



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

Design and Implementation of Video Subtitle Extractor Based on Deep Learning

School of Computer Science
University of Nottingham

Signature _____

Date 10/Sep/2020

Abstract

The exploding video data in recent years escalates the demand for video management. Video subtitles are valuable resources for video retrieval and classification. However, for the convenience of dissemination, video subtitles are often embedded in videos. Such embedded subtitles are of less reproducibility and cannot be used directly. Traditional subtitle extraction programs are complicated to use, since they require lots of manual setting of parameters, and experience low recognition accuracy. Although, there are a small number of applications of video text extraction based on deep learning, they are derived from object recognition and can be further improved and optimised for subtitle extraction. In this dissertation, I proposed a unified framework of subtitle extraction with four modules. The speed of one state of art text detection network, CTPN, has been optimised for subtitle detection via analysing subtitle features and introducing heuristic. Three public datasets are collected for training models and a testing dataset containing ten different genre of videos with a total of 46,812 frames has been created. An application that extracts hard subtitles from a video with just one single input and generates a mountable external subtitle file with timestamps is designed and successfully implemented. The experimental results on testing datasets indicate that the average F1 score of subtitle detection performance is above 91%, the correct rate of subtitle recognition is about 95% and the efficiency of subtitle detection is improved by 20%.

Keywords: Computer Science; Computer Vision; Deep Learning; Subtitle Detection; Subtitle Recognition; Subtitle Extractor; OCR; Hard Subtitle; Embedded Subtitle; CTPN.

Table of Contents

Abstract	I
Acknowledgement	II
List of Figures	V
List of Tables	VI
List of Acronyms	VII
Chapter 1 - Introduction.....	1
1.1 Motivation.....	1
1.2 Statement of the Research Problem	3
1.3 Aims and objectives	4
1.4 Structure of Dissertation	5
1.5 Contributions	6
Chapter 2 - Related Research and Technologies	7
2.1 Related Research.....	7
2.1.1 Text Detection.....	7
2.1.2 Text Recognition.....	9
2.1.3 Subtitle Extraction	10
2.2 Background Knowledge and Technologies	11
2.2.1 Features of video subtitles	11
2.2.2 Convolutional Neural Network.....	13
2.2.3 Recurrent Neural Networks	15
2.2.4 Long Short Term Memory	17
2.2.5 Gated Recurrent Unit	18
2.2.6 Connectionist Temporal Classification.....	19
2.2.7 VGGNet	21
2.2.8 DenseNet.....	21

Chapter 3 - Design of Subtitle Extractor.....	23
3.1 Evaluation	23
3.2 Methodology	23
3.3 Design	24
3.4 Data Gathering	29
Chapter 4 - Implementation of Subtitle Extractor.....	31
4.1 Frame Capture.....	31
4.2 Subtitle Detection.....	32
4.3 Subtitle Recognition.....	36
4.4 Subtitle Generation	38
Chapter 5 - Testing and Performance Evaluation	40
5.1 Evaluation Metrics	40
5.2 Experimental Results	45
5.2.1 Result of CTPN module.....	45
5.2.2 Result of OCR module.....	49
5.2.3 Accuracy of Timestamps	50
5.2.4 Speed of Subtitle Extractor System	51
5.3 Result Summary.....	52
Chapter 6 - Conclusion and Future Work	53
6.1 Conclusion	53
6.2 Future Work	54
Bibliography	56
Appendices.....	61
Appendix A: Comparison between Original and Generated SRT File	61
Appendix B: Source Code Hierarchy.....	71

List of Figures

Figure 1: Example of a Captured Video Frame with Manually Added Line Box	12
Figure 2: Example of a Local Receptive Field	14
Figure 3: Example of Max Pooling.....	15
Figure 4: a Recurrent Network	16
Figure 5: Architecture of VGG16.....	21
Figure 6: Architecture of DenseNet.....	22
Figure 7: Pipeline of Proposed Video Subtitle Extractor System.....	24
Figure 8: Text Position with Annotations.....	26
Figure 9: Description of CTPN Improvement	27
Figure 10: Creation of Testing Dataset.....	30
Figure 11: RPN Loss of First Attempt Training	33
Figure 12: Loss Value.....	34
Figure 13: OCR Module Training Details	36
Figure 14: Nine of the Highest-Activating Filters of the Final Convolutional Layer	37
Figure 15: Fragment of Raw Recognition Result in txt Format	38
Figure 16: Fragment of Original and Generated Subtitle File	44
Figure 17: Examples of FP and FN.....	46
Figure 18: Examples of TP	48
Figure 19: Performance of Ten Different Genre of Videos	48
Figure 20: Example of Incorrect Recognition	50

List of Tables

Table 1: Parameter Setting.....	34
Table 2: Confusion Matrix.....	40
Table 3: The Detection Statistics of CTPN.....	45
Table 4: The Recognition Statistics of OCR.....	49
Table 5: The Statistics of Timestamps Accuracy	51
Table 6: The Configuration of Testing Computer	51

List of Acronyms

Abbreviation	Definition/Explanation
API	Application Programming Interface
AR	Accuracy Rate
ATD	Average Timestamps Distance
BOW	Bag Of Words
CCA	Connected Components Analysis
CNN	Convolutional Neural Network
CR	Correct Rate
CTC	Connectionist Temporal Classification
CTPN	Connectionist Text Proposal Network
DP	Detection Precision
DR	Detection Recall
EAST	Efficient and Accurate Scene Text
ER	Error Rate
FC	Fully Connected
FPN	Feature Pyramid Network
FPS	Frame Per Second
GRU	Gated Recurrent Unit
LD	Levenshtein Distance
LSTM	Long Short Term Memory
NMS	Non Maximum Suppression
OCR	Optical Character Recognition
ResNet	Residual Network
RNN	Recurrent Neural Network
RRPN	Rotation Region Proposal Network
SCD	Synthetic Chinese dataset
SPAC	Spatial Autocorrelation
SW	Sliding Window
VSM	Vector Space Model

Chapter 1 - Introduction

This chapter will introduce the motivation of conducting research on subtitle extractor system and point out what research problems of this project are. Then aims and objectives are set and the potential contributions are explained.

1.1 Motivation

The rapid development of networks and assorted communication equipment have accelerated the advent of a new digital era for our lives. In the new age, 4G and 5G network technology have brought a variety of new features of higher speed, larger connection and lower latency to the multitudes, which popularises the widely spread of video as one of the main media for information dissemination.

On one hand, the surge of video data promotes the demand for an efficient approach to retrieve, manage and reuse information stored in video files. However, as a kind of unstructured image and audio stream, video lacks adequate index information expect for those titles and tags that cannot describe scenes and complement content of the video. Using titles and tags to retrieve and describe video content is no longer effective in face of bulk video data. In many cases, people can only use an ineffective and time consuming linear previews like fast forward and backward to retrieve information of their interest in the video. Thus, the problem that low-level features like titles and tags are insufficient to describe high-level semantic content of video arises.

On the other hand, subtitles is the detailed form of a video's transcription and contain high-level semantic information. They provide important and sufficient information to understand videos. Subtitles are also intended for use by viewers to remove gaps between their native language and speech used in video. They assist individuals who have auditory issues to comprehend and enjoy videos. Furthermore, a study conducted by The Office of Communications (Ofcom, 2006) suggests that around 80% of television viewers used subtitles for reasons other than hearing loss, which indicates the benefits of subtitles are beyond just providing accessibility. This ratio has maintained for many years and companies are still increasing their coverage rate of subtitles (Ofcom, 2020). Presently, subtitles play a more vital role in many real-world applications such as visual retrieval, visual problem solving. Nevertheless, for the convenience of dissemination, video subtitles are often embedded in video files in order to support different devices, which poses another problem that the reusability of subtitles and the capability of assisting information retrieval and management is greatly weakened.

Although video subtitles have great value and can be supplementary materials for understanding video and facilitating retrieval and classification of data, the embedded form of subtitles, which are merged into image sequences are unfortunately unable to be directly used. Therefore, extracting the subtitles embedded in the videos into separate text-based files is essential and beneficial. Currently, there exist several applications to execute subtitle extraction. However, most of them are facing different kind of problems. Some applications that adopt traditional computer graphic method are not user friendly and perform poorly in recognition, for they require clients to manually set a series of parameters to localise the subtitle area. In addition, in order to generate recognisable graphic subtitle files, manual

adjustment of filters for fronts of different size and styles is also unavoidable. Generally, the parameters set by users predominantly affect the accuracy of text recognition. Others that utilise deep learning techniques are mostly commercial software. They just provide application programming interfaces (APIs) and can only be accessed in online form. Users have to upload their large video files into provider's platform, and once the connection is interrupted, the whole process is terminated. Although there exist some open source projects for extracting text from images via deep learning method, plentiful and varied data sets are rarely employed for training and they are not specially optimised for subtitle extraction, which brings about issues of low recognition accuracy and low speed in execution. Motivated by these factors, the idea of designing and implementing a locally integrated subtitle extractor application based on deep learning technology emerges.

1.2 Statement of the Research Problem

In terms of subtitling approach, there are three types of subtitles which are embedded (i.e. hard-coded) subtitles, internal (i.e. soft-coded) subtitles and external subtitles. The embedded subtitles are graphic information fused with the video images. This kind of video captions cannot be altered nor directly reused for understanding, retrieving and classifying video. The internal subtitles are text data which are put into a video container along with the audio and video data. Internal video captions can be easily separated from the container into external subtitles via specific software. External subtitles are totally independent files which are capable to be directly utilised for lots of scenarios. The major research problem of this project is how to make embedded subtitles more reusable? which, in other words, means how to convert embedded video captions into external subtitles?

As a means to perform subtitle extraction, traditional methods require users to be 'professional' to manually set parameters, while it is difficult and time-consuming. In comparison, adopting deep learning techniques to perform classification task in automatically finding the subtitle areas and predicting results is a better alternative. This leads to the following research problem: Which is the ideal architecture for subtitle extraction? How to apply deep learning approach to carry out this task? How to take advantages of subtitle features to improve the architecture, reduce redundant computation (i.e. computational costs) and reach a higher accuracy as well as a faster speed?

1.3 Aims and objectives

The aims of this project is to investigate the existing problems of traditional method and deep learning approach in extracting video subtitles, design and implement an application that is able to extract the embedded graphic subtitles and convert them into a text-based subtitles document with corresponding timestamps for each input video. This project mainly focus on recognising Chinese, English and chinese-english-numerals mixed language and also concentrates on advancing the speed of extraction and the accuracy of recognition.

For the purpose of achieving these goals, the objectives of this project involves the following sections:

- Data set creation. Collect datasets of images with text in English, Chinese and Arabic numerals for training the designed network and create a dataset of videos with ground-truth subtitle files as testing dataset.

- Architecture evaluation and improvement. Examine existing deep learning architecture, select the most suitable one and improve it based on the features of subtitles to reduce computational costs and enhance the accuracy through model training.
- Testing and performance analysis. Evaluate the speed of subtitle extraction, the accuracy of timestamps and the correct rate of subtitle content by comparing the original ground-truth subtitles with the extracted (i.e. generated) files.

1.4 Structure of Dissertation

This paper consists of six chapters. The first chapter is a brief description concerning the research problem. This section elucidates the reason of conducting research on subtitle extraction and delineates what unsolved problems are and what goals and objectives of this project will be. In the second chapter, the context and relevant technologies of text detection and recognition will be elaborated. Furthermore, the related research and existing applications will be evaluated in detail and my analysis on the improvement will be presented. The third chapter will explain the methodology, design of the entire architecture and how different modules are connected. Chapter 4 will illuminate how the subtitle extractor is implemented in detail and how the models are trained. Chapter 5 will present the results and analyse the performance of detection and recognition via comparison between original subtitles and generated files. At last, based on what has been done, a conclusion and contribution will be stated and how to further develop the project will be discussed in the sixth chapter.

1.5 Contributions

The contribution of this dissertation contains three aspects: 1) With comparison between different architectures, the ideal deep learning network for video subtitle extraction has been found and by taking advantage of subtitles feature, the speed and accuracy of text detection in CTPN have been improved. 2) useful graphic subtitle data have been collected and an OCR model based on neural network for recognising subtitles in English, Arabic numerals, Chinese-English mixed language have been successfully trained. Furthermore, a testing dataset has been created for evaluating the system's performance. 3) a unified application is designed and implemented for subtitle extraction with one single input of video file and the corresponding generated subtitle files as output. With regard to the key result, the correct rate of subtitle recognition has reached about 95% based on testing dataset and the speed of detection has improved by 20%.

Chapter 2 - Related Research and Technologies

Video can be considered as a collection of a sequence of images and audio stream. The entire video subtitle extraction involves two major subtasks: subtitle area detection and subtitle recognition. These tasks are branches of text detection and recognition respectively. This chapter will introduce the related research and background knowledge of video subtitle extraction.

2.1 Related Research

2.1.1 Text Detection

Text detection is a process of predicting and localising the text areas from an image. Traditional text detection algorithms are usually based on the connectivity of text edges and texture features. Long et al. (2019) summarised text detection approaches before deep learning era and presented that the majority of traditional methods either adopt Connected Components Analysis (CCA) or Sliding Window (SW) based classification. According to Jung et al. (2004), this kind of methods take advantage of the homogeneous nature of subtitle colours, strong marginality, and large grayscale gap or color contrast between the subtitle and background to detect subtitle area. They usually need to merge tiny chunks in the image into possible text areas, and use a bottom-up structure. For instance, Song and Wang (2009) proposed a video text detection and positioning algorithm inspired by spatial autocorrelation method (SPAC) and Sobel algorithm. They use SPAC to localise potential subtitle areas, and then employ Sobel operator to extract the edges of potential subtitle areas and filter out some non-subtitle areas based on prior knowledge. After the process, the image binarization process is executed to separate the subtitle area from the complex background. Although their methodology works well especially for news videos, other genres of videos experience lots of situations with inaccuracy detection. Zhao et al. (2010) also utilised edge features and sparse

representations to localise video subtitle areas. Whereas, their proposed method has difficulty in detecting text of low resolution images. In general, the traditional image method has high requirements on the brightness and colour of video frames and need to execute repetitive pre-and-post processing. Their performance becomes poor when there are areas of the same shape or similar colour in the background, since they mainly extract low-level features which cannot address complicated cases.

With the breakthrough of convolutional neural networks (CNN) and its successful application, many text detection methods based on deep learning have been proposed. One kind of these methods is developed through general object detection methods, including Connectionist Text Proposal Network (CTPN) promoted by Tian et al. (2016), Rotation Region Proposal Networks (RRPN) presented by Ma et al. (2018) and IncepText (Yang, et al., 2018). This kind of method treats text as an object target. It first extracts image features through CNN, then generates a text candidate area by the anchor mechanism, and finally locates the text position accurately through classification and regression tasks. Another type of method is inspired by the semantic segmentation approach in traditional images method. These methods use Feature Pyramid Network (FPN) as the basic network to segment the image, and predict the connection relationship with surrounding pixels, and generate a text box based on the segmentation result and connection relationship (Lin, et al., 2017). Zhou et al. (2017) proposed the Efficient and Accurate Scene Text (EAST) algorithm, which can directly predict the quadrilateral phrases and text lines in the entire image, using a single neural network to eliminate unnecessary intermediate processing steps and simplify the entire architecture. The deep learning methods can reduce the burden of manually designing texture filters and have better versatility in detecting subtitle area than traditional approaches.

2.1.2 Text Recognition

Once text areas in the image are detected. They are usually cropped into a strip that comprises a single word or one sentence and then used as Optical Character Recognition (OCR) inputs.

