Exploring Action Recognition without using Deep Learning

Nakul Agarwal[1]

Abstract— **The goal of this report is to explore action recognition without using deep learning methods. Deep learning has brought about tremendous change in the fields of computer vision and machine learning in the past decade. Convolutional neural networks (CNNs) and the algorithms developed have achieved high accuracy on multiple tasks such as semantic segmentation, image classification, action recognition to name a few. However, since the rise of deep learning, not a lot of papers have shown proper comparison results with non deep learning methods for specific tasks. In this report, I will use some traditional computer vision methods for action recognition and compare their results with the state-of-the-art deep learning method. The experiments are run on the famous JHMDB dataset.**

I. Introduction

Computer vision can be succinctly described as finding and telling features from images to help discriminate objects and/or classes of objects. Computer vision has become one of the vital research areas and the commercial applications bounded with the use of computer vision methodologies is becoming a huge portion in industry. The accuracy and the speed of processing and identifying images captured from cameras has developed through decades. Being the well-known boy in town, deep learning is playing a major role as a computer vision tool.

That being said, a big question which must be answered: Is deep learning the only tool to perform computer vision? No. Deep learning came to the scene of computer vision couple of years back with AlexNet [1]. Before that, computer vision was mainly based on image processing algorithms and methods. The main process of computer vision was extracting the features of the image. Detecting the color, edges, corners and objects were the first step to do when performing a computer vision task. These features are human engineered and accuracy and the reliability of the models directly depend on the extracted features and on the methods used for feature extraction. In the traditional vision scope, the algorithms like SIFT (Scale-Invariant Feature Transform) [2], SURF (Speeded-Up Robust Features) [3], BRIEF (Binary Robust Independent Elementary Features) [4] plays the major role of extracting the features from the raw image. The difficulty with this approach of feature extraction in, for example image classification, is that you have to choose which features to look for in each given image. When the number of classes of the classification goes high or the image clarity goes down, its really hard to cope up with traditional computer vision algorithms.

Deep learning does not face this issue. Deep learning, which is a subset of machine learning has shown a significant performance and accuracy gain in the field of computer vision. Arguably one of the most influential papers in applying deep learning to computer vision, in 2012, a neural network [1] containing over 60 million parameters significantly beat previous state-of-the-art approaches to image recognition in a popular ImageNet [5] computer vision competition. The boom started with the convolutional neural networks and the modified architectures of ConvNets. By now it is said that some convNet architectures are so close to 100% accuracy of image classification challenges, sometimes beating the human eye! The main difference in deep learning approach of computer vision is the concept of end-to-end learning. There is no longer need of defining the features and do feature engineering as shown in Figure 1.

However, deep learning also has its drawback like, need of having huge amount of training data and need of large computation power. Therefore, it would be interesting to see if we can somehow combine the benefits of traditional computer vision methods and deep learning to come up with an even better framework. But to do that, the first step is to understand the difference between both the approaches. In this report, I make a small step towards understanding this difference by exploring a particular task, i.e. action recognition, and showing quantitative results using both deep learning and traditional computer vision methods.

II. Methods

A. Dataset and Metric

Dataset. JHMDB [6] is a subset of the famous HMDB dataset [7], It consists of 928 trimmed clips over 21 classes involving a single person in action: **brush hair, catch, clap, climb stairs, golf, jump, kick ball, pick, pour, pull-up, push, run, shoot ball, shoot bow, shoot gun, sit, stand, swing baseball, throw, walk, wave**. It has 36-55 clips per action class with each clip containing 15-40 frames. In summary, there are 31,838 annotated frames in total. The person performing the action in each frame is annotated with his/her **2D joint positions, scale, viewpoint, segmentation, puppet mask and puppet flow**. The correspondence between body joints and x,y coordinates is shown in Figure 2. There are 3 training and validation splits, and results are averaged over all the splits.

Metric. I use the standard clip level accuracy to report the results.

