DA526 Project Report

Intelligent Video Surveillance System using DeepLearning

Team 13:
Akshay Vitthal Parit (224161013)
Koppula Ajay Kumar Reddy (224161017)
Sadem Vasanth Kumar (224161010)

• Problem Statement:

The use of video surveillance systems has become an essential tool in ensuring the safety and security of public and private spaces. However, relying on human operators to continuously monitor the footage from these systems can be labor-intensive and monotonous, leading to decreased attention and performance. This can result in missed events, delayed response times, and even the potential for critical security breaches.

Furthermore, as video surveillance systems continue to increase in popularity and sophistication, the amount of data generated by these systems is growing exponentially. This creates a significant challenge in terms of managing and processing this data in a timely and efficient manner.

Therefore, there is a pressing need for automated solutions that can assist human operators in monitoring and analyzing the vast amounts of video data generated by surveillance systems. Such solutions can help to reduce the risk of security breaches, improve response times, and enhance the overall efficiency of surveillance operations.

• Data Set:

Avenue Dataset contains 16 training and 21 testing video clips. The videos are captured in CUHK campus avenue with 30652 (15328 training, 15324 testing) frames in total. Our dataset contains the following challenges.

- Slight camera shake (in testing video 2, frame 1051 - 1100) presents.
- A few outliers are included in training data.
- Some normal patterns seldom appear in

training data.The following link contains the

dataset link to download.

- Avenue Dataset (16 training videos, 21 testing videos) [4]

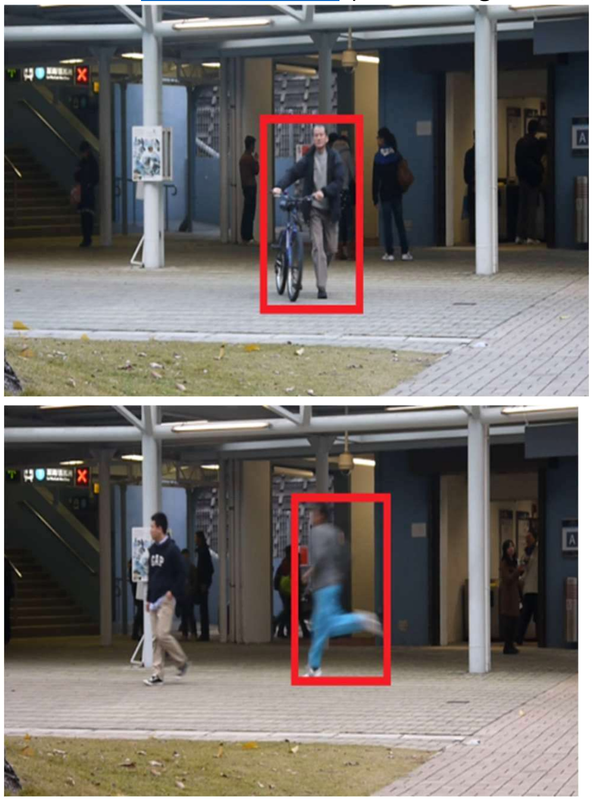


Figure 1: Few anomaly frames from our dataset

• Related Work:

Kartz and colleagues [1] proposed an HMM-based technique for anomaly detection that employed 3D Gaussian distributions of spatio-temporal gradients. However, the use of manually engineered features proved to be insufficient in addressing the complexities of video surveillance scenes, as these features had limitations in their ability to accurately represent the data.

Deep learning techniques have demonstrated significant advantages in learning features and have been highly effective in various computer vision tasks, including anomaly detection. In their work, Xu et al. [2] introduced a stacked autoencoder that automatically learns feature representations. They utilized one-class SVM models to predict anomaly scores. However, this approach extracted local image patches and flattened them into 1D vectors, disregarding spatial information. Additionally, the temporal information was provided by optical flow patches, which only captured the motion features between two frames.

