**CS 4850** 

**Rare Event Detection in Machine Vision   
using Deep Learning**

Software Design Document

Version 1.0

Group 16

Roberta Beaulieu,Rakeem Durand,Thomas Fuller,Ihsan Hashem,  
Karim Rattani

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 02/27/2018 | <1.0> | Initial Version of Document | Roberta Beaulieu |
| 03/02/2018 | <1.0> | Worked on the different sections | Thomas Fuller |
| 03/02/2018 | <1.0> | Worked on the different sections | Karim Rattani |

**Table of Contents**

[**1 Introduction**](#_30j0zll) **5**

[1.1 Purpose](#_1fob9te) 5

[1.2 Scope](#_3znysh7) 5

[1.3 Definitions, Acronyms, and Abbreviations](#_2et92p0) 5

[1.4 References](#_tyjcwt) 5

[**2 Design Overview**](#_236cm616jbc) **6**

[2.1 Introduction](#_14pguix7f1) 6

[2.2 System Architecture](#_szg5d5h5sx14) 6

[2.3 System Interfaces](#_vqwvkti6l0ht) 6

[2.4 Constraints and Assumptions](#_e4akbqkp0j3x) 7

[**3 System Object Model**](#_1t3h5sf) **7**

[3.1 Introduction](#_u314u7opkp5g) 7

[3.2 Subsystems](#_xv7gnwrvk9ip) 7

[3.3 Subsystems Interfaces](#_tv9n31ly1h5i) 8

[**4 Object Descriptions**](#_3rdcrjn) **9**

[4.1 Objects in Subsystem 1](#_661lcy210s8z) 9

[4.2 Object in Subsystem 2](#_drix8qt1zwex) 10

[**5 Object Collaboration //Ihsan**](#_35nkun2) **12**

[5.1 Objects in Subsystem 1](#_9684mepz2mgf) 12

[5.2 Objects in Subsystem 2](#_7iyb8tkt3zb2) 12

[**6 Dynamic Model**](#_2jxsxqh) **13**

[6.1 State Diagram](#_lygdc79039it) 13

[6.2 Sequence Diagrams](#_209fgrp9cw8o) 13

[**7 Data Design**](#_qvm6b4xvutyk) **15**

[**8 Non-functional Requirements**](#_1y810tw) **18**

[**9 Supplementary Documentation**](#_y0tu0dkpztd9) **18**

**Software Design Document**

# 1 Introduction

This section gives a scope description and overview of the information include in this Software Design Document (SDD). As well, this document describes the purpose and includes a table with definitions, acronyms and abbreviations.

## 1.1 Purpose

Surveillance cameras have become a part of our life. We see them in almost every corner of the street and outside restaurants. However, hiring people to monitor all the surveillance video is expensive. The purpose of our project is to implement an algorithm that will detect abnormal activity such as robbery, accident, etc. and alert the corresponding authorities.

## 1.2 Scope

The scope of this project is to create a system that will detect anomalous events in images and videos using deep learning. In the subsequent release, we will design a classification method to classify the event. After that, we will design an alert system to alert the corresponding person about anomalous event

## 1.3 Definitions, Acronyms, and Abbreviations

**Table 1- Definitions**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| CNN | Convolutional Neural Network |
| Neural Network | use for solving pattern recognition problem, involves large number of processes operating in parallel and arranged in tiers. |
| CNN | algorithm of Neural Network, analyze visual imagery. |
| Pixel | Smallest unit of picture |
| RGB | Red, Green, Blue Computer |
| Vision | algorithms for understanding of digital image |
| Anomalous | abnormal, unusual |

## 

## 1.4 References

[1] Kennesaw State University. (2018). Senior project course syllabus. Marietta, Georgia: Shi, Yong.

[2] Sultani, Waqas, et al. “Real-World Anomaly Detection in Surveillance Videos.” *Real-World Anomaly Detection in Surveillance Videos*, University of Central Florida, 12 Jan. 2018. Web.

