

Distinguishing genuine and posed smiles using computer vision deep learning approaches

Liwei Hou
u6343089

A report submitted for the course
COMP4560 Advanced Computing Project
Supervised by: Dr. Md Zakir Hossain, Mr. Moshiur
Farazi and Prof. Tom Gedeon
The Australian National University

October 2019

© Liwei Hou 2019

Except where otherwise indicated, this report is my own original work.

Liwei Hou
25 October 2019

Acknowledgments

Firstly, I would like to pay my authentic thankfulness to Prof. Tom Gedeon, Dr. Md Zakir Hossain, Mr. Moshiur Farazi and Mr. Shafin Rahman who are the supervisors of my individual project for their sincere and selfless guidance, valuable help, and patient answers to all my questions. Without their help, it was impossible for me to complete the project and finish this thesis in appropriate standard. I would like to sincerely thank Zakir for his enthusiasm, sincere friendship, helping me applying for the access to ANU NCI (National Computational Infrastructure) and patiently writing letters of recommendation for me. I would also like to thank Moshiur and Shafin for their enthusiasm, selfless helps and inspiring ideas, even though we did not have a long time to get along with each other. Besides, I would like to thank Tom for helping me applying for the access of the GPU cluster server, patiently providing help, delicious cakes and the warm tea party.

At the same time, I would like to show my authentic thankfulness to my parents for their meticulous care, careful cultivation and patient teaching. I would also like to show my special thanks to my girlfriend Tongtong Meng for her sincere love, meticulous care and companionship. Without them, my study and life in Australia will not be so colourful.

Then, I would like to show my authentic gratitude to my seniors Yue Yao, Tianyu Wang and Yuchen Li, and my partners Haowen Li, Xiang Li and Jialin Yang, for their sincere friendship, selfless help and valuable advices. Besides, I would also like to thank my friends Haoran Qu, Helai Bai and Zhou Shen, for every happy day we spent together. Finally, I would also like to thank my close friends in Sea of Miracles, the A Capella Chorus, for every colourful day, every hilarious gathering, every happy rehearsal and every excellent performance.

Abstract

Distinguishing between spontaneous and deliberate smiles is an important direction in the field of computer vision to understand and analyse human expression signals. In this thesis, an automated approach of distinction between spontaneous and deliberate smiles from videos without any manual input is developed. Combination of end-to-end deep learning architecture and face detection, alignment and cropping is applied in this approach. A complete series of processes for facial expression recognition in video from data pre-processing to prediction is designed and implemented. Multiple experiments are designed to evaluate the system. In addition, the factors that influence the ability of designed approaches to distinguish between spontaneous and deliberate smiles are discussed and analysed.

Contents

Acknowledgments	iii
Abstract	v
1 Introduction	1
1.1 Problem Statement	1
1.2 Motivations	1
1.3 Contributions	2
1.4 Project Scope	2
1.5 Report Outline	2
2 Background and Related Work	3
2.1 Background	3
2.1.1 Deep Learning	3
2.1.2 Convolutional Neural Network	3
2.1.3 Recurrent Neural Network	4
2.2 Related work	5
2.2.1 Classifications on Videos	5
2.2.2 distinguishing between spontaneous and deliberate smiles	6
2.3 Summary	7
3 Design and Implementation	9
3.1 Data description	9
3.2 Design	9
3.2.1 Overview	10
3.2.2 Data pre-processing	10
3.2.3 Convolutional Recurrent Neural Network	12
3.2.4 Two-level cross validation	13
3.3 Implementation	16
3.3.1 Overview	16
3.3.2 Frame extraction	16
3.3.3 Data pre-processing	16
3.3.4 Convolutional Recurrent Neural Network	16
3.4 Summary	17

4	Experimental Methodology	19
4.1	Overview	19
4.2	Data pre-processing	19
4.2.1	Basic pre-processing	19
4.2.2	Face detection, alignment and cropping	19
4.3	Method of unifying video length	20
4.3.1	Use the first 90 frames	20
4.3.2	Upsampling	20
4.3.3	Dimensional compression and replication	20
4.3.4	Zero-padding	20
4.4	Summery	21
5	Results	23
5.1	Overview	23
5.2	Data pre-processing	23
5.3	Method of unifying video length	23
5.4	Discussion	25
5.4.1	CRNN	25
5.4.2	Pretrained model	25
5.4.3	Video length	25
5.4.4	Face alignment	25
5.5	Summary	26
6	Conclusion	27
6.1	Challenges	27
6.1.1	Hardware limitation	27
6.1.2	Zero-padding	28
6.2	Future Work	28
	Bibliography	29
6.3	Project Title	31
6.4	Learning Objectives	31
6.5	Project Description	31
6.6	Code Outline	37
6.7	Copyright	37
6.8	Description	37
6.9	Environment Prerequisites	37
6.10	Environment Prerequisites	39
6.11	How to obtain the data	39
6.12	Workflow	39

