

SUSPENSE SCENE DETECTION USING RECURRENT NEURAL NETWORK

BY

LIM SIN HUI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfilment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (Honours)

Faculty of Information and Communication Technology

(Kampar Campus)

JAN 2021

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Suspense Scene Detection Using Recurrent Neural Network

Academic Session: Jan 2021

I LIM SIN HUI
(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

35, Jalan Besar,
34100 Selama,
Perak

Aun Yichiet
Supervisor's name

Date: 16/04/2021

Date: 16/04/2021

**SUSPENSE SCENE DETECTION USING RECURRENT NEURAL
NETWORK**

By
LIM SIN HUI

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (Honours)
Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2021

DECLARATION OF ORIGINALITY

I declare that this report entitled “**SUSPENSE SCENE DETECTION USING RECURRENT NEURAL NETWORK**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : Lim Sin Hui

Date : 16/04/2021

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to my final year project supervisor, Dr. Aun Yichiet who offered me an opportunity to work on this project. In addition, Dr. Aun has been helpful and supportive along the project by providing useful guidance and accessing to a GPU machine to reduce the training time required.

Secondly, I would like to thank to my family and friends who always being mentally supportive when I was facing difficulties in the project. Without their love and tolerance, I definitely could not have completed my project.

ABSTRACT

Detecting the onset of suspenseful scenes is helpful for optimal ad placement. Some previous works that use movie scripts to imply suspense are somewhat deprived of contextual cues found in audio and video data. Meanwhile, the lack of a public video dataset for suspense scenes adds to the challenges to train ML-based suspense scene detection (SSD). In this project, an expert-annotated suspense scenes dataset containing videos from 3 classes (football, cooking, and room escape) is collected from YouTube. The dataset collection method follows the framework outlined by VSD2014 for dataset integrity. An SSD model is trained using custom RNN-LSTM using features extracted on ResNet50 for suspense scene detection in selected short YouTube videos. First, the minority classes are 'oversampled using a custom data balancing method to preserve these extrapolated frames' temporal sequence. Then, the AX library is used to brute-force the most optimal neural network configurations and hyperparameter tuning. The experimental results showed that the SSD model is highly accurate in detecting suspense scenes on unseen videos and generalized well, scoring a 0.7642 testing accuracy.

TABLE OF CONTENTS

TITLE PAGE	i
DECLARATION OF ORIGINALITY	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF SYMBOLS	x
LIST OF ABBREVIATIONS	xi
Chapter 1 Introduction	1
1-1 Problem Statement and Motivation	1
1-2 Project Scope	1
1-3 Project Objectives	1
1-4 Impact, Significance and Contribution	2
1-5 Background Information	2
1-5-1 Shot and Scene Detection	2
1-5-2 Historical Development of Scene Detection	3
1-5-3 Classification.....	4
1-5-4 Feature Extraction	4
1-5-5 Types of Prediction on Sequence Data	4
1-5-6 Artificial Neural Network	5
1-5-7 Recurrent Neural Network	6
1-5-8 Long Short-Term Memory	7
1-5-9 Hyperparameter Tuning.....	8
1-6 Proposed Approach	8
1-7 Report Organization.....	9
Chapter 2 Literature Review	10
2-1 Literature Review.....	10
2-2 Scene Segmentation Methods	10
2-2-1 Two Pass Algorithm	10
2-2-2 Two Dimension Entropy Model	11
2-3 Scene Classification Methods	13
2-3-1 Finite State Machine	13
2-3-2 Support Vector Machine and Hidden Markov Model	14

2-4 A Summary Survey on Genres of Video Segmentation Approaches and Relative Datasets	15
2-4-1 Summary on Video Segmentation Genres	15
2-4-2 Relative Dataset Analysis	16
2-5 Suspense Detection using Text	17
2-5-1 Dramatis Framework	17
2-5-2 Seven Components-based Modelling	17
2-5-3 Hierarchical Language Model.....	17
2-6 Data Collection and Annotation Methods.....	19
2-7 Class Imbalanced and Technique to Overcome.....	21
2-8 Remarks	22
Chapter 3 System Design	23
3-1 Methodology	23
3-2 Tool and Technologies used.....	24
3-3 Standard Evaluation Metrics for Model Performance.....	25
3-4 Overall Architecture Flow	25
3-5 Dataset.....	27
3-5-1 Dataset Searching.....	27
3-5-2 Data Collection and Annotation.....	27
3-6 Data Pre-processing	30
3-6-1 Frame Extraction.....	30
3-6-2 Feature Extraction and Time Series Data Transformation	31
3-7 Model Optimization and Training	33
3-8 Network Architecture.....	36
3-9 Expected Output.....	38
3-10 Implementation Issues and Challenges	38
3-11 Timeline	40
Chapter 4 Model Implementation	42
4-1 Project Setup	42
4-2 Build the Project Steps by Steps	43
Chapter 5 Experiments and Results	45
5-1 Hyperparameters Optimization.....	45
5-2 Performance Analysis	46
5-3 Test Data and Its Variants.....	49
5-4 Further Remarks.....	51

Chapter 6 Conclusion	52
6-1 Project Review	52
6-2 Future Work	52
Bibliography	53
Appendix A: Final Year Project Source Code	A-1
Appendix B: Poster	B-1
Appendix C: Final Year Project Biweekly Report	C-1
Appendix D: Plagiarism Check Result	D-1
Appendix E: Form iad-FM-IAD-005	E-1
Appendix F: FYP 2 checklist	F-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 1-1	Neural Network Illustration	5
Figure 1-2	RNN Backpropagation	6
Figure 1-3	LSTM Cell	7
Figure 2-1	Pixel Dissimilarity Graph	12
Figure 2-2	General FSM Model	14
Figure 2-3	Audio Classification Structure	15
Figure 2-4	Dataset Analysis	16
Figure 2-5	Hierarchical Model	19
Figure 2-6	Hierarchical Bottom-up Approach	21
Figure 3-1	General Methodology	23
Figure 3-2	Proposed Generic Architecture Flow	25
Figure 3-3	Frame Extraction Flow	30
Figure 3-4	Feature Extraction and RNN Shape Transformation Flow	31
Figure 3-5	Model Optimizing and Training Flow	33
Figure 3-6	Stratified K-fold Cross Validation Illustration	34
Figure 3-7	Network Design Architecture	36
Figure 3-8	Model Summary	37
Figure 3-9	Timeline	40
Figure 5-1	Training and validation loss for initial model	45
Figure 5-2	Loss and accuracy for training and validation set for model_0743	47
Figure 5-3	Loss and accuracy for training and validation set for model_1130	48
Figure 5-4	Loss and accuracy for training and validation set for model_1312	48
Figure 5-5	Loss and accuracy for training and validation set for model_1607	48

LIST OF TABLES

Table Number	Title	Page
Table 2-1	Scenes Characteristics	13
Table 3-1	Online Dataset Sources	27
Table 3-2	Data Collection Guides	28
Table 3-3	Dataset Statistic	28-29
Table 4-1	Experiment Settings	43
Table 5-1	Confusion matrix for initial model	45
Table 5-2	Difference in settings for different model	46
Table 5-3	Result on training set based on different model	47
Table 5-4	Test data specifications	49
Table 5-5	Result on testing set ID 1 based on different model	49
Table 5-6	Result on testing set ID 2 based on different model	49
Table 5-7	Result on testing set ID 3 based on different model	50
Table 5-8	Result on testing set ID 4 based on different model	50

LIST OF SYMBOLS

E	Two dimensional entropy
N	Normalization of the two dimensional histogram
H	Two dimensional histogram
F	Frame
w	Word embedding
γ	Sentence embedding
e	Story embedding
$\emptyset(x) / \sigma$	Sigmoid function

LIST OF ABBREVIATIONS

AMT	Amazon Mechanical Turk
ANN	Artificial Neural Network
AUC	Area Under the Curve
BSC	Backward Shot Coherence
CNBC	Consumers News and Business Channel
EI	Expected Improvement
FSM	Finite State Machines
GPT-3	Generative Pre-trained Transformer
GRU	Gated Recurrent Units
HMM	Hidden Markov Models
MLED	Multimodal Emotion Recognition Dataset
MUStARD	Multimodal Sarcasm Dataset
LSTM	Long Short-Term Memory
OHIT	Oversampling method to combat the High-dimensional Imbalanced Time-series classification
PD	Pixel Dissimilarity
PSB	Potential Scene Boundaries
RAM	Random Access Memory
ROC	Receiver Operating Characteristic
RNN	Recurrent Neural Network
SD	Shot Dynamics
SMOTE	Synthetic Minority Oversampling Methods
SSD	Suspense Scene Detection
SVM	Support Vector Machine
URL	Uniform Resource Locator
VSD	Violent Scene Detection

Chapter 1 Introduction

1-1 Problem Statement and Motivation

Traffic from digital marketers has become social media centric. An article in Consumers News and Business Channel (CNBC) reported that YouTube generated \$15.15 billion revenue in year 2019 from the YouTube advertisements (Consumer News and Business Channel CNBC 2020). The statistics indicates that the digital marketers invested huge amount of money in advertising at social media's platform in order to drive the traffic. Therefore, this project claims that it is important to maintain the viewers' attention for the purpose of avoiding them from exiting the video.

From aspect of narrative, suspense is a key element to retain target's attention (Khrypko and Andreae, cited in Delatorre, León, Salguero, Palomo-Duarte and Gervás 2018). Hence, this project proposes a novel idea, which is suspense classification using video as input. Due to the lack of appropriate dataset, an annotated dataset based on suspense is introduced. This project hypothesized that the better ads placement strategy was to start playing an ads in a suspenseful scene of a video. The intuition of the work was to draw the user attention and interest as the factors to continue watching the videos rather than emphasize on minimizing the intrusiveness of the ads.

1-2 Project Scope

This project aims to introduce a suspense annotated dataset which is intended to act as a benchmark for suspense detection in YouTube videos. Using the dataset prepared, the project intends to develop a suspense binary classification system to perform binary classification on video scenes. Also, an oversampling technique, Noise Oversampling Technique for Time-series data (NOTT) is introduced to solve the data imbalance issue for recurrent neural network.

1-3 Project Objectives

The objectives of this project are:

- i. To investigate state-of-the-art suspense detection techniques and their corresponding performance.
- ii. To introduce a public video dataset for suspense classification that will set a benchmark for suspenseful scene detection.

- iii. To build a scene classification system that able to distinguish suspense and non-suspense scenes.
- iv. To introduce a novel oversampling technique used in time-series data to overcome class imbalanced issue

1-4 Impact, Significance and Contribution

There are three outcomes from this project: (a) an annotated dataset, (b) a suspense scene classification system and (c) an oversampling technique for time-series data. The annotated dataset based on suspense element will be made available to the public. Considering that dataset labelled with suspense had not been done by the previous reviewed literature, the introduced dataset will play an important role in the research of suspense scene detection that is for benchmarking suspense detection in the future research work.

On the other hand, the suspense scene classification system will be able to classify scenes into suspense and non-suspense scene. The concept and result obtained are significant as in the reviewed literature, none of the researchers solely worked on suspense classification using videos as input.

Last but not least, this project will introduce a unique oversampling method called Noise Oversampling Technique for Time-series data (NOTT) in order to handle class imbalance issue occurred in time-series data binary classification.

1-5 Background Information

1-5-1 Shot and Scene Detection

In order to process an arbitrary long video in machine learning, the prior steps required were shot and scene detection. Shot detection had divided the video into small unit of shots. A shot could be defined as a sequence of images that had the same setting within a short time interval. Nevertheless, each single shot could not give meaningful insight to the story inside the video. Thus, scene detection could be utilized to provide a better insight by grouping the similar shots together as a scene. A machine could perform scene detection through identifying shots that have similar features.

However, different types of videos had different genres of filming methods. Some videos were filmed with multiple stationary cameras while some were filmed

using moving camera, along with different scenes switching. The latter method was often seen on movies or dramas especially when: (a) the actor wanted to recall the past memories or (b) the director wanted to simultaneously show the story development of two or more sides. Hence, this led to the problem where the computer machine faced difficulties in scene segmentation as the machine could not perfectly segment and group the shots to their corresponding scenes when a scene was interrupted by another scene in the middle of the content.

1-5-2 Historical Development of Scene Detection

The previous researchers had performed scene recognition for the videos in which the setting was fixed such as news videos, talk shows and sitcoms. The study showed significance performance in scene recognition for the videos that were filmed in constrained environment. One of the foremost works in year 1998 using time-constrained clustering and scene transition graph had successfully segmented the videos into meaningful story unit (Yeung, Yeo, and Liu 1998). The videos used in the testing included sitcoms, news, documentaries, talk shows and segments of different movies where the background settings were mostly fixed. The system was able to better identify the scene boundaries in those videos due to the repetitive structure displayed within a scene.

However, as compared to fixed setting videos, the graphical method could not work well for feature videos where its content would continuously change. Both problems: (a) under segmentation, a division into too few segments and (b) over segmentation where the videos had been segmented into too many isolated scenes, could be encountered when the inappropriate approach was applied to segment the feature videos (Rasheed and Shah 2003). Two different scenes might be wrongly merged into one scene if both were having high colour similarity. Meanwhile, a scene could be wrongly fragmented into two or more scenes, action scene in particular due to the rapid change of content in such scene.

On top of that, this project focuses on detecting suspenseful scene, the term “suspense” is hard to be defined for a machine. In an article analysing surprise, suspense and curiosity, suspense is defined as a strategy in narrative to raise viewers’ expectation meanwhile ignore their fulfilment (Yuan 2018). There are also papers arguing that suspense is defined as uncertainty towards an outcome which contrast with other papers

which claim suspense can exist without uncertainty but depends on the plot arrangement (Hale; Ely, Frankel, and Kamenica, cited in Wilmot and Keller 2020). From the definitions above, there is no clear measure on suspense. Suspense could be understandable by human but hard to be recognised using machine learning based on features extracted. Therefore, this project uses manual annotation to label the suspense scene in few video categories by indicating its corresponding start frame and end frame.

1-5-3 Classification

Classification is a type of supervised learning that learns based on labelled training data. In general, classification can be categorised into binary and categorical classification. Nevertheless, this project places suspense detection problem into a binary classification where the output is either 1 or 0 to respectively represent suspense and non-suspense. Multi-categorical classification is neglected in this project because the suspense intensity level could be loosely defined when there is no clear defined standard on it.

1-5-4 Feature Extraction

Feature extraction is a process to reduce the dimensionality of the raw data so that it is more manageable in later process. Directly extracting the features from frame and input to the model to learn the features pattern may not be helpful since this project is not doing transfer learning, instead this project is building the model from scratch. Hence, in order to let the model to learn some useful features, that is high level features, this project plans to utilize pre-trained model to act as feature extractor and output high level features before inputting data to the target neural network. The pre-trained model selected here is ResNet50 and further discussion will be done in the following chapters.

1-5-5 Types of Prediction on Sequence Data

This project is trying to work on scene classification on video data, it can also be interpreted as prediction on sequence data since video data is sequence data. In an article written by Usman Malik, there are four types of prediction on sequence data are introduced (Malik, n.d.). They are One-to-One, Many-to-One, One-to-Many and Many-to-Many. In the first case scenario, there will be exactly one input and one output. For example, the input is a single image and the model is trying to predict the image class based on the input.

In the second scenario, Many-to-One type is the category this project trying to work on. This can be further elaborated as a sequence of data is feed into the neural network to obtain a single output. In this project, each sample data consists of information of k past frames and the model is trying to predict the class from frame $k+1$ which is matching the Many-to-One type.

Similarly, in the third case, there will be only one input and the target model is trying to produce a sequence of output based on that single input. While the last case, Many-to-Many happens when the model is trying to deal with a sequence of inputs and the output is also a sequence of data.

1-5-6 Artificial Neural Network

Artificial neural network (ANN) is also seen as neural network is a type of network derived in deep learning. The initial idea of neural network is to stimulate the actual human neurons which transfer signal through synapse so that the machine can learn and think in a humanlike way. Figure 1-1 below shows an example illustration of a neural network.

