INDYCAR VIDEO CRASH DETECTION USING SPATIO-TEMPORAL AUTOENCODER

Sai Prasad Parsa
MS Data Science (Residential)
Indiana University
Bloomington IN USA
saiparsa@iu.edu

Humshavarthini Karunanidhi
MS Data Science (Residential)
Indiana University
Bloomington IN USA
vkaruna@iu.edu

1. INTRODUCTION

The IndyCar Series, currently known as the NTT IndyCar Series under sponsorship, is the premier level of open-wheel racing in North America. The race is the most prestigious event of the IndyCar calendar, and one of the oldest and most important automobile races. Our project is based on developing a crash detection framework using the video sequences from the IndyCar Series.

1.1 MOTIVATION

There is an increasing need not only for recognition of objects and their behavior, but also for detecting the rare, interesting occurrences of unusual objects or suspicious behavior in the large body of ordinary data. Finding such abnormalities in Indycar race videos is crucial to provide increased safety and to develop better winning strategies for the drivers in the Series. Abnormal events that are of interest to us in the IndyCar race events are car crashes or damages. But these events often have an extremely low probability of occurring in a long video footage. As such, it is a very meticulous job to manually detect such events, or anomalies, that often requires more manpower than is generally available. This has prompted the need for developing an automated detection of anomalies or crashes in the IndyCar race events and motivated us to use a deep learning approach.

1.2 GOAL

The success of deep learning methods in various applications consequently caused the rise of such methods in anomaly detection. Our goal in this project is to develop a deep learning-based framework to represent video data by a set of general features which are inferred automatically from a long video footage without the need to manually define a specific set of features to be extracted from the dataset. Thus, we worked on an unsupervised deep learning approach that learns the useful features directly from the data with minimal preprocessing and reduced additional human efforts. Specifically, a deep neural network composed of a stack of convolutional autoencoders was used to process video frames in an unsupervised manner that captured spatial structures in the data, which, grouped together, compose the video representation. Then, this representation is fed into a stack of convolutional temporal autoencoders to learn the regular temporal patterns. Our approach is based on the principle that when an abnormal event occurs, the most recent frames of video will be significantly different than the older frames. Unlike supervised methods, these methods only require unlabeled video footages which contain little or no abnormal event.

1.2 LITERATURE

Video data is challenging to represent and model due to its high dimensionality, noise, and a huge variety of events and interactions. Anomalies are also highly context specific, ambiguous, and often vaguely defined. These challenges have made it difficult for machine learning methods to identify video patterns that produce anomalies in real-world applications. There are many successful cases in the related field of action recognition [1][6]. However, these methods are only applicable to labelled video footages where events of interest are clearly defined. But the cost of labelling every type of event is extremely high. Even so, it is not guaranteed to cover every past and future event. The recorded video footage is likely not long enough to capture all types of activities, especially abnormal activities which rarely or never occurred.

Recent effort on detecting anomalies by treating the task as a binary classification problem (normal and abnormal) proved it being effective and accurate [3], but the practicality of such a method is limited since footage of abnormal events are difficult to obtain due to its rarity. Therefore, many studies have turned to models that can be trained using little to no supervision [4], including spatiotemporal features, dictionary learning and autoencoders [1]. Since it is easier to get video data where the scene is normal in contrast to obtaining what is abnormal, related works focus on a setting where the training data contains only normal visual patterns. The popular approach is to first learn the normal patterns from the training videos, then anomalies are detected as events deviated from the normal patterns. Majority of the work on anomaly detection relies on the extraction of local features from videos, that are then used to train a normal model.

2.1 ARCHITECTURE

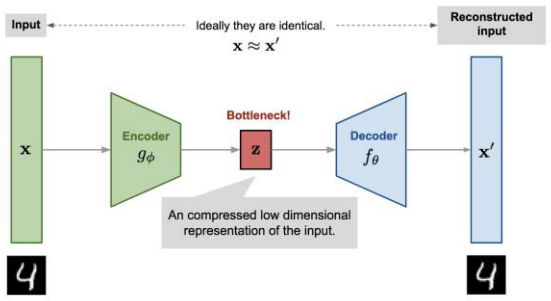

Figure 1. Basic Autoencoder representation

Our architecture is a deep neural network composed of a stack of convolutional autoencoders used to process video frames. Auto Encoder (AE), like the one shown in Figure 1 is a neural network that learns to copy input to output and the network is usually trained using back-propagation in an unsupervised manner. It consists of an encoder which acts as a down sampler while the decoder acts as an up sampler. In autoencoder, the encoder unit maps the input to a hidden layer and the units in the hidden layer maps the input to lower dimensional latent space representation in an unsupervised manner. Principal Component Analysis (PCA) algorithm also reduces dimensionality, but the uniqueness of an autoencoder is that it is applicable to non-linear data as well since the activation function is chosen to be nonlinear. AE also learns more powerful generalization and extract more useful features compared to PCA and it reconstructs the output with considerably low loss of information than PCA. The decoder unit takes the low dimensional data into reconstruction space and tries to reconstruct the output that is closer to the input with dimension equal to that of input.

