SUMMARIZATION OF LARGE-SCALE VIDEOS TO TEXT FORMAT USING SUPERVISED BASED SIMPLE RULE-BASED MACHINE LEARNING MODELS

U.K.H.A. Sugathadasa

199488F

Department of Computational Mathematics,

Faculty of Information Technology

University of Moratuwa

Sri Lanka

March 2022

SUMMARIZATION OF LARGE-SCALE VIDEOS TO TEXT FORMAT USING SUPERVISED BASED SIMPLE RULE-BASED MACHINE LEARNING MODELS

Udage Kankanange Harindu Ashan Sugathadasa

199488F

Thesis/Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Artificial Intelligence

Department of Computational Mathematics

University of Moratuwa

Sri Lanka

March 2022

Declaration

I declare that this is my own work, and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Name of the Student:	Signature:	Date:
Sugathadasa U.K.H.A.	……	…….

The above candidate has carried out research for the Masters/MPhil/PhD thesis/ dissertation under my supervision.

Name of the Supervisor:	Signature of the supervisor:	Date:
Dr. Subha Fernando	……	…….

Name of the Co-Supervisor:	Signature of the co-supervisor:	Date:
Dr. Varuna De Silva	……	…….

Abstract

Video Summarization has been one of the most interested research and development field since the late 2000s, thanks to the evolution of social media and the internet, due to the influence to provide a concise and meaningful summary of large-scale video. Even though the video summarization has been elongated through several non-ML and traditional based techniques and ML-based techniques, generation of correct and required summaries from the video is yet a limitation. To overcome this concern, different techniques have been attempted including vision-based approaches and NLP related approaches. With the inspiration of NLP related Transformer networks, researchers are looking to integrate such sequence-based learning algorithm into the video dimension as to apply spatiotemporal extractions. Despite the VS implementations, another extension of VS has been exponentially emphasized, namely TVS which generates the summaries of the video via a text format.

Simply the evolution of VS towards TVS is not a straightforward journey since a lot of blockers have been eliminated using UL, RL, and SL based frameworks. When it comes to the STOA methods in TVS, Transformer based methods are eventually highlighted along the T5 based NLP frameworks. Since this area is still at the ground level, a lot of unknow facts and issues can be explored. Especially the attention-based sequence modelling of the learning algorithm should be carefully imitated to achieve the best accuracy improvements. All the improvements are subjected to apply into a real-time application ulteriorly. To tackle such improvements, a novel standalone method should be introduced with the simplest network layout which can be applicable to the embedded devices. This is where the **Si**mple **Ru**le-based **M**achine **L**earning Network to **T**ext-based **V**ideo **S**ummarization (SiRuML-TVS) has been unveiled.

Though the network contains a single input of large-scale video and a single output of meaningful description for the given video, the high-level network layout compromises three ML modules for Video Recognition, Object Detection, and finally Text Generation. Each module is subjected to different evaluation criterions however, the end-to-end full network is evaluated on a single metric. Different combination of each module can be affected to the performance of the entire pipeline however, the

combination of Transformers and CNNs provide the better tradeoff between accuracy and the computational inferencing. This makes a hope to deploy the proposed method in an edged device thus, the gap between theoretical explanation to practical implementation will be filled.

Dedication

I dedicate this thesis to my parents, my sister, my wife, the university lecturers from undergraduate level who are always withstand in my successes and failures.

Acknowledgements

I would like to pay my greatest appreciation to Dr. Subha Fernando, who motivated my enthusiasm on research and provided guidance and advises through the research journey till the end. Her experience on research was tremendously big support to this work, and I be indebted a great deal of its success for herself.

Every person has its own shadow when at be any place. My second appreciation is to Dr. Varuna De Silva, who is from University of Loughborough, for supporting both me and supervisor when we had a doubtful circumstances and unsolvable stakes.

Next, I would like to elapse my thanks and appreciation to the staff members of the Department of Computation Mathematics, for their supportiveness and consideration for my research work.

Lastly, I would also like to express appreciation through my bottom of heart for all my Lecturers from the Undergraduate program, my fellow colleagues from the University, and all my family members, who supported me through the journey of research program in both verbally and mentally. Even though they might not know about this, the impact from them is substantial, and I might not make it to where I am now at without themselves.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Definition
VS	Video Summarization
TVS	Text-based Video Summarization
SL	Supervised Learning
UL	Unsupervised Learning
RL	Reinforcement Learning
WSL	Weakly-Supervised Learning
GAN	Generative Adversarial Network
LSTM	Long-Short Term Memory
NLP	Natural Language Processing
BiLSTM	Bi-Directional Long-Short Term Memory
CNN	Convolutional Neural Network
CSN	Channel Separated Network

Chapter 1

INTRODUCTION

1.1 Prolegomena

“The greatest asset of what you can have been not only the information but how much relevant information also you do have in your hand within a considerable time”. From this slogan, the importance of necessary information at right time has been highlighted eventually.

Since the beginning of Deep Learning related research in multimedia scope, including vision and language processing, the advancements of the field is expanded into various sub filed such as classification, recognition, summarization, and so on. The revolution of video related Machine Learning has begun the next active research are in multimedia scope. Due to the high capacity of videos been produced in every day, summarization of videos take place into number one for reduce the space required to storage and extract only the necessary information. For this purpose, there are a lot of room available for improvements.

From the evolution of the existing Deep Learning related techniques and its performances, the traditional Machine Learning has been taken out from the scope of research end. However, due to the explanation capacity of such algorithms, research community are also looking for the combination of both simplicity and performance orientation to leverage the best solutions in Video Summarization.

1.2 Background and Motivation

An active research community for multimedia related research can be seen through worldwide including China, United States of America, Singapore, Europe countries, and so on. Since late of 2000s’, summarizing the videos has been emerged due to the evolution of various contents through social media platforms and internet. Fortunately, most of the vision based traditional machine learning related tasks are converted to Convolutional Neural Networks (CNNs) and Long-Short Term Memory (LSTM) Networks from late 2015 onwards due to the exploration of these algorithms [1], [2].

Video Summarization is the fundamental baseline for the most of summarization algorithms and the heavy load of Unsupervised Learning [1], [3], [4] and Reinforcement Learning [3], [5]–[8] environments are used due to the lack of datasets involved in this area. Since the datasets are being created, the tradition of using Unsupervised Learning and Reinforcement Learning then populated into Supervised Learning [8]–[14]. With this setting, language related machine learning algorithms are also started to integrate to improve the both accuracy and inference speed [15]–[17] in the model architectural level.

The advance concept of Video Summarization is the Text-based Video Summarization that opens the vast number of real-world applications such as Video Synopsis, Controlling Algorithms to Automated Vehicles, Movie Abstraction, Guidance for Visually impaired people, and so on. Most of the Text-based Video Summarization research works cover the combination of vison and language processing tasks [7], [18], [19]. Therefore, the potential of research in Text-based Video Summarization is countless.

1.3 Aims and Objectives

1.3.1 Aim

The aim of this research project is to develop a simple and real-time text-based video summarization system for the large-scale video inputs **using simple Rule-based and Machine Learning models with Supervised Learning manner**.

1.3.2 Objectives

- To critically review the existing approaches for video summarization and then the text-based video summarization using deep learning models in supervised learning manner.
- To analyze the simple machine learning models in supervised learning environment and identify the possible extensions for text-based video summarization.

- To design and develop a system combined supervised learning-based machine learning algorithms for Text-based Video Summarization in to overcome the limitations in existing approaches.
- To evaluate the developed solution using the appropriate evaluation metrics and compare it with the existing methods and algorithms.
- To perform the system in vehicle dashcam footages and publishing the results.

1.4 Problem Definition

Currently, there are several research works for Video Summarization via Textual form. However, the implementation in the real-world application is a problematic subject of this area. Due to the high frame rate occupied in the video stream, a little inferencing time is important with the sufficient accuracy level. When it comes in the explanation level, it is hard to provide the in-depth details about the model.

Despite these issues, dataset that is going to be used for the application of interest is another issue since the dataset subjectiveness matters broadly. Most of the research works are highly subjectiveness to the dataset that is used at the training and evaluation phase and data independent algorithms in this field can be found rarely in SL frameworks. Lack of facilitation to change an algorithm to capture the information is another problematic area. If the application is moved from one domain to other, the workload that is required to amend it is extremely expensive.

The lattermost developments are yet to be addressed of these issues apparently. Therefore, a simple algorithm but achieve the same level of performance should be required for TVS model.

1.5 Proposed Solution

TimeSformer, shorthand word for Time-Space Transformer, by Bertasius et al. [20] has been introduces as the complimentary solution for CNN networks which is to cover the loss of temporal information and capture the long-range spatial information simultaneously. For Text Generation, Text-To-Text Transfer Transformer is introduced by the research team in Google, Raffel et al. [21].

In this research work, we propose a simple Supervised Machine Learning based algorithms with an appropriate Rule-based mechanism to provide the Text-based Video Summarization. To cover the proposed algorithm, we choose Vehicle Dashcam Videos as the large-scale videos which contain vehicle related actions mostly, and a very few human actions are involved, to provide an automated algorithms for controlling purpose in the Automated Vehicle System.

1.6 Resource Requirements

For the lucrative completion of the research project, as per the inherent system is based on Machine Learning algorithms, it is required to use Graphical Processing Unit (GPU) based systems for covering the model training and model inferencing.

To train the models, the high-quality datasets for intended application area are necessarily required. Both video based and NLP description-based datasets should be taken into the research work. At the end of the research, the proposed system should be deployed on a cloud-based solution since to cover portability and accessibility of the application from the point of view by the users.

1.7 Outline of the Thesis

The rest of the thesis is outlined as several chapters. Chapter 2 is highly converged to reviewing the existing research work in the domain of Video Summarization and its sub variants such as Text-based Video Summarization with both benefits and challenges of each method and highlighting the problem definition. Through Chapter 3, the core technology that is used by the research project has been described in depth. Following that, in Chapter 4, namely Approach, the high-level overview of the proposed solution can be seen in the Milestone manner. Chapter 5 is used to discuss the Design of the proposed solution in an atomic level. In the Implementation Chapter, noted as Chapter 6, the end-to-end details of the design will be covered with both high level and low-level information. A chapter before final, says Chapter 7 as Evaluation, the detailed evaluation process for the proposed solution will be covered in both qualitative and quantitative levels. As per the final chapter, Chapter 8, the conclusion

where the discussion of work carried through the project and further developments of the research work is stressed.

1.8 Summary

In this chapter, a brief but understandable opening to the research work has been carried out. It covers the background motivation of the research work, the aims, and objectives of the research with a definition for problems that are encountered, and the brief about proposed solution and its usage as an application. The required resource for the research is also highlighted in this chapter. As per the final section of this chapter, the structure of the thesis has been given briefly to get an idea about the thesis.

Chapter 2

SUMMARIZATION OF LARGE-SCALE VIDEOS TO TEXT FORMAT – DEVELOPMENTS AND ISSUES

2.1 Introduction

In the previous chapter, an introduction to the entire research project is given covering the importance of Text-based Video Summarization in Large-Scale Videos. Through this chapter, we present the detailed critical review of research and developments in Videos Summarization techniques in the sections of Supervised Learning, Unsupervised Learning, and Reinforcement Learning. This chapter is structured under several sub sections called early developments in Video Summarization (VS), breakthroughs and trends in VS, Challenges in Text based VS, the taxonomy of existing VS techniques and our coverage area out of that, and finally the problem definition.

2.2 Early Developments (Gestation) in Video Summarization

Multimedia has been recognized since the late of 2000s' due to the evolution of social media platforms, commercial advertisements, international movie industry, and the internet (a.k.a. World Wide Web). Most of the attention is preserved to the videos among other sources of information since the variety of the videos are much larger than rest of the sources, and the level of understanding is easier with respect to the other sources. Ever since the videos are popular than other sources, the interest on information in videos are getting attractive and therefore, people only want to get the most interested details through the video. This is where the engaging stage of Video Summarization (VS) in research. It was not easy to get a bibliography on newly identified research area since the lack of understanding about the area. This challenge is accepted by Barbieri et al. [2] and identify the classification of relevant areas of summarization processes, namely the characteristics of the summarization, visual content in the videos, and scenarios of targeted scopes. This is the first broad classification which was evaluated in Video Summarization as we can identify in the

history. Through the journey, a lot of research areas are involved to improve the VS as per the current standard. These improvements are not limited to a few areas, but a vast number of areas affected along the technological improvements. When it discusses with technical areas, more than a decade from 2000 has been taken with non-Machine Learning techniques which are rule-based systems and mostly with traditional statistical-based approaches in computer vision. These traditional methods are not applicable for large-scale videos which are defined in terms of both image scale and time length. Since the Convolution based Neural Networks [22] are introduced into the Machine Learning Area late 2021, the revolution has begun in VS for large-scale videos dramatically. Due to this evolution, a larger bibliography has been identified by Apostolidis et al. [1] through a survey, in the areas of Supervised Learning (SL), Unsupervised Learning (UL), and mostly with Reinforcement Learning (RL). They have generated a better taxonomy related to types of VS and technologies used in VS such as RL based Long-Short Term Memory (LSTM) [23], Generative Adversarial Network (GAN) [24] based SL approaches, and UL based solutions. According to the survey, there is a gap in very large-scale videos such as movies and surveillance videos, and it is opened to the research community.

For movies related large scale VS, Yong Rui et al. [25] attempt firstly to provide a remedy which is based on traditional ML like manual feature extraction, identification of key shots and key frames, and tied up for providing the most important scenes. However, when it comes into an application level, the feasibility of deploying this solution cannot be done. Even though, at that time, the relationship between space and time is not understood by the researchers. With the introduction of LSTM network by Hochreiter and Schmidhuber [23], the gap of such spatial-temporal, also known as spatiotemporal, has been neutralized up to an expected level. With this introduction, K.Zhang et al. [26] shows the resolution for both movies and surveillance footage related large-scale VS in SL based environment. Unfortunately, their solution is mostly affected with data subjectivity which is the model can only understand the information that are seen only from the dataset. Therefore, the resolution is not viable for an application level. To provide a remedy, a lot of research works are based upon UL and RL hereafter. The first successful UL plus RL based VS has been exhibited by Zhou

et al. [3] with the implementation of LSTM network. This approach has won the recognition to deviate dataset subjectivity totally. Even though the data subjectivity is totally nullified from the previous VS model, some researchers have claimed that at least the minimum data subjectivity should involve into the model training to make a good video synopsis. To blend some data subjectivity into the model, Y. Zhang et al. [6] uses query-based VS approach which can do users' interest based VS. To claim this model, they have harvested the previous approach with a replacement on UL network to SL network. However, both SL and UL based approaches are showing the same competition score despite the data subjectivity issue, researchers are then interesting to investigate the multimodal based VS approaches to test the hypothesis of multi-input-based SL models can perform greatly. Along a set of research works based on multi-input-based VS models, H. Li et al. [19] depicts a better algorithm which uses text, images, audio inputs, and videos indeed in an asynchronous mode. The relationship between these inputs is not required. The key benefit of this approach is that it can learn better summarization features. Even though the multi features are extracted from this approach, it cannot surpass the current best models due to convergence issue of the model. This is where the multimodal inputs are throwing away to build VS models. Instead of multimodal inputs, the recommendation content based approaches have come into the VS field with the successful model implementation done by Lee and Abu-El-Haija [27]. According to the results, this approach claims the better accuracy along with the implementation easiness. Only the required input is just the feedbacks for video contents, and it will output the correct video segment for the recommendation.

From the recommendation system-based VS, a counter hypothesis has been published which is that a text-based description can be granted for the given video. In simply, we can make a text description for a video segment. This idea has created a large sub variant in VS called Text-based Video Summarization (TVS). Through the research works covered in VS area, a successful research for TVS has been done by Park and Kim [28]. They have focused to provide a description for an image and then extend the same approach to the video content. To accommodate this algorithm, they use Natural Language Processing (NLP) based techniques along the vision based neural

networks. Due to the lack of spatiotemporal relationship in videos, this method is not popularized effectively. Though the LSTM models are used in VS, TVS models are not showing the best results based on LSTM networks. As a resolution, both Wang et al. [7] and Huang et al. [8] introduce a better approach to TVS based on Hierarchical Reinforcement Learning (HRL) method. HRL by Kulkarni et al. [29], is a replacement to LSTM networks which uses to integrate temporal abstraction and intrinsic behavior in space dimension by using the manager-employee relationship in the context of neural networks. The concept of manager-employee is that the manager provides the set of high-level tasks for employees to complete and report the status. This mechanism is much easier to control for improving accuracy. Similarly, the same concept has been used on the Generative Adversarial Networks (GAN) which is a generator tries to return the correct output while the discriminator tries to fool the generator. With the same insight of using GAN based networks, K. Zhang et al. [11] depicts the GAN based Semi-Supervised Learning (SSL) environment and Li and Yang [30] discuss a Weakly-Supervised Learning (WSL) based RL method for TVS. These are the early developments starting from basic VS towards the advance TVS methods. Based on these developments, there are so many breakthroughs and trends for VS and TVS.

