Paper Review: Two-Stream RNN/CNN for Action Recognition in 3D Videos

Gianluca Silva Croso

Reference for paper reviewed:

R. Zhao, H. Ali, and P. van der Smargt, "Two-Stream RNN/CNN for Action Recognition in 3D Videos," arXiv preprint arXiv:1703.09783, 2017

1. Introduction

1.1 Project

My project consists of creating a machine learning pipeline for classifying surgical activities. Specifically, we are working within the context of cataract surgery in a query-by-video approach. Given a database of pre-classified videos, a new query would be made and the most similar video in the database would be found.

1.2 Paper Selection

I selected this paper for review because it tackles a similar problem – action recognition – with a specific focus on how to fuse both spatial and temporal information and fuse more than one type of model together, both of which will be necessary for the successful completion of my project.

1.3 Key Contribution and Results

This paper successfully improves on current state-of-the-art results on a relevant dataset by a significant amount (14%), and contributes in the following meaningful ways:

- Proposes a novel RNN structure which converges faster and costs less computational power than the more common LSTM-based models
- Introduces two fusion methods for two-streamed networks to combine the proposed RNN structure and 3D-CNN structure: decision fusion and feature fusion, where the first is easier to implement but the latter achieves better performance

2. Background

2.1 Recurrent Neural Network

Throughout the paper, the authors experiment with several RNN architectures. Recurrent Neural Networks are network architectures that can "handle sequence information with varied length of timesteps" [1]. Specifically, the a vanilla RNN has a hidden state h_t at each time step t that is determined by the current input x_t and the previous hidden state h_{t-1}.

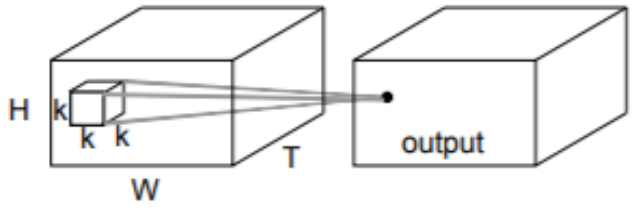
$$\mathbf{h}_t = \sigma\left(\mathbf{W}\begin{pmatrix}\mathbf{x}_t \\ \mathbf{h}_{t-1}\end{pmatrix}\right)$$

$$\mathbf{y}_t = \sigma\left(\mathbf{V}\mathbf{h}_t\right)$$

Here σ is a non-linear activation function, usually the standard logistic function or hyperbolic tangent. This allows the network to retain information from the entire sequence of inputs. This architecture, however, is ineffective for long timestep sequences due to the vanishing gradient problem.

An improvement to the vanilla RNN unit was the development of LSTM (Long Short-Term Memory) units. This architecture uses three gated cells to retain more information about the different hidden states and properly propagate errors through hundreds of thousands of timesteps. A Gated Recurrent unit is a simplification of LSTM that uses only two gates instead of three. This unit is as effective as LSTM at retaining information but can be trained and computed faster. A Bidirectional Recurrent Neural Networks is an RNN which iterates over all the timesteps twice, a forward pass and a backward pass. Batch normalization is a technique to speed-up training and converging by reducing the *internal covariate shift* [2], the variations to the distribution of each layer's inputs during training due to the changes in the previous layers parameters.

2.2 Convolutional Neural Network



A CNN is a network architecture based on convolutional filters to learn spatial features on progressively more general scale. In this case, it is the network used to learn features from RGB videos. "Regular 2D convolutions generate a series of feature maps from image" [1]. Here, instead, 3D convolutions are used to process frame clips, with timestep as the third dimension. The figure above illustrates the process of obtaining a feature descriptor from an input volume, and the descriptor should be able to encode spatiotemporal information across a small number of timesteps.

2.3 Other technical concepts

A few other background technical concepts are necessary to understand this paper, and we will *briefly* describe them below:

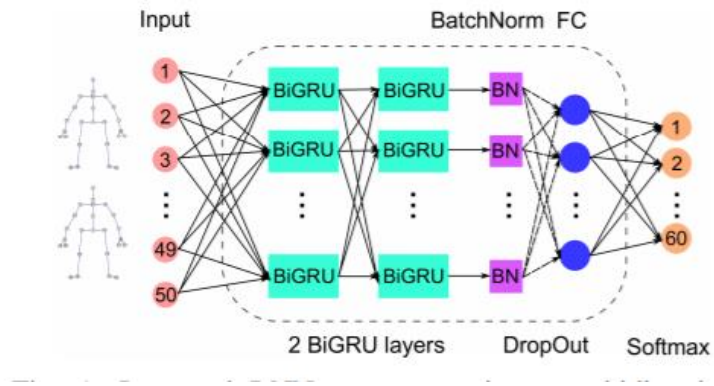
- Support Vector Machine: Multi-class classifier that attempts to maximize the boundaries between distinct classes

