RESEARCH REVIEW JOURNALS
Volume-04
Issue-06
June-2019

ISSN: 2455-3085 (Online)
RESEARCH REVIEW International Journal of Multidisciplinary
www.rrjournals.com[UGC Listed Journal]

Convolution and Recurrent Neural Network Fusion for Human Action Recognition in Controlled Environment

[1]Pati Chinmayee, [*2]Tripathy Ajaya Kumar, [3]Panda Ashok Kumar & [4]Pradhan Sateesh Kumar

[1,3,4]*Dept. of MCA, Utkal University, Bhubaneswar (India)*
[2]*Dept. of CSE, Silicon Institute of Technology, Bhubaneswar (India)*

ARTICLE DETAILS

Article History
Published Online: 10 June 2019

Keywords
Machine Learning, Neural Network, Gait Recognition, CNN, and RNN.

[*]**Corresponding Author**
Email: ashopanda[at]gmail.com

ABSTRACT

Human Action Recognition from video can play a big role in intelligent video surveillance. Majority of the approaches in the literature depends on pre environmental knowledge to extract complex handcrafted features from inputs. Convolutional neural networks (CNN) are one of the promising approaches which automate the process of feature extraction. Whereas, Recurrent Neural Network (RNN) is one of the promising technique in sequence modeling. This paper proposes a novel approach for Human action recognition by fusing the CNN and RNN architecture. The input video is converted to a sequence of images and the conjugative image difference sequence is passed to a sequence CNN and the outputs of the CNN passed to the RNN. To evaluate the performance of the proposed approach we have collected real-time walking videos of 100 persons. 10 Sec video of each person is considered as one sample. 70% of the samples are used for training purpose and remaining 30% samples are used for testing purpose. The accuracy achieved by the proposed approach is 78.32%.

1. Introduction

Real-time individual action classification brings applications in various fields including closed circuit television surveillance, individuals presence auto log generation and behavior analysis. Whereas, action classification with high accuracy is a challenging task [1-6]. Techniques like [7-11] designed their approached based on many circumstantial conditions. In reality these conditions may not satisfy. Most of the approaches first extract features from video then apply classifier learning on generated features latter. However, if some of the important features are missing the approaches are going to fail and in most of the cases the features are context dependent. CNN and RNN machine learning approaches [17-21] can learn a hierarchy of features by building high-level features from low-level ones.

Highly complex features can be generated from direct sample images by applying trainable filters and local neighborhood pooling operations. This type of approach is applied in CNN models [11]. In the literature it has been observed that with appropriate parameter tuning CNN models can achieve high performance on visual object detection [15-17].

RNN models allows connection between nodes along a temporal sequence. Therefore, RNN capture temporal dynamic properties. RNN uses its internal states to process sequence of temporal inputs [18-22].

This paper proposes a novel approach for Human action recognition by fusing the CNN and RNN architecture. The input video is converted to a sequence of images and the conjugative image difference sequence is passed to a sequence CNN and the outputs of the CNN passed to the RNN. To evaluate the performance of the proposed approach we have collected real-time walking videos of 100 persons. 10 Sec video of each person is considered as one sample. 70% of the samples are used for training purpose and remaining 30% samples are used for testing purpose. The accuracy achieved by the proposed approach is 78.32%.