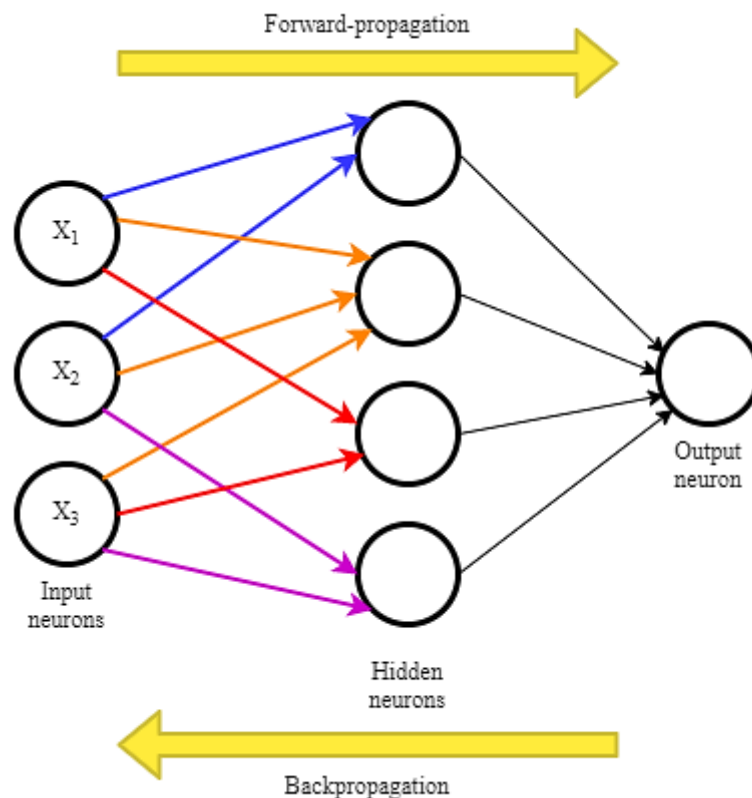


Figure 1-1: Neural Network Illustration

An artificial neural network is usually made up of three main layers, which are input, hidden and output layers. For the input layer, it represents the input variables to be trained whereas the hidden layers can involve more than one layer depending on the problem complexity and the output layer is where the neural network presents the final outcome. The whole network starts with forward-propagation that is input neurons with different weights are fed into the hidden layer and forward propagated to obtain the predicted outcome using an activation function. The neural network would then perform backpropagation by computing the difference between the predicted value and the actual value and then feeding back to the network to update the weight of input neuron. The forward-propagation and backpropagation process will repeat until reaching the minimum difference between the predicted value and actual value, which is the optimum model to fit the input data.

1-5-7 Recurrent Neural Network

Recurrent neural network (RNN) is a kind of neural network branched from ANN to solve problems that could not be handled by typical type of neural network such as handling sequential data and memorizing previous input. RNN architecture uses the rule of feeding back the input at time $t-1$ to the current input at time t to improve the output prediction.

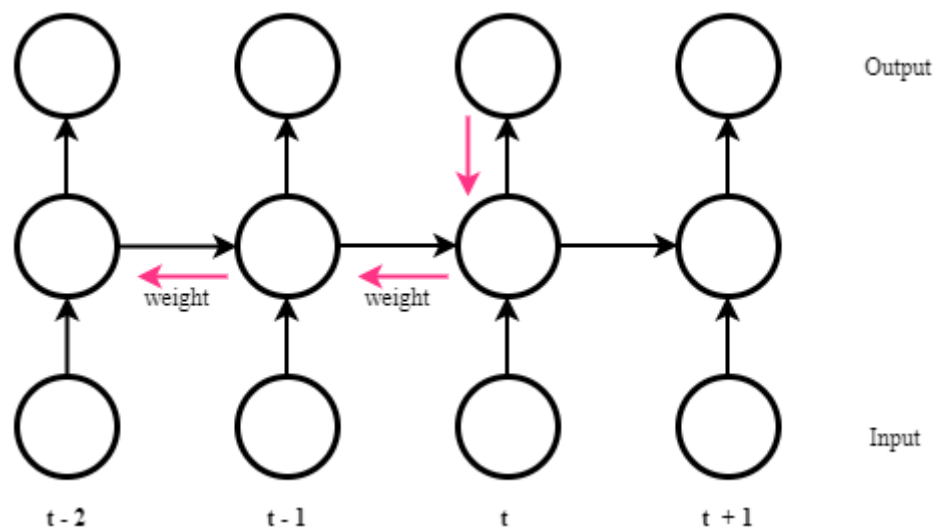


Figure 1-2: RNN Backpropagation

However, this architecture lies an issue in backpropagation where the weight updated becomes smaller as going backward if the weight is initialized close to zero, causing

vanishing gradient problem. After backpropagation, the network will start over again the forward-propagation with the new updated weight at the left most side which only has small adjustment, and this causes the training of the neural network becomes slower as it has to go through more training iteration to update the weight in order to get the optimal result. On the other hand, exploding gradient problem could occur if the weight initialization is too large. Also, RNN fails to process the data which long sequence length, in layman's terms, RNN cannot memorize the data if it carries too much of the past information. Subsequently, Long Short-Term Memory (LSTM) comes into place to solve issue occurred in typical RNN.

1-5-8 Long Short-Term Memory

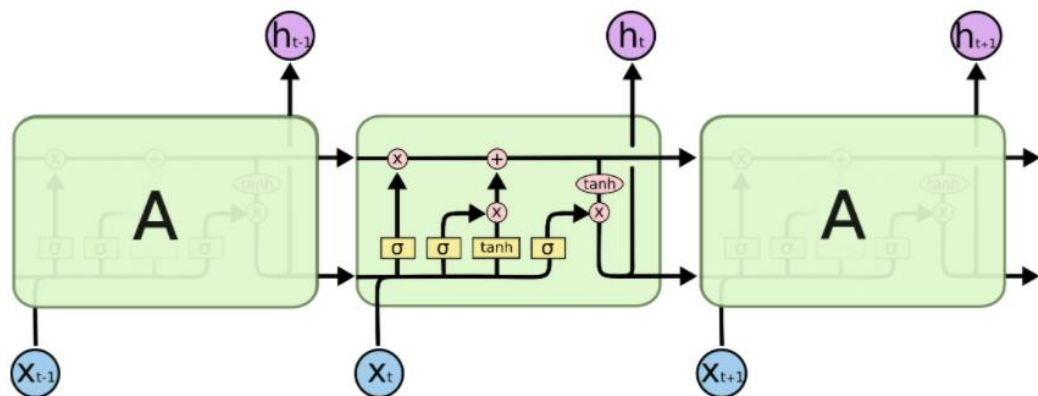


Figure 1-3: LSTM Cell

LSTM is modified from RNN with the goal that making it easier to memorize past information. Also, vanishing gradient effect previously caused by RNN is resolved in LSTM. In LSTM cell design, there are total three gates: (a) input gate, (b) forget gate and (c) output gate.

Forget gate decides which information to be thrown away or kept, The input from the previous hidden state and the current input will go through the first sigmoid function, σ at the most left hand side (Phi 2018). Sigmoid will then calculate the probability and if it is above 0.5, output is 1 and otherwise 0. The output from the sigmoid function decides the information is to be thrown away or kept, 0 indicates to forget whereas 1 indicates to keep.

For input gate, the information from the previous hidden state and current input will go through both sigmoid and tanh function located in the middle of the cell. In sigmoid function, the similar computation is done and the output is either 1 or 0 where

0 means not important and 1 means important. This will decide which information to keep from the tanh output. While tanh function will output a value between 1 to -1 and both sigmoid output and tanh output will be multiplied. Then, the previous cell state comes from the pipe at top and go through pointwise multiplication with the output from forget gate. Then, the output will further go through a pointwise addition with the output from input gate to update the cell state.

Finally, output gate decides the next hidden state by passing previous hidden state and current input to the sigmoid function at the right hand side. At the same time, the new cell state will go through the tanh function at the most right hand side, both the output from sigmoid and tanh will then go through a pointwise multiplication to generate the next hidden state and the output is then carry forward to the next cell.

1-5-9 Hyperparameter Tuning

Hyperparameter tuning is a compulsory phase to be went through in machine learning where the goal of this process is to obtain the optimal hyperparameters for that particular model. Typically, the developers do not know about the optimal model architecture that can present the optimal result and this is where hyperparameter tuning comes in place to ask the machine to explore the optimal settings that give greater model performance. There are some well-known hyperparameter tuning methods such as grid search and random search. Nevertheless, this project decides to use Ax tool from Facebook which utilizes Gaussian Processes and Bayesian Optimization to optimize the model settings.

1-6 Proposed Approach

By summarizing the discussion above, this project decides to use LSTM layer to construct the model as the model is dealing with sequence data and the ability to memorize past data in memory is compulsory in order to achieve the project goal. Also, to avoid overfitting issue, dropout layer is added during training so that a certain ratio of neurons are ignored while inputting to the next layer. Finally, the model ends with a dense layer or fully connected layer as the output layer in the model. The dense layer will compute the probability of the particular sample to be suspense. If the probability exceeds 0.5, then it will be classified as suspense and otherwise.

1-7 Report Organization

The report organization is as follows: literature review in chapter 2; system design in chapter 3; chapter 4 elaborating in details on the model implementation; evaluation on experimental results in chapter 5 and finally a conclusion to wrap up the report in chapter 6.

Chapter 2 Literature Review

2-1 Literature Review

The literature review involves few genres of academic papers, ranged from scene segmentation techniques to scene classification approaches. A survey paper analysing the segmentation approaches and their related dataset has also been included to indicate the corresponding dataset issue. On top of that, this section also includes related literatures doing suspense detection using text feature and lastly some reviews on the recent data collection methods.

2-2 Scene Segmentation Methods

2-2-1 Two Pass Algorithm

In the literature review, one of the papers focused on scene detection and representation of feature videos. A two-pass algorithm had been proposed to detect the optimal videos scene boundary (Rasheed and Shah 2003). In pass one, Backward Shot Coherence (BSC) was used to detect the shot that matched with the previous shot based on colour similarity. Shots that detected to have high similarity in colour and within the time interval set were generally having similar BSC value and they were categorized into one scene. When a new scene started, the BSC of the shot would experience a lower value and this formed valleys in the BSC graph. The valleys found could be utilized to detect the Potential Scene Boundaries (PSB).

The paper stated that action scenes might be over-segmented in this stage due to the continuous changing of content in the shots. Hence, in pass two, Shot Dynamics (SD), a function of computing the shot motion content over shot length, was used to identify the action scene among the PSB. If SD of two consecutive scenes exceeded the threshold, both were potentially to be action scenes. The weak PSB among the action scenes were removed by merging the weak PSB into one scene boundary. The correctness of scene detection was identified by comparing the result with the scene boundaries selected by a human. According to the result obtained using few Hollywood movies and a sitcom, precision score obtained ranged from 62.9% to 81.6%.

2-2-2 Two Dimension Entropy Model

The paper introduced a two dimension entropy model to partition the videos through detecting the transition boundary (Zhu and Liu 2009). The first step is to find the break points within the video. This is completed by computing the pixel dissimilarity (PD) between the predecessor and successor frames which denoted as s and t respectively. Suppose that i and j are the gray value of a pixel and mean gray value from its neighbourhood's pixel whereas E_{ij}^t and E_{ij}^s represent the two dimensional entropy model for t^{th} and s^{th} frame. L is set as 256, and $L-1$ will obtain a maximum of 255, that is the pixel value for the grayscale image which ranged from 0 to 255. In general, the pixel dissimilarity function is the summation of the minimum entropy value between the target and its successor frame from pixel 0 to 255 in two dimensional (i, j) vector.

$$PD(t) = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \min(E_{ij}^t, E_{ij}^s)$$

The two dimensional entropy model, E is defined as follow, where N represents the normalization of the two dimensional histogram.

$$E = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} E_{ij} \text{ where } E_{ij} = -N_{ij} \times \ln N_{ij}$$

The normalization of two dimensional histogram, N_{ij} is calculated by taking fraction of H_{ij} over the sum of H_{ij} which is denoted as H in the equation below. The normalization action is to ensure all the values of the two dimensional histogram are in the same range.

$$N_{ij} = H_{ij} / H \text{ where } H = \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} H_{ij}$$

The values obtained from pixel dissimilarity function across the frame number can form a graph which can be used to roughly obtain the shot changes points by looking at the valleys formed.

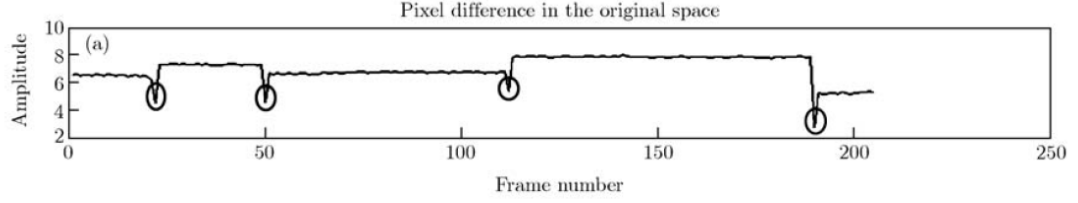


Figure 2-1: Pixel Dissimilarity Graph

The next step is to obtain the precise transition point from the rough shot changes by finding the minimum pixel dissimilarity within the T_{length} distance from the rough point, where T_{length} is set as five.

To overcome the over-detection on transition caused by the necessity in the frequency change of visual content, two types of transitions are defined: (a) gradual transition and (b) abrupt transition. Gradual transition is detected when the average of two dimensional entropy from the left frame and right frame, AE_{LF} and AE_{RF} both values are lower than the threshold mean and variance defined for a frame. Then, suppose t is the precise position. If the difference of the two dimensional entropy between the $t - 1$ and t frames over $E(t - 1)$ satisfies the following equation, that is less than the threshold value defined for gradual transition boundary T_{GB} , frame t is the starting frame of that gradual transition. Inversely, the frame t that satisfies the equation for right hand side will be the ending frame of that gradual transition.

$$\text{Left side: } \{E(t - 1) - E(t)/E(t - 1)\} < T_{GB}$$

$$\text{Right side: } \{E(t + 1) - E(t)/E(t)\} < T_{GB}$$

For abrupt transition, it is categorized into two different types which are real abrupt shot and shot with flashlight. The difference between these two shots is the latter genre usually has high similarity before and after the abrupt frame whereas for the real abrupt shot, the visual content similarity before and after the transition point is distinct. Hence, the function to compute the difference of the entropy value from the left side and the right side, $DVNR^j$ is established as follow:

$$DVNR^j = \left| \frac{VNL^i - VNR^i}{VNL^1 - VNR^1} \right|$$

VNL^i and VNR^i represent the average of the two dimensional entropy in i number of frames at the left and right side of the transition point while F is the frame that identified

to be the transition point. $E(F - f)$ will sequentially get the two dimensional entropy value of each frame starting from the nearest left hand side of transition frame to its consecutive frames when f increases from 1 to i and same applies for $E(F + f)$ which obtains the entropy values at the right side. Then, the average at each side can be obtained by dividing the summation of entropy with the total frames i . The formulas are shown below:

$$VNL^i = \frac{\sum_{f=1}^i E(F - f)}{i}, i = 10$$

$$VNR^i = \frac{\sum_{f=1}^i E(F + f)}{i}, i = 10$$

Due to the abrupt transition occurs between two frames, there is no need to detect the transition boundary for abrupt shots.

2-3 Scene Classification Methods

2-3-1 Finite State Machine

In a work published by Zhai, Rasheed and Shah (2005), the researchers proposed a scene classification technique using finite state machines (FSM), an abstract machine in which its execution flow was controlled by certain conditions given to classify the three types of movie scenes, which are conversational, non-conversational and suspense scene. The two pass algorithm in section 2-2-1 was used to perform scene detection prior to scene classification. Then, the three scene categories were defined with some features such as audio energy, activity intensity and speakers. Table 2-1 showed the characteristics correspond to their respective scene category.

Characteristics Scenes	Activity Intensity	Audio Energy	Speakers
Conversational	Low	Medium	Multiple
Suspense	Low followed with sudden change	Low followed with sudden change	
Action	Intensive	Intensive	

Table 2-1: Scenes Characteristics

For different categories of scenes, FSM sets different condition in each state to examine the scene type. Only scenes that matched their corresponding criteria would reach the final state of FSM and be classified into either one of the three types of scenes.

