Video-based Event Recognition

Ian Ballard and Lane McIntosh
CS 221 Artificial Intelligence Project Final Report
Stanford, CA 94305.

As the availability of video data rapidly increases, there is increased need for automatic methods of extracting the events or highlights from the video material. We focus on the space of surveillance camera footage, and develop systems that automatically detect moving humans, cars, and trucks. Human motion includes loading or unloading objects from a vehicle, walking or running, and getting into or out of a vehicle. Although this is an easy problem for the human visual system, noise from camera instability and wind make this a difficult problem in artificial intelligence. We investigate the binary classification performance of support vector machines and deep convolutional neural networks.

Keywords: natural movies, event recognition, sensory processing, computer vision

INTRODUCTION

The human visual system possess a remarkable ability to extract important environmental features quickly and efficiently. Recent evidence indicates that this speed and flexibility depends on visual processing at all levels of an extensive hierarchy, beginning with the retina. Although the retina is known to implement some simple filters, recent research suggests that substantial visual processing occurs at the level of the retina. In particular, the retina is thought to implement a predictive model of natural scenes which allows it to compute a compressed signal representing only surprising deviations from the statistics of natural scenes. Indeed, information from 225 million rods and cones is condensed into about a million projections into the brain through only 2 layers of neural units.

To make progress in understanding the computations implemented by the retina, one useful approach could be to develop models that can implement predictive models of natural scenes, and then re-examine neural data in relation to the properties of these models. We will therefore try to develop models which can detect when a new object has entered a natural scene, while ignoring small changes due to wind, camera instability, and other unknown factors.

FIG. 1: Example frames of people unloading objects from their vehicles. This is an example action where having information from previous and future scenes can provide a large advantage to a classifier tasked with detecting events.

BACKGROUND AND LITERATURE

There has been a great deal of work assessing videos data for interesting features. These approaches generally capitalize on strengths of the computer vision research. For instance, some techniques first perform object identification, and then track those objects [1, 2]. Other approaches extract features in the temporal domain and attempt to hierarchically condense a movie into sub sequences [3, 4], or integrate audio information [5]. Our approach differs from these because we are attempting to roughly constrain the modeling to known properties of the retina.

In particular, object recognition and multisensory integration occur in the brain, and retinal neurons have very limited capacity to integrate information across time. Thus, we do not expect our algorithm to approach any benchmarks in the event detection field, but merely acquiring above-chance performance could provide interesting insights that motivate experimental investigations.

FIG. 2: Example frame from the original VIRAT dataset before any downsampling or pre-processing.

EVALUATION

Data

We will train and test our models on the labeled VIRAT video dataset, a large-scale (over 40 GB) database of fixed surveillance camera videos with annotations [6]. The dataset contains hours of 1920 by 1080 resolution

video at 30 frames per second. The video content is primarily composed of long epochs where nothing happens (with exception of changes due to wind or camera imperfections) interspersed by short episodes of cars or people entering the scene and moving around. The annotations were performed by humans on Amazon Mechanical Turk, and include frames of when objects appear, as well as the object and action classification.

We will simplify the problem by decreasing the video resolution, reducing the framerate, and converting the RGB frames to grayscale. We also simplify the annotations, considering only the binary presence of an event, rather than the full classification of the event or action. A sample frame is shown below.

Event Examples

The movie annotations in the VIRAT database identify the start and end frames of 12 different events: 1: Person loading an Object to a Vehicle, 2: Person Unloading an Object from a Car/Vehicle, 3: Person Opening a Vehicle/Car Trunk, 4: Person Closing a Vehicle/Car Trunk, 5: Person getting into a Vehicle, 6: Person getting out of a Vehicle, 7: Person gesturing, 8: Person digging, 9: Person carrying an object, 10: Person running, 11: Person entering a facility, and 12: Person exiting a facility.

Since we aim to identify when a new object has entered the scene, we declare that an event has happened when event types 2, 6, 11, or 12 occur. Our labels are 1 for every frame that the event is taking place.