List of Figures

2.1	Relationships between Artificial Intelligence (AI), Machine Learning (ML), Neural Network (NN) and Deep Learning (DL).	4
2.2	The structure of a simple Convolutional Neural Network (CNN). Convolutional layers and pooling layers alternate in the feature extraction layer, and the extracted feature vectors are fed to the Feed-forward neural network for classification.	5
2.3	The structure of simple Recurrent Neural Network (RNN). The output of the hidden neuron is used along with the regular input as the input to the hidden neuron at the next time step.	6
3.1	Sample frames from the UvA-NEMO Smile database [Dibeklioglu et al., 2012]. Five sample subjects shown in different columns. The top row shows the neutral faces, spontaneous smiles are shown in the medium row, and the bottom row shows the deliberate smiles.	10
3.2	The overview of work flow for training the classifier model for classification.	11
3.3	Sample frames after face detection, cropping and alignment. The outer eyes and the nose in each frame are aligned to the same position, so that most of the significant features on each face are aligned to similar positions.	12
3.4	The overview of workflow for Convolutional Recurrent Neural Network. The visual features in each frame is extracted by the convolutional layer, and then the temporal features of the extracted feature vectors are learned by the recurrent layer. Finally, the output of the recurrent layer is fed to the fully-connected layer for learning or prediction.	14
3.5	The work flow of two-level cross-validation.	15

List of Tables

5.1	The experiment results.	24
-----	---------------------------------	----

Introduction

This chapter gives the introduction of the project. The problem that this project intends to solve is stated in section 1.1. Then, section 1.2 states the motivations of this research project. The contribution of this project is given in section 1.3. Section 1.4 gives the project scope, and section 1.5 lists the outline of this project thesis.

1.1 Problem Statement

In this project, I intend to develop an end-to-end method to automatically distinguish smile between spontaneous and deliberate from normalised videos using computer vision deep learning approaches. The smile videos used in this project are from UvA-NEMO Smile dataset [Dibeklioglu et al., 2012]. Compared to related works, the developed method will require no manual input and automate feature extractions and predictions on raw data.

1.2 Motivations

Human facial expressions are an important way to convey emotional and nonverbal communication, and smile is one of the most basic facial expressions. Understanding and interpreting human facial expressions may revolutionise the way human-computer interactions occur. Therefore, in the field of artificial intelligence, automatic analysis of human expression is an important research direction.

In the recognition of expressions, it is necessary to distinguish spontaneous and posed expressions. Because of complex human emotions and different contexts, spontaneous and posed expressions may convey or imply completely different emotions. In spontaneous analysis, smile is the most important expression because it is the most frequent expression [Dibeklioglu et al., 2012]. Based on context, a spontaneous smile can convey a lot of information, such as happiness, appreciation, enthusiasm, intimacy, etc. Conversely, a deliberate smile may mean embarrassment, ridicule, etc. Because it is the easiest expression to deliberately pose, it is often used to mask the real emotions [Ekman et al., 1988].

The UvA-NEMO Smile database [Dibeklioglu et al., 2012] provides a set of standardised video data for distinguishing between spontaneous and deliberate smiles,

including the use of consistent illumination, the same facial angle and position, etc., and artificially controlling the spontaneous or deliberate smile of the participants. In works related to this database, a common method is to manually place landmarks in the first frame and track the movement of these landmarks on the timeline to extract visual features [Dibeklioglu et al., 2015][Mandal and Ouarti, 2017]. However, this kind of method requires a lot of manual input, which causes limitations in practical applications. Therefore, it is necessary to propose a method to automate the process of distinguishing between spontaneous and deliberate smiles.

1.3 Contributions

The main contributions made in this research includes:

- Developed a method of automating the distinction between spontaneous and deliberate smiles from videos without any manual input.
- Designed a series of processes for facial expression recognition in video from data pre-processing to prediction.
- Discussed and analysed the factors that influence the ability of designed methods to distinguish between spontaneous and deliberate smiles.

1.4 Project Scope

In this project, a neural network model capable of automatically extracting visual features from raw video and distinguishing between spontaneous and deliberate smiles was constructed and trained using the CRNN structure, where ResNet-152 [He et al., 2015] and Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] are combined to construct the model. To improve the quality of the data, multiple data pre-processing methods are applied to automatically detect, align and crop human faces in videos for data normalisation and enhancement. Multiple experiments are designed and carried out to evaluate the performance of the methods.

1.5 Report Outline

This report is divided into six chapters. Chapter 1 gives the basic introduction of this project. The background necessary for reading this thesis is given in Chapter 2. Chapter 3 gives the high level design and detailed implementation of the method. Experimental methodology is describe in Chapter 4. The experiment results are given in Chapter 5, and Chapter 6 gives the conclusion of the project and states the challenges and future works of this project.

Background and Related Work

Section 2.1 gives necessary background materials for reading this report, and Section 2.2 gives the related works about this report.