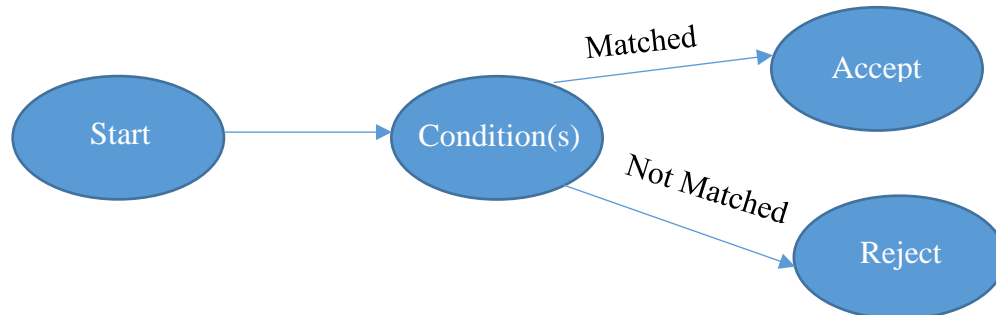


Figure 2-2: General FSM Model

2-3-2 Support Vector Machine and Hidden Markov Model

Another literature in 2009 introduced a scene categorization method using low and mid-level features with the integration of Support Vector Machine (SVM) and Hidden Markov Model (HMM) (Zhu, Yan and Liu 2009). The work used the two-dimensional entropy model reviewed in section 2-2-2 to perform the shot and scene detection. Prior to classification, audio features which are spectrum flux, zero-crossing rate, energy envelope, mel-frequency cepstral coefficients, band periodicity, spectral power and linear prediction based cepstral coefficients have been extracted from the clips. Audio features extracted are then used to first remove the silent scene. With the audio features as input to the SVM model, the non-silence scenes would be further categorized into two types, which are speech and non-speech based on the feature similarity. Scenes with speech would be split into speech with common and emotional tone whereas non-speech scenes would be identified as scene with pure music or other sound. Eventually, the remaining scenes with unidentified sound would be classified into six different classes using HMM as shown in Figure 2-3.

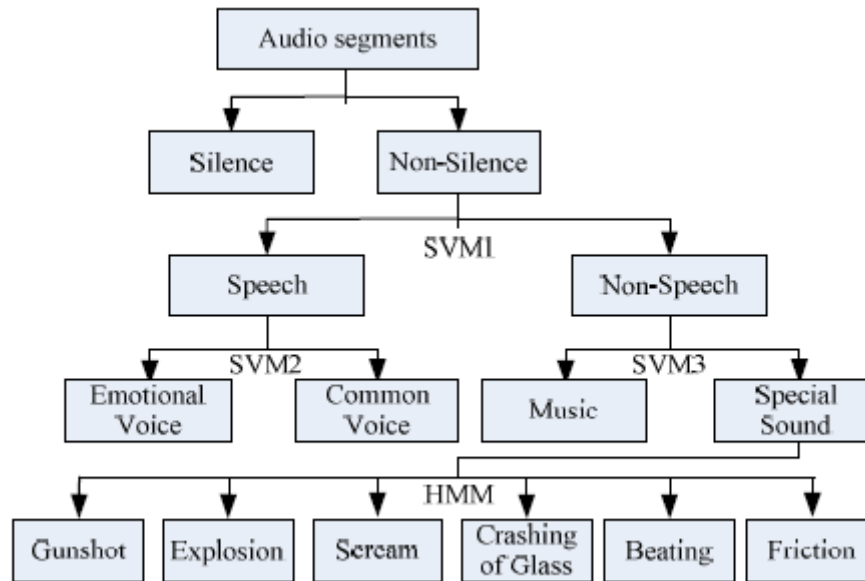


Figure 2-3: Audio Classification Structure

Other than that, visual features such as face information, illumination intensity, activity intensity and average duration are also extracted. The classification of dialogue, active and suspense scene are then performed using some pre-rules defined matching with the audio-visual features extracted to distinguish different scene types. The categorization result is then evaluated and compared with the highest voted category for each scene by the 20 volunteers, the overall performance had achieved average precision of 0.924 and recall of 0.920.

2-4 A Summary Survey on Genres of Video Segmentation Approaches and Relative Datasets

2-4-1 Summary on Video Segmentation Genres

A paper written by Del Fabro et al. (2013) has summarised the published video segmentation approaches from year 1998 to 2012 and grouped them into seven different classes based on the feature genre. The seven segmentation types respectively are (a) visual-based, (b) audio-based, (c) text-based, (d) audio-visual based, (e) visual-textual based, (f) audio-textual based and (g) hybrid methods (Del Fabro and Böszörményi 2013). Generally, each video segmentation method could be categorized into two types: (a) full segmentation and (b) partial segmentation. Full segmentation referred to the video partition where each unit belonged to the video would be kept and the original video was able to be reproduced by merging all the units together. On the other hand,

partial segmentation referred to the situation where only targeted part was extracted from the video and the rest was disregarded.

In general, segmentation approaches based on different feature types have their pros and cons. Although pure visual-based segmentation can lead to high accuracy, its computational cost is generally higher than audio and text features. On the other hand, pure auditory segmentation method has lower computational cost yet worse performance whereas video segmentation using purely text can result in high accuracy. Nevertheless, text-based segmentation is generally harder to be achieved as not all the videos are ready with fully complete metadata. On top of that, integrated feature segmentation methods are introduced with the purpose of enhancing the segmentation performance.

2-4-2 Relative Dataset Analysis

The authors have also investigated the datasets used in the papers reviewed as shown in Figure 2-4. The work stated that most of the datasets used, that is 81% of them are personal datasets which are inaccessible to the public (Del Fabro and Böszörményi 2013). This clearly indicates the difficulty for other researchers to compare their results with the former works using the same datasets. Hence, in line with the problem statement of this project, this motivates this project to introduce a public available dataset for benchmarking suspense detection.

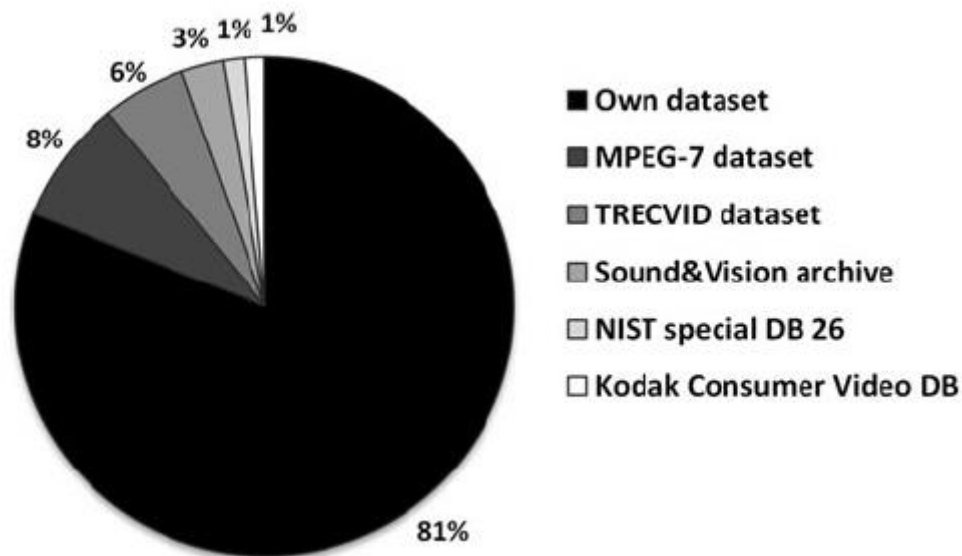


Figure 2-4: Dataset Analysis

2-5 Suspense Detection using Text

2-5-1 Dramatis Framework

The authors have developed a framework to detect the suspense through the computation of the likelihood for the protagonist to escape from the unfavourable outcome (O'Neill and Riedl 2011). The dramatis framework started by inputting the story script that contained detailed description of the characters, settings, plot and scene. Then, the system would search for the potential failure that may affect the desired state of the protagonist through the metadata in the script. The system represented a network that mapped the input script into a few possible events, then the most undesirable failure would be selected to compute the likelihood to escape from that failure. The higher the likelihood for the protagonist to avoid the undesirable outcome, the lower the suspense intensity.

2-5-2 Seven Components-based Modelling

In another literature working on suspense detection, an architecture was developed to produce more engaging stories that is narrative with higher suspense intensity. The proposed model contained seven components that started with input scripts which were already split into a set of fragments (Delatorre, Arfe, Gervás and Palomo Duarte 2016). Then, the fragmented scripts would go into the second component to compute its suspense intensity. If the result obtain was less than the pre-required intensity, these fragments would go to the third component to extract the description from the fragment. The forth component, transformer would try to increase the suspense intensity by modifying the script element followed by corpus which served to compute the new suspense intensity. Some descriptive elements were added to raise the intense of suspense. Lastly, the system would reassemble all the new and original fragments to become the final output.

2-5-3 Hierarchical Language Model

A recent literature published in 2020 by Wilmot and Keller has introduced a hierarchical language model using Recurrent Neural Network (RNN) to predict the suspense values and make comparison between the model prediction and suspense judgement based on human annotators which acted as ground truth of the model (Wilmot and Keller 2020). The paper used *WritingPrompts* dataset which consists of

300K short stories in the form of sentences and annotation process was done by the workers employed at Amazon Mechanical Turk (AMT), which it is a marketplace that allows individuals to request some workforce to perform tasks. The annotation was done sentence by sentence and the workers were instructed to annotate each sentence from 100 stories with one of the five suspense patterns which are Big Decrease, Decrease, Same, Increase and Big Increase.

On the other hand, the hierarchical language model is illustrated in Figure 2-5. The model starts with Generative Pre-trained Transformer (GPT) word encoder which transforms each word into a word embedding w_i . The embedding word would then input to RNN sentence encoder to change it to embedding sentence which represented by γ_i . Then, the last word of each sentence, $\gamma_{\max(i)}^{s_i}$ which represents its hidden state would be passed to RNN story encoder to change embedding sentence to embedding story while e_t is a value that represents the context at position t which then can be used to compute the surprise and uncertainty value. The RNN model is tested with Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM) where contain two and four layers respectively in each sentence and story encoder. The fusion layers and hidden layers for both model variants were 2 and 768 respectively. The result for LSTM was slightly worse but with shorter training time required.

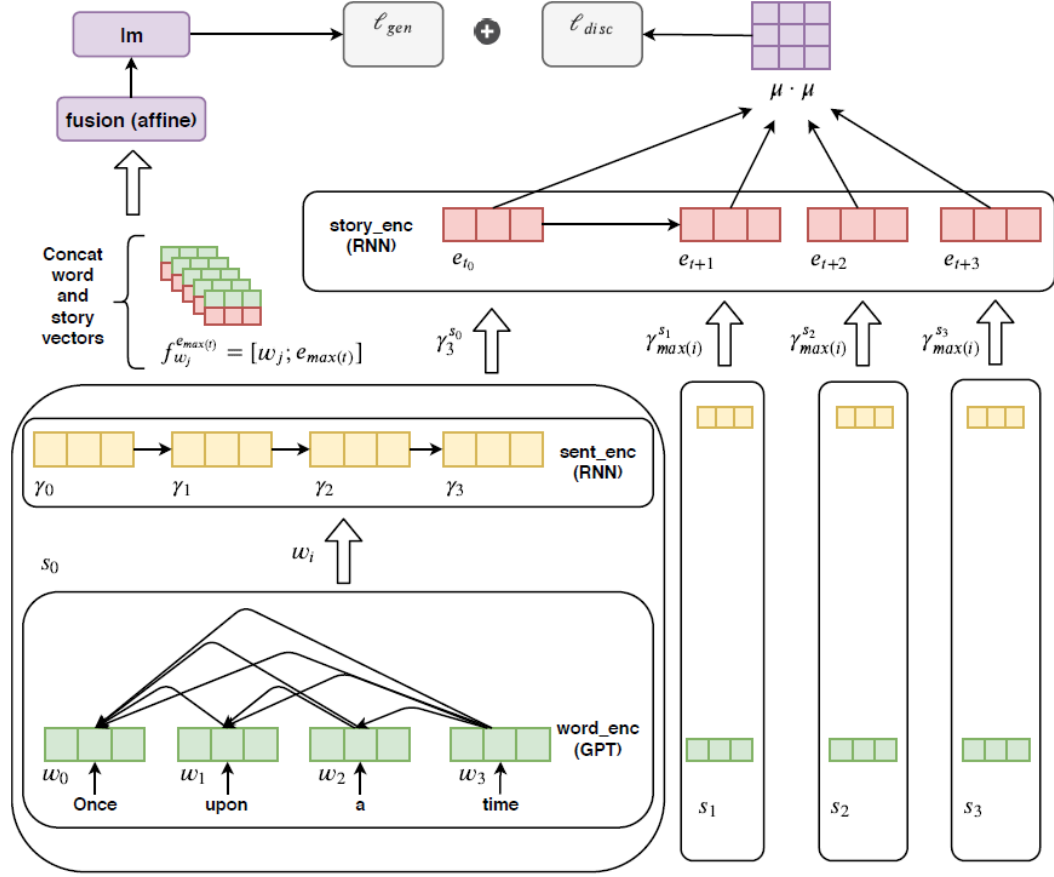


Figure 2-5: Hierarchical Model

2-6 Data Collection and Annotation Methods

One of the papers studied was to introduce an annotated dataset, named Oops! Dataset to perform classification on unintentional human action (Epstein, Chen and Vondrick 2020). The Oops! video dataset was collected from YouTube fail compilation videos and pre-processed using scikit-video to detect the scene boundaries. The total videos collected were 20,338 and the total duration was over 50 hours. The videos were then handed over to AMT for the workers to label the starting frame of the unintentional action in the scenes. To ensure high quality annotation, the researchers have restricted some rules, which was to repeat the annotation process for thrice to ensure consistency. The paper also found that humans are consistent among their labelling decision.

On the other hand, another literature which intended to perform sarcasm classification using SVM mapped with different integration of text, audio and video feature has introduced an annotated dataset prior to its classification work (Castro, Hazarika, Pérez-Rosas, Zimmermann, Mihalcea, and Poria 2019). The annotated dataset was named MUSTARD that stood for Multimodal Sarcasm Dataset. The dataset

was obtained from different sources, consists of sarcastic and non-sarcastic videos, typically from YouTube, multimodal emotion recognition dataset (MLED) and TV show called The Big Bang Theory. Two graduate students were responsible for the annotation process and a web-based annotation interface was provided with video, its corresponding text and sarcasm label request for the annotators to mark their options. Different annotation made for the same scene would be discussed and reconciled among the two annotators while the remaining unsolved cases would be handled by a third annotator to make the final option. The annotation result showed 345 sarcastic labelled videos and 6,020 non-sarcastic labelled videos.

The last paper studied on data collection method was intended to introduce a dataset for violent scenes detection, namely VSD2014 (Schedi, Sjöberg, Mironică, Ionescu, Quang, Jiang and Demarty 2015). The dataset consists of one training set and two testing sets. The training set consisted of 24 movies which would be annotated accordingly while the testing sets were split into two set, where the first set contained 7 unlabelled movies and the second set was made up of 86 YouTube videos. The training set has a total duration of 50 hours 2 minutes and the annotation was done by human in a hierarchical bottom-up approach. There are two groups of annotators from China and Vietnam which each group consists of regular annotators (graduate students) and master annotators (senior researchers). The annotation work done by regular annotators would be reviewed and judged by the master annotators in each group and later on, the work would be presented to another set of master annotators coming from six different countries to make the final judgements. The hierarchical bottom-up approach is presented as Figure 2-6 below:

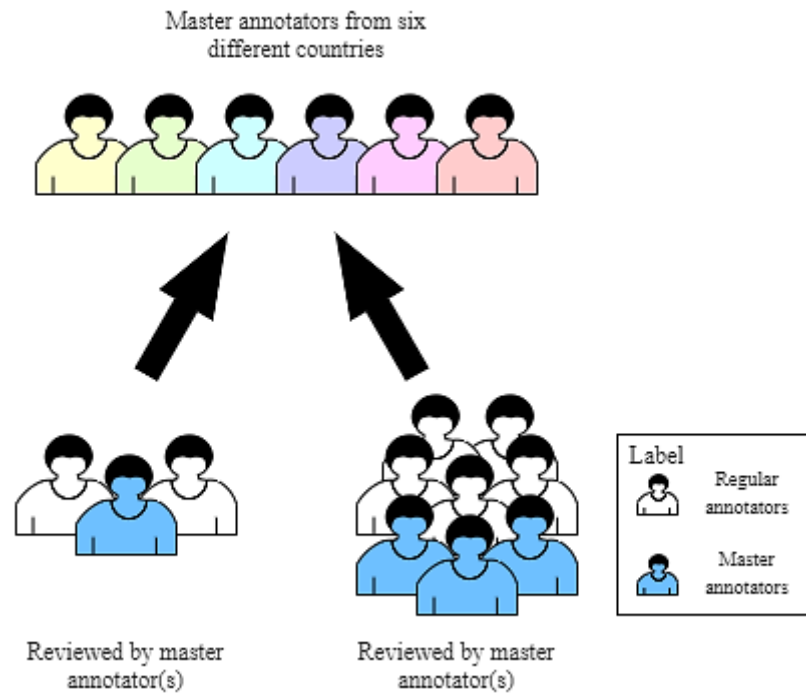


Figure 2-6: Hierarchical Bottom-up Approach

VSD2014 has classified annotation into two types which are binary and multi-categorical annotation. The former annotation contained violent scene while the latter annotation involved 7 visual and 3 audio elements such as blood, fights, presence of fire, presence of guns, presence of cold arms, car chases, gory scenes, gunshot, explosions and screams. The annotations were saved separately in text file and VSD2014 also further extracted the visual and audio features from each movie and stored them independently in mat file. This project would not include features in the dataset as this project is considering using neural network that will extract the video feature by its own.

