# Critical Event Anomaly Detection

## Contents

# 1 Introduction:

Autonomous and Intelligent systems are becoming more popular in several fields from Home vacuums, to several applications in transport. In more recent times, there have been more intelligent systems in open ended applications such as autonomous driving, food delivery, medicine etc. where the variety of scenarios and situations these systems encounter is very diverse and unlimited. In such diverse cases, it is hard to cover all the types of scenarios in the training and testing phases[5] - which necessitates millions of kilometers of drive testing to catch the edge cases that are left out during training.

Catching these rare, anomalous and edge cases in such a large drive data involving millions of kilometers is often a challenging task. It cannot be fully automated using supervised machine learning models as those models are also trained on similar datasets as the systems they are trying to evaluate. This makes these automated evaluation pipelines biased and unlikely to catch the timestamps which are not handled by the intelligent system they evaluate. Thus, in the industry, several hundreds of human hours are put for years to manually screen and verify millions of hours of drive recording data from the testing fleets of autonomous vehicles.

We thus propose a method which uses both supervised and unsupervised learning to mitigate the weaknesses in current screening algorithms which cannot be used for this particular application.

Usage of unsupervised learning reduces the dependence of any biased dataset which is the main problem in current methods. Further, we extend the usability and functionality by clustering the filtered edge cases. This helps us to reject unimportant edge cases or rare events such as intersections that are not safety critical. The solution we propose will thus serve as a comprehensive and robust automated system in the critical phase of validation of intelligent systems for safety critical open ended applications such as self-driving vehicles, aviation, robotic surgery, space rovers and beyond.

✅ Project Goals

1. Explore various pseudo-labeling methods to classify unlabeled data effectively, ensuring high-quality labels that minimize noise and improve model robustness.
2. Investigate techniques to transform and optimize features for better Gaussian distribution adherence, enhancing the model's ability to detect anomalies in high-dimensional sensor data.
3. Design a robust framework that efficiently utilizes limited labeled data from normal classes to enhance the model's understanding of sensor data, improving anomaly detection performance.
4. Perform rigorous testing on real-world datasets to validate the model's effectiveness in accurately detecting anomalies, ensuring it meets industry standards for reliability and safety in autonomous systems.
5. Facilitate the implementation of advanced unsupervised techniques, such as clustering, to further analyze sensor data and improve overall system performance in anomaly detection.
6. Assess the proposed method's performance across different datasets and scenarios to ensure applicability beyond autonomous driving, identifying any domain-specific challenges.

## 1.1 Literature Review and Background Study

### 1.1.1 Semi-supervised anomaly detection

In semi-supervised techniques, we assume that the training data contains labeled instances only for the normal class. More formally, during training, we focus on modeling P(x), which represents the probability distribution of normal data. At test time, instances with very low probabilities are classified as anomalies. The techniques considered here will work better if all the features here are Gaussian. We can take the logarithm of those features and then optimize them.

There have been methods explored by [7] which use pseudo-labeling to help to classify unlabeled data. Further, as proposed by Zhang in [8], to detect 3D objects from dashcam images, pillar-based 3D detector is adopted. Its focus on autonomous driving may reduce generalizability to other domains, and the teacher-student framework introduces computational complexity that could slow down training. In our proposed method, we use semi-supervised learning to better encode and represent

the sensor data (Images in our case) so that we can perform unsupervised methods such as anomaly detection and clustering in a better way for this specific application.

### 1.1.2 Anomaly detection of video data[6]

Various deep learning models, including CNNs, LSTMs, and autoencoders, have shown significant improvements in detecting deviations from normal behavior by learning temporal and spatial representations. This technique is crucial for applications like surveillance, healthcare, and autonomous systems. However, challenges remain in handling complex, unlabeled datasets and minimizing false positives due to the variability of "normal" activities across different contexts. Also, the applications of anomaly detection in the context of autonomous driving validation has been very minimal. The possibility was first proposed in the MIT 6S191 lectures on anomaly detection[]. In our proposed solution, we apply it to solve mainly the validation stack of autonomous driving vehicles and extend it further by implementing clustering methods on top of it.

### 1.1.3 Anomaly detection of drive data in cars

This involves data from GPS, LiDAR, radar, onboard cameras, and CAN bus data (speed, acceleration, steering angles). Advanced methods use deep learning, such as Recurrent Neural Networks (RNNs) or LSTMs, to capture the temporal dynamics of driving patterns. Despite progress, real-time detection, large datasets, and false positives present ongoing challenges, especially in complex driving conditions. As mentioned in the introduction, this proposal tries to mitigate the same with the addition of unsupervised and semi-supervised methods.

### 1.1.4 sigma errors in autonomous driving

The concept of Six Sigma errors in autonomous driving emphasizes minimizing defects to enhance system reliability and safety. There are millions of autonomous vehicles active on the road on any given day that even a small error (0.0000001% - six sigma error - a decimal value with six zeroes) can cause hundreds of accidents per day. A key strategy is to include sensor fusion and redundancy, machine learning, real-time monitoring, simulation and testing, and fault tolerance mechanisms. One of the main strategies to mitigate this is to test the vehicle in both simulations and real world and check for errors - More billions of miles verified implies lesser probability of error. By implementing these approaches, the goal is to achieve Six Sigma error rates (fewer than 3.4 defects per million instances), ensuring robust and reliable performance in autonomous systems**.**

## 1.2 Dataset Description

### 1.2.1 YouTube Dataset annotated by the team

YouTube dashcam data annotated by the team - We are using this to further cement our testing process. The team will select random YouTube dashcam videos with and without car crashes to check the validity of our model. The randomness would eradicate any bias introduced by current curated

datasets. The frames in these videos would be labeled as either 'rare', 'safety-critical' or 'normal' by the team. The dashcam videos serve as a great testbed as they are varied by:

1. Location
2. Type of dashcam
3. Type of car
4. Location of dashcam
5. Weather

One of the main reasons YouTube videos were selected for testing the results is to better represent the diversity and variety of the data these models would encounter, which may not be captured by a single dataset on Kaggle. The dataset would eventually be published to make a good benchmark for similar applications.

### 1.2.2 Car Crash Dataset (GitHub)

- https://www.kaggle.com/datasets/asefjamilajwad/car-crash-dataset-ccd/data
- https://github.com/Cogito2012/CarCrashDataset

The Car Crash Dataset (CCD) **[4]** is assembled for the analysis of traffic incidents. It encompasses actual traffic collision footage recorded by dashcams affixed to moving vehicles, which is vital for the development of secure self-driving technologies. CCD is unique compared to current datasets due to its varied accident annotations, such as environmental factors (day/night, snowy/rainy/clear weather conditions), involvement of ego-vehicles, participants in the accident, and descriptions of the causes of incidents.

| Detail | Statistics |
|---|---|
| Number of videos | 3000 normal, 1500 with car crashes |
| Number of frames per video | 50 |
| Frame per second | 15 |
| Binary Label of normal frame | 0 |
| Binary Label of anomalous frame | 1 |
| Additional Annotations (for some videos) | Time (day/night), Weather (rain/snow/clear), description and cause of accident, participants |

Feature files are present in a separate directory and consist of extracted VGG-16 features of all frames and all detected bounding boxes. All of the bounding boxes are saved in these files. How this task was achieved is present in their GitHub repository. The dataset is organized as follows

CarCrash

```
├── codes                # useful codes for analyzing the dataset
├── vgg16_features
│   ├── positive         # feature files of positive (accident) videos
│   │   ├── 000001.npz
│   │   ├── ...
│   │   └── 001500.npz
│   ├── negative         # feature files of negative (normal) videos
│   │   ├── 000001.npz
│   │   ├── ...
│   │   └── 003000.npz
│   ├── train.txt        # list file of training split
│   └── test.txt         # list file of testing split
├── videos
│   ├── Normal           # normal driving videos
│   │   ├── 000001.mp4
│   │   ├── ...
│   │   └── 003000.mp4
│   ├── Crash-1500       # crash accident videos
│   │   ├── 000001.mp4
│   │   ├── ...
│   │   └── 001500.mp4
│   └── Crash-1500.txt   # annotation file for crash accident
└── README.md
```