2.1 Background

2.1.1 Deep Learning

Deep learning originated from artificial neural networks, a branch of machine learning [Alom et al., 2019]. The relationships between deep learning, neural network and machine learning is illustrated in fig. 2.1 [Alom et al., 2019]. Deep learning can be divided into supervised, semi-supervised and unsupervised, where the deep learning approaches discussed in this project are all belong to supervised learning. Compared with traditional machine learning, by applying feature extraction algorithm to manually produce feature values and applying them to train models, deep learning approaches automatically performs feature extraction and hierarchical representation at multiple levels. This gives deep learning a natural advantage over traditional machine learning in feature extraction. In addition, deep learning is a generic learning approach that is non-task-specific and can solve almost all kinds of problems in different application fields [Bengio, 2009].

The branches of deep learning include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Generative Adversarial Networks (GAN), etc., among which the first two approaches are applied in this project. In addition, Convolutional Recurrent Neural Network (CRNN) used in this project is a hybrid deep learning approach that combines the features of CNN and RNN to enable the neural network to automatically extract features from videos and conduct classifications.

2.1.2 Convolutional Neural Network

The concept of convolutional neural networks (CNN) was first proposed in 1988 [Fukushima, 1988]. CNN's structure is similar to human visual processing systems. It hierarchically represents the information in the image and extracts visual features to "see" and "understand" 2D or 3D image.

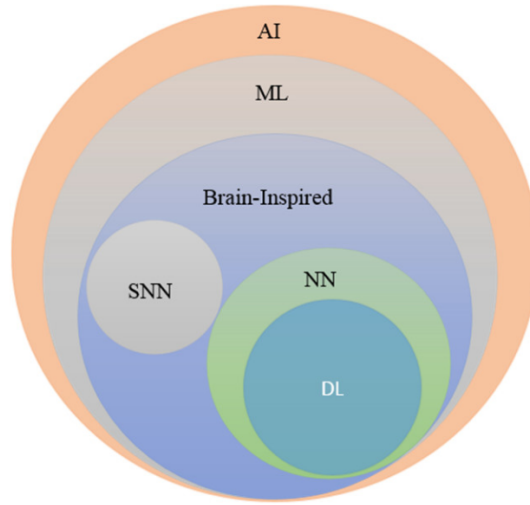


Figure 2.1: Relationships between Artificial Intelligence (AI), Machine Learning (ML), Neural Network (NN) and Deep Learning (DL).

CNN is mainly composed of feature extraction layer and classification layer. The feature extraction layer is mainly composed of convolutional layers and a pooling layers. These two kinds of layers usually appear alternately inside the CNN, with the convolution layer as the even layer and the pool layer as the odd layer. The convolutional layer functions in the extraction of image features, which convolute the input image for local perception using a convolution kernel. The pooling layer performs the spatially dimension reduction on the input image to reduce the input size, computation, and number of parameters of the next layer. The alternating use of these two layers allows the visual features in the image to be refined layer by layer and used as features to train the classifier. The classification layer is also called the fully-connected layer and is generally a Feed-forward neural network. In this layer, the features extracted from the feature extraction layer are used as inputs, and the output is the final classification result [Alom et al., 2019]. Fig. 2.2 illustrates the structure of a simple convolutional neural network [Alom et al., 2019].

There are many optimised CNN architectures such as LeNet, AlexNet, VGG16, GoogLeNet and ResNet. In this project, ResNet-152 with 152 layers of depth is used to construct the CRNN.

2.1.3 Recurrent Neural Network

Recurrent Neural Network (RNN) is used to solve problems that are completely different from CNN. When CNN focuses on extracting visual information from images, RNN focuses on processing temporal series information. RNN has a continuous memory ability to "understand" sequential information based on the understanding of the context. It is similar to human reading, where the understanding of each word is based on an understanding of the previous words that read earlier.

Traditional approaches such as deep neural networks (DNN) and convolutional

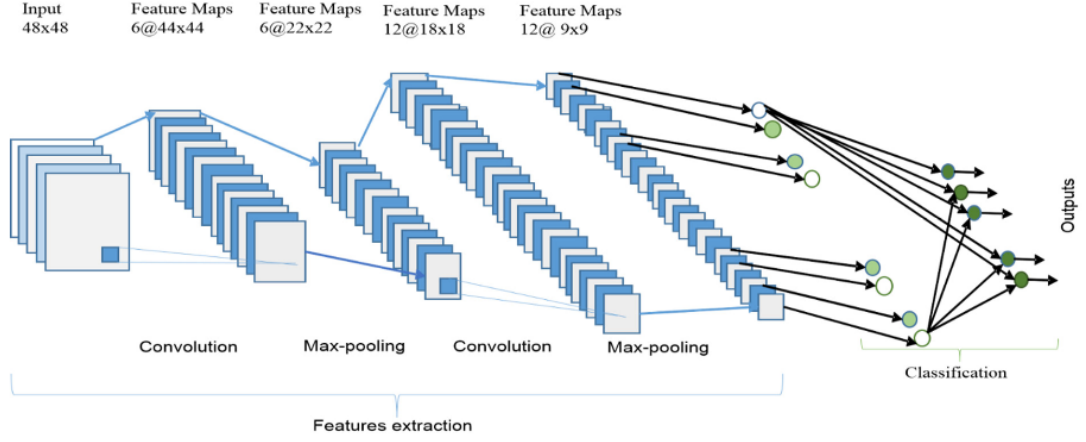


Figure 2.2: The structure of a simple Convolutional Neural Network (CNN). Convolutional layers and pooling layers alternate in the feature extraction layer, and the extracted feature vectors are fed to the Feed-forward neural network for classification.

neural networks cannot "think" like this, because they only take a fixed-size vector (e.g., a video frame) as input and output a fixed-size vector (e.g., classes), and their computational steps (e.g., layers) are fixed [Alom et al., 2019]. Unlike the above method, the RNN has memory capabilities, where the previous information on the time axis is influential for the processing of subsequent information.