2-7 Class Imbalanced and Technique to Overcome

Since the annotated dataset used in this project is having class imbalanced, literature review is also done on the effect of imbalanced class towards classification tasks and the corresponding state-of-the-art technique to overcome. In one of the papers reviewed, the paper studied on the effect of resampling on the classification performance (Burnaev, Erofeev & Papanov 2015). In the paper, it is mentioned that when there exists a minority class in the dataset, this makes the model hard to learn about the minority class as there is too little information about it. Also, often the minority class is the

primary interest in the solution and this is what exactly happens in the dataset of this project. The paper concludes that the model performance can only be improved if and only if the right sampling method is selected. Otherwise, the performance may go worse than without resampling.

Also, another paper emphasized on the study of oversampling on time-series data has introduced an Oversampling method to combat the High-dimensional Imbalanced Time-series classification (OHIT) (Zhu, Lin & Liu 2020). The algorithm emphasizes to produce synthetic samples by using estimated covariance matrices. However, the method has not been made available as library so that the public can easily install and use it.

Another popular oversampling technique is synthetic minority oversampling methods (SMOTE). Different from typical sampling techniques such as random under-sampling and random oversampling, SMOTE produces synthetic samples instead of simply duplicating the minority class (The Python Package Index 2020). However, SMOTE cannot accept to oversample time-series data and this motivates this project to oversample the minority class by adding certain noise to the existing data so that the data duplicated is not identical.

2-8 Remarks

In general, scene segmentation used the concept where same scene contains high similarity in features to segment video into scenes whereas the scene classification using machine learning techniques usually determine some rules for different types of scenes so that the machine can follow the pre-defined rules to classify each scene. On the other hand, it is found that none of the literatures showing the work on suspense classification using videos as input and this becomes one of the motivations of doing this project. On top of that, the survey paper revealed that most of the dataset presented were private dataset that are inaccessible to public and this becomes the second motivation for this project, which is to release a public dataset labelled on suspense. To perform a standard data collection process, review on related literatures doing data collection was subsequently done. Last but not least, a new oversampling technique is used in this project in order to oversample time-series data.

Chapter 3 System Design

3-1 Methodology

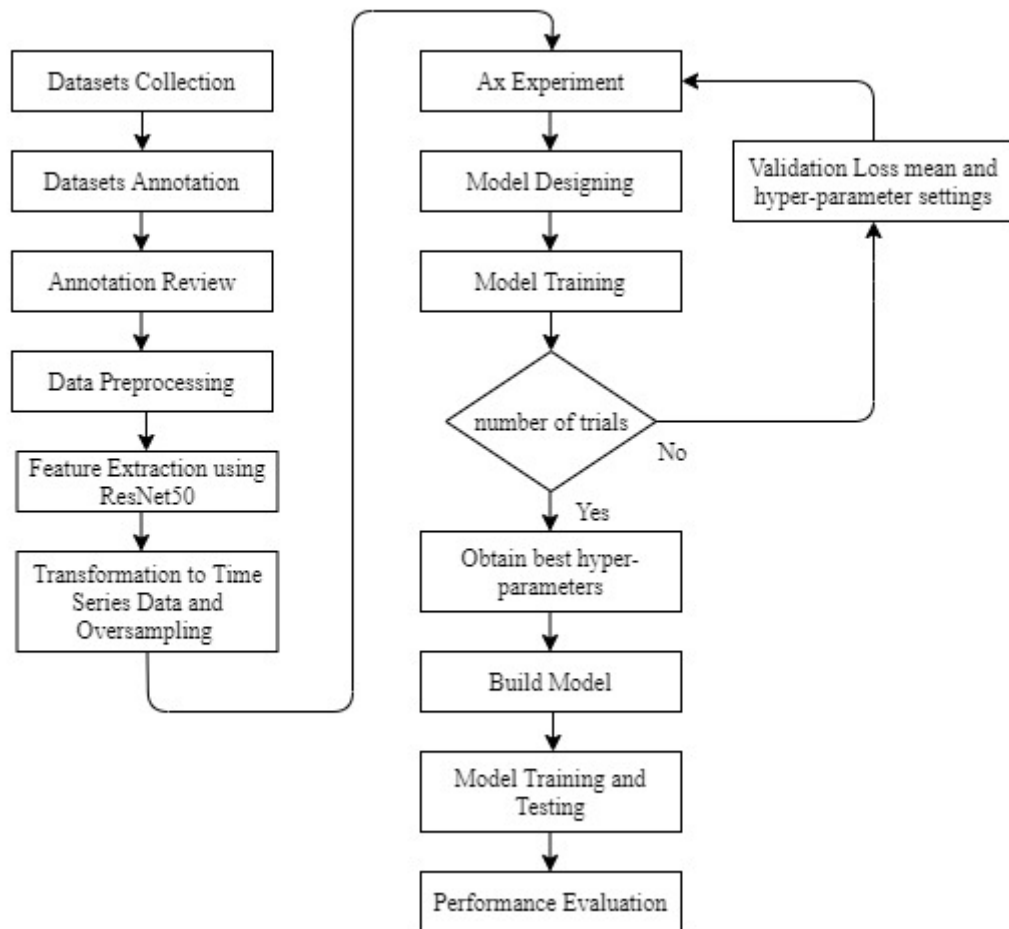


Figure 3-1: General Methodology

Figure 3-1 above is the general methodology proposed for the overall project. Project 1 has covered until the third stage, that is annotation review and this will produce an annotated dataset for suspense classification that will serve as input to the later work in Project 2. The dataset preparation process involves dataset collection and annotation that is to gather all the videos ranged from few categories and annotate the suspense scene accordingly. The annotation results are further reviewed before finalizing the dataset.

In Project 2 stage, data pre-processing is done prior to feature extraction to identify start frame and end frame of each suspense and non-suspense scenes. Frame extraction is then done and some computation is done on calculating the frame of each videos and the difference in term of the number of suspense and non-suspense frames

in each video. Feature extraction is then done using ResNet50 to extract features from extracted frames and provide high-level features to the model later. High-level features can ensure the model to learn better than using low-level features extracted directly from the images. Then, transformation of data to time-series data that suitable to be fed into the RNN model and oversampling are done. The data is then split to training and validation data where half of the training data will go through the experiment created Ax. In each trials, a different model architecture with different hyper-parameter settings will go through training and the best hyper-parameter will be selected after completely run through all the trials. The best hyper-parameters setting is selected based on the experiment objective determined by the developer, which is the mean of the validation loss in this case. With the best hyper-parameters and model architecture, the model will be built and trained with whole set of training and validation data. Performance evaluation will then be done after testing the model with unseen data during the training, which is the testing set. Testing set will go through the data pre-processing, feature extraction and data transformation stage before it can be used to evaluate the model.

3-2 Tool and Technologies used

The software used for dataset preparation phase includes *youtube-dl* and *Quick Time Player* which are available and free online. This provides feasibility for the public to recreate the dataset using the same software. Other than that, Python is the programming language would be used to build the SSD model and the associated libraries is as follows:

- python (version 3.8.5)
- numpy (version 1.19.2)
- opencv-python (version 4.5.1.48)
- ax-platform (version 0.1.20)
- tqdm (version 4.50.2)
- tensorflow-gpu (version 2.4.1)
- cudatoolkit (version 10.1.243)
- h5py (version 2.10.0)
- pillow (version 8.2.0)

Note: Installing different version of libraries stated above may result in certain incompatibility issue.

3-3 Standard Evaluation Metrics for Model Performance

To ensure the annotation quality, the annotation process is carried out independently by two annotators. Nevertheless, the annotators are provided with some benchmark descriptions on the suspense scene before the annotation. The two annotation sets are reviewed and the disagreements are further discussed and reconciled to produce the final annotation set.

On the other hand, typical performance evaluation metrics such as accuracy, precision and recall would be used to evaluate the classification performance of the model. However, since the project is dealing with imbalanced dataset, accuracy may not be a good metric to be evaluated as a model that wrongly predicts all the minority class may still result in high accuracy and yet the model actually performs badly in the classification task. Nevertheless, accuracy will still be included as a supplementary metric. Other additional evaluation metrics includes F1 score, Cohen's Kappa and ROC AUC score. The evaluation will be more focused on the precision and recall score to ensure the model is able to classify the minority class correctly.

3-4 Overall Architecture Flow

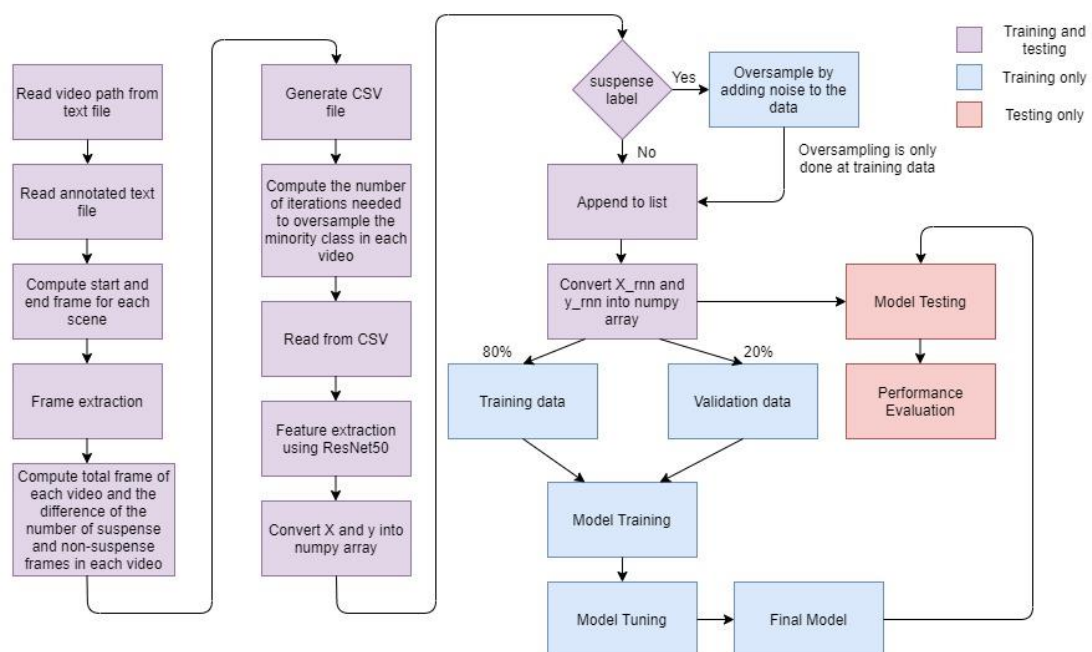


Figure 3-2: Proposed Generic Architecture Flow

Figure 3-2 above shows the generic architecture of the proposed system. Prior to this, data collection and annotation are done to label the suspense scene in the videos. The

data is first downloaded from YouTube platform and a list of frame numbers indicating the suspense scene in each video are saved into text files. The annotation sets are reviewed and reconciled between two annotators and eventually the dataset is ready to be input to the system. The system proposed is using Recurrent Neural Network (RNN) Long Short-Term Memory layers to construct the model where its characteristics are to handle sequential data and memorize what is in the previous layer and feeding forward to the next layer. The video inputs are a series of sequential data over time that would be pass to the model for training. On the other hand, the annotated set helps to indicate the suspense scene so that SSD model could learn based on the labelled data because RNN is a type of supervised machine learning.

During the data pre-processing phase, frames are extracted from the videos collected and features extraction is done through ResNet50 to obtain high level features that is useful for the later SSD model to learn. Before the training and optimizing process begin, the extracted features will first go through a function to transform data dimensionality to be match with the input shape for RNN. Also, oversampling is done here because there is too less information about the suspense frames and the SSD model tends to fail in learning efficiently if oversampling is not being done.

After that, half of the training data will be used as the input of the experiment created using Ax with the purpose of finding the best parameters in the search space. After a number of iterations, best hyperparameters will then be retrieved, a new model will be built based on the hyperparameters settings and training process will be subsequently executed. The training model will be saved for later use.

For model testing, the testing videos need to go through the same process as training videos, except no oversampling is being done for testing data. This is because the model only needs to learn the behaviour of the minority class during training and hence, oversampling needs not to be done for testing set. The testing data in time-series shape will then be used to predict the class of each frame using the model that trained previously. Finally, performance metrics such as accuracy, precision and recall will be used to evaluate the model performance.

3-5 Dataset

3-5-1 Dataset Searching

In order to start on this project, the first thing performed is the dataset finding process. A couple of online dataset platforms have been gone through and the links of the associated websites are provided in Table 3-1 below:

Websites	Links to follow
DeepMind	https://deepmind.com/research?filters=%7B%22tags%22:%5B%22Datasets%22%5D%7D
Video Dataset Overview	https://www.di.ens.fr/~miech/datasetviz/
Datalist	https://www.datasetlist.com/
UCI Machine Learning Repository	https://archive.ics.uci.edu/ml/datasets.php
Kaggle	https://www.kaggle.com/datasets
Quantum Stat	https://datasets.quantumstat.com/#/

Table 3-1: Online Dataset Sources

Nevertheless, this project fails to get an appropriate dataset from the platforms above. There is no such dataset where the videos are labelled based on suspense. Hence, this project decides to prepare the dataset prior to modelling the classification system.

3-5-2 Data Collection and Annotation

In this stage, data is collected from YouTube using *youtube-dl*, an open-source software developed to allow downloading videos from YouTube to local computer. The software is available on GitHub together with installation guides and documentation. The following URL is directed to the software's main page:

<https://github.com/yt-dl-org/youtube-dl>

A total of three video categories are selected, which are cooking, sports (football) and entertainment videos. As mentioned earlier, not all types of videos are covered in the dataset because as of now this project aims to only develop one model to fit all videos. Some specific queries are used to return the related videos. Table 3-2 below contains the queries used for each category and the general description on the predicted suspense moment before annotating them.

Category	Queries searched	General description on predicted suspense scene
Cooking	<i>MasterChef cooking</i>	<ul style="list-style-type: none"> scene before announcing the cooking result
Sports (football)	<i>fifa football match</i>	<ul style="list-style-type: none"> scene about making a goal
Entertainment	<i>Great Escape Season 2</i>	<ul style="list-style-type: none"> scene that feels certain level of uncertainty

Table 3-2: Data Collection Guides

In this project, suspense can be seen as an extent of uncertainty. The description will serve as a benchmark for the annotators to mark the suspense scene yet it is still possible for the remaining scenes to be labelled as suspense. The data collection process starts by retrieving each video uniform resource locator (URL) from YouTube. A batch file is created to place the download command together with the video URLs to automate executing all the download commands. Running the batch file will open the terminal and the command written in the file will be executed sequentially. The download command is written in a form of:

```
youtube-dl -o "%%(id)s.%(ext)s" <video_URL>
```

where the video id becomes the name of the video and the associated batch script file can be obtained from GitHub.

The dataset prepared consists of total 29 videos from the three different categories and has a total duration of 31 hours and 55 minutes. All the videos obtained are available on YouTube. Due to the absence of Creative Commons License under these videos, the distributed dataset would not contain the video data but only the associated annotation. The selected videos include 16 videos from *MasterChef Australia Season 1*, 4 videos from *FIFA World Cup Russia 2018* and 9 videos from *Great Escape Season 2*. From each category, one to few videos are highlighted in light blue and selected for testing. Table 3-3 below illustrates the dataset details.

Video details	Video ID	Total duration
MasterChef Australia Season 1	eFDPJlvSrU8	2082.62
	weUenGaF8cs	2814.75

	jdezKogGKW0	2826.77
	3eDBuzDgLP0	1499.10
	ep0FVRUzweg	4279.25
	WbN8Qwa8fWA	2806.94
	46xKDm6OFHw	1474.30
	Z5i6jdYkHcA	2728.79
	LbO-qvy9JIY	1435.53
	HfUwxxpzHrs	2800.37
	YXUmL753tYo	2680.91
	5V0LsqPzDTU	1451.41
	p8fqqMi26EI	2741.25
	ms4hO0h1dOg	1395.15
	HktqGcJQA4w	2538.34
	CFWSgo-ftrQ	2774.51
FIFA World Cup Russia 2018 i. France vs Croatia ii. German vs Mexico iii. Brazil vs Belgium iv. Portugal vs Spain	7Fau-IwbuJc	6841.43
	3fYpcapas0k	6536.88
	5OjfbYQtKtk	6650.22
	Xhu5Bz1xDf0	6625.16
Great Escape Season 2	XLWx0_I1qLQ	5368.17
	9ifbXe4TsSc	6013.48
	5dCpEzu1ka0	5513.25
	JjVO—nexcA	3961.93
	smBYcpMVeuo	5535.45
	7jpruuvWK7E	5558.42
	lpS9mYfvKIY	5300.14
	xBUu1q-1KSI	4856.29
	_y3rFsvz8qQ	7839.66
		Total = 114, 930.47 (31h55m)

Table 3-3: Dataset Statistic

The ready downloaded videos are annotated at video frame level and the annotation only involves binary classification. The annotation process is done by two annotators independently using *Quick Time Player* to retrieve the frame number. The annotators then discuss and reconcile the annotation sets to produce a final annotation set. The annotations are recorded in a format of:

```
start-frame end-frame
```

The annotation for each video is saved separately in text file using video id as the filename and the final annotation set is available in GitHub.