Re-labeling

Unfortunately, the labels were 1's only at the first frame of an event, and perhaps more catastrophically, did not include vehicles entering and exiting the scene. To overcome these shortfalls, we hand-labeled the videos so that every frame with human or vehicular motion was labeled with 1 and everyother frame was labeled with 0. In addition to improving the quality of the classification problem, this also reduced the skew of our data, from the original annotations where 99.7% of labels are 0 to our new annotations where now 64% of labels are 1.

Oracle

The ideal performance of our system would be to detect events with the reliability of human observers. Since we hand-labeled the videos, we treat the labels that we generated as ground truth. Since there is likely some human error, we both hand-labeled partially overlapping subsets of the videos and compared the fraction of our labels that agreed to establish the oracle performance level. We found that 86.9% of our labels agreed, and

FIG. 3: An illustration of the shortfalls present in the original annotations from Mechanical Turk. In both frames, the red boxes highlight human and vehicle motion, however only in one of these red boxes (top frame, unloading a car) do the annotations indicate an event is taking place. The red boxes in the bottom frame, corresponding to a person walking across a parking lot, a car driving into the parking lot, and a person crossing the street, are all indicated as "no event."

that this accuracy was fairly consistent across the different videos (standard deviation of 3% accuracy across videos). This oracle upper bound on performance gives an idea of the degree of human error that we would not expect our model to correctly classify.

Baselines

Thresholding large frame-to-frame differences

A first pass attempt at event recognition might simply look at the difference in pixel intensities between frames, the intuition being that large differences in pixel intensities implies that something different occurred in the second frame. However, even in frames where nothing in the scene changes (no new objects, and no moving objects), around 50% of the pixels are different. The majority of this difference comes from slight imperfections in the stability of the camera, either with respect to the precision of its measurements or due to barely perceptible movements of the camera due to wind.

We found the best baseline performance by varying two thresholds - 1) the smallest intensity difference between frames we care about, and 2) the smallest num-

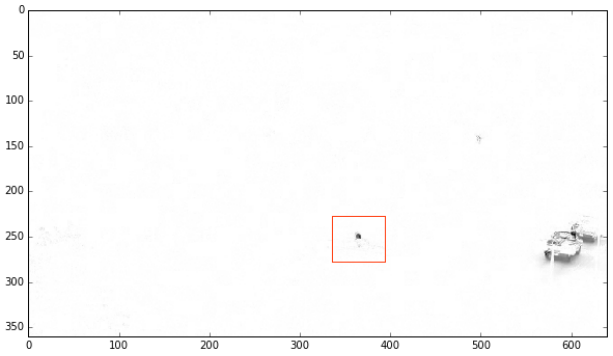

FIG. 4: Example event after taking differences between frames. This was classified in the dataset's annotations as type 6, person getting out of a vehicle. In addition to the person getting out of a vehicle (red box), a truck can be seen driving by in the lower right corner. This latter event is an example of motion that was unlabeled in the original dataset but labeled in our new ground-truth annotations. Additionally, this is one of the more stable surveillance cameras, where moving objects can be readily detected from the frame derivatives.

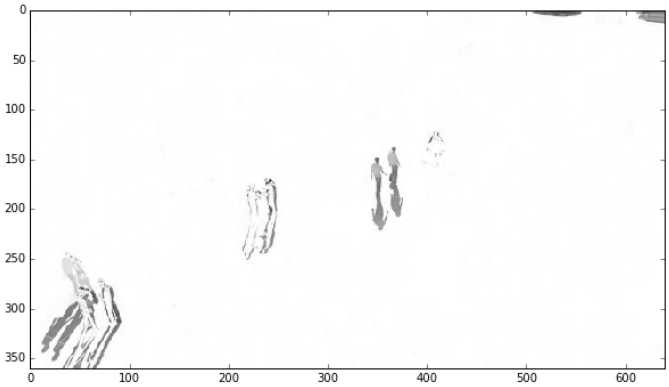

FIG. 5: Example event after taking differences between frames. This was classified in the dataset's annotations as type 2, person unloading an object from a vehicle.

