LSTM based Predictive Network for Video Anomaly Detection

MSc Research Project

Ravindra Ardhapure
Student ID: x21189838

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ravindra Ardhapure		
Student ID:	x21189838		
Programme:	Masters in Cloud computing	**Year:**	2022-2023
Module:	MSc in cloud computing		
Supervisor:	Vikas Sahni		
Submission Due Date:	14/08/2023		
Project Title:	LSTM based Predictive Network for Video Anomaly Detection		
Word Count:	**6207 Page Count: 20**		

Signature: Ravindra Ardhapure

Date: 10/08/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

LSTM based Predictive Network for Video Anomaly Detection

Ravindra Ardhapure
x21189838

Abstract

Increasing use of video surveillance has necessitated the automation of anomaly detection to provide enhanced security coverage at a lower cost due to its ubiquitous presence everywhere from large corporations to home surveillance. However, when building one such anomaly detection model, due to the absence of unexpected occurrences during training, unsupervised learning techniques have become the norm. In this study, object-level self-supervised and multitask learning is used to detect anomalous events in the video frames with the implementation of a predictive neural network framework integrated into recurrent neural networks (RNN) that learns to ascertain future data frames, allowing local predictions in each layer while only the differences between each output are passed to the corresponding network layers and this reconstruction capability can be used to make predictions. The suggested approach is flexible and can be implemented into several cutting-edge anomaly detection algorithms. To give empirical evidence of significant performance increases, the predictive coding block was integrated with long-short term memory (LSTM) RNN networks to build a prediction model that is capable of detecting anomalies and such a model can be deployed to work with any anomaly detection system in real-time with a fewer modification. The evaluation results were presented with a training accuracy of about 96.4% and 76% test accuracy and 74% test precision which asserts the model performance.

Keywords: anomaly detection, predictive block, video surveillance, deep learning, Azure ML

1 Introduction

Anomaly detection is becoming more and more common due to its use in video surveillance and the rising need for social security. Anomaly detection seeks to identify instances containing anomalous and defective patterns distinct from those observed in typical instances. Real-world applications contain an infinite number of abnormal events, making it nearly impossible to collect all types of abnormal events and apply a classification method to the problem. Since the distribution of anomaly patterns is unknown in advance, the models can be trained to learn the patterns of normal instances and identify anomaly if the test instance is not adequately represented by these models. For instance, when the data reconstruction error is high, an autoencoder that has been trained to reconstruct normal data is used to declare anomalies.

A distinct subcategory of reconstruction methods is prediction of masked information, using the reconstruction error regarding the masked information as an abnormality score (Ye et al. 2020). Methods within this subcategory conceal a portion of the input and use a deep neural network to predict the absent input data. The goal of reconstruction techniques is to reduce the reconstruction errors of training data, but these techniques cannot guarantee tiny reconstruction errors for anomalous events. Future frame prediction techniques are based on the premise that while regular events are predictable, aberrant ones are not (Lu et al., 2013).
In contrast to these methods, it was proposed to present a predictive coding network model that can make local predictions of the input before deducting them from the actual input and passing the results to the following layer and has the flexibility to be integrated into a variety of neural architectures (Ristea et al. 2022). The proposed prediction block is a self-supervised predictive block consisting of a convolutional layer that includes a loss function that minimizes the reconstruction error between the final activation maps and the input data. In other terms, the block is self-supervised trained to deduce the required information necessary to differentiate between normal and abnormal video frames for anomaly detection.

1.1 Motivation

Atypical events that deviate from the learnt nominal patterns are referred to as anomalies in the literature on anomaly detection in video surveillance. However, assuming the availability of training data that considers all conceivable nominal patterns/events is unrealistic for actual implementations. It can be difficult to tell anomalous events apart from nominal ones since they sometimes take the form of contextual events. Therefore, a practical framework should have the ability to continuously update its definition of nominal occurrences. Additionally, the sequential pattern of video anomaly recognition and the significance of online decision-making are not adequately addressed, which impels to apply the predictive network model in existing neural network architectures.

1.2 Research Problem

Although it is a difficult topic that is now the focus of much research, the automation of recognizing anomalous occurrences in videos offers numerous applications in a variety of industry sectors. In order predict future frames in a video, one prominent method involves training deep recurrent neural networks to gain a thorough grasp of the physical and causal rules in the observed situations. When the model fails to predict the pixel values in the subsequent video frame, indicating an important error in the model's prediction results, it is then feasible to determine if some anomaly happened in the video. This technique can be applied both supervised and unsupervised, making it possible to detect both specified anomalies and anomalous events which have never happened.

1.3 Research Question

Q: How does the predictive coding block-based LSTM model perform in terms of accuracy or other metrics and what advantage does it offer in its implementation?

1.4 Research Objective

The primary objective of this research is implementing an architecture based on predictive coding blocks integrated into the LSTM-RNN framework to build a predictive model capable of predicting future frames in a video sequence and the deployment of the model on AML cloud using pipelines.

1.5 Research Contribution

1) A predictive convolutional network integrated to the LSTM based DL framework was proposed that combines the benefits of prediction and reconstruction with predictive-coding network.
2) The model has the capacity to accumulate information continuously to produce precise predictions of future frames.
3) The model can be trained on UCSD anomaly dataset that consists of video frames of pedestrians' walkways where the pedestrian presence is a normal event and other events like bike, carts, animals can be considered as anomalies.
4) An evaluation methodology is provided in terms of model training accuracy, validation loss, test accuracy and precision that captures the model's performance.

