Project Title

Human Action Recognition

Author

Naimul Hasan
Md. Sabbir Irfan Chowdhury

Report Written By:
Naimul Hasan
Reg: 2017331059
Computer Science & Engineering.
Shahjalal University Of Science & Technology

Abstract

Vision-based human action recognition has received increasing attentions in computer vision and has made significant progress in recent years. In vision-based action recognition tasks, a variety of human actions are inferred based on the complete movements of this action. Recurrent neural network (RNN) and long short-term memory (LSTM) have achieved great success in processing sequential multimedia data and yielded the state-of-the-art results in speech recognition, digital signal processing, video processing, and text data analysis. In this project, we use a convolutional neural network (CNN) for extracting features the video. First, deep features are extracted from the frame of the videos. Next, the sequential information among frame features is learnt using the LSTM network. The created model is capable of learning long term sequences and can process lengthy videos and detect action of the given input video. Our implemented model gave about 74-75% validation accuracy on UCF101 dataset. Though it is not a good accuracy compares to the state-of-the art accuracy, yet it is not that bad even because we were limited by the resources and lack of knowledge of some other better techniques.

Contents

1 Introduction

Human action is the movement of parts of the body via interaction with objects in the environment. Video is a sequential data in which movements in visual contents are represented in many frames such that sequence of frames help in understanding the context of an action.
Action recognition in the video is the task of detecting the action in the video by analyzing the frames of the video. Action recognition in video sequences is a challenging problem of computer vision due to the similarity of visual contents[1], changes in the viewpoint for the same actions, camera motion with action performer, scale and pose of an actor, and different illumination conditions [2]. Human actions range from simple activity through arm or leg to complex integrated activity of combined arms, legs, and body.
One of the key motivations of action recognition is the vast domain of its applications in surveillance videos[3], robotics, human-computer interaction[4], sports analysis, video games for player characters, and management of web videos [5].Action recognition using video analysis is computationally expensive as processing a short video may take a long time due to its high frame rate. As each frame plays an important role in a video story, keeping information of sequential frames for long time, makes the system more efficient. In this project, we recognize human actions in a way similar to our observation of actions in real life. We use LSTM to consider the information of previous frames in automatic understanding of actions in videos. A kind of Recurrent neural network called LSTM is used to analyze frame to frame change of action videos. RNNs are building blocks of a connected neuron with input units, internal (or hidden) units, and output units, having an activation at time t, which can selectively process data in sequence. As it processes one element at a time, it can model outputs, consisting of sequence of elements that are not independent[6]. The RNN architecture provides strength to processing and finding hidden patterns in time-space data such as audio, video, and text. RNN processes data in sequential way such that at each time t, it gets input from the previous hidden state St1 and new data xt. The data is also multiplied with weights, biases are added, and is fed to activation functions. Due to the large number of calculations, the effect of the initial inputs becomes negligible for the upcoming sequence of data after few layers, resulting in vanishing gradient problem. The solution to this problem is LSTM. The main idea of LSTM architecture is its memory cell, input gate, output gate, and forget gate, which can maintain its state over time TN, and nonlinear gating units which regulate the information flow into/out of the cell[7].

2 Related works

Over the last decade, researchers have presented many hand-crafted and deep-nets based approaches for action recognition. The earlier work was based on hand-crafted features for non-realistic actions, where an actor used to perform some actions in a scene with simple background. Such systems extract low level features from the video data and then feed them to a classifier such as support vector machine (SVM), decision tree, and KNN for action recognition.These methods were not feasible for complex and lengthy video. To process complex datasets, we need hybrid approaches which can combine different features and preprocessing such as motion detection [8], background segmentation [9], HOG, SIFT, and SURF. But such hybrid methods increase the computational complexity of the target system. These limitations can cause difficulty for lengthy videos and real-time applications with continuous video streaming. Besides hand-crafted features based approaches for action recognition, several deep learning based methods were also proposed in recent years. Deep learning has shown significant improvement in many areas such as image classification, person re-identification, object detection, speech recognition and bioinformatics [10]. For instance, a straight forward implementation of action recognition using deep networks is developed through 3D convolutional networks by Ji et al. [11]. They applied 3D convolutional kernels on video frames in a time axis to capture both spatial and temporal information. They also claimed that their approach can capture motion and optical flow information because frames are connected by fully connected layers at the end. A multi-resolution CNN framework for connectivity of features in time domain is proposed by [12] to capture local spatio-temporal information. This method is experimentally evaluated on a new YouTube 1 million videos dataset of 487 classes. The authors claimed to have speed up the training complexity by foveated architecture of CNN. They improved the recognition rate for large dataset up to 63.9% but their recognition rate on UCF101 is 63.3%, which is still too low for such important task of action recognition. A two-stream CNN architecture is proposed by [13] in which first stream captures spatial and temporal information between frames and second one demonstrates the dense optical flow of multiple frames. They have increased the amount of data for the training CNN model by combining two datasets. Deep learning based approaches have the ability to accurately identify hidden patterns in visual data because of its huge feature representation pipeline. On the other hand, it requires huge amount of data for training and high computational power for its processing. In this work, we have balanced the complexity of the system and action recognition accuracy. For better action recognition, we have intelligently combined CNN and LSTM due to its state-of-the-art results on visual and sequential data.