3-6 Data Pre-processing

3-6-1 Frame Extraction

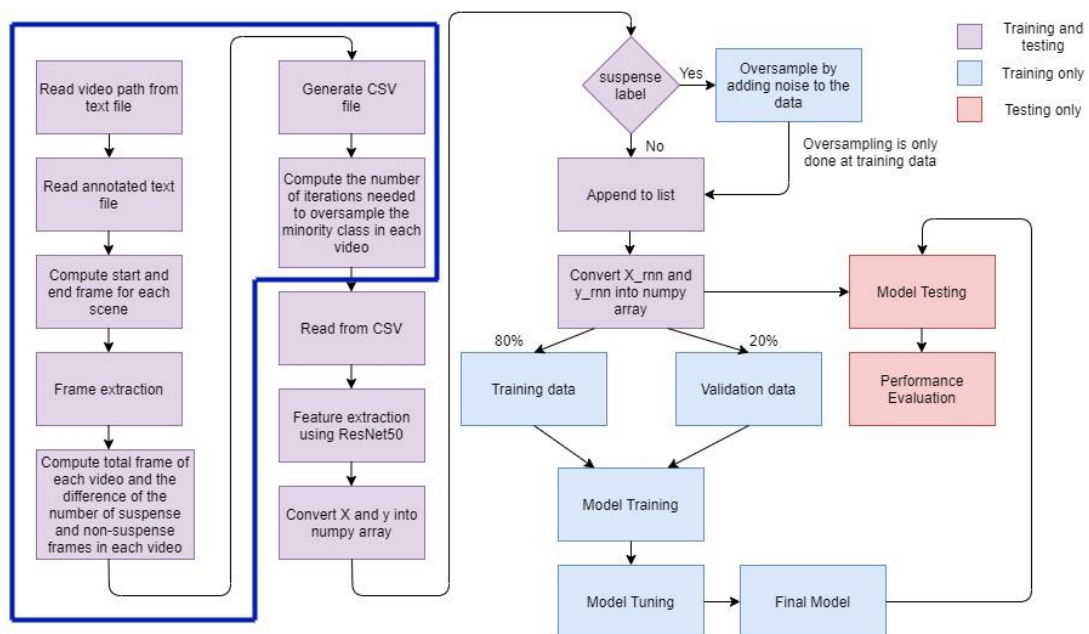


Figure 3-3: Frame Extraction Flow

Prior to frame extraction, some pre-processing needs to be done to ensure necessary information is computed and saved for later use. The pre-processing starts by reading the text file containing the video paths into the system. Then, with the video name information, the system is able to read the annotated text files of each video since the annotation files are named using the video id. At the same time, the system will compute the number of scenes exists in each video, start frame and end frame of each scene and return the whole data as DataFrame. With the information in the DataFrame, the system is now able to extract the frames from each video and name the frames according to the

class, video id and frame number they belong to. The naming convention is shown as follows:

```
<class>_<video id>_<frame number>.jpg
```

Each frame either fall under “suspense” class or “non-suspense” class, and the frequency of the frame to be extracted is set to be 1 frame for each second. All the extracted frame will be stored into a folder named “data”. Rerun the frame extraction process will then delete all the frames in the directory and re-extract again from the videos.

The next step is arrange the videos frames so that all the frames are sorted by videos and the frames in each video are well sorted based on the time sequence. This is done by referring to the frame name that consists of video id and frame number. The sorted data and its corresponding class label will saved into a comma separated value (CSV) file named ‘train_new.csv’. At the same time, some computation is done to obtain required information such as the total number of frames in each video and the difference of the number of frames in each video between suspense and non-suspense scenes. These two numpy array will be used to compute the number of times each suspense scene in each video need to be oversampled to make the class balanced.

3-6-2 Feature Extraction and Time Series Data Transformation

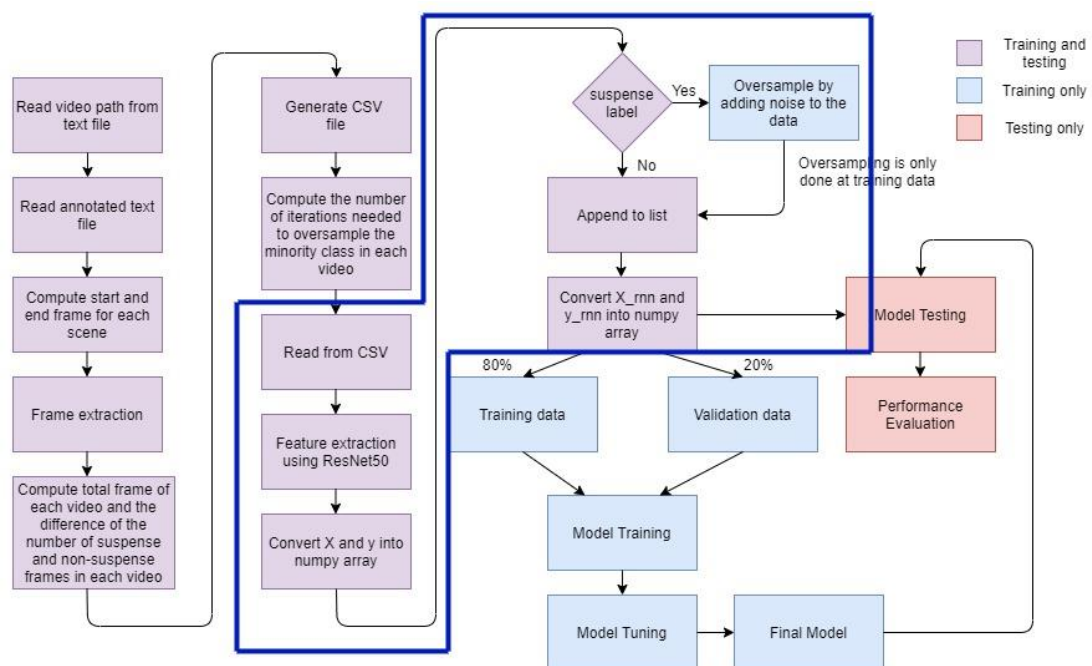


Figure 3-4: Feature Extraction and RNN Shape Transformation Flow

This section starts with reading the 'train_new.csv' from the local. Then, a pre-trained model, ResNet50 plays the role as feature extractor to extract the features from frames and output high level features that going to be fed into the model. ResNet50 has been set with `include_top = False` so that the fully connected layer is removed from the model as the goal is only extract features using the pre-trained model instead of classifying the frame at this stage. All the layers have been set to not trainable, this means the model weight would not be updated during feature extraction. Furthermore, the pooling layer used here is average pooling and the input shape of the image is (224, 224, 3). With ResNet50, the output of the image features will be reduced to (2048,).

The subsequent processes are RNN shape transformation and oversampling. The function takes in required parameters such as the total number of frames each video owns, total number of training videos, images features extracted from ResNet50, class label and lastly the number of times each suspense frame need to be oversampled in each video. After that, the function will try to convert the data into RNN shape with 3 dimensions which is (number of samples, sequence length, number of features). In this project, the sequence length defined is 20, this means the past 20 frames of each frame k are added to the second dimension of the RNN shape and each individual frame has 2048 features in total. At the same time, for each suspense frame, frame will be oversampled iteratively until reaching the number of loop count computed previously. The suspense frame is oversampled by adding some noise to the suspense frame and add it as RNN data. In the end of the function, the function will return X and y in RNN shape where X is having shape of (number of samples, 20, 2048) and y is having shape of (number of samples,). The data will be split into training and validation data to serve for later use.

3-7 Model Optimization and Training

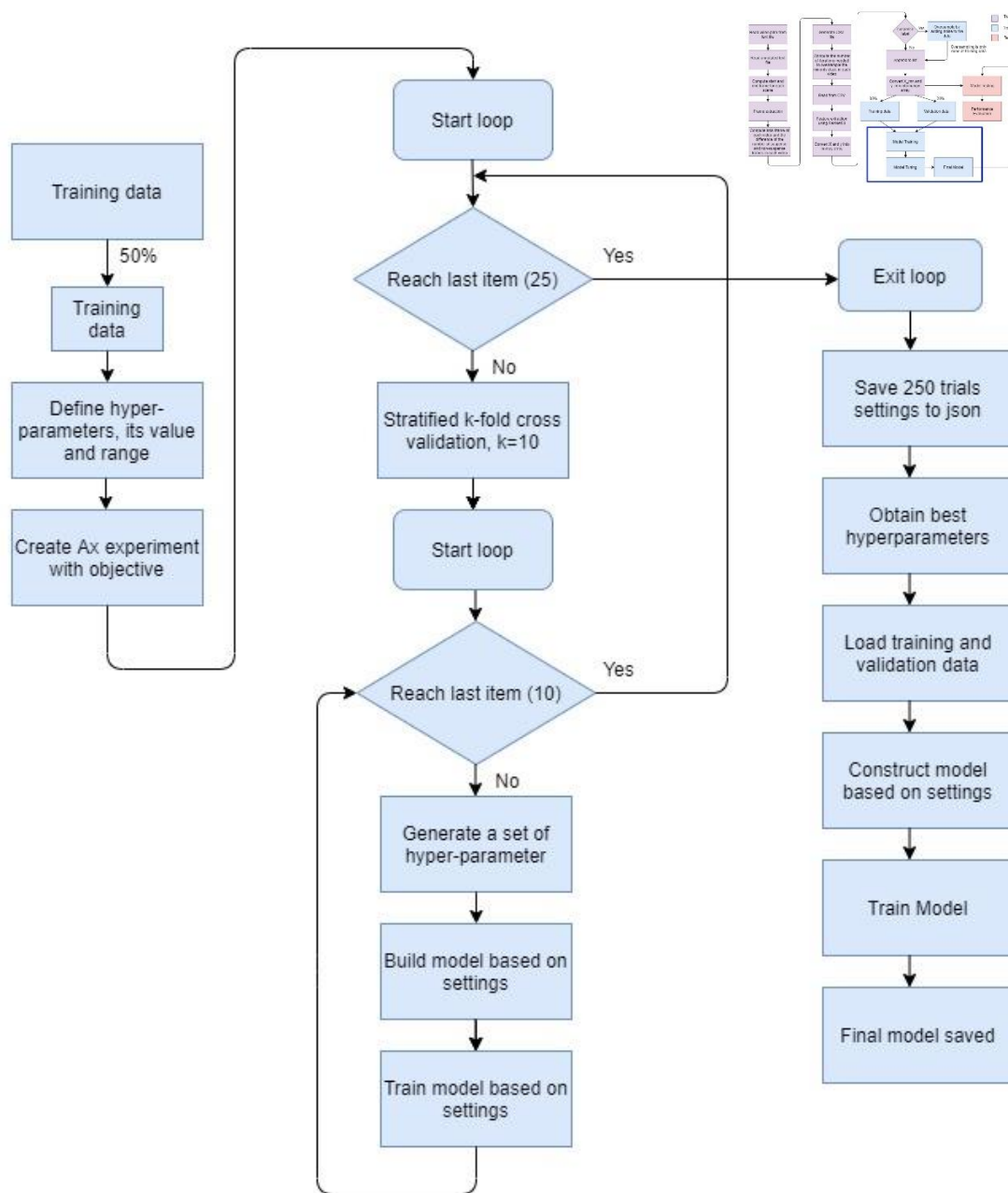


Figure 3-5: Model Optimizing and Training Flow

Half or 50% of the training set will be utilized in the Ax experiment. The reason why not utilizing 100% of the training data is simply because loading 100% of training data to random access memory (RAM) and with Ax experiment prepared will exceed the maximum RAM space the machine could allocate. Alternatively, only half of the training data is utilized in the experiment in order to find the best hyperparameters settings. Before the experiment is started, some target hyperparameters are defined with a certain range or some choices to limit the search space and following is the settings:

- Learning rate – ranging from 0.001 to 0.2
- Dropout rate – ranging from 0.1 to 0.5
- Alpha (to determine number of neurons in each LSTM layer) – ranging from 2 to 10 (set to be integer type)
- Number of LSTM layers – 2/ 3
- Number of epochs – ranging from 10 to 25 (set to be integer type)
- Batch size – 64/ 128/ 256
- Optimizer – Adam/ RmsProp/ SGD

The next step is to create an Ax experiment with the objective to find the hyperparameter settings that result in lowest validation loss mean and the training iterations begin. To avoid overfitting, stratified k-fold cross validation is set so that in each training, the validation data is different. Since the `n_splits` is set to be 10, this means the 50% training data will be split into 10 portions, and in each iteration the validation data will be selected from one of the 10 portions. The figure below illustration how k-fold cross validation works.

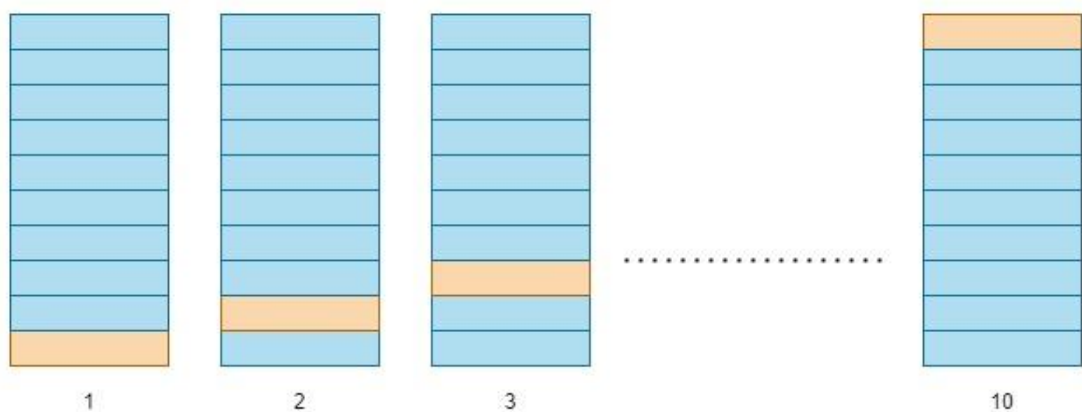


Figure 3-6: Stratified K-fold Cross Validation Illustration

Stratified k-fold cross validation helps to maintain the ratio of class in both training and validation set. In this project, it is assumed that no defined architecture before running the Ax experiment. Dynamic model architecture will be used in the optimization in Ax experiment and the best architecture settings will be selected.

The tuning phase consists of two loops where the outer loop is set to be exit at the 25th iteration while the inner loop has 10 iterations in total since `n_splits` = 10. At outer loop, each iteration will split the training data into 10 portions and each

combination of training and validation data will enter the inner loop once. At inner loop, `get_next_trial()` built in function in Ax Client is called to return a set of trial parameterization to be used in the training and a trial index. Then, the hyperparameters such as alpha and number of LSTM layers are used to build a new model architecture in each new trials. The formula on how the number of neurons in each layer is computed is shown as follows:

$$N_h = \frac{N_s}{\alpha * (N_i + N_o)}$$

where N_s represents the number of samples in the training data, N_i represents the number of input neurons and N_o represents the number of output neurons (Eckhardt 2018). N_i and N_o are 20 and 1 respectively and alpha is a factor between 2 and 10 where it is selected by Ax Client in each new trial. Whereas for the number of neurons in each hidden layer, this project follows the rule of thumb from Heaton Research, that is the number of hidden neurons should be 2/3 the size of the input layer and add with the size of output layer (Heaton Research 2017). Hence, in each LSTM, the number of neurons are at most 2/3 of the number of neurons in the previous layer. Also, deeper LSTM network may not bring significant result, hence this project decides to stop the LSTM layer at depth of 3.

After building the model, training begins by using the rest of the hyperparameters settings and validation loss mean will be returned after each training in order to track the best hyperparameters in all trials. Since outer loop has 25 iterations and stratified k-fold cross validation has split the training data into 10 portions, there are total 250 trials in the experiment and 250 different combination of hyperparameters will be generated to go through the training process. The trials results are saved into json file and the program will get the best hyperparameters from 250 trials using the built in function `get_best_parameters()`.

