Action Recognition using C3D

Brian Finnerty
Rochester Institute of Technology
1 Lomb Memorial Dr, Rochester, NY 14623
bpf7056@rit.edu

Adhiraj Sood
Rochester Institute of Technology
1 Lomb Memorial Dr
as9125@rit.edu

Abstract

We implement, understand, and explore an effective approach to action recognition problems using a large-scale labeled video dataset combined with a deep 3-dimensional convolutional network. Through the implementation of C3D [1], a simple, yet efficient algorithm that utilizes a 3-dimensional kernel to preserve spatial and temporal features, we have found that such an approach is easy to train and use while providing solid accuracy.

1. Introduction

In this digital age, the explosive growth of the number of videos online has led to an ever-increasing desire to efficiently analyze these videos. With over 300 hours of video uploaded every minute to YouTube alone, it becomes increasingly clear that to use these videos efficient and effective approaches are required

Originally introduced by Tran et al. [1], C3D leverages its properties of being generic, compact, efficient, and simple to fill the role of a generic video descriptor. While there are many similarities to 2-dimensional convolutional networks, one noticeable difference between any 2D ConvNets and C3D is the lack of temporal information. 2D ConvNets are not suitable for video tasks because they lack motion modeling [1] as a 2D kernel only captures the spatial features of any one image at a time. C3D on the other hand can utilize many spatial features over multiple frames, thus preserving temporal information as well. In this paper, we continue to explore the uses and implementation of C3D to evaluate its authors' claims of being conceptually simple and easy to train and use while furthering our understanding of video problems in the computer vision domain. In summary, our contributions to this paper are:

- We provide our own implementation of C3D, utilizing the general algorithm outlined in [1].
- We demonstrate that C3D is simple and easy to use though our own experimental results.

2. Related Work

Videos are a useful source of information, and this has become more prevalent as the amount of data has explosively increased over the past few decades. While we explore the implementation of C3D in this, 3D CNNs are not just limited to producing a generic video descriptor. Molchanov et al. [2] utilize spatiotemporal data in a 3D CNN with multiple spatial scales to provide a robust approach to Hand Gesture Recognition. Maturana and Schere [3] propose VoxNex, an architecture that integrates a supervised 3D CNN and volumetric Occupancy Grid to accomplish real-time object recognition for LiDAR and RGBD camera data. Ullah et al. [4] utilize a similar approach to C3D, with spatiotemporal features extracted from 16 frame clips through the use of a deep 3D CNN and a SoftMax classifier for Violence Detection. In addition to some simpler implementations of 3D CNNs Molchanov et al. [5] utilize a Recurrent 3D CNN for automatic detection and classification of dynamic hand gestures.

While many of the above simply scratch the surface of the use of videos in computer vision tasks, it can be seen that there are a variety of areas still improving in the field.

3. Method

In this section, we will explain the in-depth implementation of the 3D Convolutional Neural Network.

3.1 Implementation

Following the layout of Tran et al. [1] work, our implementation of C3D is the following. The input to the network is defined by $c \times d \times h \times w$ where c is the number of channels for the video, d is the number of frames in each clip, and h and w are the height and width respectively of each video. Our C3D had 8 convolutional layers, 5 max pooling layers, and 3 fully connected layers followed by a softmax classifier. Each convolutional layer and every fully connected layer besides the last were inputted through a

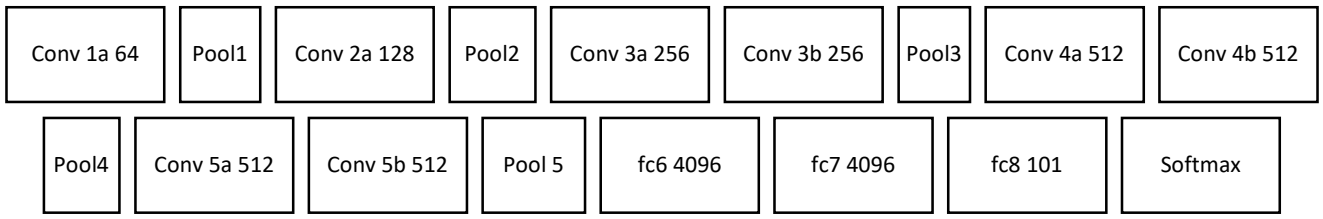

Figure 1: This outlines the architecture of the Convolutional Neural Network for our C3D implementation. Within each box are the number of outputs for each layer. A 3x3x3 kernel was applied to all convolutional layers with a stride of 1. For Pool1 a 1x2x2 filter was applied with a 1x2x2 stride. All other Pool function have a 2x2x2 filter and 2x2x2 stride. The softmax classifier follows the fc8 as the fc8 fully connected layer transforms the output to the number of labels for the UCF-101 dataset.

RELU function as that had improved the accuracy of our model. View Figure 1 for a more detailed look into the complete structure. For each layer, padding was added when it was necessary to allow the network's forward pass to run to completion.