3 Methodology

We used “Google Colab” to run the programs. It provides us approx. 12 GB of RAM and variable GPU depending on the traffic. Steps include:

1. Working with the dataset.
2. Extracting features from the frames using CNN.
3. Passing those features into a recurrent neural network to train the model and classifying the videos based on that.
4. Saving the best model and then using it to classify the videos.

3.1 Dataset

UCF101 is one of the most popular action recognition datasets of realistic action videos. It consists of 13320 videos taken from YouTube, which are divided into 101 action categories. Each category contains videos between [100, 200]. UCF101 is comparatively more challenging dataset due to its large number of action categories from five major types: 1) human-object interaction, 2) body-motion only, 3) humanhuman interaction, 4) playing musical instruments, and 5) sports. Some categories have many actions such as sports, where most of the sports are played in a similar background, i.e., greenery. Some of the videos are captured in different illuminations, poses, and from different viewpoints. One of the major challenges in this dataset is its realistic actions performed in real life, which is unique compared to other datasets where actions are performed by an actor.

3.2 Working with the dataset

We downloaded the dataset “UCF101” and extracted in google drive. It contains 13320 videos belonging to 101 different classes. Then we separated the video in two folders namely “test” and “train”. This splitting was based on the test train spilt file provided in their official site. Then, we extracted the frames from each video in their corresponding folders as .jpg files

3.3 Preparation and features extraction

To extract features from the images, we used a pre-trained model provided by Keras named Inception-V3. Inception-V3 is a model provided by Keras whose weights pre-trained on ImageNet dataset, which is a huge dataset consisting of large number of images of different classes. It saves us from building our own architecture from scratch which could take far more time and efforts and not necessarily provide better results than when using Inception-V3. Thus, in our project we used this Inception-V3 model to extract features. In our initial work, we just settled with this work by fine tuning some outermost layers and predicting the class solely based on this. But, we improved our design by using RNN which is shown in next step.

3.4 Passing the features to RNN and training the model

Instead of classifying the images just with CNN, we now use CNN+RNN model. Here the feature output from Inception-V3 are first taken by removing the top classifying layer. It gives a vector of features. We used LSTM, which is a special kind of RNN. It is also provided by keras in python. The features we obtained from Inception-V3, are passed to this LSTM architecture. After that, we defined some of our own layers. We tried with dense layers with 'relu' activation with some dropouts between multiple layers. After trying with different layers, we settled down to a 2048-wide LSTM layer followed by a 512 dense layer and used 0.5 dropout. Note that we used the features from a video as input to the LSTM. Instead of passing features from every frame, we reduced the number of frames to be passed. This reduced overfitting and also made the program faster due to less features being processed now. Features from approx. 30-50 frames per video are sent as sequence to the LSTM. At the final layer, we used 'softmax' activation, which is basically used to provide a class as output in terms of probability, i.e. it turns the input numbers to probabilities that sum to 1. Thus, it outputs a probability distribution. We used "categorical cross entropy" as a loss measure and adam optimizer was used with learning rate of .Also, the metrics for training and testing was 'accuracy' and 'top 5 categorical accuracy'. Then we compiled our model using these parameters and are ready for training.
Thus training is based on above mentioned architecture. The training and testing samples are divided according to the file provided in official website. This ensures uniform division of videos. The model is trained to learn in such a way to minimize the validation loss. When we stop learning, i.e. the validation loss keeps on increasing, then we stop the training. The accuracy metrics is used. When doing the training, we save the model weights when-

ever validation loss is decreased. When training finishes, we get the recently saved weights as the best weights and we will use those weights further in classifying our own videos.

3.5 Saving the best model and classifying Videos

As written above, we have saved the model weights for that iteration where we get the least validation loss. This is generally a good practice because if we save model with best accuracy only, no matter what the loss is, then we can't say that it is the best model. So, least loss is what we used when specifying the best model. Although it may vary depending on the tasks given, but generally for HAR, it is good enough. Now, to classify videos, we just define the same architecture we defined for training. Then we load the model weights and now we are ready to classify the videos. Just take a video, extract frames and then use Inception-V3 to extract features from that video. Pass those features into the architecture and we get the class to which the video belongs.