1.6 Thesis structure

This research report is organized into five sections:
Section – 1 introduces to the research background and what motivated to consider it for further investigation. It also presents the research problem and answers the research question.
Section – 2 elaborates on the previous research presented over the years by classifying them into subcategories based on their scope, method and relevancy to the proposed approach.
Section – 3 presents the research method in detail to offer more clarity on the research subject.
Section – 4 provides the implementation details of the proposed techniques detailing the tools and data sources used as well as how the performance evaluation of the results will be done.
Section – 5 offers a comprehensive analysis of the results and main findings of the study as well as the implications of these findings.
Section – 6 concludes the research work summarizing the approaches used, their implementation, and the results achieved with scope for further extending the research.

2 Related Work

Over the past few decades, a number of studies dealing with the issue of anomalous event detection have been done and are categorized as distance-based, change-detection, dictionary-based, probability-based, and reconstruction-based models. Despite the significant progress made by deep-learning techniques in a number of other fields, there have only been a limited number of deep-learning algorithms described for video anomaly identification. Therefore, a thorough analysis of all available deep-learning-based video anomaly detection techniques, from basic to advanced levels, is necessary.

2.1 Conventional Anomaly Detection

Zhao et al. (2011) developed a lexicon-based principled convex optimization framework that jointly infers and updates a sparse reconstruction code and an online vocabulary. This formulation was founded on the notion that an event dictionary can recreate conventional video events but not unexpected ones. As the application collects data, the base vocabulary was updated online, preventing concept drift. By considering the relationship between items, (Ren et al.; 2015) expanded dictionary learning methods in which a significant deviation of an earlier event indicates the start of an unusual happening, as per the change-detection framework, that assesses shifts throughout the video frames to find abnormalities. Unmasking or ordinal regression (Pang et al.; 2020) can differentiate anomalies after measuring changes.

Probability-based methods assume anomalies occur in low-probability areas. These methods use normal data to generate the probability density function (PDF) of test samples and evaluate them. To detect and localize anomalous behaviour in crowded scenes, (Li et al. 2014) proposed a joint detector of temporal and spatial anomalies using a centre-surround discriminant saliency detection system to produce spatial saliency scores and a simulation of normal behaviour learned from training data to produce temporal saliency scores. The probability of a field of conditional randomness that maintains global anomaly assessment consistency are the multiscale scores. Rudolph et al. (2020) used normalising flows to estimate the density of convolutional neural network (CNN) features to handle low-dimensional data distributions.

2.2 Deep Learning based Anomaly Detection

Anomaly detection models (Ye et al.; 2020; Hasselmann et al.; 2018) remove details from the input and use neural networks to infer the missing information. (Hasselmann et al.; 2018) framed anomaly detection as an inpainting problem, where portions from images were masked randomly, with surface anomaly identification relying on the pixel-wise reconstruction error of the masked patches. The Attribute Restoration Network (ARNet) (Ye et al.; 2020) uses an attribute erasing module (AEM) to disorient the model by erasing visual features including colour and orientation. ARNet trains to restore normal images and find anomalies.

Recently, reconstruction-based anomaly detection approaches have grown popular. These methods use auto-encoders (Hasan et al.; 2016) and generative adversarial networks (GANs) (Liu et al.; 2018) to learn effective reconstruction manifolds using only typical data. Hasan et al. (2016) used a reconstruction error regularity score to find anomalies. Fully linked and convolutional autoencoders replace sparse coding. Normal and erratic motions were restored by the trained autoencoder. The researchers retrain the computational models on multiple datasets to prove their results were applicable. This approach does not identify frame anomalies. Neural networks may rebuild aberrant frames with little error, making it difficult to identify anomalous from normal sequences. Researchers had improved the latent manifold's architecture and training methods to address this issue.

Adversarial training (Georgescu et al.; 2021) used gradient ascent for out-of-domain pseudo-abnormal measurements and gradient descent for normal data to create a more discriminative manifold for video anomaly identification. The Cloze problem (Lou et al.; 2020) requires finishing a video with missing frames. Georgescu et al. (2021) suggested middle frame masking as a video anomaly detecting auxiliary. Both methods assume an erased frame may be reconstructed with more precision for periodic motion. Luo et al. (2017) used CNN and LSTM to efficiently learn visual and motion properties. Liu et al. (2021) used a conditional VAE to combine optical flow reconstruction error with picture prediction error. Li et al. (2021) predicts video frames from previous frames with the prediction error pointing to abnormalities. Another GAN-based method (Sabokrou et al.; 2018) removed the patches from an image while the discriminator assesses whether they were normal or erratic. Huszar et al. (2023) employed smart networks with 3D CNN to capture spatial and temporal information, modeling dynamic interactions between subjects and objects. Additionally, a pre-trained action recognition model was implemented to detect violence detection efficiency on video surveillance systems.

2.3 Predictive Networks for Anomaly Detection

Lotter et al. (2016) proposed that a predictive neural network was designed, with each network layer also performing local predictions and forwarding only outliers. Decomposing reconstruction into prediction and refinement, (Ye et al.; 2019) proposed a predictive coding network. In this work, they seamlessly combined flow reconstruction and frame prediction, allowing the quality of flow reconstruction to significantly impact frame prediction. A comparation of the proposed predictive convolutional block implemented on LSTM with other existing approaches in the literature.