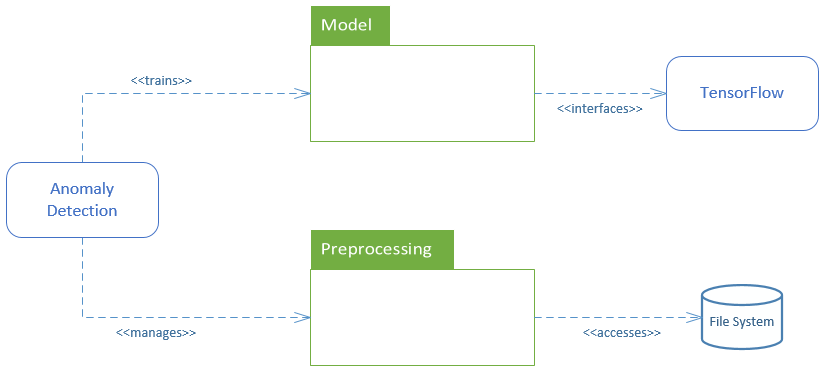
[3] Stahl, Bernd Carsten. “Identifying Ethical Issues during the Development of a Computer Vision Based 2AmI System: A Case Study.” *University of Salzburg*, 2009.

# 2 Design Overview

## 2.1 Introduction

The design of the anomaly detection system can be broken down into two main parts: the model component and the data preprocessing component. The model’s role is to interface with TensorFlow, a popular machine learning library, as well as manage a number of trained models. The preprocessing component’s role is to interact with the file system, storing and organizing the videos in a dataset. The Anomaly Detection class uses both components in order to train, test, and run the anomaly detection model.

## 2.2 System Architecture



## 2.3 System Interfaces

2.3.1 Command line Interface

This team plans to implement a CLI application that allows a user to train or run a anomaly detection model. Users will be able to easily select a trained algorithm and quickly predict anomaly scores for newly added videos. The application will also help to users to quickly train multiple models and see accuracy of each trained model.

2.3.2 TensorFlow API

The anomaly detection system must interface with TensorFlow, a popular machine learning library. The team behind TensorFlow has written APIs for both Python and C++, so the application must be able to utilize these libraries in order to take advantage of the tools and features available in TensorFlow.

2.3.3 File System Interface

The application must also interface with the file system of the OS, in order to access video frames in the dataset. A user should be able to easily add a new video to the file system, and then run the application to train or run the anomaly detection model.

## 2.4 Constraints and Assumptions

* *Dependency of libraries* - This application depends on the TensorFlow machine learning library, which constrains us to the TensorFlow Python API.
* *Available resources* - This application will require a significant amount of computing power. This project requires the use of high performance hardware, such as GPU clusters.
* *Time* - This application requires team members to understand many concepts in deep learning as well as how to use machine learning tools, like TensorFlow. The learning and research requirements will be a significant constraint in our implementation.

# 3 System Object Model

## 3.1 Introduction

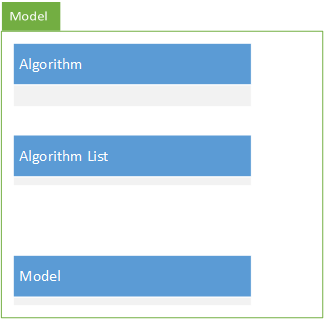
This section will briefly introduce all the subsystems that are in use to process the input data. We have divided the system in such a way that each part is independent of the other, and can be taken as the subsystem.

## 3.2 Subsystems

**Pre-processing**

****

**Training, Testing, Optimizing, and Finalizing the Model**

****

## 3.3 Subsystems Interfaces

The main class will call both subsystems and get the output. Below is the interface:

**Subsystem 1:**

1. Dataset will use video object to split up the video and stores the directory and label information inside excel file.

**Subsystem 2:**

1. The model class will get the call from the main for the algorithm. The algorithm list will call that algorithm. The main will also specify if it’s training or testing the algorithm.
2. If it’s training, then the algorithm will use the Pre-processing subsystem to process the data, and then will use the algorithm to process it, then will use the model class to save the model.
3. If it’s testing, then the algorithm class will use the testing data specified in the Pre-processing subsystem and model class to load the model.