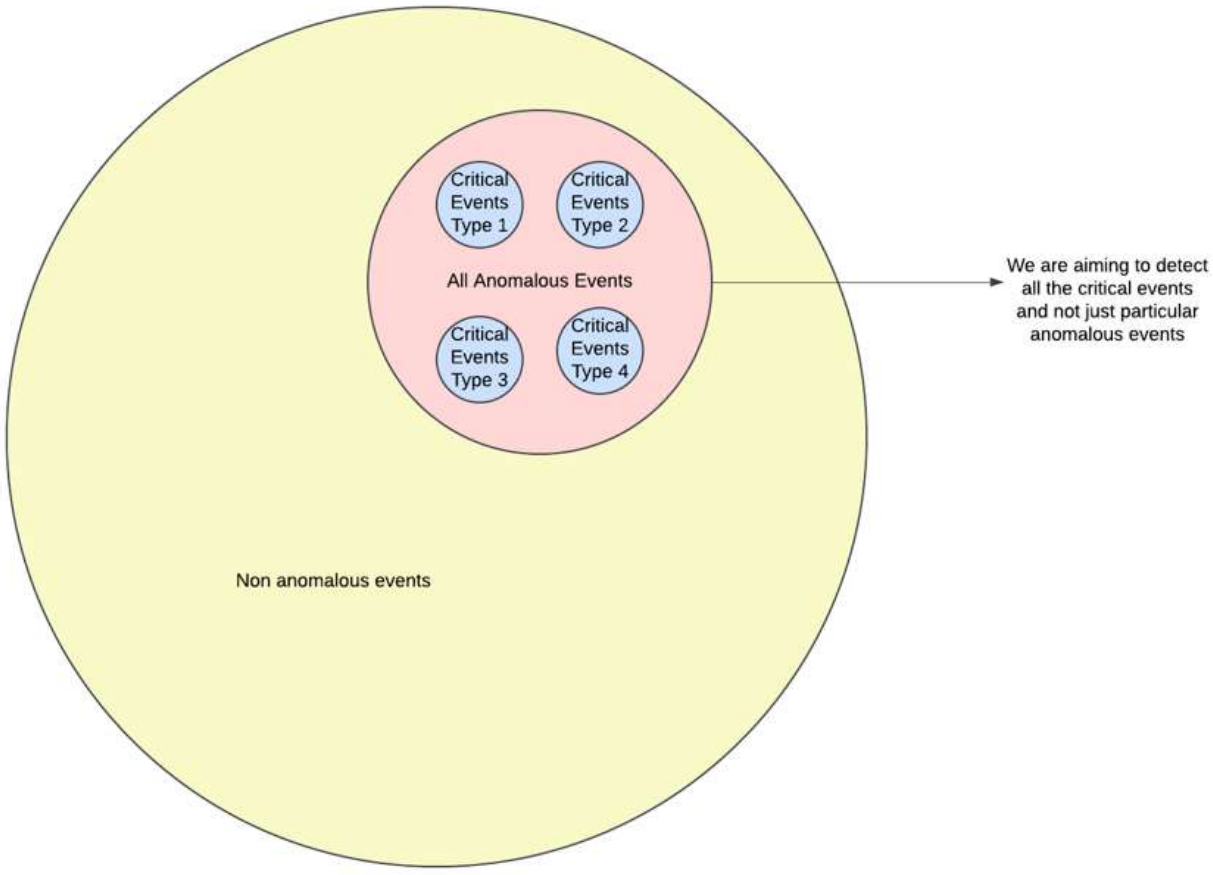
# 2 Problem Definition

Identification of Critical Rare and Anomalous events in large datasets involving sensor data such as dashcam footage or robot sensor recording for applications like autonomous driving is a time consuming task. This is further complicated by the need of classifying unseen scenarios which may not be detected by purely supervised methods. Furthermore, there is a shortage of ground truth data. **[3]** Thus, this involves 100s of hours of human manual labor to screen and flag the critical events. Such handcrafted pipelines are not efficient **[1]** and can be error-prone.  We propose a novel method

involving semi-supervised learning which efficiently classifies and clusters critical and rare events irrespective of whether it appeared in the training phase, thereby reducing human labor while automating the process accurately.

## 2.1 Motivation

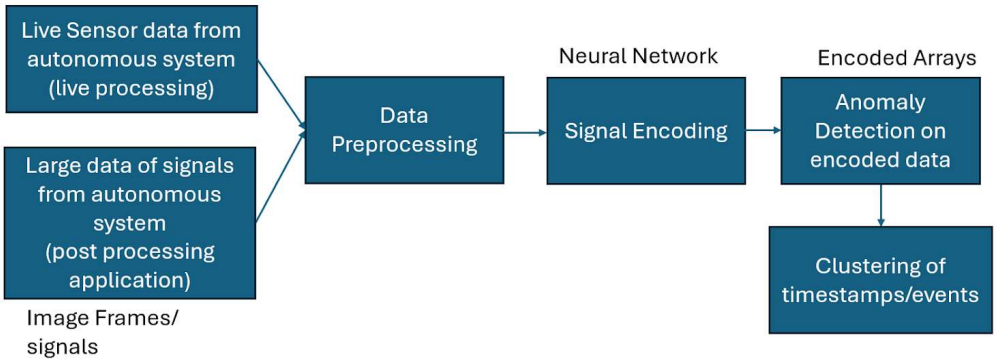
Automated driving and robots are becoming more commonplace in several industries such as transportation, industrial automation etc. Such safety critical applications need to be very accurate (within 6-sigma error) as the systems are deployed to fleets having millions of cars / robots. Given this large number, even minor errors (even 6-sigma) would lead to hundreds of accidents per day if unnoticed **[1]**. Solving this problem involves 100s of hours of manual labor to screen and flag the critical events. Further, current supervised learning models only classify a subset of critical events as they are biased by their training data. Thus we introduce an automated approach using both supervised and unsupervised learning to make sure all anomalous / critical events are filtered for facilitating further processing to improve the intelligent algorithm.




We are thus aiming to have 100% True positives and almost 0% False Negatives. Thus, the metric we aim to optimize is not the F1 score but rather

$$TargetMetric = \frac{TruePositives}{1 + FalseNegatives}$$

# 3 Methodology




## 3.1 Data Processing Methods

Image Pre-processing (before encoding) **[2]**

1. Extract frames from video (implemented)
2. Reduce the dimension of image to fit the CNN model (implemented)
3. Apply filters to image to ensure features are well represented (implemented)
4. Flatten the image from 3 channels to 1 channel grayscale (implemented)
5. CNN to convert image to vectors (implemented)

Data Pre-processing (after encoding)

1. Use Entropy to process the data (implemented)
2. Remove the rows with 0 variance (implemented)
3. Increase variance of each feature using PCA (implemented)
4. Check for rows with high covariance and choose the one with highest entropy (implemented)

### 3.1.1 Data Preprocessing - Extracting Image Frames and resizing them to feed into Autoencoder

The target video is first fed into the `extract_frames.py` where a single frame for each `t` seconds is extracted. The hyperparameter `t` is decided by the user depending on how eventful or uneventful the video is. For example, if a video consists mostly of consecutive frames of less variance, then `t` can be large, whereas if the video is from a car race/drag race where variance of scenarios is large, then `t` can be small.

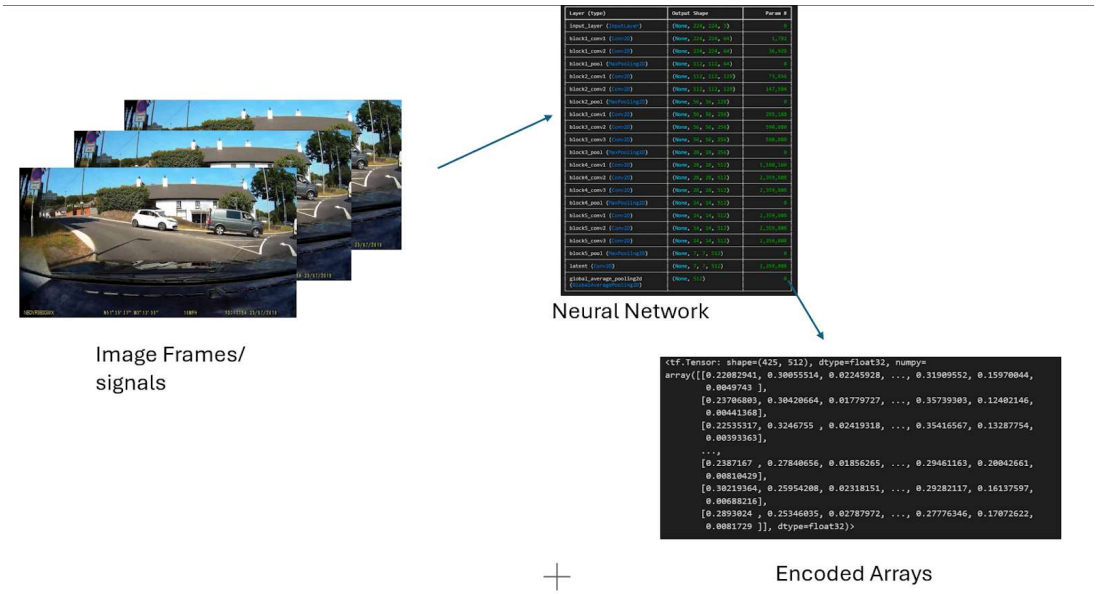
```
Hyper_parameter_1 => t
# t defines the duration per frame in extract_frames.py
```

After extracting the frames, we then scale the image dimensions to match the input dimensions of the autoencoder model.

# 3.2 ML Algorithms Used

## 3.2.1 Semi-Supervised Learning (Encoders)

The auto-encoder will be trained on automotive data (or the intended target application) first. This would further enable it to encode the signals (Image frames) into vectors that best represent them. Post training, only the encoder part of the model is used in the final pipeline




### 3.2.1 (a) CNN Auto-Encoders Introduction, why we use them?

Autoencoders are used for unsupervised learning of efficient coding. They are neural networks designed to learn a compressed representation (encoding) of input data, which can be used for tasks

such as dimensionality reduction, denoising, and anomaly detection. The autoencoder consists of an encoder that maps the input to a lower-dimensional space and a decoder that reconstructs the input from this lower-dimensional representation.

VGG16 and MobileNetV2 are popular convolutional neural network architectures used for feature extraction in image processing tasks.

### 3.2.1 (b) VGG 16 with Decoder Stripped

Known for its simplicity and depth, VGG16 consists of 16 layers and is effective in capturing high-level features from images. It is often used for tasks requiring high accuracy and detailed feature extraction.