As stated by Long (2019), the most challenging aspect of text recognition is the character segmentation. Traditional OCR techniques usually use OpenCV algorithm libraries to extract textual information from images through image processing and statistical machine learning methods, including binarization, noise filtering, correlation domain analysis (Jain & Sohal, 2014), and AdaBoost (Chen & Yuille, 2004). For images in simple scenarios, traditional OCR methods have achieved good recognition results. However, these approaches model images for specific scenes, and once the current scene is different, the model loses effectiveness.

Alternatively, deep learning-based OCR technology has gradually matured and can flexibly respond to different scenarios.

Currently, there are several routes based on deep learning to realise text recognition. One method proposed by Goodfellow et al. (2014) applies softmax in CNN and is mainly used for recognising simple and short text like street number. The architecture for each character recognition unfolds in two steps: firstly features are extracted via convolutional network, and then $N + 1$ softmax classifier are used to classify each character. Simple text sequences of indeterminate length can be effectively processed via this method, but recognition of longer character sequences does not work well. Another way proposed by Wojna et al. (2017) is to adopt attention in an amalgamation network of CNN and Recurrent Neural Network (RNN). This visual attention based algorithm model first processes the input images through a CNN and then the attentionally weighted feature sequence are taken in by an RNN. Finally, label sequences are decoded and output. The attention model used in this method allows the

decoder to compute variable context vectors via calculating weighted average of the hidden states of the encoder at each stage of the decoding operation, ensuring that the most relevant information is able to be accessed continuously without utterly depending on the hidden state at the antecedent moment (Shi et al., 2017). Although RNN can effectively learn contextual information and capture long-term dependencies, its huge recursive network calculations and the vanishing and exploding gradient problem make it difficult to train RNN models. Rather than RNN, Gao et al. (2017) have proposed stacked convolutional layers to address this issue. It combines CNN and Connectionist temporal classification (CTC) to create label sequences without any repeated connections. In comparison to RNN, their method saves lots of memory cost and improves the operation speed via parallel convolution processing. The entire architecture of this approach consists of three parts: The first part is feature encoder which extracts the feature vector of the text region in the image and generates a sequence of features. The second part is convolutional sequence modeling that converts feature sequences into two-dimension feature maps as inputs of the CNN and then obtains contextual relationships in the sequence. The last part is CTC layer that generates the final label sequence. This method has balanced both speed and recognition accuracy. There are also works on text recognition with rectification (Shi et al., 2016). However, taking the fact that video subtitles are usually not curved into account, these research on both circle and curved text recognition will not be reviewed, since they are not suitable for subtitle extraction.

2.1.3 Subtitle Extraction

At present, there are relatively few researches on applying deep learning technology to video subtitle extraction. These researches mainly consist of two categories: one is to extract audio track in the video and use speech recognition to generate the subtitle file (Sankhe et al., 2018;

Ramani et al., 2020); The other starts with the characteristics of hard subtitles, applies the scene text detection method to localise subtitle area (Yan & Xu, 2020), and then recognise the detected subtitle through traditional OCR engine such as Tesseract (Tian et al., 2018) or a deep neural network. The method of generating subtitles based on speech recognition is mainly suitable for video files that do not originally contain embedded subtitles. However, compared with videos that already contain embedded subtitles, this method consumes more calculation resources and the practical recognition accuracy is not good. Consequently, this project will not adopt this kind of method. The proposed framework of Tian et al. is named a generic Bayesian-based framework which focuses on the improvement of multi-frame integration and aims at extracting all text (i.e. both scene text and subtitle text) in the video. Since this method is not optimised according to the features of subtitles and uses a traditional OCR engine which does not support the Chinese language very well, the actual performance of their method for subtitle extraction has been thought to be improved. As an improvement, Yan and Xu's proposed to use Residual Network (ResNet), Gated Recurrent Unit (GRU) and CTC to recognise subtitles, which dramatically improve the accuracy and achieve state-of-art performance. Whereas, they directly adopt CTPN to detect subtitles, so some unnecessary calculations will be also executed, which can be avoided by improving the CTPN based on the features of subtitles.

2.2 Background Knowledge and Technologies

2.2.1 Features of video subtitles

As a kind of manually added text information, video subtitles are very rich in features. Leveraging these features is the key to detecting and recognising subtitles. Generally, the features of video subtitles can be summarised as follows:

- Video captions are usually unobstructed and located above the scene image.
- Video captions are rarely distorted and skewed. Many times they are located on the same horizontal line and centrally distributed in a rectangular text area.
- The rough position of subtitles in the same video is basically unchanged throughout the whole playback.
- In order to be observed, the video subtitles of the same content will last for a period of time in consecutive video frames, and the duration is usually greater than one second.
- Subtitles in the same video file usually have uniform brightness, size, color and font.

As an example shown in Fig. 1, the text with red outline is the video subtitle that we want to extract, while texts in the green line areas are scene texts which are only used for text detection and stored for future work. In this project, scene texts will not be used for text recognition. The text-like watermark in yellow outline is considered as an interference item.



Figure 1: Example of a Captured Video Frame with Manually Added Line Box

2.2.2 Convolutional Neural Network

In order to figure out how the computer see each video frame, detect the subtitle area and recognise text, we should understand the most import component, convolutional neural network.

CNN gets its name from the convolution operation. As Aggarwal (2018) wrote in his book, the dominant justification of executing convolution operation is to extract features from the input image, which simulates the way human beings perceive the outside world from part to whole. CNN contains convolutional layers whose parameters consist of a set of learnable filters, which can be considered as a two-dimensional digital matrix (Stanford University, n.d.). Filters can be convolved with the input image to produce an output image. Stacked convolutional layers can be used in CNN to deepen the depth of the network. As the network gets deeper, the semantic information is more abstract and more expressive. By sliding window through multiple filters on the input image, features with translation invariance are produced. Later on, the pooling layer compresses these features and utilise them to further extract key features. According to LeCun et al. (1998), three main ideas, which are local receptive fields, shared weights, and pooling are involved in CNN.

Video frames are high-dimensional inputs. It is impracticable to attach neurons to all other neurons when processing a video frame due to the huge amount of calculations. An alternative way is to connect each neuron to only a local region. Thus, the concept of local receptive fields which stands for the spatial extent of local connectivity is introduced. For instance, if the input is a 28×28 value, for the first hidden layer, each hidden layer neuron

has a connection to the 5×5 neurons in the input layer, then this 5×5 area is called the local receptive field which represents the sensory area of a hidden layer neuron in the input layer, as shown in Fig. 2 (Nielsen, 2015).

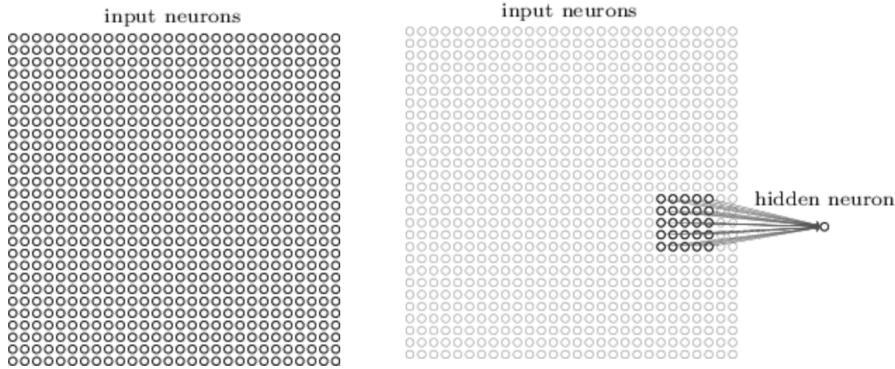


Figure 2: Example of a Local Receptive Field

Each hidden layer neuron corresponds to 5×5 (i.e. 25 in total) weights w and a base parameter b . In fact, we specify that each hidden layer neuron shares these 25 weights w and a bias value b . This means that the hidden layer neuron shares the weights. Also, it indicates that the hidden layer in the above diagram has only $5 \times 5 + 1$ (i.e. 26 in total) parameters, whereas if all connections were made, a number of $28 \times 28 \times N$ (N is the number of hidden neurons) parameters were required, which would be much more than 26 parameters. As a consequence, the shared weights would greatly reduce the quantity of parameters. For the j_{th} , k_{th} hidden neuron in the input layer x , the corresponding value is:

$$\sigma(b + \sum_{l=0}^4 \sum_{m=0}^4 w_{l,m} x_{j+l, k+m}) \quad (2-1)$$

Where σ is the neural activation function.

Neighbouring pixels in a picture have an inclination to share similar values, so do the output pixels which are adjacent to the convolutional layer. This denotes that there exist inessential

information in the output of the convolutional layer. If we want to detect a strong edge at a specific spot via utilising an edge detection filter, then other comparatively strong edge pixels around this location may also be found. However, these pixels are less of significance, for we are looking for something new. Fortunately, the pooling layer is designed to handle problem of redundant information. It can lessen the number of output values and avoid needless computation by reducing the size of the input. A Pooling process is typically done by either applying maximum, minimum filter, or performing average operations (He et al., 2014; Gong et al., 2014; Krizhevsky et al., 2017). An example of max-pooling from Standford University (n.d.) is shown in Fig. 3.

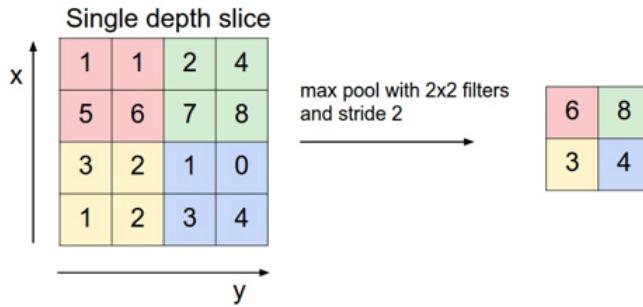


Figure 3: Example of Max Pooling

After the convolutional and pooling layers, we can get the highest level of abstraction and pass it into the fully connected (FC) layers in order to obtain the final probability. Fully connected layers act as classifiers in the entire convolutional neural network. This layer connects each neuron of the max-pooling layer to each output neurons and is responsible for converting the two-dimensional feature map output by the convolution into a one-dimensional vector.

2.2.3 Recurrent Neural Networks

The basic idea of recurrent neural networks is to harness sequential information. RNNs can save the previous input in the network and construct the mapping of all previous inputs to the target. Therefore, RNN can process the model from sequence data to the target, using

backpropagation to train. Figure 4 drawn by Britz (2015) illustrates the structure of an RNN.

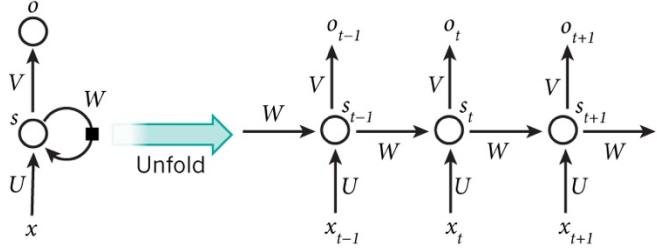


Figure 4: a Recurrent Network

In the above Figure 4, x , s and o are vectors. Specifically, x constitutes the value of the input layer. Vector s stands for the value of the hidden layer while o means the value of the output layer. U represents the weight matrix between the input layer and the hidden layer and V is another weight matrix projecting the hidden layer to the output layer. In a recurrent neural network, the value s of the hidden layer is calculated not merely on the passed x_t of current time, but also on the previous value s_{t-1} of the last hidden layer, i.e. the critical aspect is both x_t and s_{t-1} contributes to the value of s_t . W is weight matrix that contains the weight information of the last value of the hidden layer as the input for current time. At time t , the network takes the input x_t . Afterwards, the value s_t of the hidden layer, and the output result o_t are generated. We can express the computation of the recurrent neural network using the following formula:

$$o_t = g(Vs_t) \quad (2-2)$$

$$s_t = f(Ux_t + Ws_{t-1}) \quad (2-3)$$

Equation (2-2) is the formula for the output layer, which is a fully connected layer, (i.e. every neuron of the output layer is attached to every neuron of the hidden layer). g is the activation function. Eq. (2-3) is the formula for the hidden layer, which is a recurrent layer and f is the activation function. If we recursively plug Eq. (2-3) into Eq. (2-2), we will get:

$$o_t = Vf(Ux_t + Wf(Ux_{t-1} + Wf(Ux_{t-2} + Wf(Ux_{t-3} + \dots)))) \quad (2-4)$$

It can be seen from the above Eq. (2-4) that the output result of the recurrent neural network is influenced by the preceding input values $x_t, , x_{t-1}, x_{t-2}, x_{t-3}, \dots$, which indicates the reason why the recurrent neural network can look forward to any data of input values.

2.2.4 Long Short Term Memory

In video subtitle extraction, using former video frames may contribute to the subtitle detection and recognition of the current frame. If the previous information can be used to finish the present task, that would be extremely productive. The Long Short Term Memory (LSTM) architecture presented by Hochreiter and Schmidhuber (1997) can help RNN to address the problem of long term dependencies. A good example was conducted by Srivastava et al. (2016) who used multilayer LSTM network to learn representations of video sequences and successfully improved classification accuracy. The key point of LSTM is to introduce an intermediate vector, known as the cell state, C_t , that is capable to control the transmission of information to the next moment. As stated by Olah (2015), there are two sigmoid layers involved in LSTMs, which are respectively “forget gate layer” and “input gate layer”. The forget layer is responsible for determining whether to drop the hidden cell state of the previous layer with a certain probability and then output a value in the range 0 to 1 for the current cell state. The mathematical expression for forget layer is:

$$f_t = \sigma(Ws_{t-1} + Ux_t + b) \quad (2-5)$$

Where W and U are the coefficients of linear relationships, b is the bias value and σ is the sigmoid activation function, similar to those in RNN. The output f_t represents the probability of forgetting the previous hidden cell state.

The input gate handles the input for the current sequence position and decides which value will be updated. Afterwards, new candidate values potential to be stored in the cell state will be calculated via a tanh layer.

$$i_t = \sigma(Ws_{t-1} + Ux_t + b) \quad (2-6)$$

$$a_t = \tanh(Ws_{t-1} + Ux_t + b) \quad (2-7)$$

where i_t is the output of the input gate, and a_t is the result after the tanh layer.

Once the candidate values are computed, the cell state will be updated by multiplying the old state by f_t and adding the Hadamard product of i_t and a_t :

$$C_t = C_{t-1} * f_t + i_t * a_t \quad (2-8)$$

Finally, the output gate will calculate the result based on the updated cell state:

$$o_t = \sigma(Ws_{t-1} + Ux_t + b) \quad (2-9)$$

$$s_t = o_t * \tanh C_t \quad (2-10)$$

2.2.5 Gated Recurrent Unit

As an improvement, Cho et al. (2014) presented a variant of LSTM, which is known as Gated Recurrent Unit. It amalgamates the forget gate and input gate into a sole update gate and also consolidates cell state and hidden state. As a result, the GRU model is less of complexity than the typical LSTM model. There are only two gates involved in GRU, namely reset gate and

an update gate. On the whole, the reset gate adjudges how to conjoin the newly input data with the antecedent memory, whilst the update gate regulates the usage of the previous memory retained to the current time step.

The calculation method of the two gates in GRU is the same as the calculation method of the gate in LSTM. Compared with LSTM, the use of GRU can achieve the same results and performs similarly, but GRU is cheaper to train, so it is often more inclined to use GRU in practical training.

