Abulele Mditshwa

abulele@capetownai.com

**Traffic Camera Anomaly Detection with Autoencoders and Classifiers**

Table of Contents

[Abstract 2](#_Toc22190057)

[1. Introduction 3](#_Toc22190058)

[2. Theoretical Aspects 4](#_Toc22190059)

[2.1 Autoencoders 4](#_Toc22190060)

[2.1.1 Architecture of Autoencoders 4](#_Toc22190061)

[2.1.2 Convolutional Autoencoders 5](#_Toc22190062)

[2.1.3 Classification 5](#_Toc22190063)

[3. Methodology 6](#_Toc22190064)

[3.1 Data Pre-processing 6](#_Toc22190065)

[Fig. 1 Proposed Methodology. 7](#_Toc22190066)

[3.2 Feature Extraction 7](#_Toc22190067)

[Fig 2. Training and Validation Loss. 7](#_Toc22190068)

[Fig 3. Reconstructed Normal Images. 8](#_Toc22190069)

[Fig 4. Reconstructing images not in training set. 9](#_Toc22190070)

[Fig 5. Model Architecture. 10](#_Toc22190071)

[3.3 Classification Method. 11](#_Toc22190072)

[Fig 6. Classification Model 3.4 Model Evaluation. 12](#_Toc22190073)

[4. Experiments 14](#_Toc22190074)

[4.1 Datasets 14](#_Toc22190075)

[Fig 7. Sample images (LargeTraining400K). 14](#_Toc22190076)

[Fig 8. Sample images (Camera 22528). 14](#_Toc22190077)

[Fig 9. Sample images (Camera 31006). 15](#_Toc22190078)

[Fig 10. Sample images (Camera 16369). 15](#_Toc22190079)

[5. Results 16](#_Toc22190080)

[Table 1. Classification Report of Generic Model. 16](#_Toc22190081)

[Fig 11. Confusion Matrix (Generic Model). Summary of predictive results. 16](#_Toc22190082)

[Table 2. Classification Report of Camera **22528**. 17](#_Toc22190083)

[Fig 12. Confusion Matrix of Camera **22528**. 17](#_Toc22190084)

[Conclusion 20](#_Toc22190085)

[Glossary 21](#_Toc22190086)

[References 22](#_Toc22190087)

# Abstract

With the growth of image data generated by surveillance cameras, automated camera analysis has become necessary to detect unusual events. Deep learning methods have achieved the state-of-the-art results in many tasks related to computer vision.

This article aims to elaborate the use of techniques such as Autoencoders and classifiers for anomaly detection to help identify unusual events in cameras in real time. This will describe how we’ve used autoencoders and classifiers to detect faulty cameras

**Keywords.**

*Autoencoders, Deep Learning, Anomaly detection, Classification Report, Confusion Matrix and ROC curve.*

# Introduction

The amount of data generated by surveillance systems has grown over the years due to the decreasing cost of image capturing devices and the elevated concern with security. The volume of data grew much faster than the availability of human staff to observe the quality of the cameras, this led to a serious problem. A solution to this problem is the development of automated camera surveillance systems.

Computer vision typically refers to a discipline of giving machines the ability of sight, enabling machines to visually analyse their environments and the stimuli within them [1]. We have used Autoencoders, a well-known technique in the field of computer vision to detect anomalies in camera images. Autoencoders are unsupervised neural networks that learn features of an input image and outputs a reconstructed image of the input image [2], that may sound a bit useless when compared to other algorithms but they unique difference is in anomaly detection systems. They learn to recognize what is normal and anything that deviates from the norm will be an outlier. The approach to detect the outliers in camera images is by training the autoencoder with data containing only normal situations which can then be used to detect deviations.

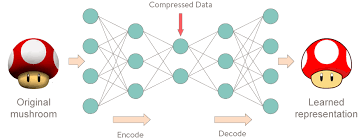
When the autoencoder reconstructs the input image they have a reconstruction error which carries information that can discriminate between a normal and anomalous image. Smaller reconstruction errors are expected of normal images since they appear a lot in the training set and higher reconstruction errors are assumed to be anomalies based on the fact that the autoencoder failed to reconstruct them.

