Video Anomaly Detection and Classification for Autonomous Deep-Space Robotics

Rohaan Ahmed
Mission Systems Engineer, MDA Space
PhD Computer Science Student, Ryerson University
Toronto, Canada
rohaan.ahmed@ryerson.ca

***Abstract*—Robotic systems operating in deep space will require high levels of autonomy, especially with regards to safety and maintenance. It would be a vital feature, therefore, to have on-board systems capable of detecting anomalous conditions which may indicate hazardous or high-priority situations. This paper introduces a video anomaly detection model which is capable of detecting anomalous situations based within video. The model presented is an Unsupervised Convolutional LSTM Autoencoder which is trained to re-produce Non-anomalous videos, which can then be used to determine whether an anomaly exists in a test video via a "Reconstruction Score". This score is then used to Classify the video as either Anomalous or Non-anomalous using two approaches: A Supervised XGBoost Model, and a Simple Linear Threshold. We find that our Unsupervised Model does very well on the Test Dataset in terms of determining when an anomalous condition occurs. However, due to the small Training Dataset, we are unable to achieve high Classification accuracies using Supervised XGBoost Model.**

I. Introduction and Motivation

Robotic systems operating in deep space will require high levels of autonomy for operation, maintenance, self-repair, and for navigating unexpected situations. These systems will face several unique challenges including communication delays, intermittent communication dropouts, high levels of electromagnetic radiation interference, restrictive communication bandwidth limits, and limited on-board crew time.

This paper proposes a proof of concept for an on-board AI system that is capable of detecting anomalies in video data generated onboard in deep space, and classify this data for transmission to ground-stations.

A direct application benefiting from this system would be the Canadarm3 robotic system, also referred to as Deep Space eXploration Robotics (DSXR), which will be Canada's contribution to the Lunar Gateway project as part of NASA's Artemis Program. The Lunar Gateway will be a space-station that will orbit the Moon, built to serve as an outpost to enable sustainable human exploration of the Moon and beyond (Fig 1).

Canadarm3 (Fig 2), a highly-dexterous and autonomous robotic system, will be used for the Gateway's construction, maintenance, inspection, capture of visiting spacecraft, module relocation, and support of astronauts during spacewalks. This role will be similar to that of Canadarm2, also known as the Mobile Servicing System (MSS), on the International Space Station (ISS) [1], with one critical difference; unlike Canadarm2, Canadarm3 will not be operated directly from the ground and will require far greater levels of autonomy.

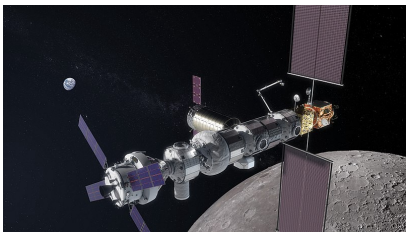

Fig. 1: Lunar Orbital Platform-Gateway (Credit: NASA)

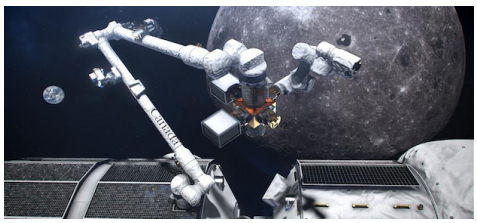

Fig. 2: Canadarm3 (Credit: Canadian Space Agency)

With Earth-based ground-stations expected to have only an estimated 8-hour communication window per week with the Gateway, a smart agent would be required on-board to both detect and assess anomalies within video and image data. The short communication window also imposes constraints on the total bandwidth of data that can be transmitted to Earth, thus making on-board intelligence necessary.

II. Problem Statement

The Canadian Space Agency has made it a priority to contribute to the Artemis program in the domains of Robotics and Artificial Intelligence. The Canadarm3 program will consist of:

1) Exploration Large Arm (XLA): an 8.5-metre-long, 715kg robotic arm
2) Exploration Dextrous Arm (XDA): A smaller, lighter, and more dexterous robotic arm for finer tasks
3) Robotic Toolbox
4) Ground Segment
5) The Control and Artificial Intelligence system to enable the autonomous functionality of the above components,

as well as the rest of Gateway. This can be thought of as the network of all subsystems that form the "brain" of Canadarm3.