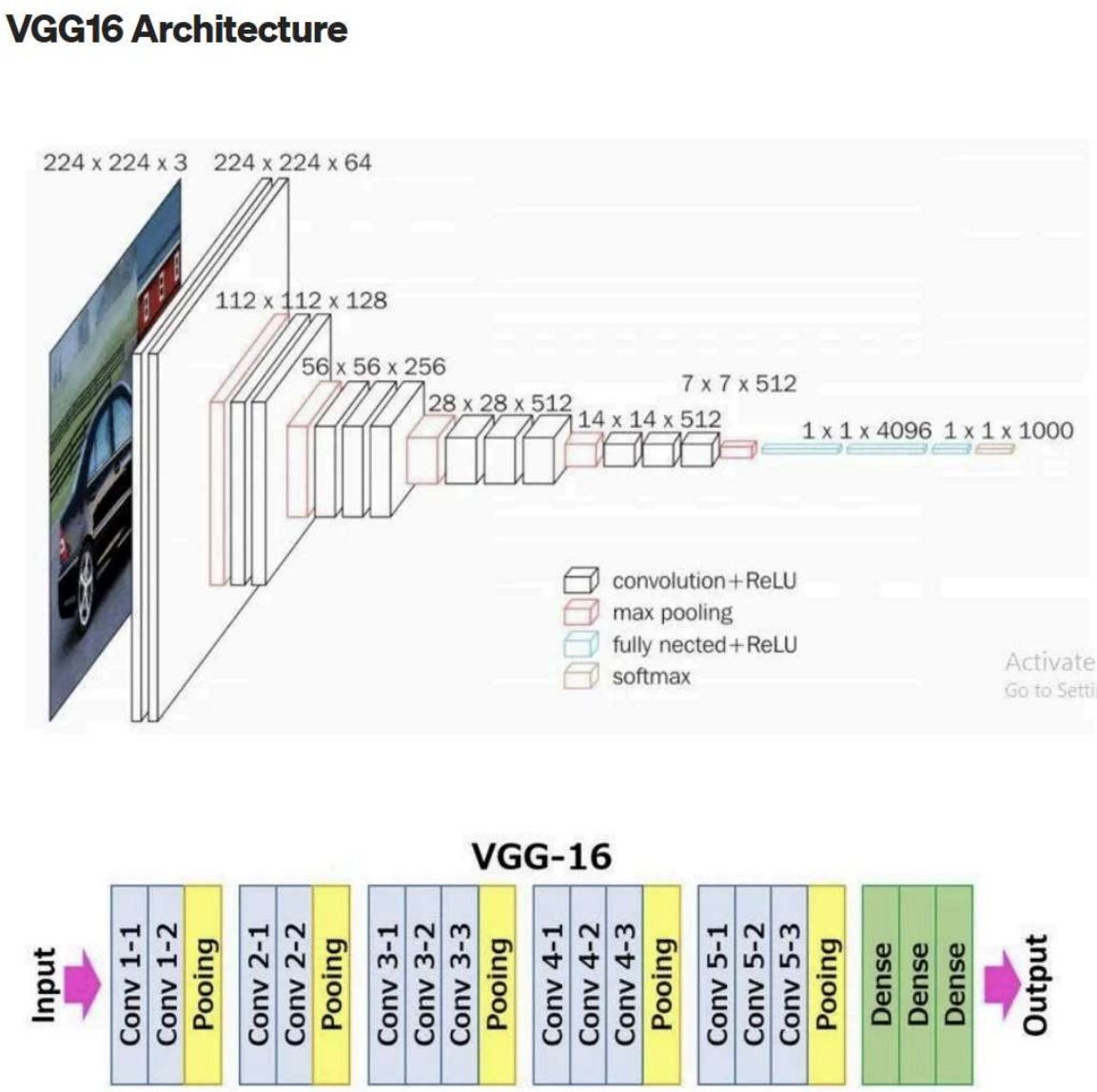
Layers: VGG16 primarily uses sequential layers of 3x3 convolution filters with a stride of 1 and padding, followed by max-pooling layers.

Convolutional Blocks: There are five main blocks in VGG16, each consisting of 2 or 3 convolutional layers followed by a max-pooling layer. The number of filters increases progressively with each block (from 64 up to 512), leading to richer feature maps with spatial resolution reduction.

Activation Function: After each convolutional layer, a ReLU activation function is applied, introducing non-linearity and helping in learning complex patterns.

Pooling: Max pooling (2x2 with stride 2) reduces the spatial dimensions of the feature maps, allowing the network to progressively capture more abstract representations.

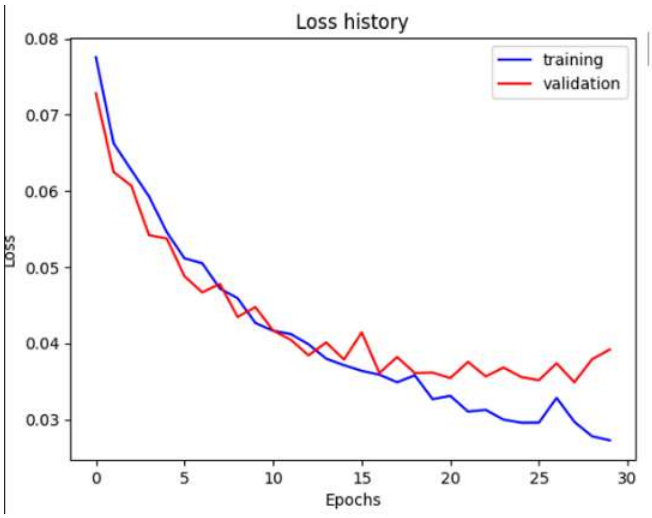
Characteristics: VGG16 is a deeper, straightforward network, emphasizing depth with many parameters (due to standard convolution layers), making it computationally intensive and requiring more memory.




Source: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

### 3.2.1 (c) Supervised Learning using our Video Dataset

We performed supervised learning with help of the Car-crash dataset. We used pre-trained VGG model and fine tuned this with our car-crash dataset. The inputs were the video frames that were extracted using a helper script. The target variable was the anomalous frames(binary). Our observation is that it improved accuracy on our dataset. However, the accuracy is still low and this model alone cannot be used for a critical event detection system by itself. The loss and accuracy graphs are as shown below.




### 3.2.1 (d) MobileNetV2 with Decoder Stripped

Designed for efficiency, MobileNetV2 uses depth-wise separable convolutions to reduce the number of parameters and computational cost. It is suitable for applications where computational resources are limited, such as mobile and embedded devices.

Layers: MobileNetV2 is based on an efficient depth-wise separable convolution technique, which reduces the number of parameters and computation compared to traditional convolutions.

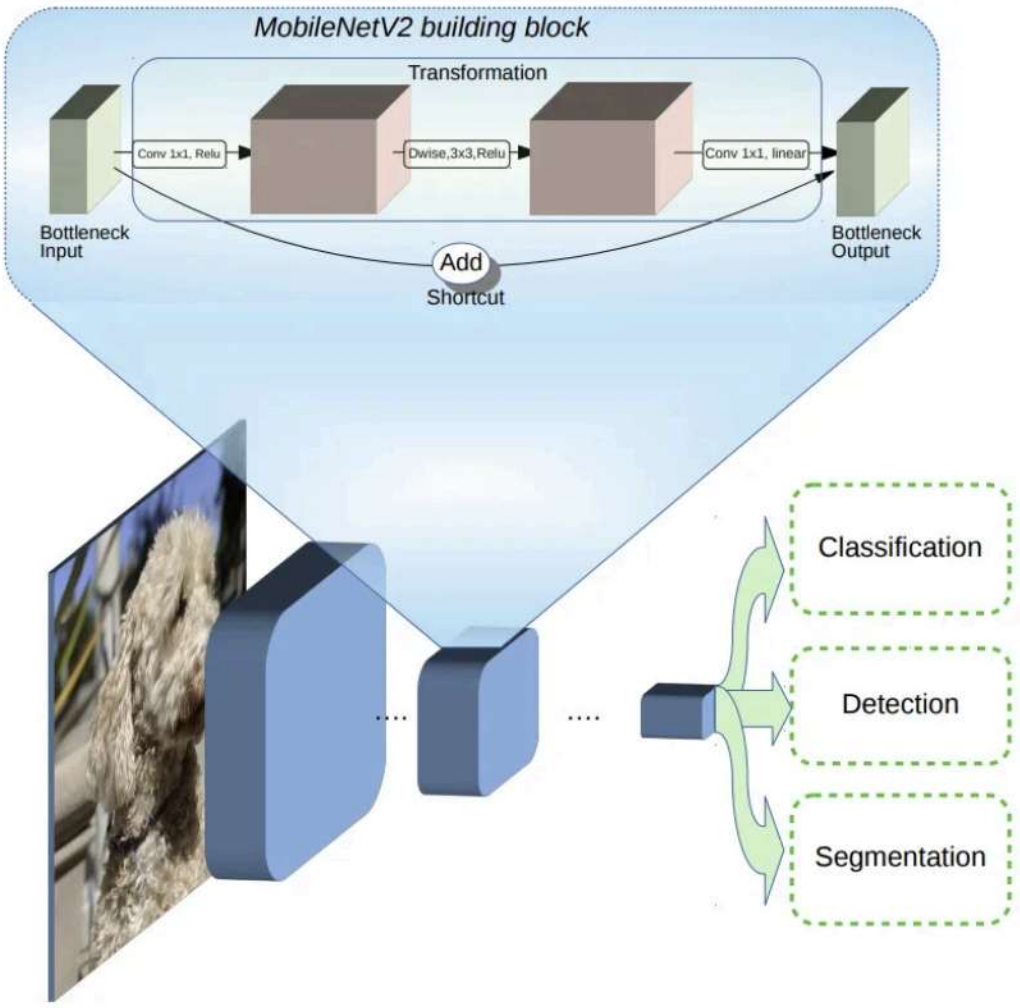
Bottleneck Residual Blocks: The architecture primarily consists of bottleneck residual blocks, with a combination of 1x1 convolutions (for dimensionality adjustment) and depth-wise 3x3 convolutions.

Inverted Residuals and Linear Bottlenecks: MobileNetV2 employs inverted residuals with linear bottlenecks. The block starts with a 1x1 convolution to expand the channels, followed by a 3x3 depth-wise convolution, and another 1x1 convolution to project it back to a lower-dimensional space.

Activation Function: ReLU6 is used after each convolution, which is particularly efficient for mobile and edge devices.

Stride and Down-sampling: Down-sampling occurs through stride depth-wise convolutions instead of max-pooling layers.

Characteristics: MobileNet V2 is designed for efficiency, focusing on reducing computational cost and memory usage, making it suitable for mobile and edge devices.




Source: https://research.google/blog/mobilenetv2-the-next-generation-of-on-device-computer-vision-networks/

### 3.2.1 (e) Observed Performance Comparison between VGG 16 and MobileNetV2 Encoding

By using VGG16 and MobileNetV2, the autoencoder can benefit from the rich, pre-learned features, improving the quality and efficiency of the encoding process.