To test this hypothesis, this work presents a Convolutional Autoencoder anomaly detection model, the goal is to determine whether a traffic camera is taking normal or anomalous images by using a classifier. The models are trained, tested and evaluated on real traffic image datasets.

# Theoretical Aspects

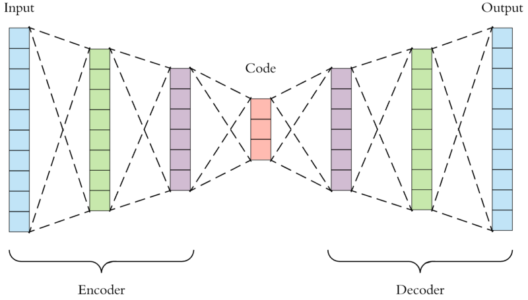
## Autoencoders

Autoencoders are machine learning algorithms where the predicted output is the input data. An input image is fed to a model and the model learns the representations of that input image, then produces an output that is like the input image.



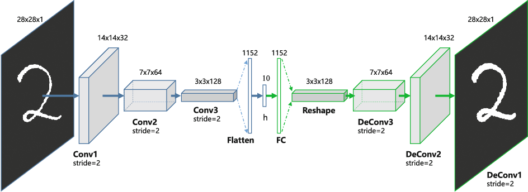
### Architecture of Autoencoders

* **Encoder:** This part of the network compresses the input image into a latent space representation [3].
* **Code (Bottleneck):** Represents the compressed input which is fed to the Decoder.
* **Decoder:** Decodes the encoded image back to the original dimension (for example 160x160).