In the simple recurrent neural network proposed by Elman [Elman, 1990], this function is implemented by using the output of the hidden layer together with the regular input as the input to the hidden layer in the next hidden state. Fig. 2.3 shows the structure of a simple RNN [Alom et al., 2019]. The working principle of RNN is expressed mathematically as [Elman, 1990]:

$$h_t = \sigma_h (w_h x_t + u_h h_{t-1} + b_h), \quad (2.1)$$

$$y_t = \sigma_y (w_y h_t + b_y), \quad (2.2)$$

where h_t is the hidden layer, x_t is the input, y_t is the output, w and u are weight matrices and b is the bias.

In this project, the more complex RNN architecture Long Short-Term Memory (LSTM) is used to construct the CRNN due to its more continuous memory and higher performance.

2.2 Related work

2.2.1 Classifications on Videos

The state-of-art video classification approaches are divided into three categories according to feature extraction methods: text-based, audio-based, and vision-based [Darji, 2017].

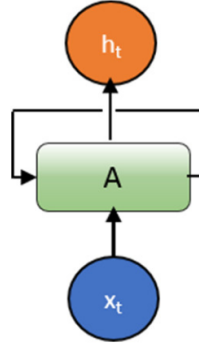


Figure 2.3: The structure of simple Recurrent Neural Network (RNN). The output of the hidden neuron is used along with the regular input as the input to the hidden neuron at the next time step.

Text-based video classification methods generally extract texts from two kinds of sources, one of which is texts from video frames, such as scoreboards in ball games, subtitles on screen, advertising signs, etc. OCR technology is generally used for this type of methods for text extraction. Another source is texts from speech, speech recognition technology is generally applied to extract subtitles of the video [Darji, 2017].

The audio-based video classification consumes much less computational resources and time than text-based and video-based methods, since the amount of computation for processing audio is much smaller than that for processing video frames. The audio signal in the video is typically sampled at a particular rate and the desired features are extracted from the sampled audio [Darji, 2017].

Most researchers have used video-based feature extraction methods, and sometimes combined with text-based and audio-based methods [Darji, 2017]. This kind of method extracts visual information from video frames as features for classification. Many literature aim to compute local feature descriptors from spatio-temporal volumes to precisely describe the temporal information of the video, and and quantise the extracted features into bag-of-words or Fisher Vector representations before feeding into classifiers [Brezeale and Cook, 2008]. The use of deep learning for automatic feature extraction and classification is a recent trend.

2.2.2 distinguishing between spontaneous and deliberate smiles

Many existing related works are manual methods in distinguishing between spontaneous and deliberate smiles [Mandal and Ouarti, 2017; Dibeklioglu et al., 2015]. They study the difference between spontaneous and deliberate smiles from psychological or biological perspectives, and manually place landmarks and track them to capture features. Dibeklioglu manually placed landmarks on the corners of the eyes, nose, mouth, etc. and tracked them to extract feature values, and then feed them into Support Vector Machine (SVM) for learning and classification [Dibeklioglu et al., 2015]. Mandal and Ouarti also applies dense optical flow based on a similar method to

Dibeklioglu [Mandal and Ouarti, 2017].

In general, these works ended up with high accuracy, but their methods are basically depend on manual input. This limits the practical application range of these methods. In their approach, the algorithm for feature extraction is manual design. However, the state-of-art technology in recent years allows us to use the deep learning approaches for feature extraction and learning to automate the process of distinguishing spontaneous and deliberate smiles.

2.3 Summary

In this chapter, the background necessary for reading this thesis and the related works are given. The architecture design and detailed implementation are given in the next chapter.

Design and Implementation

In this chapter, the design and implementation part of this project is given. The dataset used in this project is described in Chapter 3.1. Chapter 3.2 gives the design of the methods. The subsection 3.2.1 provides the overview of the method. The design of data pre-processing is given in subsection 3.2.2, and the design of classifier is given in subsection 3.2.3. Chapter 3.3 gives the detailed implementation. The implementation of data pre-processing is given in subsection 3.3.3 and the detailed implementation of the classifier is given in subsection 3.3.4. The section 3.4 gives the summary of this chapter.