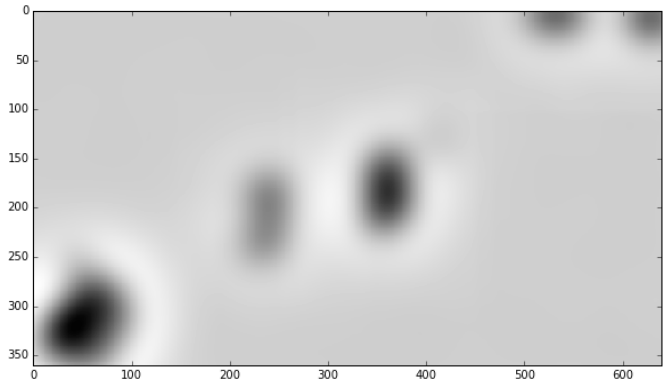

FIG. 6: The same frame difference from the previous figure, except filtered with a large spatial-scale center-surround 2d filter.

ber of different pixels according to the first threshold that we require before declaring that a new event has occurred. Mathematically, this baseline was

$$\text{prediction}_t = 1\left[\sum_{i,j} 1[|p_t^{(i,j)} - p_{t-1}^{(i,j)}| > \alpha] > \beta\right], \quad (1)$$

where α is the smallest pixel difference we count, β is the smallest number of different pixels we consider, and $p_t^{(i,j)}$ is the pixel intensity at position (i, j) in frame t. The best accuracy using this baseline was obtained by setting a low threshold that detected 100% of events but had an accuracy of only 64%.

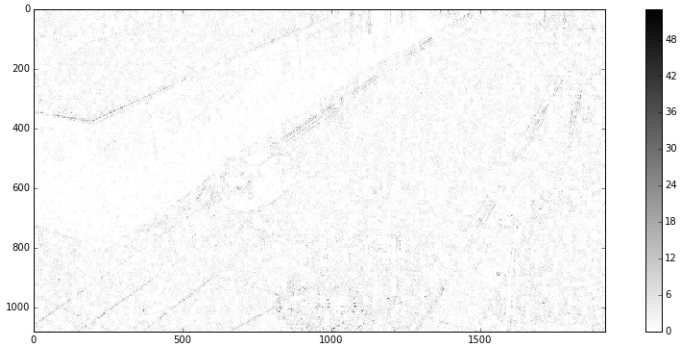

FIG. 7: Example absolute difference between two frames when there is no new event. Color bar is the value of the pixel difference (each pixel has a 0 to 255 grayscale intensity). Even though the scene is unchanged to the human observer, every non-white spot on the image denotes a pixel that changed due to wind, slight variations in sunlight, and camera fidelity and stability. In this example, 55% of the pixels are different.

All 1's or 0's baseline

In the case where a dataset is skewed to have more than 50% of one particular label, it is possible to perform significantly above chance simply by always guessing one particular label. While guessing 0's would have yielded a very high accuracy of 99.6% in our original dataset that only labeled 1's for the beginning frame of an event and didn't consider every moving object to be an event, after re-labeling the dataset the best performance from guessing only a single label would be 62% for guessing all 1's.

PREPROCESSING

Features

We have spent the bulk of our time so far tuning the preprocessing. First, we needed to implement data reduction. We reduced the frame rate to 1 fps, as well as downsampled the video resolution to 100 pixels in x.

One of the primary objectives of the early retina is thought to be extracting sparse features via zero-mean center-surround receptive fields [7]. We adopt this viewpoint and preprocess our data by spatial filtering the movies with center-surround filters. We parameterize the center-surround filters as difference of Gaussians, and create a bank of filters with varying surround sizes.

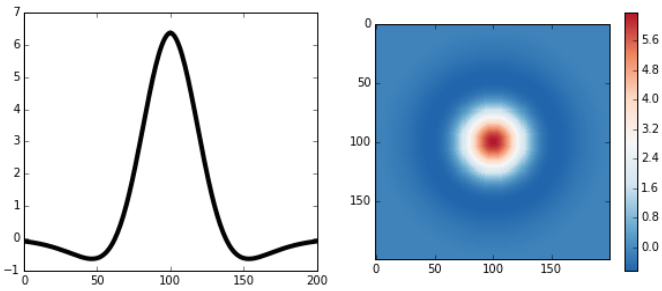

FIG. 8: Left: A 1-d slice of a center-surround spatial filter. Right: The full 2-d heatmap of an example center-surround spatial filter, very similar to the spatial receptive field of a retinal ganglion cell.