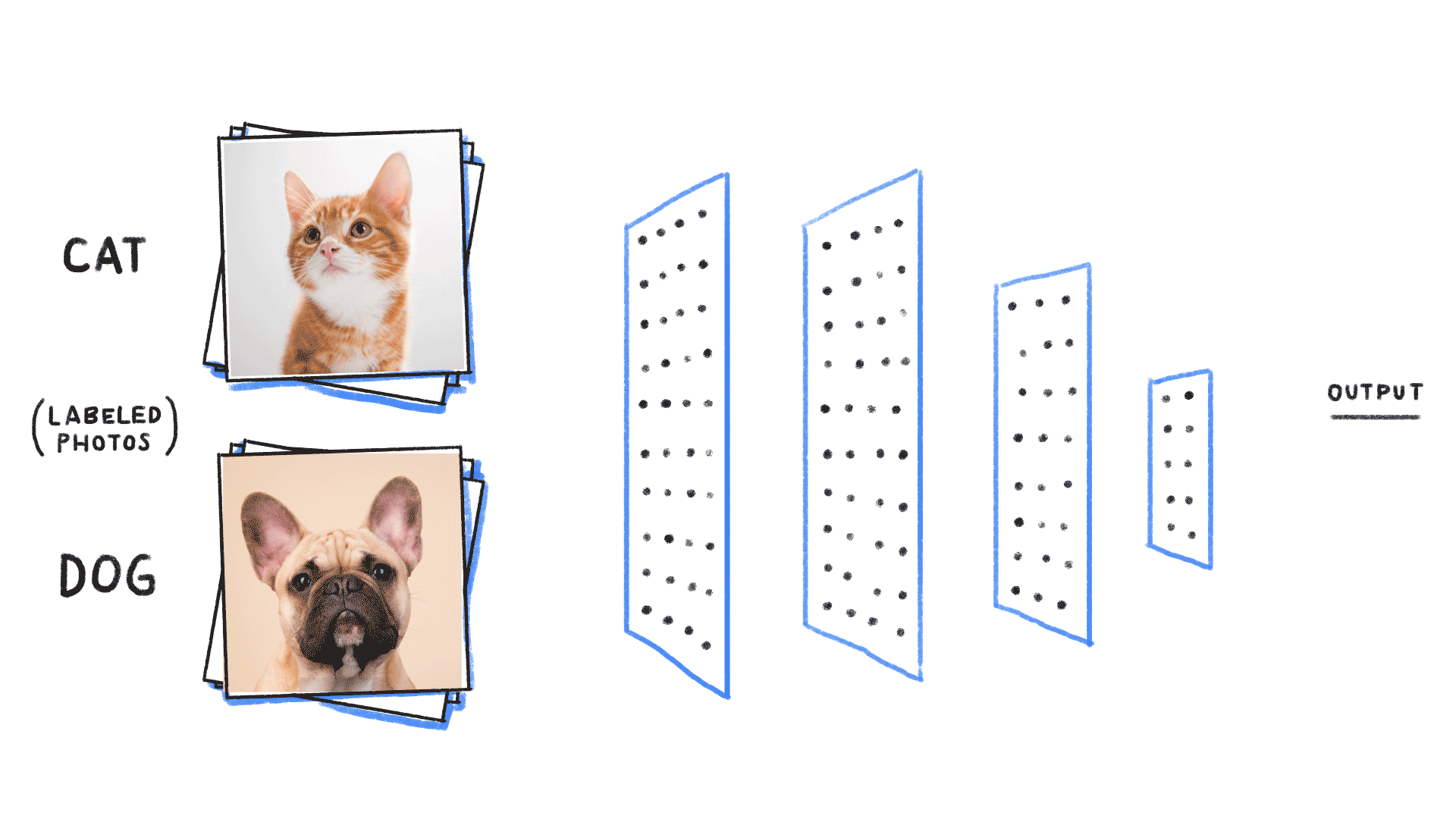
### Convolutional Autoencoders

Since our input are images its best to use convolution neural networks [4], convolutional autoencoders use convolution operators as encoders and decoders.



### Classification

Classifiers are machine learning algorithms that can predict a class label (cat vs dog).



In this step you load the weights of the trained autoencoder model into a few layers of the new model which is the classifier. You must make few layers of the new model false and compile the new classification model, train and evaluate.

# Methodology

In this case the camera anomaly detection model is divided in two main categories, the autoencoder and the classifier. In the autoencoder, the goal is to learn the features of a normal images and the goal of the classifier to classify anything that deviates from what’s normal as an anomaly. Figure 1 shows the proposed methodology.

## Data Pre-processing

To prepare the data, each dataset was treated separately for each camera (Unique camera ID’s) and a separate generic dataset was made, this is dataset that combines multiple camera datasets into one large dataset. The model takes input images as 160x160 pixels, this is then fed to an autoencoder which consists of 8 convolutional layers (Conv2D) where each layer is followed by Batch Normalization and Max-Pooling layers [5], this is just the encoder part of the autoencoder. The decoder consists of 8 Conv2D layers where each is followed by Batch-Normalization and Up- Sampling layers, Up Sampling is a technique that Up Samples images to a higher resolution [6]. These are the layers that are reconstructing the images that are fed into the network.

Data Preprocessing

75% Normal

25% Anomaly

Train the new classifier model

Load weights to few layers on new model

Train Convolutional Autoencoder

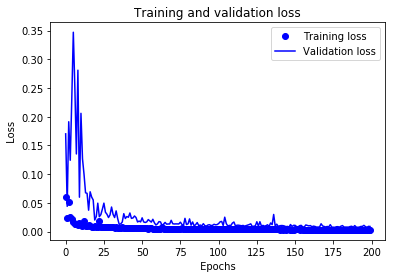
### Fig. 1 Proposed Methodology.

## Feature Extraction

The feature extraction process consists of two steps. The first step is training the CAE (Convolutional Autoencoder) using the normal dataset. CAE’s are trained separately for each dataset; the training processing stops after 200 epochs. The architecture is presented in figure 5, ReLU (Rectified Linear Unit) activation function [7] is used after convolution and deconvolution layers. Training was done using Adam [8] optimizer algorithm and a fixed learning rate of 0.00001. This network was implemented using the Keras framework using Tensorflow as the backend [9] and trained using GPU (Graphics Processing Unit), NVIDIA GeForce RTX 2060. The second step is to forward both the train and validation sets throughout the network to capture the reconstruction loss.