This AI-system proposed in this paper will augment the AI system (5) via autonomous detection of the following types of anomalies:

- Structural Anomalies: Anomalies on the Gateway's structure itself, such as fractures, tears, rips, etc.
- Proximity Anomalies: Anomalies in the vicinity of the Gateway's, such as foreign object debris (FOD) and anomalous external vehicle behavior.

III. Systems Engineering Approach

A central tenet of the Systems Engineering approach for the Lunar Gateway is to prove technologies on the ground first. Therefore, will develop and prove our model on a well-known openly available anomaly detection dataset for developing the proof of concept.

IV. Data Source: UCSD Anomaly Detection Dataset

The video data used in the project was the openly available UCSD Anomaly Detection Dataset. The dataset consists of 70 200-frame videos acquired with a stationary camera mounted at an elevation, overlooking pedestrian walkways. Under "normal" conditions, the video contains only pedestrians, whereas "abnormal" events are due to either (i) non-pedestrian objects, or (ii) anomalous motion patterns. Fig 3 shows an example of an anomalous video frame.

Fig. 3: Sample Frame from an Anomalous Video in the UCSD Dataset (Credit: UCSD)

V. Models and Data Engineering

Fundamentally, an anomaly is an occurrence that deviates from what is standard, normal, or expected. In other words, anomalies are rare or abnormal events within otherwise normal data. Therefore, the detection of anomalies requires separating rare and unusual events from the "norm".

A. Supervised vs Unsupervised Methods

Supervised learning approaches typically require well-labeled and balanced datasets. In order to use a purely supervised method of anomaly detection, we would require a dataset in which the anomalies themselves are well-represented and labeled. This conflicts with the very definition of an "anomaly" presented above, i.e., a rare event. Unsupervised learning approaches, on the other hand, learn by "clustering" similar-looking data. Therefore, they are far better suited for the task of anomaly detection, since they can cluster normal data.

In this project, we use a hybrid approach of Semi-Supervised Learning. First, an Unsupervised Model Learns "normal" behaviour in trainiing and outputs a "Reconstruction Score" for inference, which is then used by a Supervised Model to classify the video.

B. Data Pre-Processing and Data Engineering

The UCSD training dataset contains 34 different Non-anomalous surveillance videos, with each folder containing 200 sequential video frames. In order to work with this data, we first perform the following pre-processing:

1) **Convert Datatype:** From TIFF to PNG.
2) **Resizing:** Reduce the size of the images to reduce our network input and output layer sizes.
3) **Pixel Normalization:** Scale pixel values to between 1 and 0.
4) **Generate Training Data:** We divide the training videos into sub-sequences of 10 images (combine frames [1, 2, ... 10], [2, 3, ... 11], ...). Each of these sub-sequences will form a new input at each time-step.
5) **Generate Augmented Training Data:** 34 videos is not sufficient to teach our model, which consists of millions of parameters. We, therefore, augment the data by generating "new" scenes using non-sequential frames. For example, when stride is 3, we combine scenes [1, 4, 7, 10, ... , 30]. This stride is a tunable parameter.

VI. Unsupervised Machine Learning Model for Anomaly Detection, and the Evaluation Metric

The Supervised Machine Model used in our approach is a Convolutional Long Short-Term Memory (LSTM) Autoencoder, developed using a combination of the following well-known Machine Learning techniques:

A. Convolutional Neural Networks (CNN)

We use CNNs to extract and learn features within the input images. Each "Convolution Layer" convolves the input to find abstract representations of the input features. Pooling layers are added to help reduce the dimensions of these extracted feature maps. Fig 4 shows a typical CNN architecture.

Since we combine CNNs with an Autoencoder architecture (explained further on), we also make use of so-called "Deconvolutional Layers", which reproduce features and the original dimensionality from the embeddings.

B. Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN)

LSTM RNNs are regular Artificial Neural Networks (ANNs) with a feedback loops. This feedback allows the LSTM to learn from its previous output, in addition to its current input. Since we are trying to detect anomalies in time-sequence data, the feedback-loop allows our model to learn

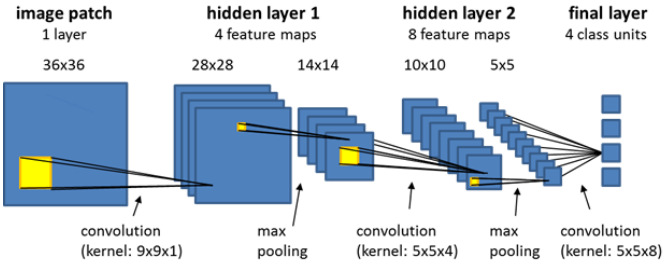

Fig. 4: Convolutional Neural Network Architecture (Credit: ecognition.com)

not only the spatial features, but also temporal features. In simpler terms, our model can learn what should happen in a scene based on what has happened in the past. We use this technique to learn "normal" motion patterns within scenes.

