A Comparative Study of Machine Learning Algorithms for Classification of Violent Events using CCTV Data Sources

Vijaysri Sriram Iyer
thisisvij98@gmail.com

Yashwanthika R
yashwanthika2000@gmail.com

Solomon Staby
solomonstaby7@gmail.com

R Krishnakumar
r_krishankumar@cb.amrita.edu

Department of Computer Science,
Amrita School of Engineering, Coimbatore
Amrita Vishwa Vidyapeetham

Abstract : Violence Detection from video feeds is one of the important applications of intelligent surveillance systems. In this paper, we propose a classical machine learning pipeline to detect violent activity in surveillance videos. The Histogram of Oriented Gradients (HOG) and Histogram of optical flow (HOF) feature descriptors are extracted from the video and sent as input to a number of tree-based classifiers. We also evaluate and compare the inference times of models in order to study the models as worthy of being deployed in real-world situations by converting them to intermediate representations in PyTorch using the hummingbird library. Our study provides promising results in terms of usability and accuracy for real-time violence surveillance scenarios.

Keywords: violence detection, abnormal event detection, surveillance video monitoring

1. Introduction

In recent years, there has been a tremendous increase in the usage of surveillance cameras across the globe. The video surveillance camera market is expected to grow up to $44 billion by 2025 [1]. CCTV surveillance cameras are an important resource for personal safety as well as to maintain law and order. It can facilitate immediate reporting of crises such as accidents and incidents involving violence to authorities within a certain region. Several governments and law enforcement agencies use mass video surveillance as a means to monitor the movements of pedestrians and traffic in public places. Countries such as the US, China and UK have reported having 15.33 million, 14.36 million and 5.2 million public surveillance cameras respectively [2]. Currently, Delhi city in India has close to 2.75 lakh CCTV surveillance cameras and has the highest number of CCTV cameras per square mile [3]. This number will only increase as the applications in digital analytics and smart cities continue to grow.

The increasing investment in elementary surveillance equipment presents a new problem. Existing surveillance architectures in most countries rely on a human to monitor and report incidents within reasonable time for the authorities to intervene. However, there is a dire shortage of personnel needed to monitor the huge volume of data generated by CCTV cameras on a daily basis. The law enforcement sectors are sparsely staffed, especially in countries with high crime

rates. Hence, it is necessary to automate the process of detecting abnormal/violent incidents in the CCTV video feed, which allows the authorities to only view the essential footage of the incident without needing to parse several gigabytes of data. This also allows for a more sustainable surveillance architecture, aiming to store only the events of particular interest to be used as evidence/proof of an incident.

Owing to the rise of big data, machine learning and deep learning, we have superior processors, algorithms and data centers to store and process surveillance data at scale. A number of efforts have already been made to develop fast and precise algorithms for this problem. Although Deep learning is the new state of the art for such video surveillance tasks, classical machine learning models still offer promising results in terms of accuracy and inference times with a fraction of training data required for a deep learning model. In this paper, we examine various classical machine learning models and their performance in terms of accuracy and inference time to study their efficacy as methods for automated surveillance. In the upcoming sections, we discuss the classical machine learning methods employed by researchers for violence detection as well as some deep learning methods. This will be followed by an analysis of 4 different tree-based and ensemble classification models in terms of their performance and inference time on a publicly available dataset, consisting of violent and non-violent incidents captured on a CCTV surveillance camera. Further, we will discuss the advantages and disadvantages of using such models and how they can be improved for real-world application scenarios of the surveillance problem.

1.1 Related Works

For the literature survey, we have chosen papers from the google scholar search engine from the year 2016-2021 based on the following criteria

1. Models: SVM, Logistic Regression, Decision Trees, Ensemble Techniques
2. Feature descriptors such as LBP, HOG, HOF

A number of works have been completed in the field of automated monitoring of surveillance feed using machine learning methods. In [4], the authors propose a one-class SVM model for abnormal event detection in crowded scenes. They evaluate this task on the UCSD anomaly detection dataset [5], using HOG-LBP and HOF as input features. [6] presents a model for the same dataset with the integration of an SVM, convolutional neural network (CNN) and a deep one-class model. In [7], the feature extraction is performed with a novel feature descriptor called ViF combined with LBP and classified with a linear SVM. Other methods such as particle filtering [8] and ELM [9] models have also been used for classification. On the deep learning front methods such as the CNN-LSTM have been used for Spatio-temporal data understanding [10]. 3D convolutional networks and auto-encoders have also been used with promising results in [11, 12]. Some works with promising results that can be compared to deep learning models are the ones that use optical flow-based methods to compute the change of motion of objects across multiple frames. Optical flow methods allow the model to maintain a record of temporal changes

in data such as video. Hence, in this work, the combination of a HOG and HOF is used to compute features for each video with tree-based and ensemble classification methods.