Author(s)	**Anomaly Detection Mechanism**	**Framework**	**Evaluated on (datasets/applications)**
Hu et. al. (2016)	Scoring	Deep Neural Network + Slow Feature Analysis	Crowd Panic Detection
Hasan et. al. (2016)	Reconstruction	Fully Convolutional 2D Autoencoder	Crowd, Subway
Saburou et. al. (2017)	Classification	Deep CNN + Autoencoder	Crowded Scenes
Medel et al. (2016)	Reconstruction & Future Frame	Convolutional LSTM	Pedestrian Walkways
Zhao et. al. (2017)	Reconstruction & Future Frame	Spatiotemporal Autoencoder	Traffic Surveillance
Saburou et. al. (2018)	Classification	Deep Anomaly	Pedestrian, Subway
Liu et. al. (2018)	Future Frame	Future Frame using U-NET	Walkways
Gong et al. (2019)	Reconstruction	Autoencoder + Attention-based Memory Module	Cyber-security
Ionescu et. al. (2019)	Classification	Object-centric convolutional autoencoders	Pedestrian, Subway
Lotter et al. (2016)	Predictive Coding Networks	Convolutional LSTM	Car-mounted video cameras
Ristea et al. (2022)	Future Frame	Self-supervised Predictive Convolutional Attentive Model	Walkways, Pedestrian
Proposed	Scoring, Future Frame & Classification	Predictive Coding Block based LSTM	Pedestrian Walkways, Video Surveillance

Table 1: Comparison of the various related literature on anomaly detection

Table-1 offers a comprehensive view of the existing research methods, and their specific objectives for anomaly detection. Most research works focused on a single objective like future frame prediction, classification or reconstruction. Also, autoencoders and LSTMs were the preferred choice in building the anomaly detection model due to natural affinity to capturing the finer details of the image/video and extracting the needed features to make the predictions. The proposed work tries to improve on the other existing approaches by combining the predictive coding block with LSTM and evaluated the model performance by performing batch scoring to validate the model on a test dataset as well as use a classifier to determine the accuracy of the prediction on a test dataset. Random Forest (RF) was used as the classifier in this study due to their ability to extract the most important features for binary classification.

3 Research Methodology

This section describes the various parts of how the research has been conducted. The research methodology consists of four steps: data collection, data pre-processing, model training and model test and evaluation.

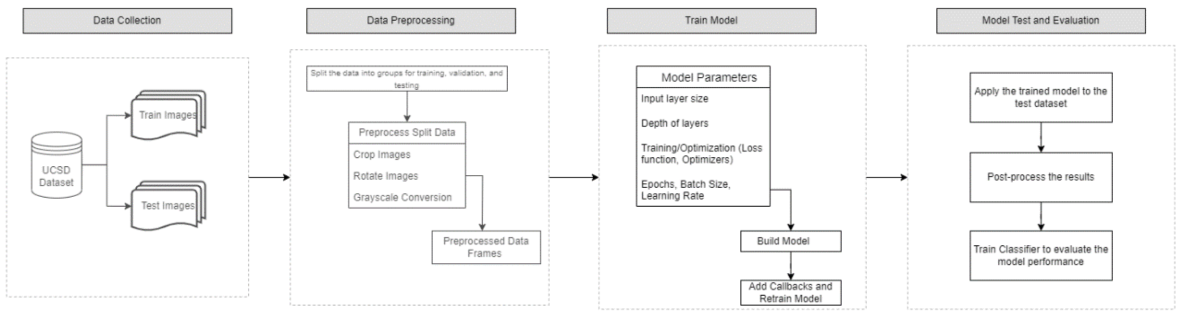
Data Collection
UCSD Dataset
Train Images
Test Images
Data Preprocessing
Split the data into groups for training, validation, and testing
Preprocess Split Data
Crop Images
Rotate Images
Grayscale Conversion
Preprocessed Data Frames
Train Model
Model Parameters
Input layer size
Depth of layers
Training/Optimization (Loss function, Optimizers)
Epochs, Batch Size, Learning Rate
Build Model
Add Callbacks and Retrain Model
Model Test and Evaluation
Apply the trained model to the test dataset
Post-process the results
Train Classifier to evaluate the model performance

Figure 1. Research Methodology

Data Collection: The dataset used in this work is UCSD anomaly detection dataset[1]. This dataset was collected using a stationary camera that was elevated and attached to a wall, gazing down on pedestrian paths. The walkways had varying densities of people, from very few to many. Each scene's video recording was divided into several clips, each with about 200 frames. The ground-truth annotation for each clip contains a binary flag for each frame that identifies whether an anomaly occurs in that frame. Additionally, a subset of 10 videos is given together with manually created pixel-level binary masks that show the locations of abnormalities.

Data Preparation: This step involves splitting the dataset into train, validation and test images in a specified ratio to facilitate the model training process. For learning of normality models, training sets with 34 clips for Pedestrian dataset1 and 16 clips for Pedestrian dataset 2 were supplied. The test set, which consists of 14 test clips for Pedestrian dataset2 and 36 test clips for Pedestrian dataset1, has clips in which one or more anomalies can be seen in one or more frames. The total number of anomalous frames (around 3400) is a bit lower than the total number of normal frames (about 5500). Identifying whether an abnormality is present in each frame of the test set is the challenge at hand. In this work, consider the UCSDPed1 dataset and split them into a group of 6800, 3600 and 3600 images for the train, test and validation data, respectively. Later, these split images from the UCSDPed1 dataset were pre-processed using python libraries which include cropping, rotating for data-augmentation and converting to the grayscale. The image resizing must be done in a way so that it matches the input layer size of the model, as there are constraints regarding the dimensions of the model's input layer. Rotating the image frames helps with data augmentation to perform generalization of the

[1] UCSD Anomaly Detection Dataset http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm

images since they must have been recorded in different angles. The information on the pre-processed images was captured into NumPy arrays of dimensions n images, height, width, depth which will be used for training the model. Another array contains information regarding where the video frames are stored.