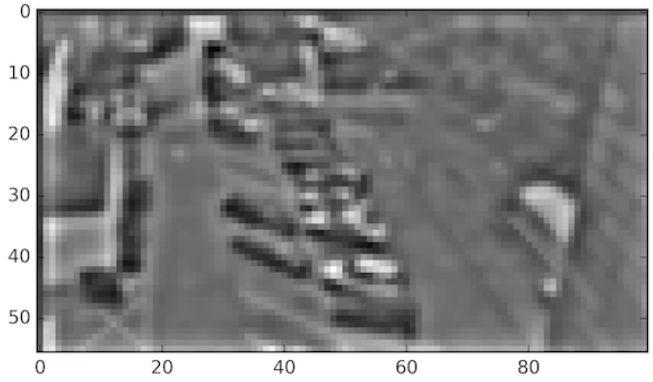

FIG. 9: An example of a filtered image

Next, since we care about the difference between successive frames, we transformed each image into a difference image between subsequent frames. We observed that as a result of the filtering described above, the resulting images and difference images were very sparse. We decided to leverage this by setting values close to 0 to 0 and using sparse features to optimize our classifier performance. We flattened the difference images into a 1D array, z-score normalized, and then set all pixels within 1/2 of a standard deviation of the mean to 0. This ensures that features are very sparse, and appears to result in very low loss of information due to our spatial filtering. Finally, we concatenate the frame difference vectors across the videos to make a single training set.

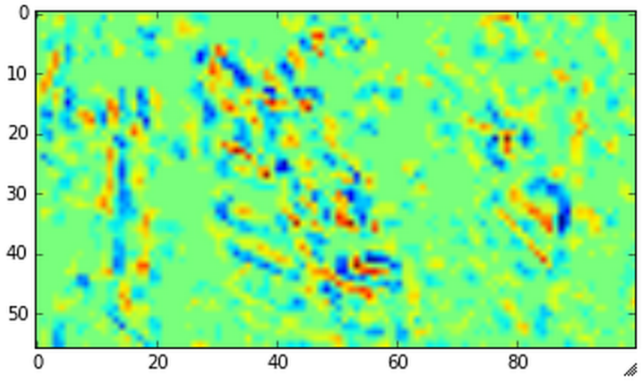

FIG. 10: An example feature image after all preprocessing.

MODELS

Support Vector Machine

We train a support vector machine using the `scikitlearn` library. We began by using a simple linear kernel. We observed 98% accuracy. However, these findings were suspicious because, across our $>$ 6000 frame differences (at 1.0 frames/second), we only have 16 labelled events. Examining confusion matrices revealed that the algorithm was making many false alarms.

Our approach was to hand-label images. For example, if a person enters the scene walking, we labelled every frame as positive while the person is in frame. The original automatic labelling only takes into account boundary conditions.

After hand labelling, we initially observed a marked decrease in classification accuracy. Closer inspection revealed that this was due to overfitting of the training data where would get near perfect (98%) training accuracy and around 54% training accuracy. This indicated a high variance problem and we took 3 main approaches to reducing the variance: increasing the regularization, limiting the number of positive weights by increasing the threshold for setting features close to zero to zero from .5 to 1 or 1.5 standard deviations from the mean.

All three of these approaches produced a strikingly similar pattern. The algorithm would produce nearly identical results until suddenly it would switch behavior a begin drastically underfitting, by guessing that nearly every frame difference is an event. This produced around 60% training accuracy and 62% test accuracy (note that 62% of our examples were positive). For all three approaches, the shift was abrupt, and we could not obtain intermediate behavior by changing the parameters.