2.2.6 Connectionist Temporal Classification

CTC is an end-to-end RNN training method proposed by Graves et al. (2006). It enables RNN to learn sequence data straight away and avoids requiring the procedure to label the mapping between input sequences and output sequences in the training data beforehand, which makes the RNN model achieve better results in sequence learning tasks such as OCR.

In general, the core ideas of CTC are divided into the following parts:

1. It extends the output layer of RNN by adding a many-to-one spatial mapping between the output sequence and the final label, and defines the CTC Loss function based on that.
2. It borrows the Forward-Backward algorithm idea of Hidden Markov Model (HMM) (Jurafsky & Martin, 2019) and uses dynamic programming algorithm to efficiently calculate CTC Loss function and its derivatives, thus solving the end-to-end training problem of RNN.
3. Finally, combined with the CTC decoding algorithm, RNN can effectively make end-to-end prediction on sequence data.

For a given input sequence, CTC gives a distribution of all possible outputs. Based on this distribution, we can infer the maximum likelihood result or obtain the probability of a certain output. In addition, CTC introduces a blank placeholder on the basis of dataset and all labels to associate the input with the output, and finally delete all the placeholders through the deduplication operation. Bridle (1990) mentioned that a CTC network possesses a softmax output layer. The standard softmax function is defined by the formula (Goodfellow et al., 2016):

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad \text{for } i = 1, \dots, k \text{ and } z = (z_1, \dots, z_k) \in \mathbb{R}^k \quad (2-11)$$

where z_i is the activation response of output node i at time t and $\sigma(z)_i$ is the final output of that node at this time. As with most supervised learning, a CTC uses the maximum likelihood criterion for training. Given input x , the conditional probability of output l is:

$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) \quad (2-12)$$

where l is the label and π is all possible correct sequences, and B is a many-to-one map representing the set of possible labellings. $B^{-1}(l)$ represents the set of strings of length T which is mapped through B with the result l . CTC assumes that the probability of the output is conditionally independent (relative to the input). Thus:

$$p(\pi|x) = \prod y_{\pi_t}^t, \forall \pi \in L^T \quad (2-13)$$

where t represents the t -th step of RNN, and π represents the label of the path π at time t .

2.2.7 VGGNet

There are many deep convolutional neural networks promoted by researchers and scholars. One of them is called VGGNet, which is presented by Visual Geometry Group of the University of Oxford (Simonyan & Zisserman, 2014) together with researchers at Google DeepMind. It contemplates the correlation between the depth of a convolutional neural network and its performance by recurrently stacking 3×3 small convolutional kernels and 2×2 maximal pooling layers. VGGNet constructs a convolutional neural network with 16 and 19 layers deep successfully. The error rate of VGGNet is significantly reduced compared to the previous state-of-art network structure. VGG has two structures, namely VGG16 and VGG19. VGG16 contains a total of 13 convolutional layers, 3 fully connected layers and 5 pooling layers. Fig.5 illustrates the entire architecture of VGG16, which is most frequently used.

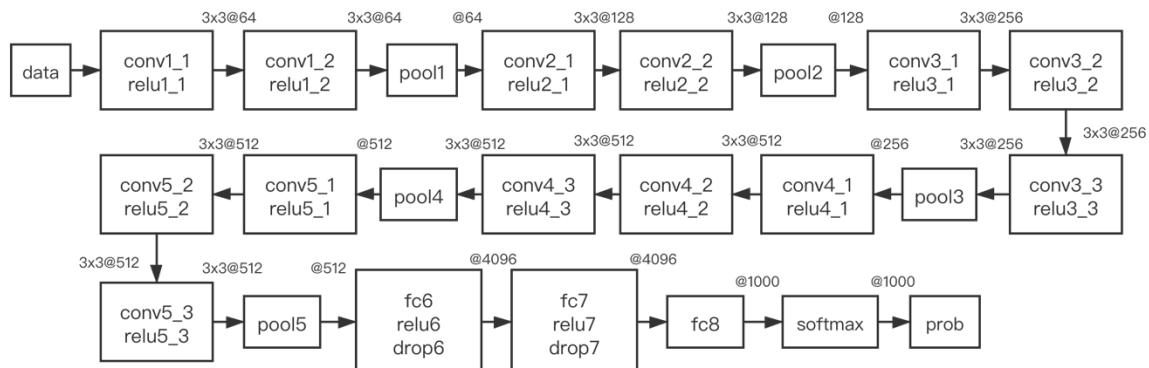


Figure 5: Architecture of VGG16

2.2.8 DenseNet

DenseNet is another convolutional neural network backbone proposed by Huang et al. (2017). The full name of DenseNet is Densely Connected Convolutional Networks. In a deep learning network, if we deepen the depth of the network, the problem of gradient vanishing is

more evident and likely to arise. To address this issue, one core idea is to establish short paths from early layers to later layers. Derived from this idea, the DenseNet directly connects each layer in the network to every other layer through a feed-forward manner (within each dense block) and meanwhile ensures information transfer between layers in the network at the maximum extent. DenseNet advances the back propagation of the gradient, reducing the difficulty to train the network. For the reason that each layer is capable to directly arrive at the final error signal, implicit "deep supervision" is accomplished. The architecture of DenseNet is shown in Fig. 6.

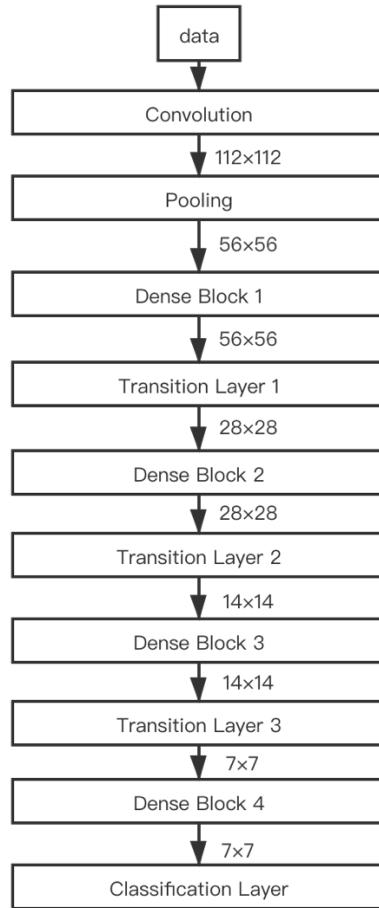


Figure 6: Architecture of DenseNet

Chapter 3 - Design of Subtitle Extractor

This chapter will introduce the design of subtitle extractor and provide detailed explanations of related methodology. Inspired by Yan and Xu (2020), I proposed an efficient subtitle extractor system based on deep learning. The flow chart of the system is illustrated in Fig. 7.

3.1 Evaluation

In the last chapter, we went through the related works and technologies and also analyse the features of subtitles. It is time to figure out which frameworks are best suitable for subtitle detection and recognition. For subtitle detection, it can be concluded that CTPN is the basic network for text detection and other methods can be considered as improved and optimised editions especially for scene text detection. Since the main objective of project is to extract subtitles and other scene text detection approach will actually result in redundant computation on addressing scene text, I choose CTPN as the basic network and improve it to localise subtitle areas. For subtitle recognition, the traditional OCR engine will face manual setting of low-level features and low accuracy of Chinese language recognition, thus we choose to train a deep learning model to address these issues. In order to balance the speed and accuracy, the structure of CNN, LSTM/GRU and CTC can be adopted.

3.2 Methodology

The video subtitle extractor system takes advantage of the subtitle characteristics, uses an improved CTPN to detect the subtitle area in video frames, and employs heuristics to optimise the speed of subtitle detection. After the detection, the coordinates of subtitle location and the corresponding cropped images of subtitles are generated. Then the cropped

images is fed in the OCR engine composed of CNN, LSTM (or GRU) and CTC in order to recognise subtitle text. Finally the subtitle text is converted into a subtitle file in *srt* format through post-processing. By training mass and diverse datasets, the model for detecting the subtitle area and the model for recognising text are obtained respectively. We can use these models to perform subtitle extraction.

3.3 Design

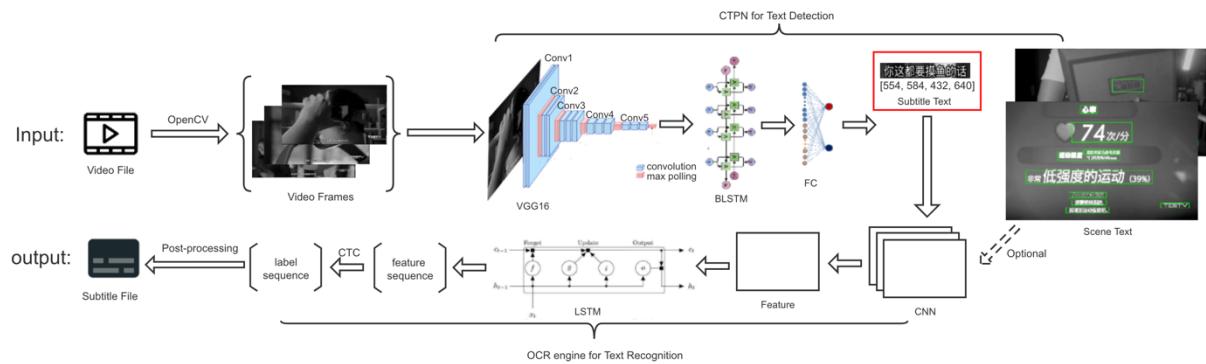


Figure 7: Pipeline of Proposed Video Subtitle Extractor System

The whole system consists of four modules: 1) pre-processing module; 2) CTPN module; 3) OCR module; 4) post-processing module. The CTPN and OCR module are composed of a neural network respectively. The interaction between these modules is relatively simple. Each module merely needs to receive the output of the previous module and execute the current process. Then its processing result can be directly output to the next stage.

The pre-processing module aims to reduce the redundant data in the original video frames. It calls the video processing interface in the OpenCV algorithm library. Through these interfaces, the local video is loaded into the memory, and the video frames are output one by

one in chronological order. However, in order to ensure the smoothness of the video, videos often contains above twenty-four frames of images per second. These video frames are mostly similar. Taking into account the features of subtitles mentioned in Chapter 3, it is unworthy and expensive to compute all frames. Thus, the system provides users with two options: 1) In order to achieve a certain time accuracy, users can choose whether to extract frame by frame. 2) By default, the system extracts one frame every second, which avoids a large number of repeated calculations and ensures that enough information is obtained, and the time accuracy is acceptable. After extracting video file into frames and discarding extra video frames, the module converts the RGB format video frames into grayscale images and feeds them into the next module. This module uses the basic image processing method and does not require the user to manually adjust the parameters according to different inputs.

The CTPN module is mainly used for the detection of video subtitle areas and utilises method proposed by Tian et al. (2016) to a certain extent. Further detailed information can be found in their article. This module improves and optimises the original method according to the characteristics of video subtitles. The convolutional neural network classifier in this module obtains text and non-text classification capabilities through learning samples. The first step of the CTPN module is the same as the generic object detection network. Initially, a backbone (i.e. VGG16) is used to extract spatial features of video frames, taking the output of VGG's conv5 layer, with an output dimension of $B \times W \times H \times C$ (batch size \times width \times height \times number of channels). Then, the CTPN slides a 3×3 window on the third convolutional layer in the fifth convolution block. After sliding window, the size of resulting feature map still remains the same, but then each pixel fuses $3 \times 3 \times C$ information. Afterwards, this feature map is reshaped and input into bi-directional LSTM so as to extract the sequence features of each line. The new resulting feature map is taken in by a fully connected layer. Then, the

output of FC layer is passed into a Three-branch RPN-like layer. The output of the first branch is the location of the anchor. The second branch is to output the scores of the foreground background (i.e. text/non-text scores), and the scores are calculated through softmax. The third branch is to output the proportion of the final horizontal side-refinement o . This is because the width of each anchor is fixed and sometimes it will cause a little inaccuracy in the horizontal direction. The refined formula is as follows:

$$o = (x_{side} - c_x^a)/w^a, o^* = (x_{side}^* - c_x^a)/w^a \quad (3-1)$$

where items with asterisk are ground truth, x_{side} represents the left or right boundary of the text box, c_x^a represents the abscissa of the anchor center, w^a is the anchor fixed width of 16 pixels. According to the above output results, a dense text proposal can be obtained, and the Non Maximum Suppression (NMS) algorithm is employed to suppress the redundant boxes. At last, the text line construction method is adopted to conjoin these proposal boxes obtained into text lines boxes and print the corresponding coordinates of $xmin$, $xmax$, $ymin$ and $ymax$, shown in Fig. 8. In order to achieve a faster speed in this module, we can make an alternative way by adopting heuristic approach to improve the original CTPN, illustrated in Fig. 9.



Figure 8: Text Position with Annotations

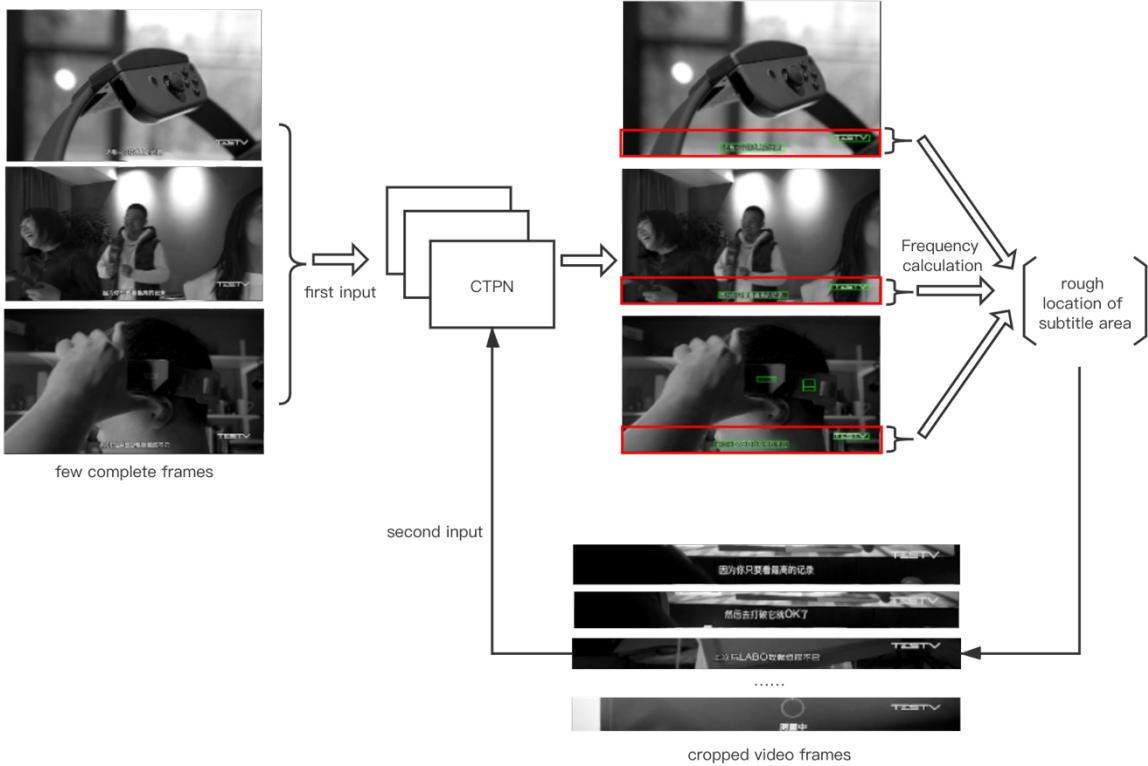


Figure 9: Description of CTPN Improvement

Instead of straightforwardly inputting complete video frames all the time, we can firstly cut a short video clip with complete frames from the original file as an input for subtitle detection. Through the calculation of CTPN, we can get a set of data, which contains the coordinates of the subtitle area and scene text areas. Since the subtitle area commonly has a relatively fixed location unlike scene text areas which appear irregularly, a rough location of the subtitle area can be determined based on the frequency of the y coordinates. Then we can lower down the computation cost by cropping the remaining video frames, leaving only the part in the rough position as inputs of CTPN. If the video contains a text-like watermark that has been detected by CTPN, it can be filtered out according to the unique characteristic that a watermark commonly has both fixed x-coordinate and the y-coordinate. If there are multiple watermarks in a video, we can remove all of them one by one via calling the corresponding function and finally find the rough subtitle area.