# 4 Object Descriptions

## 4.1 Objects in Subsystem 1

**Dataset**

|  |  |
| --- | --- |
| **Class Name:** Dataset | |
| **Brief Description:** This class will pre-process the data using private functions and will return the dataset in the form algorithm needed. | |
| **Public Methods** | |
| List getTrainingData() | Returns the training data. |
| List getTestingData() | Returns the testing data. |
| List getValidationData() | Returns the validation data. |
| **Private Methods** | |
| boolean addVideo(String video\_loc, boolean isAnomaly) | Will use the video class to resize and split the video specified, and add it into csv file with the label. If isAnomaly is true, then label will be abnormal, else normal. Will return true if add is successful else false. |
| boolean removeVideo(String video\_loc) | Will delete the directory and remove the entry from data file. |
| **Attributes** | |
| String location | Location to the csv file where Directory and label will be stored for the data. |
| boolean isAnomaly | Specified by user if the video passed is anomaly or not. |

**Video**

|  |  |
| --- | --- |
| **Class Name:** Video | |
| **Brief Description:** Splits the video into frames. | |
| **Public Methods** | |
| List getFrames() | Returns the video frames. |
| **Attributes** | |
| String location | Location of the video. |

## 

## 4.2 Object in Subsystem 2

**Algorithm**

|  |  |
| --- | --- |
| **Class Name:** Algorithm | |
| **Brief Description:** This will be the design of an algorithm. | |
| **Public Methods** | |
| Model getModel() | Returns the model |
| **Private Methods** | |
| void trainData() | Trains the data for this algorithm. |
| **Attributes** | |
| hyperParameters | Specifies different hyper parameters for model training, multiple attributes may be involved. |
| List dataSet | Contains the training or testing dataset as specified by the user, get it from dataset object. |

**Algorithm List**

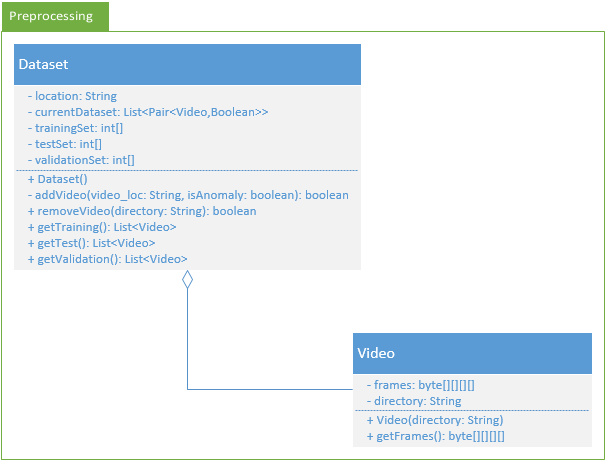
|  |  |
| --- | --- |
| **Class Name:** Algorithm List | |
| **Brief Description:** This will be the list of all the algorithms. There can be multiple algorithm class templates. | |
| **Public Methods** | |
| Model getModel() | Returns the model which is returned by the algorithm. |
| **Attributes** | |
| Integer AlgorithmId | Specifies the algorithm from the list. |

**Model**

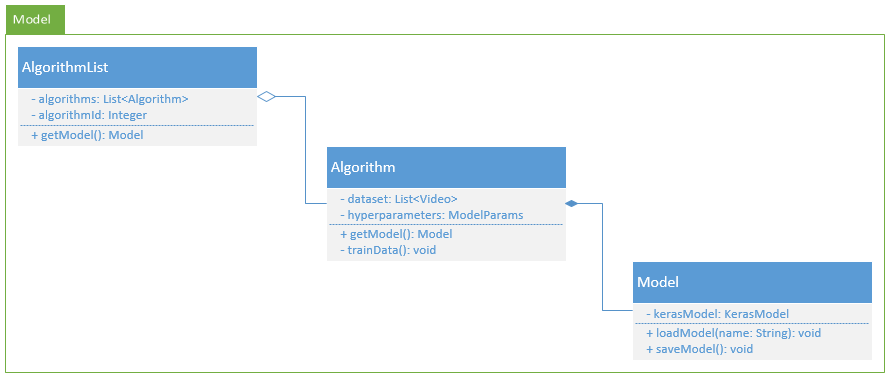
|  |  |
| --- | --- |
| **Class Name:** Model | |
| **Brief Description:** This will be able to load and save the model using Tensorflow and Keras. | |
| **Public Methods** | |
| void loadModel(String name) | Loads the specified model from the folder. |
| void saveModel() | Saves the model. Used by Algorithm class. |
| **Attributes** | |
| KerasModel kerasModel | Specifies the parameters which will be used to save the model using tensorflow and/or keras. |