### Fig 2. Training and Validation Loss.

The training and validation loss are usually the same for each dataset, this determines how well the network can reconstruct input images.

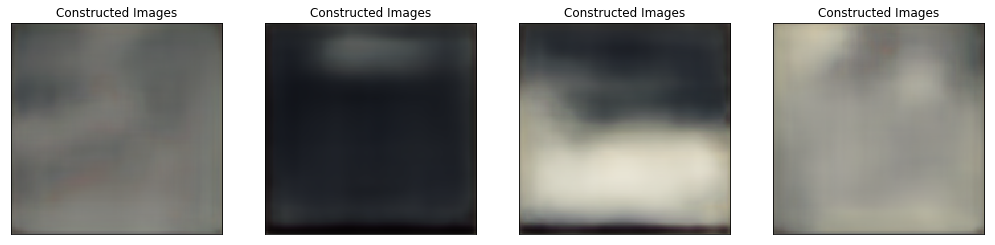


In theory the network should easily be able to reconstruct images it has seen before and struggle to reconstruct images that it hasn’t seen (not included in training set), figure 3 gives us and illustration of that and figure 4 show us what happens if we feed something else to the network.

### Fig 3. Reconstructed normal Images.



### Fig 4. Reconstructing images not in training set.



As you can see the for yourself, the network easily reconstructed normal images but largely failed to reconstruct images that it has not seen. This is what makes autoencoders unique and different from other computer vision techniques

### Fig 5. Model Architecture.

## Classification Method.

To perform classification, we use encoder part of the autoencoder and ignore the decoder (won’t need to reconstruct images), this then allows us create and define a fully connected layer that is made by stacking up with the encoder function. The next step is to Flatten the layer and use a Dense layer with your activation functions (ReLU), I then added a Regularization layer (Dropout) to attack overfitting the model. For the last layer the activation function is a softmax algorithm and returns 2 classes. Before you start training the new model, you have to freeze all the layers of the encoder part of the model and you can do that by setting those layers to **False.** Encoder part is already trained so the training will only be the fully connected part.

## Fig 6. Classification Model 3.4 Model Evaluation.

To evaluate the models, the area under the Receiver Operating Characteristic (ROC) curve and the Precision Recall curves were computed using the test set for each dataset. ROC curves summarize the trade-off between the true positive rate and the false positive rate for a predictive model using different thresholds, this helps us know how good or bad is the accuracy of our models. The confusion matrix and classification report will both be taken into consideration as well for model evaluation.

### 4. Experiments

The methodology proposed in Section 3 was applied to 5 different camera (Single camera) datasets and 2 generic (consists of many camera’s) datasets, each of these datasets has their own train, validation and test sets with ground truth for evaluation.

## 4.1 Datasets

**LargeTraining400K (Generic model):** This dataset consists of 300K training images and 100K testing images (validation and Testing included). This dataset does contain a small number of anomalies in the training set, but the dataset mostly consists of normal images. Figure 7 displays sample images of normal and abnormal images found in the dataset.

### Fig 7. Sample images (LargeTraining400K).

A car parked on the side of a road

Description automatically generated A car parked on the side of a road

Description automatically generated A car driving down a dirt road

Description automatically generated

**Camera 22528:** This dataset consists of 42K training images and 8K testing images (validation and Testing included). Training set contain no anomalies they are camera specific. Figure 8 displays sample images of normal found in the dataset.

### Fig 8. Sample images (Camera 22528).

A red car parked in a parking lot

Description automatically generated A car parked on the side of a road

Description automatically generated A car parked on the side of a road

Description automatically generated

**Camera 16433:** This dataset consists of 45K training images and 10K testing images (validation and Testing included). Training set contain no anomalies they are camera specific. Figure 8 displays sample images of normal found in the dataset

### Fig 9. Sample images (Camera 16433).

A car parked on the side of a road

Description automatically generated A car parked in front of a building

Description automatically generated A sign on the side of a road

Description automatically generated

**Camera 16369:** This dataset consists of 93K training images and 37K testing images (validation and Testing included). Training set contain no anomalies they are camera specific. Figure 8 displays sample images of normal found in the dataset