2.3 Breakthrough and Trends in Video Summarization – Latest

According to the early developments in VS and TVS areas, starting from traditional non-ML techniques to Deep Learning (DL) techniques, research community has tried several approaches in UL environment at first. Then the developments are majorly focused on RL environment. Fortunately, the revolution of dataset creation with less data subjectivity has actively supported to evolve more SL based environment developments in both VS and TVS. It is worth to take a closure overview of each environment related breakthroughs and trends.

2.3.1 Unsupervised Learning based Video Summarization

As per the terminology Unsupervised Learning (UL), the intention of using such environment is to train model with dataset which does not have annotated/labelled data samples. The sole purpose of being used UL environment is to train a model with less

data subjectivity concern and to learn itself as much as possible patterns/features without guidance. For an example, a model separates three clusters based on patterns of apples, oranges, and pineapples. There are several collaborations with UL environment that can be visualized structurally, and all these combinations are trying to alleviate the data subjectivity issue ulteriorly.

The first breakthrough of UL based environment in VS is the combination of UL plus Deep RL (DRL) in large-scale VS by Zhou et al. [3], to browse short and concise summary video contents that can screen the original content with diversification. To accommodate this approach, they have used a reward function in RL framework that can easily diversify the original content and ability to represent the diversified contents. Though this is a simple technique to amend reward function in RL framework, there is a hidden issue of allocating more rewards to learn unnecessary patterns/features from the non-labelled dataset. This is where a replacement of RL framework with GAN based approach by He et al. [31] which called UL based Attentive Conditional Generative Adversarial Network. Like the overview of GAN network, the generator tries to produce high quality frames while the discriminator tries to distinguish the outputs of the generator. This trade-off will make a quality output with conditional attentive mechanism which supposes to create sensitive and important temporal regions out of the large-scale video. With conditional attentive method, they have also used a frame-level multi-head attention technique to cover more alternatives thus to engage the best summaries of the large-scale video. Eventually, use of GAN approach will digest more computational resources as well as less sensitive to deploy in an application level. The same issues along with the accuracy can be seen in the previous method.

Rather than looking at each pattern/feature which are interested, it is worth to identify relation-aware context of the video which are the most importance scenes embedded into the video. This is where Gao et al. [32] use to prove that relation-aware method would better to exploit clip-to-clip relations for selection of key scenes/shots of the input video. The main advantage of this approach is much faster than the previous UL based method. However, when it comes to the learning algorithm, it has a high risk of learning the importance score as a flat distribution rather than some gaussian based

distributions. As per the remedy of this issue, Jung et al. [33] depict a mechanism that can eliminate an ineffective feature learning algorithm on large-scale videos. Their intelligent hypothesis is to use a two-stream network that compromise with local feature extractions and the global feature extractions. Chunk network path for extract local features, and Stride network path for extract global features will accumulate a great importance score thus the distribution curves are not getting to flat scale.

With all the previous combinations, Liang et al. [4] propose a method that covers UL based GAN network with attention-oriented Convolutional Sequence Network (CSN). They have intentionally changed the generator architecture with attention-oriented CSN algorithm which uses to extract global representative features in an input video along the frame level features and shot-level features. The corresponding objective function is made by combination of three loss functions corresponding to each feature calculation. As per the current overview of the literature review, this is the latest trend under UL based environment with state-of-the-art (STOA) results.

2.3.2 Reinforcement Learning based Video Summarization

In the UL based research works, all the researchers were trying to resolve the data subjectivity concern ulteriorly. Their main concern is to improve accuracy itself despite the deployment in application level. Due to this cause, most of the researchers are looking to integrate RL techniques with the assumption that the improvements can be intuitively done through assigning the rewards to the network. When it eventually comes to assignment of rewards, RL is the gut base framework to that. In simply, Reinforcement Learning is a technique in Machine Learning that allows an agent to learn in an environment which is interactively do actions by trial and error and gets feedbacks as the experiences. Figure 2.1 shows the high-level architecture of RL framework.

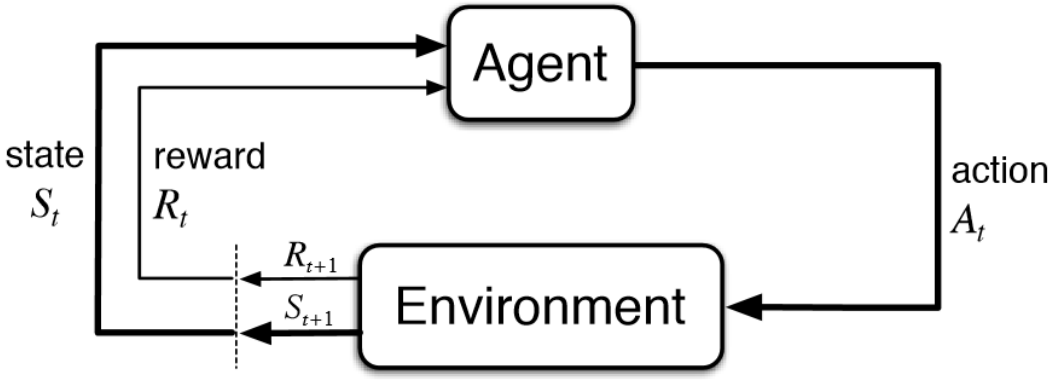


Figure 2.1 - High-level diagram of Reinforcement Learning Framework[1]

By using the RL environment, most of the research projects can be improved in both accuracy and implementation aspects with relative to the UL based developments. Despite this, RL environment enables to solve more complicated problems in VS such as TVS, Visual Storytelling, etc. This is where the TVS gets more recognition over VS field. There are several VS and TVS research projects based on SL framework and WSL framework. When it comes to WSL based RL developments in VS, K. Zhou et al. [34] have done a tremendous work to develop WSL plus RL based VS that is to exploit video summaries with category related details. For an example, through this model, particular interested area of a user can be obtained easily as a summarized context. To develop this model, they have established a sequential decision-making process which can be trained along with the deep Q-learning method in RL. To cover the long sequences of the large-scale video, a dense-ranking based reward function is introduced to resolve delays in temporal dimension and sparsity in spatial dimension. With the inspiration of this method, researchers are looking to evolve TVS along this approach with some minor modifications apparently. To extend the VS, textual related descriptions are used to depict summarized details of the large-scale videos rather than short and concise set of video clips itself. By using an HRL framework, Wang et al. [7] successfully implement a TVS model with the best accuracy and better inferencing time than the previous work. This is the first successfully design of TVS model using SL training framework. Eventually, usage of SL environment to develop TVS model

[1] This Figure is taken from https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

means that we need to pay the extra precautionary steps for selecting a proper dataset with text-based information. Despite of this, the model is lack of attention on word sequences of the annotated videos thus, there is a high probability that the model may miss the necessary information to be generated. As a remedy for lack of attention in word sequencing, H. Li et al. [35] use an NLP method to cover the issue in sliding window technique used by the previous research work. This method fixes the ranking mechanism of all the possible clip-sentence pairs in a segmented video. In addition, they have renovated SL framework to keep steady performance with RL based multi-tasking learning framework. This research makes an inspiration to develop Visual Storytelling VS which is another area of TVS. The core idea of Visual Storytelling is that the model can make storyline for the given video. Since the Visual Storytelling is yet a new area, very few research work can be found. The available emblem of Visual Storytelling research that we can see, is by Huang et al. [8]. They have merged the best learning algorithms from Wang et al. [7] and H. Li et al. [35], which is of Hierarchical Structured Reinforcement Learning (HSRL). This approach covers the creation of story description as a coherent multi-sentence basis. To implement this, two decoder-based networks have been used. In this networks, high-level decoder stands to construct a few plans for generating topics while low-level decoder tries to generate sentence sequences for each shot. As per the overview done, this is the current breakthrough in TVS with considerably best accuracy and better performance at the application level.

2.3.3 Supervised Learning based Video Summarization

With the idea of being used UL and RL frameworks in VS, we can see the significant improvements has been achieved. In the RL frameworks, the data subjectivity has been eliminated in some extend using the reward-based functions. Thus, researchers are looking to integrate SL frameworks into VS and TVS developments confidently. Despite the data subjectivity concern, a rapid number of datasets are created through the years in both qualitatively and quantitatively. Focusing on SL framework, the key idea is to train a model with the dataset which has labeled/annotated data points. The benefits of using SL framework in VS are that the model can be improved using few

datasets, ability to add some bogus noisy data points with labelling and engage the experts' insights through the datasets. With these benefits, number of sub fields in VS can be depicted recently such as query-related VS, viewer-interested VS, and semantically meaningful VS. Although the mutual inclusiveness of these methods can be seen, some deficits and positives among each method are visualized.

When it comes to query related VS, the idea of such method is that to generate description for the large-scale video by taking a query/request. As per the first phase, Huang and Worring [9] propose a methodology based on SL framework to get queries as an inputs and summarized video clip as per the result. Their learning algorithm is very simple since a controller and generator have been used to process queries and match the attention from text to video contents. Since the attention mechanism of this proposed solution is not sufficient to collect query information, Y. Zhang et al. [6] power the same approach with RL plus SL frameworks to make attention-based relationship among inputs and outputs. The novelty of this mechanism is correlated on reward factors, namely relatedness, representativeness, and diversity. In briefly, relatedness uses to catch the correlation of inputs and outputs, representativeness measures the representation score of the correct outputs, and finally diversity measures the mutually inclusiveness of the outputs. However, they have not covered the semantical information about the objects/items/events in the video. To resolve that gap, Wei et al. [10] use a method to integrate semantics meaning into the summarized video content. With relative to the previous learning algorithm/model, they have built the network with frame selector and video descriptor. The responsibility of frame selector is to select frames that consist of the semantical meaning, and the video descriptor is targeted to generate summary video contents followed by the previous method. This is not the only resolution for the same issue highlighted in the method by Y. Zhang et al. [6], K. Zhang et al. [11] also propose the same method with a different loss calculation component, called discriminative loss component. By using this loss calculation component, they have emphasized the preservation of original content can be done. These are the two-research works cover the query-related VS modelling with the best results.

Apart from the query-related VS model, viewer-interested VS models are also popularized with the perspective of recommendation systems. In brief about viewer interested VS, the interest of each viewer is different thus, the summary video contents should also cover the corresponding viewer interest. A. Kanehira et al. [12] implement a gigantic model for viewer-interested VS in text based. They have used the same mechanism from Wei et al. [10] and have embedded an optimization mechanism to generate three aspects; individual summary for key shots, relationship-bridge to maintain the flow of each summary, and finally group-summary to cover the entire text summary output. According to the overview, this is the complex yet promising method to be a next trend in the TVS.

Despite the both query-based and viewer-based TVS methods, the improvement on LSTM network has been carried forward in VS and TVS developments. This is where the attention has been tried to preserve inside the VS networks. An extension of LSTM called Bi-Directional LSTM (BiLSTM) is actively used in VS methods with mostly SL frameworks. When it considers in VS developments, Z. Ji et al. [14] try to use BiLSTM to encode contextual information in the input videos and decode the inherited information as per the summarized video clips then. This implementation is purely based with SL plus BiLSTM which shows much better results compare to the larger networks. With this base, Wang et al. [13] improve the same network with Stacked Memory Network (SMN) which is an extension of LSTM. They have resolved an issue of regular LSTM which has limited the hidden information being provided to the outside. As per a remedy, an augmented LSTM layer with an attached external layer is regulated. Therefore, usage of LSTM networks into VS is also providing a possible research opening. At a point, most of the current research interests are aligned with SL frameworks than RL and UL since the highlighted issues can be covered with the SL frameworks. However, the simple solutions can be made by using SL framework is still at the doubtful stage.

2.3.4 Latest Trend in Video Summarization

In the previous in-depth overview of literature review, there are some concerns/issues highlighted in three environments respectively. The most outstanding concern through

the literature is that only few research works have attempted to enhance the extraction of attention-based temporal information of the input large-scale videos. Extraction of attention information is considered as a sequence-based problem which is the basic intention in NLP related problems. Same theory can be applied into the image frame sequences as a substitution to extract attention-based temporal information. However, the patterns/features must be learnt by such models as much as possible to understand more information since a lot of combinations can be seen. In the latest NLP models, transfer learning on a huge unlabeled dataset is performed firstly and do the operational training on the selected labeled dataset later. This mechanism is applied in the most STOA models in NLP field such as GPT [17] and BERT [16] related models. With the inspiration of these techniques, Shang et al. [18] depict a combined model with NLP based temporal attention learning algorithm in VS which is called Time-Aware Multimodal Transformer (TAMT). In addition to the above techniques, a transformer-based network has been executed to enhance performances and accuracy towards STOA. Through this model, transformer-related networks have been emphasized to gain accuracy and performances. Therefore, the latest trend of VS is based upon Transformer-based architectures.

Despite the major developments in the VS field itself, some other related areas such as NLP area and Vision based ML area are showing the promising findings as well. Text-To-Text Transfer Transformer (T5) [21], is also recognized as a tremendous finding in NLP area since this is the latest trend for most of the NLP models which shows the better results and faster inferencing with relative to the previous STOAs. Even though the GPT [17] and BERT [16] models are highlighted, T5 model shows the absolute replacement over many existing NLP models. When it comes to Vision-based ML, Bertasius et al. [20] who are working in Facebook, have introduced a complete anti-CNN network, namely, TimeSformer (from Time-Space Transformer) which is inspired from the Transformer architecture [36] in NLP task and it is used in Video Recognition tasks. As per their claim, TimeSformer method has surpassed the existing STOAs in CNN-based networks with three time faster in training process and one-tenth time faster at inferencing level. Since the depicted performance in both training

and application level, developments in VS are also focused to adopt this method exponentially.

2.4 Summary of Past related research

In the literature review, major benefits and issues have been identified in the most cited research. These are highlighted in Table 2.1

Table 2.1- Summary of benefits and issues in the past related researches

Research	Benefits	Issues
VS via Semantic Attended Networks [10]	Query-controllable VS for better user experience.	Keyword queries are the only supported. It cannot use sentence or paragraph-based queries.
Retrospective Encoders for Video Summarization [11]	Minimal annotation embedded dataset is required.	Performance of the system is not sufficient.
UL Video Summarization with Attentive Conditional GANs [31]	Ability to capture long-range temporal information with self-attention multi head network.	Generator in GAN network drops information. Complicated optimization process is required to make network into converged state.
Discriminative Feature Learning for Unsupervised Video Summarization [33]	Easy to capture local and global temporal views. Resolve flat importance score distribution issue.	Network is too complicated to apply in an application.
Cycle-SUM: Cycle-consistent Adversarial LSTM networks for UL Video Summarization [37]	A complete free of labeled dataset approach to train network which eliminates dataset subjectivity and biasness.	Information loss at the training phase.
Unsupervised VS with a Convolutional Attentive Adversarial Network [4]	Augmented memory layer in LSTM improves the ability to capture long-range temporal information.	An expensive network to train and hard to understand the architecture of the network.
Video Summarization by Classification with Deep RL [34]	Weakly SL based RL with dense reward in Deep Q-Learning to nullify credit assignment issue.	-

Video Captioning via Hierarchical Reinforcement Learning [7]	Resolve the deficiency in stochastic and deterministic policy gradients.	Single Manager as an agent to multiple workers is not sufficient to cover the workload.
Read, Watch, and Move: RL for Temporally Grounding NLP Descriptions in Videos [15]	NLP descriptions for videos without enabling sliding over video mechanism.	Complicated network to train and not feasible to apply in real applications.
Multimodal Video Summarization via Time-Aware Transformers [18]	Both Spatial and Temporal information is preserved.	Training time will be taken than CNN based training.

2.5 Taxonomy of the Video Summarization Algorithms

The Figure 2.1 shows the taxonomy of Video Summarization, and Our targeted area is highlighted with dotted box.

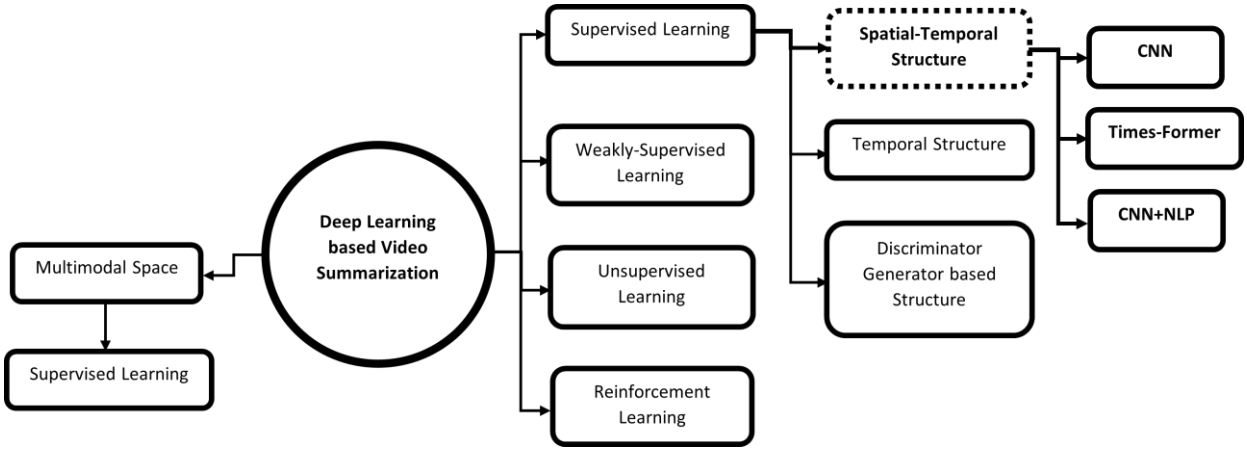


Figure 2.2- Taxonomy of Video Summarization

2.6 Challenges in Video Summarization in Text Format

According to the current research knowledge in the previous sections, a few challenges can be emphasized.