2. Convolutional Neural Network

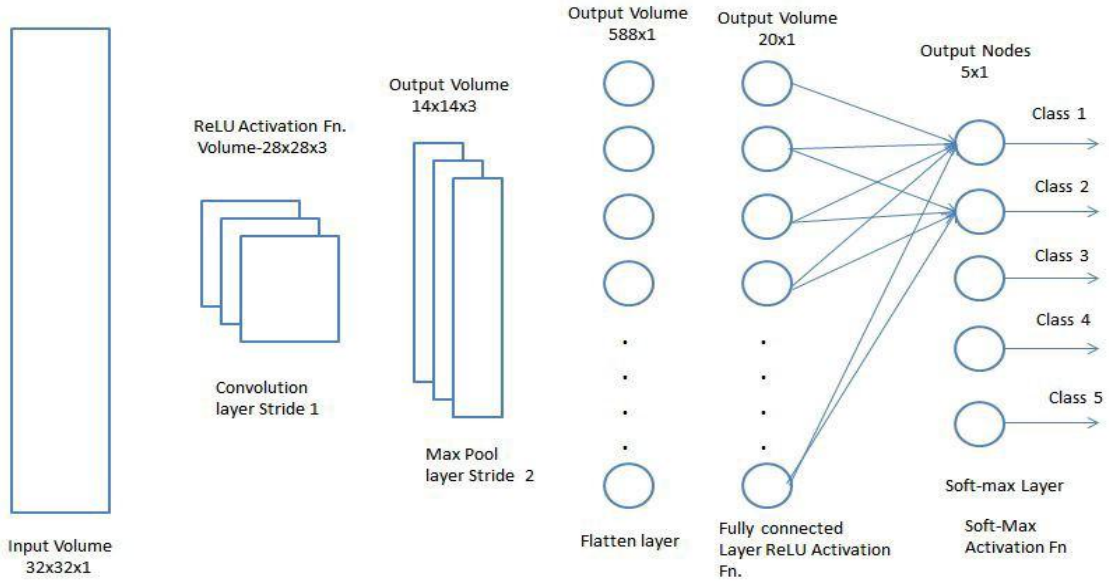
Convolutional Neural Networks are very similar to ordinary Neural Networks from the previous chapter: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g.SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in

a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension.

Fig. 1: Convolutional Neural Network Conceptual Steps



3D volumes of neurons. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. The conceptual steps of CNN is presented in Fig. 1. (Fig.1 is taken from [23])

Recurrent Neural Network

Recurrent Neural Networks are the state of the art algorithm for sequential data and among others used by Apples Siri and Googles Voice Search. This is because it is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for Machine Learning problems that involve sequential data. It is one of the algorithms behind the scenes of the amazing achievements of Deep Learning in the past few years. In this post, you will learn the basic concepts of how Recurrent Neural Networks work, what the biggest issues are and how to solve them. Recurrent Neural Networks (RNN) are a powerful and robust type of neural networks and belong to the most promising algorithms out there at the moment because they are the only ones with an internal memory.

RNN's are relatively old, like many other deep learning algorithms. They were initially created in the 1980's, but can only show their real potential since a few years, because of the increase in available computational power, the massive amounts of data that we have nowadays and the invention of LSTM in the 1990's.

Because of their internal memory, RNN's are able to remember important things about the input they received, which enables them to be very precise in predicting what's coming next. This is the reason why they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more because they can form a much deeper understanding of a sequence and its context, compared to other algorithms.

Recurrent Neural Networks produce predictive results in sequential data that other algorithms can't. "Whenever there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame." Since they are being used in the software behind Siri from Apple and Google Translate, Recurrent Neural Networks are showing up everywhere.

Sequential data is just ordered data, where related things follow each other. Examples are financial data or the DNA sequence. The most popular type of sequential data is perhaps Time series data, which is just a series of data points that are listed in time order. RNN's and Feed-Forward Neural Networks are both named after the way they channel information. In a Feed-Forward neural network, the information only moves in one direction, from the input layer, through the hidden layers, to the output layer. The information moves straight through the network. Because of that, the information never touches a node twice.

Feed-Forward Neural Networks, have no memory of the input they received previously and are therefore bad in predicting what's coming next. Because a feedforward network only considers the current input, it has no notion of order in time. They simply can't remember anything about what happened in the past, except their training.