Model Design and Training: This step involves creating the proposed predictive network model and training it on the pre-processed dataset. The predictive network model was built upon a stacked convolutional LSTM based on predictive coding principles (Lotter et al; 2016). The model has input layers, 2D convolutional layers, Max Pooling layer, RELU activation layer, sigmoid activation and a fully connected layer. The model is of length, 3 which would mean that it's a three-layer architecture. The number of channels in targets (A) and predictions ($\widehat{A}$) in each layer of the architecture is represented by stack sizes. The second and third layers contain 16 and 32 channels. The convolutional filter sizes were then defined for the target modules. The targets for layers 2 and 3 were computed by a 3x3 error (E) convolution matrix from the layer below, followed by max-pooling. The filter size for the prediction modules has length equal to the length of stack sizes, which would mean that each layer predictions were computed by a 3x3 convolution of the representation (R) modules at each layer. The filter sizes for the representation (R) modules have length equal to length of stack sizes and corresponds to the filter sizes for all convolutions in the LSTM. There were activation functions defined each process in the model: LSTM activation function for the cell and hidden states of the model defined by *tanh*, LSTM inner activation function for the LSTM gates defined by *sigmoid*, error activation for the error units defined by *relu*, and target activation function also defined by *relu*. The model was trained with the hyperparameter setting with 50 epochs, Adam optimizer, mean absolute error as loss function, and ReLU activation function. The model was implemented on TensorFlow and Kera's deep learning framework on Azure ML cloud computing environment. The model was built using the above parameter settings and the model build process involves three steps: *inputs*, a tensor that provides input to the predictive network model, *errors*, the result of applying predictive network to the inputs and *final errors*, errors weighted by layer, *layer_loss_weight.* The model was compiled and fitted using a *Sequence Generator*. Callbacks is a Kera's feature that is utilized in training to run and change the model parameters dynamically. *LearningRateScheduler,* to decrease the learning rate of the model after certain epochs, *Early Stopping,* to stop the training if no improvements were observed on the validation loss and *Model checkpoint,* to save the model and training weights. The model training was evaluated using *validation loss* as the evaluation metric. The trained model was then saved in. Json format and the weights were saved in *.hdf5* format.

Model Testing and Evaluation: The trained model was subsequently utilized on the test dataset, i.e., anomalous videos. The pixel-by-pixel predicted outcome of the model for each video frame was predicted by constructing an array with the exact same structure as the pre-processed video data (n images, height, width, depth). This facilitates to perform statistical analysis on the difference between the ground truth i.e., real video frames and the model's predictions. The mean squared error, and standard deviation of the model's prediction error for each frame were computed and these metrics were saved to a compressed pandas data frame as the final step.

4. Design Specifications

In the scope of this study, the design of a predictive attentive block network was implemented on stacked convolutional LSTM whose goal is to acquire the ability to reconstruct concealed information by utilizing contextual information. To obtain extremely accurate reconstruction results, the block is obligated to examine the global structure of the recently discovered local patterns.

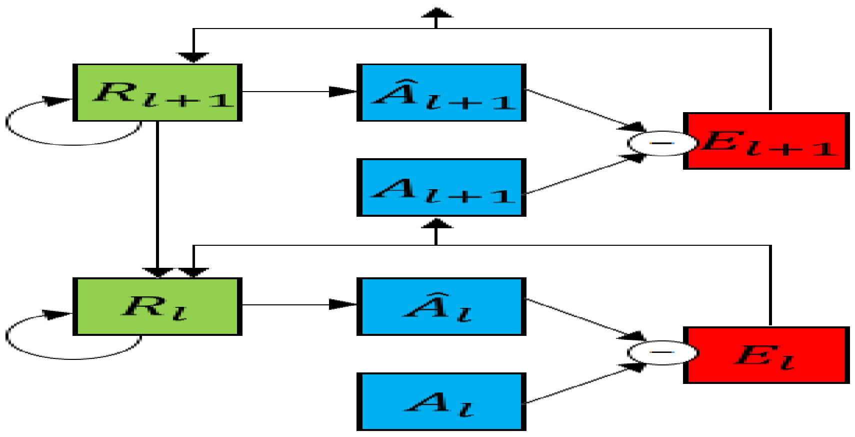

Figure 2: Predictive Attentive Block

.

The network is made up of numerous repeated stacked modules that make local predictions of the input before deducting them from the actual input and passing the results to the following layer as in Figure.1. The input convolutional layer (A_l), recurrent representation layer (R_l), prediction layer ($\hat{A}_l$), and error representation layer (E_l) are the four fundamental components that make up each module of the network. A prediction, Al, of what the layer input, A_l, will be on the next frame is produced by the representation layer, R_l, a recurrent convolutional network. The difference between (A_l) and ($\hat{A}_l$) is fed into the network, which then generates an error representation, (E_l), that is divided into separate rectified positive and negative error populations. After passing through a convolutional layer, the error, (E_l), is then used as the input for the subsequent layer, (A_l+1). Along with top-down input from the representation layer of the network's subsequent level (R_l+1), the recurrent prediction layer R_l also receives a copy of the error signal E_l that can be used to compute the reconstruction loss.

The way the network is set up makes the "right" side of the network, which consists of A_l's and E_l's, equal to a typical deep convolutional network on the first-time step of operation. When compared to a generative deconvolutional network with local recurrence at each stage, the "left" side of the network (the R_ls) is equivalent. Equations (1) through (4) list all update rules in their entirety. To minimize the weighted sum of the activity of the error units, the model is trained. Equation 5 explicitly formalizes the training loss using weighting factors for time (t), layer (l), and number of units (n_l), where n_l is the number of units in the lth layer. The loss at each layer is equivalent to an L1 error when error units were made up of subtraction and ReLU activation (Lotter et al,; 2016).