2.6.1 Selection of Dataset

Data Subjectivity has been identified in the UL based approaches and the reason to use UL based methods is to eliminate data subjectivity concern. Data Subjectivity has been recognized as the preparation of dataset according to the knowledge, scope, vision, and other factors such as biasness towards the specific application. Hence, the selection of complete data subjectivity free dataset is a challenging step.

2.6.2 Unsuccessful Multimodal approaches

Multimodal approaches did not show the performance against the Unimodal approaches [1] even though the rich information has been embedded into the model. Despite the multi-input models, combination of videos with text descriptions are also showing some deficits such as hard convergence of the model. Improving the Multimodal networks is relatively harder than developing Unimodal networks.

2.6.3 Lack of facilitation to change the model

Almost 100% of the models are developed in single chassis with multiple sub modules. However, when it subjects to change the atomic levels in the model chassis, it is the hardest phase with comparing to development the model at scratch phase. Interrelationship of each module is not a mutually inclusiveness thus, single change in a module may reflect an unstable stage to the model chassis.

2.6.4 Lack of explanation capability

Apart from the Deep Learning based models, current research community is looking for explanation of the models (XAI) rather than keep them as a black box. All the research works performed so far are unable to explain the activities going in the model level. This will create a blackhole about the model and thus it makes much harder to identify any vulnerabilities inside the model level.

2.6.5 Deployment

Most of the VS models and TVS models are unable to deploy into the application stage with the existing low-level hardware components such as personal computer, or an edged device. This is because of the network architectures are extremely huge and can only be executed in cloud-based solutions. This is the most common problem in Deep Learning methods of TVS and a lot of room available to resolve this problem.

2.7 Problem Definition

Existing research in the field of VS does not have an enough practical implementations in the real world. Hence a simple but on par with STOA method for VS is required. The VS needs to use simple machine learning algorithms in end-to-end framework. Apart from that, the usage of datasets for the system should not be focused into one orientation but should cover the multiple aspects to get the broader picture about the field. Therefore, the problem definition must be placed to cover the challenges identified in the previous section.

In this research, we will present a method to build an efficient textual summarization for large-scale videos using simple supervised learning-based machine learning algorithms and a simple rule-based system.

2.8 Summary

In this chapter, a critical review of VS and its variants has been covered through early evolution, breakthroughs and trends, and challenges with existing techniques. In the next chapter, we will present the highlight on technology used in Text based Video Summarization.

Chapter 3

TECHNOLOGIES USED FOR VIDEO SUMMARIZATION

3.1 Introduction

In the previous chapter, in depth and critical literature review for Text-based Video Summarization is presented in the sections of early developments, breakthroughs and trends, the latest trend of TVS, challenges of Video Summarization in Text form, and finally the problem definition. Through the latest trends in Video Summarization section, both TimeSformer [20] and Text-To-Text Transfer Transformer [21] have been emphasized on the potential benefits of using them into Video Summarization. In this chapter, we thoroughly discuss the technology used in this research for sub tasks such as Object Detection, Video Classification, and Text Generation under the areas of different variants of the model, its mathematical explanation, and the effectiveness into our research.

Figure 3.1 shows the intended technology being used for each sub tasks of the research work.

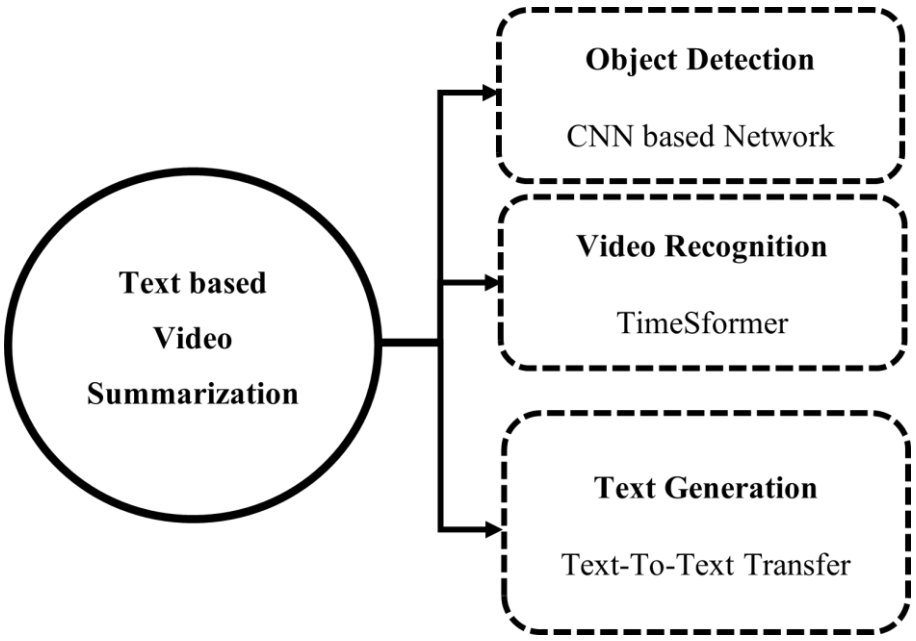


Figure 3.1- Technology Stack of Text based Video Summarization

3.2 Convolutional Neural Networks – for Object Detection

Convolutional Neural Networks (CNNs) are a type of feed-forward neural networks which the connections between layers are inspired from the structure of visual cortex in the human brain. It takes the input as an image, identify some importance aspects of the image such as edges, intensity, borders, corners, etc. into reconfigurable or learnable weights and biases, and then provide the label/class of the input image [22]. This method erupts the traditional machine learning methods which are supposed to implement manually in image classification with only a set of labeled data.

The Figure 3.1 shows the brief architecture of the CNN network.

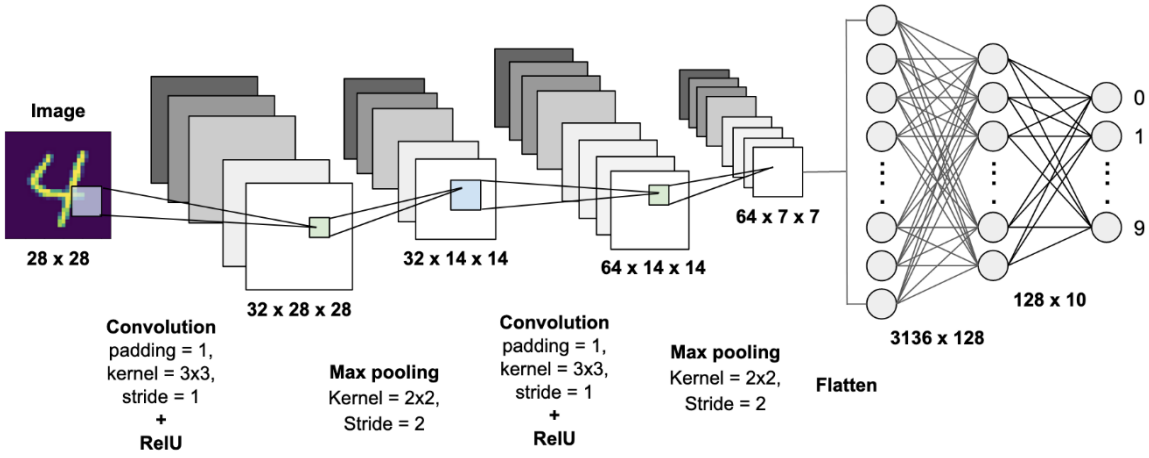


Figure 3.2- CNN Architecture in Image Classification task

In the image classification task, CNN plays an important role with a set of convolutional layers, imposes to do convolution operation on '*nxn'* matrices, followed by both activation layers and sampling layers (max pool layers), respectively. With stacking of few combined convolution layer, activation layer, and pooling layer sets, final output will return as a classification result at the flatten stage which is also considered as fully connected layer. At operation level, convolution layers supposed to extract the high-level features such as edges, corners, borders, etc. which are the basic shapes of an image. When the layers are getting into deeper level, more high-level features will be captured of the image and providing simple input feeds to the flatten stage. The intended objective of pooling layers is to provide importance value

out of a selected scale thus, it will handle more robust and feature invariant operations despite the image orientation and scale changes. The first layer of the network does the conversion of input image features into relevant feature map scale at CNN layer.

The evolution of image classification task in ML has begun due to the introduction of CNNs. Despite image classification, other fields of the ML such as object detection and object recognition are powered by mostly CNN based architectures. Hence, this smart trend shows the deviation of traditional vision-based ML tasks into new era. Due to the supplement of CNNs and its architecture, it has been used for Video Classification tasks to simplify feature space in the videos as described in Chapter 2.

In our research, we use CNN based network called YoloV5 [38] to recognize objects in the videos without changing the primary layout of CNNs. To add more Convolutional layers, we change the last layer in framework, called Detect. Along the Text-based Video Summarization pipeline, this framework provides the main objects in the video that can identify within a short amount of time.

Our next target is to capture the main events/behaviors in the large-scale video input. To accommodate it, we use TimeSformer network which will be described in the next section.

3.3 TimeSformer – A new trend

This is the core of our research, namely, TimeSformer [20] which is used to recognize actions in the video space. Through this section, the model description in a form of computational orientation, and the variants available in the network are being discussed thoroughly.

3.3.1 Mathematical Description – TimeSformer

Input clip and Decompositions of patches. The TimeSformer model takes an input as a video clip of samples that consisting of F RGB frames in H x W size from the original video. Input clip X is defined as equation (1).

$$X \in \mathbb{R}^{H \times W \times 3 \times F} \quad (1)$$

N non-overlapping patches will be decomposed on each frame from the *X* clip, each of size $P \times P$, as following mapper equation (2).

$$N = \frac{HW}{P^2} \quad (2)$$

These *N* patches are then subjected to vector through flatten operation as $x_{(p,t)} \in \mathbb{R}^{3P^2}$ where $p = 1, \ldots, N$ indicating spatial locations while $t = 1, \ldots, F$ indicating an index over frames.

Embeddings and Query-Key-Value Calculation. Each $x_{(p,t)}$ is then mapped linearly into an embedding vector $z_{(p,t)}^{(0)} \in \mathbb{R}^D$ through a learnable matrix $E \in \mathbb{R}^{D \times 3P^2}$ as the equation (3).

$$z_{(p,t)}^{(0)} = Ex_{(p,t)} + e_{(p,t)}^{pos} \quad (3)$$

In equation (3), $e_{(p,t)}^{pos} \in \mathbb{R}^D$ represents learnable positional embedding vector which is used to encode the spatiotemporal features of each *N* patch. These input sequences represent the same input configuration to the Transformer module in the NLP. A special learnable sequence vector $z_{(0,0)}^{(0)} \in \mathbb{R}^D$ is added to depict the classification token. The next part is the inherent Transformer block.

This Transformer contains *L* encoding blocks and each block, indicates ℓ, the separate query, key, and value vectors are computed for each patch from $z_{(p,t)}^{(\ell,a)}$ as equations (4), (5), and (6).

$$q_{(p,t)}^{(\ell,a)} = W_Q^{(\ell,a)} LN\left(z_{(p,t)}^{(\ell-1)}\right) \in \mathbb{R}^{D_h} \quad (4)$$

$$k_{(p,t)}^{(\ell,a)} = W_K^{(\ell,a)} LN\left(z_{(p,t)}^{(\ell-1)}\right) \in \mathbb{R}^{D_h} \quad (5)$$

$$v_{(p,t)}^{(\ell,a)} = W_V^{(\ell,a)} LN\left(z_{(p,t)}^{(\ell-1)}\right) \in \mathbb{R}^{D_h} \quad (6)$$

Layer Norm equation is denoted as $LN()$ and $a = 1, \ldots, \mathcal{A}$ is an index through the multiple attention heads. $\mathcal{A}$ says the total number of attention heads in the module. D_h is the latent dimensionality in each attention head that is calculated by $D_h = D/\mathcal{A}$.

Encoding using Self-Attention. Computation of self-attention weights $\alpha_{(p,t)}^{(\ell,a)} \in \mathbb{R}^{NF+1}$ are done by using a SoftMax activation of dot product function shown in equation (7).

$$\alpha_{(p,t)}^{(\ell,a)} = SM\left(\frac{q_{(p,t)}^{(\ell,a)^{\mathrm{T}}}}{\sqrt{D_h}} \cdot \left[k_{(0,0)}^{(\ell,a)}\{k_{(p',t')}^{(\ell,a)}\}_{\substack{p'=1,\ldots,N \\ t'=1,\ldots,F}}\right]\right) \quad (7)$$

If the attention is computed only over one dimension, say temporal-only, then the computational cost is drastically reduced as the indication of equation (8).

$$\alpha_{(p,t)}^{(\ell,a)time} = SM\left(\frac{q_{(p,t)}^{(\ell,a)^{\mathrm{T}}}}{\sqrt{D_h}} \cdot \left[k_{(0,0)}^{(\ell,a)}\{k_{(p',t')}^{(\ell,a)}\}_{t'=1,\ldots,F}\right]\right) \quad (8)$$

The final encoding $z_{(p,t)}^{(\ell)}$ at block ℓ is attained by calculating weighted summation of self-attention coefficients from each attention head, provides in equation (9), and then concatenate such output vectors through Multilayer Perceptron (MLP), using residual connections after each operation, depicts in equation (10) and (11).

$$s_{(p,t)}^{(\ell,a)} = \alpha_{(p,t),(0,0)}^{(\ell,a)} v_{(0,0)}^{(\ell,a)} + \sum_{p'=1}^{N}\sum_{t'=1}^{F} \alpha_{(p,t),(p',t')}^{(\ell,a)} v_{(p',t')}^{(\ell,a)} \quad (9)$$

$$z'^{(\ell)}_{(p,t)} = W_O \begin{bmatrix} s_{(p,t)}^{(\ell,1)} \\ . \\ . \\ . \\ s_{(p,t)}^{(\ell,\mathcal{A})} \end{bmatrix} + z_{(p,t)}^{(\ell-1)} \quad (10)$$

$$z_{(p,t)}^{(\ell)} = MLP\left(LN\left(z'^{(\ell)}_{(p,t)}\right)\right) + z'^{(\ell)}_{(p,t)} \quad (11)$$

The final video clip embedding output is taken from the equation (12) with the final block classification token.

$$y = LN\left(z_{(0,0)}^{(L)}\right) \quad (12)$$

3.3.2 Sub variants of TimeSformer

There are sub variants of TimeSformer architecture introduced and shown their high-level architecture in Figure 3.2

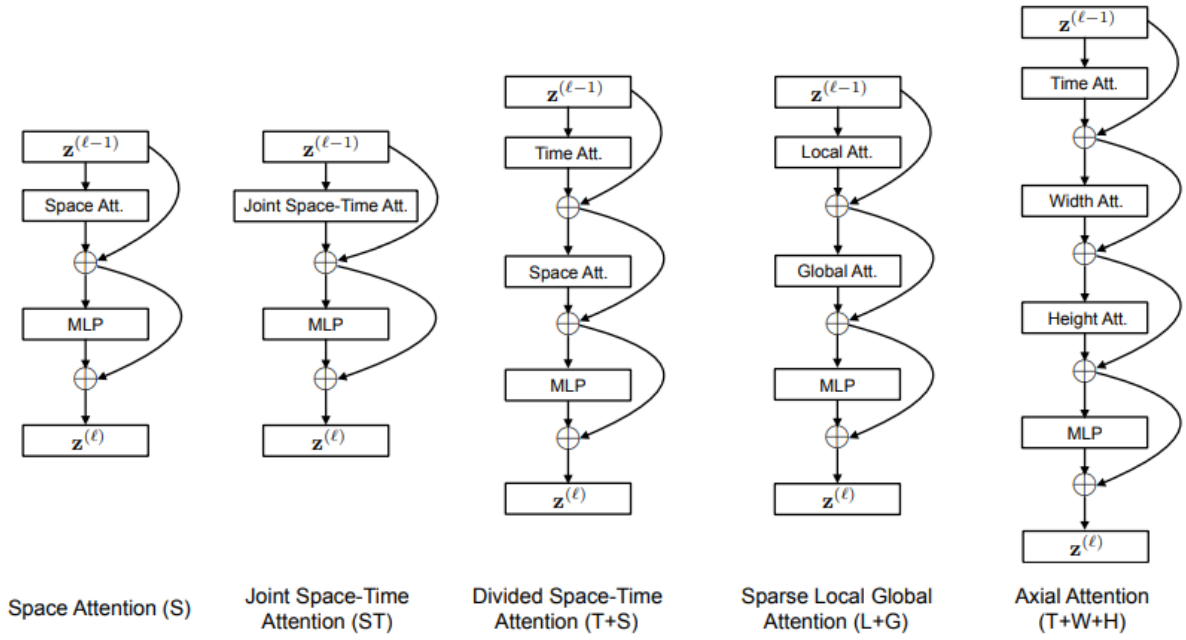


Figure 3.3- Different model architectures for TimeSformer [20]

Space Attention TimeSformer can only tackle the spatial information while it does not have the ability to capture time related information. This will again revert to the issue in CNN architecture. Joint Space-Time Attention (ST) TimeSformer, as described in equation (7), can capture both spatiotemporal information. Though the information harvest is superior in ST than Space Attention only, it takes too much long to do inferencing and drag days of training time. To alleviate this inefficiency, a superior method has been introduced, named Divided Space-Time Attention (T+S as a notation) where temporal attention and spatial attention are applied separately and one after other.