Then, the real model training process starts. The system will need to first load the whole training and validation set into the RAM and begin to build and train the model based on the best hyperparameters found. In this training phase, this project utilizes tool such as `ModelCheckPoint` and `EarlyStopping` where the former helps to monitor the validation loss and save the model with lowest validation loss. Whereas for `EarlyStopping`, it is a technique to avoid overfitting by stopping the training process

before it reaches the maximum number of epochs. The stop condition depends on what is the target to be monitor and over how many epochs the training will be stopped if there is no improvement on the target event monitored. Eventually, the well-trained model will be saved for later use in testing.

For more information about Ax, please visit: <https://ax.dev/api/service.html>

3-8 Network Architecture

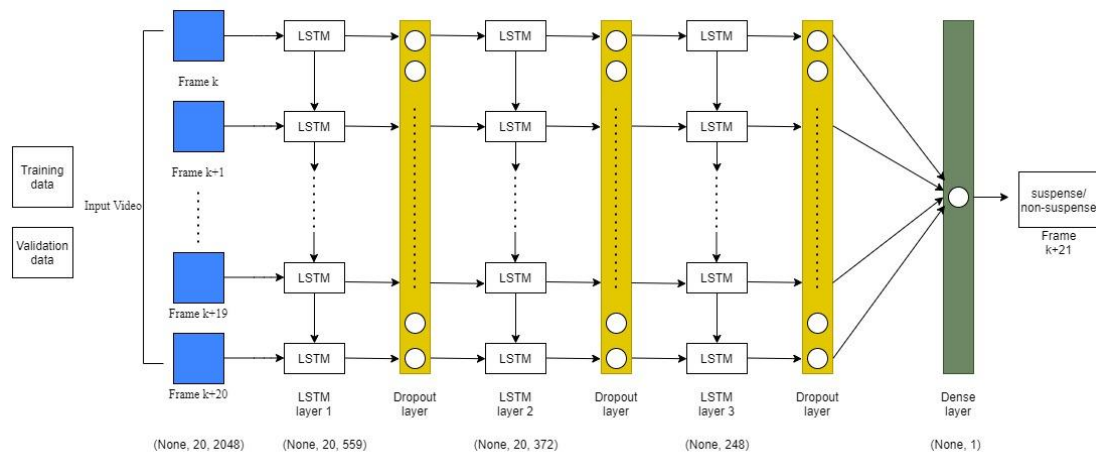


Figure 3-7: Network Design Architecture

This section will be further elaborating on the network architecture of the chosen model. Each video frame has the shape of (sequence length, number of features) where sequence length is 20 and number of features is 2048. Then each of the video frame in the form of sequence data will be fed into the chosen model. The model has total three LSTM layers with one dropout layer after each LSTM layer. The model ends with a dense layer which connects all the neurons to the dense layer and output the result as binary class. In each LSTM layer, the number of neurons are reduced by 1/3 as compared to the previous LSTM layer and each dropout layer has the number of neurons same as the LSTM right before the dropout layer. Since the training data has a total of 117,408 samples and the alpha value selected is 10 in this case, the neurons at the first LSTM layer is 559 and for the subsequent LSTM layers, the number of neurons are reduced by a factor of 3. Below is the figure showing the model architecture.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_3 (LSTM)	(None, 20, 559)	5831488
dropout_3 (Dropout)	(None, 20, 559)	0
lstm_4 (LSTM)	(None, 20, 372)	1386816
dropout_4 (Dropout)	(None, 20, 372)	0
lstm_5 (LSTM)	(None, 248)	616032
dropout_5 (Dropout)	(None, 248)	0
dense_1 (Dense)	(None, 1)	249
Total params: 7,834,585		
Trainable params: 7,834,585		
Non-trainable params: 0		

Figure 3-8: Model Summary

During the training process, the input at time t are always the combinations of input at time t and $t - 1$ as LSTM will save the input from the previous layer to feed it to the current layer. The neural network will first forward the data and produce the output. Similar to typical neural network, loss function will be used to compute the difference between actual and predicted output. In this case, binary cross entropy function will be used to compute the loss in line with the binary classification. The network will then be back-propagated to update the weights accordingly. The design of LSTM cell will decide which information to keep when going through the forget gate, and the output from the forget gate and input gate will be used to update the cell state which will then be forwarded to the next LSTM cell as shown in Figure 3-7. Also, the output gate helps to decide the next hidden state to be fed to the next LSTM cell. At the dropout layer, some neurons will be neglected to avoid overfitting and the process will be repeated until reaching the fully connected layer.

After that, dense layer will compute the probability of that particular sample using sigmoid activation function. A sigmoid function as shown below will be used to compute the probability of the scene to be suspense or non-suspense.

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

If the probability happens to be higher than 0.5, the class predicted will be 1 and otherwise 0.

3-9 Expected Output

The output of this model is a prediction on the 21st frame counting from the start frame of the sequence data. This model maps previous 20 frames features to the model and expect the model to return the prediction of the class the 21st frame belongs to. Each of the frames will be predicted except the first 20 frames in each video because there is insufficient past information to be fed into the neural network. Furthermore, since the model is a binary classification task, the returned predicted class is either 1 or 0, where 1 indicates suspense and 0 indicated non-suspense.

3-10 Implementation Issues and Challenges

The difficulties faced in this project is to determine the types of videos to be collected. The annotated dataset would be eventually fed into only one neural network instead of building different neural network models for each category defined. The reason of not doing the latter approach is because the video categories originated from YouTube are general and not always accurate as they could be decided by the uploaders themselves. This leads to the problem where the accuracy may still suffer even this project is doing multi-model training. Hence, this project will only be doing one neural network model that fits best within the dataset. Nevertheless, this project would like to include more than one type of videos for training to ensure certain level of robustness. Although this project may not able to achieve high accuracy in detecting the suspense scene, the dataset introduced and result obtained are still significant from the novel aspect.

Other implementation challenge faced in this project would be the class imbalanced issue that results in poor model performance as the model fails to learn on the minority class. On top of that, typical oversampling techniques are not applicable because the project is working on time-series data/ sequence data where the sequence plays a significant role in the data. Typical oversampling technique cannot take RNN input shape that consists of 3 dimensions and oversampling before the data is transformed into RNN shape making the sequence information to be lost. Hence, this motivates the project to oversample the time-series data by adding certain noise to the

existing data from the minority class. Nevertheless, the impact of this oversampling technique towards the model, especially for time-series data still remains questionable.

3-11 Timeline

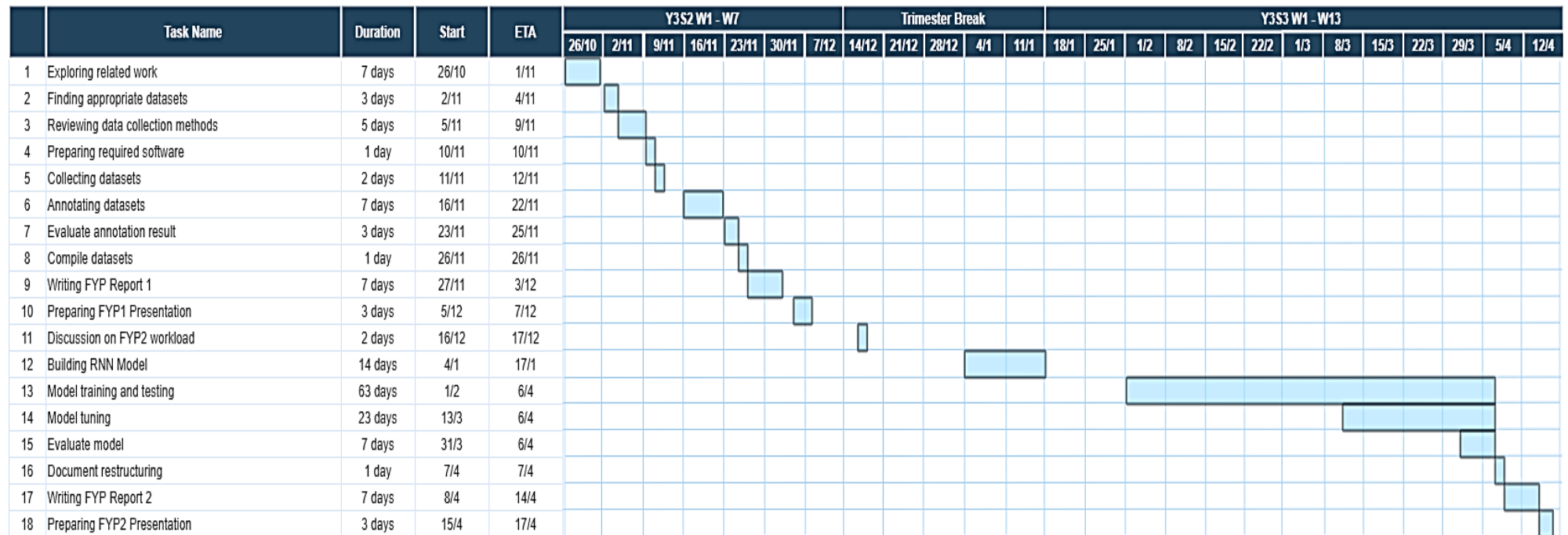


Figure 3-9: Timeline

This project is estimated to be completed within two trimesters which consists of 25 weeks including trimester break and to be submitted on Year 3 Trimester 3 Week 13 Friday. Project 1 mainly focuses in exploring the former works, collecting data and annotate the suspense dataset. Some of the parts in proposal is amended in Project 1 report to enhance and fit with the current project work. The report is expected to be submitted on 4th December attached with the annotated dataset. The discussion on FYP2 would be done in the mid of December and the related project work would start with modelling during the trimester break.

Chapter 4 Model Implementation

4-1 Project Setup

Prior to process any data, download *youtube-dl* software package and the videos download script from the GitHub repository. Windows user will need to download the batch file with extension .bat while Mac user will need to download the file with extension .sh. Windows user can simply click on the batch file to download the dataset required while Mac user will need to open a terminal and execute the script by typing “bash download.sh”.

After acquiring the dataset, the dataset is split into training videos and testing videos, where training videos will be stored in the folder named “videos” and testing videos will be in the “testvideos” folder. The annotation files for training and testing are split where training annotation files are in “textfiles” folder and testing annotation files are in “testfiles” folder. After done doing all these, there are two text files named trainlist01.txt and testlist01.txt to store the list of videos paths for training and testing. All the annotation files and text files can be obtained from the GitHub repository. The, the proposed model can be trained either using python file that executing via terminal or ipynb file through Jupyter Notebook. Prior to execute the files, there are some pre-installation needs to be done.

```
conda install python=3.8.5
pip install opencv-python==4.5.1.48
pip install ax-platform==0.1.20
pip install tqdm==4.50.2
conda install -c anaconda numpy
conda install -c anaconda tensorflow-gpu
conda install -c anaconda cudatoolkit
conda install -c anaconda h5py
conda install pillow
```

The commands above are needed to be executed in the terminal to install the libraries required.

Table 4-1 below shows some information that is constant throughout the experiment:

Number of training videos	20
Number of samples before oversampling	80569
Number of samples after oversampling	146760 (117408 for training, 29352 for validation)
Number of LSTM layers	3
Batch size	64
Optimizer	SGD
Number of epochs	22
Dropout rate	0.2792
Learning rate	0.1669922
Number of neurons in the 1 st LSTM layer	559
Number of neurons in the 2 nd LSTM layer	372
Number of neurons in the 3 rd LSTM layer	248

Table 4-1: Experiment Settings

4-2 Build the Project Steps by Steps

In this section, the cases for python file and ipynb file are the same and hence the steps mentioned in the following are applicable to both files unless stated.

First, running *data_preprocessing* file can help to read the frames from videos and arrange the frames by videos and frame numbers. The arrangement of the frames is a compulsory and important step in RNN pre-processing because this project is working on multiple videos and RNN takes in the past frames of the target frame to predict the class of the target frame. Hence, the sequence information cannot be lost during frame extraction. A CSV file named *train_new.csv* helps to preserve the sequence information for each video.

Then, running *feature_extraction* file helps to extract features from the images and reduce the size of the features through ResNet50. The feature extraction process is done by following the sequence of the frames in '*train_new.csv*'. Hence, the sequence information for the output features and class label at this stage is still preserved.

After that, *transformation_to_rnn* file will take in several parameters, which includes the total number of frames in each video, the number of videos, image features

and class label which represents X and y respectively and finally the number of times each suspense frame to be oversampled. With the required parameters, *transformation_to_rnn* will transform the image features to become a time-series data where each of the samples carries image features of the past 20 frames, whereas y is the actual class annotated for the 21st frame. The training set will be split into training and validation set with the ratio 8:2.

Subsequently, Ax experiment can come into place by running *ax_experiment* file. The experiment will utilize half of the training data to find the optimal hyperparameters settings in the defined search space. Ax utilizes acquisition function, typically expected improvement (EI) to sample the next set of hyperparameters with the goal to improve over the current best (Lambert 2020). This means Ax does not simply generate the hyperparameters set randomly but instead Ax uses some technique to learn during the iterations and always suggest the hyperparameters set that Ax thinks will perform better than the previous iterations. In the end, all the trials will be saved into a JSON file for future analysis and the best hyperparameters set is saved into a pickle file.

The last step in training phase is to run the *training* file that will load the whole training and validation set as well as the optimal hyperparameters set found. Nevertheless, the best hyperparameters set found seems to have high learning rate which is approximately 0.16699. Hence, the project decides to reduce the learning rate before start training the model. In fact, the initial optimal hyperparameters set is tested but the result is not convincing. Since the learning rate is reduced, the number of epochs need to be increased as the model now requires more steps to improve the performance. With this settings, *training* file is executed and the ModelCheckPoint for the minimum validation loss and the complete model are saved for later use.

After completing the training phase, testing stage begins by running the *test_preprocessing* file to pre-process the testing data into time-series data. Subsequently, *testing* file is executed to load the trained model and predict the class of the testing set.

Chapter 5 Experiments and Results

5-1 Hyperparameters Optimization

This section will discuss about the hyperparameters optimization after getting the optimal hyperparameters from Ax experiment. Previously, smaller size of training data is fed into the Ax experiment and the learning rate will only be decayed if the initial learning rate is larger than 0.1 and otherwise the learning rate will not be reduced. This causes the learning rate to be high across the training process and the model tends to learn too fast and result in overfitting issue, where the model is able to achieve training accuracy of 0.9976 and validation accuracy of 0.9967 during the training process. The following is the training and validation loss of the initial model across the training process.

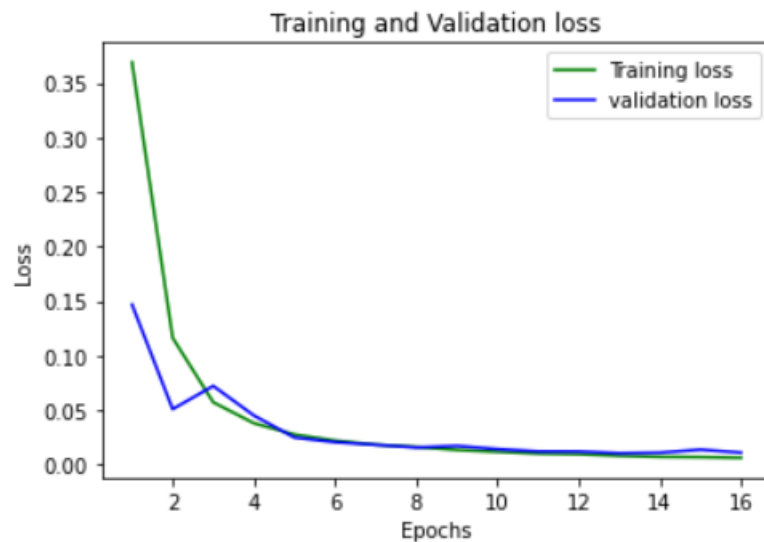


Figure 5-1: Training and validation loss for initial model

Yet, the model still performs worse prediction on testing set where only 122 suspense frames are predicted correctly. At the same time, the model classifies 1422 suspense samples to be non-suspense and 602 non-suspense samples to be suspense. The following is the illustration of the confusion matrix on testing set.

		Actual class	
		Negative	Positive
Predicted class	Negative	23794	602
	Positive	1422	122

Table 5-1: Confusion matrix for initial model

Some inferences have been made towards the model performance and this project decides to keep most of the hyperparameters settings meanwhile manually tune some of the settings. The settings to be altered includes learning rate which most of the inferences are made based on the high learning rate used throughout the training process. Hence, this project decides to reduce the learning rate by dividing the initial learning rate with 10000. Also, previous settings only cause an exponential decay when the learning rate defined is larger than 0.1 but this will be always false in the new settings. Therefore, the algorithm is modified so that each learning rate that fed into the neural network go through an exponential decay in certain number of steps.

