Real-time Activity Recognition with Google Glass

Shraman Ray Chaudhuri
shraman@mit.edu

Abstract

The increased popularity of "life-logging" comes with new demand for accurate activity classification techniques across mutiple media. In this paper, we present the methods and results of an enhanced activity classification method using Convolutional Neural Networks (CNNs) on 30fps video input from Google Glass. We focused on two categories of activities: reading and exercising. Classifying video input with CNNs remains an open problem with many challenges, particularly maintaining high throughput without diminishing classification accuracy. We present a hybrid model composed of fast preprocessing techniques and inference with a fine-tuned AlexNet that achieves high precision despite these challenges. We also outline various failure modes when fine-tuning AlexNet on noisy videos with sparse labels, and provide a training scheme to circumvent these issues. Our methods achieve an overall classification accuracy of 68% (with many sequences achieving up to 99% accuracy) with real-time computability on large amounts of video footage.

1. Introduction

Life-logging is a process whereby individuals document personal history through an ensemble of video clips, images, music, and other media describing physical activities, facilitated by state-of-the-art wearable technologies such as Google Glass, Kudu VideoGlass, and Memoto. The trend in life-logging follows an increased popularity of activity sharing in social media—the automation of which is key to user satisfaction. At the other end of the spectrum, life-logging technologies are being leveraged as an aid for memory prosthesis by means of auto-segmentation [1]. Thus, the breadth and depth of high-performant classification methods for activity tracking based on video input is ever-increasing.

Convolutional Neural Networks (hereon referred to as CNNs) have proven quite effective for classification and comprehension of image content, attaining remarkable results on the problem of object recognition [2]. The success of CNNs on image classification, however, is not so easily adaptable to the video domain due to the increase in data volume, decrease in resolution, and non-trivial exploitation of time-domain correlation between image frames. Nonetheless, we were encouraged to explore the performance of these CNNs on video activity.

We decided to use Google Glass as our primary method of data collection, and this came with several challenges. The default frame rate of even lightweight technologies such as Google Glass hovers around 30fps, with high degrees of correlation between individual frames. This means that in a mere 15 minutes of video, we have 27,000 frames of raw data with little variation in the amount of information we can extract. Furthermore, the challenge was compounded by the fact that the deep CNN architecture used in our model takes considerably long periods of time to optimize the parameters. Our original multi-GPU environment in which we sought to train our model was taken offline, and we had to opt for commodity hardware with significantly lower compute capability.

After a brief survey of tasks most relevant in everyday lives of college students here at MIT, we found that reading and exercising were the two most commonplace yet challenging activities to classify. We chose to focus only on these two activities to mitigate the problem of not having any training data prior to our experiment, and the time constraint involved in this project. Training data was collected by our team members themselves, taking turns to record daily activities whenever time permitted. "Reading" was limited to text on laptops, phones, tablets, and books, and "exercising" was limited to working out on athletics equipment particular to gyms and fitness centers. These definitions and clear-cut boundaries were necessary in removing the semantic overlap between broad categories of activities (i.e. walking through a hallway could be constituted by "reading" signs and "exercising" legs).

Finally, to transform the problem of video classification into one of image classification required careful preprocessing so as to maintain the fidelity of the reduced data set to the original videos, as well as remove the correlation between samples (frames). Because we opted for CNNs rather than hard-to-train RNN models, we were able to experiment with several machine learning techniques, such as SVMs and CRFs, to preprocess our data before training the net.

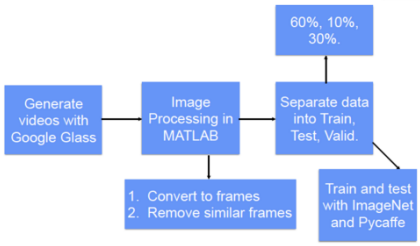

Figure 1. Overview of experiment design.

2. Related Works

There has been extensive research in applying scene and object recognition techniques to classify still-images [7,8], and much of this knowledge can be applied to video classification after reducing the problem space. A setback with this approach, however, is the lack of an open database of video data like the popular ImageNet database. Many strides have been made to specifically architect neural nets to adapt to the shortcomings of less diversified datasets. These neural nets employ advanced and effective techniques to extract as much information from these correlated frames as possible. For example, we have RNNs (Recursive Neural Nets) and LSTM (Long Short Term Memory) architectures [9] which are able to use classification results from prior runs in analysis of future runs (i.e. "memory"). Other techniques have stuck to CNNs but have augmented them with a variety of substructures analogous to video processing in the human brain, e.g. multiresolution foveated imaging across multiple frames of reference [10].