In order to try to counteract the high bias introduced by these techniques, we also tried using a radial basis function kernel, reasoning that the larger hypoth-

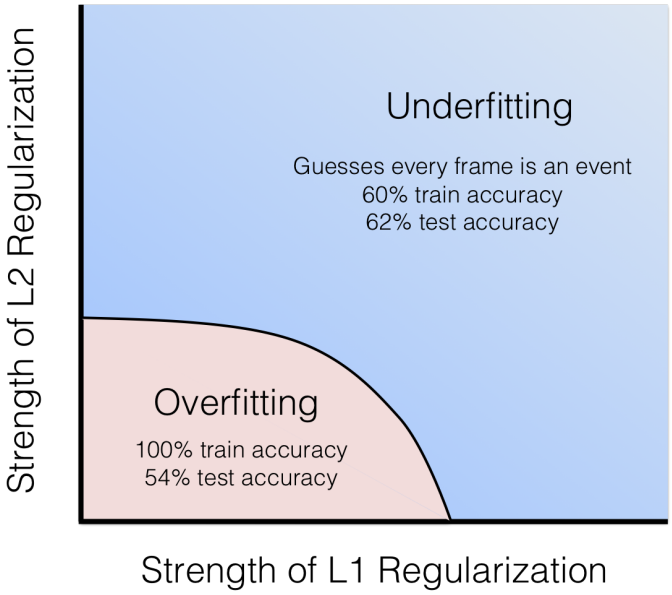

FIG. 11: Before augmenting our SVM feature templates, we experienced a phase transition between overfitting and underfitting.

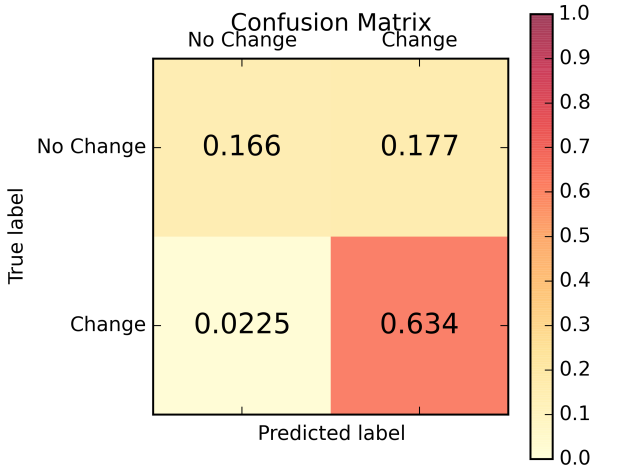

FIG. 12: The training error for the radial basis function kernel SVM.

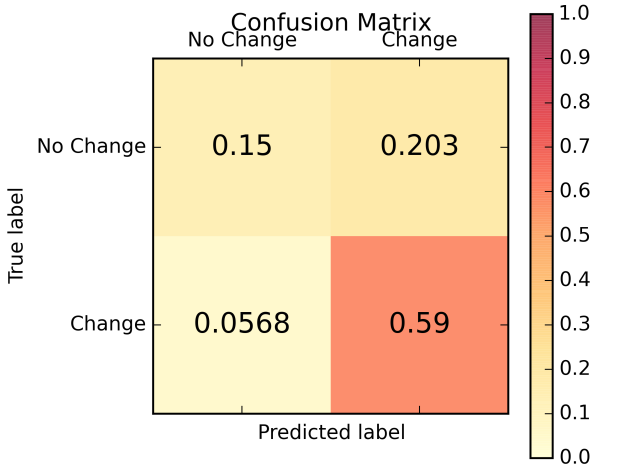

FIG. 13: The test error for the radial basis function kernel SVM. Note the overall bias to predict Change.

esis class of nonlinear features may reduce underfitting. However, this merely caused a return to the high variance problem (overfitting of training data).

We next turned to our pre-processing regime. We were at this point using a relatively wide spatial filter that blurred out most of the details of the image. We chose this initially because we reasoned this would help reduce effects related to small deviations due to wind. By reducing the size of this filter, we were able to achieve 74% test accuracy with a radial basis function kernel.