On the other hand, Hasan et al. [3] proposed a fully-convolutional autoencoder to learn spatio- temporal regularities. Although their model took multiple frames as input, it completely collapsed the temporal information during the spatial convolution operations. To address the limitations of existing deep learning methods, this paper introduces a novel model that incorporates 3D convolution operations to extract features from both spatial and temporal dimensions. By leveraging this approach, the proposed model overcomes the drawbacks of previous deep learning-based approaches in anomaly detection

• Methodology:

The project is divided into small modules which carry out a part of the work in the project, and the modules are namely

- Input video to Image frames
- Pre-Processing
- Feature learning
- Reconstruction Error
- Thresholding and predicting the anomaly in the event.

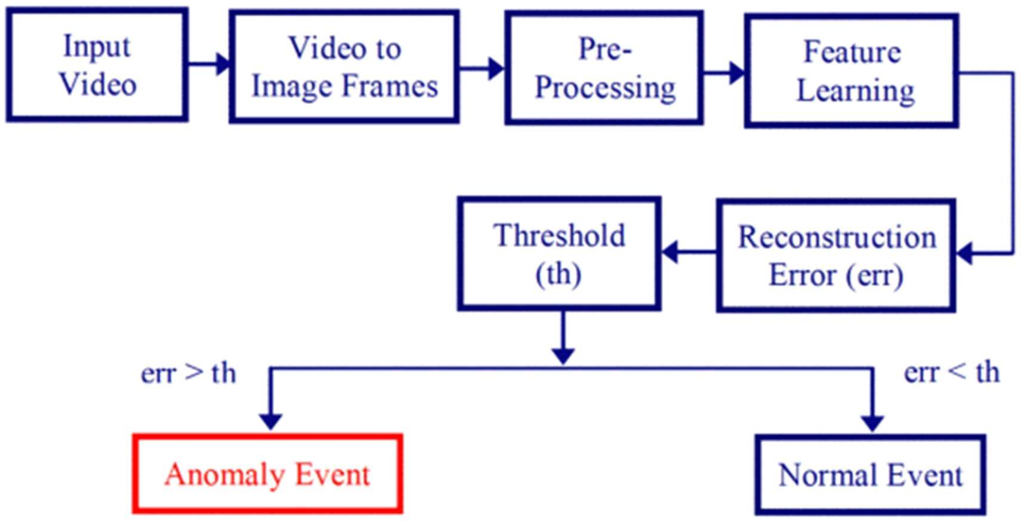


Figure 1: Flow diagram of Methodology

- ## Input video to Image frames:

The proposed system uses input as video. Experiments are performed on .mp4 and .avi formats. The required video processing is carried out in further stages of the system.

One of the techniques used for data augmentation involves creating videos that contain objects intended for training in deep learning applications. These videos are subsequently processed to extract frames, treating each frame as an individual image. The extraction process often involves the utilization of FFMPEG, an open-source transcoding tool**.**

- **FFmpeg software:**

FFmpeg (version 4.4) which is used in this project is a powerful open-source software suite designed for handling multimedia data. It provides a wide range of tools and libraries for transcoding, encoding, decoding, streaming, and manipulating audio and video files.

One of the key features of FFmpeg is its ability to extract frames from videos. With the ‘ffmpeg’ command-line tool, users can easily extract individual frames from a video file and save them as images. This functionality is particularly useful for tasks such as data augmentation in deep learning applications.

FFmpeg offers numerous additional features and options for advanced video processing and manipulation. It supports a wide range of video and audio codecs, containers, and formats, making it a versatile tool for multimedia operations. Its extensive documentation and active user community further contribute to its popularity and usefulness in various applications.

- **Preprocessing:**

In the preprocessing stage, raw image data needs to be converted into a suitable format that can be fed into the model. The following steps are typically carried out during this stage:

- **Resizing and Scaling**: Each extracted frame is resized to dimensions of 227× 277 pixels from original 640x360. This step ensures that all input image frames have the same size, which is often required by the model architecture.
- **Normalization:** To ensure consistency and comparability, all pixel values in the resized frames are scaled between 0 and 1. This normalization step helps to bring the pixel values within a standardized range. It is important to note that the specific values and equations used for normalization may vary depending on the requirements of the model and the nature of the dataset. These preprocessing steps play a vital role in preparing the input data for the model, ensuring that it is in an appropriate format and facilitating effective training and inference.
- Therefore, Processed frames undergo normalization to have unit variance and zero mean.
- Image frames are converted into gray scale to alleviate dimensionality.