The OCR module uses a convolutional neural network as a recognition model and requires a training process for the network. The model is trained on the basis of text images of different fonts and qualities generated from about 6000 common Chinese characters, 52 English letters (26 lowercases and 26 upercases), ten Arabic numerals and above 100 different punctuation marks. Through CNN, the output of the previous CTPN module can be directly used as the input of the current model without manual intervention. CNN avoids complicated pre-processing before recognising subtitle and the high-level features extracted by the convolutional neural network is far more effective than the artificially designed low-level features. Since the task of text recognition is relatively simple compared to the task of localising subtitle regions, the OCR module uses a neural network of relatively simple structure. In deep learning, a complex network structure does not mean that the model will achieve better recognition effect. One explanation is that when the network structure is too complex and the corresponding training set is not sufficient, over-fitting will occur. The over-fitting model has very poor performance on results of unfamiliar data, which is not satisfactory. On the other hand, complex models mean more parameters need to be learned, which will lead to more computational costs.

The post-processing module generates an external subtitle file with timestamps from the extracted text according to the format of subtitle file. Usually there are two formats of subtitle files, *srt* and *ass* (or *ssa*). The subtitle file in *ass* format has special effects and can record the position of the subtitles, while the subtitle file in *srt* format cannot. Although the project can record the original position of the subtitles when generating the subtitle file, the focus of the project is not to generate subtitle files with special effects, so the project will eventually generate subtitles in *srt* format. The module also deletes duplicate subtitle lines and

watermark text if the previous choice is to keep scene text in the final subtitle file. Ultimately, it tries to introduce a language correction model, *pycorrector*, developed by Xu et al. (2020) to correct the misrecognised text in the subtitle file.

3.4 Data Gathering

Neural network is a data-driven method. The performance of the model is not only related to the structural optimisation design of the neural network, but also depends on large-scale and diversified data. In order to achieve the aim of extracting subtitles, we need to train the corresponding models for the CTPN and OCR modules respectively, so the collection of data sets is essential. In this project, I collect the ICDAR 2017 MLT dataset for training CTPN model, while the MJSynth dataset from VGG Lab used for recognising English and the Synthetic Chinese dataset (SCD) provided by Senlinuc (n.d.) used for recognising Chinese are both gathered for training subtitle recognition model. Similar to the artificially synthesized datasets, video captions are also manually synthesized into video frames. They share common features in certain extent. Thus, using synthetic datasets for training the model may achieve good results. In addition, I also create a testing dataset for the entire subtitle extraction by downloading video files and their corresponding subtitle files on YouTube. The dataset contains a total of ten videos with a total of 46,812 frames. The dataset contains videos of nine genres, which are 1) Unboxing, 2) Music Video, 3) Movie clip, 4) Amine clip, 5) Game clip, 6) Sports, 7) TV series, 8) Documentary, 9) Vlog, and video captions in this dataset involve two types of languages, which are Chinese and English. The creation of this dataset is shown in Fig. 10. Videos with embedded subtitles are used as inputs while manually labelled external subtitle files and original subtitle files downloaded are regard as ground truth. These subtitle files contain the correct timestamps and contents. By burning the

subtitle files into the corresponding videos through an open-source video handling suite, *ffmpeg*, we can get the video files with embedded hard subtitles. By extracting these videos, we can obtain the generated subtitle files. Finally we compare the generated subtitle files with the original files to evaluate the performance of the designed subtitle extractor system.

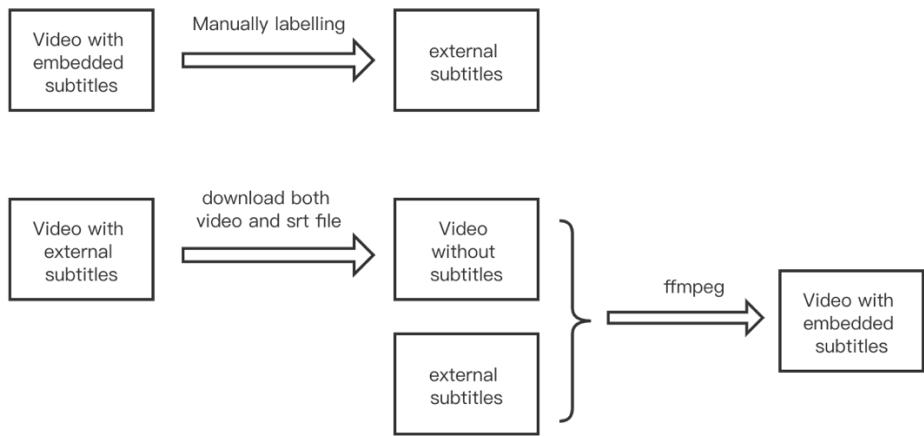


Figure 10: Creation of Testing Dataset

Chapter 4 - Implementation of Subtitle Extractor

In the previous chapter, four modules are introduced. This chapter describes how each of these modules is implemented. Firstly, how frames are captured is explained. Then, the training of the two neural network is illustrated. Finally, how the subtitle file is generated is delineated.

4.1 Frame Capture

The first step of localising subtitle areas in videos is to capture video frames, since the smallest unit for detection process is a single image. As stated by Singh and Kaur (2015), videos comprise of a number of basic temporal units, called shots, that are compartmentalised from one another through gradual and abrupt scene change. In order to lower down the cost of detecting text in videos and avoid repetitive and redundant computation, researchers proposed a dissimilarity index to distinguish different shots and extract key frames. This method is very effective for detecting scene text because the scene text will change with the transition of scene. However, if the key frame extraction method is directly used for subtitle detection, many subtitles will be discarded, for in the same shot, there are also subtitle changes. Therefore, a reasonable method is to capture video frames at a suitable frequency by analysing the characteristics of subtitles that appear in the video. In addition, video subtitles are clearer than scene text and usually do not appear blurry. Therefore, when extracting video frames, there is no need to use multi-frame fusion technology to enhance subtitles.

In this project, the frame capture task is implemented via *video_to_frames* function which calls algorithms in OpenCV libraries. First, it makes use of the *get* method in the

VideoCapture interface to read the basic information of the video and obtains the number of frames played per second of the video. Next, the module function captures the video frames at a certain frequency through the *read* method according to the *fps* information of input video. Finally, the *cvtColor* method is called to convert the captured video frame to greyscale image. The greyscale images are saved and then prepared to input into the next module. In an RGB colour image, a colour is formed by mixing the three primary colours of R (red), G (green), and B (blue) in proportion. The basic unit of an image is a pixel, and a pixel needs three representations, representing R, G, and B, respectively. The greyscale image uses black with different saturation to represent each image point. The conversion between RGB value and grayscale is actually the conversion of the human eye's perception of colour to brightness. According to Macedo et al. (2015), there are eighteen colour-to-greyscale algorithms and one algorithm which is adopted in this project called Luminance uses the formula:

$$\text{Grey} = 0.21 \times R + 0.71 \times G + 0.07 \times B \quad (4-1)$$

Based on this formula, we can read the R, G, B values of each pixel in turn, calculate the grey value which is converted to an integer number, and assign the grey value to the corresponding position of the new image. After all the pixels are traversed, the conversion is completed.

4.2 Subtitle Detection

After we have captured video frames and converted them into greyscale images, the next step is to detect subtitle areas in these images. The key to implement this is to train a detection model. The whole pipeline is trained on a Linux PC with a 3.8 GHz Xeon(TM) CPU and 32

GB RAM using two NVIDIA TITAN X (Pascal) GPU with CUDA API. The network architecture of CTPN module for the Base CNN model is the deep VGG16 model. The model is trained with approximately 3000 natural text images, each annotated with text-line bounding boxes. In addition, the detection model is implemented in Python 3 and uses the open-source deep learning framework TensorFlow for training our neural networks, which is optimised for quickly computing multidimensional data on GPUs. All models have a summary writer during training which is able to record the layer architecture as well as the models' weights. This makes it easy to load, evaluate, and deploy later. All measurements are logged and visualized using TensorBoard. An example of training from scratch is shown in Fig. 11.

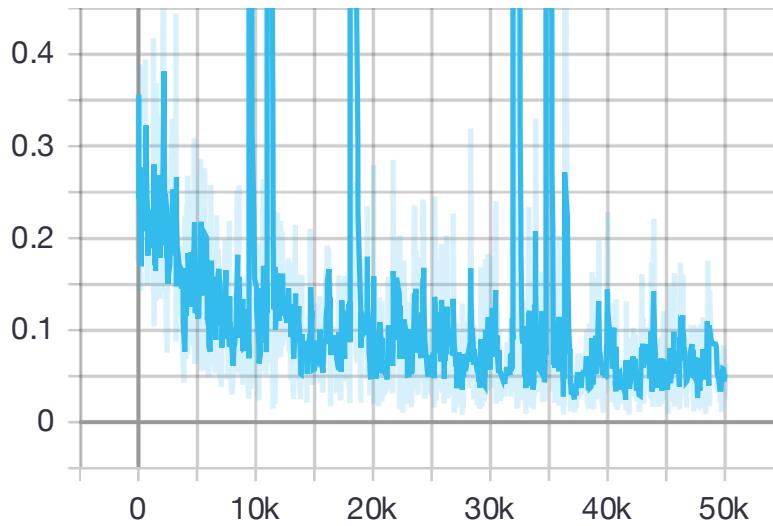


Figure 11: RPN Loss of First Attempt Training

While training from scratch, the RPN loss sometimes spikes and then decreasing in a strange manner. I traced the code and found that in such special cases where RPN loss becomes unreasonably high, the area of some text proposals become zero. Thus, it seems the RPN could not propose very small anchors. So I adopted the restriction for the box sizes. In this way, some very small boxes are filtered out thus avoid the model producing the text proposals with small area. I also observed that loading pre-trained weight, rather than starting from scratch can also avoid this problem. Thus I finally train the models with restricted box scale

and remove the filter after some epochs as well as try to reduce the learning rate at the same time. Finally an ideal loss cost tend was achieved, which is shown in Fig. 12.

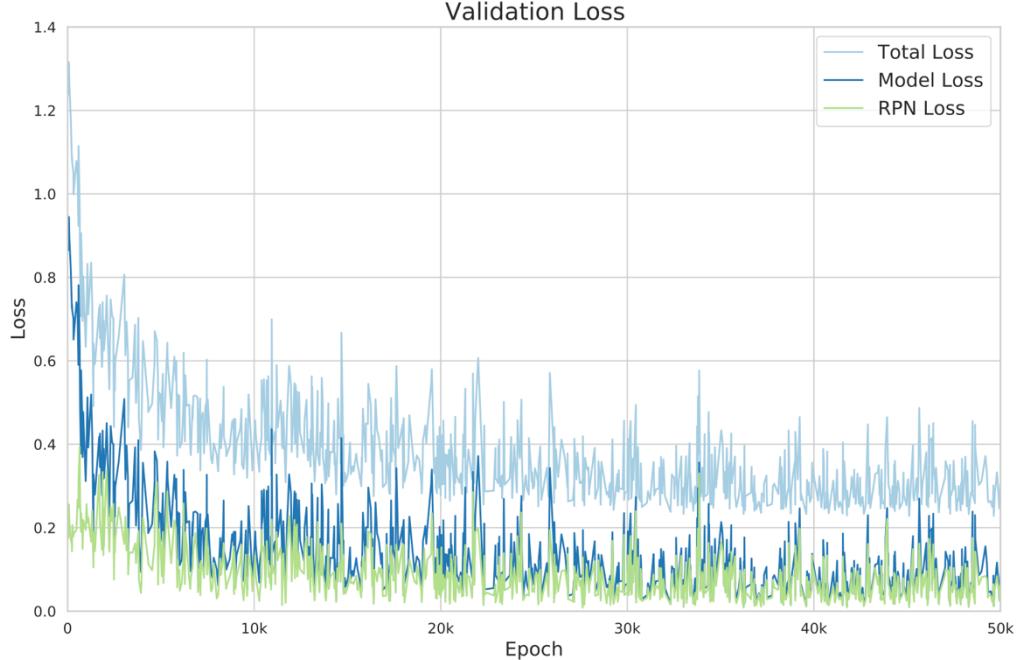


Figure 12: Loss Value

To provide more details, I further investigate the curves of training loss. Obviously, the proposed RNN layer greatly facilitates model optimisation and convergence as indicated by the reduction in training loss. This provides a theoretical evidence that the recurrent design is efficient. The parameter setting is shown in Table 1.

Table 1: Parameter Setting

Parameters	Configurations
Momentum	0.9
Weight decay	0.0005
Learning Rate	First: 1e-5 Last: 1e-6
Epoch	50K

The loss function consists of three parts as:

$$L(s_i, v_j, o_k) = \frac{1}{N_s} \sum_i L_s^{cl} (s_i, s_i^*) + \frac{\lambda_1}{N_v} \sum_j L_v^{re} (v_j, v_j^*) + \frac{\lambda_2}{N_o} \sum_k L_o^{re} (o_k, o_k^*) \quad (4-2)$$

- **Anchor Softmax loss** is used to supervise whether each anchor contains the text.
- **Anchor y coordinate regression loss** is used to supervise the bounding box regression offset along y axis for those anchors containing the text, which in some ways similar to smooth L1 loss. The v_j are those anchors which are classified as “positive”, in other words, their vertical IoU with ground truth is larger than 0.5.
- **Anchor x coordinate regression loss** is used to supervise the bounding box regression offset along x axis. Similar as Faster R-CNN method, during the training process of bounding box regression, it seems that the model only need to pay attention to those anchors who have been classified as positive while no need to worry about the messy rest.

There is an important part in the implementation which is side-refinement method. It combines multiple fine-scale text proposals into a text line, using a couple of rules, i.e. under which circumstance two fine-scale text proposals could merge into one. Here is the algorithm's pipeline:

1. Sort the anchors by their horizontal x coordinates
2. For each anchor box_i

Do forward and backward search for the pair box_j which satisfies three conditions: first, the distance between box_i and box_j along the x axis is less than 50. Second the vertical direction overlap between box_i and box_j is greater than 0.7. If both the two conditions are satisfied, choose the one with the maximum softmax score.

Set $Graph(i,j)$ equal to True.

In this way, a $N \times N$ (positive anchors' size) connected graph was built and we can find the connected components by a simple traversal.

4.3 Subtitle Recognition

A primary advantage of deep neural networks is their ability to identify and learn useful features without requiring a data scientist to manually define these. Therefore, no prior domain knowledge is needed, a fact that makes these techniques so powerful and versatile. From an outsider's perspective, deep learning can appear as a black box, and it remains unclear which features were ultimately deemed relevant.

The OCR module employs CNN followed by bidirectional RNN or GRU layers. To implement subtitle recognition, a model is trained end-to-end using a CTC loss function. The CRNN text recognition model was implemented based on Apache MXNet (incubating). During training, The loss curves are observed and recorded, which is illustrated in Fig. 13.