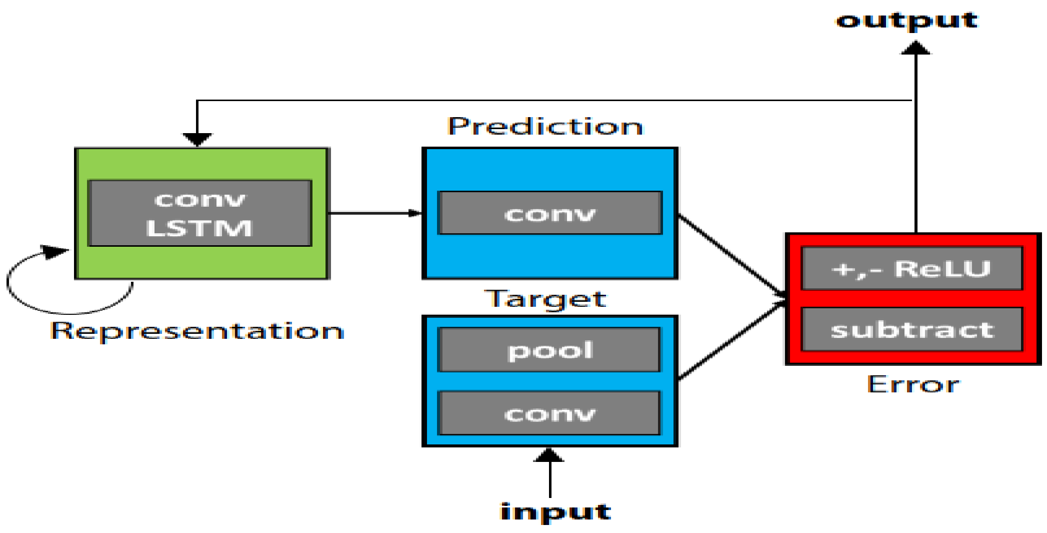

Figure 3: Predictive LSTM model for anomaly detection

$$A_l^t = \begin{cases} x_t & \text{if } l = 0 \\ \text{MaxPooL}\left(\text{RELU}\left(\text{Conv}\left(E_{l-1}^t\right)\right)\right) & l > 0 \end{cases} \quad (1)$$

$$\hat{A}_l^t = \text{ReLU}\left(\text{Conv}(R_l^t)\right) \quad (2)$$

$$E_l^t = \left[\text{ReLU}\left(A_l^t - \hat{A}_l^t\right); \text{ReLU}\left(\hat{A}_l^t - A_l^t\right)\right] \quad (3)$$

$$R_l^t = \text{ConvLSTM}(E_l^{t-1}, R_l^{t-1}, UPSAMPLE(R_{l+1}^t)) \quad (4)$$

$$L_{\text{train}} = \sum_t \lambda_t \sum_l \frac{\lambda_l}{n_l} \sum_{n_l} E_l^t \quad (5)$$

Additionally, the sequence in which each unit in the model is updated must be specified. Two passes were used to update the states: a top-down pass to compute the R_l^t states and a forward pass to compute forecasts, errors, and higher-level targets. The final point worth mentioning is that R_l and E_l were initialized to zero, which, because the network is convolutional, results in an initial prediction that is spatially uniform. The number of Conv-LSTM layers affects the number of possibilities there were for temporal information to be communicated in a model. For instance, the input and the two-prior time-steps were the only sources of information for the first layer's third time-step. The input of each time-step from the second layer may receive information from the previous time-steps of that layer, but the input of the same time-step at the third layer would receive the same. However, there is a limit to how many layers may be added before the data that has been sent through time becomes obsolete i.e., the reconstruction loss will be minimal, but the structural integrity of the data itself could have been compromised. It is also important to select a proper non-linearity function to prevent the issue of vanishing gradients, a suitable loss function to calculate the deviation between the

target and model outputs, and an appropriate update function to learn weights efficiently (Lotter et al.; 2016).

5. Implementation using Azure Pipelines

The Azure ML pipeline implementation on the cloud involves the following steps:

A. DATA PREPARATION

The first step is to get the data into shape for training the model.

1. Split the data into sets for training, validation, and testing. (80-20)
2. Load the individual images, then:
   - Resize the image to match the size of the model.
   - Insert them to a NumPy array which can be used for training (dimensions: images, height, width, depth).
   - Create a second array that contains the folder in which each video frame was stored.

B. MODEL IMPLEMENTATION

This involves the following steps:

- Define the hyperparameters of the model: Size of the input layer, how to sample the input data, Model parameters, Training/Optimization Parameters.
- Decide how to weight prediction errors for each video frame in a sequence and across layers of the model.
- Build the model using the above settings
- Add callbacks that allow you to schedule changes and to monitor progress.
- Train the model

C. BATCH SCORING

- Applying the trained model to the test dataset.
- Configure the output type the model needs to produce, i.e., the output mode is set to *prediction* to make predictions for the next frame.
```
layer_config = train_model.layers[1].get_config()
layer_config['output_mode'] = 'prediction'
```
- Load the weights (weights.hdf5) and model architecture (model.json) to apply it to the test dataset.
- The predictions were evaluated using metrics and saved to '*.pickle'* dataframe.

D. CLASSIFIER

- A random forest classifier model was used to classify the anomalous frames.
- This was treated as a binary classification problem, where the performance metrics were features, and the anomalies were the target label.