- Fine-Tuning: Training a neural-network that has been initialized with pre-trained weights based on another dataset
- Dropout: Training technique to prevent overfitting where some layers are skipped during backpropagation with a pre-determined probability

3. Methodology

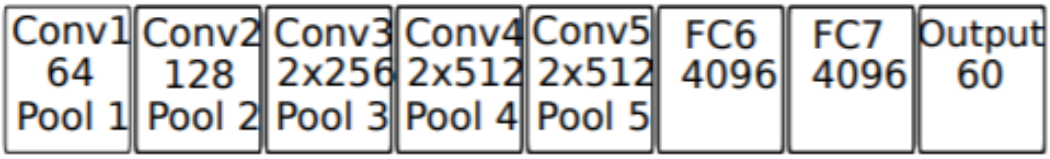
3.1 RNN

The proposed structure for RNN consists of two layers of bidirectional GRU units followed by Batch normalization, and then a fully connected layer. These are trained with droupout with probability 25%. Finally, a softmax layer is used to map the action features to the 60 possible classes. This architecture was designed by the authors and outperforms other state-of-the-art RNNs on this dataset. It is illustrated below



3.2 CNN

The CNN architecture used is directly from the work of *Tran et al.* [3], which is among the best known RGB based models for this problem. It is a 3D-CNN with the following architecture

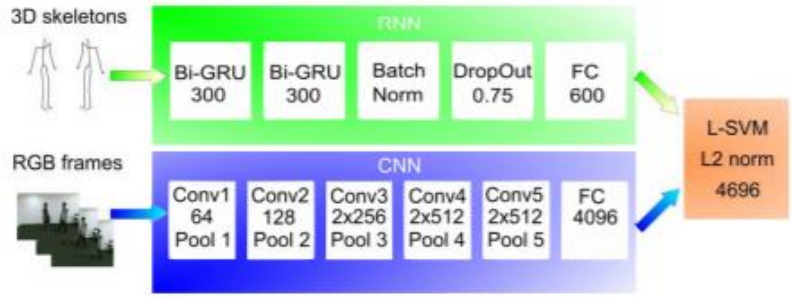


The training step consisted on finetuning the pretrained weights from the Sports-1M dataset.

3.3 Two-Stream RNN/CNN

The authors took a novel approach in using both RNN and CNN to classify the video data, and then fuse the information obtained from each network to obtain an improved final classification. The RNN structure was used to classify based on skeleton data, while the CNN architecture was

applied to the accompanying RGB video. Two ways of fusing this information were proposed by the authors



3.3.1 Decision Fusion

This method consists of classifying the inputs with both networks (streams) independently and using a voting-like approach for final classification. Both networks are trained on the same training set and validated on the same validation set. The softmax layer will provide a probabilistic confidence level on the classification. Then, the decision of each stream is weighted by a parameter proportional to the accuracy of the network on a validation dataset.

3.3.2 Feature Fusion

Instead of considering the final decision of each network after the softmax layer, here we take the feature representation of the output after the first fully connected layer in each model. Once again, the networks are trained independently on the same training set. These features are then concatenated, L2 normalized and the fused feature vector is classified into one of the 60 classes by a linear SVM. The SVM parameter is determined based on the validation dataset. It is important to maintain a third, independent test set for final testing of the architecture.

4. Experiments

This model was trained and evaluated on the NTU RGB+D dataset. As described by the authors, “The dataset consists of more than 56k action videos and 4 million frames, which were collected by 3 Kinect V2 cameras from 40 distinct subjects, and divided into 60 different action classes including 40 daily (drinking, eating, reading, etc.), 9 health related (sneezing, staggering, falling down, etc.), and 11 mutual (punching, kicking, hugging, etc.) actions. It has four major data modalities provided by the Kinect sensor: 3D coordinates of 25 joints for each person (skeleton), RGB frames, depth maps, and IR sequences.” Only the first two modalities are used in the paper. The two ways in which the dataset is evaluated are cross-subject, in which half of the subjects are used for training and validation and the other half left for testing, or cross-view, in which 2 of the viewpoints are used for training and validation and the last left for testing.