# 5 Object Collaboration

## 5.1 Objects in Subsystem 1



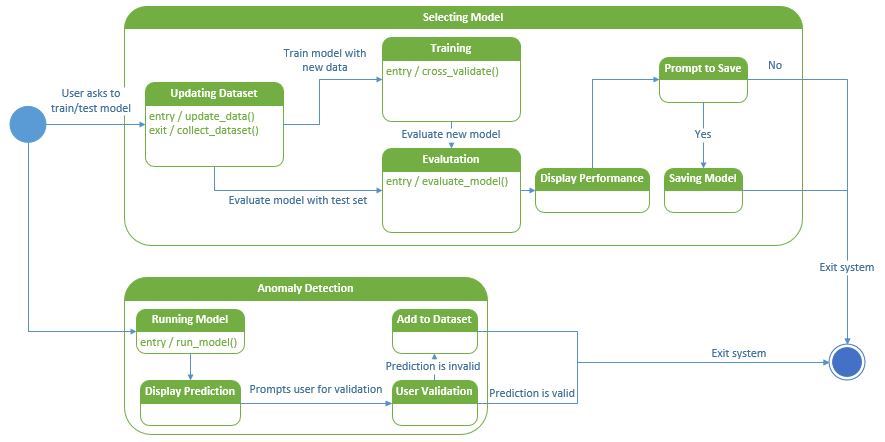
## 5.2 Objects in Subsystem 2



# 6 Dynamic Model

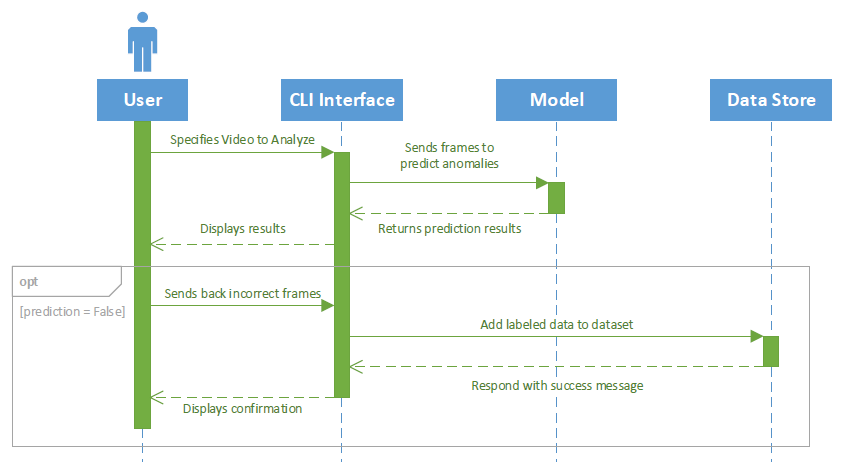
## 6.1 State Diagram

The following section includes a large UML state diagram that outlines the state changes throughout the life of the system. There are two main states of the system: Anomaly Detection (used for predicting the presence of anomalies in video) and Selecting Model (used for training/testing a model).