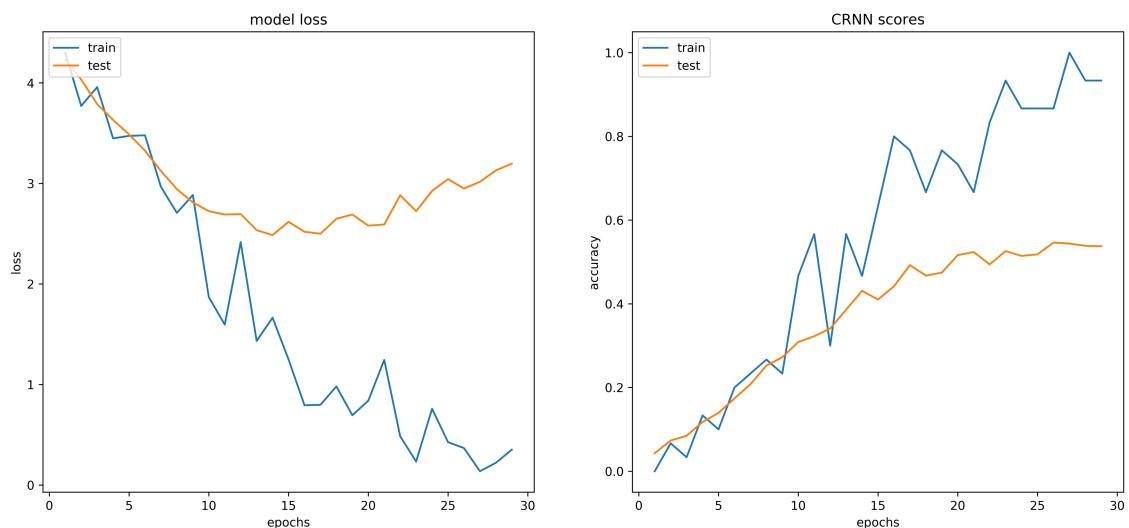


Figure 13: OCR Module Training Details

Since RNN models with CTC loss function tend not to converge if the training is not carefully set. I carefully check the CTC loss output in order to make sure whether a model would converge (i.e. the CTC loss at each batch should fluctuates instead of shrinks to a stable value).

In order to gain a better understanding of the model, I visualised nine of the highest-activating filters of the final convolutional layer, which are shown in Fig. 14. To obtain these filters, I performed backpropagation from the output of each filter back to the input image. This yielded the gradients in the filter output with respect to the input image pixels. I used these to perform gradient ascent, searching for the image pixels that maximise the output of the filter.

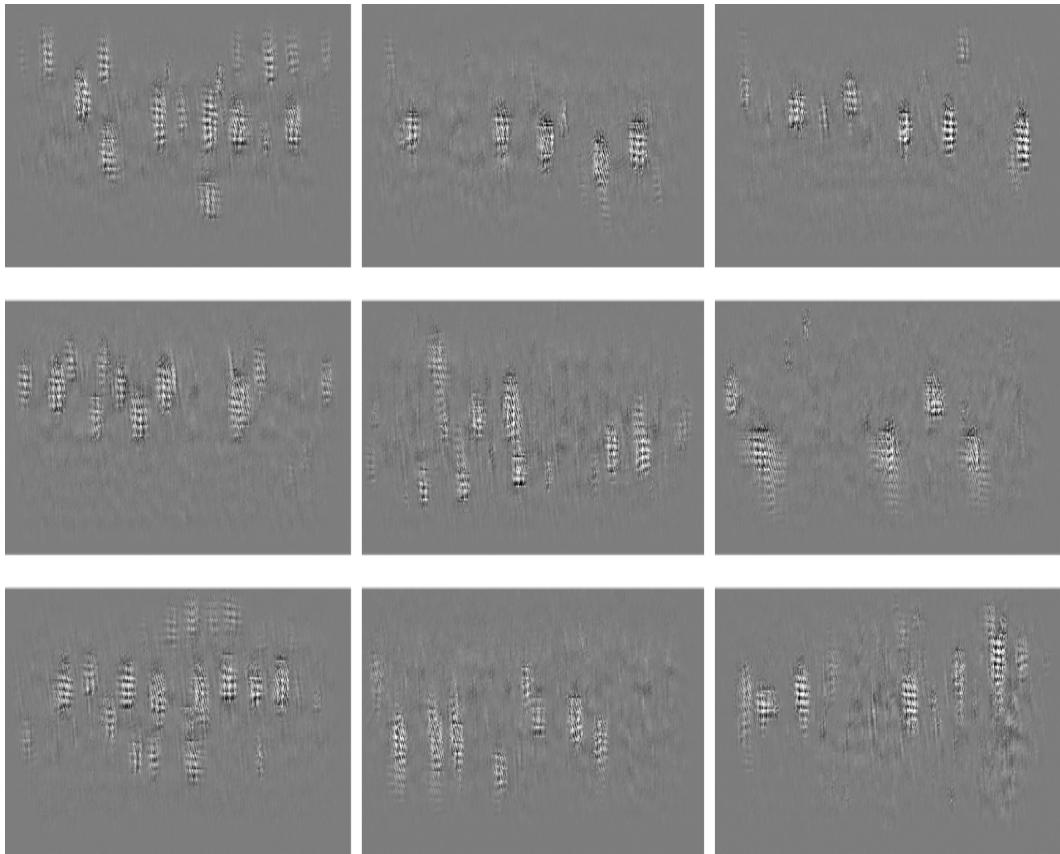


Figure 14: Nine of the Highest-Activating Filters of the Final Convolutional Layer

4.4 Subtitle Generation

After all subtitles in the video are recognised and converted to text, a raw text file in *txt* format will be generated (An example of the fragment is shown in Fig. 15). This *txt* file contains the indices of frames that involves subtitle information, the original position and the content of subtitles.

1266	[555, 584, 208, 864]	在每个人都认为自己可以胜任工作的世界中, 看看这个家伙
1289	[555, 583, 208, 864]	在每个人都认为自己可以胜任工作的世界中, 看看这个家伙
1312	[555, 583, 480, 592]	我小时候
1335	[556, 583, 384, 672]	告诉人们我要当喜剧演员
1358	[556, 583, 384, 672]	告诉人们我要当喜剧演员
1381	[555, 583, 384, 672]	告诉人们我要当喜剧演员
1404	[555, 584, 432, 640]	每个人都嘲笑我
1427	[555, 583, 432, 624]	每个人都嘲笑我
1450	[555, 583, 400, 672]	好吧, 现在没有人在笑
1473	[556, 583, 400, 672]	好吧, 现在没有人在笑
1519	[555, 584, 400, 672]	薏你可以再说臺遍, 朋友
1818	[557, 583, 448, 608]	太可怕了吧
1841	[557, 584, 448, 608]	太可怕了吧
1979	[556, 583, 448, 624]	在我的一生中
2002	[554, 583, 448, 624]	在我的一生中
2071	[555, 583, 384, 672]	我不知道我是否真的存在
2094	[555, 583, 384, 672]	我不知道我是否真的存在
2117	[555, 583, 384, 672]	我不知道我是否真的存在
2186	[557, 582, 464, 608]	但是我愿意
2209	[557, 582, 464, 608]	但是我愿意
2255	[555, 584, 432, 624]	人们开始注意到
2278	[555, 584, 432, 624]	人们开始注意到
2301	[555, 584, 432, 624]	人们开始注意到
2347	[556, 584, 416, 640]	你觉得这很有趣吗

Figure 15: Fragment of Raw Recognition Result in *txt* Format

We can call the *frames_to_timecode* function to convert the frame indices into timecode. For a subtitle line, the index of its first occurrence is recorded as the starting time while the index of the last occurrence is recorded as ending time. By connecting the starting time and ending time according to the standard format of the *srt* file, we can generate the timestamp of this subtitles line. At the same time, we set a similarity threshold, and judge the similarity between the current and next subtitle line. If the similarity of the two lines of subtitles is lower than the set threshold, it is considered that the content of the subtitles has changed.

Then, the ending time of current subtitle line and a new starting time of next line will be recorded. If a subtitle line appears only once, it means that the duration of this subtitle line in the original video is relatively short, so we can set the ending time of this subtitle line to the starting time of the next subtitle line. The similarity threshold is set according the correct rate and the trained OCR model. If the performance of the trained model is very excellent, then a higher threshold can be set. There are several ways to calculate the similarity of two subtitle lines. One of them is call the cosine similarity which is defined as:

$$\text{Similarity} = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (4-3)$$

The cosine similarity is defined in the vector space model (VSM). When using it to measure the similarity of two strings, it is needed to vectorise the strings first, which is performed by using a bag-of-words (BOW) model. Another method is to compute the hamming distance, but this approach only works when the two input strings are equal in length. In this project, an easy implementation is to adopt the *fuzzywuzzy* python library, which employs Levenshtein Distance (LD) algorithm to evaluate the difference and then use the difference value to compute the score of similarity. Further details of LD are described in Chapter 5.

Chapter 5 - Testing and Performance Evaluation

In this part, the performance of the designed video subtitle extractor system will be evaluated with practical testing according to the detection precision, recognition accuracy and the average timestamps distance of generated subtitle file.

5.1 Evaluation Metrics

As a way to evaluate the performance of the implemented subtitle extractor, we analyse each module separately. Since subtitle detection is a common binary classification problem, it is helpful to introduce the confusion matrix, which is defined in Table 2, when we want to effectively evaluate the detection result of this module:

Table 2: Confusion Matrix

		Prediction Class	
		True	False
Actual Class	True	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (TN)

True Positive: When a subtitle area or a scene text area is correctly classified as a text area.

$$\text{TP} = \text{Number of text areas classified as text area correctly}$$

False Positive: When a non-subtitle area or a non-scene text area is incorrectly classified as a text area.

$$\text{FP} = \text{Number of non text areas classified as text area incorrectly}$$

True Negative: When a non-subtitle area or a non-scene text area is correctly classified as a non-text area.

$$\text{TN} = \text{Number of non text areas classified as non text area correctly}$$

False Negative: When a subtitle area or a scene text area is incorrectly classified as a non-text area.

$$FN = \text{Number of text areas classified as non text area incorrectly}$$

Accuracy: refers to a percentage of the amount of correct predictions divided by the amount of all prediction on classifying the text area. It can be formally defined as follows.

$$A = \frac{\text{Number of correct predictions}}{\text{Number of all predictions}} = \frac{TN+TP}{TN+TP+FN+FP} \quad (5-1)$$

Error Rate: refers to a percentage of the number of wrong predictions divided by the number of all prediction on classifying the text area. It can be formally defined as follows.

$$ER = \frac{\text{Number of incorrect predictions}}{\text{Number of all predictions}} = \frac{FN+FP}{TN+TP+FN+FP} \quad (5-2)$$

Detection Precision: tells how many of those which are classified as subtitle area or scene text area are real text area.

$$DP = \frac{TP}{TP+FP} \quad (5-3)$$

Detection Recall: tells how many of real subtitle areas and scene text area are predicted as text areas.

$$DR = \frac{TP}{TP+FN} \quad (5-4)$$

F_β-score: balances detection precision and detection recall and computes the mean; β is used for weighting precision and recall. When we set β to 1, both precision and recall will be valued equally.

$$F_{\beta} = \frac{(1+\beta^2)DP \times DR}{\beta^2(DP+DR)} \quad (5-5)$$

$$F_1 = \frac{2DP \times DR}{DP + DR} \quad (5-6)$$

In text detection and classification, since TN is difficult to count, we use DP, DR and F_1 as the main metrics to reflect the performance of detection module.

For the OCR module, we use the performance evaluation standard of the text line recognition algorithm defined by the ICDAR2013 competition as metrics. The Correct Rate (CR) of recognition is represented by:

$$CR = \frac{N_t - D_\varepsilon - S_\varepsilon}{N_t} \quad (5-7)$$

The Accuracy Rate (AR) can be calculated by :

$$AR = \frac{N_t - D_\varepsilon - S_\varepsilon - I_\varepsilon}{N_t} \quad (5-8)$$

where S_ε represents the minimum number of substitutions, D_ε is the minimum number of for deletions, N_t stands for the total number of Chinese characters and English letters, and I_ε is the minimum number of insertions. In this project, the I_ε do not involve the correctly recognised and inserted scene text, since scene text is an option that user can choose to keep in the generated file. Instead, I_ε means those which do not actually belong to text but mistakenly recognised as text and inserted in the final subtitle file.

When we recognise the text in the subtitle text area, the results obtained may have difference to the ground truth. To calculate the difference and the sum of S_ε , D_ε and I_ε , we can introduce the concept of Levenshtein Distance. Levenshtein distance (also known as Edit

Distance) stands for the minimum amount of edit operations essential for converting two strings from one to the other. The greater the distance between two string sequences, the more different they are. A smaller distance reflects a higher accuracy of OCR recognition.

The Levenshtein distance between two strings s_1, s_2 (of length $|s_1|$ and $|s_1|$ respectively) is given by $Lev_{s_1,s_2}(|s_1|, |s_1|)$ where

$$Lev_{s_1,s_2}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} Lev_{s_1,s_2}(i-1,j) + 1 \\ Lev_{s_1,s_2}(i,j-1) + 1 \\ Lev_{s_1,s_2}(i-1,j-1) + 1_{(s_1^{(i)} \neq s_2^{(i)})} \end{cases} & \text{otherwise} \end{cases} \quad (5-9)$$

where $1_{(s_1^{(i)} \neq s_2^{(i)})}$ is the indicator function equal to 0 when $s_1^{(i)} = s_2^{(i)}$ and equal to 1 otherwise, and $Lev_{s_1,s_2}(i,j)$ is the distance between the first i characters of s_1 and the first j characters of s_2 . i and j are 1-based indices (Wikipedia, 2020). An algorithm implementation using dynamic programming is described as follows (Hjelmqvist, 2012):

Algorithm Levenshtein Distance

- 1: Construct a two-dimensional array with $m+1$ rows and $n+1$ columns to store the number of operations that is required to complete a certain conversion. The amount of operations essential for converting the string $s_1[1..n]$ to the string $s_2[1..m]$ is the value of array[n][m];
- 2: Initialise the first row of the array from 0 to n , and the first column from 0 to m .
array[0][j] indicates the value of the j -th column in the first row. This value represents the number of operations required to convert the string $s_1[1..0]$ to $s_2[1..j]$.
- 3: Inspect each character of s_1 (i from 1 to n).
- 4: Inspect each character of s_2 (j from 1 to m).
- 5: If $s_1[i]$ equals $s_2[j]$, the cost is 0.
If $s_1[i]$ is not equal to $s_2[j]$, the cost is 1.
- 6: Set cell matrix[i][j] equal to the minimum of:
 - a. The cell immediately to the left add 1: array[i-1, j] + 1.
 - b. The cell immediately above add 1: array[i, j-1] + 1.
 - c. The cell diagonally above and to the left add the cost: array[i-1, j-1] + cost.
- 7: Iterate steps 3, 4, 5, 6 until the matrix is completed, then the distance of s_1 and s_2 can be found in the cell array[n, m].