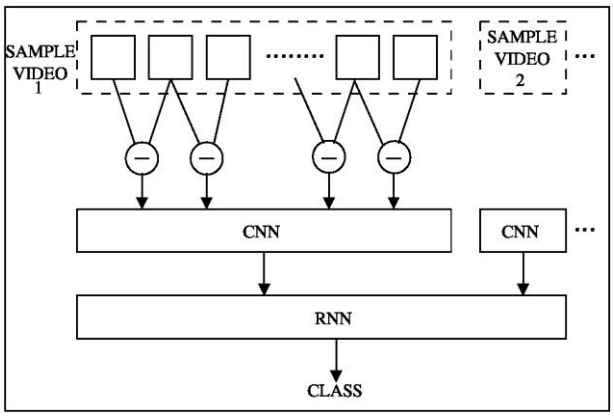
In a RNN, the information cycles through a loop. When it makes a decision, it takes into consideration the current input and also what it has learned from the inputs it received previously. A usual RNN has a short-term memory. In combination with a LSTM they also have a long-term memory. Therefore a Recurrent Neural Network has two inputs, the present and the recent past. This is important because the sequence of data contains crucial information about what is coming next, which is why a RNN can do things other algorithms can't.

A Feed-Forward Neural Network assigns, like all other Deep Learning algorithms, a weight matrix to its inputs and then produces the output. Note that RNN's apply weights to the current and also to the previous input. Furthermore they also tweak their weights for both through gradient descent and Backpropagation Through Time, which we will discuss in the next section below.

3. Proposed Approach

In the proposed approach, two different neural networks techniques are used for improvement of classification. Object Tracking: The target object in the video is selected using OpenCV tracker specified in the cv2 class of python. As CRST tracker was stated as the best, we used it. However, for experimental purposes, we used MIL tracker and Boosting tracker too out of which boosting performed the better. Difference Image: The difference value of two consecutive frames was found out to detect change in the two images. The result is supposed to be fed into the CNN to find out the complete pattern.

Architect of neural networks to be used



The input video is converted to a sequence of images and the conjugative image difference sequence is passed to a sequence CNN and the outputs of the CNN passed to the RNN. To evaluate the performance of the proposed approach we have collected real-time walking videos of 100 persons. 10 Sec video of each person is considered as one sample.

Description of Data Set

The dataset has been generated by us, under the guidance of our guide.The data set is taken by proposing a two camera system by which we can extract data from front view as well as side view.

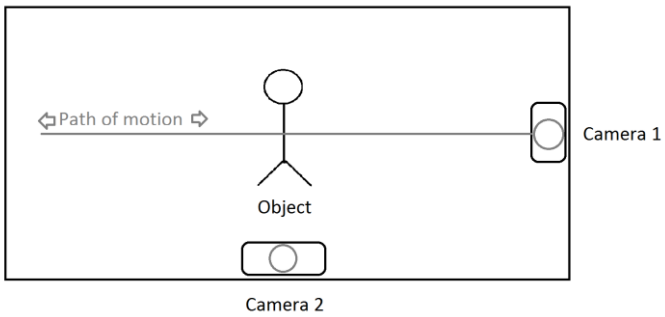


Fig. 4.**Error! No text of specified style in document.**.1 Setup for data generation

As we can see from the set-up how the data was extracted. To be closer to real life scenario, the data was extracted using web cameras which have resolution comparable to surveillance cameras and are cheap to install if implemented. The frames are extracted from the different videos to find the difference image which is fed into the convolutional neural network. As only the image difference is required, the frames are converted into gray-scale and are later thresholded.