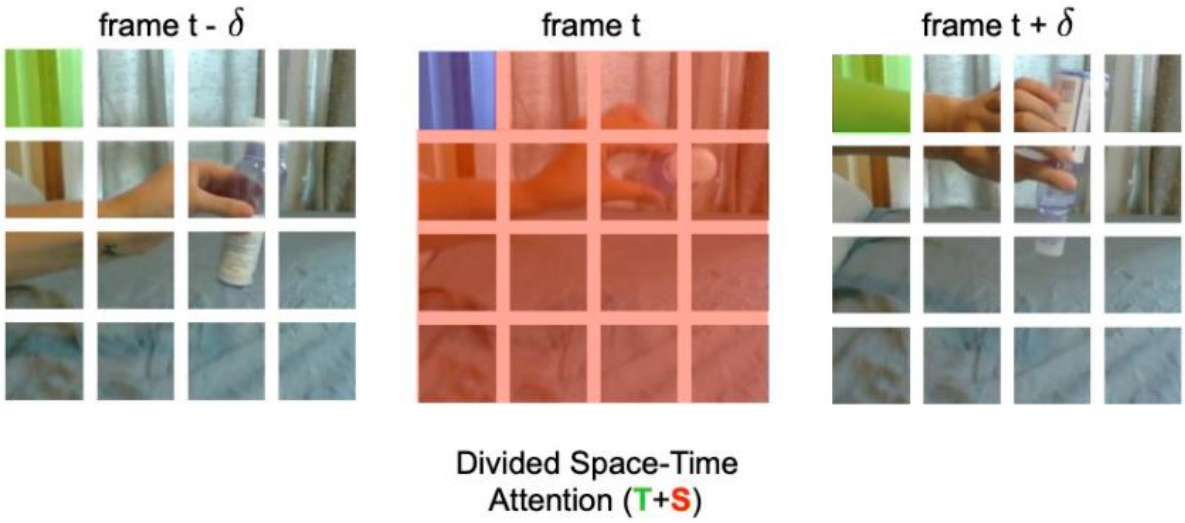


Figure 3.4- Visualization of Divide Space-Time Attention Module

When temporal attention is applied, each patch is only compared with patches at the same spatial place in other frames. In Figure 3.3, blue color patch at frame t is only compared with green color patch in $t - \delta$ frame, $t + \delta$ frame, and rest of the frames. For this calculation, equation (8) is used. When spatial attention is used, the patch at frame t only compared with rest of the patches in the same frame t. In the same Figure 3.3, blue color patch is compared with red color patches at frame t.

Divide Space-Time Attention TimeSformer architecture is preferred to use in video classification in Video Summarization. With both video classification and object detection, our next target is to capture the NLP sentence for the given information.

3.4 Text-To-Text Transfer Transformer – for Text output

Text-To-Text Transfer Transformer [21] is also based on Transformer [36] technique and leveraging the transfer learning method. Through this section, the model input and output, the model in brief, and its sub variants will be discussed.

3.4.1 Input and Output of T5

T5 model takes input as a text stream and returns the output as a text also. Due the model is trained on large unlabeled data, it can use the same intension and architecture to perform language translation, text regression, text classification, and text

summarization. These tasks are defined in the input text where at front of the input and separate the input objective with a colon.

3.4.2 T5 Model Architecture

T5 model is essentially an Encoder-Decoder Transformer, based on Vaswani et al. [36] with some modifications, similar to equation (4), (5), and (6) in TimeSformer model, such as applying Layer Normalization prior to the sub blocks and then summing the initial input to the sub block output later. Attention of each Query, Key, and Value is summarized on the equation (13).

$$Attention(Q, K, V) = SoftMax\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V \quad (13)$$

Model configuration is equivalent to the BERT base configuration [16].

3.4.3 T5 Model Variants

As in the TimeSformer model, T5 also consists a few sub variants and trained each variant to experimentally prove which is the best model variant. The Figure 3.5 shows the high-level architecture of each variant.

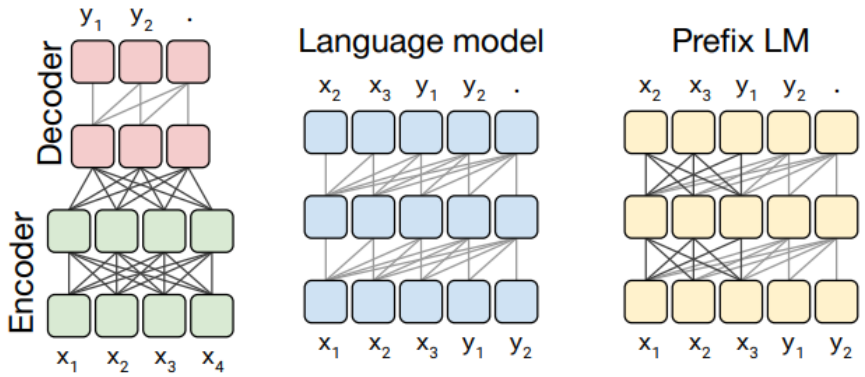


Figure 3.5 - Model Architectures of T5

Encoder-Decoder (left of Figure 3.5) is used the standard encoder-decoder, seq2seq architecture. Encoder is moreover trained in a BERT style (says fully visible) that is every token uses to calculate how much attention relates to each other token in the

sequence, while Decoder is trained in mostly a GPT style (says casual) that is all token attend to every token which occur before in the sequence.

Language Model (middle of Figure 3.5) is mostly a casual attention mechanism and autoregressive modeling approach.

Prefix Language Model (right of Figure 3.5) is a combined approach of both BERT style and Language Model which supposed to translate one language to other language.

Out of these variants, Encoder-Decoder based T5 is the best and we also select the same architecture for our Text Generation task.

3.5 Summary

Through this chapter, we cover the technology stack for our research interest in the areas of Object Detection, Video Recognition, and finally Text Generation. An insight of CNN based Network and YoloV5 as a selection for Object Detection task is delivered while the depth and breadth of TimeSformer and its usage is covered with enough detailed. For Text Generation, our focus of T5 model is elaborated to understand its power and the usage. In the next chapter, the overall research approach will be discussed through inputs, outputs, process, features, and users along the hypothesis.

Chapter 4

APPROACH

4.1 Introduction

The previous chapter described the technologies and the concepts that are adopted for this research. In this chapter, chapter four, we extensively describe our approach for implementing those technological terms of input, output, users, features, process, and hypothesis.

The Figure 4.1 shows our proposed approach for Text-based Video Summarization as an IPO (Input-Process-Output) format.

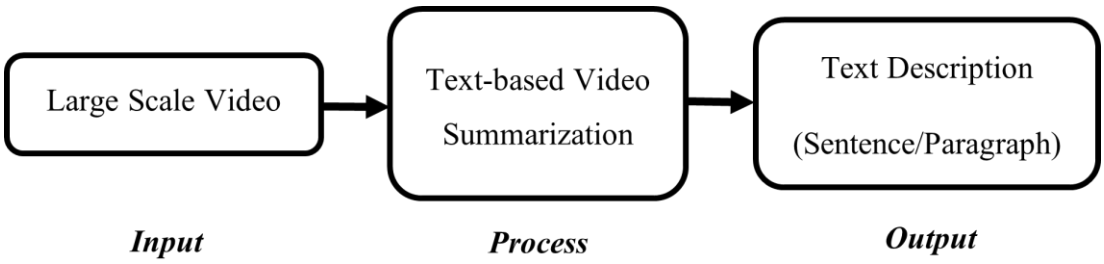


Figure 4.1- Input-Process-Output for Text-based Video Summarization

4.2 Hypothesis

We hypothesize that **Supervised-based Simple Machine Learning with Rule-based algorithm** can be used to develop the Text-based Video Summarization for large-scale videos so that the system will use less computation resources and easily deploy into the real-world application.

4.3 Input

The input for the Text-based Video Summarization is the single large-scale video, or a surveillance video. This file should be an offline, or pre saved file.

4.4 Output

The out of the system is either a sentence that describes the key idea upon the context or a set of sentences as a paragraph that will provide the main points through the input it found.

4.5 Process

We propose the system called SSML-TVS, namely Simple Supervised Machine Learning for Textual Video Summarization.

For the given input, the system will create a set of small clips, each has a period of one to two minutes in average. For each clip, the system will recognize the actions involved in it, and the objects that is subjected for each action.

With both action and object, the system will prepare action-object pairs with the best effort by use of a simple rule-based mechanism. Then the corresponding action-object pairs will be input into the Text-creator to get the meaningful and short form sentences.

Finally, all the generated sentences will be accumulated and return as an output.

4.6 Users

The users can be identified as people who are in multiple areas such as social media, security and internal affairs, aviation, naval, transportation, educational system, government, movie industry, news industry, and so on. Thus, this is not a narrow shape of project consider the users it has.

4.7 Features

Features, on the other hand, can be seen as low-cost implementation since it uses only simple machine learning algorithms with a simple rule-based system.

The system does not need to use the specific dataset to be trained since the system entities can be trained individually without respect to the end approach.

The approach also provides the facility for controlling application level such as aid for visually impaired people, prepare abstracts for movies in movie industry, and much more.

4.8 Summary

This chapter provides our approach of the research and describes the hypothesis of the research, what are the inputs, outputs, features, users, and the process in detailed wise.

In the next chapter, we discuss the design that implement the technology described in the previous chapter, chapter three, with the approach described in this chapter.

Chapter 5

DESIGN

5.1 Introduction

In the previous chapter, the approach for the research is described in the sections of hypothesis, input, output, process, features, etc. In the chapter before, the technological concepts which are related to the research have been discussed clearly. Through this chapter, the top-level design of the research will be discussed extensively with the support of both design chapter and the technology chapter.

5.2 High-level Architecture of the Design

Our proposed **T**ext-based **V**ideo **S**ummarization using **Si**mple **Ru**le-based **M**achine **L**earning system (SiRuML-TVS) consists of three main modules, namely, Action Recognition, Object Detection and Natural Language Processing (NLP) based Text Creator. With these major modules, the system contains a simple module for small-clip generation. Each module is described in detail in the following sections.

The top-level architecture of the SSML-TVS is shown in the Figure 5.1.

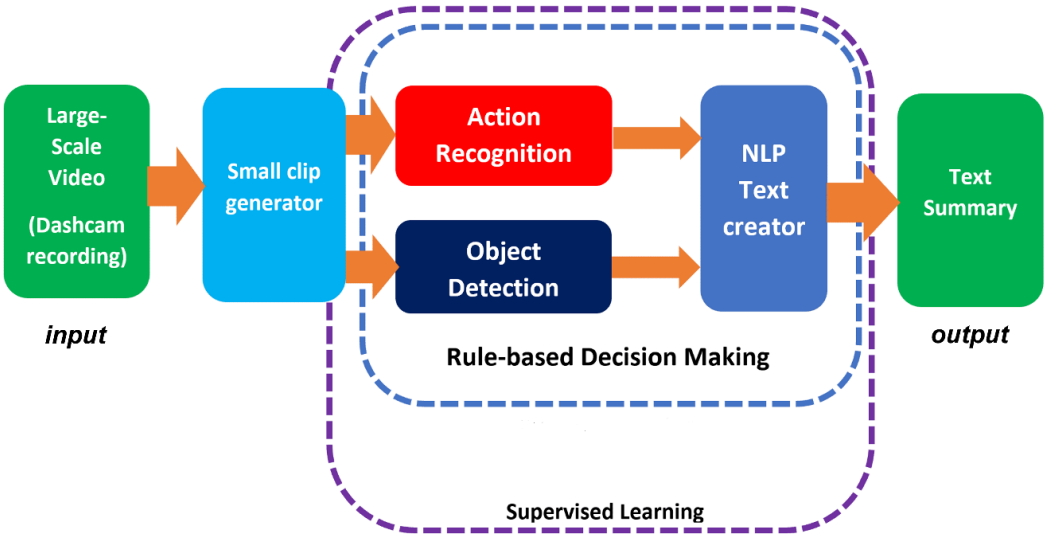


Figure 5.1 - Top-level Architecture of SSML-TVS

5.3 Small-clip Generator

This module stands right after the input which is a large-scale video, and the module supposes to create mini clips from the given input video. Creating the mini clips from a large-scale video is necessary to the main modules since the processing power and the computation time should be kept at the moderate level, and to provide the results within an acceptable time rather than hanging in the time.

This is the first module in the simple rule-engine based system, and it does not use any Machine-Learning approach to create mini clips however, it will create a clip with 1–2-minute time length.

Generated mini clips will be directly used by two main modules, called Action Recognition and Object Detection.

5.4 Action Recognition

Action Recognition module is the core module out of three main modules. This module provides the main action or event of the video clip, and by which object. For an example, if a video contains the orchestra and a pianist is playing a piano, then the module will say as an action called “playing music instrument” and the object as “person”. Main role of this module is to provide the main action and the corresponding object of responsible to the action. This module uses a mini clip as an input from the small-clip generator module.

Action Recognition module is a Machine Learning based module and is trained on the dataset which has a set of actions in the selected scope. This module is limited for the application of interested. Identified action and object of each mini clip are taken by the NLP Text Creator module.

5.5 Object Detection

Object Detection module is the next module which uses the same input as Action Recognition module. This module uses to identify each, and every object reside in the video clip. By using the same example in the previous section, which is a pianist is

playing a piano in an orchestra, the object detection module will capture all the objects such as piano, person, violin, drum, stage, etc.

This module is also based on the Machine Learning and is trained on the dataset of application related only. A very simple Object Detection algorithm is enough to perform this task. Detected objects are given to the NLP Text Creator module.

5.6 NLP Text Creator

Natural Language Processing (NLP) based Text Creator module is the last module out of the main modules. This module takes inputs from two main modules, namely Action Recognition and Object Detection. This module contains three sub modules as depicted in the following Figure.

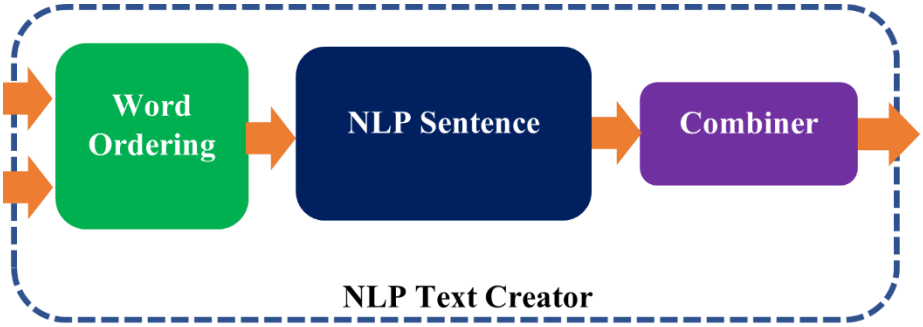


Figure 5.2 - Sub modules in the NLP Text Creator

In the Figure, two arrows at the left are shown the inputs from the Action Recognition and Object Detection modules respectively. Each sub module is described as following sections.

5.6.1 Word Ordering

Word Ordering module assembles the key words from the Action Recognition and Object Detection modules in an order of main object, main action, and rest of the objects. This order of words is then subjected to the next module called NLP Sentence.

5.6.2 NLP Sentence

This module takes the input from Word Ordering which is a order of words, and processes to a sentence which is about the context of the application of interested. This module is based on Machine Learning and supposes to create a meaningful sentence by the given a set of words. For an example, if this module gets the words like person, playing music instrument, piano, violin, stage, etc. and then the module generates a sentence like “A person is playing a piano on the stage with other instruments of violin, saxophone, flute.”

The generated sentence is then transferred to the next sub module called Combiner in NLP Text Creator.

5.6.3 Combiner

Combiner module takes the sentences from the NLP Sentence module and composes into the paragraph, else takes a sentence and outputs the same.

The final output of the NLP Text creator is a either paragraph or a single sentence which can explain the context of the application.

5.7 Summary

This chapter explained the design of the Text-based Video Summarization using Simple Rule-based Machine Learning Algorithms for testing our hypothesis. The top-level architectural design was discussed thoroughly. In the next chapter, we cover the details on the implementation of the design discussed in this chapter.

Chapter 6

IMPLEMENTATION

6.1 Introduction

In the previous chapter, the design of Text-based Video Summarization using Simple Rule-based Machine Learning system was discussed extensively with the top-level architecture. In this section, we dive into the deep of the concepts, mathematics, datasets, and the frameworks used for the implementing the proposed design in the previous chapter.

6.2 System Requirements

To initiate the design of application, we select a GPU supported cloud server from Lambda-Labs[2] cloud solutions with the following system specifications.

