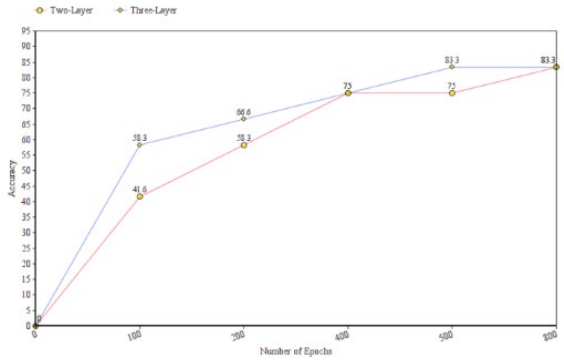

Figure 3. Three-layer model outperforms two-layer model.

Using comparatively smaller training dataset, power limitations of the architecture that is used for the training purposes have resulted in lesser accuracy in comparison to the other models it is being compared to, which on the other hand are using much more data and better powerful architectures. However, it is also not necessary that increasing the hidden layers may increase the accuracy. In the Microsoft research conducted by the presented article, they have described that as they went on increasing the hidden layers, after a certain point upon increasing the layers, the classification error increased.[15] Having the limited training dataset, adding layers could increase the complexity of the model and various problems may arise, like, the impact of back-propagation reduces, the overfitting problems may increase and the optimization of weights could be troublesome. That's why we settled down for two and three hidden layers for this model, keeping in view the dataset that is being used and the other limitations.

5. Conclusion and Future Work

The proposed work, aimed at using stacked auto-encoders for human action classification. The pre-training of the auto-encoders in unsupervised manner help reduce the training time. The results were satisfying and matched the state of the art models applied to the same classification problem. The classification strategy also works well in the real-time, and still can be improved. We found that stacked auto-encoders are capable of classifying human actions in a video to a great expectation. The results in the confusion matrices reveal interpretable data.

6. Acknowledgements

This research is supported by the International Islamic University Malaysia under the Research Initiative Grants Scheme (RIGS): RIGS16-346-0510

7. References

1. Geoffrey EH, Maxwell S, Yee WT. A fast learning algorithm for deep belief nets. Neural Comput. 2006 Jul; 18(7):1527–54. crossref PMid:16764513
2. Shandong W, Moore BE, Shah M. Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. Computer Vision and Pattern Recognition (CVPR). 2010 IEEE Conference on IEEE; 2010. crossref
3. Duan-Yu C, Po-Chung H. Motion-based unusual event detection in human crowds. Journal of Visual Communication and Image Representation 22.2; 2011. p. 178–86. crossref
4. Efros AA, Berg AC, Mori G, Malik J. Recognizing action at a distance. In Proceedings of the 9th IEEE International Conference on Computer Vision; 2003 Oct. p. 726–33. crossref
5. Jianpeng Z, Hoang J. Real time robust human detection and tracking system. Computer Vision and Pattern Recognition-Workshops. CVPR Workshops. IEEE Computer Society Conference on IEEE; 2005.
6. Shu W, Miao Z. Anomaly detection in crowd scene using historical information. Intelligent Signal Processing and Communication Systems (ISPACS). 2010 International Symposium on IEEE; 2010.

7. Cheng-Bin J et al. Real-Time Human Action Recognition Using CNN Over Temporal Images for Static Video Surveillance Cameras. Advances in Multimedia Information Processing–PCM 2015. Springer International Publishing; 2015. p. 330–9. PMCid: PMC4509706
8. Shuiwang J et al. 3D convolutional neural networks for human action recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 35.1; 2013. p. 221–31. crossref PMid:22392705
9. Tal H. A critical review of action recognition benchmarks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2013. crossref
10. Navneet D, Triggs B. Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. 2005;1. crossref
11. Ronald P. A survey on vision-based human action recognition. Image and vision computing 28.6; 2010. p. 976–90. crossref
12. Carlos NJ, Wang H, Fei-Fei L. Unsupervised learning of human action categories using spatial-temporal words. International journal of computer vision 79.3; 2008. p. 299–318. crossref
13. Hueihan J et al. A biologically inspired system for action recognition. Computer Vision. ICCV 2007. IEEE 11th International Conference on IEEE; 2007. crossref
14. UFLDLTutorial by Ng, A Ngiam J, Foo CY, Mai Y, Suen C; 2010. http://ufld l.stanford.edu/wiki/index.php/UFLDL_Tutorial (2010)
15. Kaiming H et al. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. crossref
16. Lu X, Chen CC, Aggarwal JK. View invariant human action recognition using histograms of 3d joints. Computer Vision and Pattern Recognition Workshops (CVPRW). 2012 IEEE Computer Society Conferenc e on IEEE; 2012. crossref
17. Weilong Y, Wang Y, Mori G. Human action recognition from a single clip per action.˝ Computer Vision Workshops (ICCV Workshops). 2009 IEEE 12th International Conference on IEEE; 2009.