In a simple autoencoder, the layers are fully connected to each other. But in the proposed model, a convolutional autoencoder is used which extends the basic structure of the simple auto encoder but the fully connected layers are changed to convolution layers. The encoder network has convolutional layers, and the decoder network has transposed convolutional layers. The primary purpose of using convolutional layers in the model is to extract features from the input image [3][5]. It learns the image features using small squares of input data and thus it preserves the spatial relationship between the pixels in the image. Once the number of filters, filter size, and the number of layers is defined before training, convolutional layers learn the values of these filters on their own during the training process. Convolution does dot product of these different filters and local regions of the input to learn the features in the input image**.**

Learning temporal dependencies between inputs are important in tasks involving sequences, where the output vector is influenced not only by the input vector but also on the entire history of inputs. Recurrent Neural Networks (RNN) fail here due to vanishing gradients. These networks are limited to looking back only a few steps. Long Short-Term Memory (LSTM) networks are used to overcome the vanishing gradients problem. These networks are well-known for learning temporal patterns and predicting time series data and they can work on long sequences and capture higher level information when stacked together. Recent studies show that a variant of the fully connected LSTM architecture, namely Convolutional LSTM (ConvLSTM) is promising for learning the regular temporal patterns in videos and they are widely used for video frame prediction. ConvLSTM has its matrix operations replaced with convolutions. By using convolution for both input-to-hidden and hidden-to-hidden connections, ConvLSTM requires fewer weights and yields better spatial feature maps.

The proposed model thus takes advantage of Convolutional Neural Network and Recurrent Neural Network and leverages a Convolutional LSTMs Auto-Encoder (ConvLSTM-AE) to model normal appearance as well as the motion patterns at the same time. The model has two parts: a stack of convolutional spatial autoencoders and a stack of convolutional temporal encoder-decoder. The spatial encoder and decoder have two convolutional and deconvolutional layers respectively and learn the spatial features in the video frame while the temporal encoder is a three-layer Convolutional LSTM model and learn the temporal patterns of the encoded spatial structures.

2.2 IMPLEMENTATION

An end-to-end model that consists of a spatial feature extractor and a temporal encoder-decoder which together learns the temporal patterns of the input volume of frames is trained [1]. The model is trained with video volumes consisting of only normal scenes. The objective here is to minimize the reconstruction error between the input video volume and the output video volume reconstructed by the learned model. The trained model is expected to give low reconstruction error for normal video volume, whereas video volume consisting of abnormal scenes is expected to have high reconstruction error.

Hyperbolic tangent as the activation function for spatial encoder and decoder. Rectified linear unit (ReLU) activation is not used despite its regularization ability because activated values from ReLU have no upper bound and to ensure the symmetry of the encoding and decoding. The loss function is Mean Squared Error (mse) and the optimizer is ADAM optimizer with default learning rate 0.001 for this constrained optimization problem.

Implementation can be divided into 3 main stages:

1. Preprocessing
2. Spatio Temporal Feature Extraction
3. Thresholding and Reconstruction Error

2.2.1 PREPROCESSING

The first stage of implementation is to convert the raw video into video volumes which is the input requirement of the proposed network. Preprocessing step involves 3 tasks.

1. Frame Extraction
2. Normalization and Scaling
3. Grayscale Conversion

Each frame is extracted from the raw videos and resized to 227 x 227 pixels. These input images (extracted frames) are normalized by subtracting every frame from its global mean image where global mean image is calculated by averaging the pixel values at each location of every frame in the training dataset.

The pixel values are scaled between 0 and 1 to ensure that the images are all on the same scale. These images are further converted to grayscale to reduce the dimensionality and are then normalized to have zero mean and unit variance. Channel wise grayscale conversion is employed to ensure that the different lighting conditions in the video are balanced. Input to the model is video volumes, where each volume consists of 10 consecutive processed frames.

2.2.2 FEATURE EXTRACTION

The model we are using to learn the regular patterns in the training videos is a convolutional spatiotemporal autoencoder. The architecture consists of two parts.

