CS230

SwimMaster
CS230-Winter 2019

Erick Cardenas
ecasas@stanford.edu

Troy Ling
troyling@stanford.edu

Emilio Unda
eunda@stanford.edu

https://github.com/unda91/SwimMaster

Abstract

The goal of this work is to build an action detection system that can automatically recognize swimming styles. To achieve this, we investigate various deep learning algorithms; we focus on implementing the long-term recurrent convolutional network (LRCN) architecture, a combination of a convolutional neural network (CNN) and a recurrent neural network (RNN). We experiment with different long-short term memory (LSTM) models for the RNN module. Our best LRCN model consists of a pre-trained CNN and a 4-layer LSTM model as the recurrent module.

1 Introduction

Action recognition has many applications in the sports field. It can help create an automatic comment generation system to aid accessibility for the visually impaired. It could also be used for referee assistance, marking violations where a referee is not looking, or even to help athletes measure and track their performance. Based on this motivation, we have implemented a neural network to recognize one of the five different swimming styles that a person could be using: 'Butterfly', 'Backstroke', 'Breaststroke', 'Front crawl' and 'Start'.

More concretely, the input to our algorithm is a swimming video recorded with a smartphone. We then use a pre-trained CNN network, Inception V3, to extract features from each video frame which are later passed into a LSTM network. The LSTM includes a softmax layer that outputs the predicted swimming style in the video.

2 Related work

As the amount of data increases in the web, there has been an immediate need for automatic recognition of actions in videos. Many researchers have published related papers on different areas.

Sports Videos in the Wild 2015 (1) (SVW) has explored several approaches in the machine learning area. The authors presented different algorithms for the genre categorization problem, which are based on features extracted from dense trajectories, Bag of Words (BoW), and a motion-assisted context-based algorithm. These approaches have better performance (i.e., motion-based has 61.53% compared to ours 44.03%), but they come at the expense of computational cost and memory.

On deep learning, "Moments in Time Dataset report" (3) uses several neural networks based on different modalities (spatial: ResNet50-ImageNet, spatiotemporal: I3D and auditory: SoundNet) to predict the action event (e.g., swimming). These approaches have a robust prediction because they are based on spatial, temporal and auditory information. However, their model might have some challenges to scale given the level of complexity.

Finally, there were past class projects which examined similar problems, but they were about Tennis stroke recognition (4) and Tennis match prediction (5). Both of them also used LSTM models to predict the labels. However, we based our model on "Long-term Recurrent Convolutional Networks for Visual Recognition and Description." (6) This approach gave our model the capability of learning over a period of time instead of one frame at a time.

3 Dataset and Features

3.1 Dataset

We are basing our project on the Sports Videos in the Wild, 2015 (1) (SVW) dataset, which "is comprised of 4200 videos captured solely with smartphones by users of Coach's Eye smartphone app... by TechSmith corporation. SVW includes 30 categories of sports and 44 different actions. Due to imperfect practice of amateur players and unprofessional capturing by amateur users, SVW is very challenging for automated analysis." according to its website.

We are using a subset of this dataset which includes only swimming videos. The dataset uses 'Butterfly', 'Backstroke', 'Breaststroke', 'Front crawl' and 'Start' as labels and has 119 labeled examples in 244 swimming videos. The videos vary across the following properties: 1) height and width, 2) length (time), 3) camera angle, and 4) multiple people on the same video, all of which make the training problem more complex. Figure 1 shows a sample video from the original dataset.

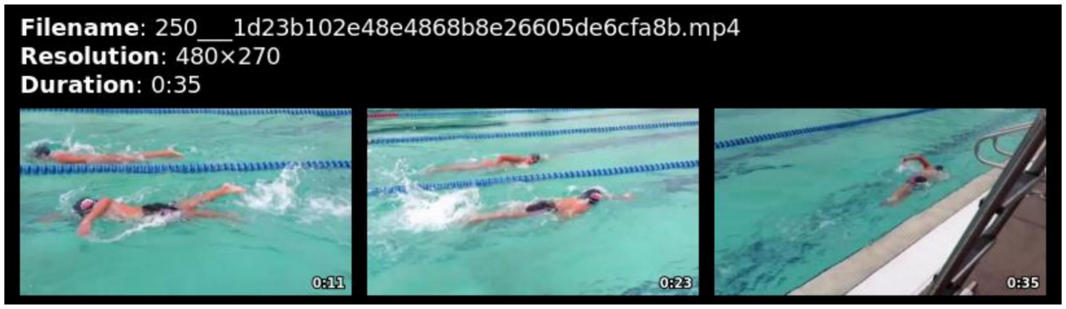

Figure 1: An example video from the SVW dataset, this 35 second long video captures a group of people using the 'Front crawl' style. As it can be seen from the images, the camera angle changes throughout the video, which represents an extra challenge for the classifier.