3.1 Data description

This project is carried out around the UvA-NEMO Smile database [Dibeklioglu et al., 2012]. This database consists of 1240 videos (in RGB colour) recorded in the same lighting environment by the same camera with the resolution of 1920 * 1080 pixels. The camera was kept at a distance of 1.5 meters from the participant. The videos were recorded at 50 frames per second, and each video is between about 100 and 700 frames in length. Participants were shown short, interesting video clips to collect spontaneous smiles, while for deliberate smiles, they were asked to pose a smile as realistically as possible. The database contains over 400 participants, each of whom is between the ages of 8 and 76. Sample frames from the database are shown in fig. 3.1, where the top row shows the neutral faces of each participant, spontaneous smiles are shown in the medium row, and the bottom row shows the deliberate smiles. The database includes an experimental protocol that recommends cross validation to be more fair to compare with other studies.

3.2 Design

In this section, subsection 3.2.1 gives the overview of workflow for training the classifier model. The other subsections give the designed methods respectfully.



Figure 3.1: Sample frames from the UvA-NEMO Smile database [Dibeklioglu et al., 2012]. Five sample subjects shown in different columns. The top row shows the neural faces, spontaneous smiles are shown in the medium row, and the bottom row shows the deliberate smiles.

3.2.1 Overview

Fig. 3.2 gives an overview of workflow for training the classifier model. Firstly, frames are extracted automatically from each video and the raw frame dataset is formed. In the raw frame dataset, frames of each video are included in the corresponding folder. Then the dataset is pre-processed before training. As the experimental protocol of UvA-NEMO Smile database [Dibeklioglu et al., 2012] suggested, two-level cross-validation is applied on the complete pre-processed dataset for parameter optimisation. After the decision of optimised parameters, the pre-processed dataset is separated into training/validation dataset and test dataset.

3.2.2 Data pre-processing

In order to improve the quality of the dataset, the following pre-processing methods are applied to this project. Firstly, frame extraction is carried out to extract frames from each video. Then face detection, alignment and cropping is carried out to normalise the images with the facial information.

Frame extraction

Frame extraction is performed first on the raw dataset. Each frame is extracted from the video, and archived in the same directory, so that the dataset is converted from 1240 videos to 1240 directories, each has the same name with a video, and stores all frames of the video.

Face detection, alignment and cropping

Although the collection of the dataset is carried out under strict control of variables such as lighting and distance, the dataset still has deficiencies. First, the participants

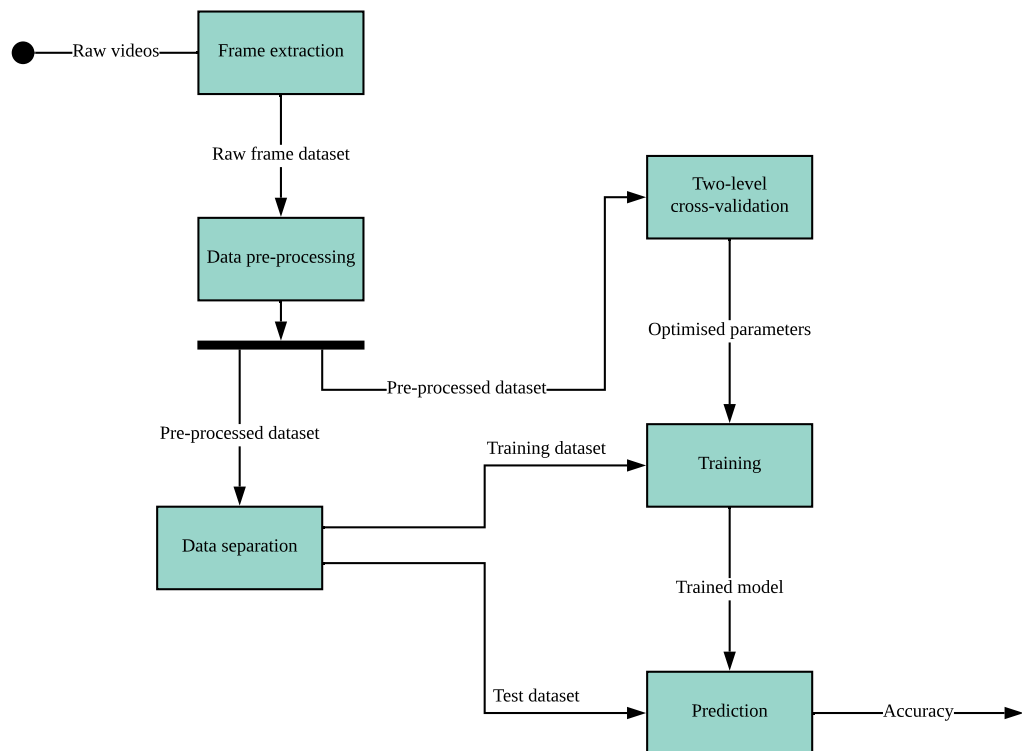


Figure 3.2: The overview of work flow for training the classifier model for classification.