1. Spatial Autoencoder
2. Temporal Encoder-Decoder

Spatio Autoencoder shown in Figure 2 learns spatial structures of each video frame. The primary purpose of this network is to extract features from the input image by preserving the spatial relationship between pixels. The spatial encoder takes 10 frames at a time as input and encodes their features, these encoded features are concatenated and fed into temporal encoder for motion encoding. The decoders mirror the encoders to reconstruct the video volume.

The temporal encoder-decoder shown in Figure 3 learns temporal patterns of the encoded spatial structures. The ConvLSTM determines the future state of a certain cell in the grid by the inputs and past states of its local neighbors. Hence it is responsible for motion encoding. ConvLSTM can propagate spatial characteristics temporally through each ConvLSTM state. Thus, it is preferred over the FC-LSTM.

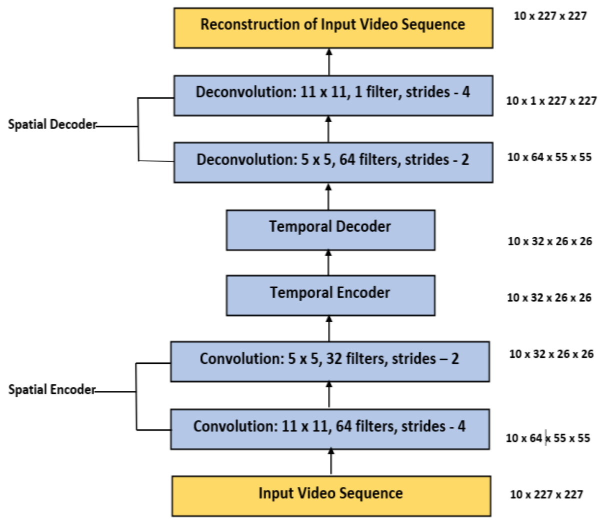
Reconstruction of Input Video Sequence
10 x 227 x 227
Deconvolution: 11 x 11, 1 filter, strides - 4
10 x 1 x 227 x 227
Spatial Decoder
Deconvolution: 5 x 5, 64 filters, strides - 2
10 x 64 x 55 x 55
Temporal Decoder
10 x 32 x 26 x 26
Temporal Encoder
10 x 32 x 26 x 26
Convolution: 5 x 5, 32 filters, strides – 2
10 x 32 x 26 x 26
Spatial Encoder
Convolution: 11 x 11, 64 filters, strides - 4
10 x 64 x 55 x 55
Input Video Sequence
10 x 227 x 227

Figure 2. Spatial Auto-Encoder

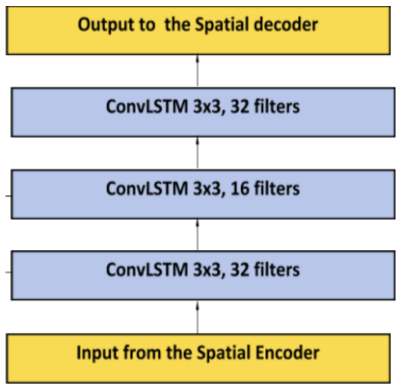

Figure 3. Temporal Encoder-Decoder

2.2.3 THRESHOLDING AND ERROR

The performance of the trained model is evaluated by observing its performance on the unseen test videos with abnormal events. When an unseen video is given as the input model delivers an anomaly score per bunch of 10 frames. This anomaly score is calculated by taking the Euclidean distance between the input frame and the reconstructed frame. The anomaly score is then compared to the model specific threshold value which is the mean of reconstruction errors on all the training data. This threshold value can be altered to make the model more sensitive to the anomalies in terms of the standard deviation of the training error. The video bunch is categorized as a bunch with anomaly if its anomaly score is higher than the threshold calculated.

3. EXPERIMENTS

Since the project deals with a large volume of data, Google Drive is used as a repository where the manually downloaded and edited videos were uploaded. The drive was mounted, and the videos were read from the drive during implementation. Google Colab Pro version is used for writing Python code implementation for the model design and training. Since the data was huge and the RAM storage was running out often while preprocessing and model training, Colab sessions were crashing. This hindered uninterrupted training for a longer time and Colab Pro version was used in place of the Google Colab Free Version which offered a high GPU memory and High-RAM runtime environment with lenient idle time too. A simple web page was built using Python Flask APIs and the trained model to display the video frames labeled as normal and anomalous frames.

