Exploration of Anomaly detection through CCTV Cameras: Computer Vision

Wilson Ler
Stanford University
Stanford, CA 94305
lws803@stanford.edu

Sean Decker
Stanford University
Stanford, CA 94305
skdecker@stanford.edu

Introduction

Although closed-circuit television (CCTV) are ubiquitous in the modern world, the footage from these cameras often goes unexamined; there is just too much footage to parse. We propose that the use of anomaly detection to filter CCTV footage in real time can be used to solve crimes that have no in-person witness or could be used to alert authorities that medical attention is needed in the case of an emergency. A model of this sort would allow moderators to not need to continuously scan one video stream, but instead only requires the moderator for interpretation once an anomaly occurs.

In this project, we develop a neural network which takes grey-scale images from CCTV cameras as inputs and outputs a binary prediction on if an anomaly is present. Furthermore, when our model finds an anomaly, it also outputs a salience test to explain what part of the image contains the anomaly.

Related Work

In this project, we take inspiration from Perera and Patel's paper *Learning Deep Features for One-Class Classification* (1), specifically drawing on their paper in help with the particulars of the architecture of our neural network for training a network for one-class classification, in which one only has training examples from the class in question. To train this network, Perera et al. use something that is called compactness loss in order to assess the compactness of the class being learned in the one-class classification task by measuring the compactness in feature space of the set of feature extracted from images of the class of interest. One innovation in the Perera et al. is the introduction a parallel neural network, meant to be trained in parallel to the one trained one examples from the class of interest. This separate network trains on a reference dataset or multi-class dataset, which contains examples of various other reference classes. Then this network optimizes according to descriptiveness loss, which measures how distinct the representations for images from different classes are. Although we didn't utilize this parallel network in this project, this would be a good addition for future work.

In order to explain the anomalies that our model encounters, we use a method similar to SHapley Additive exPlanations (SHAP), which is discussed in Lundberg, Scott M and Lee, Su-In's paper: "A Unified Approach to Interpreting Model Predictions."(3). Similar to their research, we use a sliding window of occlusion in order to calculate the importance of certain pixels in the image to the outputted value. Our calculated values for the importance of certain pixels are not SHAP values however, and we do not use any code or algorithms from Lundberg et al.'s paper.2

Dataset and Features

We built our machine learning algorithm on UC San Deigo's Statistical Visual Computing Lab's UCSD Anomaly Detection Dataset (2). This is a dataset of CCTV footage, acquired from a stationary camera mounted above a pedestrian walkway. The footage contains various crowd densities and twos

Preprint. Under review.

of people walking. Abnormal events in a one-class context can be defined as whatever examples are not present in the training dataset. In this project, we define abnormal frames as frames with non pedestrians on the walkways or pedestrians on the grass as opposed to a walkway. For example, some examples of non pedestrians include wheelchair users, bikers, skaters, and gold carts.

Figure 1: Example frames from the UCSD Anomaly Detection Dataset. The left frame is a normal example and the right frame is an anomalous frame because of the presence of a gold cart

Now for the particulars of the dataset. The data we worked with came split already into training and testing sets, with the training set including 34 video samples and the testing set containing 36 video samples. Each video sample is a series of 200 gray scale, no alpha channel tiff images with a resolution of 238 by 158. All videos in the training set do not contain an anomaly, while every testing set video contains an anomaly, but the anomaly is only present in the video for some subset of the frames. The testing set comes labelled with the exact frames the anomaly is in view.

Figure 2: An example of a training image that had two squares of error added to it, one of blur and one of noise

Training sets for one class learning come with only class-examples, and that is the case in this UCSD datset. Because this is the case, we produce anomalies randomly during the training phase by choose half of the training frames to have errors added to them. Specifically, we randomly add Gaussian noise and/or blur to with 1/2 probability to each of the training images. When we choose to add noise or blur, we choose to add one 50 pixel square of blur/noise with 1/2 probability, and two 50 pixel squares the rest of the time. For each blur/noise square added, we add Gaussian noise with 1/2 probability and we add Gaussian blur the rest of the time. An example of a training image that had two squares of error added to it, one of blur and one of noise is given in 2.