3.2 Preprocessing

The first step in the preprocessing pipeline is to normalize all of the frames into a standard size using a crop and re-size technique. First, we crop out the largest possible square from the center of the image and then we re-size each frame to the desired resolution. Cropping before re-sizing the video ensures that we do not accidentally distort the proportions of the people in the video. Although a resolution of 128 x 128 pixels is enough for a human to recognize the swimming style of a video, we opt for a resolution of 299 x 299 pixels since that is the input size that the Inception v3 (2) model accepts.

Afterwards we split the videos into one second long fragments. Since all of the videos have 30 frames per second (fps) each training example has 30 frames. After this step we now have a dataset with more than 2,400 training samples of shape (30, 299, 299, 3) (n-frames, height, width, n-channels). For example the video in Figure 1 would be split into 35 different examples.

3.3 Train-Dev Split

Finally we split the training examples into a training set and a development set. We use the following constraints for this last step:

Percentage Split: 80% of the examples will go to the training set and 20% to the dev set.

Stratified Sampling: Both sets should have the same proportion of labels as the full dataset.

No Leaking: An original video can only appear in the training set or on the dev set but not on both. The original videos were split into smaller training examples. This prevents the algorithm from memorizing something in the background to predict the label (on the dev set), and thus avoids leakages from the training set into the dev set.

4 Methods

We implement a Long-Term Recurrent Convolutional Network (LRCN) model with Keras (8) and TensorFlow (7). The LRCN model, based on Donahue et al (6), combines a deep hierarchical visual feature extractor (such as a CNN), with a recurrent sequence model that can learn to understand the sequential data (such as LSTM). Our final LRCN model uses a pre-trained Inception V3 to encode each frame of the videos from the training set to a 2048 feature vector, which are then passed to a LSTM module.

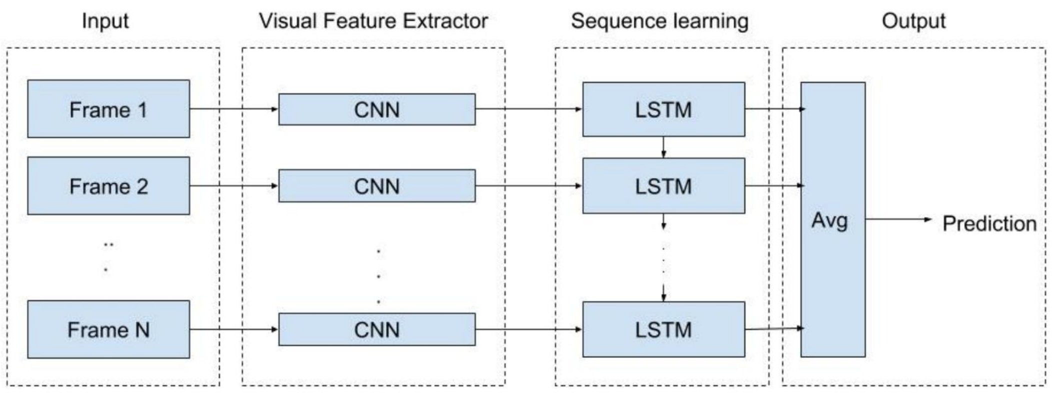

Figure 2: Our final LRCN model architecture

For this model, we use the categorical cross entropy as the loss function:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_j^{(i)} \log(P_j^{(i)})$$

where N is the number of training examples, C is the number of classes, y is the ground truth label, and P is the predicted probability of the label.

5 Experiments/Results/Discussion

Table 1 describes some of the different architectures that we experiment with. We will call our best performing model DeepSwimMaster (in reference to the number of LSTM layers it has). It manages to have the best balance between fitting to the training data, while still being able to generalize to the dev dataset.

During experimentation we realize that our model training spends the majority of the time in computing the outputs of the InceptionV3 model. Since we are not fine tuning its layers, we precompute the outputs for all our training data, and use them to train the recurrent part of our model in an independent way. This optimization improves our training velocity from 10 min per epoch to 2 seconds per epoch, and it allows us to increase our batch size from 8 to 1024.