We were unable to get good performance for a linear kernel. We speculate that this is because movement generally involves deviations in adjacent pixels. Features that pool over patches of adjacent pixels may more robustly detect deviations due to genuine movement.

Examination of the confusion matrices revealed an overall bias in our classifier to predict Change. Indeed, it predicts change 80% of the time. This could be because the training set is unbalanced, and so it would be beneficial overall to be biased towards predicting change. We re-ran the above analysis and randomly eliminate examples so that the proportion was equal. Accuracy on the test set was reduced to 65%. Interestingly, this did not eliminate bias: the classifier still disproportionately predicted change (78% of the time).

This bias is perhaps unsurprising, given that the movies contain many frames where there is movement during wind and camera instability. A cursory analysis of the false positives indicate that this is likely the cause of the problem. These problems could be addressed in a variety of ways. One potential solution would be to create features that took into account several successive frames. Genuinely moving objects would cause pixels to change over a relatively larger span of the image than a tree blowing in the wind. Nonlinear features could potentially detect this type of motion. Another possibility would be to create artificial negative data where the frames are shifted by several pixels in different directions. This may provide enough information for the classifier to detect that whole image shifts are not genuine movement.

Convolutional Neural Network

We implemented a convolutional network in Berkeley Vision and Learning Center's Caffe based on the winning 2012 ImageNet Challenge architecture of Krizhevsky,

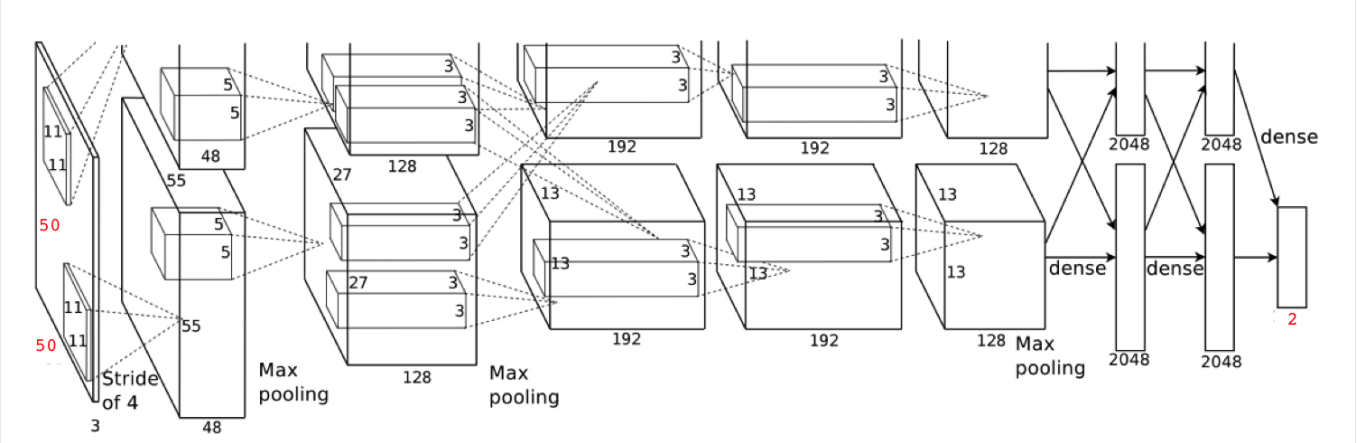

FIG. 14: The architecture of our convolutional neural network, adapted from Krizhevsky et al., 2012 [8]. Red font denotes parameters that were changed from the original architecture.

et al. [8, 9]. This network consists of 8 layers, 650,000 neurons, and 60 million parameters, and training it for 50,000 iterations on Stanford Rye servers' Nvidia Tesla GPUs took around 10 hours. Some modifications of the network from the published network included reducing the number of iterations from 145,000 to 50,000, changing the data layer to accommodate 56x100 grayscale pre-processed movie frames instead of 256x256 RGB ImageNet images, and changing the output from a 1000-way softmax to a simple binary classifier. Pre-processing was the same as for the SVM, except there was no need to vectorize the pre-processed images since the convolutional neural network requires nearby pixels to remain nearby.