1. CPU Processor – AMD EPYC™ 2.50 GHz with 28 Virtual CPUs
2. RAM capacity – 200 GiB
3. GPU – 2x Nvidia RTX™ A6000 with 48 GiB VRAM each
4. On-Demand Price - $4.50 per hour

6.3 Framework for Action Recognition

There are many opensource resources available for Action Recognition task including open-source GitHub projects. However, the initialization of setup and resource usage are quite large and cannot afford. Due to these reasons, we select MMAction2[3] from OpenMMLab[4] framework. OpenMMLab provides a few advantages with relative to the other frameworks which are,

1. The framework covers more than 20 fields, including Action Recognition under MMAction2, with more than 250 algorithms and more than 2000 pre-trained models.

[2] Official website for Lambda Cloud Labs: https://lambdalabs.com/service/gpu-cloud
[3] GitHub repositories for MMAction2: https://github.com/open-mmlab/mmaction2
[4] GitHub repository for OpenMMLab: https://github.com/open-mmlab

2. Both CPU and GPU supportability even at the training stage.
3. Flexibility and Easiness for using with well-maintained documents and community support.
4. Multiple Dataset structures such as COCO [39], VOC Pascal[5], MovieNet[6], etc. are supported as the configuration into the framework.
5. The framework is written by using Python programming language hence it is easy to understand.

To use MMAction2 framework, we only require providing the script of model, which is going to be used, the configuration file for appropriate dataset, and the system configuration file such as number of GPUs, batch size, number of epochs, learning rate, etc. Once the job is submitted, the model training plots against the number of epochs can be visualized by using either TensorFlow Board or Weights and Biases, also known as Wandb[7], web-based platforms.

6.4 Framework for Object Detection

For the Object Detection framework, we select Ultralytics[8] open-source YoloV5 framework that is based on the original YoloV3 Darknet Object Detection Algorithm. This framework supports model training, model evaluation, and model deployment in the Object Detection applications with minimum requirements.

YoloV5 framework also provides the best real-time object detection relative to the other frameworks, and only harvests the minimum resources. This framework gives all the advantages explained in the previous section. Since we only require detecting objects with a single Machine Learning algorithm, this framework is selected.

[5] Official PASCAL VOC website: http://host.robots.ox.ac.uk/pascal/VOC/
[6] Official MovieNet dataset website: http://movienet.site/
[7] Wandb MLOps platform website: https://wandb.ai/site
[8] Ultralytics YoloV5 Official website: https://github.com/ultralytics/yolov5

6.5 Dataset Preparation

6.5.1 Action Recognition and Object Detection

We select METEOR: A Massive Dense and Heterogeneous Behavior Dataset for Autonomous Driving to train both Action Recognition and Object Detection Models. The size of Dataset is more than 100 GB, and it requires to get the permission by the original Author, Chandra et al. [40].

Once the dataset gets downloaded, the folder structure of the dataset should be organized with annotations and frames out of the videos. Folder structure of the METEOR dataset is shows as the Figure 6.1.

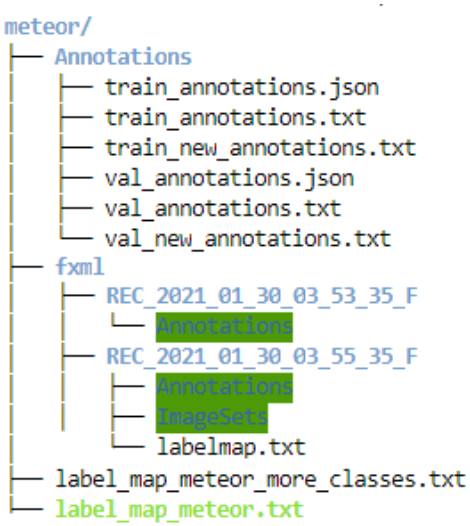


Figure 6.1- Folder Structure of METEOR Dataset

To generate frames from the videos, we modify the available script from the GitHub repository by METEOR according to the structure above. Since the original dataset contains only few classes of action-behavior related, we require to extend the classes from original format to indicate actions and behaviors separately since original format shows single label for a set of actions. For example, Action label compromises U-turn, Left-Turn, Right-Turn actions together. The following classes, shown in the Figure 6.3, are the extended version of the original classes, shown in Figure 6.2, and this operation provides 18 classes to achieve the model training into atomic level.

```
new_labels = {'LaneChanging(m)': {'True'}, 'Cutting': {'True'}, 'Yield': {'True'}, 'OverTaking': {'True'},
              'LaneChanging': {'True'}, 'ZigzagMovement': {'True'}, 'OverSpeeding': {'True'},
              'RuleBreak': {'TrafficLight', 'WrongLane', 'WrongTurn'},
              'Action': {'Breaking', 'Keep', 'LeftTurn', 'RightTurn', 'UTurn'},
              'Scenes': {'Intersections', 'RoundAbout', 'StraightRoad', 'TrafficSignal'}}
```

Figure 6.2 - Original classes provided from METEOR dataset

```
new_labels_mapping = {'LaneChanging': 0, 'OverTaking': 1, 'Yield': 2,
                      'Cutting': 3, 'ZigzagMovement': 4, 'OverSpeeding': 5,
                      'TrafficLightRuleBreak': 6, 'WrongLaneRuleBreak': 7,
                      'WrongTurnRuleBreak': 8,
                      'BreakingAction': 9, 'KeepAction': 10, 'LeftTurnAction': 11,
                      'RightTurnAction': 12, 'UTurnAction': 13,
                      'IntersectionsScenes': 14, 'RoundAboutScenes': 15,
                      'StraightRoadScenes': 16, 'TrafficSignalScenes': 17,
                      'LaneChanging(m)': 0}
```

Figure 6.3 - Extended classes from the original classes

For Object Detection task, the given classes are sufficient to the application of interested.

We split the dataset of both tasks for training and validation with the ratio of 80:20 once the folder structure is finalized.

6.5.2 NLP Text Creator

For this purpose, two datasets are used where one of them is a large dataset while the other dataset is a simple and small which related to the scope of this research project.

WebNLG [41] Corpus is the large dataset that is selected for model training at first phase. This corpus maps RDF-triples to an English text. For RDF-triples, DBpedia[9] web source is selected since it is a gigantic knowledgebase that uses for preservation information to query through SPARQL. To represent them as text format in dataset, texts from DBpedia were collected using crowdsourcing method.

Then, we require to create a small dataset that contains the application related sentences to learn the context. To achieve this operation, 1000 sentences are generated manually and then split the dataset to training and validation with 70:30 ratio.

[9] Official DBpedia website: https://www.dbpedia.org/

This dataset has the structure of input text and target text. The shape of Input text should be matched to this structure, which is main object, type of behavior, action/behavior, rest of object/s respectively. The target is a set of sentences to one input shape. Figure 6.4 shows the structure of Input Text and Target Text.

input_text	target_text
Car | scenario | lanechanging	Car is trying to change the lane.
Car | scenario | lanechanging	Car has changed its lane.
Car | scenario | lanechanging	Car changes the lane.
Car | scenario | lanechanging && Car | rulebreak | wronglane	Car changes the lane but it's a wrong lane.
Car | scenario | lanechanging && Car | rulebreak | wronglane	Car has changed to the wrong lane.
Car | scenario | lanechanging && Car | rulebreak | wronglane	Car is changing the lane in a wrong way.
Bus | scenario | lanechanging	Bus is trying to change the lane.
Bus | scenario | lanechanging	Bus has changed its lane.
Bus | scenario | lanechanging	Bus changes the lane.
Bus | scenario | lanechanging && Bus | rulebreak | wronglane	Bus changes the lane but it's a wrong lane.
Bus | scenario | lanechanging && Bus | rulebreak | wronglane	Bus has changed to the wrong lane.
Bus | scenario | lanechanging && Bus | rulebreak | wronglane	Bus is changing the lane in a wrong way.
motorcycle | scenario | lanechanging	motorcycle is trying to change the lane.
motorcycle | scenario | lanechanging	motorcycle has changed its lane.
motorcycle | scenario | lanechanging	motorcycle changes the lane.
motorcycle | scenario | lanechanging && motorcycle | rulebreak | wronglane	motorcycle changes the lane but it's a wrong lane.
motorcycle | scenario | lanechanging && motorcycle | rulebreak | wronglane	motorcycle has changed to the wrong lane.
motorcycle | scenario | lanechanging && motorcycle | rulebreak | wronglane	motorcycle is changing the lane in a wrong way.

Figure 6.4 - Shape of Input Text and Target Text for NLP Text Creator Model

6.6 Setup Training Process

6.6.1 Training Process for Action Recognition

In MMAction2 framework, the configuration script must be submitted to initiate the training process. This script contains a set of information describes under the following sub points.

1. Model Architecture – Type of model must be selected from the available models in the framework or a customize script. It requires image size of the dataset, number of input channels, and the pretrained weights if it is available.
2. Dataset Settings – Both training and validation locations along the type of dataset must be provided. According to the dataset we use, it requires the dataset type of “Raw Frame Dataset”.

3. Pipeline Configuration – Training, Validation, and Testing pipeline should be configured. Then the configured pipelines must be injected to the data pipeline which sets the appropriate dataset locations with number of classes.
4. Optimizer – Details of the optimizer should be defined.
5. Checkpoint location, Initial Learning rate, total number of epochs to be used for training are then defined to complete the script information.

The sample of Configuration file is shown in Figure 6.5.

```
_base_ = ['/home/ubuntu/harindu/f02-github_repos/meteor/mmaction2/configs/_base_/default_runtime.py']

# model settings
model = dict(
    type='Recognizer3D',
    backbone=dict(
        type='TimeSformer',
        pretrained=  # noqa: E251
        'https://download.openmmlab.com/mmaction/recognition/timesformer/vit_base_patch16_224.pth',  # noqa: E501
        num_frames=8,
        img_size=224,
        patch_size=16,
        embed_dims=768,
        in_channels=3,
        dropout_ratio=0.,
        transformer_layers=None,
        attention_type='divided_space_time',
        norm_cfg=dict(type='LN', eps=1e-6)),
    cls_head=dict(
        type='TimeSformerHead', num_classes=18, in_channels=768, multi_class=True),
    # model training and testing settings
    train_cfg=None,
    test_cfg=dict(average_clips='prob'))

# dataset settings
dataset_type = 'RawframeDataset'
data_root = '/home/ubuntu/harindu/datasets/meteor/New_Scenes' #'data/kinetics400/rawframes_train'
data_root_val = '/home/ubuntu/harindu/datasets/meteor/New_Scenes' #'data/kinetics400/rawframes_val'
ann_file_train = '/home/ubuntu/harindu/datasets/meteor/Annotations/train_new_annotations.txt' #'data/kinetics400/kinetics400_train_list_rawframes.txt'
ann_file_val = '/home/ubuntu/harindu/datasets/meteor/Annotations/val_new_annotations.txt' #'data/kinetics400/kinetics400_val_list_rawframes.txt'
ann_file_test = '/home/ubuntu/harindu/datasets/meteor/Annotations/val_new_annotations.txt' #'data/kinetics400/kinetics400_val_list_rawframes.txt'

img_norm_cfg = dict(
    mean=[127.5, 127.5, 127.5], std=[127.5, 127.5, 127.5], to_bgr=False)

train_pipeline = [
    dict(type='SampleFrames', clip_len=8, frame_interval=32, num_clips=1),
    dict(type='RawFrameDecode'),
    dict(type='RandomRescale', scale_range=(256, 320)),
    dict(type='RandomCrop', size=224),
    dict(type='Flip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='Collect', keys=['imgs', 'label'], meta_keys=[]),
    dict(type='ToTensor', keys=['imgs', 'label'])
]
```

Figure 6.5 - Sample configuration file of Training in MMAction2

We choose data augmentation facilities such as random clipping of range between 256 to 320 image sizes, random cropping at pixels of 224, flipping with 0.5 probability to the dataset, and normalization to increase the training process.

For Optimizer, we use Stochastic Gradient Descent (SGD) Algorithm as per the Optimization algorithm since it is commonly used in research field. SGD is used with momentum value 0.9 and the weight decaying factor of $1x10^{-4}$.

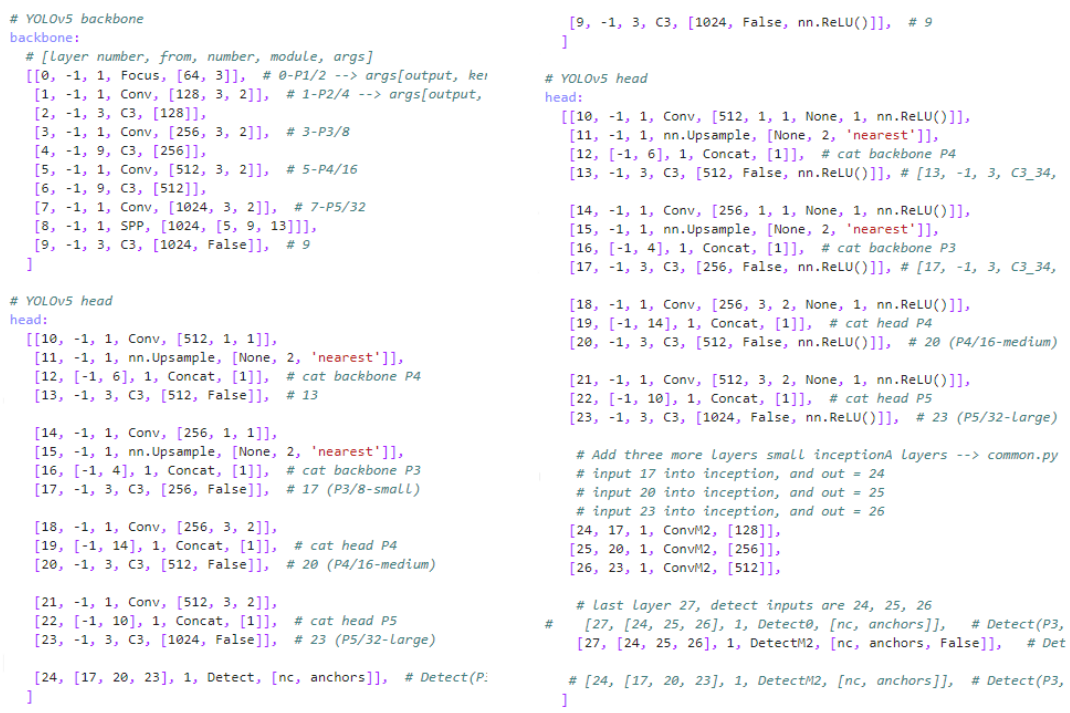
For Learning rate adjustment, we use Step Policy Algorithm at epochs 20, 50, 80, and 130 with total number of epochs of 140.

The results of the training process will be discussed under Evaluation chapter and the Appendix I shows how to inferencing the Video Recognition module.

6.6.2 Training Process for Object Detection

In Ultralytics YoloV5 framework, Model configuration file is submitted separately, and other training parameters must be provided at the initialization stage which is at the submit process to the server through Linux terminal.

For Model architecture, we select the available simple architecture from the framework, namely “YoloV5-Small”. However, we modify the last layer of the model to isolate from convolution operations. This modification enables to change the number of convolutions required at the last layer (Appendix II). Figure 6.6 shows the original model architecture and the modified model architecture at left and right respectively.



```
# YOLOv5 backbone
backbone:
  # [layer number, from, number, module, args]
  [[0, -1, 1, Focus, [64, 3]],  # 0-P1/2 --> args[output, ke
   [1, -1, 1, Conv, [128, 3, 2]],  # 1-P2/4 --> args[output,
   [2, -1, 3, C3, [128]],
   [3, -1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [4, -1, 9, C3, [256]],
   [5, -1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [6, -1, 9, C3, [512]],
   [7, -1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [8, -1, 1, SPP, [1024, [5, 9, 13]]],
   [9, -1, 3, C3, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[10, -1, 1, Conv, [512, 1, 1]],
   [11, -1, 1, nn.Upsample, [None, 2, 'nearest']],
   [12, [-1, 6], 1, Concat, [1]],  # cat backbone P4
   [13, -1, 3, C3, [512, False]],  # 13

   [14, -1, 1, Conv, [256, 1, 1]],
   [15, -1, 1, nn.Upsample, [None, 2, 'nearest']],
   [16, [-1, 4], 1, Concat, [1]],  # cat backbone P3
   [17, -1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [18, -1, 1, Conv, [256, 3, 2]],
   [19, [-1, 14], 1, Concat, [1]],  # cat head P4
   [20, -1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [21, -1, 1, Conv, [512, 3, 2]],
   [22, [-1, 10], 1, Concat, [1]],  # cat head P5
   [23, -1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [24, [17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P:
  ]
```

```
   [9, -1, 3, C3, [1024, False, nn.ReLU()]],  # 9
  ]

# YOLOv5 head
head:
  [[10, -1, 1, Conv, [512, 1, 1, None, 1, nn.ReLU()]],
   [11, -1, 1, nn.Upsample, [None, 2, 'nearest']],
   [12, [-1, 6], 1, Concat, [1]],  # cat backbone P4
   [13, -1, 3, C3, [512, False, nn.ReLU()]], # [13, -1, 3, C3_34,

   [14, -1, 1, Conv, [256, 1, 1, None, 1, nn.ReLU()]],
   [15, -1, 1, nn.Upsample, [None, 2, 'nearest']],
   [16, [-1, 4], 1, Concat, [1]],  # cat backbone P3
   [17, -1, 3, C3, [256, False, nn.ReLU()]], # [17, -1, 3, C3_34,

   [18, -1, 1, Conv, [256, 3, 2, None, 1, nn.ReLU()]],
   [19, [-1, 14], 1, Concat, [1]],  # cat head P4
   [20, -1, 3, C3, [512, False, nn.ReLU()]],  # 20 (P4/16-medium)

   [21, -1, 1, Conv, [512, 3, 2, None, 1, nn.ReLU()]],
   [22, [-1, 10], 1, Concat, [1]],  # cat head P5
   [23, -1, 3, C3, [1024, False, nn.ReLU()]],  # 23 (P5/32-large)

    # Add three more layers small inceptionA layers --> common.py
    # input 17 into inception, and out = 24
    # input 20 into inception, and out = 25
    # input 23 into inception, and out = 26
   [24, 17, 1, ConvM2, [128]],
   [25, 20, 1, ConvM2, [256]],
   [26, 23, 1, ConvM2, [512]],

    # last layer 27, detect inputs are 24, 25, 26
#    [27, [24, 25, 26], 1, Detect0, [nc, anchors]],   # Detect(P3,
    [27, [24, 25, 26], 1, DetectM2, [nc, anchors, False]],   # Det

   # [24, [17, 20, 23], 1, DetectM2, [nc, anchors]],  # Detect(P3,
  ]
```

Figure 6.6 - Comparison of YoloV5-Small original model (left) with the Modified YoloV5-Small (right)

We train the model at two phases, first phase with the ImageNet Dataset, only for backbone, and the second and final phase with both METEOR and COCO Datasets. For the first phase, we train the model for 150 epochs with initial learning rate of $1x10^{-3}$, SGD optimizer with weight decaying of $1x10^{-4}$ and momentum of 0.9333, batch size of 128 frames per batch, and image size of 640 width and 640 heights. Then the pretrained model is used for the second phase of training for 100 epochs with initial learning rate of $1x10^{-4}$, same optimizer configuration, 32 batch size, and the same image size which is 640x640.