C. Autoencoders

Autoencoders are a popular ANN-architecture used to recreate an input. We use an Autoencoder to teach our model to reconstruct the input data as best as possible. It consists of three sub-sections:

1) Encoder: Given an input, the encoder layers learn representations of the inputs while reducing dimensionality. Our model consists of two encoder layers, one for spatial encoding (for feature encoding) and one temporal encoding (for motion encoding).
2) Bottleneck: Taking the encoder's output as input, the bottleneck is used to both combine and reduce the dimensionality of the encoded features, and prepare it for decoding.
3) Decoder: The decoder layers recreate the original input as best as possible from the feature embeddings. The output of the last decoder layer is a best-possible reproduction of the original input image.

Fig 5 shows a typical Autoencoder architecture.

We train the model only on non-anomalous data, thus teaching our model to reconstruct only non-anomalous data well. Then, we use our model to perform inference (predictions) on anomalous test data. Since the model was never trained to recreate anomalies, the model should not be able to recreate the test data with great confidence. We develop a metric for this confidence, calling it the "Reconstruction Score".

D. Evaluation Metric: Reconstruction Score

We expect our model to reconstruct "normal" scenes with low error, and reconstruct anomalous scenes with large errors. We quantify this error as Reconstruction Score, as follows:

First, we calculate the reconstruction error of each pixel using the L2 norm (Euclidean Distance):

$$e(x, y, t) = ||I(x, y, t) - f_W(I(x, y, t))||_2 \quad (1)$$

where:
I - Pixel Intensity
(x, y) – Pixel Location

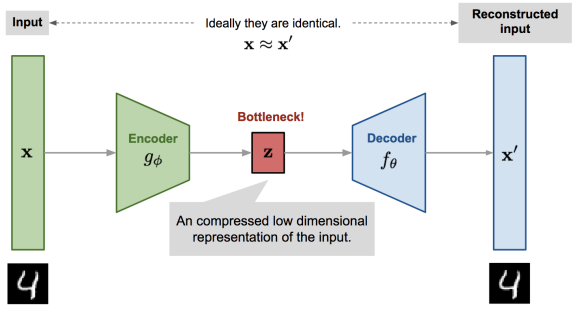

Fig. 5: Illustration of Autoencoder Architecture (Credit: Lilian Weng lilianweng.github.io/)

t – Frame
f_w – LSTM CNN Autoencoder model

Next, we calculate the reconstruction error of the entire frame t by summing the pixel errors:

$$e(t) = \sum_{(x,y)} e(x, y, t) \quad (2)$$

Then we calculate the reconstruction cost of the entire sequence of 10 input frames:

$$c_r = \sum_{(i=t)}^{t+10} e(i) \quad (3)$$

Next, we normalize the reconstruction cost and call it the "irregularity score":

$$s_a(t) = \frac{c_r(t) + c_r(t)_{min}}{c_r(t)_{max}} \quad (4)$$

The "Reconstruction Score" (or Regularity Score) is then:

$$s_r(t) = 1 - s_a(t) \quad (5)$$

Finally, we compute the Reconstruction Score for each 10-frame sequence in the video and plot it, as illustrated in Fig 6.

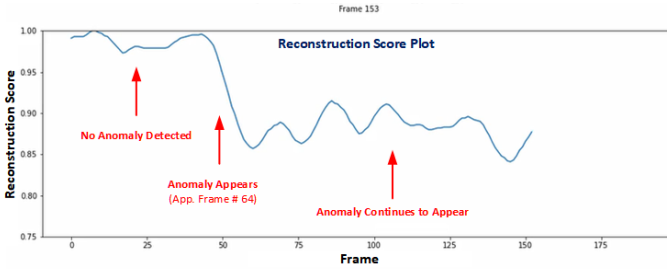

Fig. 6: Sample Plot of the Reconstruction Score for an Entire Test Video

For non-anomalous frames, we expect the Reconstruction Score to be high (closer to 1), whereas for anomalous frames, we expect this score to be lower.