With learning rate as low as 1.6699×10^{-5} , increase number of epochs in training is a must as with current settings, it can be foreseen that the model will train slower and hence it requires more epochs to obtain a greater performance. Few different settings are set to carry out the training process and this will be further discussed in section 5-2.

5-2 Performance Analysis

	model_0743	model_1130	model_1312	model_1607
Number of epochs	300	150	300	500
Initial learning rate	1.6699×10^{-5}			
Exponential decay settings	decay each 40000 steps by 0.96	decay each 10000 steps by 0.96	decay each 10000 steps by 0.96	decay each 10000 steps by 0.96

Table 5-2: Difference in settings for different model

Table 5-2 above shows the different settings used in in each of the models. The settings that are not being mentioned remains the same as what have been discussed in Chapter 4. The four models above have been trained and their corresponding performance during training are illustrated as follows:

	Accuracy	Precision	Recall	F1 Score	Cohen's Kappa	ROC AUC Score
model_0743	0.9585	0.6410	0.9615	0.7692	0.7474	0.9902
model_1130	0.7263	0.1692	0.7180	0.2739	0.1782	0.8005
model_1312	0.7850	0.2182	0.7703	0.3400	0.2567	0.8602
model_1607	0.8064	0.2433	0.8017	0.3733	0.2955	0.8875

Table 5-3: Result on training set based on different model

From the training result, it can temporarily concludes model_0743 is the model performs the best within the four models. It performs great performance in the metrics such as accuracy, recall and ROC AUC score, which are almost near to 1. Whereas for other metrics such as precision, F1 score and Cohen's Kappa score, although the performance are slightly worse the than previous three metrics but they still outperforms as compared to the model performance given by the rest of the three models. The second best model is model_1607, followed by model_1312 and model_1130, where the comparison of the performance metrics between the three models do not have huge difference as compared to the model_0743. The following are the graphs illustrating the loss and accuracy for training and validation set during the training phase.

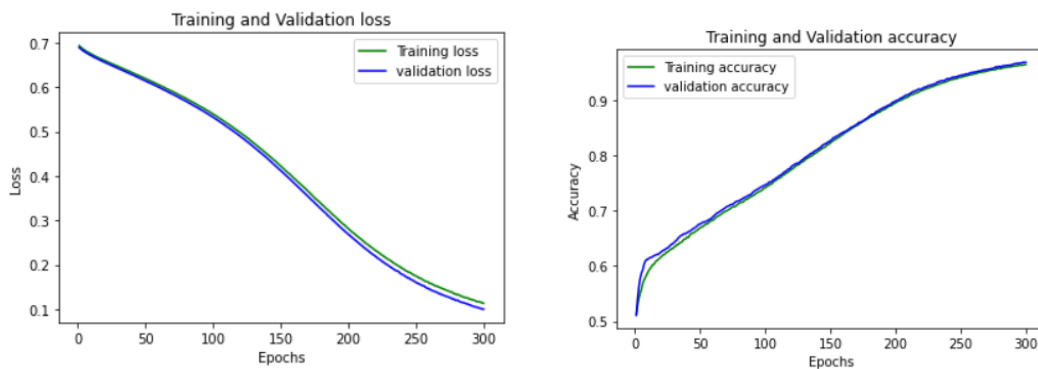


Figure 5-2: Loss and accuracy for training and validation set for model_0743

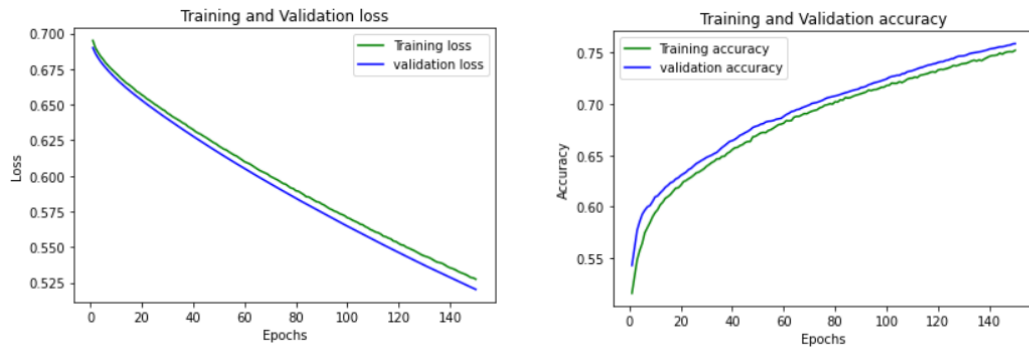


Figure 5-3: Loss and accuracy for training and validation set for model_1130

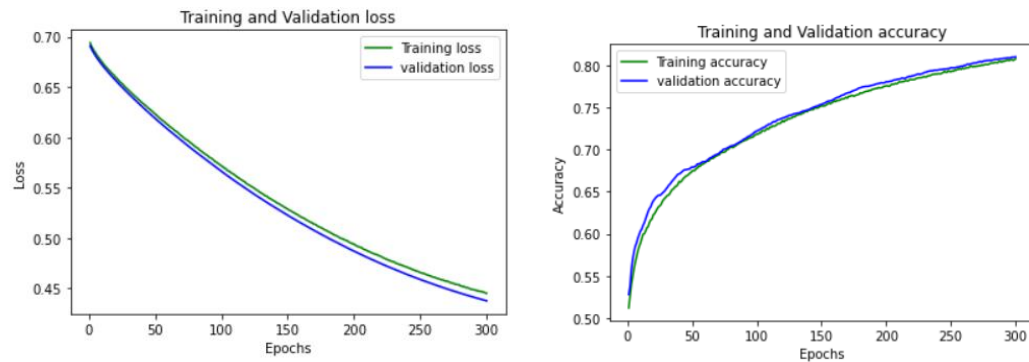


Figure 5-4: Loss and accuracy for training and validation set for model_1130

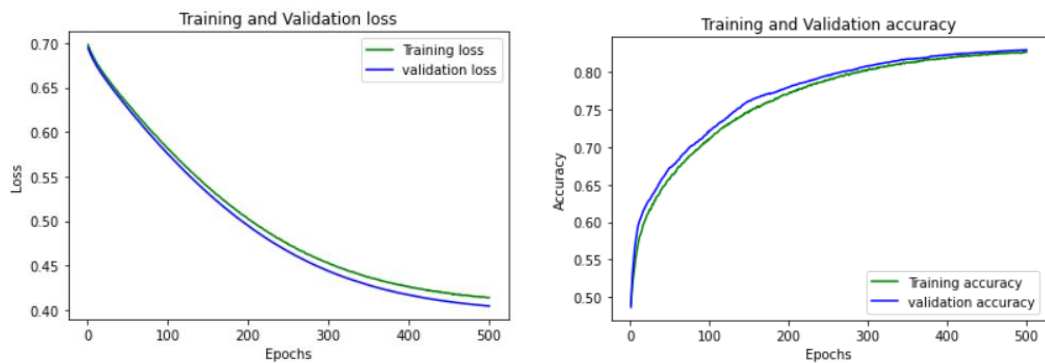


Figure 5-5: Loss and accuracy for training and validation set for model_1607

We can see that except for model_0743 has reached validation loss at around 0.1, the rest of the models have their training and validation loss floating between 0.5 and 0.4. This explains why the model_0743 outperforms over the other three models at predicting the training samples, because in the training process itself, the model already reach a relatively low loss. However, this does not mean the model_0743 is able to generalize well and has the same level of performance for unseen data because there is possibility where the model overfits the training data. In other words, great performance

at predicting the training data is still not supportive enough to prove that the model is the best and most robust among all. Hence, model performance on testing set will be further discussed in the next section.

5-3 Test Data and Its Variants

ID	Type of testing set
1	Football videos only
2	Cooking competition videos only
3	Room escape videos only
4	Football, cooking competition and room escape videos

Table 5-4: Test data specifications

Table 5-4 above shows four types of testing set where the first three types contains only one video category and the last type use all the whole testing set to evaluate the model performance. The following are the model performance on different set of testing data in each model.

	Accuracy	Precision	Recall	F1 Score	Cohen's Kappa	ROC AUC Score
model_0743	0.9120	0.0397	0.0372	0.0384	-0.0077	0.5037
model_1130	0.8340	0.0470	0.1300	0.0690	-0.0005	0.4805
model_1312	0.9002	0.0837	0.1115	0.0956	0.0440	0.6433
model_1607	0.9060	0.0557	0.0619	0.0587	0.0093	0.6002

Table 5-5: Result on testing set ID 1 based on different model

	Accuracy	Precision	Recall	F1 Score	Cohen's Kappa	ROC AUC Score
model_0743	0.7423	0.1368	0.3452	0.1960	0.0755	0.6248
model_1130	0.5087	0.1205	0.6984	0.2055	0.0596	0.5955
model_1312	0.5340	0.1092	0.5757	0.1835	0.0361	0.5599
model_1607	0.5840	0.1348	0.6594	0.2239	0.0858	0.6586

Table 5-6: Result on testing set ID 2 based on different model

	Accuracy	Precision	Recall	F1 Score	Cohen's Kappa	ROC AUC Score
model_0743	0.8907	0.0995	0.0861	0.0923	0.0345	0.6138
model_1130	0.7479	0.1342	0.5330	0.2144	0.1241	0.6694
model_1312	0.8058	0.1664	0.5009	0.2498	0.1693	0.6614
model_1607	0.8068	0.1659	0.4948	0.2485	0.1680	0.6593

Table 5-7: Result on testing set ID 3 based on different model

	Accuracy	Precision	Recall	F1 Score	Cohen's Kappa	ROC AUC Score
model_0743	0.8534	0.1178	0.1757	0.1410	0.0643	0.6163
model_1130	0.6981	0.1202	0.5390	0.1966	0.0952	0.6464
model_1312	0.7485	0.1312	0.4751	0.2056	0.1101	0.6596
model_1607	0.7642	0.1445	0.4964	0.2239	0.1317	0.6690

Table 5-8: Result on testing set ID 4 based on different model

Recall that model_0743 is the model which outperforms the rest during the evaluation using training set. The model performance on different types of testing set are illustrated in the tables above. From the tables, it is noticed that the model_0743 no longer outperforms over the rest. Although it is still having highest accuracy score in the four categories of testing set, accuracy score is not a good metric to be evaluated in this project because this project is dealing with imbalanced dataset. Hence, more focus should be devoted to other performance metrics such as precision, recall and F1 score. In the circumstances of recall score, in most cases, all the models seem to have similar precision and recall score except for model_0743. As seen in Table 5-6, when the other three models are having recall score around 0.55 to 0.70, model_0743 is having the recall score almost two times lesser than the rest of the models. Another example can be seen from Table 5-8 where the rest of the models are having recall score around 0.47 to 0.54, model_0743 is having recall score as low as 0.1757 only. As shown in the tables above, model_0743 almost only rank on top in the comparison of accuracy score and it performs worse than the other in other performance metrics. By summarizing the model performance for model_0743 for training and testing data, it can be concluded that there

is high possibility the model_0743 is overfitting the training data and failing in generalizing well on new data.

Moreover, the models generally have higher recall score than precision, this means all the models are better at predicting the non-suspense class as the difference of recall and precision are mainly contributed from the false negative and false positive output samples. In this case, the models are always having lesser false negative as compared to false positive and this results in the huge difference of recall and precision score.

On the other hand, when comparing the models performance, it is noticed that model_1312 and model 1607 are having greater model performance over the other two models as they are ranked on top for most of the performance metrics evaluation. Nevertheless, recall that this project is building a classification system with only one neural network that intends to identify the suspense and non-suspense class for all types of videos, working well in one type of videos may not be the model this project is looking for. Therefore, model_1607 is selected as the optimal model among the four models as it has the best performance when dealing with three types of videos in this project.

An additional remarks to the model performance is although model_1607 is better than the others, this does not mean the model is performing good and right prediction. As seen in Table 5-8, the model still achieves a relatively low metrics score in precision, F1 score, Cohen's Kappa score and ROC AUC score. With ROC AUC score of 0.6690, the model developed is still not a good classifier.

5-4 Further Remarks

Overall, the model developed is still not a great classification model at predicting the class of the frame. There are few possible reasons resulting in poor performance:

- There maybe insufficient data for the model to learn and generalize well
- There maybe some bad hyperparameters settings selected to train the target model
- There maybe exist inconsistency in labelling the dataset
- The oversampling technique may not work well in this case

Chapter 6 Conclusion

6-1 Project Review

In short, this project is motivated to detect the suspense scene with the purpose of placing advertisement in the suspense scene could better retain the viewers' attention. Due to the lack of suspense related video dataset and suspense detection system using videos as input from the former work, this project presented a novel idea to introduce suspense annotated video dataset and develop suspense classification model using the dataset introduced. The dataset consists of manual binary annotations of suspense and non-suspense scenes from the selected video categories and it intends to serve as a benchmark in suspense detection area of study.

On the other hand, a suspense classification model is developed with the use of LSTM layers, dropout layers and dense layer. In addition, ResNet50 is used as feature extractor to extract the images features and an oversampling method is introduced in this work to oversample the time-series data. Ax that utilizes Bayesian Optimization to optimize problems is used in this project to obtain the optimal hyperparameters settings to the RNN-LSTM network. Last but not least, the trained model is used to perform evaluation on the testing set.

Although the model performance is not well and it is yet ready to be used for real-world suspense scene detection issue. However, it is still too early to give a conclusion that the suspense scene detection task is not doable in deep learning area of study. I believe there exists methods to further improve the model performance but due to the time constraint that this project has, this project has to stop at this stage and I hope the work could contribute to the future suspense scene detection task.

6-2 Future Work

The possible future work to be done to better predict the suspense frame is a video separation pipeline is needed where before a video is fed into the RNN-LSTM network for class prediction, it has to go through another neural network to determine its video type and the output of the neural network will direct the video to its dedicated RNN-LSTM network. In other words, each video category has an isolated network attached to it.

Bibliography

- Burnaev, E., Erofeev, P. and Papanov, A., 2015, December. Influence of resampling on accuracy of imbalanced classification. In Eighth international conference on machine vision (ICMV 2015) (Vol. 9875, p. 987521). International Society for Optics and Photonics.
- Castro, S., Hazarika, D., Pérez-Rosas, V., Zimmermann, R., Mihalcea, R. and Poria, S., 2019. *Towards Multimodal Sarcasm Detection (An _Obviously_ Perfect Paper)*. arXiv preprint arXiv:1906.01815.
- Consumer News and Business Channel CNBC 2020, *Alphabet discloses YouTube ad revenues of \$15.15 billion, cloud revenues of \$8.92 billion for 2019*. Available from: <<https://www.cnbc.com/2020/02/03/alphabet-discloses-youtube-cloud-revenues-for-the-first-time.html>>. [5 September 2020].
- Delatorre, P., Arfe, B., Gervás, P. and Palomo Duarte, M., 2016. *A component-based architecture for suspense modelling*.
- Delatorre, P., León, C., Salguero, A., Palomo-Duarte, M. and Gervás, P., 2018. *Confronting a paradox: a new perspective of the impact of uncertainty in suspense*. *Frontiers in psychology*, 9, p.1392.
- Del Fabro, M. and Böszörményi, L., 2013. *State-of-the-art and future challenges in video scene detection: a survey*. *Multimedia systems*, 19(5), pp.427-454.
- Eckhardt, K. 2018, *Choosing the right Hyperparameters for a simple LSTM using Keras*. Available from: <<https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046>>. [12 April 2021].
- Epstein, D., Chen, B. and Vondrick, C., 2020. *Oops! Predicting Unintentional Action in Video*. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 919-929).
- Heaton Research 2017, *The Number of Hidden Layers*. Available from: <<https://www.heatonresearch.com/2017/06/01/hidden-layers.html>>. [12 April 2021].
- Lambert, N. 2020, *Design Optimization with Ax in Python*. Available from:

- <<https://towardsdatascience.com/design-optimization-with-ax-in-python-957b1fec776f>>. [12 April 2021].
- Malik, U n.d., *Solving Sequence Problems with LSTM in Keras*. Available from: <<https://stackabuse.com/solving-sequence-problems-with-lstm-in-keras/>>. [12 April 2021].
- O'Neill, B. and Riedl, M., 2011, October. *Toward a computational framework of suspense and dramatic arc*. In *International Conference on Affective Computing and Intelligent Interaction* (pp. 246-255). Springer, Berlin, Heidelberg.
- Phi, M., 2018. *Illustrated Guide to LSTM's and GRU's: A step by step explanation*. Available from: <<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>>. [12 April 2021].
- Rasheed, Z. and Shah, M., 2003, June. *Scene detection in Hollywood movies and TV shows*. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. (Vol. 2, pp. II-343)*. IEEE.
- Schedi, M., Sjöberg, M., Mironică, I., Ionescu, B., Quang, V.L., Jiang, Y.G. and Demarty, C.H., 2015, June. *Vsd2014: a dataset for violent scenes detection in hollywood movies and web videos*. In *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)* (pp. 1-6). IEEE.
- The Python Package Index 2020, *smote-variants 0.4.0*. Available from: <<https://pypi.org/project/smote-variants/#and-smote>>. [12 April 2021].
- Wilmot, D. and Keller, F., 2020. *Modelling Suspense in Short Stories as Uncertainty Reduction over Neural Representation*. arXiv preprint arXiv:2004.14905.
- Yeung, M., Yeo, B.L. and Liu, B., 1998. *Segmentation of video by clustering and graph analysis*. *Computer vision and image understanding*, 71(1), pp.94-109.
- Yuan, Y., 2018. *Framing surprise, suspense, and curiosity: a cognitive approach to the emotional effects of narrative*. *Neohelicon*, 45(2), pp.517-531.
- Zhai, Y., Rasheed, Z. and Shah, M., 2005. *Semantic classification of movie scenes using finite state machines*. *IEE Proceedings-Vision, Image and Signal Processing*, 152(6), pp.896-901.