However, we take a different (arguably less advanced) approach than the ones mentioned above, to investigate whether a pre-existing CNN architecture can perform well in classifying videos with representative image frames, by front-loading the work on the preprocessing step to extract these key frames. We have yet to come across a study that takes our particular approach in video classification, in context of simple, high-throughput activity tracking.

3. Approach

An overview of the design of our experiment is shown in Fig. 1. Here I will expand on each section separately, and the specific methods used in each step.

3.1. Data Collection

Each member was initially responsible for collecting 1-2 hours of video consisting of reading from laptops and/or books, exercising at dormitory gyms or Z-Center, and a variety of other activities to be classified under "neither." We increased the diversity in our dataset by varying the locations where the videos were taken, covering as much of the campus as possible. We all lived in different parts of campus (some of us, off-campus) and our day-to-day activities were quite orthogonal in both location and content (everything from biking down Mass Ave. to roaming the halls of Building 46).

3.2. Image Preprocessing

Frames were extracted from the video via a simple MATLAB script, which then automatically resized the images to 256x256, reduced to grayscale, and marshalled the frames into different categories based on the activities they represented. Each member labeled his/her own portion of the data set and removed clips that would be troublesome to classify (e.g. when hair fell in front of the camera or dimmed lighting created too low contrast to adequately distinguish objects). We (randomly) designated 60% of the data for training (approx. 200,000 frames), 30% for validation (approx. 100,000 frames), and 10% for testing (approx. 31,000 frames). Fig. 2 shows a few of the raw frames comprising our data set, and Fig. 3 shows the labels assigned to a sample video sequence.

After manual filtering of the frames, much of the raw image data was still highly correlated and running the raw frames through the CNN would not only lead to wasting valuable compute time, but also run the risk of unevenly fitting the data to a non-uniform distribution of unique samples (e.g. walking in a hallway for 5 seconds as opposed to reading a book for 20 minutes). Furthermore, there was no natural segmentation of video components into individual sequences and transitions between clips of one activity versus another activity were often defined by extremely blurry frames in between. These frames could not be labeled and we needed a quantifiable means by which we could filter out bad frames. Instead of toying with regularization parameters and the objective function of the neural net gradient descent algorithm, we chose a more intuitive, cleaner, and compute-friendly method of *preprocessing* the data before feeding into the neural net, as outlined below.

3.2.1 Step 1: Detecting Anomalous Frames

To quickly filter out bad frames, we used a hybrid technique of performing maximum-margin separation for anomalous frames, after decomposing the raw pixel values into low-pass and band-pass filtered coefficients in the wavelet domain using the "discrete wavelet transform." This technique was performed on video segments of about 30 seconds long.

The discrete wavelet transform can be used to transform any signal into a reduced set of coefficients in which we capture information in both spatial and frequency domains by correlating it with a local-support "wavelet" function dilated across several scales [5]. This is actually a key pro-