4 Experimental evaluation

We trained our model on UCF101 dataset using the architecture described in the previous section and the results were obtained. We experimented with different hidden layers added to our architecture so as to get the best model. Earlier the method we used included CNN only and the validation accuracy was very low in that case when compared to the new model we defined. The following graphs shows various accuracies and loss we found out during the experiment:

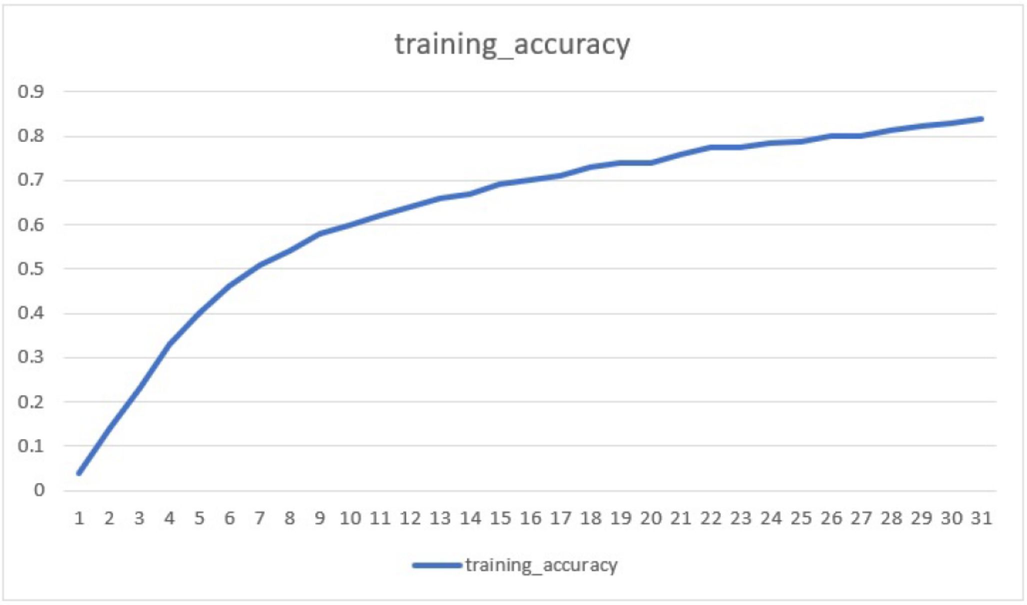

Figure 1: Training accuracy

We reached training accuracy of approx. 84 percent and that accuracy was achieved at around 30 epochs.

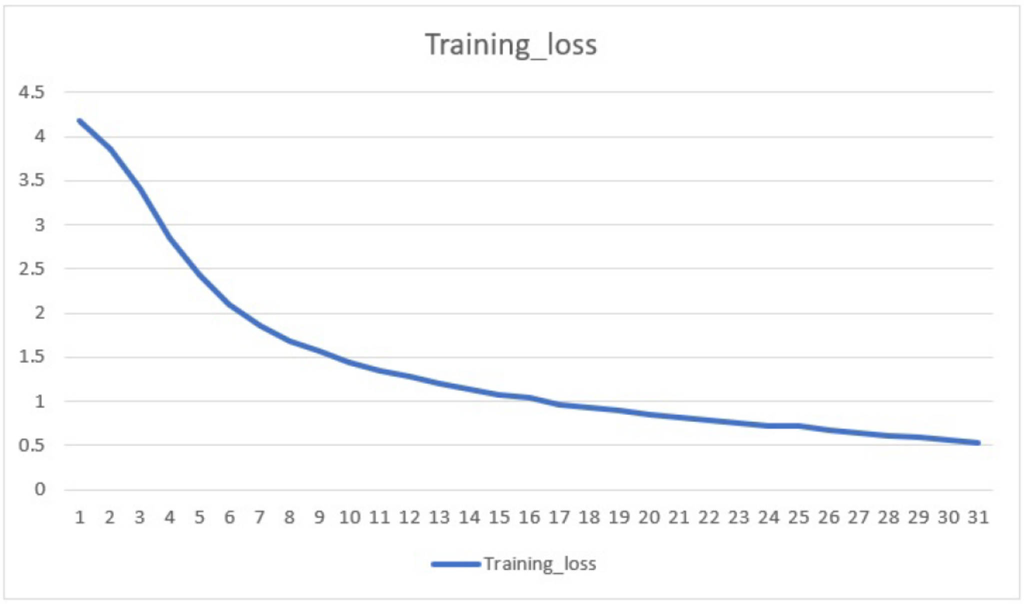

Figure 2: Training loss

The training loss decreased with number of epochs. It dropped down from 4.2 to 0.6 with 30 epochs.