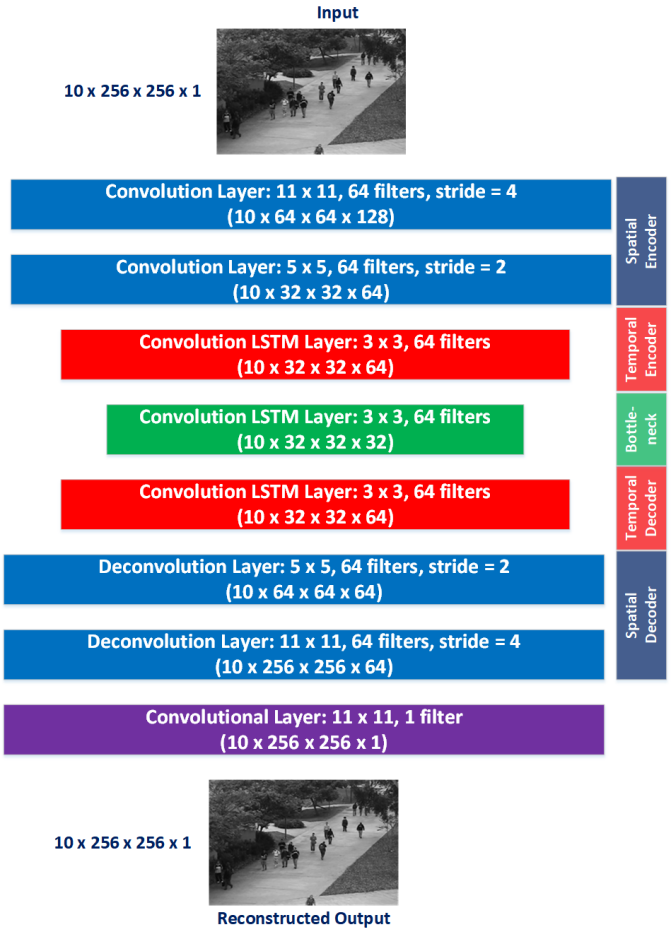

Fig. 7: The Model: A Convolutional LSTM Autoencoder

E. Model Generation and Training

Fig 7 illustrates the Convolutional LSTM Autoencoder architecture used for Unsupervised Learning.

Fig 8 shows the complete cycle of Model Generation, Training, and Inference employed in this project.

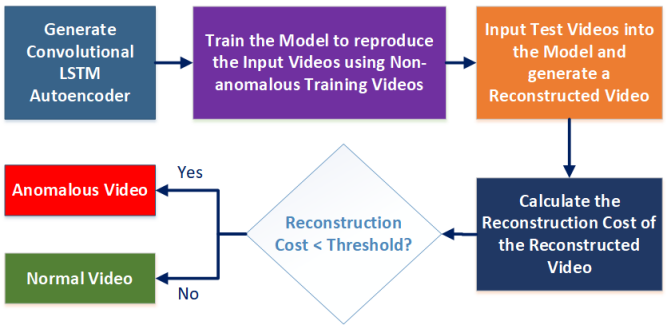

Fig. 8: Model Generation, Training, and Inference Cycle

The Training Data consists of 34 non-anomalous video sequences of 200 frames each, divided into 190 inputs. We train the model on non-anomalous data using Euclidean Loss as our optimization metric, with the goal of recreating the input sequences. We use the "Adam" optimizer, and set the learning rate as a decay function, starting at 10^{-4} and ending with 10^{-5}. Lastly, we use Layer Normalization to normalize the activities of the neurons in the LSTM layers, which helps reduce the training time. We then train for Epochs to 100 witha Batch Size of 1.

In each training cycle (illustrated in Fig 9), our model takes a single 10-frame image sequence as input, reconstructs the image using the Convolutional LSTM Autoencoder, calculates the L2 Loss of the reconstruction, and updates the model's parameters based on the Loss.