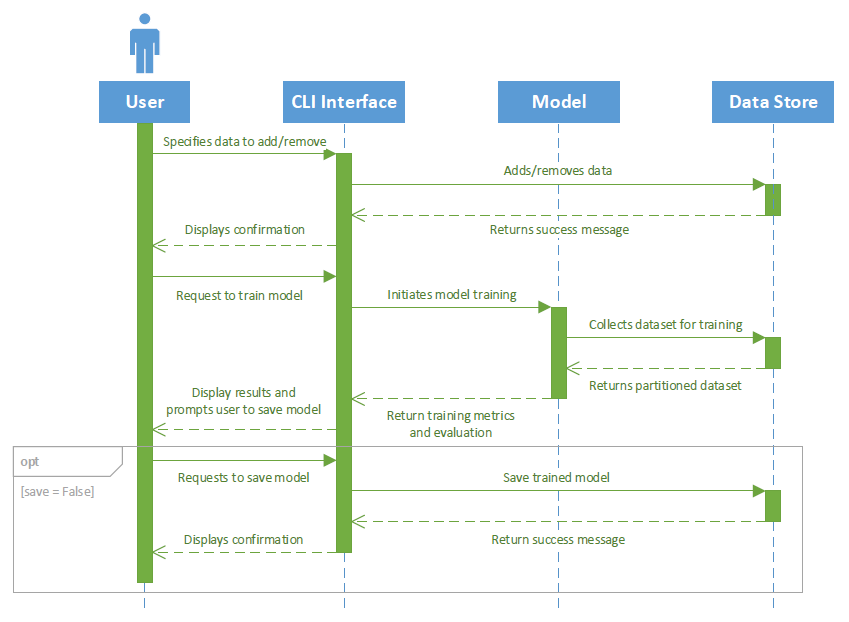
****

## 6.2 Sequence Diagrams

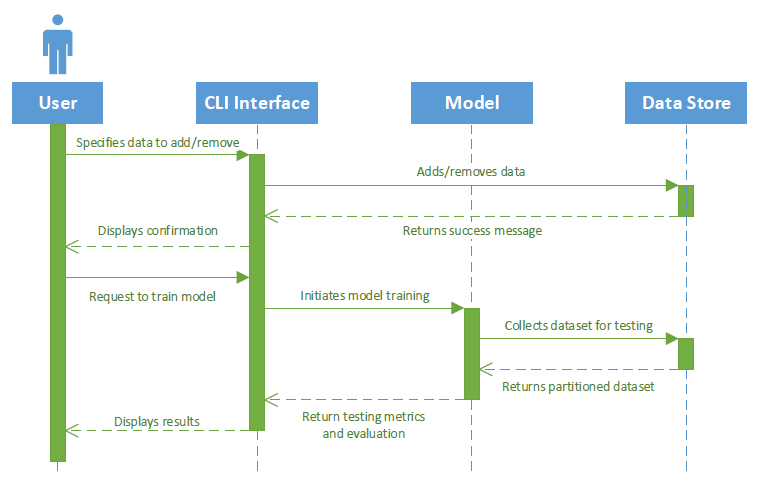
6.2.1 Sequence Diagram 1 - Predict Anomalies

****

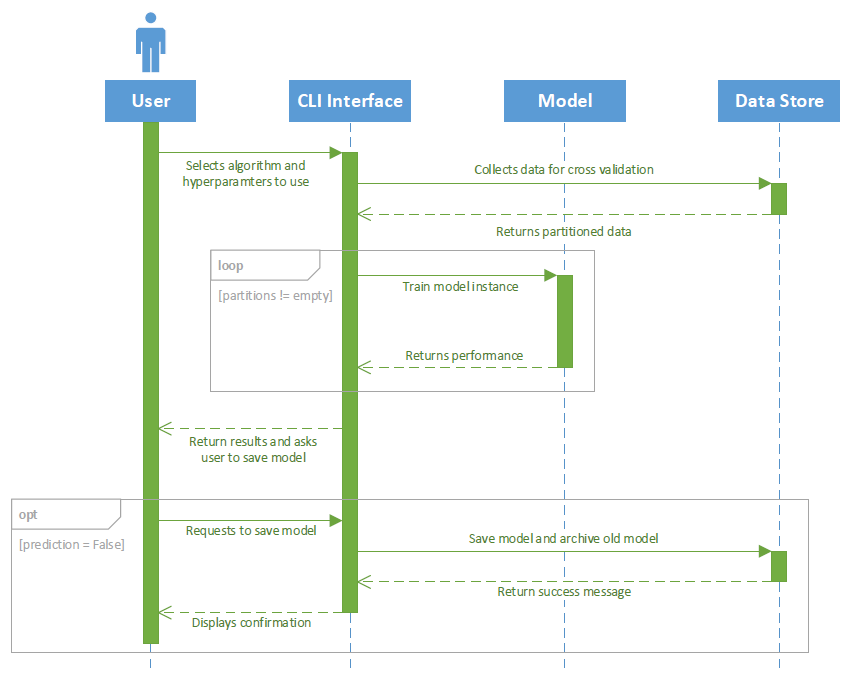
6.2.2 Sequence Diagram 2 - Model Training

