Crime Detection in Security Camera Footage

Jee Hun Kang
Graduate School of Arts and Sciences
Fordham University
New York, United States
jkang78@fordham.edu

John Grossi
Graduate School of Arts and Sciences
Fordham University
New York, United States
jgrossi2@fordham.edu

Son Tung Do
Fordham College at Lincoln Center
Fordham University
New York, United States
sdo3@fordham.edu

***Abstract*—Cameras can capture thousands of hours of footage, making manual review an arduous task. By creating a model that can detect if a crime is taking place in a still image, we can assist in reducing this review process. In this research project, we investigated how to apply state-of-the art computer vision models to identify anomalous images among 14 different classes.**

***Index Terms*—Deep Learning, Computer Vision, Anomaly Detection, Crime Detection, ResNet, DenseNet**

I. Introduction

Security cameras can provide protection, guarantee accountability, and serve as evidence in court. However, cameras can capture hundreds if not thousands of hours of footage that must be manually reviewed in the event of a crime. This arduous task allows for human error to creep into the review process. As the vast majority of camera footage is day to day events, footage of a crime is a rare exception and can be a needle in a haystack to find when working without reference. We would like to propose a model to aid in identifying when a crime occurs in security camera footage to aid in the footage review process. The related work has been done by Sanskar Hasija, the grandmaster at Kaggle who tried to produce the best model using the same dataset. Our goal is to produce the better result using the same dataset used by Sanskar Hasija.

In this research project, we adopted computer vision techniques to help in security camera footage review. Our model will identify when a crime occurs in a static image. If a crime is present, it will then identify which crime is taking place from a predetermined set of 13 possible crimes. As existing computer vision models have been trained on datasets far larger than we can, we have chosen to utilize existing pre-trained models and finetune them in anomaly detection and crime classification.

This paper is organized as follows. Section II will discuss our dataset and preprocessing steps taken. Sections III describes our results for our experiments and models chosen. Section IV will cover the reporting of the results of our experiments. Section V covers the conclusion of this research and future work.

II. Data

A. Dataset

A readily available dataset was found from UCF Crime Dataset, a publicly available dataset on Kaggle. [1] The dataset contains extracted images from the UCF crime dataset used for Real-world Anomaly Detection in Surveillance Videos. The dataset contains images extracted from every video from the UCF Crime Dataset. Every 10th frame is extracted from each full-length video and combined for every video in that class. All the images have a size of 64*64 and they are in .png format. This dataset falls under CC0: Public Domain.

The dataset contains 1,377,653 images that encompass 14 different classes. The 13 crime classifications are: abuse, arrest, arson, assault, burglary, explosion, fighting, road accidents, robbery, shooting, shoplifting, stealing, and vandalism. There is an additional ”NormalVideos (Non-Crime)” class which accounts for the majority of the images in both our training and test sets as seen in Figure 1.

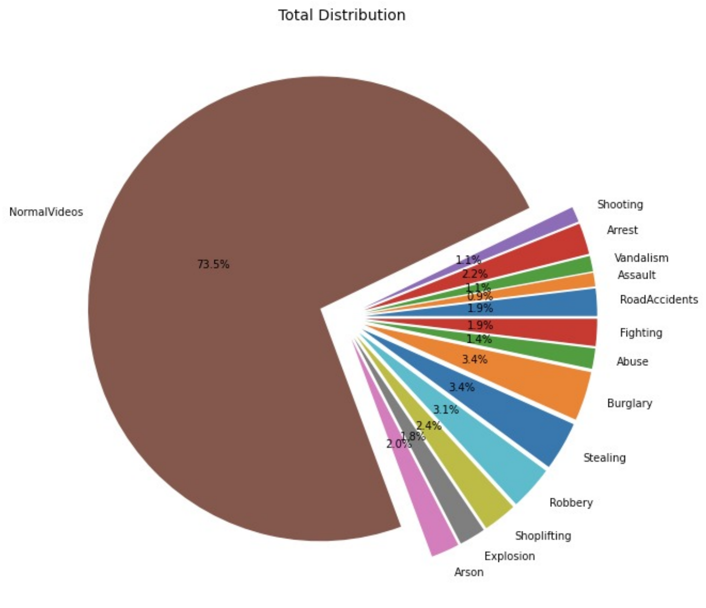

Fig. 1. Total Distribution of Dataset

Fig. 2. Before Preprocessing

B. Data Preprocessing

To preprocess our images, we performed traditional computer vision techniques - image rescaling, width and height shift, conversion into an encoded array, and division into train, validation, and test sets. The challenge of preprocessing came with scaling these traditional techniques to such a large dataset. Opening each image individually and applying a pre-processing function written by us simply took too much time to be deemed acceptable. Therefore, we opted to incorporate a readily available preprocessing function provided by Keras. The preprocessing function used in this project is Densenet.preprocessing, which converts images to numpy arrays in the proper input shape that Densenet models use. We then create an image generator using the ImageDataGenerator package. This package allows us to generate batches of tensor image data with real-time data augmentation. [2] ImageDataGenerator allows us to easily incorporate preprocessing techniques such as horizontal and vertical flips, image re-scaling, and validation splits. This package used in conjunction with Densenet's preprocessing function enables us to build an input pipeline that streamlines our image loading, preprocessing, and categorization. An example image is provided in Figure 2 to illustrate an image before preprocessing. Figure 3 displays an image after preprocessing, converted from a numpy array back to an image.

