Violence Detection using Neural Networks

Anirudh Ganguly
Dept. Software Engineering
San Jose State University
anirudh.ganguly@sjsu.edu

Riddhi Jain
Dept. Software Engineering
San Jose State University
riddhi.jain@sjsu.edu

Sheema Murugesh Babu
Dept. Software Engineering
San Jose State University
sheema.murugesh@sjsu.edu

Somya Mishra
Dept. Software Engineering
San Jose State University
somya.mishra@sjsu.edu

ABSTRACT

We live in a world where we rely heavily on usage of surveillance cameras for our ensuring high levels of security. Even though these smart cameras are easily available and used everywhere, these monitoring is highly ineffective and unreliable due to a number of reasons. The most important being monitoring of huge quantities of video footage to be able to detect any incidents of violence. Although Violence detection has been receiving significant attention due to its numerous practical use cases, it lacks in-depth research due to more focus on action recognition based on detecting simple actions like clapping, walking, or jogging etc.

Keywords—

Convolutional neural networks, Long Short-term Memories, VGG16, and VGG19, ResNet50, TFX, Inception V3, Transfer Learning, Statistics Gen, Schema Gen, Example Validator, Trainer, Pusher

INTRODUCTION

Vision-based Action Recognition is one of the most interesting as well as challenging research topics of computer vision and developing a technique for the analysis of videos in order to identify any violence is what this paper focuses on.

The problem statement for the project consists of two parts: feature extraction. and classification. The approach applied for feature extraction will be using deep convolutional neural networks-based representations (CNNs). Once feature extraction is completed, Long Short-term Memories (LSTM) will be used for modeling the temporal information, as they help to find out relationships between the consecutive frames through their memory ability. The proposed model will use a spatial feature extractor followed by Long Short-Term Memory (LSTM) as a temporal feature extractor and sequence of fully connected layers for classification purposes with attention mechanisms that would help enhance the performance of the LSTM. In summary, CNN + LSTM network will be used for this action recognition due its proven high performance in previous models.

RELATED WORK

There has been some research previously to identify violence in images as well as videos. [1] uses 3 ImageNet models; VGG16, and VGG19, ResNet50 for extracting features. And then the extracted features from frames of videos were fed into an LSTM network. Furthermore, they have applied attention to the features extracted from the frames through a spatial transformer network which also enables transformations like rotation, translation and scale. Along with these models, they designed a custom convolutional neural network (CNN) as a feature extractor and a pretrained model which is initially trained on a movie violence dataset. In the end, the features extracted from the ResNet50 pretrained model proved to be more important towards detecting violence.

Another research work in the field of action recognition, specifically violence detection [2] consists of a CNN for frame level feature extraction followed by feature aggregation in the temporal domain using convLSTM because they stated that the consLSTM is capable of encoding the spatial and temporal changes using the convolutional gates present in them and this helps in generating a better representation of the video under analysis.

The network consists of a series of convolutional layers followed by max pooling operations for extracting discriminant features and convolutional long short memory (convLSTM) for encoding the frame level changes that characterizes violent scenes, existing in the video.

A research also talks about Violence Detection using a different type of CNN- it uses Modified Xception CNN for training using fight videos so that the CNN is more familiar with the input and extracts more relevant features from them, coupled with a (Bi-LSTM) and a self-attention layer for better model's performance.

DATASET

We evaluated the performance of our models on 2 standard publicly available datasets: first Hockey Fight Dataset and second Movies Dataset. They contain videos captured using mobile phones, CCTV cameras and high-resolution video cameras.

Hockey Fight Dataset:

1. Created by collecting videos of ice hockey matches.
2. Contains 500 fighting and 500 non-fighting videos.
3. Similar background and subjects in videos.

Movies Dataset:

1. Created by collecting videos of stadium crowds from football matches.
2. Contains 100 fighting and 100 non-fighting videos.
3. Different background and subjects in videos.

OUR APPROACH:

The approach followed by us for the project can be listed as follows:

1. Data Preprocessing (converting video to frames, data augmentation)
2. Models' Implementation
3. Visualization
4. TFX Pipeline Integration

5. Deployment

Each of these steps are elaborated in the following sections.