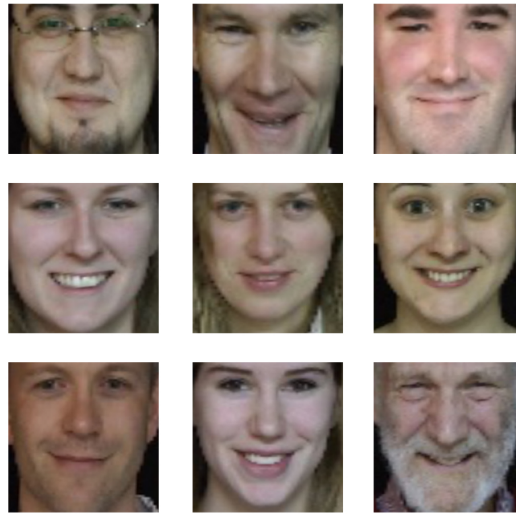


Figure 3.3: Sample frames after face detection, cropping and alignment. The outer eyes and the nose in each frame are aligned to the same position, so that most of the significant features on each face are aligned to similar positions.

in the videos are not always perfectly facing the camera, and their orientations tend to be biased. And their posture is always changing, especially when the participants make a smile, their faces would naturally sway. This can have a negative impact on the training of the model, especially for models that are sensitive to transformation such as Convolutional Neural Networks. Besides, the clothes worn by the participants, the numbers jumping on the screen after them, etc. were all irrelevant to distinguishing the spontaneous and deliberate smile of the participants. Therefore, these elements may become noises that mislead the learning of the classifier and should be masked out during the pre-processing stage.

In this project, face detection, alignment and cropping methods are applied on the frames in each video in order to address these problems. Firstly, the face is detected in each frame. Then the posture of the face is estimated, and an affine transformation is applied to align the outer eyes and bottom lip so that they appear at the same position on each frame. After that, the aligned faces are cropped and resized to a uniform size. Fig. 3.3 shows the sample frames after face detection, cropping and alignment. As can be seen from the figure, the face in each frame is aligned according to the outer eyes and the nose. In the image, only the human face is retained, and many interfering elements such as background, clothes and numbers are reduced.

3.2.3 Convolutional Recurrent Neural Network

In order to perform expression classification on the videos, Convolutional Recurrent Neural Network (CRNN), an end-to-end trainable neural network is constructed [Shi et al., 2015]. This neural network structure can naturally process sequential image-based information such as video and images containing text. In CRNN, Convolutional

Neural Networks (CNN) and Recurrent Neural Networks (RNN) are combined and connected together. The model consists of three layers: the convolutional layer, the recurrent layer and the fully-connected layer. The implementation of this model is based on the code from GitHub [HHTseng, 2019].

The convolutional layer is a Convolutional Neural Network with its fully-connected layer removed, which makes the network a visual feature extractor. For each two-dimensional RGB frame $x(t)$, the CNN is applied to encode it into a one-dimensional vector $z(t)$ by

$$f_{\text{CNN}}(\mathbf{x}^{(t)}) = \mathbf{z}^{(t)}. \quad (3.1)$$

Training a CNN model that can accurately extract important features from a smiling face from scratch can be very time consuming, therefore, in this project, a pre-trained ResNet-152 model is used [He et al., 2015]. ResNet is a residual learning framework that can be used to construct very deep CNN up to 152 layers by learning residual representation functions rather than directly learning signals [He et al., 2015].

The Recurrent layer is a Recurrent Neural Network, which receives each one-dimensional vector $z(t)$ produced by the CNN encoder as input, learns from the input visual features and the sequence, and another one-dimensional vector $h(t)$ is outputted. In this project, Long Short-Term Memory (LSTM) with longer memory capacity is used instead of standard RNN.

At the end of the CRNN structure, all outputs from the RNN layer are connected to the fully connected layer, which is a shallow neural network. In this layer, the output from each RNN time step is learned and the prediction is carried out to label the category of the video.

Fig. 3.4 shows a overview of CRNN's workflow. The CNN encoder is first used as a visual feature extractor to convolve each two-dimensional RGB frame into a one-dimensional feature vector. Thereafter, all feature vectors are combined into a two-dimensional matrix, where the first dimension represents the sequence of frames and the second dimension represents the feature vector of each frame. The RNN decoder is then applied to this matrix to learn the sequence, and provide an output for each frame. Finally, all outputs from the RNN layer are fed into a fully-connected layer for the final classification.

3.2.4 Two-level cross validation

According to the experimental protocol of UvA-NEMO database [Dibeklioglu et al., 2012], two-level cross validation should be applied for a fair comparison with other studies. The work flow of two-level cross validation is shown in fig. 3.5. The external layer (level 1) is a 10-fold cross validation, in this layer, each time a test fold is separated, a 9-fold cross validation is performed as the internal layer (level 2). In the internal layer, each time a validation fold is separated, the remained 8 folds are used for training CRNN model. There is no video overlap allowed between folds. The parameters are optimised in the internal layer, and the average prediction accuracy of the external layer is computed to evaluate the model.