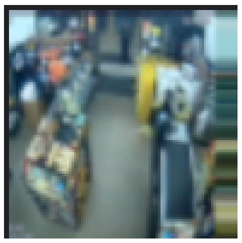

Fig. 3. After Preprocessing

Once our preprocessing pipeline is constructed, we pass both our training and test sets through. The pipeline finished loading both training and test images after 1092.27s. There are a total of 1,377,653 images in our dataset. The train-test split is approximately 92-8: 92 percent of images belong to the training set and 8 percent belong to the test set. The distribution is illustrated in Figure 4.

Our training data has a total image count of 1,266,345; class distribution is displayed in Figure 5. Our test data has a total image count of 111,308; class distribution is displayed in Figure 6.

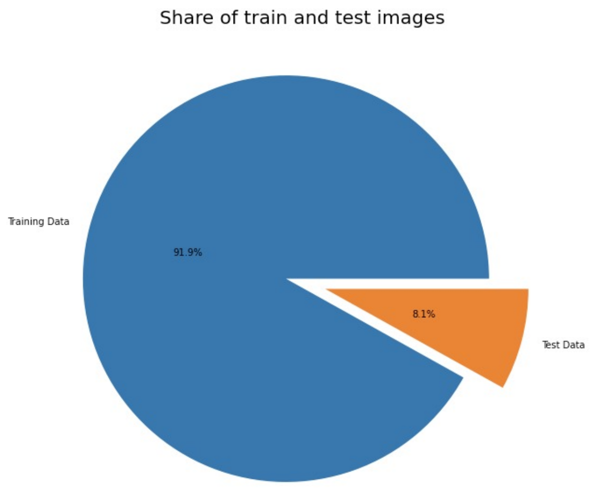

Fig. 4. Traing Test Set Distribution

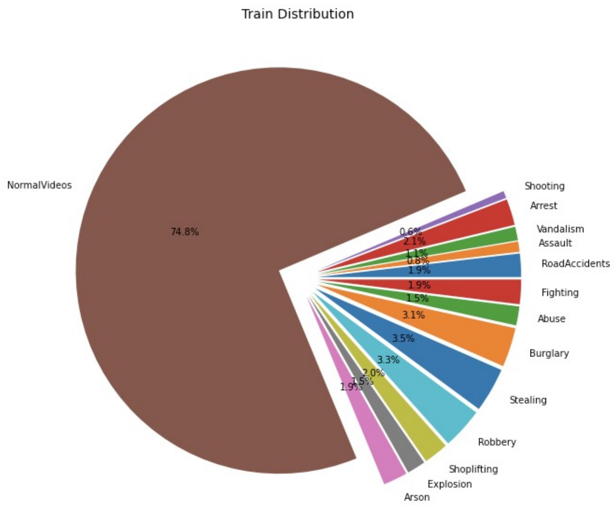

Fig. 5. Class Distribution of Training Set

III. Methodology

We utilize pretrained Convolutional Neural Networks (CNN) models as feature extractors. These CNNs are trained on the ImageNet dataset [3] and are publicly available. Python packages Tensorflow and Keras are used to implement the experiments in this project, and Kaggle environment provides the hardware necessary for training (GPU P100). The pretrained models used in this project are outlined in Table I.

We also try combining 3 different CNNs extractors to get a larger feature map. The features are then fed into fully connected layers to output predictions.

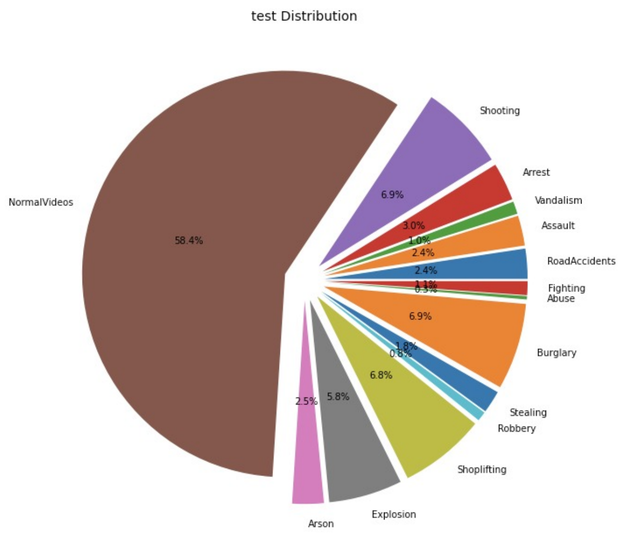

Fig. 6. Class Distribution of Test Set

DenseNet121 [4]
DenseNet201 [4]
EfficientNetB6 [5]
DenseNet121 + ResNet50 + VGG16 [4], [6], [7]

TABLE I
PRETRAINED CNNS USED

IV. RESULTS

In this section we will discuss our results and other interesting findings from our experiments. The best model on Kaggle achieves an ROC AUC score of 0.833 [8].