The authors implement the architectures as described in the Methodologies section, including both fusion alternatives. For the RNN architecture, the authors also perform several tests with simpler architectures utilizing fewer or simpler RNN units.

5. Results

Nr.	Method	cross subject	cross view
01	Skeleton Quads [2], [9]	38.62%	41.36%
02	Lie Group [2], [10]	50.08%	52.76%
03	FTP Dynamic Skeletons [2], [11]	60.23%	65.22%
04	HBRNN-L [2], [3]	59.07%	63.97%
05	Deep RNN [2]	56.29%	64.09%
06	Deep LSTM [2]	60.69%	67.29%
07	Part-aware LSTM [2]	62.93%	70.27%
08	ST-LSTM (Tree) + Trust Gate [4]	69.2%	77.7%
09	1 Layer RNN	18.74%	20.27%
10	1 Layer LSTM	60.99%	64.68%
11	1 Layer LSTM-BN	64.07%	71.86%
12	1 Layer LSTM-BN-DP	64.69%	73.48%
13	1 Layer GRU-BN-DP	65.21%	70.36%
14	1 Layer BI-GRU-BN-DP	64.78%	73.12%
15	2 Layer BI-GRU-BN-DP	66.21%	72.46%
16	2 Layer BI-GRU-BN-DP-H	70.70%	80.23%
17	3D-CNN [8]	79.75%	83.95%
18	Decision Fusion	82.05%	86.68%
19	Feature Fusion	**83.74%**	**93.65%**

The first 8 rows are other RNN based methods presented previously. Rows 9-16 are experiments run by the authors with varied RNN architectures, showing the best results when using the architecture proposed in the methodology (2 Layer BI-GRU-BN-DP-H). Layer 17 displays the results of *Tran et al.* [3] on RGB videos, which outperforms the pure RNN methods. The last two rows demonstrate the improvement achieved by fusing both types of information. Decision fusion improves the 3D-CNN results by an order of 3%, and Feature fusion by an order of 10-14%.

The authors also provide confusion matrices showing that most errors by the network were related to action classes that are intrinsically similar, such as taking off and putting on shoes or clapping and rubbing hands.

6. Assessment

In our evaluation of this paper, we considered the following points in favor and against it

6.1 In favor

- The paper provides and in-depth explanation and experimentation of technical aspects of the chosen RNN architecture
- The paper achieves strong results in improving performance against current state-of-the-art methods for this dataset
- The paper clearly presents two valid alternative fusion methods, both of which are successful in improving accuracy when compared to any single-streamed approach
- The dataset is varied with multiple viewpoints, and includes actions that are hard to distinguish

- Confusion matrix shows that mistakes are directly related to similar looking actions – putting and taking off shoes, clapping and rubbing hands, etc.

6.2 Against

- Despite the title, the data being processed is not exactly what would come to mind given the description as 3D video. RGB+D data not used, and the skeleton data used by the RNN is more analogous to kinematic data, while the CNN uses only RGB
- In row 12 of the results table, LSTM-BN-DP performs as well as or better than other single layer RNN architectures, but no 2-layer version is considered or presented
- It is not clear how the two-streamed network with CNN and RNN conciliates the fact that RNN considers the entire video but CNN only 16 frames at a time

6.3 Relevance for project

This paper introduces important alternatives and distinctions in architectures for obtaining both spatial and temporal features. It provides a very in-depth breakdown of RNN architectures which we could use for our video-feature extractor. Finally, it displays alternatives for fusing different types of information. Unfortunately, the latter point is not directly applicable to our project. Since we have no kinematic data, in our case the output of the CNN would be the input to the RNN. However, we may want to use similar fusion approaches to include tool labeling information. A point that should be made is that, even though the dataset considered in this paper includes good variation, our dataset is even harder.

7. Conclusion

We conclude that this paper is very valuable in providing a solid theoretical background to the use of RNN and CNN for action recognition, as well as how to fuse information from both types of architectures. The paper achieves strong results in improving accuracy on this dataset, and two-streamed networks are flexible and could have many applications. This paper gave us good information that can be useful for our project. It is worth mentioning, however, that it is a very technical overview of the techniques and it is not clear that the authors had any particular application in mind.

8. References:

[1] R. Zhao, H. Ali, and P. van der Smargt, “Two-Stream RNN/CNN for Action Recognition in 3D Videos,” arXiv preprint arXiv:1703.09783, 2017

[2] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.03167, 2015.

[3] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, 2015, pp. 4489–4497.