**Summary of MobilenetV2 and VGG16**

| Aspect | VGG16 | MobileNet V2 |
|---|---|---|

| | | |
|---|---|---|
| **Depth and Complexity** | Deeper with many parameters (16 convolutional layers) | Shallow with fewer parameters due to depth-wise separable convolutions |
| **Convolution Type** | Standard 3x3 convolutions | Depth-wise separable convolutions with 1x1 convolutions |
| **Pooling** | Max pooling after each block | Stride convolutions for down-sampling |
| **Activation** | ReLU after each convolution | ReLU6 for activation efficiency |
| **Computational Cost** | High due to standard convolutions | Lower due to separable convolutions and efficient architecture |
| **Memory Efficiency** | High memory usage | Optimized for lower memory consumption |
| **Use Cases** | Suitable for high-power devices (e.g., servers) | Optimized for mobile and edge devices |

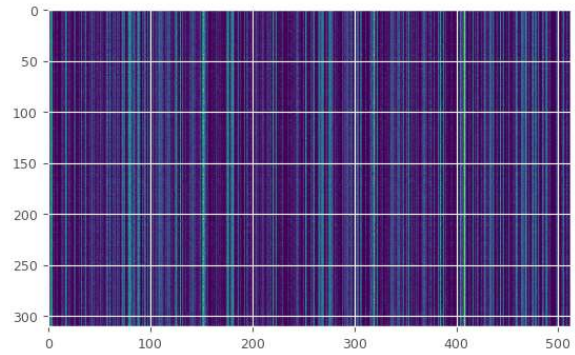
The following table represents the statistics of both the autoencoders on the system in terms of number of parameters

| Model | Total Params | Trainable Params | Non-trainable Params |
|---|---|---|---|
| **VGG16** | 17,074,496 (65.13 MB) | 2,359,808 (9.00 MB) | 14,714,688 (56.13 MB) |
| **MobileNetV2** | 3,317,056 (12.65 MB) | 1,475,072 (5.63 MB) | 1,841,984 (7.03 MB) |

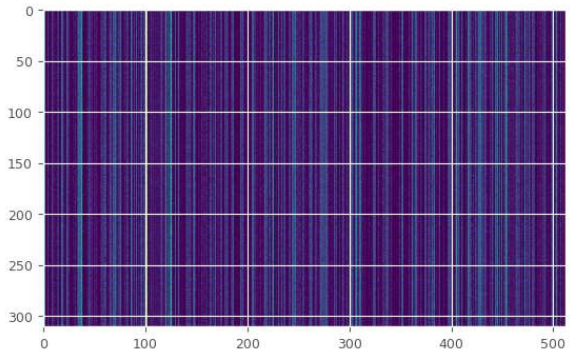
The following table represents the evaluation parameter in terms of total time to encode the video and the entropy value obtained.

| Model | Total Time to Encode the video | Entropy during one sample test run for one video |
|---|---|---|
| **VGG16** | 4.8 s | 5.4331095209855285 |
| **MobileNetV2** | 9.01 s | 5.3710871885056655 |

The following images show the VGG16 Encoded Image Vector and MobileNetV2 Encoded Image Vector representation for 311 images of the test dataset. Note: These vectors are not human-understandable, but we can see the differences in the encoding. Each row is a single frame represented in a `512` feature vector.




VGG 16 Encoded Vectors




MobileNet V2 Encoded Vectors

## 3.2.2 Post Processing of the Encoded Data before Anomaly Detection

**→ Multiplying with a custom Hyperparameter to the encoded vectors to scale up the variance**

We are multiplying the image values by 10 to increase the variance by a factor of 10, so that it is easier for anomaly detection in the following steps

```
Hyper_parameter_2 => variance_multiplier
# variance_multiplier defines by how much the variance is scaled up
```

**→ Seeing which encoded features have low standard deviation for feature drop-off**

Some features had very less standard deviation (1e-10 or even 0). These features did not contribute anything to the final probability calculations in anomaly detection and thus were identified and dropped off.

Best encoded image was by `VGG16` during one test run and had the entropy: `5.4331095209855285`. Hence we chose this for all future steps.

## 3.2.3 Post Processing of Encoded data using PCA

To increase the entropy of the images, PCA is performed and the principal component with the highest variance is selected.

1. **Convert to Numpy Array**: The encoded images are converted to a numpy array if they are not already in that format.
2. **Ensure Non-Negative Values**: The absolute values of the encoded images are taken to ensure all values are non-negative.
3. **Apply PCA**: If n_components is specified, PCA is applied to the encoded images to reduce the number of dimensions to n_components.
4. **Ensure Non-Negative Values After PCA**: The absolute values of the encoded images are taken again to ensure all values are non-negative after PCA.
5. **Calculate Entropy**: The entropy for each feature (column) is calculated.

By applying PCA, we are ensuring that the most significant features are retained, which can lead to a more accurate and meaningful entropy calculation.

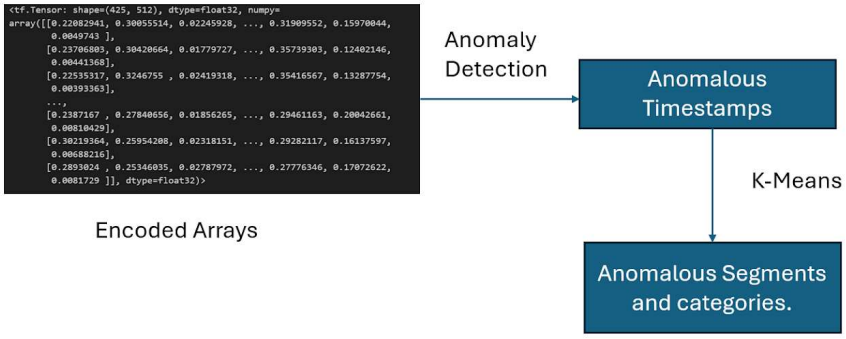
The following output shows the entropies for VGG16 and MobileNetV2

```
Entropies before PCA: Entropies: [np.float64(5.2808), np.float64(5.438)]
```

```
Entropies after PCA: Entropies: [np.float64(5.670), np.float64(5.657)]
```

Hence, PCA improves the entropy and is a necessary step.

## 3.3 Anomaly Detection (unsupervised learning)




### 3.3.1 Approach 1: Calculating probabilities using Multivariate Gaussian

- **Description:**

- The multivariate Gaussian distribution is an extension of the normal distribution to multiple dimensions, characterized by a mean vector and a covariance matrix.
- It models the probability of data points in a multi-dimensional space where each dimension corresponds to a feature.

- **Usage:**
  - **Feature Encoding:** Images are passed through CNN encoders (like VGG16 or MobileNetV2) to obtain lower-dimensional feature vectors.
  - **Anomaly Scoring:**
    - The assumption is that normal data points (features) follow a multivariate Gaussian distribution.
    - The mean and covariance of the distribution are estimated from the encoded features of the training data.
    - **Anomaly Detection:** An anomaly score for each data point is computed based on its likelihood under the estimated multivariate Gaussian distribution. Points with low likelihoods are considered anomalies.

Please see section 4.2.1 for the results using this method.

### 3.3.2 Approach 2: Using GMM

- **Description:**
  - GMMs are probabilistic models that assume data is generated from a mixture of several Gaussian distributions, each with its own mean and covariance.
  - Each Gaussian component represents a cluster within the overall data distribution.
- **Usage:**
  - **Model Fitting:**
    - The encoded feature vectors are used to fit a GMM, capturing the underlying clusters in the data.
    - **Number of Components:** The number of Gaussian components is a hyperparameter that can be tuned based on the data.
  - **Anomaly Scoring:**
    - For each data point, the model computes the probability of belonging to each Gaussian component.

- **Anomaly Detection:** Data points with low maximum probabilities across all components are flagged as anomalies since they do not fit well into any of the learned clusters.

Please see section 4.2.2 for the results using this method.

### 3.3.3 Approach 3 (FAILED APPROACH): Assuming each variable is independent and calculating image probabilities

1. **Independence Assumption:**
   - Each element in the encoded image vectors is considered independent.
   - The joint probability of an encoded image is the product of the probabilities of its individual features.
2. **Probability Computation:**
   - **Parameters Estimation:**
     - Calculate the mean (`mu`) and standard deviation (`sigma`) of each feature across all encoded images.
   - **Probability Density Function (PDF):**
     - For each feature, compute the probability of the observed value using the Gaussian (normal) distribution PDF
   - **Joint Probability:**
     - Multiply the probabilities of all features to get the joint probability for the image
     - Where ( N ) is the number of features in the encoded image.
3. **Anomaly Detection:**
   - Images with low joint probabilities are considered anomalies.
   - A threshold is set (e.g., based on percentiles) to identify anomalous images.

**Reasons for Failure of this Approach**

- **Independence Assumption:** Features extracted from images are often correlated. Ignoring this can reduce detection accuracy.
- **Loss of Information:** The method may not capture interactions between features that are critical for identifying anomalies.
-

Here is a comparison of the three approaches:

| Aspect | Approach 1: Multivariate Gaussian | Approach 2: GMM | Approach 3: Independent Features |
|---|---|---|---|
| Core Assumption | Data follows a single multivariate Gaussian distribution | Data follows a mixture of Gaussian distributions | Features are independently distributed |
| Model Complexity | Medium - Single distribution with covariance matrix | High - Multiple distributions with parameters | Low - Individual feature distributions |
| Feature Correlation | Considers correlations between features | Considers correlations within each component | Ignores feature correlations |
| Flexibility | Limited to unimodal distributions | Can model complex, multimodal distributions | Very limited, assumes simple distributions |
| Performance | Good (F1 Score: 0.537) - performance is benchmarked using one sample video | Better (F1 Score: 0.558) performance is benchmarked using one sample video | F1 Score : 0 (Failed approach) |
| Computational Cost | Moderate | Higher due to multiple components | Low |
| Main Limitation | Assumes the data should follow multivariate gaussian distribution. | Requires choosing number of components which is difficult for open ended problems like anomaly detection | independence assumption is not fully valid |

## 3.4 Deciding the Percentile of Anomalous Images

In order to threshold the probabilities from the models, we decide how much percentile of the images are anomalous. This `percentile_anomaly` is a user defined parameter, based on which is then used to calculate the threshold probabilities for filtering.