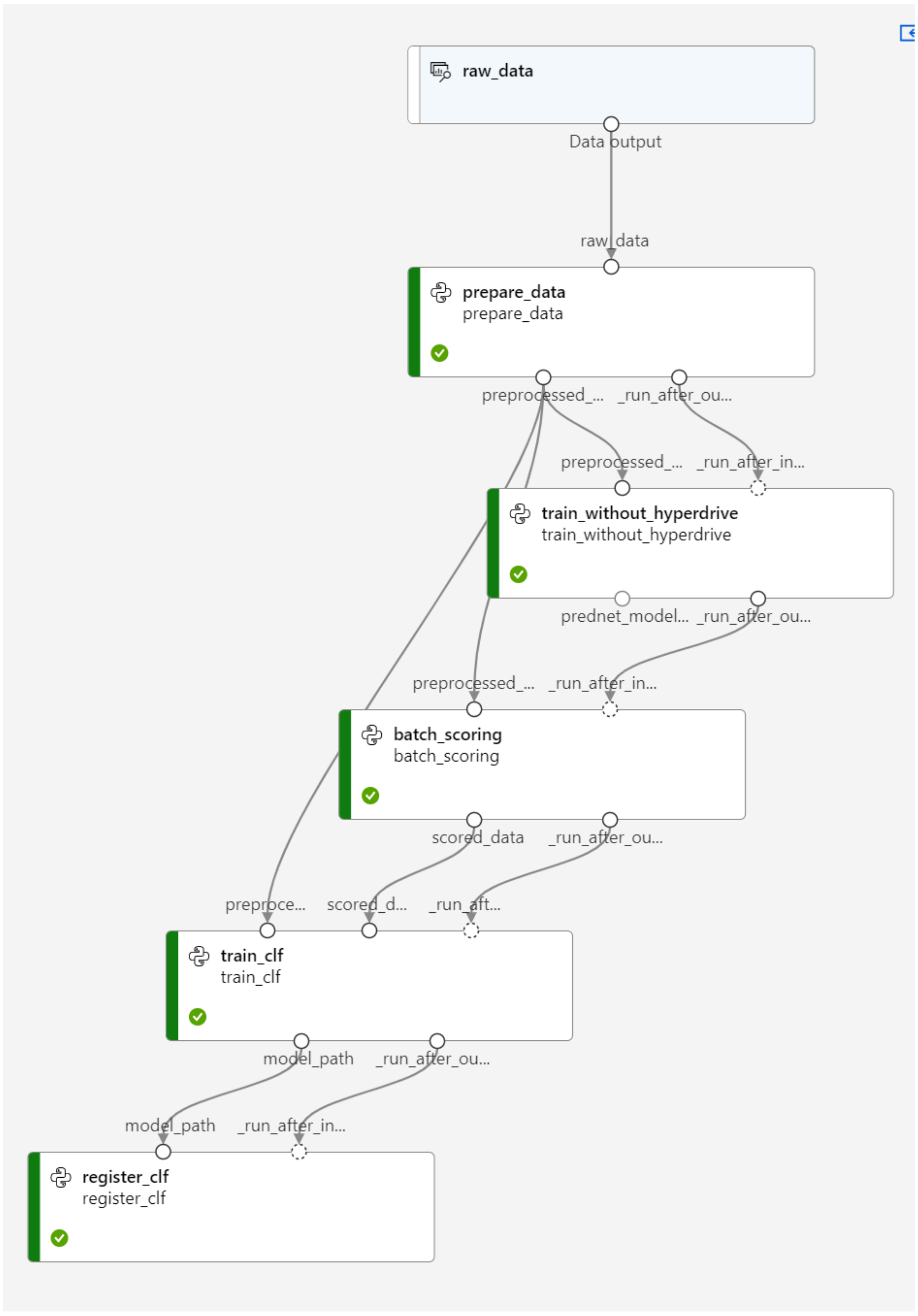

Figure 4: Azure Pipeline implementation of the Anomaly Detection Model

6. Evaluation

The model evaluation was done in three stages:

Model Training – The first step of the evaluation process is to the compile and build the model on the pre-processed UCSD dataset. The training is executed with the model parameters presented in Table 1. The training epochs was set to 10, and 100 epochs each time and the trained model was saved as *model. Json* file and the weights were saved as *weights.hdf5* file. The model training accuracy and validation loss were evaluated to determine the *early stop* criteria of the model. The Plot of validation loss versus training loss for epochs 10, and 100 were presented in Figures 5, and 6. It was inferred that both the training and validation losses decrease with the number of epochs making the model better at predictions. From the validation loss values for epochs 10 and 100 are presented in Table-II, training the model for more epochs significantly reduces the validation loss which in turn can improve the model prediction accuracy

Parameter	Values
Dataset	UCSDped1
Type	Image
Batch Size	4
Filter Size	3,3,3
Freeze Layers	0,1,2,3
Stack Sizes	48, 96, 192
Learning rate (lr)	0.001
Lr_decay	1.0000e-9