Methods

Logistic Regression

Because this is binary classification, a naive attempt to solve this problem would be to use logistic regression. This makes logistic regression a good baseline model to compare our own model to. We reproduce the equation for logistic regression below:

$$p(y = 1|x; \theta) = \frac{1}{1 + e^{-\theta^T x}}$$

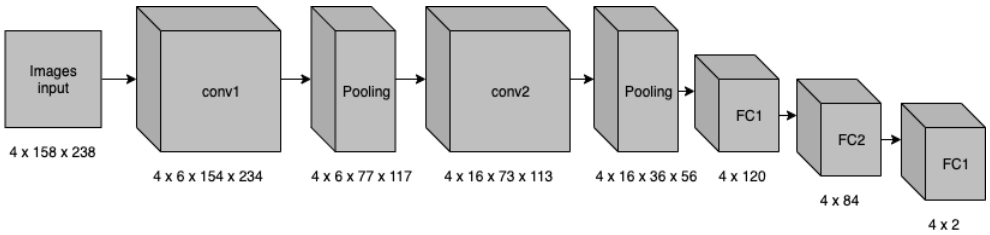

Figure 3: Our neural network's architecture.

According to logistic regression, given some input x, the probability that the label $y = 1$ for that input is given by the sigmoid of the dot product of x and some weights vector.

Neural Network

Logistic regression however has major problems with one-class classification as we discuss in the Experiments / Results and Discussion section. To address these shortcomings, the model that we focus on for this project is a neural network. Our neural network architecture is described by the figure in 3. This neural network architecture is inspired by (1). All that each layer in a neural network represents is the multiplication and addition of some learned weights and biases to a given input to create an output. The first two layers of weights are the convolutional layers, with the first being a simple 1 to 1 layer to preserve as many features as possible and the second layer convolving each 3 x 3 section of its input with a 3 x 3 polynomial. The final 3 layers of the neural network are all simply multiply their inputs by some matrix and add a bias vector:

$$f_{\text{linear}} = xA^T + b$$

What makes the neural network more powerful than the simple baseline logistic regression is the addition of non-linear operations between each of these layers. In our model we use a ReLu, which corresponds to the following simple formula:

$$f(x) = max(0, x)$$

. Starting with all the weights in each of its layers uninitialized, the neural network learns its appropriate weights through the use of backpropagation.

Saliency Test

Although neural network are extremely powerful for prediction, they do not explain how they came to their prediction. All this decision making is somewhat hidden away inside its hidden layers. In order to explain how our model came to its decision on a given image, we built a separate model which, using our neural network as a block box, is able to generate scores for pixels in the input image that represent how important the pixel is to the output prediction. These pixel scores can then be translated into a heat map which describe the most important features of the image. You can see an example heat map generated by this at 6.

The test we run can be thought of as a sliding window occlusion because in basic terms, we slide a window of pixels from a normal class example over the input image to occlude part of the image and see how it changes the output prediction. The test similar to, but not based on (3). More specifically the test works as follows:

1. Store output prediction score for target anomalous image as Y.
2. Sliding over the entire image with a 20 by 20 pixel window, take a non-anomalous image and crop the pixels from the that image into the the exact window dimensions and location to replace in the target image
3. Feed the resultant image into the model and obtain the score Y^*.
4. Calculate the difference between $Y^* - Y$.
5. If $Y^* - Y$ is negative, multiply the value by a red mask on output image to generate a heat patch. Larger $Y^* - Y$ difference implies a more distinct heat patch

Experiments/ Results and Discussion

A key characteristic of anomaly detection is that normal examples vastly outnumber anomalies. This means that having any non-zero false-positive rate, i.e. labelling any normal frames as anomalous corresponds to many mislabelled frames which even for small false-positive rate will outnumber the actual anomalies. With this in mind, we made it the goal of our models to have as high of a true positive rate while keeping the false negative rate at 0.

Another thing to keep in mind is that a given anomaly corresponds to many frames, not just a single frame of a given video. Because of this fact, our testing dataset inputs are not single frames, but 200 frame videos, where one anomaly is present in each video but only a fraction of the total frames. Thus in this context, we define a true positive as a model identifying any frames for a given anomaly as anomalous while also identifying all frames without an anomaly as non anomalous. Note that this is a strict view of true positive since the model must have all true negatives for a given video and at least one true positive to be considered a success. A false negative is defined as identifying no frames of the given anomaly as anomalous and identifying all frames without an anomaly as non anomalous. A false positive is defined as identifying any frame that does not have an anomaly as anomalous. There will be no true negatives in this context because each test video has an anomaly and because true positive's definition encompasses true negatives.

Logistic Regression Results

We trained our logistic regression on the raw training dataset without adding Gaussian blur or noise. This is because given that the testing dataset is similarly distributed to the training dataset, we predicted that logistic regression would learn to output around half true and half false predictions for anomalies given an input frame. By a similar argument however, in the end our logistic regression learned to only output false, that there was no anomaly. This did pass our criterion for a model, never having a false positive, but it means that it also had a 0% true positive rate.