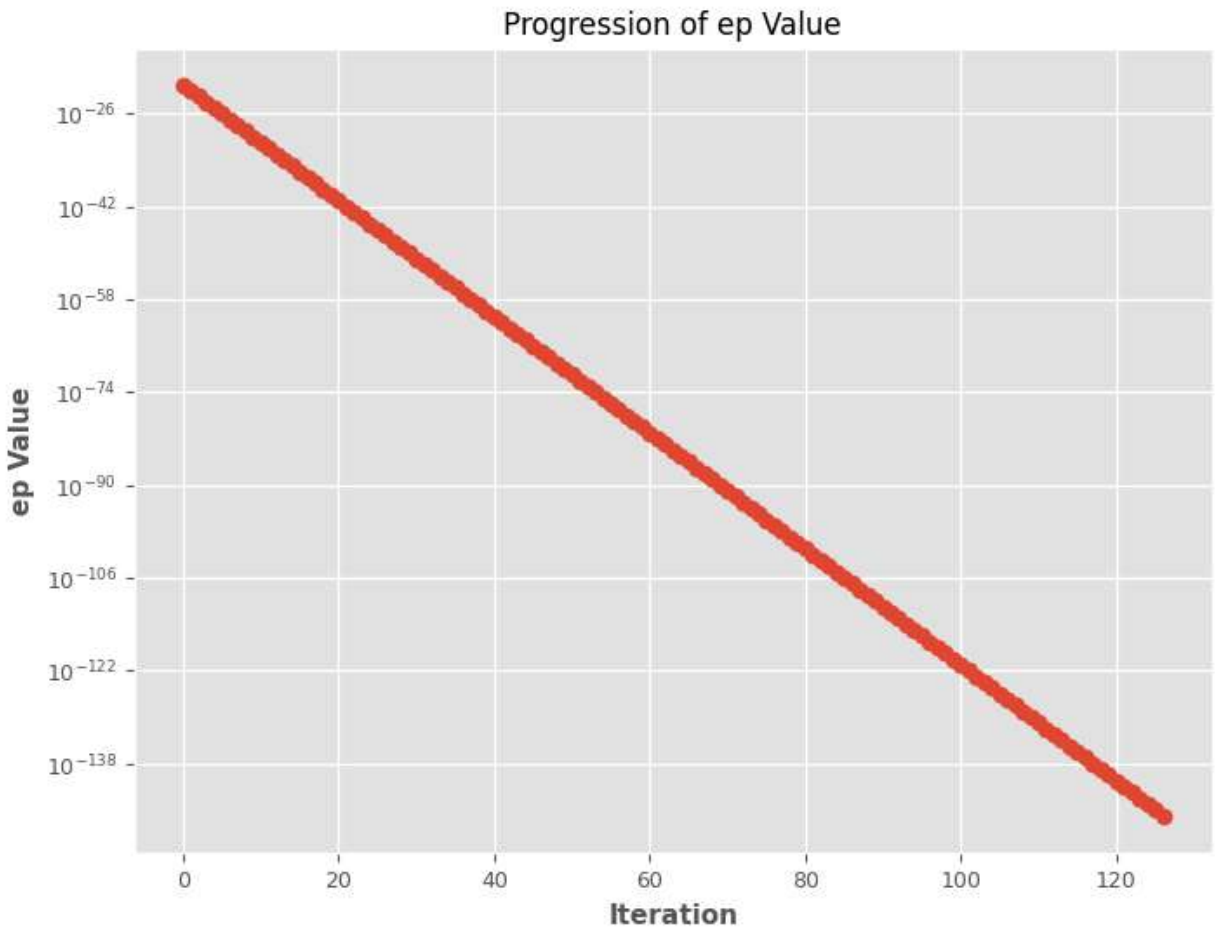
This parameter is dependent on the scenario / video being examined. For example, a high risk sample such as a cop chase would have a higher `percentile_anomaly` whereas a low risk sample such as highway driving will have a lower `percentile_anoaly` .

```
Hyperparameter_3 => percentile_anomaly
# tells the model the portion of the images to filter
```

Currently threshold values are calculated based on this hyper parameter as follows

```
def calculate_ep_val(p, percentage):
    ep = 0.00000000000000000001
    outliers = np.array((p < ep))
    while (outliers.sum() > len(encoded_images)*percentage):
        outliers = np.array((p < ep))
```

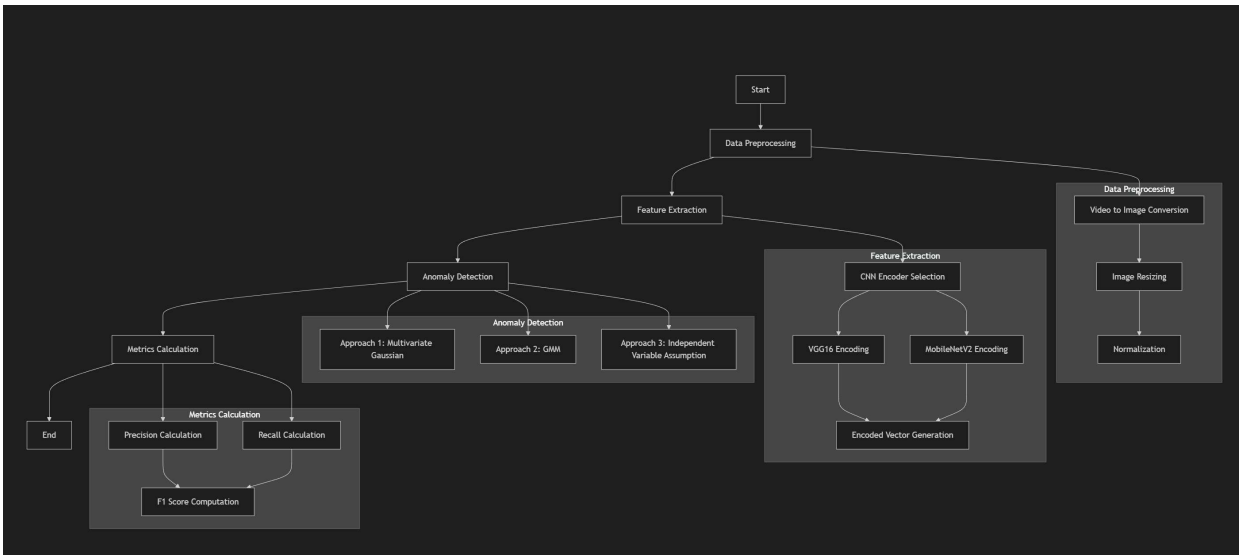
```
        ep = ep / 10
    return ep
```




The image shows a graph titled "Progression of ep Value". It displays the change in the "ep Value" (y-axis) over iterations (x-axis). The y-axis uses a logarithmic scale, ranging from 10^-138 to 10^-26. The x-axis ranges from 0 to 120 iterations.

We intend to automate this parameter as the percentile of anomalies is directly proportional to the variance in the entire dataset. This will be done in the final report.

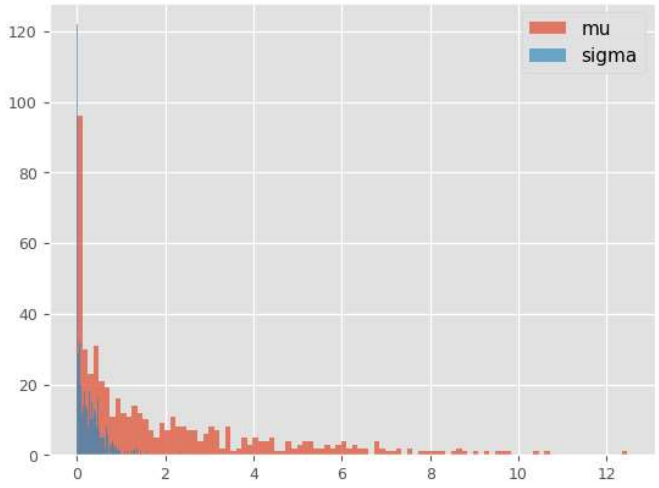
## 3.5 Summary of Methodology




# 4. Results before Clustering

## 4.1 Results for Multivariate Gaussian (assuming mutual independence of features)

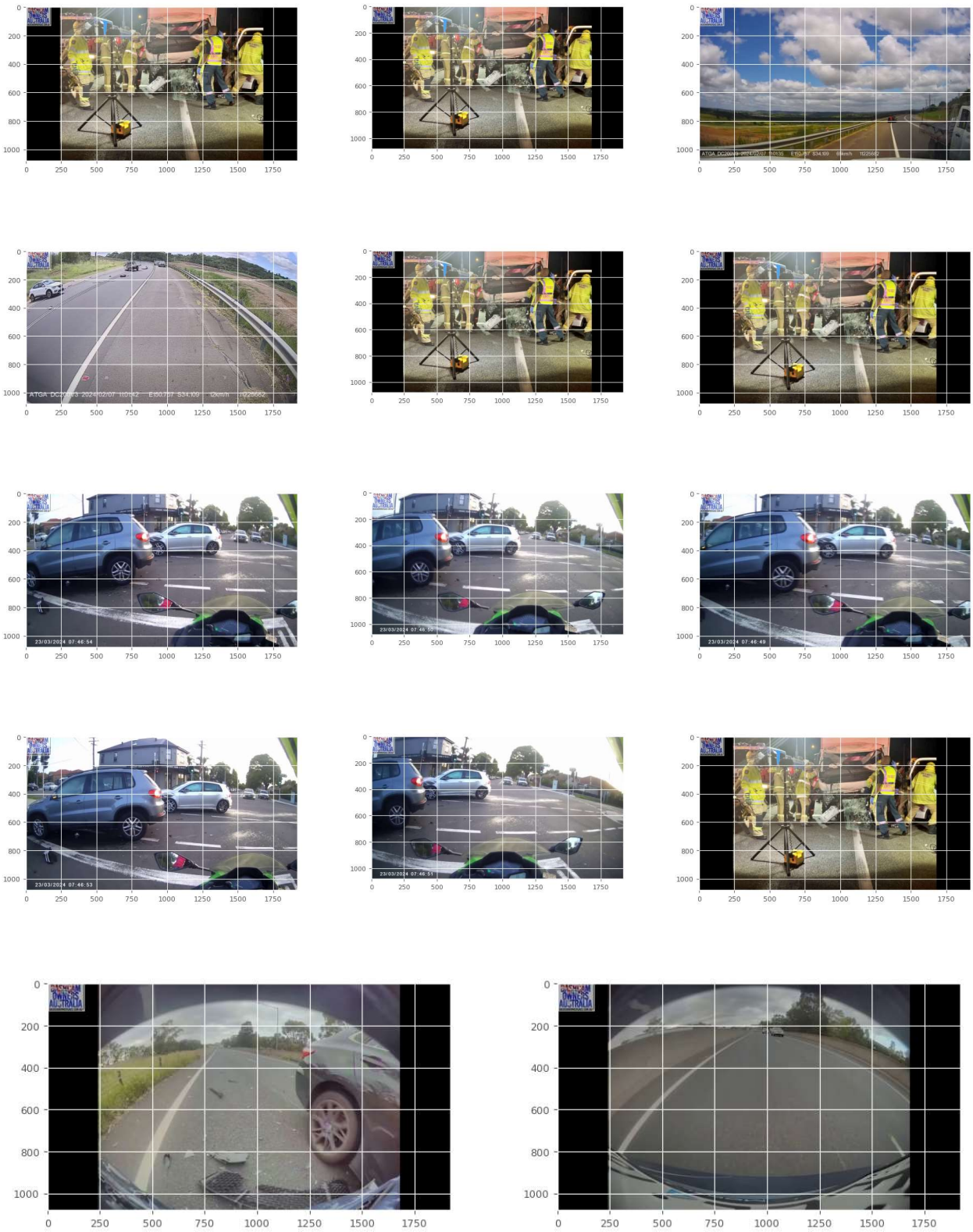
Anomalies were calculated based on the following `mu` and `sigma` values outputted by the model for the encoded features.