Table 2: Model Training Parameters

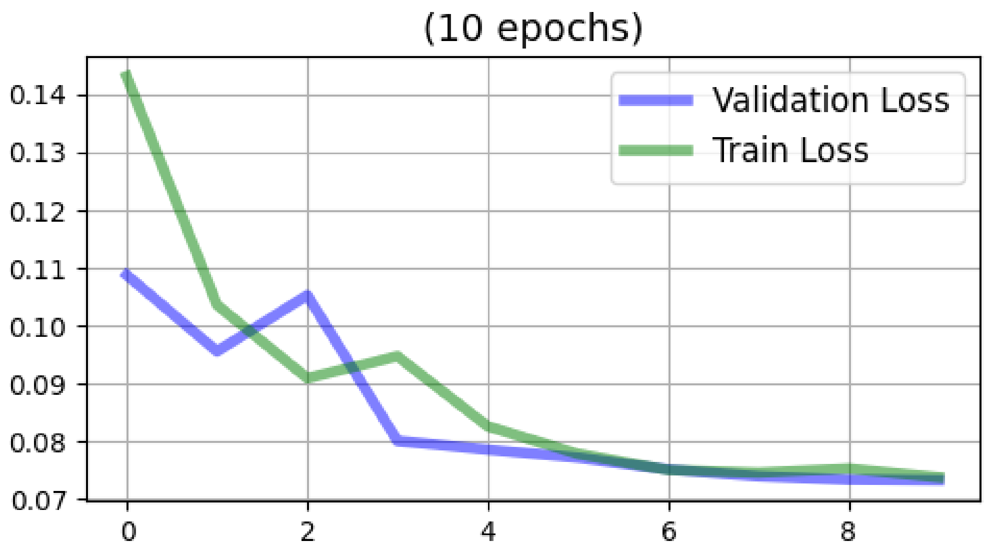

Figure 5: Model Loss (10 Epochs) – Train Loss Vs Validation Loss

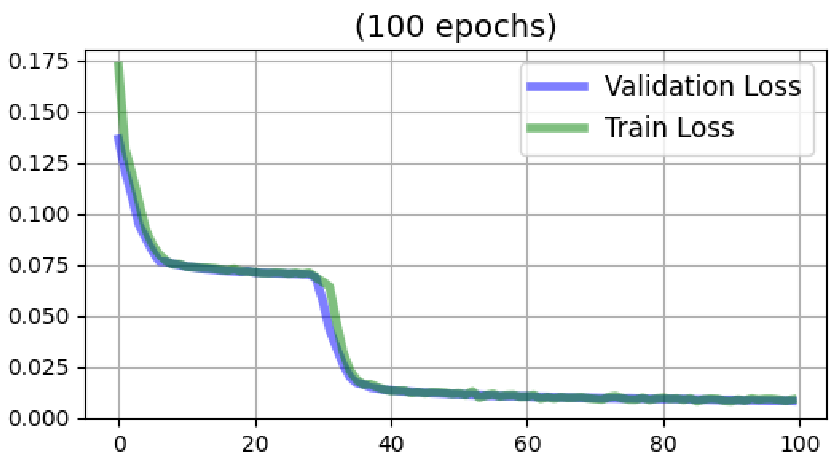

Figure 6: Model Loss (100 Epochs) – Train Loss Vs Validation Loss

Table 3: Validation Loss (10, 100 Epochs)

Validation Loss	Min	Max	Last
10 Epochs	0.07383574	0.1431231	0.07383574
100 Epochs	0.008097777	0.1366134	0.008097777

Batch Scoring: Batch scoring is done by applying the trained model to the test dataset, where it can make future frame predictions. The saved model and its weights are applied to the test dataset and the predictions were saved to a data frame. pickle as 'df.pkl.gz' in the scored data datastore blob container on AML. This pickle file will be used to train the classifier based on the scored data.

Model Classifier: The model classifier used in this research is random forest (RF), selected due to its simplicity and faster classification efficiency in working with binary problems, i.e., for this case, whether there's an anomaly or not in the video frame. It can also present good predictions on large datasets which was one of the primary reasons for its selection as a classifier model for this research. The classifier efficiency is presented in terms of accuracy, precision and recall.

Table 4: Model Classification Metrics

Model	**ROC**	**Accuracy**	**Precision**	**Recall**
10 epochs	0.7067783	0.7611111	0.7649770	0.5030303
100 epochs	0.7500797	0.7933333	0.7950820	0.5878788

Table 4 presents the RF model classification metrics for two different models trained for 10 and 100 epochs, respectively. All the model parameters, including accuracy, precision and ROC have improved significantly in comparison with the model trained for 10 epochs. In general, the results can be considered on par for a model that is self-supervised. The results achieved with this model was compared with other existing approaches on anomaly detection problem as shown in Table-5. The model performance is evaluated in terms of accuracy score which is the common metric among all other models used in the comparison. The proposed model's accuracy score of 79.33% was found to be on par with the compared research works with the work proposed by (Park et al.; 2020) achieving a better accuracy of 82.8% on the Avenue dataset. The work proposed by (Ionescu et al.; 2020) achieved a slightly lesser prediction accuracy (79.2) compared to this proposed work on the same UCSD dataset.

Research	**Approach**	**Dataset used**	**Accuracy Score**
Ionescu et. al. (2019)	Convolutional Autoencoder, SVM classifier	UCSD	79.2
Yu et al. (2020)	Deep Neural Networks (DNN), Future Frame	ShanghaiTech	74.8
Dong et al. (2020)	Generative Adversarial Network (GAN), Future Frame	ShanghaiTech	73.7
Park et al. (2020)	LSTM, Feature Frame	Avenue	82.8
Proposed Model	Predictive Network LSTM, Random Forest Classifier	UCSD	79.33

Table-5: Comparison of model evaluation accuracy score with other research works on Anomaly Detection

Discussions

The result evaluation presented in the previous sections brings us to this discussion. The model training performance improved considerably with the increase in the number of epochs as shown by the decrease in the validation loss. The classifier performance was compared with another similar research works as in Table-5 and was better than average than the
research works in discussion. The primary advantage of this research work had to be the ability to evaluate multiple research problems with anomaly detection, namely, future frame prediction, scoring and classification together, which was not found to be in other research with the same anomaly detection problem. Though, the model did not perform in a widely distinguishable manner in term of the evaluation results, it had to be asserted that this

experimental work on predictive block integration with LSTM for anomaly detection was in par with other research works considered.