2. Preliminaries

2.1 Histogram of Oriented Gradients (HOG)

HOG was introduced as a technique in [13, 14] and is a gradient-based feature descriptor used for object detection tasks. This technique is based on the idea that the shape of a local object can be described by the changes in intensity in the pixels surrounding the object. Hence, the gradient determines the magnitude of changes in orientation in the x and y directions for each pixel, within a localized portion of the image known as a cell. The equations for gradient magnitude and angle calculation are shown in (1),(2).

$$G_x(r,c) = I(r,c+1) - I(r,c-1)$$
$$G_y(r,c) = I(r-1,c) - I(r+1,c) \quad (1)$$

$$magnitude = \sqrt{G_x^2 + G^2 y}$$
$$Angle = |\tan^{-1}(\frac{Gy}{G_x}) \quad (2)$$

The orientations of each pixel within are combined to form a histogram for the cell. The histograms of all cells are combined to build the descriptor. The advantage of HOG over the other feature descriptors is that it uses the magnitude and angle of the gradient for computation. This allows it to enjoy invariance to geometric and photometric transformations. For this video classification task, the OpenCV (C++) [15] implementation of the HOG descriptor was used. An illustration of the HOG descriptor is provided in fig 3.

2.2 Dense Optical Flow and Histogram of Optical Flow (HOF)

The HOG descriptor is very useful in order to determine shapes and features of static objects. However, in order to perform activity detection, the motion trajectory of an object is equally important. The optical flow feature descriptor provides insight into the displacement of objects in an image over the course of multiple frames. This is known as a flow vector. There are two major optical flow algorithms in traditional computer vision, the Lucas-Kanade method (sparse optical flow) and the Gunnar-Farneback method (dense optical flow) [16]. While sparse optical flow provides flow vectors of regions of interest in an image, the method provides flow vectors for the entire frame. The equation of optical flow is shown in (3).

$$\frac{dI}{dx}u + \frac{dI}{dy}v + \frac{dI}{dt} = 0$$

(3)

Since, the violent event classification task is performed on a dataset of crowded scenes, the Gunnar-Farneback method of optical flow calculation is used within the OpenCV library. Like the previous descriptor, the HOF is a histogram consisting of flow vectors in the entire frame. An illustration of the HOF feature descriptor is provided in fig 1.

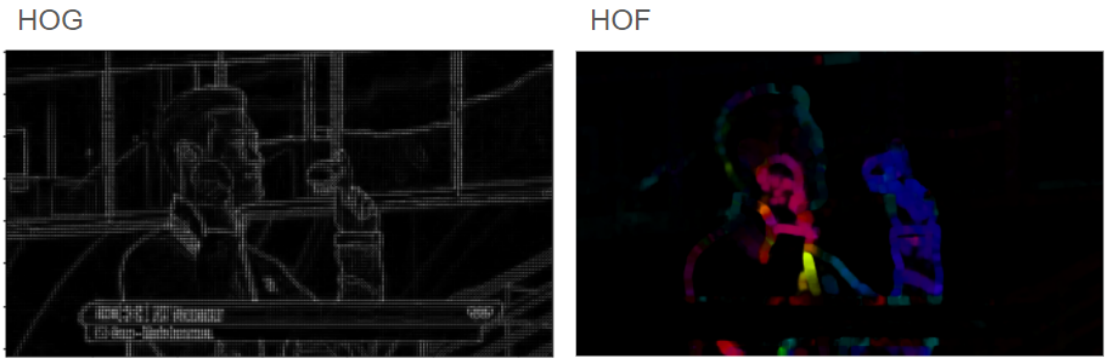

Fig.1. Visualization of HOG(left) and HOF(right) Feature descriptors

2.3 Decision Tree

Decision tree [17] is a binary branching structure built through a process of recursive partitioning. Each node of the tree does the feature comparison and returns either true or false which determines whether to proceed along the left or right child of the node. Decisions can follow multiple algorithms such as ID3, CART and C4.5 for node splitting. However, the CART algorithm which uses the Gini index to construct the decision tree proves to be very effective for the video classification task. The equation for gini index is provided in (5).

$$Gini\ Index = 1 - \sum_{i=1}^{n} p_i^2$$

(5)

2.4 Random Forest Classifier

Random Forest [18] is a bagging based ensemble machine learning model that contains a number of decision trees or forest built on various subsets of features selected at random and takes the average to improve the accuracy of prediction. The random forest takes the prediction from each tree and based on the majority of the votes the final output is predicted. The higher the number of trees leads to higher accuracy and prevents overfitting.