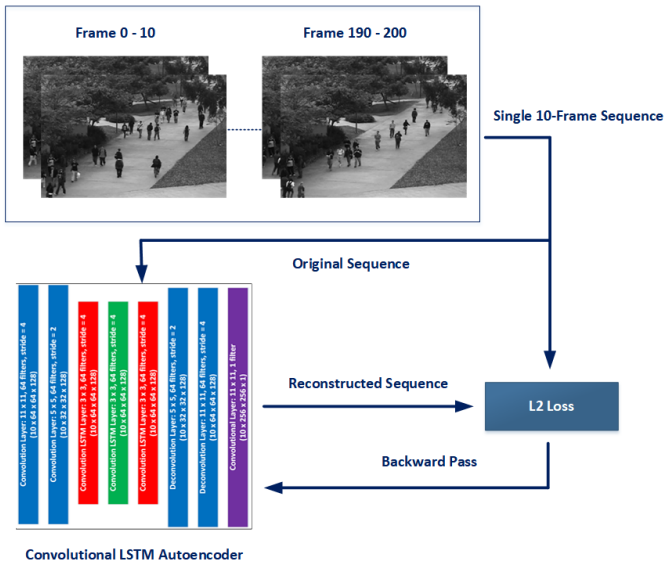

Fig. 9: Model Training Cycle

F. Model Inference

Testing Data consists of 36 anomalous videos of 200 frames each. Anomalies broadly categorized into two categories:

1) Non-pedestrian anomalies: These include wheelchairs, skaters, bicyclists, carts, etc.
2) Pedestrian anomalies: These include abnormal movement patterns from pedestrians, such as runners, pedestrians walking across the grass, etc.

In each inference cycle (illustrated in Fig 10), our model reconstructs the input, and outputs the Reconstruction Score as a vector.

VII. Supervised Classification

In order to Classify our videos as Anomalous or Non-anomalous, we must introduce data-labels (i.e. Supervision). Fig 11 shows a plot of the Reconstruction Scores for all Test and Training datasets

It is obvious from the plot that there is no clear separation boundary between anomalous and non-anomalous Reconstruction Scores. We, thus, experiment with two approaches for classification.

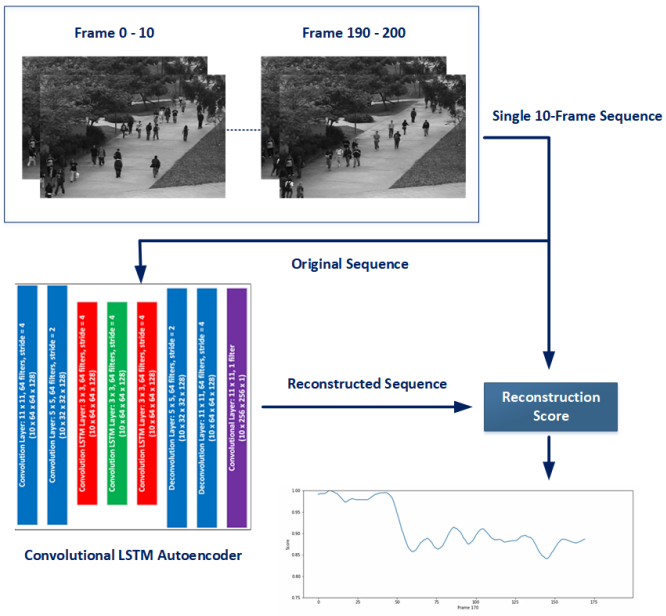

Fig. 10: Model Inference Cycle

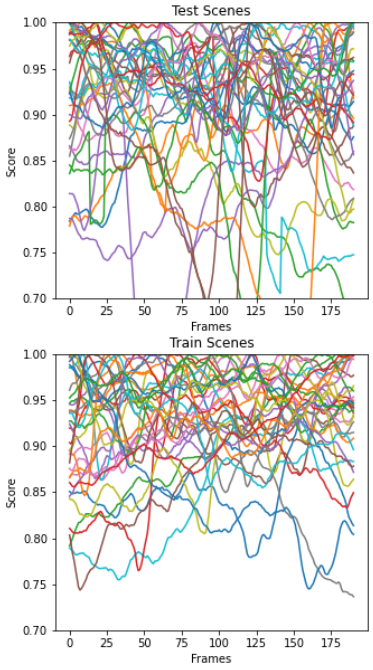

Fig. 11: Combined Plot of the Reconstruction Scores for All Training and Test Videos

A. Classification Using a Simple Linear Threshold

The first approach looks to find an "optimal" threshold which gives us the highest overall accuracy, with room for some mis-classification. We find this value by generating an Accuracy vs Threshold plot, shown in Fig 12.