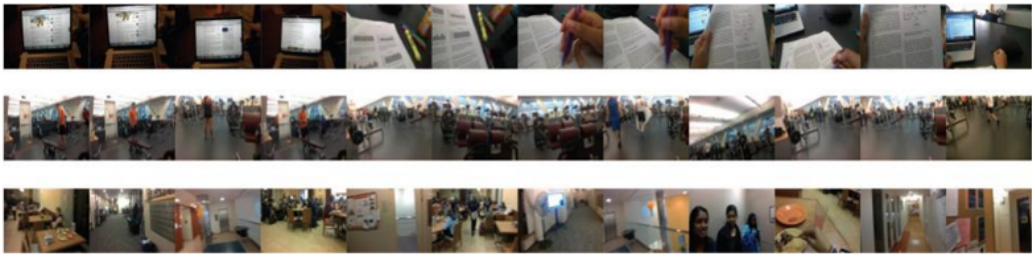

Figure 2. Raw frames captured during data collection

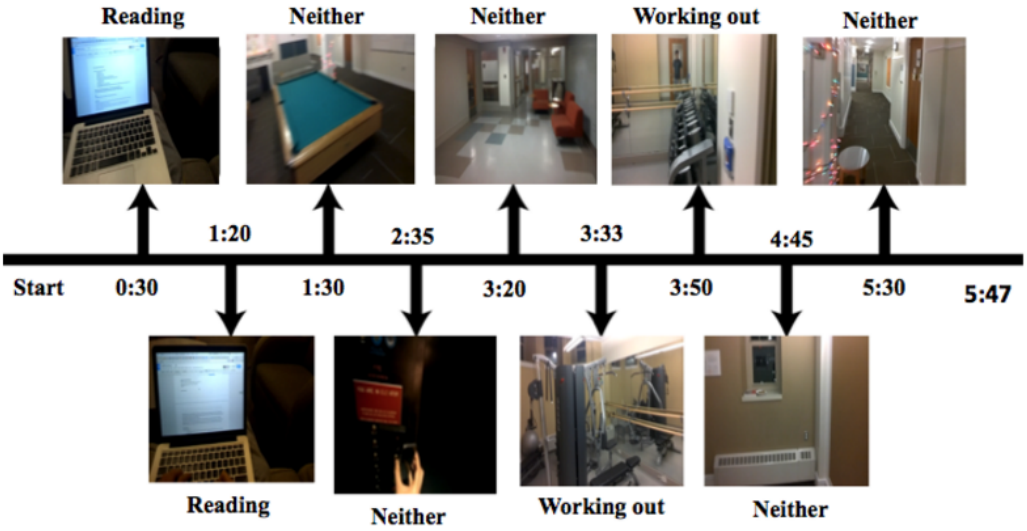

Figure 3. Manually labeled test frames

cessing step used in the widely popular JPEG2000 format [3]. We used the canonical "Fast Wavelet Transform," an $O(n)$ algorithm to correlate a signal with a representative wavelet function which captures the polynomial nature of the signal.[1] This works with images as well, but specifically requires a *bi*-orthogonal wavelet to capture correlation in two dimensions. We used the popular PyWavelets library [4] to perform the wavelet decomposition.

The wavelet coefficients were very important in determining bad frames since blurry images have more prominent low-frequency components (cf. low-pass Gaussian filtering used to blur images) while good frames were more evenly distributed in their frequency components. Therefore, after performing the wavelet decomposition, we used the lowest and highest frequency coefficients as feature vectors for our maximum-margin separator.

A *one-class* SVM was used since our primary goal was not binary classification, but rather detection of a small subset of bad frames. With feature vectors that best captured the disparity between good and bad frames, we were able to consistently and accurately prune out bad frames using an RBF kernel to create a hyper-sphere around the good frames and eliminate all images which fell beyond the boundaries of the hyper-sphere. Python's `scikit-learn` library helped us develop this end-to-end filtering method within only a few dozen lines of code [6]. All in all, this preprocessing step proved to be remarkably efficient and accurately reduced our dataset to the best representative set of frames for our CNN.

3.2.2 Step 2: K-Medoids Filtering

In addition to the dataset reduction above, we experimented with K-medoids clustering on larger segments of the video

[1] I leave out details about the wavelet math and implementation of algorithms in this report since they comprise a whole field in and of themselves, and are outside the scope of our project

Parameter	Value
Momentum	0.9
Max iterations	50,000
γ-value	0.1
Test iteration intervals	1,000

Table 1. Parameters used while training AlexNet

Activity	Error Rate
Reading	1.79%
Neither	5.42%
Exercising	85.7%

Table 2. Results of classification on test data

(5-10 minutes) to see if we could categorize the frames into K specific activities and take only the best representative frames (the "medoids") from each cluster. The goal of this technique was to reduce the number of identical frames (see similar images in Fig. 2)—however the technique was a little too effective for our needs. It would have been more impactful given large amounts of diverse video footage, whereas in our case, we were faced with a dearth of diverse footage. As such, our algorithm picked several frames from the edge of each cluster instead of the medoid alone, to compensate for this lack of data.

3.3. CNN Training and Validation

We chose the AlexNet architecture for our CNN after careful analysis of several alternatives (including MatConvNet and VGG). As delineated in [2], AlexNet is an optimized deep network for recognizing *objects*, which is the single-most important heuristic we can use for classifying activities such as reading and exercising. Furthermore, AlexNet was designed for GPU optimization (similar to our environment), ripe with fast ReLU and normalization units for better classification performance. A copy of AlexNet's diagram in [2] is provided in Fig. 3 for reference—we translated this design to a `prototxt` file to run in our environment with Pycaffe. The parameters of the network are outlined in Table 1.

Between iterations of modifying our preprocessing algorithm and our CNN parameters, we trained on the 60% of training data and ascertained a degree of generalizability with our validation set. We used our testing set only when we recorded new footage to augment our data set with fresh footage.

4. Experimental Results and Analysis

The CNN achieved 99% accuracy on the reading classification task with the first round of testing—however, this should be qualified with the fact that we are training on similar bits of video footage due to a lack of diversity in our data set. Although our testing set is significantly different in some ways to our training set, we still have overlapping frames and this gave the network an advantage in classifying frames which were part of the same video sequence.