Since this project finally generates mountable external subtitle files, it is necessary to evaluate how accurate the timestamps is compared to the original. I proposed the Average Timestamps Distance as a metric:

Average Timestamps Distance: refers to the average distance between the timestamps in generated subtitle files and the timestamps in original subtitle files.

$$ATD = \frac{\sum_i^n (|S_o^i - S_g^i| + |E_o^i - E_g^i|)}{2n} \quad (5-10)$$

where n indicates the number of subtitle lines, S_o^i is the starting time (converted to milliseconds) of the i -th subtitle line in the original subtitle file, S_g^i is the starting time of the i -th subtitle line in the generated subtitle file, E_o^i is the ending time of the i -th subtitle line in the original subtitle file and E_g^i is the ending time of the i -th subtitle line in the generated subtitle file. A subtitle fragment with comparison between original subtitle file and generated subtitle file is shown in the Fig. 16.

original.srt		generated.srt	
1	1	1	1
2	00:00:00,000 --> 00:00:02,160	00:00:00,01 --> 00:00:02,01	00:00:00,01 --> 00:00:02,01
3	因为你只要看最高的记录	因为你只要看最高的记录	因为你只要看最高的记录
4			
5	2	2	2
6	00:00:02,161 --> 00:00:04,440	00:00:02,01 --> 00:00:04,01	00:00:02,01 --> 00:00:04,01
7	然后去打破它就OK了	然后去打破它就OK了	然后去打破它就OK了
8			
9	3	3	3
10	00:00:04,440 --> 00:00:06,400	00:00:04,01 --> 00:00:06,01	00:00:04,01 --> 00:00:06,01
11	还有一个技术上的问题	还有一个技术的问题	还有一个技术的问题
12			
13	4	4	4
14	00:00:06,401 --> 00:00:08,280	00:00:06,01 --> 00:00:07,01	00:00:06,01 --> 00:00:07,01
15	Joy-Con是真的牛皮	Joy-Con是真的牛皮	Joy-Con是真的牛皮
16			
17	5	5	5
18	00:00:08,281 --> 00:00:10,560	00:00:07,01 --> 00:00:10,01	00:00:07,01 --> 00:00:10,01
19	上次玩LABO我就惊叹不已	上次玩LLABO我就惊叹不已	上次玩LLABO我就惊叹不已
20			
21	6	6	6
22	00:00:10,560 --> 00:00:14,079	00:00:10,01 --> 00:00:13,01	00:00:10,01 --> 00:00:13,01
23	这次不但承担起了整个健身环的运动判定	这次不但担起了整个健身环的运动判定	这次不但担起了整个健身环的运动判定
24			

Figure 16: Fragment of Original and Generated Subtitle File

It is worth noting that the content of S_o^i needs to correspond to S_g^i , and similarly the content of E_o^i needs to correspond to E_g^i , which, in other words, if there are new content and corresponding new timestamps in the generated subtitle file or there are deletions , we will not use this part for calculation. Moreover, the addition and deletion of spaces are also not included in the calculation.

5.2 Experimental Results

In compliance with the metrics narrated above, experimental results can be extracted and summarised from the output generated by subtitle extractor after each module finished. Then, we use these data to generate tables and plot line graph in order to understand the performance of the design system intuitively. Afterwards, the detailed analysis will be present to explain those outstanding figures.

5.2.1 Result of CTPN module

Table 3: The Detection Statistics of CTPN

Input	Results							
	Total Frames	Subtitle Frames	TP	FP	FN	DP	DR	F1
Video 1	920	36	102	3	8	0.971	0.927	0.948
Video 2	747	31	70	7	4	0.909	0.946	0.927
Video 3	6775	226	626	6	120	0.990	0.839	0.908
Video 4	3456	76	75	35	8	0.682	0.904	0.777
Video 5	4698	134	231	21	6	0.917	0.975	0.945
Video 6	2875	109	429	36	1	0.923	0.998	0.959
Video 7	3002	116	671	34	27	0.952	0.961	0.956
Video 8	4355	125	148	1	107	0.993	0.580	0.732
Video 9	6314	231	366	29	3	0.926	0.992	0.958
Video 10	13670	265	545	8	0	0.995	1.000	0.997

The Table 3 shows the result of CTPN module after ten different videos were input. Looking at the data from the DP column, it is clear that the detection precision has achieved 90% higher except for the fourth video. One reason for this is that the fourth video is an excerpt taken from a movie and movies have more complex scenes and more disturbing features, compared with other types of videos. Another reason is movies have lots of scene elements and caption text that resemble subtitles and bring about false positive. There are 35 non-text areas in this video that were incorrectly classified as text areas and as a consequence the number of FP rises that contributes to the decline of DP. An example of the misclassification (the mouth of clown) is illustrated in Fig. 17a. Another important column is the DR, which refers to the detection recall and is one of the main metrics for CTPN module. We can easily calculate the mean and notice that the average DP of these ten videos has reached 91.2% higher. One outstanding figure in this column is from the eighth video which is an episode of TV series. The FN of the eighth video is very high, since there is an illegible text watermark (shown in Fig. 17b with red elliptic outline) throughout the video and every frame contains this watermark which the CTPN failed to detect and classify as text areas.

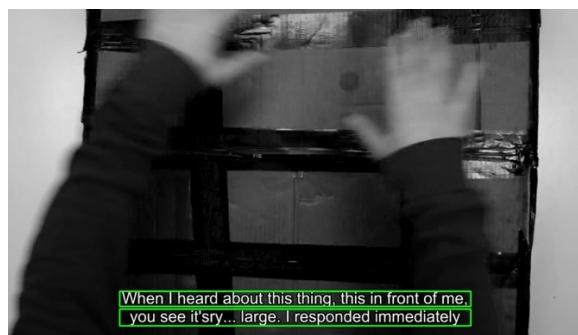


Figure 17: Examples of FP and FN

The tenth video has attained the best performance with zero FN and few FP, for it is a vlog that has a very simple scene background of white colour and rare scene switching can be seen. Examples of TP of the testing dataset can be found in Fig. 18.



(a) Unboxing video (Chinese and few English)



(b) Unboxing video (English)



(c) Music Video (Chinese)



(d) movie (English)



(e) Amine (Chinese)



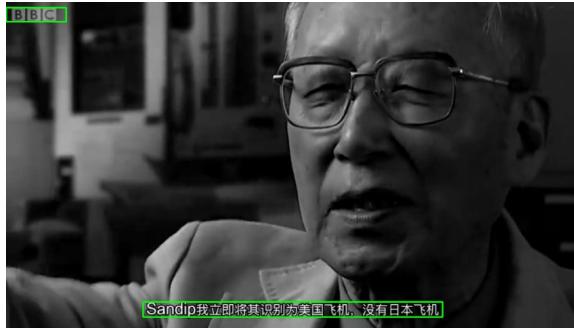
(f) game (English)



(g) sports (Chinese and few English)



(h) TV series (Chinese)



(i) documentary (Chinese and few English)



(j) Vlog (English)

Figure 18: Examples of TP

As we can see from Fig. 18, all subtitle areas are detected perfectly although there exist interference items (i.e. watermarks, shown in Fig. 18 *a, f, g, h and i*) which can be filtered via improved CTPN or the post-processing module. Finally, we can utilise the F1 column to check the balanced value of DP and DR. As we evaluate the mean, we can find that the average F1 for those ten testing video is above 91% which generally shows an excellent performance on text detection. To understand the performance more clearly and deeply on different genre of videos, a line graph is shown in Fig. 19.

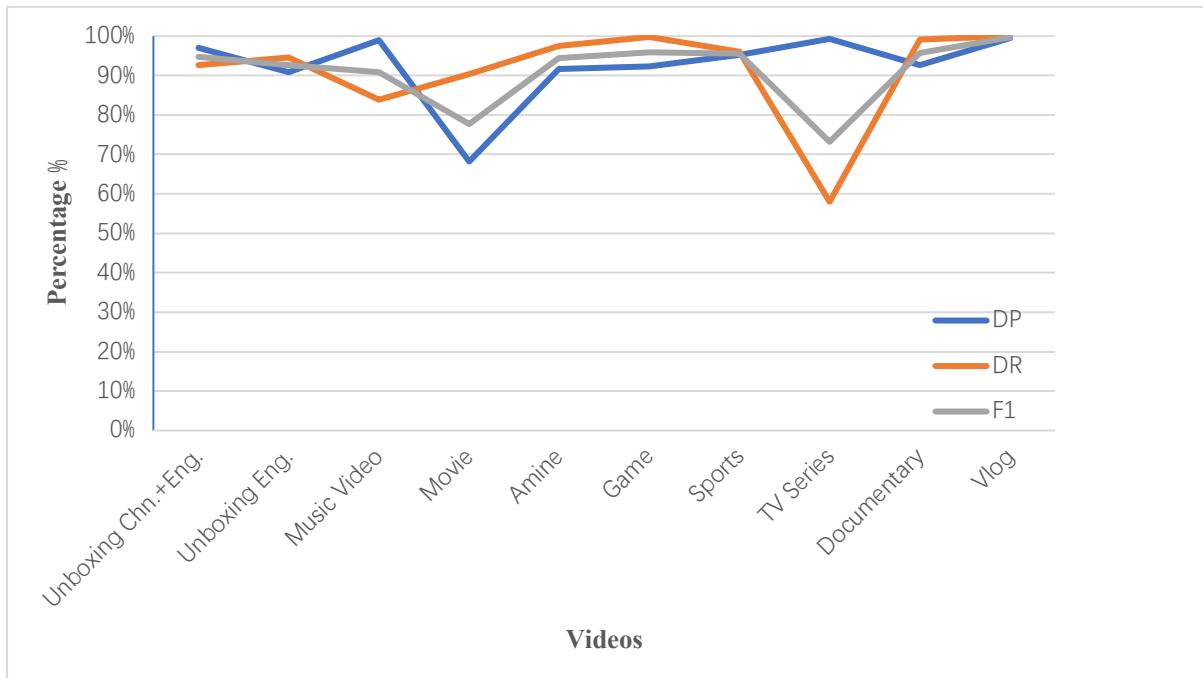


Figure 19: Performance of Ten Different Genre of Videos

It is noticeable from the line graph in Fig. 19 that the performance has fluctuated around different genres especially Movie and TV series. An excellent model is supposed to experience less fluctuation, and this can be further improved by optimising the text detection task. One possible solution is to separate the whole detection task into two subtasks. One task spotlights scene text detection and the other concentrates on subtitle detection. By introducing an additional neural network or training an additional model, this objective could be accomplished. and a more satisfactory performance cloud be achieved.

5.2.2 Result of OCR module

Table 4: The Recognition Statistics of OCR

Input	Results						
	N_t	D_ε	S_ε	I_ε	LD	AR	CR
Video 1	136	2	1	2	5	0.963	0.978
Video 2	426	16	27	0	43	0.899	0.899
Video 3	153	3	7	4	14	0.908	0.935
Video 4	240	3	1	1	5	0.980	0.983
Video 5	293	1	3	2	6	0.980	0.986
Video 6	457	17	16	5	38	0.917	0.928
Video 7	162	5	4	2	11	0.932	0.944
Video 8	284	0	7	3	10	0.965	0.975
Video 9	242	4	13	2	19	0.921	0.930
Video 10	4251	NA	NA	42	306	0.928	0.938
<i>Average</i>	-	-	-	-	-	0.939	0.950

The result of OCR module is shown in Table 4, and further information about the details can be found in Appendix A. By calling the *HtmlDiff* method in python's *difflib* library, we can obtain a difference report between the original subtitle file and the generated subtitle file. The report shows a side by side, line by line distinction of text with inter-line and intra-line change marked with colours. After we analyse the generated report, we can believe that the

system designed for subtitle recognition is efficient, whose average correct rate is over 95%. Video 2, Video 6 and Video 10 are using English subtitles while others uses Chinese subtitles or English-Chinese mixed subtitles. Since the recognition model trained in this project aims at supporting both languages, some mistakes can be identified such that an English letter was mistakenly recognised as a Chinese character. An example is illustrated in Fig. 20.

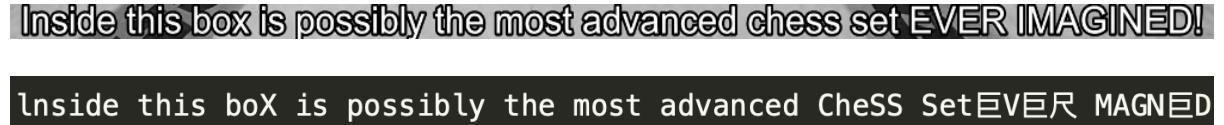


Figure 20: Example of Incorrect Recognition

This is probably the feature of the uppercase English letter 'E' resembles the Chinese character "巨" and 'R' looks like "尺". An improvement to address this issue may be introducing a language correction module using natural language process method. Due to project time constraints, the module was not adapted in the project, and the relative effect was not good. Another issue of the OCR module is that some of numbers and English letters with smaller widths may be dropped, such as "11" and "ll". A possible explanation for this is perhaps the receptive field of CNN has a too wide vision to see the text gap.

5.2.3 Accuracy of Timestamps

The final output of this project is a mountable external subtitle file that matches the original video. Thus, the accuracy of timestamps is also a significant metric. I have computed the average timestamp distance mentioned above and the result is shown in Table 5. It can be inferred the accuracy of timestamp is affected by the frequency of video frame extraction. The higher the frequency of video frame extraction, the more accurate the corresponding timestamps. For the purpose of balancing the speed and accuracy of the subtitle extraction

system, we choose to extract one frame per second. In this case, the average ATD of ten test videos is 1106 milliseconds. Furthermore, if user wants to obtain more accurate timestamps, frequency of frame extraction can be modified. However, higher *FPS* means more computation on the detection and recognition process. Empirically, the average timestamp distance under 1200 milliseconds is acceptable for most kind of videos.

Table 5: The Statistics of Timestamps Accuracy

	Subtitle Lines	Frame Extraction Frequency	ATD
Video 1	15	1fps	498ms
Video 2	6	1fps	1241ms
Video 3	26	1fps	1642ms
Video 4	28	1fps	782ms
Video 5	31	1fps	1623ms
Video 6	16	1fps	823ms
Video 7	14	1fps	743ms
Video 8	17	1fps	1285ms
Video 9	16	1fps	1143ms
Video 10	65	1fps	1280ms
<i>Average</i>	-	1fps	1106ms

5.2.4 Speed of Subtitle Extractor System

Table 6: The Configuration of Testing Computer

Testing Environment	
Model	MacBook Pro 2017 15-inch
CPU	Quad-Core Intel Core i7
Clock Speed	2.8 GHz
RAM	16 GB 2133 MHz LPDDR3
Core	4 Cores
Operating System	MacOS Big Sur Beta 11.0

The speed of extracting subtitles from a video differs with the configuration of testing computer. The testing computer used for evaluating the speed of the designed system is given

in Table 6. I have tested ten videos on the original CTPN method via CPU and thus conclude that the average time of processing a single frame is about 2.5 seconds. A video with 30 seconds duration will cost one and a half minutes. After we adopt heuristic approach to improve the CTPN, the optimised speed of processing a frame will drop to approximately 2 seconds, which dramatically improves the efficiency of detecting subtitle areas by 20%. If it is accelerated with a powerful external GPU, we can get a 5 times enhancement on the performance, reaching a processing speed of 0.4 second per frame.

5.3 Result Summary

So far, the aims and objectives of this project are achieved. An application extracting subtitles in videos has been successfully designed and implemented with an excellent performance on detecting text and recognising two different languages. With one simple input of video, the generation of subtitle file is automatically accomplished without manually adjusting parameters. The original CTPN approach is adopted and efficiently improved for detecting subtitles with its speed advanced. A model of higher recognition rate is trained and achieves a correct rate of over 95%, which is a remarkable result, compared with traditional method. Furthermore, this project provides a novel module of directly generating mountable subtitle files, which other existing systems do not include.

Chapter 6 - Conclusion and Future Work

In this chapter, the conclusion concerning the performance of subtitle extractor is drawn. The method of addressing the research problem is briefly summarised and the current issues and the future improvements of the system is pointed out.