2.5 Gradient Boosting Classifier

This classifier [19] is a combination of AdaBoost with weighted minimization followed by recalculation of input weights. The algorithm tries to minimize errors in sequential models by using an iterative gradient algorithm. Each classifier tries to improve on its predecessor by reducing the bias. Instead of fitting a classifier on each iteration, the algorithm fits a new classifier to the residual errors made by predecessors.

2.6 XGBoost

XGBoost stands for "eXtreme Gradient Boosting", which is a highly-scalable customized version of gradient boosting with increased speed and performance [20]. It is an optimized gradient boosting algorithm through parallel processing, tree pruning, effective handling of missing values and usage of regularization to avoid overfitting. As the training data was large it was split into three halves and incremental training was performed .

3. Materials and Methods

3.1 Dataset

The models were trained on a publicly available ‘Real-world situations of violence’ dataset available on Kaggle [21]. This dataset consists of 2000 videos equally distributed into two classes namely “violent” and “non-violent”. Since the dataset was collected from online sources featuring videos of mob violence, accidents and street fights, the instances are diverse in nature and represent situations similar to those in a real-world setting. A number of images from the dataset are shown in fig 2. For training the model on a CPU configuration machine, the dataset was divided into 3 runs, consisting of 500 videos (250 videos per class) for training. This was evaluated on a holdout set sampled from the data consisting of 500 videos. The ratio of total training to testing data is 3:1.

Fig.2. Real-world situations of violence Dataset samples of both violent and non-violent classes

3.3 Proposed Method

In this paper, we define the task as a framewise classification task, where each frame of a video is labeled as violent/non-violent by the machine learning model. So, for every video the HOG and HOF descriptors are computed once every 3 frames. This reduces the overall training time of the dataset while providing a high accuracy. The HOG and HOF features are stacked together and sent as input to a classification model.

The pre-processed feature vectors are stacked and sent to various tree-based ensemble models for classification. The models used in this paper are Decision Tree, Random Forest, Gradient Boosting Trees and XgBoost. The parameters for each of the tree-based models including the choice decision tree algorithm was chosen using a Grid Search [22] method.

The models were trained on a Google Colab notebook, on a single core CPU. All the models took less than 30 mins to train on a single run. All the models, except for the xgboost model, were trained on the entire data in one-shot.

4. Results and Discussion

4.1 Model Performance

Model	Accuracy	F1-score	ROC-AUC Score
Decision Tree Classifier	**.95**	**.95**	**.95**
Gradient Boosting Classifier	**.96**	**.97**	**.97**
Random Forest Classifier	**.94**	**.94**	**.94**
XGBoost	**.95**	**.94**	**.95**

Table 1. Accuracy, F1 Score and ROC-AUC score for various machine learning algorithms we used for classification. Gradient boosting classifier clearly performs better

4.2 Inference Time

In order to assess the suitability of the trained models on a real-time use case, the inference times were also analyzed on a small holdout set of 10 videos (784 frames) per class. Scikit-learn models are not optimized to utilize tensor computation methods and for CPU and GPU compatibility. Hence, the hummingbird [23] open-source library by microsoft, was used to convert the trained models to an intermediate representation in the pytorch framework. The resultant models showed improvements in inference time with added support for GPU processing capabilities. The inference times of these models are provided in table 2.

Machine learning model	Inference time		
	sklearn	**pytorch -CPU**	**pytorch -GPU**
Decision Tree Classifier	7.32 µs	7.88 µs	7.84 µs
Gradient Boosting Classifier	9.91 µs	15.05 µs	7.93 µs
Random Forest Classifier	74.36 µs	303.57 µs	40.17 µs
XGBoost	43.87 µs	257.65 µs	24.23 µs

Table 2. Inference time taken by the algorithms on different machine learning platforms, and various hardwares

4.3 Model Deployment Methods

Using the hummingbird library, the scikit-learn models can be converted into tensor computations in pytorch/ONNX, which can be used to deploy the models on edge devices. This makes it possible to use tree-based models for video classification purposes on real-time devices on CCTV cameras.

Conclusion

In this paper, we have tried to bring different insights on how classical machine learning algorithms work on detecting violence from a video. HOG and HOF algorithms were used for image processing. Gradient boosting classifier was the one which gave best accuracy among the algorithms we tested, whereas decision tree classifier topped in the inference time when predicted with CPU as well as GPU with Google Colab-pro support. Since the dataset has a number of examples with high diversity, the testing accuracies of the models vary across frames. Since CCTV cameras are usually stationary, a model trained with similar background settings would perform better in such scenarios. Further developments include searching for cleaner and more homogenous data while moving towards more accurate and fast classical machine learning models for real-time automated surveillance.

