Sapienza University of Rome

Convolutional 3D
Body Movement Recognition

Ibis Prevedello, Jean-Pierre Richa

November 4, 2018

1 Introduction

The goal from this project was to train a model based on Convolutional 3D to recognize the human body movements in videos. To realize this task, the C3D network was used in combination with Openpose, which are coded using tensonflow. Openpose network was essential here, to extract the body pose, and apply it to the frame extracted from the video, then to train the C3D network to recognize the body motion.

The dataset used for the training was a subset of the Human 3.6M collection, which were taken in a local laboratory, and then labeled by the students who were taking the course of elective in Artificial Intelligence, which is a computer vision based elective course. The labeling was done using a son file, which was later split into a training set and a test set. In order to have a well balanced dataset for all of the classes, half of the classes that has an almost equal number of labels were considered for the training.

First, the images with the poses applied were converted to tfrecords in order to minimize the training complexity and time.

The training was done using a Google Compute Engine instance running a Tesla K80 GPU

2 Dataset Preparation and Augmentation

The labeled dataset contained in the json file was split into test set and training set, which are composed of 52 classes. Due to the difference of labeled samples between the classes, those who has the highest numbers were considered for the training.

A total of 26 classes were taken into account to train the model, because the limited number of samples of the other classes provides overfitting a bigger chance to occur, and reduce generalization to other data. Check the **Classes section** for details about the used labels.

Classes

Head	
	Turn right
	Turn left
	Raise
	Lean forward
Right/Left Arm	
	Shoulder extension
	Shoulder adduction
	Shoulder flexion
	Shoulder abduction
	Elbow flexion
	Elbow extension
	Roll the wrist
Head	
	Turn right
	Turn left
	Raise
	Lean forward

Since the data used was also not enough for the training to generalize, augmenting the data was necessary. In order to maintain data consistency the following approach was applied; the idea was to flip the images labeled as arm and leg and add them to the opposite class, so for example the right shoulder extension becomes left shoulder extension and vice versa. After performing these manipulations, the final dataset was extended to 400 samples for each class, with a total of 10,400 samples for the training set and 1900 samples for the test set, which enlarged the original dataset by approximately the third.

3 Body Movement Recognition

Body movements are too complex to be recognized, because when a person moves his/her head or any body part, only small variations can be seen in the features extracted. The main problem is that not only the specific part that the person intended to move is affected, but also other parts of the body, and this holds especially if the classifier should predict the movements in the real life, which is the case in this project. Even if the movement is small, it will affect the features extracted from other areas also, and each time with a different

intensity, which makes it even harder to predict which part should be the main focus.

For these reasons, it was decided to use the help of openpose to extract a map of the current pose, which gives an accurate pose estimation of the body state, that should help the Convolutional 3D network to predict the body movement more accurately.

After estimating the pose of the body using the pre-trained model of openpose, the extracted maps were applied to the corresponding frames and then fed into the Convolutional 3D network using batches of 10 samples.

4 Network Training

The network was trained using transfer learning, which was achieved by employing the weights of the pre-trained model **sports1m finetuning ucf101**, used to classify sports activities. It wasn't that easy to decide on a number of epochs that's enough to get a good accuracy, so after some testing it was discovered that the best to prevent overfitting and get the highest accuracy possible was 30 epochs, with a batch size of 10 samples.

Following the transfer learning approach, the last layer was deleted in order to exclude the final classification results of the previous model, and to be able to include the new total number of classes for this specific task. The training consisted of 2 phases, the first one was accomplished through 1^{st} training only the last layer weights for 3 epochs with a learning rate of $\mathbf{1 \times 10^{-3}}$, 2^{nd} training the weights of the whole network with a learning rate of $\mathbf{1 \times 10^{-4}}$.

After each epoch, the updated weights were used to get the accuracy of the prediction made on the training set and the test set.