The results of the Object Detection training process will be delivered in the Evaluation chapter.

6.6.3 Training Process for NLP Text Creator

In Chapter 3, Technology used, Model T5 is selected to get text output for given keywords. Firstly, we decided to train T5 model for a large dataset, WebNLG corpus, which consists a few domains outside the scope in this research.

For WebNLG dataset corpus, it is required to preprocess the data from RDF-triples form to sentences with main keywords separate with a pipe symbol. Appendix I shows the preprocessing part for NLP Text Creator. For training-validating ratio, 60:40 is selected since the dataset is too large which is more than 350k sentences in the corpus. When it comes to select Tokenizer, T5-based tokenizer has been selected with pretrained weights. For optimizer process, AdaFactor optimizer is selected, as per the experiment done through the paper [21]. The initial parametric values in the optimizer are preserved since that the claimed results have been taken with initial values. We run the training process for 100 epochs and record the loss against steps, which is 100 times of epochs. Figure 6.7 depicts the results.

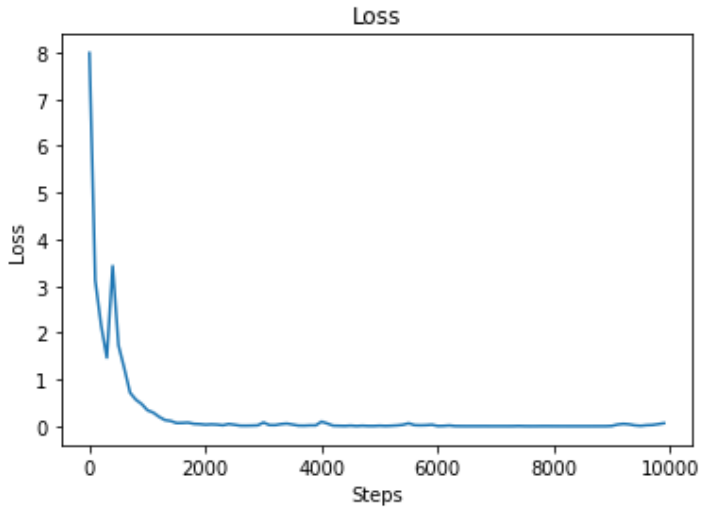


Figure 6.7- Training loss against steps in WebNLG based training

Once the model is trained on WebNLG corpus, then trained model has been set as pretrained model for next training phase with our custom created dataset. For this phase, we set 50 epochs since we only require letting model learn the missing information from the dataset corpus. The same optimizer with original parametric values has been used. Figure 6.8 shows the results of loss against steps.

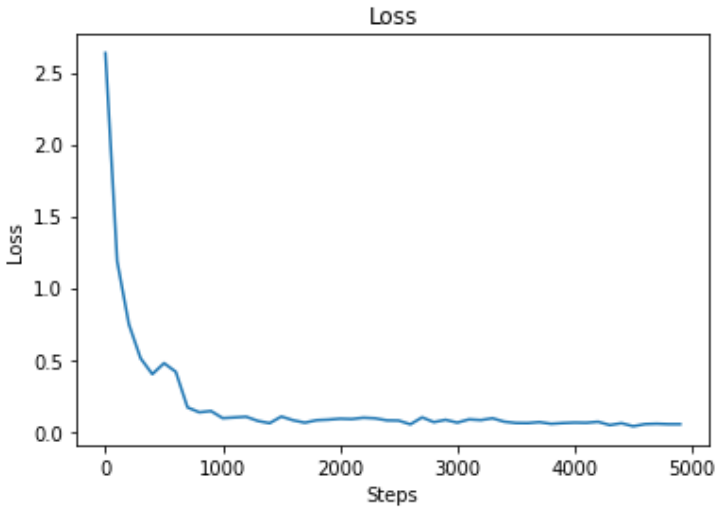


Figure 6.8 - Training results with custom created dataset

The final trained model is saved as '.bin' format for next stage in the process. (Appendix III)

6.7 Rule-based Algorithm

Our goal is to provide a simple yet powerful approach which can stand with the available methods, in real-time application perspective. Using a rule-based method for controlling outputs from each model at three stages, namely Video Recognition, Object Detection and Text Generation, empowers a non-ML approach. (Appendix IV)

Rule-based pseudocode is shown below.

---

procedure CLIP_GENERATOR (*input_video*)

    if **time_length**(input_video) is greater than 120 seconds:

        return splits of input_video with 120 seconds

    return input_video

procedure SENTENCE_GENERATOR (*action, main_object, **objects*)

$$input \leftarrow action \cup object_{main} \cup \bigcup_{i=0}^{len(objects)} object$$

return $Model\_NLP(input)$

6.8 Summary

In this chapter, the focus is to illuminate a complete implementation of the design in code base level, perpetration of datasets, neural network model configuration and training aspect, and merging the segments together. Through the next chapter, we describe how the implemented system is evaluated to achieve the desired expectations.

Chapter 7

EVALUATION

7.1 Introduction

In Chapter 6, the detailed implementation is discussed with the Technology Used described in Chapter 3. The main goal of this chapter, Evaluation, is to compare the SiRuML-TVS method among the existing state-of-the-art (STOA) methods. The detailed comparison will be conducted in an empirical evaluation for each model and for the entire system. Finally, we deliver both qualitative and quantitative results to elaborate our system's performance. Through this chapter, evaluation strategy, experiment setup for SiRuML-TVS, and the use case of this system are thoroughly delivered.

7.2 Evaluation Strategy

Evaluation strategy of this research is divided into two-fold, evaluation at training phase and the overall evaluation. A depth discussion on these two evaluations is discussed under two sub sections.

7.2.1 Evaluation at Training Phase

At the training phase, three ML modules need to be evaluated for different datasets and under different evaluation criterions. However, for each model training, starting from Action Recognition to NLP Text Generation through Object Detection, 30% of the total dataset has been allocated to validate model accuracy/loss at the end of each epoch. Since there are three independent modules used to develop the system, three different evaluation criterions must be required.

7.2.2 Overall Evaluation to the System

7.2.2.1 Create Evaluation Dataset

There is no standalone method for evaluating three independent models with a single evaluation metric. However, inspiration on evaluation method of T5 NLP model, we create a custom sample validation dataset which consists of 100 short video clips with each of clip is 1 minutes to 2 minutes long and annotated each clip with a single sentence that can be described the event/action with necessary details. Each video and the text file corresponding to outcome of that video have been saved as the same name for '.mp4' and '.txt' extensions. Figures 7.1 and 7.2 show the folder structure of custom validation dataset and outcome of a video, respectively.

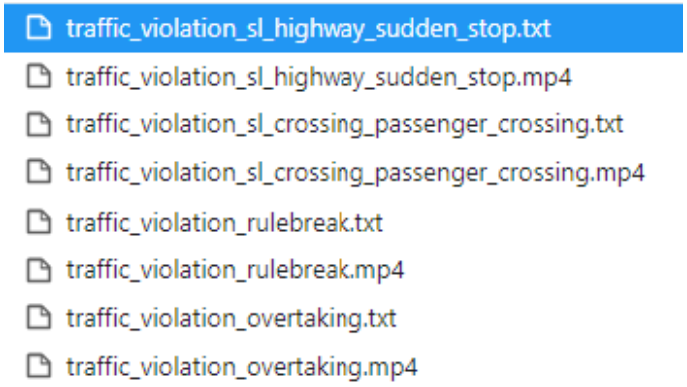


Figure 7.1- Structure of Videos and Text files of custom validation dataset

Figure 7.2 - Sample frames of a video and corresponding outcome as a sentence

7.2.2.2 Evaluation against the Custom Dataset

For this purpose, a Python script is used to get the video clips as input and provide the evaluated results against original contexts. For the final evaluation, we follow BLEU [42] criterion since the most NLP related research works use this metric as a norm. In the BLEU score evaluation, the quality of the predicted outcome is compared against

the original outcome that should be provided. After running through all the input videos, it calculates BLEU value as per the final evaluated value.

The overall evaluation is implicitly based on the evaluation of each three independent models. If one of the components does not achieve a good result, the entire system will suffer. It is worth to overview of each metrics under preceding sub sections.

7.2.3 Model Evaluation

7.2.3.1 mAP

For Object Detection task, mean average precision (mAP) is used with MS-COCO [39] dataset and METEOR [40] dataset. This metric is popular to measure the performance of models in object detection tasks and information retrieval tasks. The mAP value is calculated by using the equation (2) where it inherits from equation (1).

$$AP = \frac{1}{GTP}\sum_{k}^{n} P@k \times rel@k \quad (1)$$

In equation (1), GTP refers the total number of positive ground truths, n refers the total number of labels, $P@k$ represents the precision of k instance, and $rel@k$ refers the relevance function.

$$mAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \quad (2)$$

In Equation (2), Q represents the number of queries in the dataset and $AP(q)$ stands for Average Precision (AP) for the given query q.

The highest mAP value provides from the model is the best model for given validation dataset. Our Object Detection model, YOLO-V5, and Action Recognition model, TimeSformer, are evaluated from this metric.

7.2.3.2 BLEU

For NLP Text Generation which is T5 model, and the entire model evaluation, BLEU [42] metric is used. BLEU is calculated using a set of $ngram$ modified precisions depict in equation (3).

$$BLEU = BP.\, e^{\sum_{n=1}^{N} w_n \log p_n} \quad (3)$$

In equation (3), p_n is referred as modified precision for $ngram$. Base of the logarithm is exponential while w_n is the weight between 0 and 1. BP stands for brevity penalty to penalize the short translation by the machine/model. BP is defined in equation (4).

$$BP = \begin{cases} 1, & c > r \\ e^{\left(1-\frac{r}{c}\right)}, & c \leq r \end{cases} \quad (4)$$

In this equation, c is number of $ngram$ in dataset, and r stands for the best match up for each candidate sentence in the corpus.

The upcoming section elaborates the experiment setup is conducted to evaluate the SiRuML-TVS model over the defined metrics.

7.3 Experiment Setup for SiRuML-TVS

Main experiment setup for the Text-based Video Summarization (TVS) can be described with the preceding steps.

1. Obtain traffic related dashcam footages from YouTube. We only consider 100 of such YouTube Videos and the time frame is not considered.
2. Get relevant caption/sentence for each video from unknown third party.
3. Input each video clip into the TVS system.
4. Generated output sentence is then subjected to Evaluation criteria.
5. Comparison among different NLP related Video Summarization.

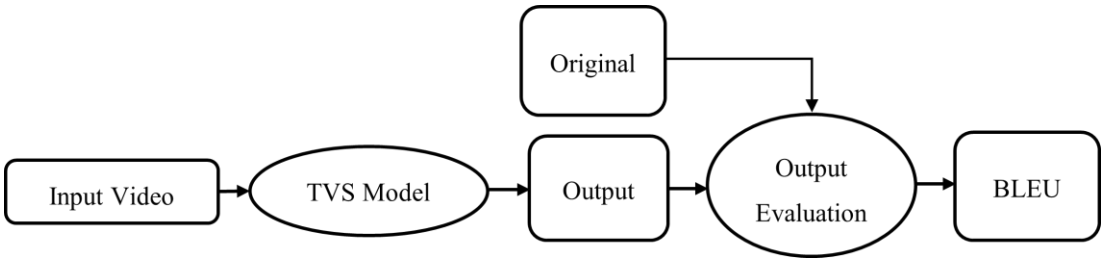


Figure 7.3 - Experiment Setup for SiRuML-TVS evaluation

In addition to this, training evaluation is also discussed through the following sub section.

7.4 TVS Models Comparison

Quantitative results for three independent models have been provided through tables 7.1, 7.2, and 7.3. A comparison has been done for the proposed SiRuML-TVS system against the STOA methods. For the comparison, we only use the datasets described in Section 6.5.

Table 7.1 - Quantitative Comparison of Object Detection modules

Model	**mAP 0.50 on COCO Dataset**	**Image Size**
SwinV2-G [43]	**63.1**	1536x1536
Florence-CoSwin-H [44]	62.4	384x384
YOLO-V5s-Mod [38]	**61.9**	640x640
Soft Teacher + Swin-L [45]	61.3	384x384
YOLOv4-P7 with TTA [46]	55.8	640x640

From the results of Table 7.1, except two YOLO variants, all other object detection models are based on Transformer mechanism which is the current STOA based technology. From these results, our conjecture that is training the model on a large dataset first and then training on the relevant dataset explained in Section 6.6.2, has been proved with respect to the YOLOv4 [46] model. Integration of CNN based Object Detection with comparatively good mAP score is preferred since the most of hardware platforms are yet supporting CNN based architectures.

Table 7.2 - Quantitative Comparison of Video Recognition modules

Model	Pretrain	METEOR Training Time (hours)	METEOR Accuracy	K400 Accuracy (From Paper)	Inference TFLOPs
ir-CSN-152 [47]	IG-65M	264	**89.3**	82.6	0.967
TimeSformer [20]	ImageNet-21K	144	87.01	78.0	0.59
TPN-R101 [48]	ImageNet-21K	192	86.14	78.9	-

According to the Table 7.2, Channel Separated Convolutional Network [47] is much better than TimeSformer [20] method. However, when it comes to implement the model in real-world application such as in embedded devices, TimeSformer models are much preffered. It requires only a few Gigabytes of GPU memory than other models. Therefore, the model can be integrated into a multiprocessing thread. All these models are trained with 140 epochs as described in Section 6.6.1.

Table 7.3 - Qualitative Results for NLP Model

Model	Pretrain	BLEU on WebNLG	Training Time (hours)
Control Prefixes [49]	-	55.41	-
T5 [21]	C4 [21]	51.71	2

In Table 7.3, T5 model is evaluated on WebNLG dataset and results as described in Section 6.6.3. Control Prefixes method is also based on T5-large model and the latest STOA.

Further to select which models' combination to be used for TVS system, we empirically conduct an evaluation process for such combinations. Table 7.4 provides the different combined models' results against the custom dataset described in Section 7.2.2. Based on evaluation results of the experiment, our intended technology base that is the combination of TimeSformer, YOLO-V5s, and T5 provides the best BLEU-1 and BLEU-4. According to these results, Object detection model is considerably supported to T5 model for getting more objects into the sentences that can enrich the meaning of the statement.

Table 7.4 - Quantitative Evaluation on different models

Dataset	TVS Configuration	BLEU-1	BLEU-4
Custom Dataset	**TimeSformer + YOLO-V5s + T5**	**23.83**	**8.36**
	TPN-R101 + YOLO-V5s + T5	22.09	7.64
	TPN-R101 + YOLO-V4 + T5	22.02	7.72
	TimeSformer + YOLO-V4 + T5	20.94	6.87

In addition to the quantitative evaluation, a qualitative evaluation is also conducted for further clarification of the system. For this evaluation, a random dashcam footage from a vehicle owner is taken with owner's description to the video. Then the owner has been asked to align each sentence into time stamp without explicitly saying about the time limit of 2 minutes. Then the video is used in the small-clip generator method and cross check the split video clips have a sentence or not, according to the time stamp. If a clip does not contain at least a single sentence, then we decide to discard that input. Once the result has been generated from the system, same vehicle owner is requested to validate the output. Similarly, we conduct the same for a couple of videos to ensure the model consistence and the performance to claim quantitatively.