Although we expected convolutional neural networks to significantly outperform SVMs, since most of the current image classification benchmarks are held by convolutional neural networks [8, 10], the network had an accuracy of 74.48% on held-out test data.

Perhaps due to the relative parcity of data we used (approximately 6,000 frames versus 1.2 million images in the ImageNet database), our network was strongly overfitting, and had a training error of 94% compared to the test error of 74%.

To compensate for this overfitting, we should regularize our network by either 1) reducing the model complexity, in particular the number of layers, 2) increasing the fraction of neurons chosen for random dropout [11], or 3) increasing the amount of labeled data. In general, this was a good exercise to see that although convolutional neural networks are currently very popular, they can be difficult and slow to train, and out-of-the-box results might not surpass the accuracy of simpler models like support vector machines.

To continue work on this convolutional neural network, one significant implementation hurdle would be to either crop the movie frames to be square and re-label the dataset, or to implement the network in Theano, Torch7, or pylearn2 instead of Caffe. As it stands, Caffe only supports square inputs, and so our 56x100 frames were cropped to be 50x50. While this may be acceptable for most image classification problems where the object of interest is in the center, for several of our videos the most motion stemmed from roads at the boundaries of the movie. Given that many of the frames may not contain any motion after cropping, it is surprising that the convolutional neural network can still obtain performance on par with the SVM.

CONCLUSIONS

We found that both support vector machines and convolutional neural networks performed similarly after downsampling the spatiotemporal resolution of movie frames, taking temporal differences, and convolving the difference frames with tuned edge-detecting filters. This perforamce of around 74% represented an increase of about 10% from the baseline, or about half of the difference between our baseline and oracle performance.

We noticed large performance gains using smaller center-surround filters, suggesting fine-grained structure may be important for distinguishing between events and non-events, for instance a human walking down the street versus a plant swaying in the wind. For future work, it would be interesting to see how skipping the downsampling step might increase performance on the convolutional neural network. For the SVM, we predict that this would have less of an effect. Additionally, we found that using nonlinear kernels for the SVM was also essential for above-chance performance.

Overall, we found that re-labeling and pre-processing was critical for achieving above chance performance. We also found that while regularization controls a trade-off between overfitting and underfitting, it was necessary to increase the number of feature templates (filters) we

used to explore the space between these regimes.

We were excited to find that convolutional networks could perform as well as SVMs in this dataset. These networks are based off of neural network models that could conceivably apply to the retina. Going forward, it will be fascinating to see whether a more realistic retinal model (incorporating known anatomical patterns) could recapitulate some of these findings. Such a model would suggest that computation in the retina can be considerably more sophisticated than has been appreciated by the field, and would have important implications for retinal prosthetics.

We would like to thank Jiaji Hu for his advice and feedback and Ben Poole for tips on how to speed up aspects of OpenCV data handling.

[1] Wayne Wolf, Burak Ozer, and Tiehan Lv. Smart cameras as embedded systems. *Computer*, 35(9):48–53, 2002.

[2] Stephan Hengstler, Daniel Prashanth, Sufen Fong, and Hamid Aghajan. Mesheye: a hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 360–369. ACM, 2007.

[3] Lihi Zelnik-Manor and Michal Irani. Event-based analysis of video. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages II–123. IEEE, 2001.

[4] Dong Xu and Shih-Fu Chang. Video event recognition using kernel methods with multilevel temporal alignment. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(11):1985–1997, 2008.

[5] Marco Cristani, Manuele Bicego, and Vittorio Murino. Audio-visual event recognition in surveillance video sequences. *Multimedia, IEEE Transactions on*, 9(2):257–267, 2007.

[6] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3153–3160. IEEE, 2011.

[7] Robert W Rodieck. The vertebrate retina: Principles of structure and function. 1973.

[8] A Krizhevsky, I Sutskever, and GE Hinton. Imagenet classification with deep convolutional neural networks. nips 25, 2012.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[10] Rodrigo Benenson. Classification dataset results. `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`, 2014. [Online; accessed December 12, 2014].

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.