Data Preprocessing

For this large number of video dataset. Performed following as a preprocessing steps:

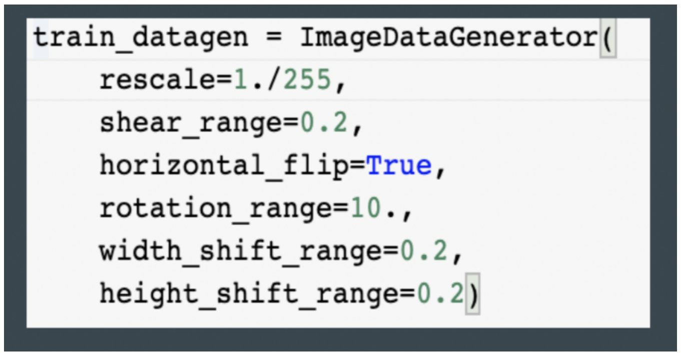
1. Convert Video to Image Frames
- Approach 1: Clipped the full video into frames and saved it as an image in a folder which will be then used for performing training and validation for model

- Approach 2: Converted the full video clip into 20 image frames which were normalised and it is used to deal with less but quality data

2. Resizing images
Once a video clip is converted into frames, it is then resized into the shape of (299, 299) to match the size of the input layer of inception v3 model.

3. Data Augmentation
Data augmentation is the process which helps in increasing the relevant amount of data needed to train the model. Here we have followed the following augmentation techniques:
- Horizontal Flip
- Rescale
- Shear Range
- Image Rotation
- Width and height shift.

Below is the image of our code snippet and a grid of the final data augmented images.



```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    horizontal_flip=True,
    rotation_range=10.,
    width_shift_range=0.2,
    height_shift_range=0.2)
```

Figure 1: Data augmentation types applied to the frames.

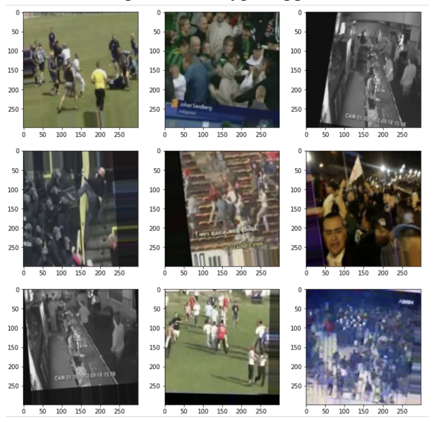


Figure 2: Data Augmented Images.

Models

Inception V3 + Transfer Learning

Inception v3 is a state of the art model widely used for image recognition and has been shown to attain greater accuracy of 78% with the widely used imagenet dataset. In this model, we are using the inception v3 pretrained model to perform transfer learning which helps in increasing the accuracy of our model to 90% with less epochs and less training dataset.

On the base inception v3 output layer, we added global average pooling 2D layer + Dense layer with relu activation function + Dense layer with softmax activation function.

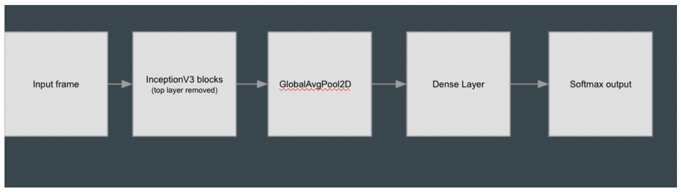


Figure 3: Inception V3 workflow.

Two Training Flows we followed.
- Freeze all except the top layer
In this approach, we freeze all but the top layer of inception v3 model and train the top layer with the training data. This resulted in 82% accuracy.

- Freeze all except the top and the middle layer
In this approach, we freeze all but the top and the middle layer of inception v3 model and train the top layer with the training data. This results in 90% accuracy.

CNN + LSTM

For our second technique, we use CNN and LSTM to take into account the spatial information in an image frame and the temporal information across image frames in the video. the network is divided into two parts. In the first part, we use the VGG16 model. VGG16 is a state of the art CNN based model which is pretrained on the imagenet dataset. In the second part, we use the LSTM network to capture the temporal information. Figure 4 summarizes the network architecture.