6.1 Conclusion

In this dissertation, a unified subtitle extractor application was implemented to address the issue of data retrieval in videos. This application can meet the needs of enterprises for hard subtitle extraction in real world scenarios, assisting staff in classification and retrieval based on video content. Traditional subtitle extraction methods require users to manually set a series of parameters to extract low level features, which is sophisticated and the correct rate of recognition is not satisfactory. In order to further achieve a better performance, this project utilises deep-learning based text detection method to automatically extract high level features and localise the subtitle areas, and then trains an OCR model to recognise the detected subtitles in English and Chinese. The text detection method has been improved according to the feature of subtitles and the speed of detection is accelerated via adopting heuristic. An ideal architecture of extracting subtitle has been found with CTPN used for subtitle detection and neural network based OCR employed for subtitle recognition. Training datasets have been collected and a testing dataset with ten videos of different genres has been created. Two models were successfully trained with satisfactory performance. From the result, all subtitle areas in the testing video have been correctly detected although there exist FN and FP in detecting scene text. The average F1-score of text detection for the testing videos is above 91% and the correct rate of recognition is about 95%.

6.2 Future Work

The project has designed a system that can extract hard subtitles from videos of different genres with complex backgrounds into separate external files. Although the system simplifies the operation of the extraction, which lowers the threshold for users to learn to use the software, the system is not user friendly enough. On the other hand, the extracted subtitle files have the following problems:

1. The timestamps of the subtitles are not accurate enough, which have an average deviation of about 1100 milliseconds.
2. There are misrecognised characters in the generated subtitle file. In addition, the same strategy as the Chinese recognition is used when recognising English. Thus, all spaces are deleted, which results in the fact that the recognised English sentences have no word spacing.
3. The subtitle recognition module retains the original subtitle position information, which can be further used to generate more advanced subtitle file with effects and location information.
4. Although the input of the implemented program is very simple, it is currently a command line application, which is not user friendly enough.

Based on the listed issues, there are many improvements can be made: 1) By making full use of the audio resource in the video, we can proofread and refine the subtitle timestamps according to the voice frequency. This can be done by extracting the audio, comparing the parts with dramatic voice frequency changes and writing a script to automatically adjust the timestamps. 2) Introduce natural language processing (NLP) methods to correct errors in

subtitle files, and improve the quality of subtitles through grammar and phrase collocation analysis. Separate different words in English sentences with space, or modify the recognition strategy of the OCR module and present an algorithm to make sure the English subtitles are generated with spaces if we want to use one model to recognise both English and Chinese. 3) Explore more advanced subtitle format like *ass*, and use the subtitle position information in the raw file to generate subtitles with special effects. 4) Implement a good user interface, set up independent buttons for each module, so that we can realise a *pause* function of the system. Users can pause after executing a module, and resume the next module after a period of time. In addition, the system currently supports the recognition of Chinese, English and Arabic numerals while more languages can be supported if we collect more datasets, add a neural network of classifying the kind of language, and train more models. In this case, different languages can be assigned into different models and then recognised simultaneously. Another challenging advance of the system is to provide a translation module for the generated subtitle file, which can make full use of the value of subtitles.

Bibliography

- Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer.
- Bridle, J. (1990). Probabilistic interpretation of feed-forward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing: Algorithms, architectures and applications*, pp. 227–236.
- Britz, D. (2015). *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. Retrieved from <http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>
- Chen, X., & Yuille, A. (2004). AdaBoost Learning for Detecting and Recognizing Text. *Computer Vision and Pattern Recognition*, pp. 366-373.
- Cho, K., Merrienboer, B. v., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014, Oct). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724-1734.
- Gao, Y., Chen, Y., Wang, J., & Lu, H. (2017). Reading scene text with attention convolutional sequence modeling.
- Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014). Multi-scale orderless pooling of deep convolutional activation features. *European conference on computer vision* (pp. 392-407). Cham: Springer.
- Goodfellow , I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2014). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). 6.2.2.3 Softmax Units for Multinoulli Output Distributions. In *Deep Learning* (pp. 180-184). MIT Press.
- Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *In Proceedings of the 23rd international conference on Machine learning (ICML '06)*, pp. 369–376.

He, K., Zhang, X., Ren, S., & Sun, J. (2014, September). Spatial Pyramid Pooling in Deep Convolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1904-1916.

Hjelmqvist, S. (2012, March 26). *Fast, memory efficient Levenshtein algorithm*. Retrieved from <https://www.codeproject.com/Articles/13525/Fast-memory-efficient-Levenshtein-algorithm-2>

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, pp. 1735–1780.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261-2269.

Jain, S., & Sohal, H. S. (2014, August). A Novel Approach for Word Segmentation in Correlation based OCR System. *International Journal of Computer Applications*.

Jung, K., Kim, K., & Jainc, A. K. (2004, May). Text information extraction in images and video: a survey. *Pattern Recognition*, pp. 977-997 .

Jurafsky, D., & Martin, J. H. (2019). *Hidden Markov Models*. Retrieved from <https://web.stanford.edu/~jurafsky/slp3/A.pdf>

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017, May). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, pp. 84–90.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pp. 2278-2324.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature Pyramid Networks for Object Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936-944.

Long, S., He, X., & Yao, C. (2019). Scene Text Detection and Recognition: The Deep Learning Era.

Ma, J., Shao, W., Ye, H., Wang , L., Wang , H., Zheng , Y., & Xue, X. (2018). Arbitrary-

- Oriented Scene Text Detection via Rotation Proposals. *IEEE Transactions on Multimedia*, pp. 3111-3122.
- Macêdo, S., Melo, G., & Kelner, J. (2015). A Comparative Study of Grayscale Conversion Techniques Applied to SIFT Descriptors. *SBC Journal on Interactive Systems*.
- Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.
- Ofcom. (2006). *Television Access Services*. Retrieved from
https://www.ofcom.org.uk/__data/assets/pdf_file/0016/42442/access.pdf
- Ofcom. (2020). *Review of Ofcom's Code on Television Access Services and Guidance on BBC Accessibility*. Retrieved from
https://www.ofcom.org.uk/__data/assets/pdf_file/0035/179963/consultation-access-services-code-bbc-accessibility.pdf
- Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved from
<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Ramani, A., Rao, A., V, V., & Prasad, V. B. (2020). Automatic Subtitle Generation for Videos. *2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS)*.
- Sankhe, A., Patil, V., Patel, S., & Bhagat, S. (2018). Video based Subtitle Generation. *International Journal for Research in Applied Science & Engineering Technology*.
- Senlinuc. (n.d.). Retrieved from Synthetic Chinese dataset:
<https://pan.baidu.com/s/1TXeZUcpVOUxrtfRtWKMSGg>
- Shafkat, I. (2018, Jun). *Intuitively Understanding Convolutions for Deep Learning*. Retrieved from <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>
- Shi, B., Bai, X., & Yao , C. (2017, Nov). An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2298-2304.
- Shi, B., Wang, X., Lyu, P., Yao, C., & Bai, X. (2016). Robust Scene Text Recognition with

- Automatic Rectification. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4168–4176.
- Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv*.
- Singh, M., & Kaur, A. (2015). An Efficient Hybrid Scheme for Key Frame Extraction and Text Localization in Video. *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1250-1254.
- Song, Y., & Wang, W. (2009). Text Localization and Detection for News Video. *2009 Second International Conference on Information and Computing Science*, pp. 98-101.
- Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). *Unsupervised Learning of Video Representations using LSTMs*.
- Standford, U. (n.d.). Retrieved from Convolutional Neural Networks (CNNs / ConvNets):
<https://cs231n.github.io/convolutional-networks/>
- Tian, S., Yin, X., Su, Y., & Hao, H. (2018, March). A Unified Framework for Tracking Based Text Detection and Recognition from Web Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 542-554.
- Tian, Z., Huang, W., He, T., He, P., & Qiao, Y. (2016). Detecting Text in Natural Image with Connectionist Text Proposal Network. *Computer Vision and Pattern Recognition*.
- Wikipedia. (2015). *Levenshtein distance*. Retrieved from
https://en.wikipedia.org/wiki/Levenshtein_distance#cite_note-1
- Wojna, Z., Gorban, A., Lee, D.-S., Murphy, K., Yu, Q., Li, Y., & Ibarz, J. (2017). Attention-Based Extraction of Structured Information from Street View Imagery. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, (pp. 844-850).
- Xu, M., Clauss, C., & Yang, L. (2020). *pycorrector*. Retrieved from
<https://github.com/shibing624/pycorrector>
- Yan, H., & Xu, X. (2020, March). End-to-end video subtitle recognition via a deep Residual

Neural Network. *Pattern Recognition Letters*, pp. 368-375.

Yang, Q., Cheng, M., Zhou, W., Chen, Y., Qiu , M., Lin, W., & Chu, W. (2018). IncepText: A New Inception-Text Module with Deformable PSROI Pooling for Multi-Oriented Scene Text Detection. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Zhao, M., Li, S., & Kwok, J. (2010, December). Text detection in images using sparse representation with discriminative dictionaries. *Image and Vision Computing*, pp. 1590-1599.

Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). EAST: An Efficient and Accurate Scene Text Detector. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2642-2651.

Appendices

Appendix A: Comparison between Original and Generated SRT File

The demo video of subtitle extractor can be found via <https://youtu.be/0gq8FQHb448>.

Testing Video 1:

Unbox ring fit adventure	
Genre	Unboxing Video
Subtitle Language	Chinese + few English
FPS	25.0
Total Frames	920
Video Resolution	(1080, 1920)

Original		Generated	
<u>f</u>	1因为你只要看最高的记录	<u>f</u>	1因为你只要看最高的记录
	2然后去打破它就OK了		2然后去打破它就OK了
<u>n</u>	3还有一个技术 <u>上</u> 的问题	<u>n</u>	3还有一个技术的问题
	4Joy-Con是真的牛皮		4Joy-Con是真的牛皮
<u>n</u>	5上次玩LABO我就惊叹不已	<u>n</u>	5上次玩 <u>L</u> ABO我就惊叹不已
	6这次不但 <u>承担</u> 起了整个健身环的运动判定		6这次不但 <u>担</u> 起了整个健身环的运动判定
	7它居然还有心率检测		7它居然还有心率检测
<u>n</u>		<u>n</u>	8 <u>测量中</u>
			9就增加运动强度吧
	8可是呢		10可是呢
	9它的技术还是没能完美的检测你的运动		11它的技术还是没能完美的检测你的运动
	10比如抬手压环的动作		12比如抬手压环的动作
<u>n</u>	11你可以不用 <u>抬手</u>	<u>n</u>	13你可以不用 <u>抬手</u>
	12它是检测不到的		14它是检测不到的
<u>t</u>	13这样偷懒的方式有很多	<u>t</u>	15这样偷懒的方式有很多—
	14但是大兄弟		16但是大兄弟
	15你这都要摸鱼的话		17你这都要摸鱼的话

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=fqMoEk5Yb-A>

Testing Video 2:

What Magic Is This?	
Genre	Unboxing Video
Subtitle Language	English
FPS	23.976
Total Frames	747
Video Resolution	(1080, 1920)

Original:

```

n 1 Today... I might very well blow your mind! And my own? Possibly Jack's.
n 2
n 3 This is a very exiting moment there on unbox therapy because...
n 4
n 5 When I heard about this thing, this in front of me,
n 6
n 7 you see it's... large. I responded immediately
n 8
n 9 to the email. I was like "Send this thing to me!""
n 10
n 11 Because what I have in here... is some tech. It's almost MAGICAL.
n 12
t 13 Inside this box is possibly the most advanced chess set EVER IMAGINED!

```

Generated:

```

n 1 Today. 6 might Very Well blow your mind And my own Possibly JackS.
n 2
n 3 This is a very exiting moment there on unbox therapy becaSeo
n 4
n 5 When heard about this thing, this in front of me,
n 6
n 7 you see itsry. large. responded immediately
n 8
n 9 to the email. I was likeSend this thing to me
n 10
n 11 Because what l have in here.. issome tech. l salmost MAGCA
n 12
t 13 Inside this boX ispossibly the most advanced CheSSset █v █尺 MAGN █D

```

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=XBjEU1bJd5U>

Testing Video 3:

Yi Jian Mei	
Genre	Music Video
Subtitle Language	Chinese
FPS	29.970
Total Frames	6775
Video Resolution	(1080, 1920)

	Original	Generated
n	1 剪梅-费玉清	1 剪梅-费玉清
	2 真情像草原广阔	2 真情像草原广阔
n	3 层层风雨不能阻隔	3 层层风雨不 不 能阻隔
	4 总有云开日出时候	4 总有云开日出时 候
	5 万丈阳光照耀你我	5 万丈阳光照耀你我
	6 真情像梅花开过	6 真情像梅花开过
n	7 冷冷冰雪不能掩没	7 冷冷冰雪不能掩没 滚
	8 就在最冷	8 就在最冷
	9 枝头绽放	9 枝头绽放
	10 看见春天走向你我	10 看见春天走向你我
	11 雪花飘飘北风啸 啸	11 雪花飘飘北风啸 肃
	12 天地一片苍茫	12 天地一片苍茫
n	13 剪寒梅	13 剪寒梅
	14 傲立雪中	14 傲立雪中
	15 只为伊人飘香	15 只为伊人飘香
	16 爱我所爱无怨无悔	16 爱我所爱无怨无悔
	17 此情长留	17 比情长留
t	18 心间	18 八间
	19 雪花飘飘北风啸 啸	19 雪花飘飘北风啸 噩
	20 天地一片苍茫	20 天地一片苍茫
n	21 剪寒梅	21 剪寒梅
	22 傲立雪中	22 傲立雪中
	23 只为伊人飘香	23 只为伊人飘香
	24 爱我所爱无怨无悔	24 爱我所爱无怨无悔
	25 此情长留	25 此情长留
	26 心间	t 26 八心间

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: https://www.youtube.com/watch?v=dvi_KVo2xOc

Testing Video 4:

JOKER - Final Trailer	
Genre	Movie
Subtitle Language	Chinese
FPS	23.976
Total Frames	3456
Video Resolution	(1080, 1920)

Original

Generated

n 1啊	n 1请你别再打扰我的孩子了吗
2请你别再打扰我的孩子了吗	2抱歉
3抱歉	3亚瑟
4亚瑟	4我有个坏消息要给你
5我有个坏消息要给你	5这是我们最后一次见面
6这是我们最后一次见面	6你不听吗
7你不听吗	7您每个星期都问同样的问题
8您每个星期都问同样的问题	8你工作怎么样
9你工作怎么样	9您有负面想法吗
10您有负面想法吗	10我所有的都是消极的想法
11我所有的都是消极的想法	11在每个人都认为自己可以胜任工作的世界中,看看这个家伙
n 12在每个人都认为自己可以胜任工作的世界中,看看这个家伙	12我小时候
13我小时候	13告诉人们我要当喜剧演员
14告诉人们我要当喜剧演员	14每个人都嘲笑我
15每个人都嘲笑我	15好吧,现在没有人在笑
n 16好吧,现在没有人在笑	16薏你可以再说一遍,朋友
17你可以再说一遍,朋友	17太可怕了吧
18太可怕了吧	18在我的一生中
19在我的一生中	19我不知道我是否真的存在
20我不知道我是否真的存在	20但是我愿意
21但是我愿意	21人们开始注意到
22人们开始注意到	22你觉得这很有趣吗
23你觉得这很有趣吗	23哈哈哈哈哈
24哈哈哈哈哈	t 24-这是你的玩笑吗-哈哈哈哈哈
t 25-这是你的玩笑吗-哈哈哈哈哈	25嗯.很抱歉,一件小事
26嗯...很抱歉,一件小事	26-是吗-当你带我出去时,
27-是吗-当你带我出去时,	27可以介绍我为小丑吗
28可以介绍我为小丑吗	