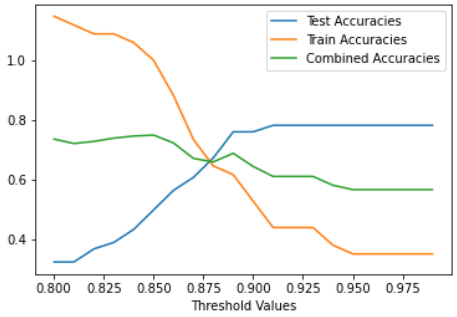

Fig. 12: Accuracy vs Threshold Plot

The values in the Threshold vs Accuracy plot were calculated using the algorithm in Appendix A. From the plot, we see that we achieve the best overall accuracy results at a threshold value of approximately 0.875 (shown in Fig 13).

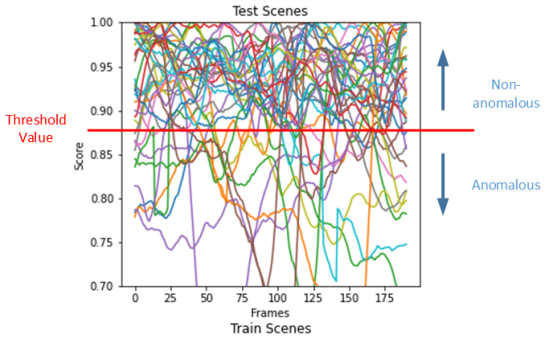

Fig. 13: Classification Using a Linear Threshold

With this simple classification approach, we obtain a combined accuracy of approximately 70%.

B. Classification Using XGBoost, a Supervised Learning Model

In this approach, we use Extreme Gradient Boosting Trees (XGBoost) to develop a non-linear classification model. XGBoost is a tree-based technique able to classify data using an ensemble of several weak random-forest trees. We use XGBoost as follows:

- Assign Class Labels of "1" or "0" to each Anomalous and Non-anomalous Reconstruction Score vector, respectively.
- Divide the data into a Train Set, containing 90% of the data, and Test data, containing 10% of the data.
- Train the XGBoost Classifer on the Train Set
- Evaluate the XGBoost model on the test dataset

As a result of performing the above steps, we obtain an XGBoost classifier with an accuracy of 0.50, which is quite poor. Analogously, we would be equally successful if we tossed a coin to classify the videos! This low accuracy is because our dataset is far too small for a model to learn

from with reasonable accuracy. We only have 70 distinct Reconstruction Score vectors (videos), with each vector containing 190 features (sequences). This also makes it difficult to evaluate our model against current best standards using the same dataset.

VIII. Results and Discussions

Our model was evaluated on the complete Test Dataset, which consisted only of anomalous videos. Fig 14 shows the result obtained for Test Data 2 (filename: Test002).

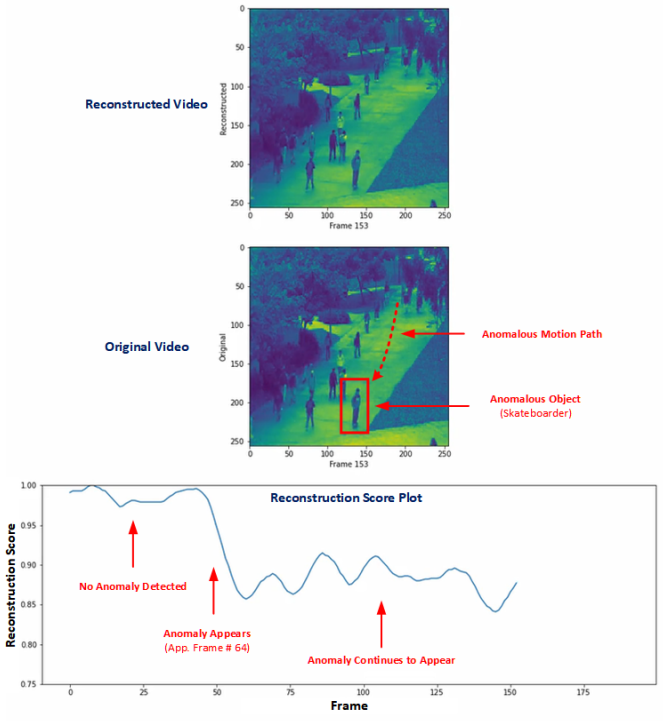

Fig. 14: Evaluating the Model on a Single Test Video

It can be seen that our model's Reconstruction Score reflects the appearance of anomalous data within the scene. In this test case, we see that the score falls sharply when the anomaly (skateboarder) appears in the video around Frame 64. The skateboarder is anomalous due to his/her speed and path of movement when compared with "normal" walking pedestrians.