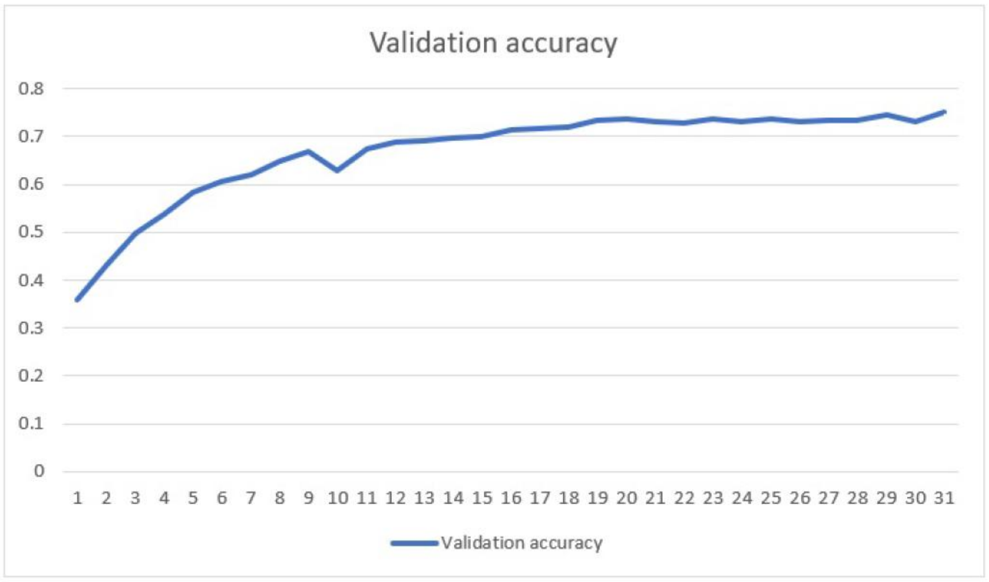

Figure 3: validation accuracy

The validation accuracy we got on UCF101 data was around 74 percent and was achieved with 30 epochs. The validation loss was initially 3.8 and gradually it decreases to around 1 after 30 epochs. In CNN only model, we obtained best loss of 7.2 which is much higher than what we obtained in our new model which was nearly 1. We reached around 25 percent accuracy only on validation data when using the CNN model only. This is very much less compared to 74 percent accuracy we obtained on CNN+RNN model.

5 Conclusion

The implemented model gave about 74-75% validation accuracy on such a big dataset UCF101. And the loss was approximately 1. Whereas, the previous model just gave 25% accuracy with a higher loss. Although this 74% accuracy achieved is not that good when compared to 94% state-of-the art accuracy, yet it is not that bad even because we were limited by the resources and lack of knowledge of some other better techniques. During this whole project, we learned many new things. We learned about neural network, their different types: CNN and RNN. Since we had to study them in detail for the implementation, it enhanced our knowledge about neural networks. With these learnings, we hope to work more towards this field as in the future most of the things will be computational and we would like to keep pace with it. Also, it is very interesting to learn that how a machine is made to act and think like humans. We learned about Human Action Recognition methods and also how it can very useful in several applications. We learned many new libraries provided in Python. It makes it easy to implement our ideas. Main focus of learning was CNN and RNN. We learnt them from scratch and also saw various applications where these can be used.

References

[1] A. Nanda, D. S. Chauhan, P. K. Sa, and S. Bakshi, "Illumination and scale invariant relevant visual features with hypergraph-based learning for multi-shot person re-identification," *Multimedia Tools and Applications*, vol. 78, no. 4, pp. 3885–3910, 2019.

[2] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[3] A. Nanda, P. K. Sa, S. K. Choudhury, S. Bakshi, and B. Majhi, "A neuromorphic person re-identification framework for video surveillance," *IEEE Access*, vol. 5, pp. 6471–6482, 2017.

[4] S. A. Aly, T. A. Alghamdi, M. Salim, and A. A. Gutub, "Data dissemination and collection algorithms for collaborative sensor devices using dynamic cluster heads," *Trends in Applied Sciences Research*, vol. 8, no. 2, p. 55, 2013.

[5] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4694–4702, 2015.

[6] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[7] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[8] S. K. Choudhury, P. K. Sa, S. Bakshi, and B. Majhi, "An evaluation of background subtraction for object detection vis-a-vis mitigating challenging scenarios," *IEEE Access*, vol. 4, pp. 6133–6150, 2016.

[9] S. K. Choudhury, P. K. Sa, K.-K. R. Choo, and S. Bakshi, "Segmenting foreground objects in a multi-modal background using modified z-score," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2017.

[10] S. K. Choudhury, P. K. Sa, R. P. Padhy, S. Sharma, and S. Bakshi, "Improved pedestrian detection using motion segmentation and silhouette orientation," *Multimedia Tools and Applications*, vol. 77, no. 11, pp. 13075–13114, 2018.

[11] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 1, pp. 221–231, 2012.

[12] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[13] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," *arXiv preprint arXiv:1406.2199*, 2014.