We use a small learning rate (0.00005) which helps the model avoid local minima. Since we have an unbalanced balanced dataset, the learning algorithm when left to its own devices would learn to predict the majority label. Having a small learning rate gives the W matrix time to develop itself before the bias vector takes over control of the predictions.

Name	Visual Feature Extractor	Sequence Learning	Output	Dev Accuracy
Deep Swim-Master	InceptionV3	LSTMx4 (512,256,128,64)	Dense Softmax	44%
Wide Swim-Master	InceptionV3	LSTMx1 (1024)	Dense Softmax	34%
Convolutional Baseline	Conv (3x3x5), (5x5x10), (5x5x40)	Flatten	Dropout(0.4), Dense Softmax	28%

Table 1: Experiments and Observations

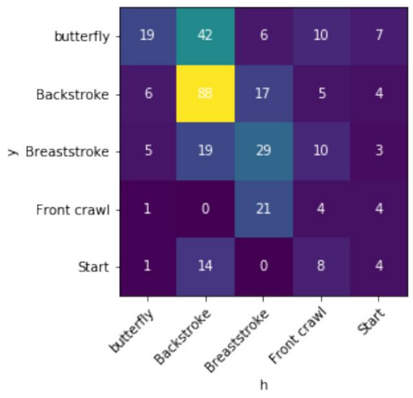

Figure 3: Confusion Matrix for DeepSwimMaster

In Figure 3, we present the confusion matrix of the DeepSwimMaster model. It shows a dataset that is heavily unbalanced with the majority of labels and predictions under the ‘Backstroke’ class. This imbalance in our label distribution proved to be a significant challenge for the model.

6 Conclusion/Future Work

In conclusion, our final LRCN model, using a pre-trained Inception V3 model to extract video features that are later consumed by a 4-layer LSTM model, achieve approximately 44% accuracy. Throughout the project, we discover that the dataset we use is not large enough to build a robust action recognition classifier. The dataset is also imbalanced and skewed to a certain label (i.e. Backstroke), which leads our models to overfit the training data. In addition, our experiment indicates that using a deeper LSTM model results into a better performance, as the model is more capable of understanding the temporal features from video data.

Regarding the future work, we would like to

- **Build a more robust data preprocessing pipeline**. While we perform various feature normalization techniques over the video data, the processed video features still contain video sequences in which no swimming action can be observed. A more robust data preprocessing pipeline would need to be implemented to detect and remove these sequences, eventually reducing the noise across the whole dataset.
- **Build a model for multiple object action detection**. Some inputs had multiple actions in the same video which might mislabel the data. To factor this into our model, we could add multi-labels in the output.

7 Contributions

Erick Cardenas implemented the Convolutional Baseline model, setup and managed the AWS instances and tried different hyperparameters for our model.

Troy Ling implemented the LCRM model and tried different hyperparameters for our model. He also pioneered the InceptionV3 + LSTM architecture for transfer learning.

Emilio Unda implemented the data preprocessing, train-dev split and tried different hyperparameters for our model. He also helped set up the ssh conection and port forwarding to use jupyter notebooks from the AWS servers.

Sagar Honnungar provided mentorship and guidance through the whole project.

References

[1] Seyed Morteza Safdarnejad, Xiaoming Liu, Lalita Udpa, Brooks Andrus, John Wood, Dean Craven
Sports Videos in the Wild (SVW): A Video Dataset for Sports Analysis
Proc. International Conference on Automatic Face and Gesture Recognition (FG 2015), Ljubljana, Slovenia, May. 2015

[2] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna
Rethinking the Inception Architecture for Computer Vision
CoRR, 2015

[3] Monfort, Mathew, et al.
Moments in time dataset: one million videos for event understanding.
IEEE transactions on pattern analysis and machine intelligence (2019).

[4] Ohiremen Dibua, Vincent Hsu Yu Chow.
Tennis stroke recognition using deep neural networks.

[5] Mitchell Allen Dumovic, Trevor Noel Howarth: Monfort, Mathew, et al.
Tennis Match Predictions using Neural Nets.

[6] Donahue, Jeffrey, et al.
Long-term recurrent convolutional networks for visual recognition and description.
Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[7] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.
TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.
Software available from tensorflow.org.

[8] Chollet, François and others
Keras 2015,
https://keras.io

[9] Eric Jones and Travis Oliphant and Pearu Peterson and others
SciPy: Open source scientific tools for Python
2001, http://www.scipy.org/

[10] Bradski, G.
The OpenCV Library
2000, Dr. Dobb's Journal of Software Tools