7. Conclusion

This research focused on presenting a reinforcement learning based model for the detection of video anomalies on a cloud computing platform like Azure ML using pipelines. The model design was tied to the integration of the predictive coding block with the LSTM based neural network used in this research. The model implementation was done on AML cloud using ML pipelines, that created individual pipelines for preprocessing the dataset, training the model, performing batch scoring and running the classifier. The AML pipeline was executed successfully on the UCSD anomaly detection dataset, and the model was registered on the cloud. The model evaluation returned an accuracy of 79.33 percent, and a precision of 79.5 percent which stands in comparison with other research works tackling the video anomaly detection in video surveillance systems. The use of AML pipelines for the implementation of the model will help with the faster deployment of the model on the cloud as it assists in fast tracking the published model to be used with other video anomaly detection frameworks.

References

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S. and Hengel, A.V.D., 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1705-1714).

Georgescu, M.I., Ionescu, R.T., Khan, F.S., Popescu, M. and Shah, M., 2021. A background-agnostic framework with adversarial training for abnormal event detection in video. *IEEE transactions on pattern analysis and machine intelligence*, *44*(9), pp.4505-4523.

Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K. and Davis, L.S., 2016. Learning temporal regularity in video sequences. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 733-742).

Haselmann, M., Gruber, D.P. and Tabatabai, P., 2018, December. Anomaly detection using deep learning-based image completion. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 1237-1242). IEEE.

Hu, X., Hu, S., Huang, Y., Zhang, H. and Wu, H., 2016. Video anomaly detection using deep incremental slow feature analysis network. *IET Computer Vision*, *10*(4), pp.258-267.

Huszár, V.D., Adhikarla, V.K., Négyesi, I. and Krasznay, C., 2023. Toward Fast and Accurate Violence Detection for Automated Video Surveillance Applications. *IEEE Access*, *11*, pp.18772-18793.

Ionescu, R.T., Khan, F.S., Georgescu, M.I. and Shao, L., 2019. Object-centric auto-encoders and dummy anomalies for abnormal event detection in video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7842-7851).

Kune, R., Konugurthi, P., Agarwal, A., Rao, C. R. and Buyya, R. (2016). The anatomy of big data computing, *Softw., Pract. Exper.* 46(1): 79–105.

Li, W., Mahadevan, V. and Vasconcelos, N., 2013. Anomaly detection and localization in crowded scenes. *IEEE transactions on pattern analysis and machine intelligence*, *36*(1), pp.18-32.

Li, C.L., Sohn, K., Yoon, J. and Pfister, T., 2021. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 9664-9674).

Liu, W., Luo, W., Lian, D. and Gao, S., 2018. Future frame prediction for anomaly detection–a new baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6536-6545).

Liu, Z., Nie, Y., Long, C., Zhang, Q. and Li, G., 2021. A hybrid video anomaly detection framework via memory-augmented flow reconstruction and flow-guided frame prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 13588-13597).

Lotter, W., Kreiman, G. and Cox, D., 2016. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*.

Lu, C., Shi, J. and Jia, J., 2013. Abnormal event detection at 150 fps in matlab. In *Proceedings of the IEEE international conference on computer vision* (pp. 2720-2727).

Luo, W., Liu, W. and Gao, S., 2017, July. Remembering history with convolutional lstm for anomaly detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 439-444). IEEE.

Luo, D., Liu, C., Zhou, Y., Yang, D., Ma, C., Ye, Q. and Wang, W., 2020, April. Video cloze procedure for self-supervised spatio-temporal learning. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 07, pp. 11701-11708).

Pang, G., Yan, C., Shen, C., Hengel, A.V.D. and Bai, X., 2020. Self-trained deep ordinal regression for end-to-end video anomaly detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12173-12182).

Ren, H., Liu, W., Olsen, S.I., Escalera, S. and Moeslund, T.B., 2015. Unsupervised behavior-specific dictionary learning for abnormal event detection. In *British Machine Vision Conference 2015: Machine Vision of Animals and their Behaviour* (pp. 28-1). British Machine Vision Association.

Ristea, N.C., Madan, N., Ionescu, R.T., Nasrollahi, K., Khan, F.S., Moeslund, T.B. and Shah, M., 2022. Self-supervised predictive convolutional attentive block for anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13576-13586).

Rudolph, M., Wandt, B. and Rosenhahn, B., 2021. Same same but differnet: Semi-supervised defect detection with normalizing flows. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 1907-1916).

Sabokrou, M., Fayyaz, M., Fathy, M. and Klette, R., 2017. Deep-cascade: Cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. *IEEE Transactions on Image Processing*, *26*(4), pp.1992-2004.

Sabokrou, M., Khalooei, M., Fathy, M. and Adeli, E., 2018. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3379-3388).

Sabokrou, M., Fayyaz, M., Fathy, M., Moayed, Z. and Klette, R., 2018. Deep anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes. *Computer Vision and Image Understanding*, *172*, pp.88-97.

Sultani, W., Chen, C. and Shah, M., 2018. Real-world anomaly detection in surveillance videos. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6479-6488).

Ye, F., Huang, C., Cao, J., Li, M., Zhang, Y. and Lu, C., 2020. Attribute restoration framework for anomaly detection. *IEEE Transactions on Multimedia*, *24*, pp.116-127.

Zhao, B., Fei-Fei, L. and Xing, E.P., 2011, June. Online detection of unusual events in videos via dynamic sparse coding. In *CVPR 2011* (pp. 3313-3320). IEEE.

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).