Two different loss functions were used to train the network and calculate the loss:

- **tf.nn.l2_loss:** the Least Squares Errors function, which was used inside the fully connected hidden layers, and its mathematical representation can be seen **below**

$$L2LossFunction = \sum_{i=1,n} (y - \hat{y})^2 \tag{1}$$

- **tf.nn.softmax_cross_entropy_with_logits:** the Cross Entropy loss function, which was used for calculating the loss on the output layer, then its gradient was calculated and back propagated through the network to update the weights. Its mathematical representation can be seen **below**.

$$H(y, \hat{y}) = -\sum_{i} (yi \ log \ \hat{yi}) \tag{2}$$

A summary that included the accuracy of both sets was created and updated through the training, which can be visualized using Tensorboard.

5 Results

The network was trained for 30 epochs only, because it was enough to see how the accuracy got better with each epoch passed. It was best to stop the training after this number of epochs also, because the dataset is quite large, which makes the training take more time to improve the accuracy, and it wasn't possible to keep on training for more time, this was due to the fact that the training was done on the cloud and it's paid by hour.

After each epoch the accuracy of the model was saved using the summary collector, this way the accuracy can be checked using Tensorboard, which also gives the chance to extract the plot that is easily visualized **below**.

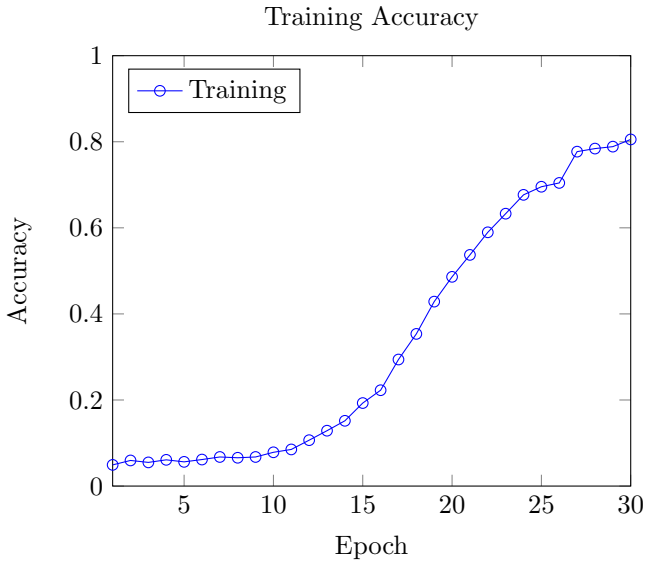

Figure 1: The plot shows how the training accuracy improves after each epoch.

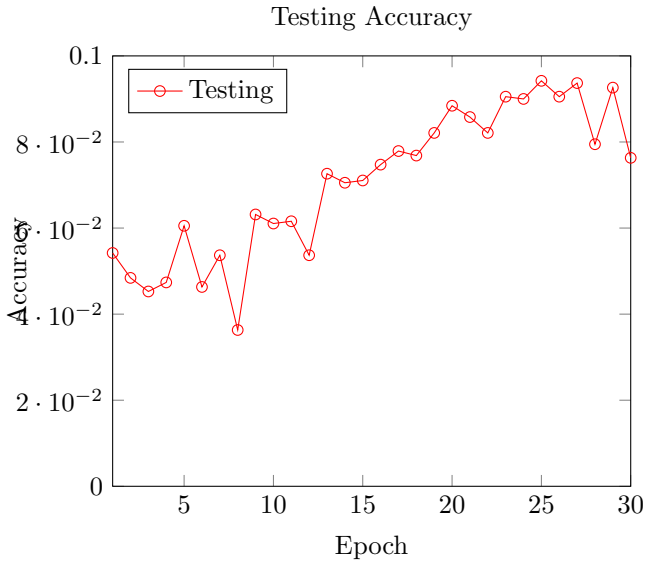

Figure 2: The plot shows how the testing accuracy improves after each epoch.

6 Implementation

The project has been implemented in Python using Tensorflow. Below, the list of Python files implemented for the project with a brief description of their behavior.

- **generate_tfrecords.py:** generating the tfrecord files used for the training and evaluation of the network.
- **train.py:** train the network specifying parameters and the dataset.
- **activities.py:** list of activities to be trained and the number of samples per activity to augment data.
- **c3d_model.py:** C3D model implementation.

Extra file:

- **pose_list.py:** shows the list of activities and the frequency of activities chosen to the training.

7 Conclusion