### Fig 10. Sample images (Camera 16369).

A car parked on the side of a road

Description automatically generated A car parked on the side of a road

Description automatically generated A car parked on the side of a road

Description automatically generated

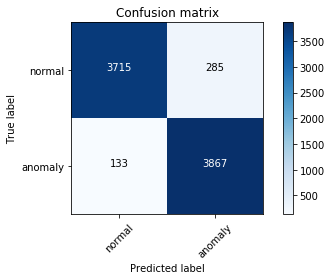
# 5. Results

The results reported in this section regards only the classification using different classification thresholds (0.1 – 1.0), which is a point of balance between True positive rates (TPR) and False positives rates (FPR). The figures below display different ROC, precision-recall curves for all the models. The best results were obtained using camera specific (22528) dataset, although one of the problems with this method is that you have to train a model each time a new camera is setup which might take up some time to setup. So far we haven’t gathered large enough data from all cameras and train one very large autoencoder and classifier, I think this would be an ideal state if we had to push these models to production.

## Table 1. Classification Report of Generic Model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 Score** | **Support** |
| Normal | 0.97 | 0.93 | 0.95 | 4000 |
| Anomaly | 0.93 | 0.97 | 0.95 | 4000 |
| Accuracy |  |  | 0.95 | 8000 |
| Macro Average | 0.95 | 0.95 | 0.95 | 8000 |
| Weighted Average | 0.95 | 0.95 | 0.95 | 8000 |
|  | | | | |

## Fig 11. Confusion Matrix (Generic Model). Summary of predictive results.



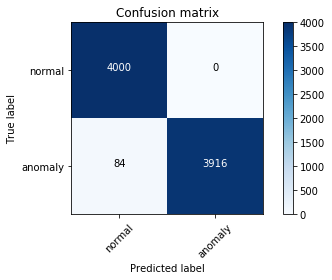
**True Positive**

**False Positive**

## Table 2. Classification Report of Camera **22528**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 Score** | **Support** |
| Normal | 0.98 | 0.99 | 0.99 | 4000 |
| Anomaly | 0.93 | 0.97 | 0.95 | 4000 |
| Accuracy |  |  | 0.98 | 8000 |
| Macro Average | 0.97 | 0.97 | 0.97 | 8000 |
| Weighted Average | 0.97 | 0.97 | 0.97 | 8000 |
|  | | | | |

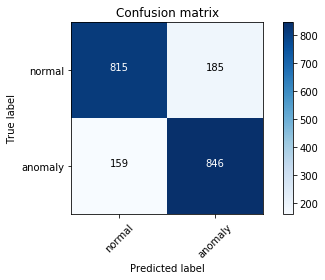
## Fig 12. Confusion Matrix of Camera **22528**.



## Table 3. Classification Report of Camera **22528-all\_classifier**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 Score** | **Support** |
| Normal | 0.84 | 0.81 | 0.81 | 1000 |
| Anomaly | 0.82 | 0.82 | 0.83 | 1005 |
| Accuracy |  |  | 0.83 | 2005 |
| Macro Average | 0.83 | 0.83 | 0.83 | 2005 |
| Weighted Average | 0.83 | 0.93 | 0.93 | 2005 |
|  | | | | |

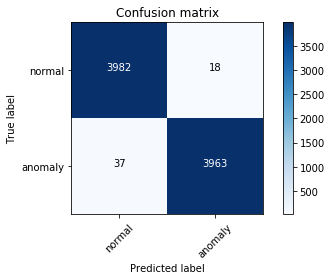
## Fig 13. Confusion Matrix of Camera **22528-all\_classifier**.