The goal of our neural network is to get the same false positive rate with at least some true positives.

Neural Network Results

For our neural network model, there were a few variables that we needed to play with in order to have the model converge. These included the size of the window that we added noise to in the training step, and the learn rate. With the window size set to 100 pixels, about the size of a golf cart in these frames, and the learning rate set to 0.01, our model was able to converge in around 20 epochs of stochastic gradient descent with a mean squared error loss function.

With our model trained, we test on a test video by predicting labels for each of the 200 frames of the given video and then comparing those predictions to the ground true labels. We then create a confusion matrix shown in 4.

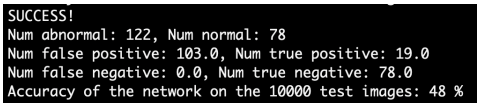

Figure 4: Confusion matrix for one test set video of 200 frames.

From this confusion matrix and the definitions of true positive, false positive and false negative for a video, we can then use this confusion matrix for all 200 frames to label the video. The example in 4 is a true positive. In this example, and in all test videos, our model has 0 false positives. Thus our model only fails with false negatives. In 5, we graph the percentage of true positives for our model. Note that our model does poorly on all anomalies besides golf carts, which it identifies perfectly.

To verify that our model is actually learning how to identify golf carts in CCTV footage, we use a saliency test as shown in 6. Note that the model indeed identifies the golf cart as the anomaly in the frame.

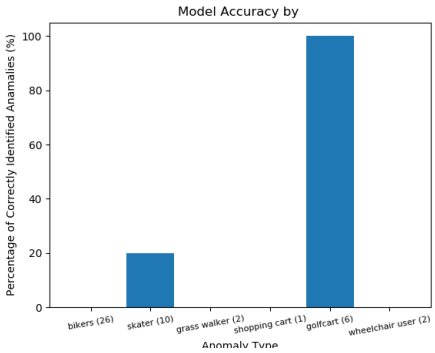

Figure 5: Percentage of false negatives vs true positives for all 38 test videos, split based on type of anomaly. Number in parentheses indicts the number of test videos for that anomaly.

Figure 6: Left: Example image that is predicted to be an anomaly by our model. Right: Heat-map generated by saliency test on the input image. Note that the golf cart is the most heavily colored portion of the frame, indicating that the model correctly labelled this scene as anomalous because of the presence of the golf cart.

Although our model fails to identify any anomalies besides golf carts, the fact that it is able to identify gold cart anomalies perfectly for this testing dataset is encouraging, especially considering the strict restriction to our model that it needs to have 0 false positives for any frames of footage.

We hypothesize that the model is only able to perfectly identify golf carts is because they are the largest and most distinct anomaly. Also, the random noise that we add to our training samples is around the size of a golf cart so we may be over-fitting to this anomaly. For smaller noise box sizes however our model doesn't converge.

Conclusion/ Next steps

Given more time we would try to make our feature set richer, which we believe would allow us to reduce the size of our added square of error and still have our learning converge. This richness could be provided by adding a pre-trained model for feature extraction, having RGBA values in our training set, training and testing on multiple images at a time (use temporal relationship between images). Another, more ambitious addition to this project to also train using a multi-class reference dataset (1) to use as added errors as opposed to Gaussian noise and blur.

It seems that our model would expand to detecting anomalies in medical images very easily. Sonogram and CT scan images are also gray-scale, low resolution images. Furthermore, the fact that our model is built to explain the anomalies that it finds allow it to help not direct doctors.

Contributions

Wilson Ler: Had the idea for a project on anomaly detection and found the dataset we used. Built the neural network model and the saliency test to explain the model's results.

Sean Decker: Built the logistic regression model and wrote the evaluation script for the neural network and logistic regression models. Designed and wrote the poster and wrote the final report for the project.

References

[1] Perera, Pramuditha, and Vishal M. Patel. "Learning Deep Features for One-Class Classification." IEEE Transactions on Image Processing 28.11 (2019): 5450–5463. Crossref. Web.

[2] "Anomaly Detection and Localization in Crowded Scenes." Anomaly Detection and Localization in Crowded Scenes, *Statistical Visual Computing Lab: UC San Diego*, http://www.svcl.ucsd.edu/projects/anomaly/.

[3] Lundberg, Scott M and Lee, Su-In. "A Unified Approach to Interpreting Model Predictions." Advances in Neural Information Processing Systems 30 (2017): 4765-4774. Curran Associates, Inc. Web. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.