To mitigate this problem, we performed a second round of testing with newly recorded test videos while keeping our training set stagnant. The results for these additional tests are recorded in Table 2. The CNN still performed well while classifying the "reading" and "neither" categories since there was quite a bit of diversity in the footage used in the training set. However, the error rate for exercising was shockingly high at 1271 out of 1400 frames misclassified. The high error rate is most likely due to the lack of diversity in the training set for exercising, where we only had a few gyms and not many unique angles to choose from. The network, therefore, suffered from inability to generalize for exercising in different physical locations (e.g. the Z-Center versus the BC gym), and we weren't able to extract the features to most accurately classify "exercise" in these gyms. However, the easiest fix for this would be to record more diverse footage for exercise, capturing different physical activities and a wider variety of gyms outside of MIT.

5. Conclusions

Overall, our results were higher than expected in some cases, and lower in others, but generally aligned well with our hypotheses going into the experiment. We found the the accuracy of network classification for activities, which largely depends on object recognition, works best with diversity in our training set. Reading was easily diversified by capturing different media, different angles and different content type, so that the underlying nature of "textual" content was extracted quite well by the CNN (leading to remarkably low error rates). However, the case for exercise was not as glamorous since the training set was very much homogenous—too homogenous to achieve correct results even with our pipeline of preprocessing and the use of AlexNet.

We've highlighted some of the key challenges in video classification in this study, successfully addressing some of them and emphasizing the difficulty in others. Highly correlated video data leads to a bloated training set which doesn't provide for as rich context as image databases such as ImageNet. Furthermore, video processing is crucial in allaying difficulties in training an image-centric CNN to perform the more complex task of video classification. At the same time, the results attained demonstrate the true potential in CNNs as a means for advanced tasks such as life-logging. Furthermore, having achieved the high throughput that we did in the end-to-end classification pipeline, we see the po-

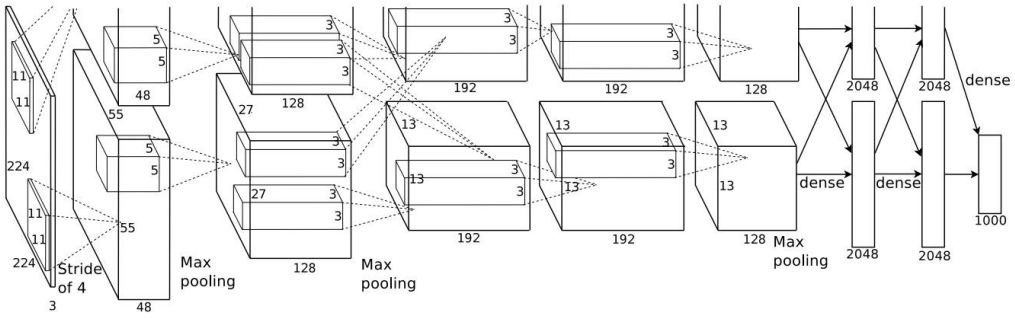

Figure 4. AlexNet architecture [2] as replicated for our CNN model of choice

tential to integrate this system into an ad-hoc environment where users can upload short bits of video from lightweight technologies such as Google Glass and track their activities on-the-go.

References

[1] S. Hodges, E. Berry, K. Wood SenseCam: A wearable camera which stimulates and rehabilitates autobiographical memory *Memory*, 2011.

[2] A. Krizhevsky. I. Sutskever, G. E. Hinton ImageNet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems*, 2012.

[3] M. Marcellin. M. Gormish, A. Bilgin, M. Boliek An Overview of JPEG-2000 *IEEE Data Compression Conference*, 2000.

[4] F. Wasilewski PyWavelets: Analysis and Classification of Medical Signals using Wavelet Transforms *Warsaw School*, 2006.

[5] P. Schroder, W. Sweldens Build Your Own Wavelets At Home *SIGGRAPH*, 1996.

[6] Scipy One-class SVM with non-linear kernel (RBF), 2010.

[7] L. Li, H. Su, Y. Lim, L. Fei-Fei Objects as attributes for scene classification *European Conference for Computer Vision*, 2010.

[8] L. Li, L. Fei-Fei What, where and who? Classifying events by scene and object recognition *International Conference for Computer Vision*, 2007.

[9] J. Y. Ng, M. Hausknecht, S. Viyanarasimhan, O. Vinyals, R. Monga, G. Toderici Beyond Short Snippets: Deep Networks for Video Classification 2015.

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei Large-scale video classification with convolutional neural networks *Computer Vision Proceedings Report*, 2014.