References

[1] Whitney, L., 2020. Demand for video surveillance cameras expected to skyrocket. [online] TechRepublic. Available at: link .

[2] Desk, A., 2022. Top 10 Countries and Cities by Number of CCTV Cameras. [online] AiThority. Available at: link .

[3] News, C. and News, d., 2022. Delhi has maximum CCTVs per sq mile in world | Delhi News - Times of India. [online] The Times of India. Available at: link

[4] Amraee, S., Vafaei, A., Jamshidi, K. and Adibi, P., 2018. Abnormal event detection in crowded scenes using one-class SVM. Signal, Image and Video Processing, 12(6), pp.1115-1123.
[5] Accattoli, S., Sernani, P., Falcionelli, N., Mekuria, D.N. and Dragoni, A.F., 2020. Violence detection in videos by combining 3D convolutional neural networks and support vector machines. Applied Artificial Intelligence, 34(4), pp.329-344.
[6] Sun, J., Shao, J. and He, C., 2019. Abnormal event detection for video surveillance using deep one-class learning. Multimedia Tools and Applications, 78(3), pp.3633-3647.
[7] Vashistha, P., Bhatnagar, C. and Khan, M.A., 2018, March. An architecture to identify violence in video surveillance system using ViF and LBP. In 2018 4th international conference on recent advances in information technology (RAIT) (pp. 1-6). IEEE.
[8] Tariq, S., Farooq, H., Jaleel, A. and Wasif, S.M., 2021. Anomaly detection with particle filtering for online video surveillance. IEEE Access, 9, pp.19457-19468.
[9] Guangli, W.U., Liping, L.I.U., Chen, Z. and Dengtai, T.A.N., 2019, July. Video abnormal event detection based on ELM. In 2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP) (pp. 367-371). IEEE.
[10] Ye, O., Deng, J., Yu, Z., Liu, T. and Dong, L., 2020. Abnormal event detection via feature expectation subgraph calibrating classification in video surveillance scenes. IEEE Access, 8, pp.97564-97575.
[11] Asad, M., Yang, J., Tu, E., Chen, L. and He, X., 2021. Anomaly3D: Video anomaly detection based on 3D-normality clusters. Journal of Visual Communication and Image Representation, 75, p.103047.
[12] Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H. and Hua, X.S., 2017, October. Spatio-temporal autoencoder for video anomaly detection. In Proceedings of the 25th ACM international conference on Multimedia (pp. 1933-1941).
[13] Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H. and Yoshimoto, M., 2012, October. Architectural study of HOG feature extraction processor for real-time object detection. In 2012 IEEE Workshop on Signal Processing Systems (pp. 197-202). IEEE.
[14] Minetto, R., Thome, N., Cord, M., Leite, N.J. and Stolfi, J., 2013. T-HOG: An effective grdient-based descriptor for single line text regions. Pattern recognition, 46(3), pp.1078-1090.
[15] https://docs.opencv.org/3.4/d5/d33/structcv_1_1HOGDescriptor.html
[16] Alvarez, L., Weickert, J. and Sánchez, J., 2000. Reliable estimation of dense optical flow fields with large displacements. International Journal of Computer Vision, 39(1), pp.41-56.
[17] Priyam, A., Abhijeeta, G.R., Rathee, A. and Srivastava, S., 2013. Comparative analysis of decision tree classification algorithms. International Journal of current engineering and technology, 3(2), pp.334-337.
[18] Liu, Y., Wang, Y. and Zhang, J., 2012, September. New machine learning algorithm: Random forest. In International Conference on Information Computing and Applications (pp. 246-252). Springer, Berlin, Heidelberg.

[19] Bentéjac, C., Csörgő, A. and Martínez-Muñoz, G., 2021. A comparative analysis of gradient boosting algorithms. Artificial Intelligence Review, 54(3), pp.1937-1967.
[20] Ramraj, S., Uzir, N., Sunil, R. and Banerjee, S., 2016. Experimenting XGBoost algorithm for prediction and classification of different datasets. International Journal of Control Theory and Applications, 9(40).
[21] Kaggle.com. Real Life Violence Situations Dataset. link
[22] Del Cueto, M., 2020. Grid Search in Python from scratch— Hyperparameter tuning. [online] Medium. Link.
[23] https://github.com/microsoft/hummingbird