4. Results and Analysis

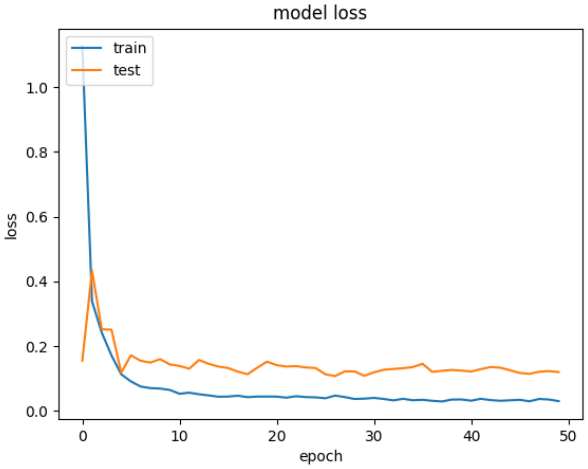
Experiments are carried out using the different architecture by varying the free parameters. Then, the two architectures, with relatively good results were obtained, have been selected for the proposed work. In the present investigation, one architecture of single hidden layer CNN, and one architecture of deeply convoluted CNN are considered. The architectures (number of nodes in the input layer, number of nodes in the hidden layer, number of nodes in the output layer) of the single hidden layer CNN is the number of pixels of the input image. The architectures (number of nodes in the input layer, number of nodes in the first hidden layer, number of nodes in the second hidden layer, number of nodes in the output layer) of the deeply convoluted hidden layer CNN is a factor of the number of pixels as it will reduce the number of nodes in a single layer and can be expanded to other layers. Experiment is carried out using k-fold (here the value of k is 3) cross validation. The patterns are selected randomly to divide the whole data set into k mutually exclusive and (nearly) equal sized subsets. Each of the subsets is considered the test set, and the classifier is trained using the patterns belonging to the union of all other subsets. Then, cross validation is performed (here 3 times) for each training and test set pair. The average values of all the performance-measuring indices (over nine simulations) for seven MLPs and three ensembles are considered.

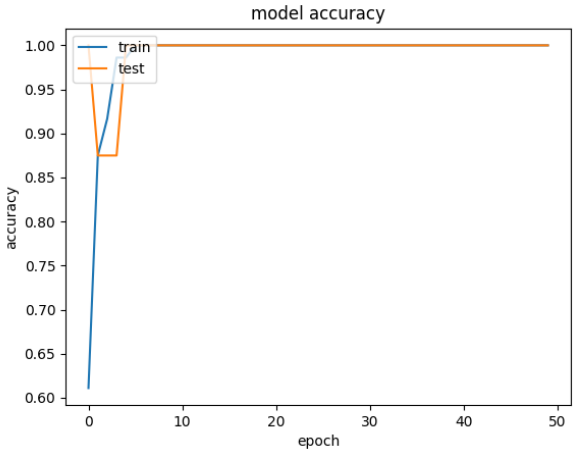
From Table I, it can be seen that the proposed trackers used for object tracking and Charts 1 through 6 show their results. This may be due to the fact that a tracker works well in this given environment. However, we have observed that all the trackers failed terribly to track objects in vertical motion for the frontal view. For this reason, the accuracy of the system suffers as accurate silhouette images couldn't be found out and a lot of

time and energy was wasted during the initial phase of work. At the same time, most of the changed class patterns may also be wrongly assigned to the unchanged class, thereby leading to the increase in the number of missed alarms. It has been found that the trackers worked really well when the video is taken from the side. The probable reason behind this observation is that most research focus on the side view for gait recognition or object tracking and hence the trackers are designed in a certain way according to that. This may yield better performance in the case of creating custom tracker function. However, due to time constraint, it was not possible. As a scope for this project's future implementation, a custom tracker can be made which will increase the accuracy as it will be made according to the requirement of dataset. In the other validation technique, the scope of betterment using an tracker is found to be higher than the cross validation technique due to the usage of self-generated data.

After feeding into the RNN, the result was 100% accuracy in mere 6 epoches out of the 50 performed. As shown in Chart 5.1, the training set shows under fitting for the very early epochs but later is highly accurate. We used the 'keras' model for sequential datasets here. The activation function used was 'softmax'. For calculation of loss, 'Sparse Categorical Cross Entropy' was used, as shown by Chart 5.2.

Model Accuracy





5. Conclusion

In this project, we have worked on different architectures of convolutional neural networks. This gave us a thorough practical knowledge of working with machine learning while using the supervised classification technique. The main architectures undertaken were shallow and deep convolutional network and recurrent neural network for sequential image classification. The respective results were noted of these different architectures. In the present dataset, object tracking failed. Therefore, we conclude that the readily available resources are insufficient for improving the performance of the proposed architectures used. Neural networks have been proven to perform less when given a small dataset.