We chose ROC AUC as our main metric on which to base our analysis as we are performing multi-class analysis. Figure 7 illustrates the ROC AUC curve for our combined CNN model. We believed this model would yield the best results as it would be combing three different CNN models. However, results were not as impressive as anticipated, class results are shown in Table III.

Surprisingly, DenseNet121 was the best performing model, yielding the highest ROC AUC score of all the tested models. Figure 8 and Table IV.

Across the board, certain classes tended to consistently under perform. For example, even on our best model, DenseNet121, classes such as Arrest and Fighting perform worse than random guessing. Abuse and Vandalism perform just as well as random guessing. We believe this is due to the nature of the crime being looked at. For example, fighting would involve two or more people on camera pushing, kicking,

Pretrained CNN feature extractor	ROC AUC Score
DenseNet121	0.837
DenseNet201	0.831
EfficientNetB6	0.771
DenseNet121 + ResNet50 + VGG16	0.816

TABLE II
ROC AUC BY MODEL

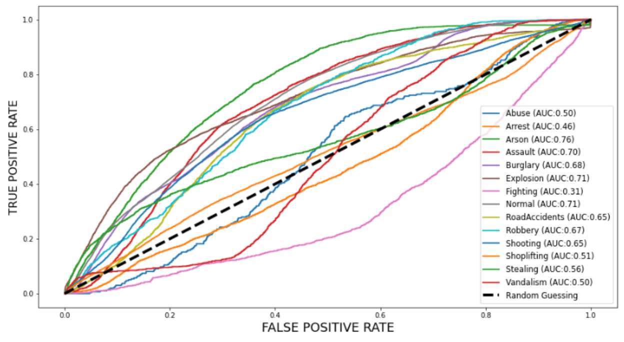

Fig. 7. ROC Curve of 3 CNN Model

Class	ROC AUC Score
Abuse	0.5
Arrest	0.46
Arson	0.76
Assault	0.7
Burglary	0.68
Explosion	0.71
Fighting	0.31
Normal	0.71
RoadAccidents	0.65
Robbery	0.67
Shooting	0.65
Shoplifting	0.51
Stealing	0.56
Vandalism	0.5

TABLE III
ROC AUC CLASS SCORE OF 3 CNN MODEL

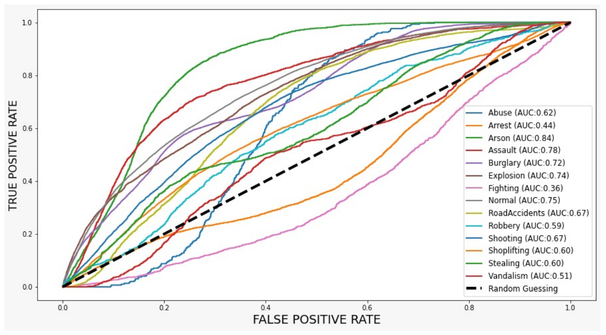

Fig. 8. ROC Curve of Best Model

Class	ROC AUC Score
Abuse	0.62
Arrest	0.44
Arson	0.84
Assault	0.78
Burglary	0.72
Explosion	0.74
Fighting	0.36
Normal	0.75
RoadAccidents	0.67
Robbery	0.59
Shooting	0.67
Shoplifting	0.6
Stealing	0.6
Vandalism	0.51

TABLE IV
ROC AUC CLASS SCORE OF BEST MODEL

shoving, etc. To our model, this may look no different than two or more people in close proximity which could easily be misclassified as NormalVideo.

V. Conclusion

In this research project we have shown interesting findings by performing computer vision methods on image classification and anomaly detection tasks. The DenseNet121 model returns the best results when analyzing the ROC AUC values. However, we could not achieve a substantial increase in performance compared to the current best result on Kaggle. In particular, the result obtained from combining the feature maps of 3 CNNs did not perform as expected.

Overall, the model performs really well on some crimes such as arson or explosion but does poorly on fighting or arrest. Various pretrained CNNs have similar overall performance while suffering from the same poor performance on the fighting class. This suggests that the model's performance bottleneck might be its lack of temporal knowledge.

The largest avenue for future work to continue this project would be to include temporal modules to learn from videos. As we are only dealing with still images sampled from video, many images do not contain any action occurring and can easily be misclassified. Temporal modules would allow the model to look over a sequence of images through time, and we believe this would lead to better results. This method also requires more computational power, as the current vision model already takes approximately 3 hours to train.

References

[1] S. Hasija, "Ucf crime dataset," *Kaggle*, 2021.

[2] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[4] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1608.06993

[5] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2019. [Online]. Available: https://arxiv.org/abs/1905.11946

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[8] S. Hasija, "Video anomaly detection," 2021. [Online]. Available: https://www.kaggle.com/code/odins0n/video-anomaly-detection