Histogram for mean (mu) and variance (sigma)

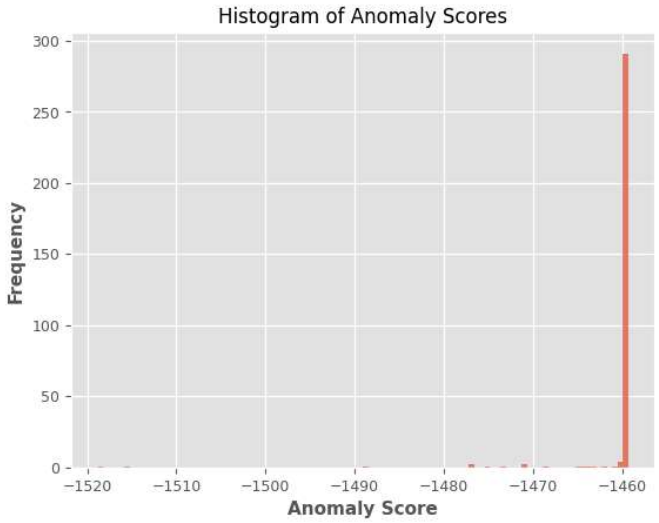
```
Number of anomalies: 14
Outliers indices :  [ 34  98 102 161 163 172 181 203 211 218 232 264 297 299]
```

Anomalous Images from Multivariate Gaussian:



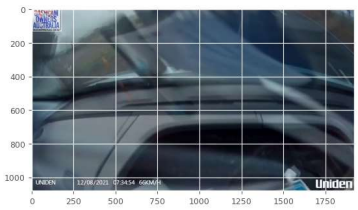
## 4.2 Results for GMM

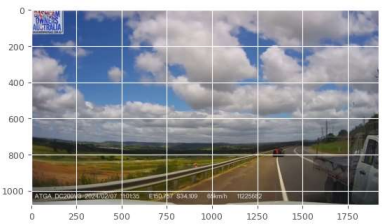
The following graph shows the histogram of anomaly scores predicted by the GMM model.

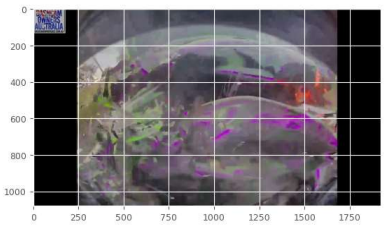


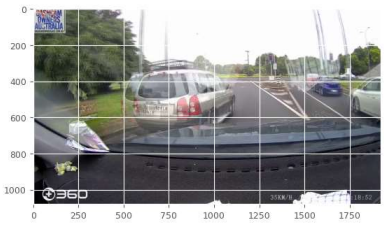
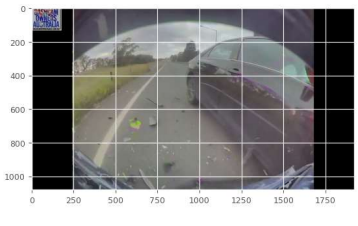
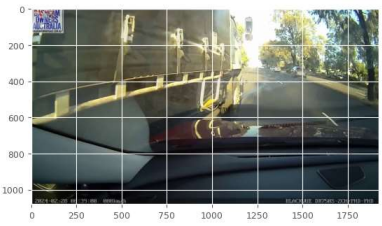
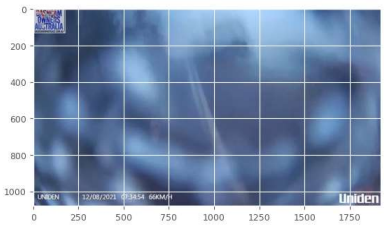
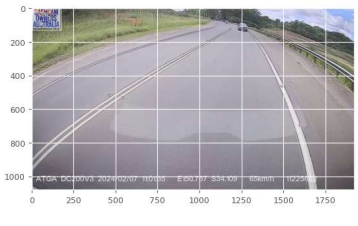
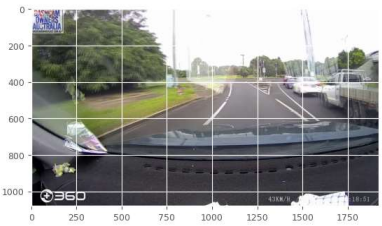
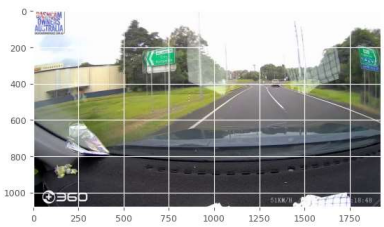

```
Number of anomalies: 16
Indices of anomalous samples: [ 31  79 102 107 151 189 198 226 245 246 248 25
8 293 297 298 305]
```

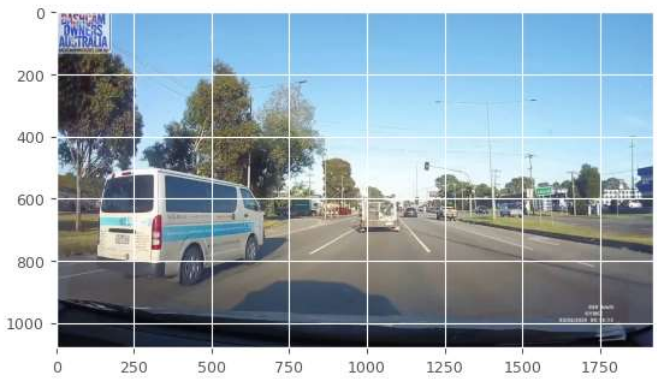
Anomalous Images from GMM:







## 4.3 Result Inference : F1 - Score

The **F1 score** is a measure used to evaluate the performance of a classification model. It combines precision and recall into a single metric. The F1 score is the harmonic mean of precision and recall and ranges from 0 to 1, where 1 is perfect precision and recall.

- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.
- **Recall**: The ratio of correctly predicted positive observations to all observations in the actual class.

We annotated the test video manually and assigned each frame a 1/0 value. 1 means anomaly and 0 means normal frame. We then used this and compared this to the anomalous frames reported by Multivariate Gaussian and the GMM and determined the F1 Scores. We obtained the following F1-Scores:

```
F1 Score GMM: 0.5581395348837209
F1 Score Gaussian: 0.5365853658536586
```

## 4.4 Our intended target metric (True positive vs False Negative)

If the dataset is highly imbalanced with very few anomalies, achieving high F1 scores can be challenging. These scores might be good depending on the baseline performance and the difficulty of the task.

But on closer analysis, we found out that our models predicted more frames as anomalous. This is acceptable to some degree in the case of critical anomaly detection as we may tolerate false positives but we do not want false negatives. Hence we use the following formula:

$$TargetMetric = \frac{TruePositives}{1 + FalseNegatives}$$

**Key Characteristics:**

1. **Range**:
   - The metric is always between 0 and 1:

$$TP \geq 0$$

$$FN \geq 0$$

$$\frac{\mathrm{TP}}{1+\mathrm{FN}} \in [0,1]$$

**What is a Good or Bad Value?**

**Good Value:**

- A **good value** depends on the domain and task, but typically a value closer to 1 (e.g. > 0.8) is generally considered good

**Bad Value:**

- A **bad value** indicates that there are very few true positives or many false negatives. eg: 0.0 to 0.2.

When we calculated the metric after anomaly detection using multivariate Gaussian and GMM for a youtube video , we achieved the following scores.

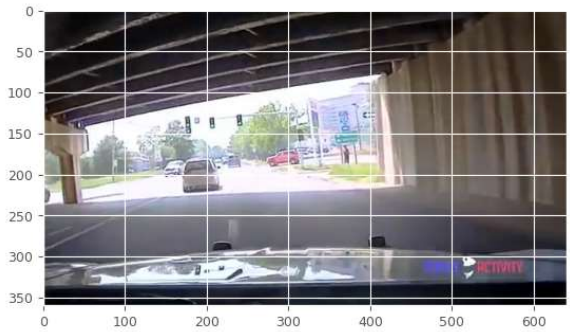
```
New Metric gmm (TP / (1 + FN)): 0.6363636363636364
New Metric gaussian (TP / (1 + FN)): 0.7419354838709677
```

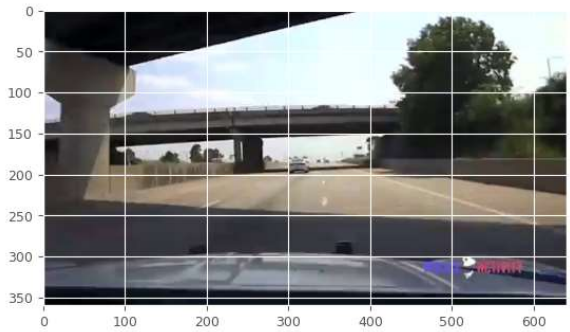
These scores show us that the Multivariate Gaussian performs better than the GMM model. We confirmed this by manually examining the anomalous images that were generated.