References

1. Y. Wang and G. Mori, "Hidden Part Models for Human Action Recognition: Probabilistic versus Max Margin," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 7, pp. 1310- 1323, July 2011.
2. H. Wang, M.M. Ullah, A. Kla¨ser, I. Laptev, and C. Schmid, "Evaluation of Local Spatio-Temporal Features for Action Recognition," Proc. British Machine Vision Conf., p. 127, 2009.
3. M. Marszalek, I. Laptev, and C. Schmid, "Actions in Context," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 2929- 2936, 2009.
4. I. Junejo, E. Dexter, I. Laptev, and P. Pe´rez, "View-Independent Action Recognition from Temporal Self-Similarities," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 33, no. 1, pp. 172-185, Jan. 2011.
5. V. Delaitre, I. Laptev, and J. Sivic, "Recognizing Human Actions in Still Images: A Study of Bag-of-Features and Part-Based Representations," Proc. 21st British Machine Vision Conf., 2010.
6. Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning Hierarchical Invariant Spatio-Temporal Features for Action Recognition with Independent Subspace Analysis," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 3361-3368, 2011.
7. A.A. Efros, A.C. Berg, G. Mori, and J. Malik, "Recognizing Action at a Distance," Proc. Ninth IEEE Int'l Conf. Computer Vision, pp. 726-733, 2003.
8. C. Schu¨ ldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," Proc. 17th Int'l Conf. Pattern Recognition, pp. 32-36, 2004. [14] P. Dolla´ r, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," Proc. IEEE Int'l Workshop Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 65-72, 2005.
9. I. Laptev and P. Pe´rez, "Retrieving Actions in Movies," Proc. 11th IEEE Int'l Conf. Computer Vision, pp. 1-8, 2007.
10. H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A Biologically Inspired System for Action Recognition," Proc. 11th IEEE Int'l Conf. Computer Vision, pp. 1-8, 2007.
11. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

12. G.E. Hinton and R.R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks,” Science, vol. 313, no. 5786, pp. 504-507, July 2006.
13. G.E. Hinton, S. Osindero, and Y. Teh, “A Fast Learning Algorithm for Deep Belief Nets,” Neural Computation, vol. 18, pp. 1527-1554, 2006.
14. Y. Bengio, “Learning Deep Architectures for AI,” Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.
15. A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, “Training Hierarchical Feed-Forward Visual Recognition Models Using Transfer Learning from Pseudo-Tasks,” Proc. 10th European Conf. Computer Vision, pp. 69-82, 2008.
16. K. Yu, W. Xu, and Y. Gong, “Deep Learning with Kernel Regularization for Visual Recognition,” Proc. Advances in Neural Information Processing Systems 21, pp. 1889-1896, 2009.
17. H. Mobahi, R. Collobert, and J. Weston, “Deep Learning from Temporal Coherence in Video,” Proc. 26th Ann. Int’l Conf. Machine Learning, pp. 737-744, 2009.
18. C. L. Giles, G. M. Kuhn, and R. J. Williams, “Dynamic recurrent neural networks: Theory and applications,” *IEEE Trans. Neural Networks,* vol. 5, pp. 153–156, Apr. 1994.
19. B. A. Pearlmutter, “Learning state space trajectories in recurrent neural networks,” *Neural Comput.*, vol. 1, pp. 263–269, 1989.
20. A. J. Robinson, “An application of recurrent neural nets to phone probability estimation,” *IEEE Trans. Neural Networks,* vol. 5, pp. 298–305, Apr. 1994.
21. T. Robinson, M. Hochberg, and S. Renals, “The use of recurrent neural networks in continuous speech recognition,” in *Automatic Speech Recognition: Advanced Topics,* C. H. Lee, F. K. Soong, and K. K. Paliwal, Eds. Boston, MA: Kluwer, 1996, pp. 233–258.
22. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error backpropagation,” in *Parallel Distributed Processing, vol. 1*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.
23. S. Saha, “A comprehensive guide to convolutional neural networks--the ELI5 way”, in https://towardsdatascience.com, 2018.