****

6.2.3 Sequence Diagram 3 - Add Data to Testing Set

****

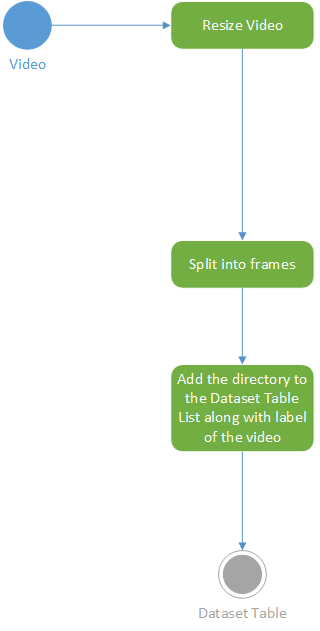
6.2.4 Sequence Diagram 4 - Optimization

****

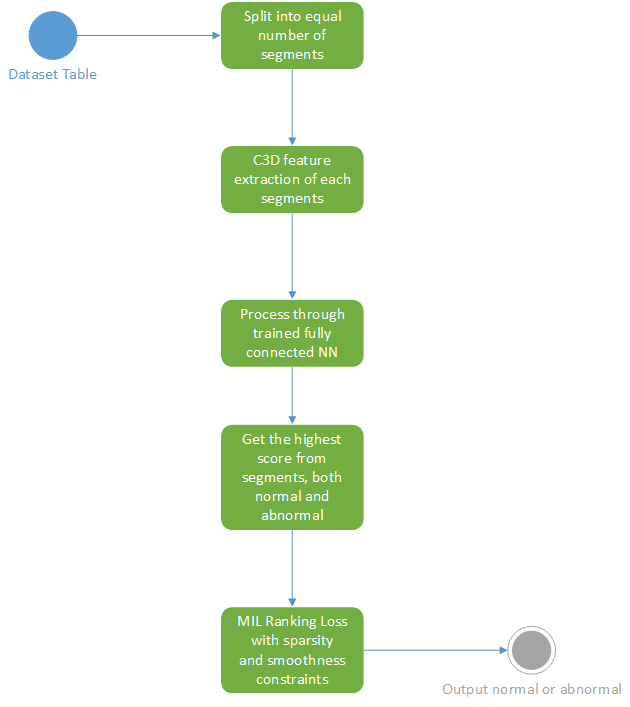
# 7 Data Design

The design structure which we will have for this project is for the dataset and for the training the model. This design is specific to the algorithm we chose to work on.

**Dataset:**



**Training**

****

# 

# 8 Non-functional Requirements

* Data should be labeled for training.
* Train model to detect specific types of anomalies.
* Have an interface where the user can upload a data entry so we can run the trained model over it.
* CPU/GPU power to train the model in a sufficient amount of time.
* Use Linux Operating System as the default platform for the system
* The system will be implemented using Python 3.6, so Python 3.6 interpreter is require.

## 

# 9 Supplementary Documentation

“Real-world Anomaly Detection in Surveillance Videos.” 12 Jan 2018.

<https://drive.google.com/open?id=169WEm5Ce2SdwnYQiqw5kvhvoXIfc5uy9>