Bibliography

- Zhu, S. and Liu, Y., 2009. *Two-dimensional entropy model for video shot partitioning*. Science in China Series F: Information Sciences, 52(2), pp.183-194.
- Zhu, S., Yan, J. and Liu, Y., 2009, September. *Improving semantic scene categorization by exploiting audio-visual features*. In 2009 Fifth International Conference on Image and Graphics (pp. 435-440). IEEE.
- Zhu, T., Lin, Y. and Liu, Y., 2020. Oversampling for Imbalanced Time Series Data. arXiv preprint arXiv:2004.06373.

Appendix A: Final Year Project Source Code

GitHub Link: <https://github.com/sinhui99/FinalYearProject-sinhui>

README.md



FinalYearProject-sinhui

This is a guide on how to build the system step by step. You can either use ipynb file or py file provided here.

For those who intend to only test the model with testing set and saved model, kindly skip to instruction 11.

If you have downloaded any pickle from GitHub or Drive, kindly put them in a folder named 'pickle' before running the program.

1. Download the youtube-dl package. For Windows user, please go to folder ForWindowsUser. For Mac user, please go to folder ForMacUser.
2. The package includes the videos download script, named download.bat (Windows) or download.sh (Mac).
3. For Windows user, kindly download the videos to local by clicking the download.bat. For Mac user, kindly open up a terminal and type the following:

```
bash download.sh
```

If the permission is denied, then do the following:

```
chmod 777 download.sh
```

4. Split the videos into training and testing set.

Name training videos folder to videos

```
videos/jdezKogGKW8.mp4
videos/1jVO--nexcA.mkv
videos/Lb0-qvy9J1Y.mp4
videos/1p59mYfvKITV.mp4
videos/p8fqQM126ET.mp4
videos/smBYcpMveuo.mp4
videos/wbNBQW8fWA.mp4
videos/weUenGaF8cs.mp4
videos/YXUml.753tYo.mp4
videos/Z5i6jdYkHcA.mp4
videos/3fYpcapas8k.mp4
videos/5dCpEzu1ka8.mkv
videos/50JfbYQtKtk.mp4
videos/5V8LsqPzDTU.mp4
videos/7Fau-IwbuJc.mp4
videos/7jpruuvWK7E.mp4
videos/9ifbXe4TsSc.mp4
videos/eFDPJ1vsrU8.mp4
videos/ep0FVRUzweg.mp4
videos/HFUwxxpzHrs.mp4
```

Name testing videos folder to testvideos

```
testvideos/Xhu5Bz1xDf0.mp4
testvideos/XLkx0_I1qLQ.mp4
testvideos/_y3rFsvz8qQ.mkv
testvideos/3eDBuzDgLP0.mp4
testvideos/46xKDn60Fhw.mp4
testvideos/CFwSgo-ftrQ.mp4
testvideos/HktqGcJQA4w.mp4
testvideos/ms4h00h1d0g.mp4
testvideos/xBUu1q-1KSI.mkv
```

Place the videos to their corresponding folder.

5. Download the following:

```
trainlist01.txt  
testlist01.txt  
items in textfiles folder  
items in testfiles folder
```

Note that if the videos extension have changed upon downloading, you have to modify the trainlist01.txt or testlist01.txt accordingly.

6. Install the libraries required by executing the following commands:

```
conda install python=3.8.5  
pip install opencv-python==4.5.1.48  
pip install ax-platform==0.1.20  
pip install tqdm==4.50.2  
conda install -c anaconda numpy  
conda install -c anaconda tensorflow-gpu  
conda install -c anaconda cudatoolkit  
conda install -c anaconda h5py  
conda install pillow
```

7. To rebuild the system, run the following files: (please get data_preprocessing file from the folder ForMacUser or ForWindowsUser)

```
data_preprocessing.ipynb or data_preprocessing.py  
feature_extraction.ipynb or feature_extraction.py
```

This may take few hours to process, if you wish to save the time, then skip running the files and do the following:

Download the pickle folder containing:

```
each_video_extra_frame.pickle  
each_video_frame.pickle  
n.pickle  
video_rnn.pickle  
X.pickle  
y.pickle
```

Note: For X.pickle, the file size has exceeded 25 MB, kindly get at Drive. Drive link:
<https://drive.google.com/drive/folders/1LyCji2UXEcTU0RtSFMSIhBtflXtMnzJ?usp=sharing>

8. Next, to transform feature into RNN shape, run

```
transformation_to_rnn.ipynb or transformation_to_rnn.py
```

9. Run the Ax experiment.

```
ax_experiment.ipynb or ax_experiment.py
```

You can skip this step if you want to use the optimal hyperparameters available in the pickle folder to train the model. Go pickle folder to download the pickle file.

```
best_hyperparameters.pickle
```

10. Train the model by running:

```
training.ipynb or training.py
```

Upon complete running the file, the trained model will be saved. You may need to modify the name of the model and the name of the h5py file if needed.

11. Run the following to preprocess the testing data:

```
test_preprocessing.ipynb or test_preprocessing.py
```

If you wish to skip processing the testing data, you can download the pickle files from drive. Drive link: <https://drive.google.com/drive/folders/1LyCji2UXEcTU0RtSFMSIhBtifiXtMnzJ?usp=sharing>

```
X_testing.pickle  
y_testing.pickle
```

12. Download the saved model from Drive because the saved model has exceeded 25 MB. You need not to do this step if you are building from scratch. Drive link:

<https://drive.google.com/drive/folders/1LyCji2UXEcTU0RtSFMSIhBtifiXtMnzJ?usp=sharing>

You can choose to use the saved model in folder or the h5py file to predict the class on testing set. By default, it is choosing the h5py file, kindly change it if needed.

13. Load and test the model by running:

```
testing.ipynb or testing.py
```

The default model to be loaded is model_1607, if you wish to test on different models, kindly change to the corresponding model name.

Note 1: ax_client_snapshot contains details on the 250 trials using Ax experiment

Note 2: train_new.csv and test_new.csv are just for your reference, it will be generated if you run the system step by step and if you are not, you do not need the files.

For any further inquiries or issue faced while executing the files, kindly contact the following email: sinhui1999@1utar.my

Drive link: <https://drive.google.com/drive/folders/1LyCji2UXEcTU0RtSFMSIhBtifiXtMnzJ?usp=sharing>

Appendix B: Poster

Suspense Scene Detection Using Recurrent Neural Network

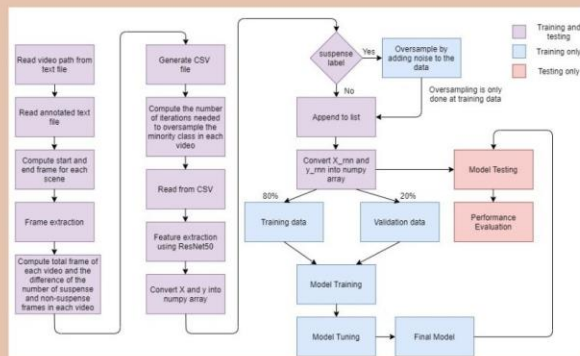
Introduction

This project claims a suspense moment is a better ads placement position which could retain viewer's attention to the video. Hence, the project aims to develop a suspense binary detection system using LSTM layers.

Objectives:

- To introduce a suspense labelled dataset
- To build a suspense binary detection system
- To introduce an oversampling technique used for time series data

Methodology:



Result:

Training Set:

Accuracy: 0.8064

Precision: 0.2433

Recall: 0.8017

Testing Set:

Accuracy: 0.7642

Precision: 0.1445

Recall: 0.4964

Discussions:

Recall score is generally performs better than precision score. The model is better at predicting non-suspense class and it had wrongly classified many samples of non-suspense class to be suspense.

Conclusion:

Although the model is not generalizing well, this does not mean suspense scene detection is not doable, the use of video separation pipeline may help to better detect the suspense scene in different types of videos

Appendix C: Final Year Project Biweekly Report

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 2
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Writing algorithm for data pre-processing. (Until frame extraction)

2. WORK TO BE DONE

- i. Extract features.
- ii. Transform samples to RNN shape.
- iii. Model building.

3. PROBLEMS ENCOUNTERED

N/A

4. SELF EVALUATION OF THE PROGRESS

- i. Progress is slightly slow as more time need to be devoted for training and tuning.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 4
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Done writing code for features extraction and RNN shape transformation.

2. WORK TO BE DONE

- i. Model building.
- ii. Model training and testing.

3. PROBLEMS ENCOUNTERED

- i. Spent some time in thinking the algorithm to transform the features into RNN shape (num_of_samples, seq_len, num_of_features) by videos.

4. SELF EVALUATION OF THE PROGRESS

- i. Work is on track.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 6
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Done building the model architecture.
- ii. Done training on small set of data.

2. WORK TO BE DONE

- i. Solving class imbalance issue.
- ii. Model tuning and testing.
- iii. Finding some rules of thumb in building the model architecture.

3. PROBLEMS ENCOUNTERED

- i. Class imbalanced in the dataset causing the model fails to learn on the minority class.
- ii. Dataset is too large to train the model with complete data.

4. SELF EVALUATION OF THE PROGRESS

- i. Work is on track but discovered issues are pending to solve.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 8
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Research on techniques to overcome class imbalanced issue.
- ii. Research on some rules to follow when building lstm layers

2. WORK TO BE DONE

- i. Implementing technique to solve class imbalanced issue.
- ii. Retrain and test the model.
- iii. Model tuning.
- iv. Reconstruct and writing report.

3. PROBLEMS ENCOUNTERED

- i. Fail to find technique to solve class imbalanced issue for time-series data.

4. SELF EVALUATION OF THE PROGRESS

- i. Progress is delayed due to workloads from other courses.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 10
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Review and modify fyp1 report.
- ii. Using noise to oversample data from minority class.
- iii. Model training.

2. WORK TO BE DONE

- i. Writing report.
- ii. Hyper-parameter tuning.

3. PROBLEMS ENCOUNTERED

- i. Model overfitting.
- ii. Hardware specification limits the training data can be trained.

4. SELF EVALUATION OF THE PROGRESS

- i. Work is on track but model performance is not good.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project II)

Trimester, Year: Y3S3	Study week no.: 12
Student Name & ID: Lim Sin Hui & 17ACB04528	
Supervisor: Dr. Aun Yichiet	
Project Title: Suspense Scene Detection Using Recurrent Neural Network	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- i. Continue writing on report.
- ii. Implementing code for hyper-parameter tuning.
- iii. Model training and testing.

2. WORK TO BE DONE

- i. Complete fyp2 report.
- ii. Create GitHub repository and upload related prototype.
- iii. Code rearrangement.

3. PROBLEMS ENCOUNTERED

- i. Overfitting issue still exists.

4. SELF EVALUATION OF THE PROGRESS

- i. Work is on track.



Supervisor's signature



Student's signature

Appendix D: Plagiarism Check Result

Feedback Studio - Google Chrome
 ev.turnitin.com/app/carta/en_us/?u=1111742742&ts=1&student_user=1&lang=en_us&o=1560514892

feedback studio | Lim Sin Hui | Suspense Scene Detection Using Recurrent Neural Network | -- /0 ?

ACKNOWLEDGEMENTS

³⁴ I would like to express my special thanks of gratitude to my final year project supervisor, Dr. Aun Yichiet who offered me an opportunity to work on this project. In addition, Dr. Aun has been helpful and supportive along the project by providing useful guidance and accessing to a GPU machine to reduce the training time required.

³⁹ Secondly, I would like to thank to my family and friends who always being mentally supportive when I was facing difficulties in the project. Without their love and tolerance, I definitely could not have completed my project.

Match Overview

5%

1	ijtte.com Internet Source	<1%
2	www.amrita.edu Internet Source	<1%
3	ionut.mironica.ro Internet Source	<1%
4	"Advanced Machine Le... Publication	<1%
5	Dipanjana Moitra, Rakes... Publication	<1%
6	Yiming Chen, Lu Yu, Ya... Publication	<1%
7	dokumen.pub Internet Source	<1%
8	towardsdatascience.co... Internet Source	<1%
9	Submitted to Cambridg... Student Paper	<1%

Page: 1 of 53 | Word Count: 13247 | Text-only Report | High Resolution On

Turnitin Originality Report

Processed on: 16-Apr-2021 09:53 +08

ID: 1560514892

Word Count: 13247

Submitted: 1

Suspense Scene Detection Using Recurrent Neur... By Lim Sin Hui

Similarity Index	Similarity by Source
5%	Internet Sources: 3%
	Publications: 4%
	Student Papers: 1%

include quoted include bibliography excluding matches < 8 words	mode: <input type="text" value="quickview (classic) report"/> Change mode print download
<1% match (Internet from 20-Apr-2019) http://ijtte.com	
<1% match (Internet from 25-Feb-2021) https://www.amrita.edu/faculty/kp-soman	
<1% match (Internet from 24-Oct-2019) http://ionut.mironica.ro	
<1% match (publications) "Advanced Machine Learning Technologies and Applications", Springer Science and Business Media LLC, 2021	
<1% match (publications) Dipanjan Moitra, Rakesh Kumar Mandal. "Prediction of Non-small Cell Lung Cancer Histology by a Deep Ensemble of Convolutional and Bidirectional Recurrent Neural Network", Journal of Digital Imaging, 2020	
<1% match (Internet from 19-Feb-2021) https://dokumen.pub/python-machine-learning-machine-learning-and-deep-learning-with-python-scikit-learn-and-tensorflow-2-3nbsped-1789955750-978-1789955750.html	
<1% match (publications) Yiming Chen, Lu Yu, Yanyan Yao, Lei Zhu. "Individual Identification Technology of Communication Radiation Sources Based on Deep Learning", 2020 IEEE 20th International Conference on Communication Technology (ICCT), 2020	
<1% match (Internet from 16-Nov-2020) https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21?gi=1fac09cb4216	
<1% match (student papers from 12-May-2019) Submitted to Cambridge Education Group on 2019-05-12	

Appendix E: Form iad-FM-IAD-005

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Lim Sin Hui
ID Number(s)	17ACB04528
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	Suspense Scene Detection Using Recurrent Neural Network

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u>5</u> % Similarity by source Internet Sources: <u>3</u> % Publications: <u>4</u> % Student Papers: <u>1</u> %	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor

Name: Aun Yichiet

Date: 16/04/2021

Signature of Co-Supervisor

Name: _____

Date: _____

Appendix F: FYP 2 checklist



UNIVERSITI TUNKU ABDUL RAHMAN



**FACULTY OF INFORMATION & COMMUNICATION
TECHNOLOGY (KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB04528
Student Name	Lim Sin Hui
Supervisor Name	Dr. Aun Yichiet

TICK (✓)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
/	Front Cover
/	Signed Report Status Declaration Form
/	Title Page
/	Signed form of the Declaration of Originality
/	Acknowledgement
/	Abstract
/	Table of Contents
/	List of Figures (if applicable)
/	List of Tables (if applicable)
/	List of Symbols (if applicable)
/	List of Abbreviations (if applicable)
/	Chapters / Content
/	Bibliography (or References)
/	All references in bibliography are cited in the thesis, especially in the chapter of literature review
/	Appendices (if applicable)
/	Poster
/	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <p></p> <p>_____ (Signature of Student) Date: 16/04/2021</p>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <p></p> <p>_____ (Signature of Supervisor) Date: 16/04/2021</p>
---	--