- **Feature Learning:**

In the proposed architecture, a convolutional spatial and temporal autoencoder is employed to learn normal patterns from the training dataset. The architecture consists of two main components:

- **Spatial Autoencoder**: This component is responsible for learning the spatial structure of each video frame. It utilizes convolutional layers to encode the spatial information of the frames into a compressed representation. The encoded representation is then decoded back into the original spatial structure using transposed convolutional layers.

- **Temporal Encoder and Decoder**: This component focuses on learning the temporal patterns within the spatially encoded structures. The temporal encoder takes the encoded representations from the spatial autoencoder and captures the temporal dependencies between consecutive frames. It extracts temporal features that represent the temporal evolution of the spatial structures. The temporal decoder then reconstructs the spatially encoded structures from the temporal features, enabling the model to generate reconstructed frames.

By combining the spatial autoencoder and the temporal encoder/decoder, the proposed architecture can effectively learn both the spatial and temporal characteristics of the input video data. This allows it to capture and understand normal patterns within the training dataset, facilitating subsequent anomaly detection and classification tasks.

An autoencoder is a type of artificial neural network that combines an encoder and a decoder. It is commonly used for unsupervised learning to efficiently encode data. Autoencoders are

particularly useful for dimensionality reduction tasks. The encoder component reduces the dimensionality of the input, and the decoder aims to reconstruct the original input from the encoded representation. The model is trained using backpropagation to minimize the reconstruction error between the decoded output and the original input.

In the context of the described architecture, convolutional layers are used to extract features from the input image frames of the video. Convolutional operations help preserve the spatial relationships between pixels by learning image features using small squares of input data. In image processing, a convolution operation involves taking the dot product between a filter anda local section of the input image frame

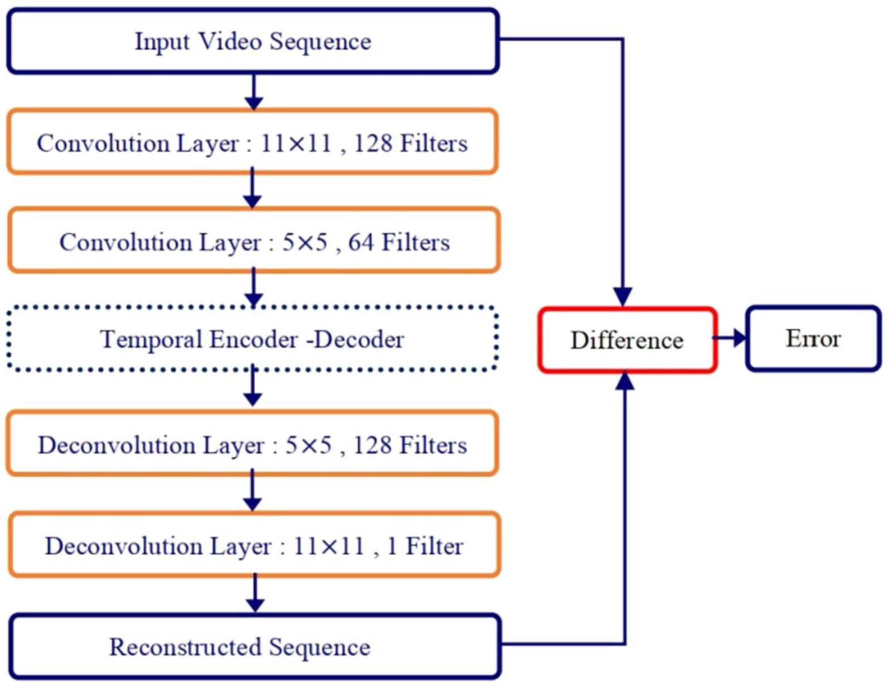
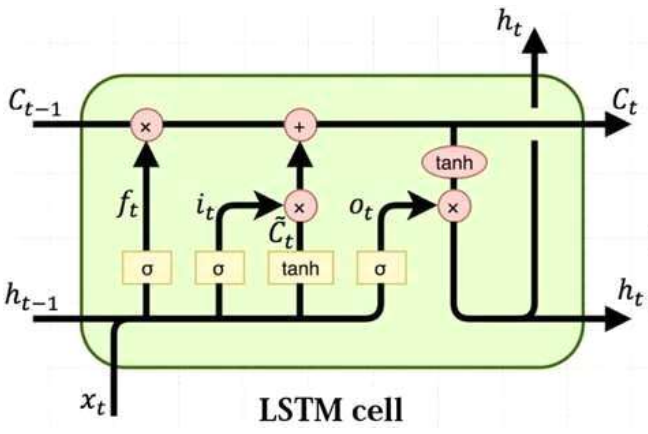