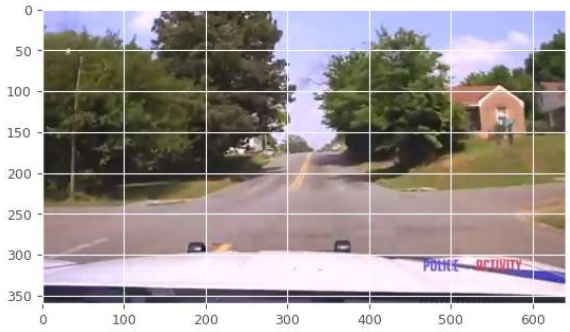
## 4.5 Observations from the results before Clustering

Currently, the intersections and crossings are also flagged as anomalous as those too are rare events. This is okay for small datasets but when the corpus is large, it could be a tedious task to group, classify and remove these anomalous but non critical events.

Example (tunnels and intersections are also tagged as anomalies in the model):






For this, we intend to cluster and classify these classes so that they can be segregated out.

# 5 Clustering to Classify, Group and Weed-out Critical Anomalous Events

Proper detection of anomalous events.

For Example: (representative of the input data)

Typical frame from the video- hours long

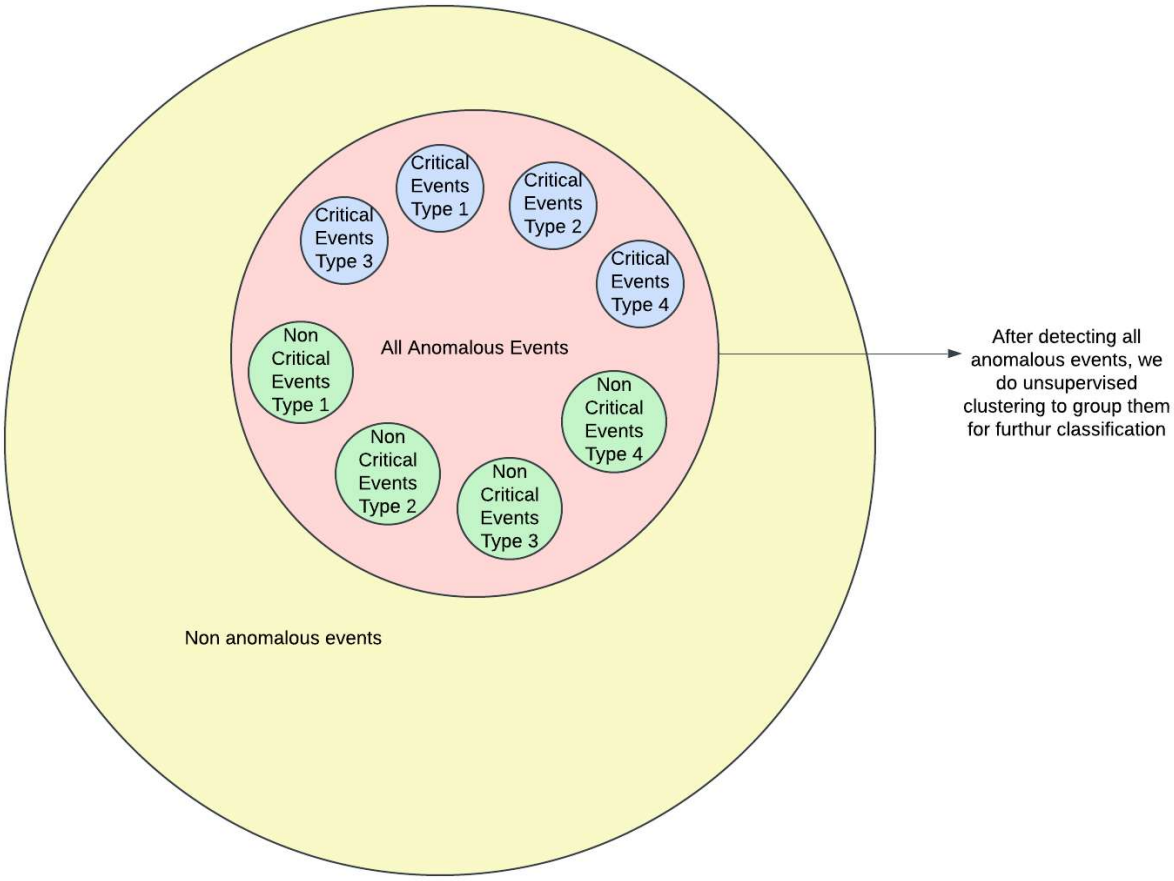
A typical drive data would contain a lot of frames like the one above, which would span a lot of hours, and without many interesting events. Human screening of such data is a tedious and time consuming process.

We expect the pipeline of our models to give clustered anomalous timestamps so that interesting events can be easily identified.

In the context of Anomaly Detection, until the previous steps, we have been filtering images with the least probability. However, it was found that few non-critical events that were rare or just anomalous

were also classified as anomalies such as intersections and driving under bridges.

Thus, we use K - Means to cluster the anomalies into groups, so that non-critical groups can be either flagged out by humans or by automated methods such as Supervised Learning

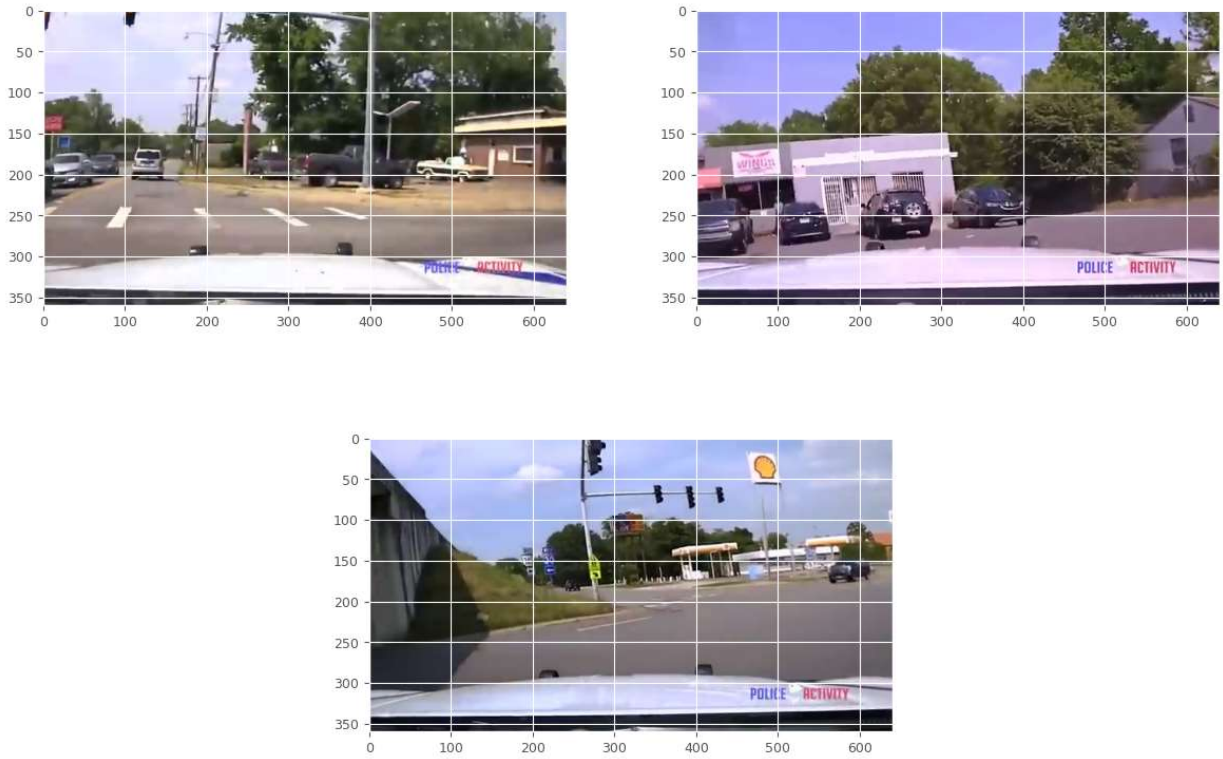



This method proved to be extremely useful in grouping and weeding out the non-critical events that were classified as Anomalies.