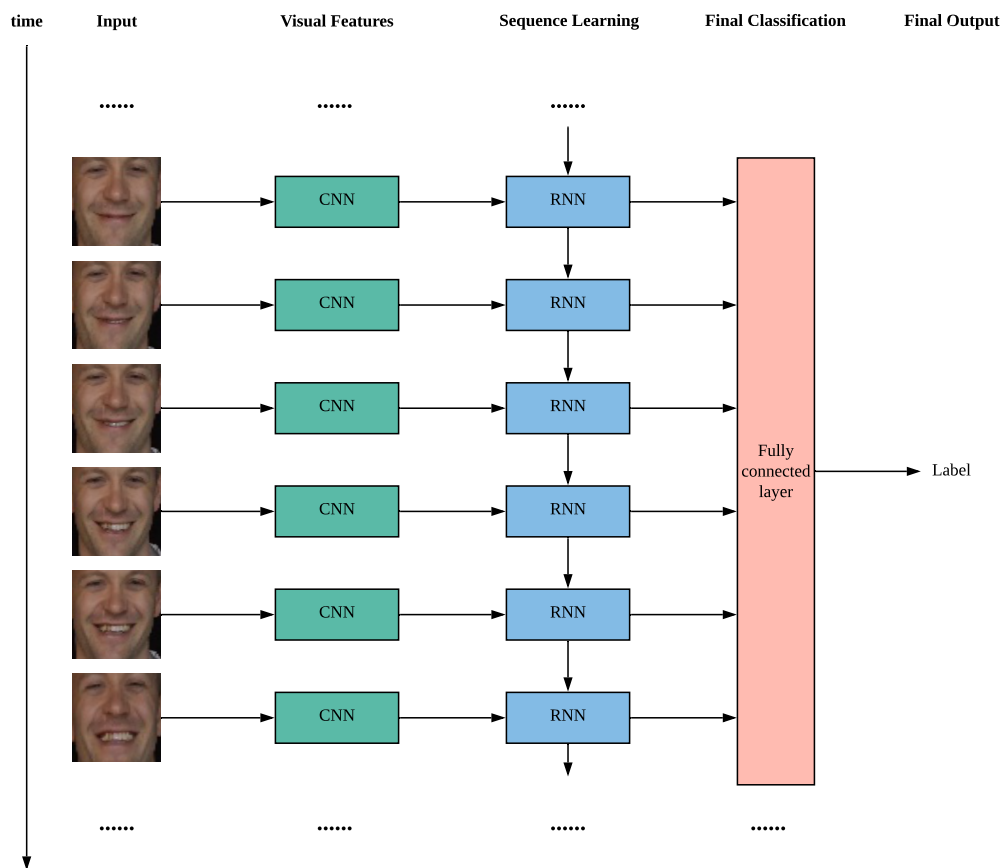
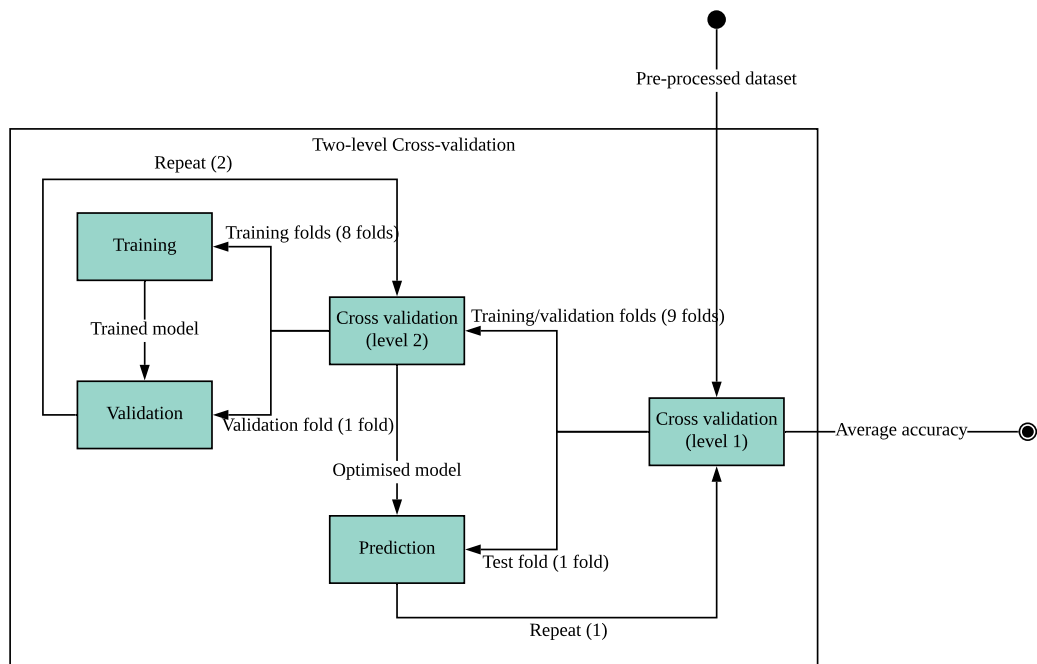


Figure 3.4: The overview of workflow for Convolutional Recurrent Neural Network. The visual features in each frame is extracted by the convolutional layer, and then the temporal features of the extracted feature vectors are learned by the recurrent layer. Finally, the output of the recurrent layer is fed to the fully-connected layer for learning or prediction.