Input Video Sequence
Convolution Layer : 11×11 , 128 Filters
Convolution Layer : 5×5 , 64 Filters
Temporal Encoder -Decoder
Deconvolution Layer : 5×5 , 128 Filters
Deconvolution Layer : 11×11 , 1 Filter
Reconstructed Sequence
Difference
Error

Figure 2: Architecture of Proposed System

In the spatial encoder component (as shown in above Figure), two convolutional layers are employed. The first layer consists of 128 filters with a size of 11×11, while the second layer has 64 filters with a size of 5×5. These layers are responsible for encoding the input video frames and extracting relevant features. In the spatial decoding part, two deconvolutional layers are used to reconstruct the input video sequence. The first layer consists of 128 filters with a size of 5×5, and the second layer utilizes a single filter with a size of 11×11 to perform the decoding process

In this architecture, three convolutional LSTM layers are utilized. Each convolutional LSTM layer has a size of 3×3, with the central layer having 32 filters and the outer layers having 64 filters. The structure of a typical LSTM is illustrated below. A typical convolutional LSTM call for temporal learning will be as shown in the below figure borrowed from internet source.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Figure 3: A Typical LSTM cell

In these equations from the above figure 3, the input gate (i_t), forget gate (f_t), and output gate (o_t) control the flow of information in the LSTM unit. The sigmoid function (σ ()) is applied element-wise to the weighted sum of the previous hidden state (h_{t-1}), current input (x_t), and the previous cell state ($\tilde{C}_{t-1}$). The tanh () function is applied to the weighted sum of the previous hidden state and the current input to compute the current cell state (C_t). The final hidden state (h_t) is obtained by element-wise multiplication of the output gate with the hyperbolic tangent of the cell state.

These equations describe the computations involved in the convolutional LSTM, which enable it to capture temporal dependencies within the spatially encoded structures and learn temporal patterns in the input video data.

- **Thresholding:**

The determination of whether an event is detected as an anomaly or normal is based on a threshold. The reconstruction error, which measures the dissimilarity between the input data and the reconstructed output, is compared to an optimum threshold.

If the reconstruction error is greater than or equal to the optimum threshold, it is classified as an anomaly event. On the other hand, if the reconstruction error is less than the optimum threshold,it is classified as a normal event.

In summary:

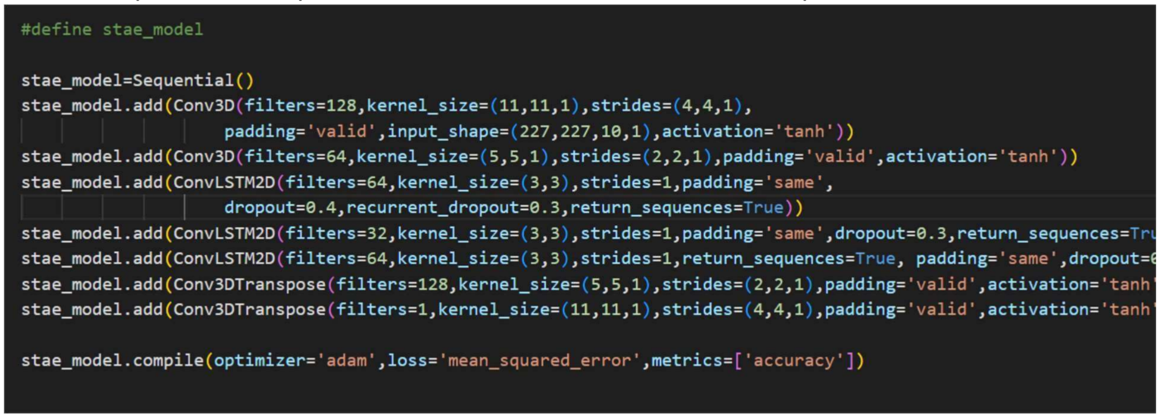
- Anomaly Event: Reconstruction error $\geq$ Optimum Threshold
- Normal Event: Reconstruction error < Optimum Threshold
- This threshold-based approach allows for the differentiation between normal and anomalous events based on the level of dissimilarity between the input and reconstructed data.

• Experiments and Results

• Modelling:

This project utilizes several libraries for various purposes. Firstly, TensorFlow's Keras module is used, providing functions like `*img_to_array*` and `*load_img*` for image processing tasks. NumPy is used for numerical computations and array manipulation. The `glob` module is used to retrieve file paths based on specific patterns, while the `*os*` module allows interaction with the operating system for file and directory operations. OpenCV (`cv2`) is employed for image processing and computer vision tasks. Keras provides layers such as `*Conv3D*`, `*ConvLSTM2D*`, and
`*Conv3DTranspose*` for building the architecture. The `*Sequential*` model is used to stack layers, and callbacks like `*ModelCheckpoint*` and `*EarlyStopping*` monitor and control the training process. Lastly, `*imutils*` offers utility functions for image and video processing tasks in OpenCV and Python.

The following code snippet shows the modelling of the auto encoder i.e reconstruction from the actual image using afore mentioned libraries which is used to compare mean squared loss from the actual frames and optimizer such as adam.



```
#define stae_model

stae_model=Sequential()
stae_model.add(Conv3D(filters=128,kernel_size=(11,11,1),strides=(4,4,1),
                    padding='valid',input_shape=(227,227,10,1),activation='tanh'))
stae_model.add(Conv3D(filters=64,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'))
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,padding='same',
                    dropout=0.4,recurrent_dropout=0.3,return_sequences=True))
stae_model.add(ConvLSTM2D(filters=32,kernel_size=(3,3),strides=1,padding='same',dropout=0.3,return_sequences=Tru
stae_model.add(ConvLSTM2D(filters=64,kernel_size=(3,3),strides=1,return_sequences=True, padding='same',dropout=0
stae_model.add(Conv3DTranspose(filters=128,kernel_size=(5,5,1),strides=(2,2,1),padding='valid',activation='tanh'
stae_model.add(Conv3DTranspose(filters=1,kernel_size=(11,11,1),strides=(4,4,1),padding='valid',activation='tanh'

stae_model.compile(optimizer='adam',loss='mean_squared_error',metrics=['accuracy'])
```

Figure 4: Code snippet of the Stae Model

During testing of the model, we use the same preprocessing steps used for training images and take the first 10 frames and they are tested for different trained models with different epochs and batch sizes. The model detects the anomalies based on the threshold value.