3.1.1 HARDWARE

- CPU - Colab (Pro) --- Intel(R) Xeon(R) CPU @ 2.30GHz
- GPU - Colab (Pro) ---Tesla P100-PCIE-16GB
- RAM - Colab(Pro) --- 26GB High RAM

3.1.2 SOFTWARE

- Programming Language --- Python
- Libraries to handle videos --- FFmpeg, openCV
- Frameworks --- Tensorflow, Flask
- Video Downloader --- 4K Video Downloader
- Video Editor --- Video Editor by Windows10

3.2 DATASET

The model is trained on car race videos scraped from YouTube mainly consisting of IndyCar race events. The videos were manually edited specially to obtain sequences with reasonable duration of abnormal events within the long footage. Training data consists of 50 video clips where the duration of each clip varies between less than a minute and two minutes long and test data consists of 10 video clips. Each video clip contains about 600 frames making up data of about 6GB. All videos are taken from a fixed camera angle for both training and testing sets, where the car moves parallel to the camera plane i.e, visor cam view. The reason for choosing this is to make sure the camera angle is maintained uniformly in all videos across the dataset. All training videos contain only normal events where the car moves in forward direction on the racetrack without any interference, whereas testing videos contain both normal and abnormal events like cars crashing on to each other, or cars crashing on some other stationary objects on the sides, or when some human comes into the frame during pit stops.

Dataset Drive Link:
https://drive.google.com/drive/folders/1YORUAkGg_rlVqh6gZWDnnSPqH6H7KzPS?usp=sharing

3.3 RESULTS

The implementation is experimented with two different model architectures and the corresponding performances are recorded for three different levels of dropouts to arrive at the design decisions. Each training volume is trained as a single batch for a maximum of 150 epochs or until the reconstruction loss of validation data stops converging for 3 consecutive epochs.

AUC - ROC score is used as a performance measurement here since it is used for classification problems at various threshold settings [2]. ROC is a probability curve and AUC represent degree or measure of separability. The performance analysis of the two models is shown in Table 1.

Dropout	Model 1: AUC %	Model 2: AUC %
No Dropout	54	52.6
Dropout – Variation 1 FC Dropout - 0.2 Recurrent Dropout – 0.1	63.5	68
Dropout – Variation 2 FC Dropout - 0.4 Recurrent Dropout – 0.3	62	65.4

Table 1. Comparison of area under ROC curve (AUC) for the two models over three dropout variations

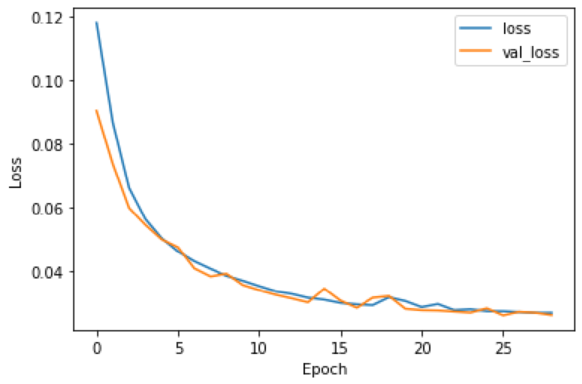

Figure 4. Model 2 Dropout Variation 2 Convergence

Based on the model convergence and the performance metrics as shown in Figure 4, Model 2 with dropout variation 2 was chosen as optimal for the project.

3.3.1 PERFORMANCE AND ANALYSIS

Input1: Crash Video

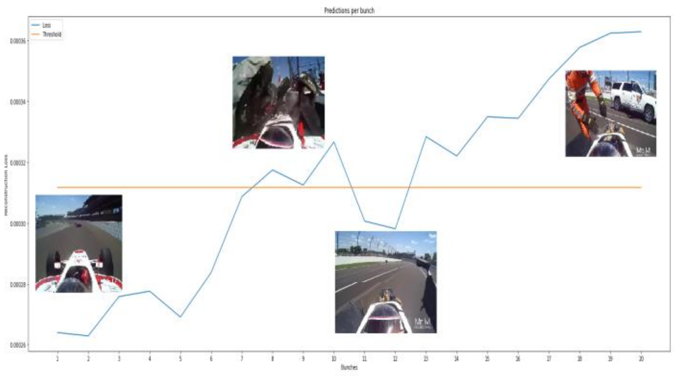

Figure 5. Abnormality score per bunch

When the performance of the model for video input with multiple anomalies like crash and people on the racetrack was observed, it was evident that the anomaly score stays below the threshold for the normal case (no crash) and spikes up over the threshold value before the collision. The model predicts events like off road frames or cars abnormally closer to each other which represent an anomaly and a possible collision. In the above Figure 5, the anomaly score is the highest when a person comes onto the track as it is a rarer event compared to a car crash to occur in a race. It is also observed that the model tags event as no anomaly when the car spins back on to the road amidst the anomalies.