## Table 4. Classification Report of Camera **16369**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 Score** | **Support** |
| Normal | 0.98 | 0.99 | 0.99 | 4000 |
| Anomaly | 0.93 | 0.97 | 0.95 | 4000 |
| Accuracy |  |  | 0.98 | 8000 |
| Macro Average | 0.97 | 0.97 | 0.97 | 8000 |
| Weighted Average | 0.97 | 0.97 | 0.97 | 8000 |
|  | | | | |

## Fig 14. Confusion Matrix of Camera 16369 .



## Table 5. Classification Report of Camera **16433**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F1 Score** | **Support** |
| Normal | 0.92 | 0.98 | 0.96 | 4000 |
| Anomaly | 0.98 | 0.91 | 0.95 | 4000 |
| Accuracy |  |  | 0.95 | 8000 |
| Macro Average | 0.96 | 0.95 | 0.95 | 8000 |
| Weighted Average | 0.96 | 0.95 | 0.95 | 8000 |
|  | | | | |

## Fig 15. Confusion Matrix of Camera **16433** .

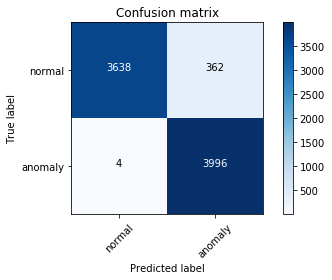


Table 6. Performance summary of models.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | F1 Score | AUC | Average Precision |
| Generic Model | 0.949 | 0.957 | 0.917 |
| Camera 22528 | 0.989 | 0.951 | 0.960 |
| Camera 22528-all\_classifier | 0.831 | 0.871 | 0.770 |
| Camera 16369 | 0.993 | 0.995 | 0.995 |
| Camera 16433 | 0.956 | 0.958 | 0.917 |
| Total Avg | 0.943 | 0.946 | 0.901 |

Table 7. Average Confusion matrix (all models combined).

|  |  |  |
| --- | --- | --- |
|  | Predicted Labels | |
| Normal | Anomaly |
| Normal | 3230 | 170 |
| Anomaly | 83 | 3318 |

# Conclusion

# Glossary

## Autoencoders

Autoencoders are machine learning algorithms where the predicted output is the input data. An input image is fed to a model and the model learns the representations of that input image, then produces an output that is like the input image.

## Anomaly Detection

Anomaly detection is the process of identifying unexpected items or events in datasets, which differ from the norm. Anomaly detection is often applied on unlabelled data which is known as unsupervised anomaly detection.

## Classification Report

Classification reports provide convenient reports when working on classification problems. They display the precision, recall, F1 score and support for each class.

## Confusion Matrix

The confusion matrix is a handy presentation of the accuracy of a model with two or more classes, its job is to summarize all the predictive results.

## ROC Curve

ROC curves summarize the trade-off between the true positive rate and the false positive rate for a predictive model using different classification thresholds.

# References

[1]. Benjamin F. Duffy 2017, A Year in Computer Vision, The M Tank, 15 Oct. 19, <https://www.themtank.org/a-year-in-computer-vision>

[2]. Ian GoodFellow and Yoshua Bengio and Arron Courville, (2016). *Deep Learning,* MIT Press.

[3]. Sayantini May 2019, Autoencoders Tutorial, Edureka, 15 Oct. 19,

<https://www.edureka.co/blog/autoencoders-tutorial/>

[4]. Sumit Saha 2018, A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way, Towards DataScience,16 Oct. 19, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[5]. Jason Brownlee PhD 2018, How to Accelerate Learning of Deep Neural Networks with Batch Normalization, Machine Learning Mastery, 16 Oct. 19, <https://machinelearningmastery.com/how-to-accelerate-learning-of-deep-neural-networks-with-batch-normalization/>

[6]. Matthew Stewart PhD Researcher 2018, Advanced Topics in Convolutional Neural Networks, Towards DataScience, 16 Oct. 19, <https://towardsdatascience.com/advanced-topics-in-deep-convolutional-neural-networks-71ef1190522d>

[7]. Dang Lui 2017, A Practical Guide to ReLU , Medium, 16 Oct. 19, <https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7>

[8]. Jason Brownlee PhD 2018, Gentle Introduction to the Adam Optimization Algorithm for Deep Learning, Machine Learning Mastery, 16 Oct. 19, <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

[9]. Tensorflow 2019, Keras, 16 Oct. 19, <https://www.tensorflow.org/guide/keras>