# 6 Results with Clustering

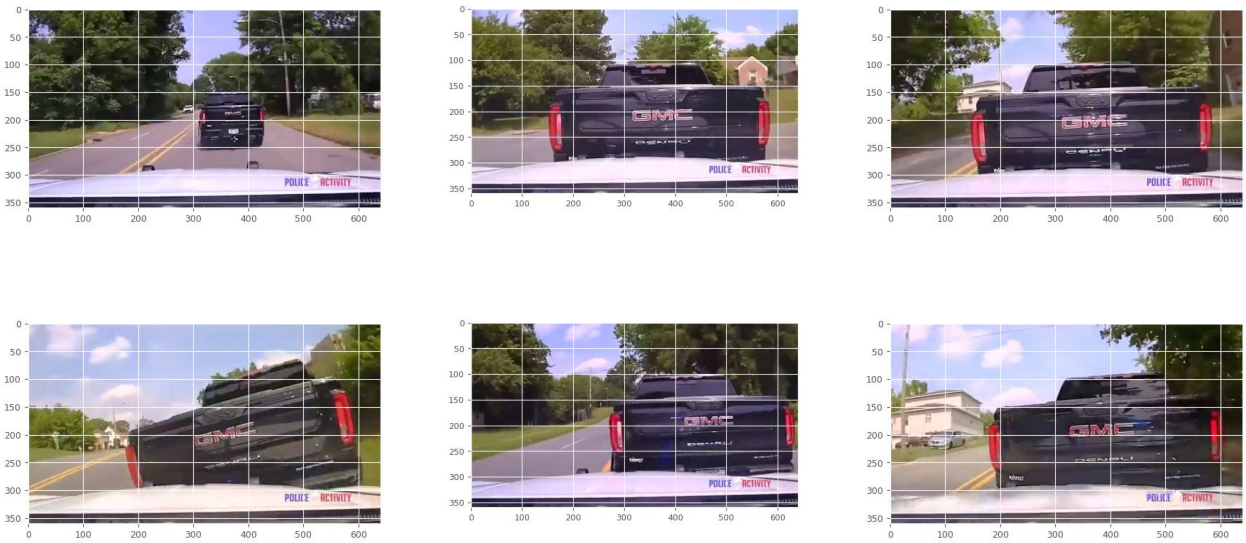
Video Source : https://www.youtube.com/watch?v=I_W4UrZIrV8&t=4s

Cluster 1: Intersections and Turnings (Non-Critical but Highly Informative/risky)



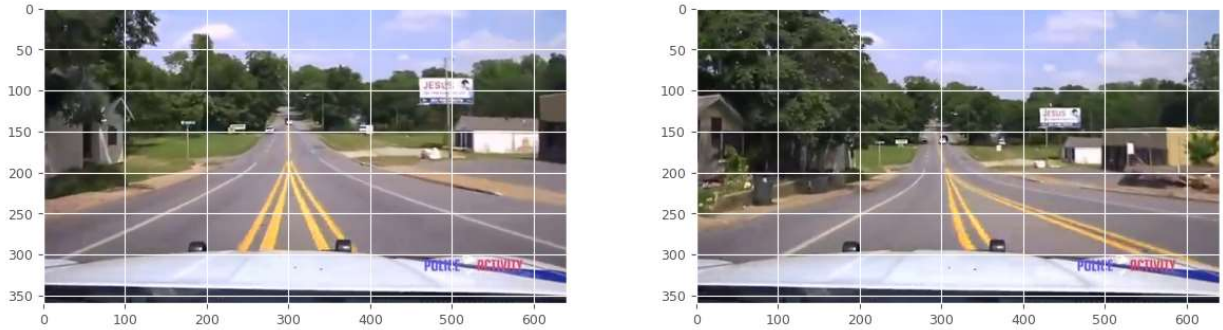
Cluster 2: Tunnels (Non-Critical but informative to check sensor/radar signals)

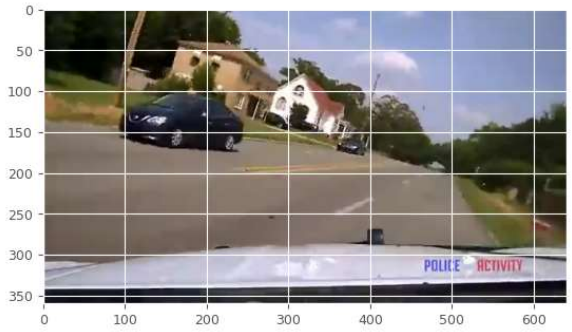
Cluster 3: (Rear Ending Critical Event)

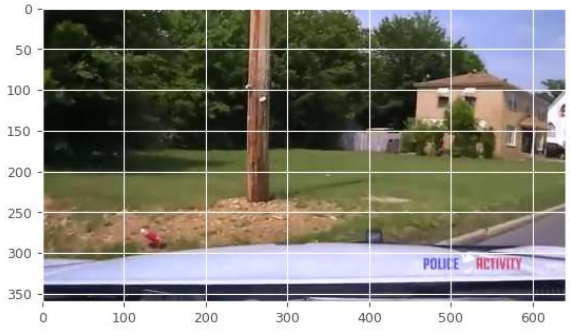


Cluster 4: Highways (Non critical)

Cluster 5: Anomaly in a wooded area (Critical - car is steering off lane and loosing control)






# 7. Discussion and Observation on Results

**Supervised VGG auto encoder fine tuning with our data:**

This approach increases the representation of encoded vectors. This makes the whole pipeline more efficient as the vectors capture more features of the videos for this particular application.

PCA:

Due to PCA,  the encoded images are transformed into a lower-dimensional space where the most significant features are retained. This can lead to a more accurate and efficient entropy calculation. We observed that this improved the performance of the whole pipeline.

**Multivariate Gaussian and GMM Results**

The initial anomaly detection using multivariate Gaussian and GMM approaches showed promising results in identifying unusual events in driving footage. The target metric, defined as TP/(1+FN),

achieved scores indicating reasonable detection capability, though exact values varied between models. Multivariate Gaussian performed the best because it assumes correlation of features while being agnostic to different types of encodings and being less sensitive to varied hyperparameters. The GMMs performed okay, but performance suffered due to the need for calculating n components which was not ideal as the encodings change with each video. The Independent assumption approach failed because the features are correlated.

**Challenges with Initial Detection**

A key limitation emerged where both models flagged non-critical but rare events (like intersections and tunnels) as anomalies. While technically unusual in the dataset, these events don't represent critical safety concerns that require intervention.

**Clustering Enhancement**

The introduction of K-means clustering significantly improved the practical utility of the anomaly detection system by grouping similar events together. This allowed for better discrimination between truly critical anomalies and benign rare events. The clustering revealed distinct groups:

- Cluster 1: Intersections and turnings (non-critical but informative)
- Cluster 2: Tunnels (useful for sensor validation)
- Cluster 3: Critical rear-ending events
- Cluster 4: Highway scenes (non-critical)
- Cluster 5: Lane deviation events in wooded areas (critical)

We have ensured that the encoded vectors have the highest entropy and variance (due to PCA and other preprocessing methods). Hence, this results in proper clustering of anomalous events which on manual examination, can be classified into different types of critical events.

**Model Performance Analysis**

The combination of anomaly detection with clustering proved more effective than either approach alone:

- Multivariate Gaussian showed strength in initial anomaly identification but suffered from high false positives
- GMM provided better handling of complex data distributions
- K-means clustering successfully separated critical from non-critical anomalies, making the system more practical for real-world applications

These results suggest that while pure statistical anomaly detection is valuable, the addition of clustering provides necessary context and categorization that makes the system more practical for real-world autonomous driving applications.

### Next Steps:

1. Running the code on every single video of the Car-Crash Dataset (we have executed the code for a subset)
2. Write functions to automate hyperparameter calculation based on variance in frames.
3. Use more autoencoders such as Resnet and DenseNet in addition to Vgg16 and mobilenetv2
4. Evaluate the clusters (This is not possible as dataset does not have labels for various critical event tags).
5. Auto-detect the criticality in each cluster - use metadata other than videos such as sensor signals from car if available to evaluate a criticality score for each anomaly cluster.

## 8 References

[1]Haresh, S., Kumar, S., Zia, M. Z., & Tran, Q.. Towards Anomaly Detection in Dashcam Videos. *Towards Anomaly Detection in Dashcam Videos*. 2020. https://doi.org/10.1109/iv47402.2020.9304576

[2] Ullah, W., Ullah, A., Haq, I. U., Muhammad, K., Sajjad, M., & Baik, S. W.. CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks. *Multimedia Tools and Applications*, *80*(11), 16979–16995. 2020. https://doi.org/10.1007/s11042-020-09406-3

[3] Chatterjee, A., & Ahmed, B. S. IoT anomaly detection methods and applications: A survey. *Internet of Things*, *19*, 100568. 2022. https://doi.org/10.1016/j.iot.2022.100568

[4] *Car Crash Dataset (CCD)*. Kaggle. July, 2022. https://www.kaggle.com/datasets/asefjamilajwad/car-crash-dataset-ccd/data

[5] Toohey, J. R., Raunak, M. S., & Binkley, D. From neuron coverage to steering angle: Testing autonomous vehicles effectively. *Computer*, *54*(8), 77–85. 2021. https://doi.org/10.1109/mc.2021.3079921

[6] Duong, H., Le, V., & Hoang, V. T. Deep Learning-Based Anomaly Detection in Video Surveillance: A survey. *Sensors*, *23*(11), 5024. 2023. https://doi.org/10.3390/s23115024

[7] Yoon, J., Sohn, K., Li, C., Arik, S. O., & Pfister, T. *SPADE: Semi-supervised Anomaly Detection under Distribution Mismatch*. arXiv.org. November 2022. https://arxiv.org/abs/2212.00173

[8] Zhang, J., Liu, H., & Lu, J. A semi-supervised 3D object detection method for autonomous driving. *Displays*, *71*, 102117. 2021. https://doi.org/10.1016/j.displa.2021.102117

# 9. Gantt Chart:

https://gtvault.sharepoint.com/:x:/r/sites/CS7641MachineLearningProjectTeam/_layouts/15/Doc.aspx?sourcedoc={C6BEB847-8FD0-443B-AD44-94FB2CC3BE18}&file=GanttChart_Project_Final_Report - Copy.xlsx&action=default&mobileredirect=true&wdOrigin=OUTLOOK-METAOS.FILEBROWSER.FILES-HOME

# 10. Contribution Table:

| Name | Final Contributions |
|---|---|
| Karatattu Padmanabha, Sathvik | Calculation of Entropy, function for MobileNetV2 encoder, GMM, implementing environment on PACE with GPU, F1-score, PCA, windowing mechanism. |
| Karunanidhi, Niraj Kamal | Main code logic, VGG16, Multivariate Anomaly Detection, Compute Probability Function, Encoding, visualization of features, K-means |
| Namburi, Meghana Chowdary | Script for dataset loading, supervised training for vgg16, Optimizing epsilon value, filtering logic, Final Report writing, state of the art review |
| Sinha, Ankit | Reviewing existing anomaly detection models, code refactoring, CCD dataset processing script |
| Yadav, Dinesh | Video Annotation, testing, debugging entropy, code refactoring. final testing |