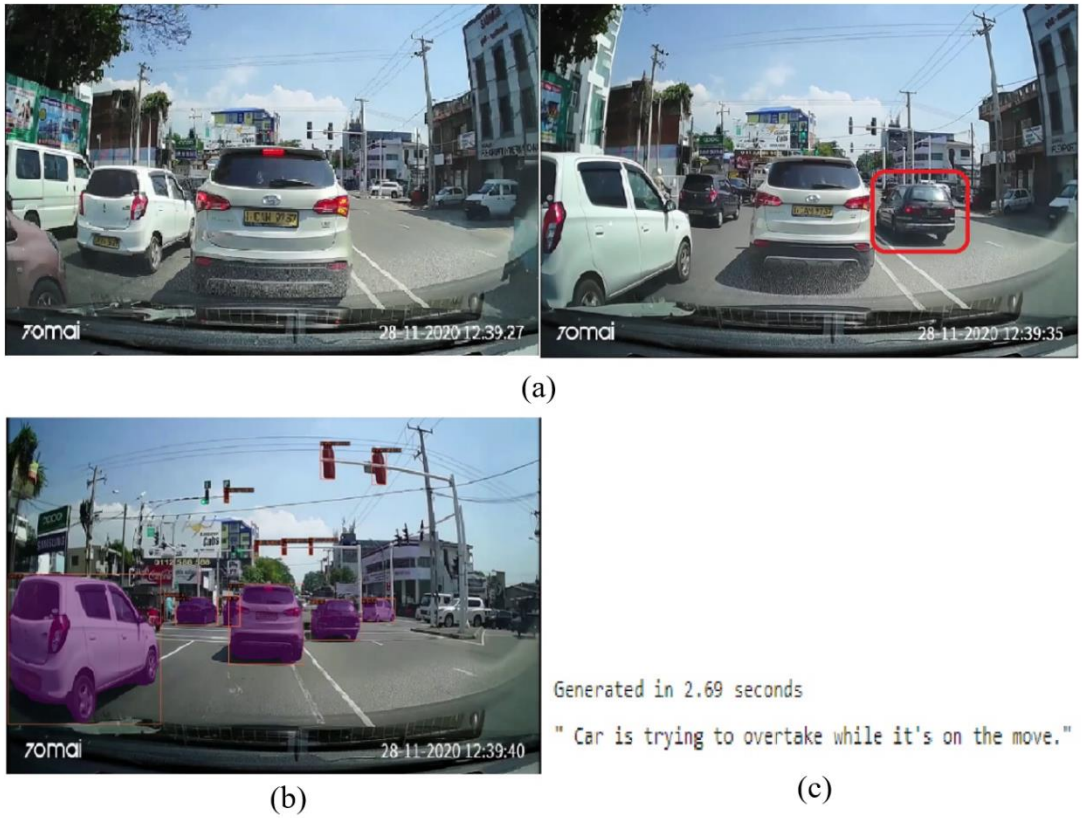


Figure 7.4 - Quantitative Results of the TVS system. In part (a), original video clip is shown. Black color car (highlighted by red box) is overtaking the white car. In part (b), objects of the video has been shown. In part (c), the final result is provided.

7.5 Use Case of the SiRuML-TVS System

Text-based Video Summarization is heavily interested in the video synopsis of movie database, summarization of lecture series to simple text document, control algorithms in automated vehicle industry, and for visually impaired people. With the current developments and technologies, TVS has a good potential to growth in lot of fields.

However, when it comes to implementation in real world device, it is still challenging since most of the applications are empowered by an embedded device which has a limited resource space for computation and low power usage. Selection of models to these devices bring a complicated process. To address these vulnerabilities, our SiRuML-TVS fills that gap with a simple ML based approach which consists of the hardware-friendly ML models. Through the research, we prove that our approach and

application is quite enough to bring a real-time application in both quantitatively and qualitatively.

Since the SiRuML-TVS system is inherently made for reconfigurable models, it is easy to replace any model out of three independent models to improve the system performance and system reliability along the accuracy without hesitation.

7.6 Summary

In this chapter, the empirical evaluation has been carried to understand the application performance and results, and individual evaluation in each model in the system. Through this chapter, what is the evaluation strategy used for SiRuML-TVS system, both qualitative and quantitative results of the system, and finally the use case of the application are discussed thoroughly. Based on the evaluated results, the overall conclusion can be derived and pointed out in the next chapter.

Chapter 8

CONCLUSION AND FURTHER WORK

8.1 Introduction

In the previous chapter, a detailed evaluation procedure has been carried to depict the SiRuML-TVS pipeline's performance in both quantitative and qualitative methods. Starting from the first chapter till the previous chapter, the hypothesis of the research project is subjected to enrich with a simple system. Through this chapter, we conclude the dissertation by highlighting the achievements of the objectives described in chapter 1, appeal to overall conclusions and further work to be delivered in this area with the support of the details in evaluation chapter and latest research works.

8.2 Conclusion

8.2.1 Achievements of Project Objectives

The overall project is aligned on achieving the objectives defined through the Chapter 1, Introduction. Following the proper project process is helped to achieve the expected objectives accordingly. This section has been allocated to discuss the achievements of the objectives thoroughly.

With a in depth, breadth and critical literature review, early gestation and the latest developments in Video Summarization and NLP based Video Summarization has been identified. Through the same literature review, several drawbacks in the existing technologies and the remedies attempted are understood wisely. By emphasizing the current trends in VS, one of the replacements for the existing technologies is considered to lay down as a new resolution to the TVS which is Transformer based Action Recognition. Most of the VS methods are developed on top of CNN approaches and a few of them used the latest trend. However, the alignment of the research is based upon usage of simple machine learning for each phase. With that assumption, we suggest three independent ML models and discuss the base technologies of them.

All the SiRuML-TVS techniques are implemented into an end-to-end pipeline. As indicated in Design chapter and Implementation chapter, three main models are trained on the corresponding datasets and saved the best model for integration into the final pipeline. Through the NLP Text Generation part, we create a custom dataset on top of the WebNLG to incorporate the scope of the application which enriches the information of the area of interest. For the entire pipeline evaluation, we create another few samples, around 100 of video clips with relevant description to each video clip. All these datasets are created manually.

After the implementation stage, our next target is to evaluate the full system against the newly created dataset. For internal model evaluation, two benchmark indexes are used and for the final evaluation, NLP standard index BLEU has been used. Moreover, combinations of different models at each stage are also shown to explain the best combination.

According to the last objective defined under the Aims and Objectives, with the random video footages, we attain the responses on our system which are positive and pleasing. Finally, at the time of the dissertation submission, one conference paper is being drafted with the study and expecting to submit to a good conference.

8.2.2 Overall Conclusion

To overcome the issues such as complexity of the models and dataset subjectivity, our hypothesis of by using the simple ML models as a hierarchical pipeline is thoroughly examined. Most of the STOA research works propose Transformer based networks with the inspiration of NLP field. However, both space and temporal information should be captured to get an in-depth overview of the scene. CNN based frameworks are shown the best performance to extract over the spatial related dimension. Hence, our appetite of mixing both Transformer and CNN networks to extract the overall spatial and temporal information is proved by both quantitative and qualitative results.

With the evaluation results and hierarchical outcomes, our hypothesis of the simple ML models with a rule-based system for text-based video summarization architecture is clearly certified.

8.3 Limitations and Further Works

Even though both Transformer based mechanism and CNN based mechanism are used in the TVS pipeline, the same results can be achieved by using Transformer only methods which is the latest trend when the dissertation is being written. It seems like in-depth analysis on Transformer based networks should be done to understand the potential growth of this technique.

Due to the limited time, a few models are evaluated which is not adequate to improve our system's performances. We need to get a proper benchmark dataset for evaluate our model against the rest of the existing models since the created custom dataset has only 100 of data points which might not be viable to conclude a proper outcome of the system. Creation of benchmark dataset takes more time however it is useful for future studies.

According to the real application, we try to deploy the pipeline into an embedded device such as Raspberry-Pi and Nvidia Jetson V2 board to depict our claim which is also debited into future work. Finally, we try to deploy the system as an automated vehicle controlling method which will maneuver the vehicle with respect to the environmental behavior.

8.4 Summary

By bringing to the end of dissertation, this chapter claims to which level of coverage the objectives are achieved, the overall conclusion of the research and the further works to be done.

REFERENCES

[1] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras, "Video Summarization Using Deep Neural Networks: A Survey," *arXiv:2101.06072 [cs]*, Jan. 2021, Accessed: Jul. 25, 2021. [Online]. Available: http://arxiv.org/abs/2101.06072

[2] M. Barbieri, L. Agnihotri, and N. Dimitrova, "Video summarization: methods and landscape," Orlando, FL, Nov. 2003, pp. 1–13. doi: 10.1117/12.515733.

[3] K. Zhou, Y. Qiao, and T. Xiang, "Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity-Representativeness Reward," *arXiv:1801.00054 [cs]*, Feb. 2018, Accessed: Jul. 19, 2021. [Online]. Available: http://arxiv.org/abs/1801.00054

[4] G. Liang, Y. Lv, S. Li, S. Zhang, and Y. Zhang, "Unsupervised Video Summarization with a Convolutional Attentive Adversarial Network," *arXiv:2105.11131 [cs]*, May 2021, Accessed: Jul. 19, 2021. [Online]. Available: http://arxiv.org/abs/2105.11131

[5] M. A. Samsuden, N. M. Diah, and N. A. Rahman, "A Review Paper on Implementing Reinforcement Learning Technique in Optimising Games Performance," in *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, Shah Alam, Malaysia, Oct. 2019, pp. 258–263. doi: 10.1109/ICSEngT.2019.8906400.

[6] Y. Zhang, M. Kampffmeyer, X. Zhao, and M. Tan, "Deep Reinforcement Learning for Query-Conditioned Video Summarization," *Applied Sciences*, vol. 9, no. 4, p. 750, Feb. 2019, doi: 10.3390/app9040750.

[7] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Y. Wang, "Video Captioning via Hierarchical Reinforcement Learning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018, pp. 4213–4222. doi: 10.1109/CVPR.2018.00443.

[8] Q. Huang, Z. Gan, A. Celikyilmaz, D. Wu, J. Wang, and X. He, "Hierarchically Structured Reinforcement Learning for Topically Coherent Visual Story Generation," *arXiv:1805.08191 [cs]*, Jan. 2019, Accessed: Jul. 25, 2021. [Online]. Available: http://arxiv.org/abs/1805.08191

[9] J.-H. Huang and M. Worring, "Query-controllable Video Summarization," *arXiv:2004.03661 [cs]*, Apr. 2020, Accessed: Jul. 19, 2021. [Online]. Available: http://arxiv.org/abs/2004.03661

[10] H. Wei, B. Ni, Y. Yan, H. Yu, and X. Yang, "Video Summarization via Semantic Attended Networks," p. 8.

[11] K. Zhang, K. Grauman, and F. Sha, "Retrospective Encoders for Video Summarization," in *Computer Vision – ECCV 2018*, vol. 11212, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 391–408. doi: 10.1007/978-3-030-01237-3_24.

[12] A. Kanehira, L. Van Gool, Y. Ushiku, and T. Harada, "Viewpoint-Aware Video Summarization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, Jun. 2018, pp. 7435–7444. doi: 10.1109/CVPR.2018.00776.

[13] J. Wang, W. Wang, Z. Wang, L. Wang, D. Feng, and T. Tan, "Stacked Memory Network for Video Summarization," in *Proceedings of the 27th ACM*

International Conference on Multimedia, Nice France, Oct. 2019, pp. 836–844. doi: 10.1145/3343031.3350992.

[14] Z. Ji, K. Xiong, Y. Pang, and X. Li, "Video Summarization With Attention-Based Encoder–Decoder Networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 6, pp. 1709–1717, Jun. 2020, doi: 10.1109/TCSVT.2019.2904996.

[15] D. He, X. Zhao, J. Huang, F. Li, X. Liu, and S. Wen, "Read, Watch, and Move: Reinforcement Learning for Temporally Grounding Natural Language Descriptions in Videos," *AAAI*, vol. 33, pp. 8393–8400, Jul. 2019, doi: 10.1609/aaai.v33i01.33018393.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, Accessed: Feb. 27, 2022. [Online]. Available: http://arxiv.org/abs/1810.04805

[17] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," p. 12.

[18] X. Shang, Z. Yuan, A. Wang, and C. Wang, "Multimodal Video Summarization via Time-Aware Transformers," in *Proceedings of the 29th ACM International Conference on Multimedia*, Virtual Event China, Oct. 2021, pp. 1756–1765. doi: 10.1145/3474085.3475321.

[19] H. Li, J. Zhu, C. Ma, J. Zhang, and C. Zong, "Multi-modal Summarization for Asynchronous Collection of Text, Image, Audio and Video," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sep. 2017, pp. 1092–1102. doi: 10.18653/v1/D17-1114.

[20] G. Bertasius, H. Wang, and L. Torresani, "Is Space-Time Attention All You Need for Video Understanding?," *arXiv:2102.05095 [cs]*, Jun. 2021, Accessed: Jan. 06, 2022. [Online]. Available: http://arxiv.org/abs/2102.05095

[21] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *arXiv:1910.10683 [cs, stat]*, Jul. 2020, Accessed: Feb. 26, 2022. [Online]. Available: http://arxiv.org/abs/1910.10683

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, vol. 25. Accessed: Feb. 26, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[23] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[24] I. J. Goodfellow *et al.*, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, Jun. 2014, Accessed: Mar. 11, 2022. [Online]. Available: http://arxiv.org/abs/1406.2661

[25] Yong Rui, T. S. Huang, and S. Mehrotra, "Exploring video structure beyond the shots," in *Proceedings. IEEE International Conference on Multimedia Computing and Systems (Cat. No.98TB100241)*, Austin, TX, USA, 1998, pp. 237–240. doi: 10.1109/MMCS.1998.693648.

[26] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video Summarization with Long Short-term Memory," *arXiv:1605.08110 [cs]*, Jul. 2016, Accessed: Feb. 25, 2022. [Online]. Available: http://arxiv.org/abs/1605.08110

[27] J. Lee and S. Abu-El-Haija, “Large-Scale Content-Only Video Recommendation,” in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Venice, Oct. 2017, pp. 987–995. doi: 10.1109/ICCVW.2017.121.

[28] C. C. Park and G. Kim, “Expressing an Image Stream with a Sequence of Natural Sentences,” in *Advances in Neural Information Processing Systems*, 2015, vol. 28. Accessed: Feb. 25, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/17e62166fc8586dfa4d1bc0e1742c08b-Abstract.html

[29] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, “Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation,” in *Advances in Neural Information Processing Systems*, 2016, vol. 29. Accessed: Feb. 25, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2016/hash/f442d33fa06832082290ad8544a8da27-Abstract.html

[30] Z. Li and L. Yang, “Weakly Supervised Deep Reinforcement Learning for Video Summarization With Semantically Meaningful Reward,” in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, Waikoloa, HI, USA, Jan. 2021, pp. 3238–3246. doi: 10.1109/WACV48630.2021.00328.

[31] X. He *et al.*, “Unsupervised Video Summarization with Attentive Conditional Generative Adversarial Networks,” in *Proceedings of the 27th ACM International Conference on Multimedia*, Nice France, Oct. 2019, pp. 2296–2304. doi: 10.1145/3343031.3351056.

[32] J. Gao, X. Yang, Y. Zhang, and C. Xu, “Unsupervised Video Summarization via Relation-aware Assignment Learning,” *IEEE Trans. Multimedia*, pp. 1–1, 2020, doi: 10.1109/TMM.2020.3021980.

[33] Y. Jung, D. Cho, D. Kim, S. Woo, and I. S. Kweon, “Discriminative Feature Learning for Unsupervised Video Summarization,” *arXiv:1811.09791 [cs]*, Nov. 2018, Accessed: Jul. 19, 2021. [Online]. Available: http://arxiv.org/abs/1811.09791

[34] K. Zhou, T. Xiang, and A. Cavallaro, “Video Summarisation by Classification with Deep Reinforcement Learning,” *arXiv:1807.03089 [cs]*, Sep. 2018, Accessed: Jul. 25, 2021. [Online]. Available: http://arxiv.org/abs/1807.03089

[35] H. Li, J. Zhu, C. Ma, J. Zhang, and C. Zong, “Read, Watch, Listen, and Summarize: Multi-Modal Summarization for Asynchronous Text, Image, Audio and Video,” *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 996–1009, May 2019, doi: 10.1109/TKDE.2018.2848260.

[36] A. Vaswani *et al.*, “Attention Is All You Need,” *arXiv:1706.03762 [cs]*, Dec. 2017, Accessed: Feb. 26, 2022. [Online]. Available: http://arxiv.org/abs/1706.03762

[37] L. Yuan, F. E. Tay, P. Li, L. Zhou, and J. Feng, “Cycle-SUM: Cycle-consistent Adversarial LSTM Networks for Unsupervised Video Summarization,” *arXiv:1904.08265 [cs]*, Apr. 2019, Accessed: Jul. 19, 2021. [Online]. Available: http://arxiv.org/abs/1904.08265

[38] G. Jocher *et al., ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations*. Zenodo, 2021. doi: 10.5281/zenodo.4679653.