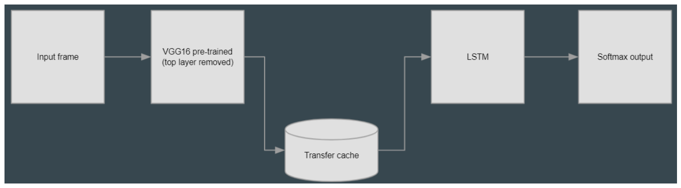
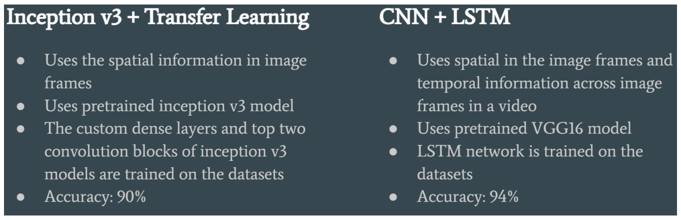


Figure 4: CNN + LSTM architecture.

As part of the training flow, first the VGG16 model is used as is and inference is run on the training images. the last layer which is the classification layer of the model is removed. This results in feature map generation. The result from the VGG16 model is saved in a cache file in .h5 format. This is done because the VGG16 model execution on the dataset is computationally expensive and time consuming. Once the transfer cache is available, we pass the cache as in input to the LSTM model. The final layer of the LSTM model is a softmax layer with two neurons which give the final classification result.

MODELS COMPARISON

Figure 5 summarizes and compares the key features of two models that are described in this paper.



Inception v3 + Transfer Learning	CNN + LSTM
• Uses the spatial information in image frames • Uses pretrained inception v3 model • The custom dense layers and top two convolution blocks of inception v3 models are trained on the datasets • Accuracy: 90%	• Uses spatial in the image frames and temporal information across image frames in a video • Uses pretrained VGG16 model • LSTM network is trained on the datasets • Accuracy: 94%

Figure 5: CNN + LSTM architecture.

ML - OPS

MLOPS is a compound of machine learning + information technology. It is a new discipline/focus/practice for collaboration and communication between data scientists and information technology (IT) professionals while automating and productizing machine learning algorithms.

TFX Pipeline

TFX pipeline is a sequence of components that implements a Machine Learning pipeline which is specifically designed for scalable, high-performance machine learning tasks. That includes modeling, training, serving inference, and managing deployments to online, native mobile, and web applications.

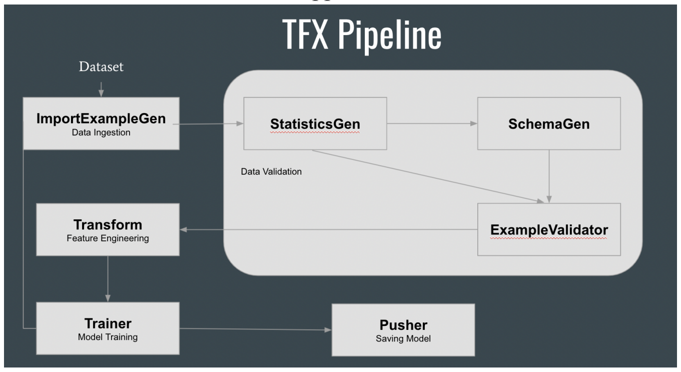


Figure 6: TFX Pipeline flow.

We have implemented following components:

1. Import Example Gen

It is an initial component of a pipeline which reads data as tfRecords and splits it into training and validation dataset and finally saves as tfRecord files for future access.

2. Statistics Gen

It is the second step of the pipeline which calculates statistics for both the train and validation data. It takes tf record files generated by import example gen as input.

3. Schema Gen

It is the third step of the pipeline which generates the schema of the tf record files generated by the import example gen component. It examines the characteristics such as data type for each column in the input dataset.

4. Example Validator

It is equivalent to the data validation step of a traditional machine learning pipeline. Output from both the Statistics Gen and Schema Gen is then passed on to the example validator to detect if there are any anomalies present in both the train and validation dataset.

5. Transform

It is equivalent to the feature engineering step of the traditional machine learning pipeline. Import example gen components generated tfRecord files are fed as an input to the transform step and data augmentation such as Rotation, ReScaling, flip etc.