[1] University of California, Merced nagarwal2@ucmerced.edu

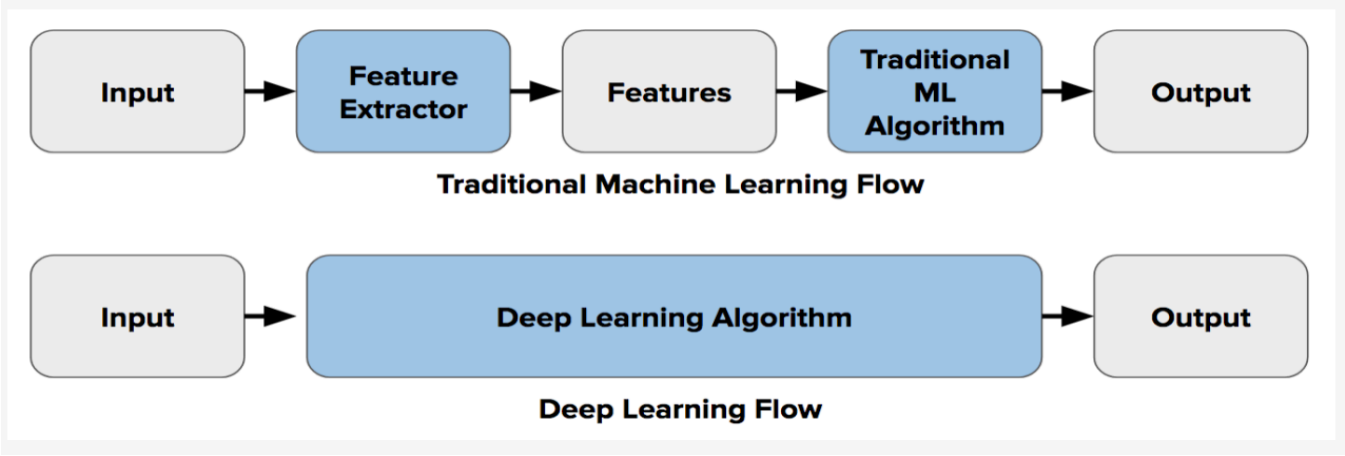

Fig. 1. Traditional machine learning flow vs deep learning flow

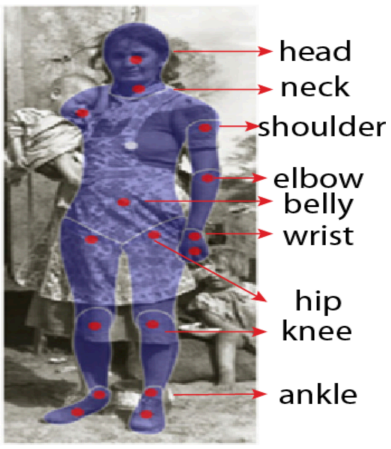

Fig. 2. The correspondence between body joints and x,y coordinates in the JHMDB dataset

B. Features

I use the following types of features and their representations:

- **Pose**
  For each frame, I have the x- and y coordinates of 15 joints, as shown in Figure 2. I first normalize joint positions w.r.t the scale of the underlying puppet. I then use 3 types of features:
  - translation of the normalized joint positions along the x and y-coordinates (dx, dy).
  - the direction of the translational vector ($\arctan(\frac{dy}{dx})$)
  - relative positions of normalized joint positions w.r.t the puppet center in a sequence of T frames. T is set to 7 empirically.
    Note that due to the nature of the puppet annotation tool, all 15 joint positions are available even if they are not annotated when they are occluded or outside the frame. In this case, the joints are in the neutral puppet positions. Unless otherwise specified, I use all 15 joints regardless of their visibility. There are totally 75 descriptor types (30 for positions, 30 for translations, and 15 for directions).
- **Pose+**
  Since it has been shown in [8] that relational features describing geometric relations between joints perform better than using normalized joint positions, I also extract a set of relational features:
  - $\binom{15}{2} = 105$ distances between all the pairs of joints
  - 105 orientations of the vector connecting two joints
  - 3 $\binom{15}{3} = 1365$ inner angles spanned by two vectors connecting all the triples of joints.
    All possible relational features are computed for each frame, yielding 1575 descriptor types. In addition to using relational features, I also use the differences of relations between adjacent frames. I also use pose features described previously (Pose). Therefore, there are in total 3225 descriptor types (75 for Pose, 1575 for relations, and 1575 for their difference).
- **I3D**
  I3D [9] is a very popular 3D convolutional deep model used for action recognition. In this report, I make use of I3D as a reference to compare with the traditional computer vision methods.

III. Results

The results can be seen in Table I. Below are some implementation details.

A. Implementation details

Pose and Pose+. For each descriptor type, all the training samples are used to generate a BoW (Bag of Words) [10] model. I use a vocabulary size of 20, as each descriptor has small dimensionality. For classification, I use a non-linear SVM with RBF kernel. The multi-class classification is done by LIBSVM [11] using a one-vs-all approach.

Method	Split 1	Split 2	Split 3	Overall
Pose	70.35	69.20	68.50	69.35
Pose+	76.20	75.10	74.90	75.40
I3D	83.91	82.30	81.80	82.67

TABLE I
QUANTITATIVE RESULTS USING DIFFERENT METHODS ON JHMDB

I3D. The I3D model I use has been pre-trained on ImageNet [5] and Kinetics [9] dataset. I fine-tune on JHMDB using the pre-trained model. Each input sequence has 20 frames. I use a batch size of 8 along with batch normalization. Model is trained for 10k iterations using the Pytorch [12] framework and 2 Nvidia Titan Xp GPUs.

IV. DISCUSSION

As expected, I3D performs the best. However, the pose features extracted using the ground truth joint positions and a SVM classifier also perform reasonably well. Also, extraction and classification of these pose features takes less time than training and evaluating the I3D model. This shows that the *'right'* hand-crafted features and traditional computer vision approaches can also do well while being more efficient. However, the catch here is that it's difficult to identify the *'right'* kind of features using traditional computer vision methods for a particular task.
Because of the above reason, it would be interesting to see if we can somehow combine the benefits of both traditional computer vision approaches and deep learning.

V. CONCLUSION

In this report, I make an attempt to understand the difference between traditional computer vision methods and deep learning model on a particular task, i.e. action recognition. I do this by quantifying results on a popular action recognition dataset, **JHMDB**, using a deep learning model and traditional computer vision approaches. Results show that although the deep learning model performs better, the traditional computer vision approaches (Pose features + SVM) also perform reasonably well and it would be interesting to see if we can somehow leverage the benefits of both traditional computer vision approaches and deep learning to come up with an even better framework.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[6] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. J. Black, "Towards understanding action recognition," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 3192–3199.

[7] H. Jhuang, H. Garrote, E. Poggio, T. Serre, and T. Hmdb, "A large video database for human motion recognition," in *Proc. of IEEE International Conference on Computer Vision*, vol. 4, no. 5, 2011, p. 6.

[8] A. Yao, J. Gall, and L. Van Gool, "Coupled action recognition and pose estimation from multiple views," *International journal of computer vision*, vol. 100, no. 1, pp. 16–37, 2012.

[9] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[10] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22. Prague, 2004, pp. 1–2.

[11] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[12] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.