For every video in the Test Data, we can see a reflection in the Reconstruction Score when an anomaly appears. Therefore, we can safely say that our model does a good job of finding whether an anomaly exists in the video.

For the purpose of comparison, we ran our model on Non-anomalous data as well. Fig 15 shows an example of one such video, Training Data 2 (filename: Train002).

As can be seen in the figure that our model's Reconstruction Score remains high (closer to 1), thus indicating that no anomalous behaviour was detected.

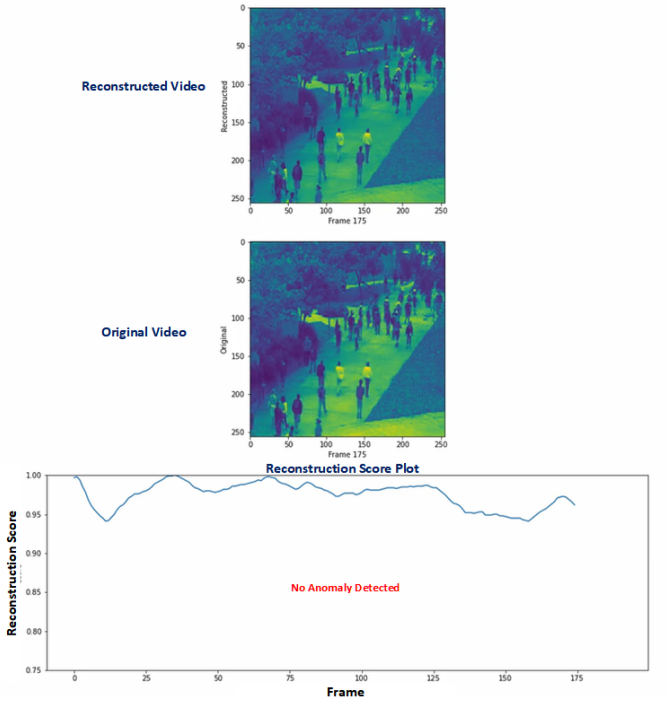

Fig. 15: Evaluating the Model on a Single Train Video

IX. Implementation and Code

The code for this project can be found on the author's GitHub page (Github.com/rohaan-ahmed). A list of packages used in the project can be found in Appendix B

X. Future Work

The project presented in this paper is part of a larger project for anomaly detection using in-space video. Next steps in this project are:

- Improve the Unsupervised Learning Model per feedback from Dr. Nariman Farsad.
- Determine a more accurate method of Supervised Classification.
- Evaluate the model on a larger dataset.
- Generate a synthetic space-representative dataset, and fine-tune the model on that dataset.

References

[1] About Canadarm3. Canada.ca. Published July 7, 2020. Accessed October 1, 2020. https://www.asc-csa.gc.ca/eng/canadarm3/about.asp.

[2] Yong Shean Chong, Abnormal Event Detection in Videos using Spatiotemporal Autoencoder (2017), arXiv:1701.01546.

[3] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, Learning Temporal Regularity in Video Sequences (2016), arXiv:1604.04574.

[4] UCSD Anomaly Detection Dataset. UCSD.edu. Accessed September 19, 2020. http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm.

APPENDIX A

The values in the Threshold vs Accuracy plot were calculated using the following algorithm:

---

for $Threshold \leftarrow$ 0.85 to 0.95 **do**
  for each Reconstruction Score vector **do**
    If the vector contains a value below the $Threshold$, classify that vector as Anomalous
  end for
  Calculate the Following Values:
  Test Set Accuracy
  Train Set Accuracy
  Combined Accuracy
end for

$$\text{Test Set Accuracy} = \frac{\text{Number of Predicted Anomalies (i.e. True Positives)}}{\text{Total Size of Test Dataset}}$$

$$\text{Train Set Accuracy} = \frac{\text{(Total Size of Train Dataset - Number of Predicted Anomalies (i.e. False Positives))}}{\text{Total Size of Train Dataset}}$$

$$\text{Combined Accuracy} = \frac{\text{(Test Set Accuracy + Train Set Accuracy)}}{2}$$

APPENDIX B

The Python packages used in this project are:

- Keras: For the Convolutional LSTM Autoencoder. Keras uses Tensorflow as a backend.
- Matplotlib Pyplot: for generating and saving plots
- Numpy: For array and matrix manipulation
- XGBoost: For Supervised Classification Model
- Pickle: For saving and reading in data from hard drive