6. Trainer

This component takes the model which we are planning to use for our problem and the transformed data as input. It then trains the model for the specified number of epochs. Once the model is trained on the training set, a validation set is used to calculate the accuracy of the model. After completing the weights of the model, the model is saved as a saved_model folder for future deployment.

7. Pusher

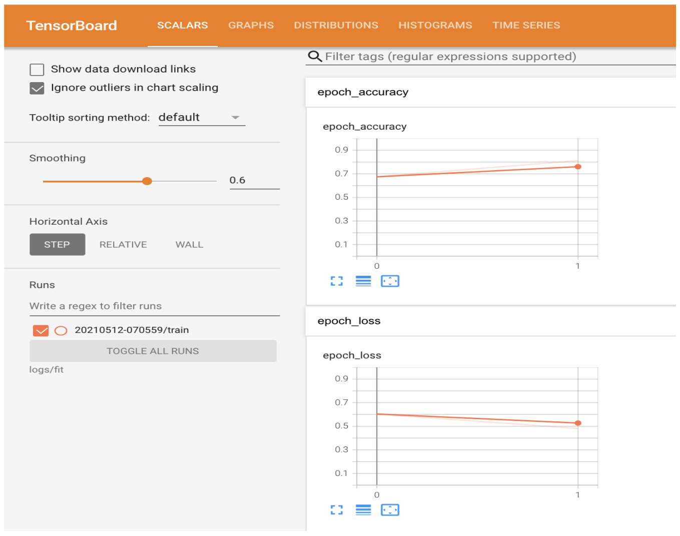
This is the last component which is used to push the saved model and deploy it on the server url specified along with the weights learned so far in the training step. We can visualize the output of pusher using its output method.

VISUALIZATION

Visualizations are vital to depict information of the findings. TensorBoard visualizations [4] for depicting accuracy and loss are provided for our models.

Tensor Board

Below is the image of the visualization and have also uploaded this to the tensorboard.dev.



Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP RELATIVE WALL
Runs
Write a regex to filter runs
20210512-070559/train
logs/fit

Figure 7: TensorBoard Outputs.

The tensor board dev live link can be found at, https://tensorboard.dev/experiment/EbeqmH2ZQKmGi00dbLmfQg/.

Python Flask Application

A simple Flask application is created and deployed on heroku which takes the input as a video url and returns the output where the video contains violence or no voilence. The model saved in the trainer step is loaded using tensorflow keras library using method load_model. The loaded model is then used for predicting the data. As a preprocessing step for test url, cv2 library of python is used to read the contents of video present at the url. The video is then converted into n/20 frames.
Model is used to predict the outcome for each frame. Based on the highest number of either no violence or violence, the result is then returned to the user.

CONCLUSION

Surveillance and public monitoring has resulted in cameras and videography in our everyday life. It has become extremely important to build solutions to automatically identify the kind of activity happening in a video footage especially if it is a violent or harming action. In this paper, we have described two techniques using deep learning principles to address this problem. The first technique uses CNN based model which is known to capture the spatial information in images or video frames, and in the second approach uses CNN + LSTM model to capture both spatial and temporal information in the data. We have experimented with publicly available datasets like the hockey dataset and movies dataset and captures the performance of both the approaches. Finally, we observed that the CNN+LSTM model gives a slightly better accuracy of 94% compared to the CNN only model which gave an accuracy of 90%.

References

[1] Shakil Ahmed Sumon, Raihan Goni, Niyaz Bin Hashem, Tanzil Shahria and Rashedur M. Rahman, in "Violence Detection by Pretrained Modules with Different Deep Learning Approaches". Available: https://www.worldscientific.com/doi/pdf/10.1142/S2196888820500013
[2]Swathikiran Sudhakaran and Oswald Lanz, in "Learning to Detect Violent Videos using Convolutional Long Short-Term Memory". Available: https://arxiv.org/pdf/1709.06531.pdf
[3]Seymanur Akti, Gozde Ayse Tataroglu and Hazim Kemal Ekenel, in "Vision-based Fight Detection from Surveillance Cameras". Available: https://arxiv.org/pdf/2002.04355.pdf
[4]TensorBoard Tutorial. Available at: https://www.tensorflow.org/tensorboard/get_started