Input2: No Crash Video

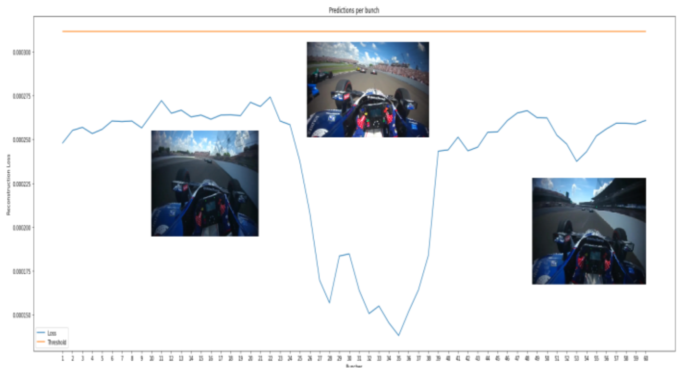

Figure 6. Abnormality score per bunch

If a no crash video as in Figure 6 is given as the input, the anomaly score stays well below the threshold, but the score varies slightly based on the lighting conditions in the video frames. The model being trained predominantly on videos with better lighting conditions might give higher anomaly scores owing to the bad lighting conditions.

4. CONCLUSION

Deep learning approach is successfully applied to a challenging video anomaly detection problem. The problem is formulated as a spatiotemporal sequence outlier detection problem and a combination of spatial feature extractor and temporal sequencer ConvLSTM is applied to tackle the problem. The ConvLSTM layer not only preserves the advantages of FC-LSTM but is also suitable for spatiotemporal data due to its inherent convolutional structure. By incorporating convolutional feature extractor in both spatial and temporal space into the encoding-decoding structure, an end-to-end trainable model is for video anomaly detection. The advantage of the proposed model is that it is not supervised – the only ingredient required is a long video segment containing only normal events in a fixed view.

The primary challenge is data collection for both training and test datasets as there are very few IndyCar race videos from the visor cam view. Since the videos are scraped from multiple sources, the quality of the footage varies for each video and that affects the model learning. Convolutional LSTM layers applied in the model are memory-intensive and so the training will need to be executed on very small mini batches, which results in slow training time of about 10 hours for fairly 25 epochs.

The proposed method is limited in implementation since video footage of abnormal events like crashes, pit stops, and damages are difficult to obtain from the IndyCar race videos online. Even so, the recorded footage from the desired camera angle is likely not long enough to capture all types of abnormal activities in the scene. The model is sensitive to outliers in the video like low light conditions and quality of the video. The experiment does not consider the information like the relative location of the cars on the track and the distance from each other.

The size of training data can be increased by performing data augmentation in the temporal dimension. Frames with stride-1, stride-2, and stride-3 can be concatenated to generate more volume of training data. Generative Adversarial Networks can be used to generate more sequential images by setting different stride values. The model could be extended to include telemetry data from the Indycar Series, the position and GPS location of the cars on the track during the event which might give better results in detecting anomalies along with the video data.

5. ACKNOWLEDGMENT

We acknowledge the support of our Professor Dr. Judy Qiu and our mentor Selahattin Akkas for providing us enough motivation, encouragement, and guidance throughout the term to complete the term project. We would like to thank them for their timely support for sharing relevant research papers, past projects, and scripts to help us in the execution of our idea through every step of the project.

REFERENCES

[1] Yong Shean Chong, Yong Haur Tay, "Abnormal Event Detection in Videos using Spatiotemporal Autoencoder", *International Symposium on Neural Networks*, 2017.

[2] Wen Liu, Weixin Luo, Dongze Lian, Shenghua Gao, "Future Frame Prediction for Anomaly Detection – A New Baseline", *IEEE International Conference on Multimedia and Expo (ICME)*, 2017.

[3] Zhou, S., Shen, W., Zeng, D., Fang, M., Wei, Y., Zhang, Z, "Spatial temporal convolutional neural networks for anomaly detection and localization in crowded scenes. Signal Processing: Image Communication" 47,358–368, 2016.

[4] Nitish Srivastava, Elman Mansimov, Ruslan Salakhutdinov, "Unsupervised Learning of Video Representations using LSTMs", *International conference on machine learning*, pp. 843–852, 2015.

[5] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M," Learning spatiotemporal features with 3d convolutional networks", *IEEE International Conference on Computer Vision (ICCV)*, pp. 4489–4497, 2015.

[6] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei,L, "Large-scale video classification with convolutional neural networks", *IEEE Conference on Computer Vision and Pattern Recognition*, pp.1725–1732, 2014.

[7] https://www.tensorflow.org/tutorials/generative/cvae

[8]https://www.tensorflow.org/tutorials/structured_data/time_series