3.2 Dataset

The dataset used for training and testing our C3D implementation was the UCF-101 dataset. The dataset has 13,320 videos which fall into 101 different categories of human actions. The videos were initially split up in three ways training, testing, and validations sets. Each video was resized to 128 x 171 resolution, which was about half of the resolution of the original frame. The data was divided into 16-frame clips that did not overlap and were faced with random crops to a size of 3 x 16 x 112 x 112 down from its original 3 x 16 x 128 x 171. Roughly 64% of the data was placed into the training set where each subcategory in the training set was a specific human action label. Following the same scheme, 20% of the data was placed into the testing set and 16% was stored in the validation set. All of this happened as a preprocessing step before the C3D algorithm was executed. Clips were broken down into 16-frame segments and stored under their appropriate action label in either the training, testing, or validation set.

3.3 Training

The model was trained using the C3D implementation. In this training implementation, the model was either trained from scratch if the resume epoch value is 0 otherwise, we start from the previous checkpoint. In our project, we used the already trained model due to GPU restrictions. To start the training, we set the parameters which have the dataset, the directory where we must store, the number of classes which are 101 for ucf101, the learning rate, and the number of epochs. We set the train parameters of our C3D model based on the learning rate. We set the optimizer using stochastic gradient descent and pass the training parameters. The StepLR is used to divide the learning rate by 10 after every 10 epochs. The dataset was fetched using the torch DataLoader and bifurcated the training, validation, and testing datasets. The clip length for all datasets was set to 16. We start training the model using the training dataset. After computing the forward pass, we fetch the inputs and labels using the torch variable that results in the computational graph. Then we clear the gradient and set the gradient to zero. While training we calculate the loss and perform a backward pass and a single optimization step. After performing all the steps, we calculate the epoch loss and the accuracy of the model with the time taken for training the model. The model at the end was saved in the directory passed in the train method arguments. The model's name is stored in a format so that we can get the epochs performed and can resume from the epoch later on.

4. Experiments

Here we go into detail about the setup process and experimental results of our C3D implementation on the UCF-101 dataset.

4.1 Dataset and Features

As stated previously, the dataset we used for training C3D to recognize action recognition features is UCF-101. This was divided into a 64%,20%, and 16% split for training, testing, and validation sets respectively. For both training and testing, when a clip was loaded, the 128x171 clip was further processed to apply the cropping to jitter the image with the 112 x 112 crop. For testing data, this was further preprocessed when the clip was loaded as there was a 50% chance of the clip being flipped horizontally. The 16-frame clip would then be used to train or test the model. As each clip was associated with a directory denoting its related actions, the list of labels was stored by the DataLoader and written to a file for easy reference.

Figure 2: Display of key frames of C3D's action recognition prediction. Above a correct classification (top) and misclassification (bottom) are labeled with C3D's predicted label and confidence probability.

4.2 Model and Testing

The processes of testing the C3D model are as follows: each 16-frame clip was fed to the trained C3D model, features were extracted from the model, and they were placed through a softmax classifier to provide a prediction for the action label. Then the loss and accuracy of the test were recorded. In order the get a better understanding of the accuracy of our model, we also produced outputted videos that label both the prediction and the actual value. Refer to figure 2 for a more detailed explanation.

4.3 Results

After a run of roughly 30 minutes with 2701 clips comprising the testing set, our C3D trained model was able to achieve an accuracy of 66.34% with an epoch loss of 1.33. This was accomplished through the use of the pretrained C3D-ucf101_epoch-19.pth.tar model that was provided to us. Although we would have liked to further tinker and test with our model, time constraints with the system we were using would have caused a full training session to take several days due to the hardware we had on hand. These results were of our final implementation of the C3D model. Previous iterations utilized different forms of padding and not all layers were followed by a RELU layer as we were curious about how the different implementations would alter the model.

5. Conclusion

In this paper, we attempted to implement and understand the C3D algorithm as an action recognition algorithm utilizing spatiotemporal features for videos and a 3D convolutional network. Using the large-scale video dataset UCF-101, we were able to produce somewhat straightforward and simple. With a multitude of the design decision to make, it required an iterative testing the process between Brian and Adhiraj to settle on the optimal implementation of the C3D model.

Brian's primary contributions to this project were the implementation of the forward function and initialization of the C3D model, as well as performing iterative tests to improve the results of the model until they were satisfactory.

Adhiraj was responsible for the implementation of the labeled test videos. After Brian implemented the model to a satisfactory level, Adhiraj worked on generating and selecting demonstrations of action recognition to be provided in the report and the general project.

As previously stated, given more powerful machines we would have enjoyed tinkering with the C3D model. Seeing as each training epoch would have taken roughly 4 hours, a 19-epoch iteration of the model would have been computationally and resource-intensive and would not guarantee a more accurate model at the end of the day.

6. References

[1] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, Manohar Paluri; Learning Spatiotemporal Features with 3D Convolutional Networks; Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4489-4497

[2] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, Jan Kautz; Hand Gesture Recognition With 3D Convolutional Neural Networks; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2015, pp. 1-7

[3] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 922-928

[4] Ullah, F.U.M.; Ullah, A.; Muhammad, K.; Haq, I.U.; Baik, S.W. Violence Detection Using Spatiotemporal Features with 3D Convolutional Neural Network. Sensors 2019, 19, 2472.

[5] Pavlo Molchanov, Xiaodong Yang, Shalini Gupta, Kihwan Kim, Stephen Tyree, Jan Kautz; Online Detection and Classification of Dynamic Hand Gestures With Recurrent 3D Convolutional Neural Network; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4207-4215