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=zAGVQLHvwOY>

Testing Video 5:

The Beauty Of Your Name (Kimi no na wa)	
Genre	Amine
Subtitle Language	Chinese
FPS	23.976
Total Frames	4698
Video Resolution	(1080, 1920)

Original	Generated
<p>f 1看来这个世界 2似乎想要驯服我的样子呢 3这样正合我意 4我会美丽地挣扎一番 5让我们就这样</p> <p>n 6对望着彼此手中的沙漏亲吻 7让我们相约在 8离再见最遥远的地方会合 9我憎恨着</p> <p>n 10这个由字典中的辞藻组成的世界 11万花筒中映照着8月的某个早晨 12你就站在我的面前 13明明脉脉含羞却又故作淡然</p> <p>n 14展露这世界上教科书一般的笑容 15而今这一刻终于到来 16昨天为止只是序章的前言 17所以粗略地跳读就好了 18现在开始轮到我的章节了</p> <p>n 19woooo 20我带着迄今的经验和知识 21以及那锈迹斑斑的勇气</p> <p>t 22以前所未有的速度飞奔到你的身边 23朦胧睡意中 24透过微温的可乐 25我梦见了不曾见过的风景 26那是在教室的窗外 27以及随着电车的运行而颠簸的清晨 28就连表达爱的方式 29也弥漫着你的气息 30就连走路的方式 31也萦绕着你的笑声</p>	<p>f 1看来这个世界 2似乎想要驯服我的样子呢 3这样正合我意 4我会美丽地挣扎一番 5让我们就这样</p> <p>n 6对望着彼此手申的沙漏亲吻 7让我们相约在 8离再见最遥远的地方会合 9我憎恨着</p> <p>n 10这个由字典中的辞藻组成的世界 11万花筒中映照着8月的某个早晨 12你就立站在我的面前 13明明脉脉含羞却又故作淡然</p> <p>n 14展露这世界上教科书一般的笑容 15而今这一刻终于到来 16昨天为止只是序章的前言 17所以粗略地跳读就好了 18现在开始轮到我的章节了</p> <p>n 19Woooo 20我带着迄今的经验和知识 21以及那锈迹斑斑的勇气</p> <p>t 22以前所未有的速度飞奔到你的身边 23朦胧睡意中1 24透过微温的可乐 25我梦见了不曾见过的风景 26那是在教室的窗外 27以及随着电车的运行而颠簸的清晨 28就连表达爱的方式 29也弥漫着你的气息 30就连走路的方式 31也萦绕着你的笑声</p>

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=OJAvgHSeIhI&t=23s>

Testing Video 6:

ASSASSIN'S CREED ODYSSEY: Final Trailer (2018)	
Genre	Game
Subtitle Language	English
FPS	25
Total Frames	2875
Video Resolution	(1080, 1920)

Original	Generated
<pre> n 1TheGodshavespoken. 2 n 3Youshallneverleavethisisland 4 n 5thatyoucallhome. 6 7Youshallneverstepfoot 8 n 9outofyourownvillage, 10 n 11todareandtraveltheworld. 12 n 13Youshallliveasmall, 14 15tediouslife, 16 n 17inthecomfortofyourhome. 18 19Afterall, 20 21foreveryhero, 22 n 23theworldneedscowardslikeyourself. 24 25Foryourentireexistence,youshall 26 27neverhavetofacedanger, 28 n 29tofightforyourownlife. 30 n 31Noryouwillknowcourage, 32 33boldness 34 n 35orbravery. 36 37Andintheend,alone, 38 39youshallvanish, 40 41forgottenforeternity. 42 n 43ThatisthewilloftheGods 44 t 45andyourdestinyshallbefullfi... </pre>	<pre> n 1TheGodshaveSpoken. 2 n 3YoushantereaveVethisisland 4 n 5thatYouCahome 6 7Youshallneverstepfoot 8 n 9outofyourOwnvillage, 10 n 11todareandtraveltheWrld 12 n 13Y0ushallliveaSmall, 14 15tediouslife 16 n 17inthecomfortofYourhome. 18 19Afterall, 20 21foreveryhero, 22 n 23theworldneedScoWardSikeyourSelf 24 25Foryourentireexistence,youshall 26 27neverhavetofacedanger, 28 n 29tofightforyourOwnlife 30 n 31Norwilyouknowcourage, 32 33boldness 34 n 35orbraveLry. 36 37Andintheend,alone, 38 39youshallvanish, 40 41forgottenforeternity. 42 n 43ThatisthewiloftheGodS 44 t 45andyourdestinyshallbefullfi. </pre>

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=Md-uL6opUFY>

Testing Video 7:

CHAMPIONS! Liverpool's Premier League trophy lift	
Genre	Sports
Subtitle Language	Chinese
FPS	25.0
Total Frames	3002
Video Resolution	(1080, 1920)

	Original	Generated
n	1队长来了！球队的领袖 2Jordan向前领取他的奖牌 3以及捧起英超金杯 4Hendo一路走来 n 5首先，他们征服了欧洲 6接着他们称霸世界 7现在他们已经在自己的国家称王 n 8Henderson,利物浦的队长 9只创造历史、打破纪录的球队 10现在利物浦回到属于他们的地方 t 11John, 30年的痛现在要画下句点了 12再一次的冠军！利物浦！ 13享受一切吧伙伴们！ 14你永不独行放送	n 1队长来了球队的领袖 2Jordlan向前岭领取他的奖牌 3以及捧起英超金杯 4Hendo一路走来 n 5首先，他们征服了欧洲 6接着他们称霸世界 7现在他们已经在自己的国家称王 n 8Henderson,利物浦的队长 9-只创造历史打破纪录的球队 10现在利物浦回到属于他们的地方 t 11John, 30年的痛现在要画下句点了 12再一次的冠军利物浦 13享受一切吧伙伴们 14你永不独行放送

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=UAixsyx70AM>

Testing Video 8:

CHERNOBYL Official Trailer (2019)	
Genre	TV Series
Subtitle Language	Chinese
FPS	29.970
Total Frames	4355
Video Resolution	(1080, 1920)

Original

f 1我们做的一件好事都没关系	f 1我们做的一件好事都没关系
n 2重要的是,对对他们来说,正义已经完成,看到正义的世界是理智的	n 2重要的是,对对他们来说,正义已经完成,看到正义的世界是理智的
3世界	3世界
4切尔诺贝尔没有什么理智的	4切尔诺贝尔没有什么理智的
5我很高兴在此报告这种情况	5我很高兴在此报告这种情况
6切尔诺贝尔在辐射方面很稳定,我被告知这是等效的	6切尔诺贝尔在辐射方面很稳定,我被告知这是等效的
7x光检查没有切尔诺贝尔着火	7x光检查没有切尔诺贝尔着火
8铀的每个原子都像子弹一样穿透了它的路径	8铀的每个原子都像子弹一样穿透了它的路径
9金属混凝土肉现在切尔诺贝尔拥有超过三万亿枚这些子弹	9金属混凝土肉现在切尔诺贝尔拥有超过三万亿枚这些子弹
n 10他们中的一些人将不会停止射击50,000年,告诉我该如何放置	n 10他们中的一些人将不会停止射击50,000年,告诉我该如何放置
11您正在处理这个星球上从未发生过的事情	11您正在处理这个星球上从未发生过的事情
12切断电话线包含错误信息的传播	12切断电话线包含错误信息的传播
13我们的男孩三天之内难以想象的痛苦	13我们的男孩三天之内难以想象的痛苦
n 14到三个星期,你看不到,不能碰他,你明白吗	n 14到三个星期,你看不到,不能碰他,你明白吗
15那天晚上发生了什么	15那天晚上发生了什么
t 16意外错过正确的问题,我们会给你真相,没有	t 16意外错过正确的问题,我们会给你真相,没有
17真相那里发生了什么事	17真相那里发生了什么事

Generated

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=Rle1Bywi6IM>

Testing Video 9:

Hiroshima: Dropping The Bomb - Hiroshima - BBC	
Genre	Documentary
Subtitle Language	Chinese
FPS	25.0
Total Frames	6314
Video Resolution	(480, 854)

Original

f 1我没看过笑话
n 2我会用广角上方的小飞机像银落的小飞机
 3Sandip我立即将其识别为美国飞机, 没有日本飞机
 4可以在那个最可爱的时候飞到那个高度
 5飞机, 所以我以为它像往常一样经过
 6我很满足, 例如妮莎
 7尖叫去做谁, 我擦了炸弹被炸毁时的桌面
 8掉了
n 9从炸弹离开飞机到飞机停飞, 我花了大约45秒钟
 10爆炸了, 我认为飞机上没有一个人
 11他的手表计时或计数或做某事我确定炸弹是
 12我肯定不会工作的傻瓜
n 13跌落43秒后, 时间和气压触发开始
 14发射装置伊朗子弹将一桶子弹发射到铀桶中
 15他们一起开始了核链反应, 固体物质开始出现
 16释放出无限量的能量

Generated

f 1我没看过笑话
n 2我会甩广角上方的小飞机像银落的小飞机
 3Sandip我泣立即将其识别伪美国机, 没有日本飞机
 4可以在那个最爱的时候飞到那个高度
 5飞机, 所以我以为它像往常一样经过
 6我很满足, 例如妮莎
 7尖叫去做谁, 我擦了炸弹被炸毁时的桌面
 8掉了
n 9从炸弹离开飞机到飞机停飞, 我花了大约45秒钟
 10爆炸了, 我认为飞机上没有一个公
 11他的手表计时或计数或做某事我确定炸弹是
 12我肯定不会工作的傻瓜
n 13跌落48秒后, 时间和气压触发开始
 14发射装置伊朗子弹将桶子弹发射到铀桶中
 15他们起采始了核链反应, 固体物质采始出现
 16释放出无限量的能量
t 17who help fund new BBC programmes,

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=3wxWNAM8Cso>

Testing Video 10:

Vlog: Best places to go in Nottingham city centre	
Genre	Vlog
Subtitle Language	English
FPS	50.0
Total Frames	13670
Video Resolution	(1080, 1920)

Original (part)	Generated (part)
<pre> 1 Hey my name's Jodie. I'm in my third year 2 3 History and Politics student and today I'm 4 5 going to be talking to you about the areas in 6 7 Nottingham that you need to know in order to 8 9 get a better idea of what the city is like 10 11 and hopefully you can get a better feel of 12 13 what it's like to be a student at the University of Nottingham. 14 15 So, I've got my laptop here, and I'm just 16 17 going to get straight into it because I have 18 19 quite a lot to get through. 20 21 So, the first area which we're going to talk 22 23 about is Old Market Square, which is the centre 24 25 of the city. 26 27 Old Market Square is actually where a lot 28 29 of events are held by the city council and 30 31 it's also where the town hall is actually located. 32 33 Two events that come to mind straight away when 34 35 I think of Old Market Square is Old Market Scare, 36 37 an event put on during Halloween and then 38 39 also Christmas time there is an event called 40 41 Winter Wonderland which happens every single year. 42 43 I'm sure if you ask any student at the Uni 44 45 they've either heard of it or they've been. 46 47 So that is something definitely to checkout 48 49 and keep in mind and find out what's going 50 51 on in the Square. 52 53 It's also a really good location to start off 54 55 at because from Old Market Square you 56 57 can go to the shops you can go to restaurants, 58 59 you can go to bars. You can basically do anything 60 61 and everything. 62 63 So literally at two minute walk away from the Old 64 65 Market Square you have the Victoria Centre </pre>	<pre> 1 Hey my name's Jodie. mathirdyear 2 3 History and Politics student and today I'm 4 5 going to be talking to you about the areas in 6 7 Nottingham that you need to know in order to 8 9 get a better idea of what the city is like 10 11 and hopefully you can get a better feel of 12 13 what it's like to be a student at the University of Nottingham. 14 15 So, vegotmylaptohere, and I'm just 16 17 going to get straight into it because I have 18 19 So, the first area which we're going to talk 20 21 about is Old Market Square, which is the centre 22 23 of the city. 24 25 Old Market Square is actually where a lot 26 27 of events are held by the city council and 28 29 it's also where the town hall is actually located. 30 31 Two events that come to mind straight away when 32 33 I think of Old Market Square is Old Market Scare, 34 35 an event put on during Halloween and then 36 37 also Christmas time there is an event called 38 39 Winter Wonderland which happens every single year. 40 41 I'm sure if you ask any student at the Uni 42 43 they've either heard of it or they've been. 44 45 So that is something definitely to checkout 46 47 and keep in mind and find out what's going 48 49 On in the Square. 50 51 at because from Old Market Square you 52 53 It's also a really good location to start off 54 55 at because from Old Market Square you 56 57 can go to the shops you can go to restaurants, 58 59 you can go to bars. You can basically do anything 60 61 and everything. 62 63 So literally at two minute walk away from the Old 64 65 Market Square you have the Victoria Centre </pre>

Legends

Colors	Links
Added	(f)irst change
Changed	(n)ext change
Deleted	(t)op

Source: <https://www.youtube.com/watch?v=PLtd-uP9gKE>

Appendix B: Source Code Hierarchy

```
video-subtitle-extractor
├── checkpoints_mlt -- TensorFlow model trained by me based on ICDAR 2017 MLT[1]
│   ├── checkpoint
│   ├── ctpn_50000.ckpt.data-00000-of-00001 -- model file
│   ├── ctpn_50000.ckpt.index
│   └── ctpn_50000.ckpt.meta
├── data -- program output
│   ├── frames -- output of the pre-processing module
│   ├── text_position -- output of the CTPN module
│   ├── to_ocr -- output of the CTPN module
│   └── to_srt
│       ├── to_srt.txt -- raw output of OCR module
│       └── to_srt.srt/generated.srt -- output of the post-processing module
├── ipynb
│   ├── Accuracy Evaluation.ipynb -- python notebook written from scratch
│   ├── GPU speed up.ipynb -- written from scratch
│   ├── image localisation.ipynb -- written from scratch
│   ├── OCR.ipynb -- written from scratch
│   ├── opencvsubtitles ver1.ipynb -- written from scratch
│   ├── subtitle area detection.ipynb -- written from scratch
│   └── subtitle generation.ipynb -- written from scratch
├── main -- main code of the project
│   ├── accuracyCal.py -- written from scratch
│   ├── demo.py -- written from scratch
│   └── train.py -- code adopted from eragonruan[2]
├── model -- model file trained by me with referencing daomin's[3] code
│   └── ocr -- MXNet model based on MJSynth dataset[4] and SCD[5]
├── nets -- Backbone Network of CNN
│   ├── model_train.py -- code adopted from eragonruan
│   └── vgg.py -- code adopted from eragonruan
└── utils -- Third party code
    ├── bbox
    ├── dataset
    ├── prepare
    └── text_connector
```

Testing Dataset -- Dataset created and collected by me.

Testing Results -- Testing outputs and reports

[1] ICDAR 2017 MLT: <https://rrc.cvc.uab.es/?ch=8&com=downloads>

[2] Eragonruan's code: <https://github.com/eragonruan/text-detection-ctpn>

[3] Daomin's code: <https://github.com/diaomin/crnn-mxnet-chinese-text-recognition>

[4] MJSynth Dataset: <http://www.robots.ox.ac.uk/~vgg/data/text/>

[5] SCD: <https://pan.baidu.com/s/1TXeZUcpVOUxrtfRtWKMSGg> (Token: ujjc)