- **Results:**

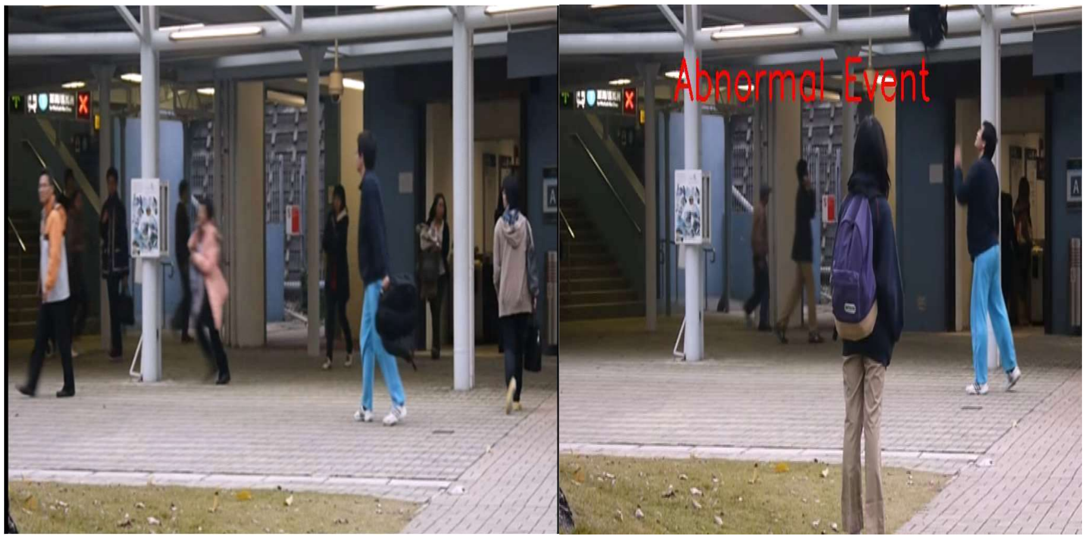
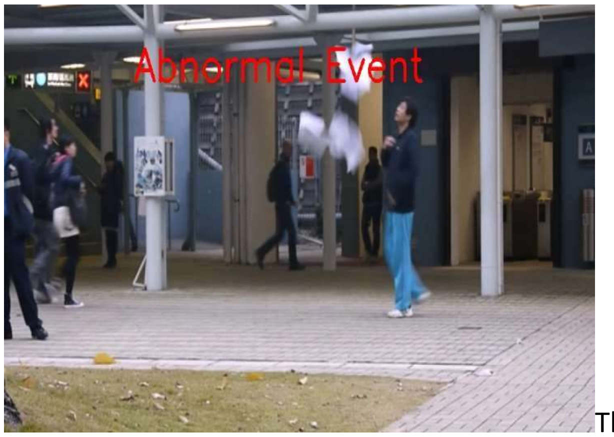


Figure 5: Left shows normal event and right shows abnormal event frames in same video



The above image shows two different frames from the same video right frame corresponding to abnormal event and left is as normal as the man in left frame is still having the bag in his hand and no abnormal thing has been detected, once the man throws bag in the air there is increase in mean squared error as the actual and reconstructed frame will differ a lot from each other.

Figure 6: Figure showing a man throwingpapers in the avenue.

This image shows the abnormal event as papers in the frame occupy the space and the reconstructed image will be in contrast to the actual event.

• Conclusion

In this project, deep learning techniques have been successfully applied to detect anomalies in videos from standard benchmarks. The task of anomaly detection is framed as a binary classification problem, where video frames are transformed into spatiotemporal sequences and outliers are identified. The approach leverages appropriate convolutional layers to extract spatial features and convLSTM layers to capture temporal patterns. By using convolutional operations, the model effectively extracts relevant features from the spatiotemporal data. Notably, this method does not require manual labeling and can be considered a partially supervised approach.

However, despite its ability to detect anomalies, the proposed approach may generate false alarms due to the complexity of activities in the scenes. This limits its suitability for real-time applications. Additionally, the inclusion of LSTM layers in the model increases the training time and necessitates a large database, which can be challenging to acquire. For future work, incorporating a supervised approach could be explored to improve performance and address these limitations.

References:

- Louis Kratz and Ko Nishino. 2009. Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models. In Computer Vision and Pattern Recognition, 2009.
CVPR 2009. IEEE Conference on. IEEE, 1446–1453

- Dan Xu, Elisa Ricci, Yan Yan, Jingkuan Song, and Nicu Sebe. 2015. Learning deep representations of appearance and motion for anomalous event detection. arXiv preprint arXiv:1510.01553 (2015).

- Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K Roy-Chowdhury, and Larry S Davis. 2016. Learning temporal regularity in video sequences. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 733–742

- http://www.cse.cuhk.edu.hk/leojia/projects/detectabnormal/dataset.html