[39] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *arXiv:1405.0312 [cs]*, Feb. 2015, Accessed: Mar. 03, 2022. [Online]. Available: http://arxiv.org/abs/1405.0312
[40] R. Chandra *et al.*, “METEOR: A Massive Dense & Heterogeneous Behavior Dataset for Autonomous Driving,” *arXiv:2109.07648 [cs]*, Sep. 2021, Accessed: Jan. 06, 2022. [Online]. Available: http://arxiv.org/abs/2109.07648
[41] “Documentation - WebNLG Challenges.” https://webnlg-challenge.loria.fr/docs/ (accessed Mar. 02, 2022).
[42] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a Method for Automatic Evaluation of Machine Translation,” in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, Jul. 2002, pp. 311–318. doi: 10.3115/1073083.1073135.
[43] Z. Liu *et al.*, “Swin Transformer V2: Scaling Up Capacity and Resolution,” *arXiv:2111.09883 [cs]*, Nov. 2021, Accessed: Mar. 03, 2022. [Online]. Available: http://arxiv.org/abs/2111.09883
[44] L. Yuan *et al.*, “Florence: A New Foundation Model for Computer Vision,” *arXiv:2111.11432 [cs]*, Nov. 2021, Accessed: Mar. 03, 2022. [Online]. Available: http://arxiv.org/abs/2111.11432
[45] M. Xu *et al.*, “End-to-End Semi-Supervised Object Detection with Soft Teacher,” *arXiv:2106.09018 [cs]*, Aug. 2021, Accessed: Mar. 03, 2022. [Online]. Available: http://arxiv.org/abs/2106.09018
[46] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Scaled-YOLOv4: Scaling Cross Stage Partial Network,” *arXiv:2011.08036 [cs]*, Feb. 2021, Accessed: Mar. 03, 2022. [Online]. Available: http://arxiv.org/abs/2011.08036
[47] D. Tran, H. Wang, M. Feiszli, and L. Torresani, “Video Classification With Channel-Separated Convolutional Networks,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Seoul, Korea (South), Oct. 2019, pp. 5551–5560. doi: 10.1109/ICCV.2019.00565.
[48] C. Yang, Y. Xu, J. Shi, B. Dai, and B. Zhou, “Temporal Pyramid Network for Action Recognition,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 588–597. doi: 10.1109/CVPR42600.2020.00067.
[49] J. Clive, K. Cao, and M. Rei, “Control Prefixes for Text Generation,” *arXiv:2110.08329 [cs]*, Oct. 2021, Accessed: Mar. 04, 2022. [Online]. Available: http://arxiv.org/abs/2110.08329

APPENDIX

Appendix I: Video Recognition module for Inferencing

Video Inference

[1]:
```python
import torch

from mmaction.apis import init_recognizer, inference_recognizer
```

[2]:
```python
# config_file = 'configs/recognition/tsn/tsn_r50_video_inference_1x1x3_100e_kinetics400_rgb.py'
# config_file = 'Scripts/configs/tsn_config_scratch_more_classes.py'
# config_file = 'mmaction2/configs/recognition/tsn/tsn_r50_inference_1x1x3_100e_kinetics400_rgb.py'
config_file = 'Scripts/configs/timesformer_divST_8x32x1_100e_meteor_rgb.py'

# download the checkpoint from model zoo and put it in `checkpoints/`
# checkpoint_file = 'checkpoints/tsn_r50_1x1x3_100e_kinetics400_rgb_20200614-e508be42.pth'
# checkpoint_file = 'work_dirs/tsn_r101_1x1x5_50e_meteor_rgb/epoch_145.pth'
checkpoint_file = 'work_dirs/timesformer_divST_8x32x1_100e_meteor_rgb/best_mmit_mean_average_precision_epoch_10.pth'

# assign the desired device.
device = 'cuda:0' # or 'cpu'
device = torch.device(device)
```

[3]:
```python
 # build the model from a config file and a checkpoint file
model = init_recognizer(config_file, checkpoint_file, device=device)

# test a single video and show the result:
# video = '/home/ubuntu/harindu/datasets/meteor/Scenes/008949'
# video = '/home/ubuntu/harindu/testing-sources/traffic_violation_overtaking.mp4'
video = '/home/ubuntu/harindu/testing-sources/traffic_violation_rulebreak.mp4'
# labels = '/home/ubuntu/harindu/datasets/meteor/label_map_meteor.txt'
labels = '/home/ubuntu/harindu/datasets/meteor/label_map_meteor_more_classes.txt'
# results = inference_recognizer(model, video, labels, use_frames=True)
results = inference_recognizer(model, video)

# show the results
# labels = open('tools/data/kinetics/label_map_k400.txt').readlines()
# labels = open('/home/ubuntu/harindu/datasets/meteor/label_map_meteor.txt').readlines()
labels = open(labels).readlines()
labels = [x.strip() for x in labels]
results = [(labels[k[0]], k[1]) for k in results]

print(f'The top-5 labels with corresponding scores are:')
for result in results:
    print(f'{result[0]}: ', result[1])
```

```
Use load_from_local loader
The top-5 labels with corresponding scores are:
keep_action:  0.37378168
straightroad_scenes:  0.34774598
overtaking_scenario:  0.19553989
lanechanging_scenario:  0.04777838
intersections_scenes:  0.009364306
```

Appendix II: Change of DetectM2 class in Object Detection Module, YoloV5

```
class DetectM2(nn.Module):
    stride = None  # strides computed during build
    onnx_dynamic = False  # ONNX export parameter

    def __init__(self, nc=80, anchors=(), internal_conv=True, ch=(), inplace=True, conv=ConvM2):  # detection layer
        super().__init__()

        try:
            self.internal_conv = internal_conv
        except:
            print('Exception setting internal conv to True!')
            self.internal_conv = True
        self.nc = nc  # number of classes
        self.no = nc + 5  # number of outputs per anchor
        print(anchors)
        self.nl = len(anchors)  # number of detection layers
        self.na = len(anchors[0]) // 2  # number of anchors
        self.grid = [torch.zeros(1)] * self.nl  # init grid
        a = torch.tensor(anchors).float().view(self.nl, -1, 2)
        self.register_buffer('anchors', a)  # shape(nl,na,2)
        self.register_buffer('anchor_grid', a.clone().view(self.nl, 1, -1, 1, 1, 2))  # shape(nl,1,na,1,1,2)
        if self.internal_conv:
            self.m = nn.ModuleList(conv(x, self.no * self.na, 1) for x in ch)  # output conv
        self.inplace = inplace  # use in-place ops (e.g. slice assignment)

    def forward(self, x):
        z = []  # inference output

        for i in range(self.nl):
            if self.internal_conv:
                x[i] = self.m[i](x[i])  # conv

            bs, _, ny, nx = x[i].shape  # x(bs,255,20,20) to x(bs,3,20,20,85)
            x[i] = x[i].view(bs, self.na, self.no, ny, nx).permute(0, 1, 3, 4, 2).contiguous()

            if not self.training:  # inference
                if self.grid[i].shape[2:4] != x[i].shape[2:4] or self.onnx_dynamic:
                    self.grid[i] = self._make_grid(nx, ny).to(x[i].device)

                y = x[i].sigmoid()
                if self.inplace:
#                     print('--> inplace grid', len(self.grid), len(self.stride))
                    y[..., 0:2] = (y[..., 0:2] * 2. - 0.5 + self.grid[i]) * self.stride[i]  # xy
                    y[..., 2:4] = (y[..., 2:4] * 2) ** 2 * self.anchor_grid[i]  # wh
                else:  # for YOLOv5 on AWS Inferentia https://github.com/ultralytics/yolov5/pull/2953
#                     print('--> grid', len(self.grid), len(self.stride))
                    xy = (y[..., 0:2] * 2. - 0.5 + self.grid[i]) * self.stride[i]  # xy
                    wh = (y[..., 2:4] * 2) ** 2 * self.anchor_grid[i].view(1, self.na, 1, 1, 2)  # wh
                    y = torch.cat((xy, wh, y[..., 4:]), -1)
                z.append(y.view(bs, -1, self.no))

        return x if self.training else (torch.cat(z, 1), x)
```

Appendix III: Training Script for T5 Module

Please look at the code here to preprocess the data

[5]:

```
import urllib.request
import zipfile

url = 'https://gitlab.com/shimorina/webnlg-dataset/-/archive/master/webnlg-dataset-master.zip?path=release_v3.0/en/train'
urllib.request.urlretrieve(url, 'web.zip')
with zipfile.ZipFile('web.zip', 'r') as zip_ref:
    zip_ref.extractall('web')
import glob
import os
import re
import xml.etree.ElementTree as ET
import pandas as pd
files = glob.glob("web/webnlg-dataset-master-release_v3.0-en-train/release_v3.0/en/train/**/*.xml", recursive=True)
triple_re=re.compile('(\d)triples')
data_dct={}
for file in files:
    tree = ET.parse(file)
    root = tree.getroot()
    triples_num=int(triple_re.findall(file)[0])
    for sub_root in root:
        for ss_root in sub_root:
            strutured_master=[]
            unstructured=[]
            for entry in ss_root:
                unstructured.append(entry.text)
                strutured=[triple.text for triple in entry]
                strutured_master.extend(strutured)
            unstructured=[i for i in unstructured if i.replace('\n','').strip()!='' ]
            strutured_master=strutured_master[-triples_num:]
            strutured_master_str=(' && ').join(strutured_master)
            data_dct[strutured_master_str]=unstructured
mdata_dct={"prefix":[], "input_text":[], "target_text":[]}
for st,unst in data_dct.items():
    for i in unst:
        mdata_dct['prefix'].append('webNLG')
        mdata_dct['input_text'].append(st)
        mdata_dct['target_text'].append(i)


df=pd.DataFrame(mdata_dct)
df.to_csv('webNLG2020_train.csv')
```

Loading the pretrained model and tokenizer

[13]:

```
tokenizer = T5Tokenizer.from_pretrained('t5-base')
# model = T5ForConditionalGeneration.from_pretrained('t5-base', return_dict=True)
model = T5ForConditionalGeneration.from_pretrained('pytoch_model_webnlg.bin', return_dict=True, config='t5-base-config.json')
#moving the model to device(GPU/CPU)
model.to(dev)
```

```
(0): T5Block(
  (layer): ModuleList(
    (0): T5LayerSelfAttention(
      (SelfAttention): T5Attention(
        (q): Linear(in_features=768, out_features=768, bias=False)
        (k): Linear(in_features=768, out_features=768, bias=False)
        (v): Linear(in_features=768, out_features=768, bias=False)
        (o): Linear(in_features=768, out_features=768, bias=False)
        (relative_attention_bias): Embedding(32, 12)
      )
      (layer_norm): T5LayerNorm()
      (dropout): Dropout(p=0.1, inplace=False)
```

Initializing the Adafactor optimizer with parameter values suggested for t5

[14]:

```
optimizer = Adafactor(
    model.parameters(),
    lr=1e-3,
    eps=(1e-30, 1e-3),
    clip_threshold=1.0,
    decay_rate=-0.8,
    beta1=None,
    weight_decay=0.0,
    relative_step=False,
    scale_parameter=False,
    warmup_init=False
)
```

Training the model

[17]:

```
#Sets the module in training mode
model.train()

loss_per_10_steps=[]
for epoch in range(1,num_of_epochs+1):
    print('Running epoch: {}'.format(epoch))

    running_loss=0

    out = display(progress(1, num_of_batches+1), display_id=True)
    for i in range(num_of_batches):
        inputbatch=[]
        labelbatch=[]
        new_df=train_df[i*batch_size:i*batch_size+batch_size]
        for indx,row in new_df.iterrows():
            input = 'WebNLG: '+row['input_text']+'</s>'
            labels = row['target_text']+'</s>'
            inputbatch.append(input)
            labelbatch.append(labels)
    inputbatch=tokenizer.batch_encode_plus(inputbatch,padding=True,max_length=400,return_tensors='pt')["input_ids"]
    labelbatch=tokenizer.batch_encode_plus(labelbatch,padding=True,max_length=400,return_tensors="pt") ["input_ids"]
    inputbatch=inputbatch.to(dev)
    labelbatch=labelbatch.to(dev)

    # clear out the gradients of all Variables
    optimizer.zero_grad()

    # Forward propogation
    outputs = model(input_ids=inputbatch, labels=labelbatch)
    loss = outputs.loss
    loss_num=loss.item()
    logits = outputs.logits
    running_loss+=loss_num
    print('i --> ', i)
#     if i%10 ==0:
    loss_per_10_steps.append(running_loss) #loss_num)
    out.update(progress(loss_num,i, num_of_batches+1))

    # calculating the gradients
    loss.backward()

    #updating the params
    optimizer.step()

    running_loss=running_loss/int(num_of_batches)
    print('Epoch: {} , Running loss: {}'.format(epoch,running_loss))

```

Appendix IV: Full System of SiRuML-TVS

Summarization of Action Recognition for Vehicle Events

This script is used to demonstrate the action recognition for vehicle events and then summarize the event via textual format.

[1]:
```
%load_ext autoreload
%autoreload 2
import os
if 1:
    os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
    os.environ["CUDA_VISIBLE_DEVICES"]="1"
```

Load Dependencies

[2]:
```
import os
import cv2
import torch

from mmaction.apis import init_recognizer, inference_recognizer
from mmdet.apis import inference_detector, init_detector, show_result_pyplot

from transformers import T5Tokenizer, T5ForConditionalGeneration
import time
import warnings
warnings.filterwarnings('ignore')
```

GPU selection

[3]:
```
device = 'cuda:0' # or 'cpu'
device = torch.device(device)
```

Configs and Model files

[4]:
```
# Action
config_file_action = 'Scripts/configs/timesformer_divST_8x32x1_100e_meteor_rgb.py'
checkpoint_file_action = 'work_dirs/timesformer_divST_8x32x1_100e_meteor_rgb/best_mmit_mean_average_precision_epoch_10.pth'

# Detection
config_file_detection = 'Scripts/configs/scnet_r50_fpn_1x_coco.py'
checkpoint_file_detection = 'scnet_r50_fpn_1x_coco-c3f09857.pth'
```

Load Models

[5]:
```
# Action model
model_action = init_recognizer(config_file_action, checkpoint_file_action, device=device)
# Detection model
model_detection = init_detector(config_file_detection, checkpoint_file_detection, device=device)
# NLG model
tokenizer = T5Tokenizer.from_pretrained('t5-base')
model_nlg = T5ForConditionalGeneration.from_pretrained('T5_nlg/pytoch_model_meteor.bin', return_dict=True,config='T5_nlg/t5-base-config.json')
```

Inputs

[6]:
```
# video file
# video = '/home/ubuntu/harindu/testing-sources/traffic_violation_rulebreak.mp4'
video = '/home/ubuntu/harindu/testing-sources/traffic_violate_sl_overtaking_junction.mkv'
# video = '/home/ubuntu/harindu/testing-sources/traffic_violation_sl_crossing_passenger_crossing_short.mp4'


# video = '/home/ubuntu/harindu/testing-sources/traffic_violation_sl_highway_sudden_stop.mp4'

# Actions
labels_action = '/home/ubuntu/harindu/datasets/meteor/label_map_meteor_more_classes.txt'
```

Preprocessor

NLG

[7]:
```
def generate(text):
    model_nlg.eval()
    input_ids = tokenizer.encode("WebNLG:{} </s>".format(text), return_tensors="pt")  # Batch size 1
    # input_ids.to(dev)
    s = time.time()
    outputs = model_nlg.generate(input_ids)
    gen_text = tokenizer.decode(outputs[0]).replace('<pad>','').replace('</s>','')
    elapsed = time.time() - s
    print('Generated in {} seconds'.format(str(elapsed)[:4]))

    return gen_text
```

Image

[8]:
```
main_img_detection = ''

vidcap = cv2.VideoCapture(video)
length = int(vidcap.get(cv2.CAP_PROP_FRAME_COUNT))
cnt = 1
print(length // 2)
while True:
    ret, image_bgr = vidcap.read()
    if ret:
        image = image_bgr# image_bgr[:, :, [2, 1, 0]]
        if cnt == length // 2:
            main_img_detection = image
            break
        cnt += 1

cv2.destroyAllWindows()
vidcap.release()
```

Load Results

[9]:
```
# Action results
results_action = inference_recognizer(model_action, video)
# Detection results
results_detection = inference_detector(model_detection, main_img_detection)
```

[10]:
```
labels_action = open(labels_action).readlines()
labels_action = [x.strip() for x in labels_action]
results_action = [(labels_action[k[0]], k[1]) for k in results_action]
main_actions = []

print(f'The top-5 labels with corresponding scores are:')
for result in results_action:
    print(f'{result[0]}: ', result[1])
    if result[1] >= 0.05:
        main_actions.append(result[0])
print(main_actions)
```

```
The top-5 labels with corresponding scores are:
keep_action:  0.36030072
straightroad_scenes:  0.29706997
overtaking_scenario:  0.24626523
lanechanging_scenario:  0.034766193
trafficsignal_scenes:  0.019153459
['keep_action', 'straightroad_scenes', 'overtaking_scenario']
```

[11]:
```
# Generate summary sentence
word_keys = main_actions[2].split('_')
```

[12]:
```
keys = f'Car | {word_keys[1]} | {word_keys[0]}'
generate(keys)
```

```
Generated in 2.69 seconds
```

[12]:
```
" Car is trying to overtake while it's on the move."
```

[13]:
```
show_result_pyplot(
    model_detection,
    main_img_detection,
    results_detection,
    score_thr=0.8,
    title='Dashcam Image')
```