Note:

- Repeat (1): In cross validation (level 1), the training/testing process repeats 10 times. On the i th repetition, the i th fold becomes the test fold, and the remaining 9 folds become training/validation folds for cross validation (level 2).
- Repeat (2): In cross validation (level 2), the training/validation process repeats 9 times. On the i th repetition, the i th fold becomes the validation fold, and the remaining 8 folds become training folds.

Figure 3.5: The work flow of two-level cross-validation.

3.3 Implementation

In this section, subsection 3.3.1 outlines the implementation, and the other subsections gives the detailed implementation of the designed methods.

3.3.1 Overview

In this project, all the code is built on Python 3.7 on Windows 10 or Linux (Red Hat and Ubuntu). Anaconda 3 is used as an environment manager. In the implementation of data preprocessing, openface [Amos et al., 2016], dlib [King, 2009], opencv [Bradski, 2000] and other libraries are used. In the implementation of the classification, PyTorch [Paszke et al., 2017], Numpy [Oliphant, 2006], Pandas [McKinney, 2010], Scikit-learn [Pedregosa et al., 2011], Matplotlib [Hunter, 2007], tqdm and other libraries are used. The code has been tested to run on windows 10 and linux (ubuntu). Nvidia CUDA is used to transfer data to the GPU for computation to reduce the training and prediction time of the model. Most of the code is written on Visual Studio Code, and a small amount of code is written on Jupyter Notebook.

3.3.2 Frame extraction

In order to optimise the efficiency of the code and simplify the process, all the frames in each video are extracted as a frame dataset for classification, rather than directly use the raw videos. Firstly, the original database is scanned and the name of each video is extracted. Then methods from the opencv library [Bradski, 2000] is used to open each video and save all the frames into the corresponding directory. Therefore, before frame extraction, the dataset is a directory of videos. After that, the dataset becomes a directory of sub-directories, each includes all the frames of the corresponding video.

3.3.3 Data pre-processing

In this project, code from the openface library [Amos et al., 2016] and the dlib library [King, 2009] is used to implement face detection, alignment and cropping. According to the requirements of the pre-trained ResNet-152 model [He et al., 2015], the frames is normalised using mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225] before inputted into the CRNN model. Meanwhile, all frames are also re-sized to 224×224 in RGB colours to accommodate the training configuration of the pre-trained ResNet-152 model.

3.3.4 Convolutional Recurrent Neural Network

In this project, the implementation of Convolutional Recurrent Neural Network is carried out using PyTorch [Paszke et al., 2017] and other libraries. Database objects that supports various PyTorch operations are built by inheriting the `Torch.Util.data.Dataset` class from PyTorch. Besides, `DataLoader` from PyTorch is used to iterate the dataset and load the items as needed. First the dataset is scanned and the name and label

of each video is extracted. Then the dataset is separated into 80% training set and 20% test set. After that, according to the separated names, two dataset objects are initialised to characterise the training set and test set. Then, a DataLoader object is initialised for each dataset object.

In CRNN, the neural network layers are implemented by inheriting the `Torch.nn.Module` class. The convolutional layer is constructed by a ResNet-152 model pretrained on the image dataset ILSVRC-2012-CLS [Russakovsky et al., 2015] with the original fully-connected layer removed. Besides, three additional fully-connected layers are added at the end with a batch normalisation layer between each two of them. These extra layers are used to reduce the dimensions of the output from the convolutional layer to 512. The recurrent layer has the same input size with the output size of convolutional layer. The LSTM has 3 hidden layers and 512 hidden nodes. Two fully-connected layer is concatenated at the end of LSTM for the final prediction with input size of 512 and output size of 2.

3.4 Summary

This chapter introduced the design of methodologies and the detailed implementation of the designs. In the next chapter, the experiments will be given.

Experimental Methodology

In this chapter, experiments are designed to evaluate the performance of different methods applied in the project. Section 4.1 gives the overview of the experiment methodology. The other sections give the detailed experiment methodology of assessing each method.

4.1 Overview

In this project, cross entropy is used as loss function on training and testing, and prediction accuracy is used as the primary means of model evaluation. In the next sections, the performance of several methods are assessed. The final result of the best model is compared with relevant research and the final conclusion is reached.

4.2 Data pre-processing

4.2.1 Basic pre-processing

When creating the baseline model, only the basic data pre-processing method is applied, which is the image normalisation required for the pre-trained ResNet-152 model as introduced in 3.3.3. The frames are normalised using means = 0.485, 0.456, 0.406 on R, G, and B channels respectively, and standard deviations = 0.229, 0.224, 0.225 on R, G, and B channels respectively. At the same time, all frames are resized to 224×224 RGB images to accommodate the training configuration of the pre-trained ResNet-152 model.

4.2.2 Face detection, alignment and cropping

Face detection, alignment, and cropping is applied to provide a better model performance. In order to assess whether this method is indeed useful, its application or not was used as a variable for multiple sets of controlled experiments. In the baseline, this method was not applied, and in the experimental group, the faces were aligned according to the outer eyes and the nose.

4.3 Method of unifying video length

Since PyTorch cannot assemble multiple sets of data of different lengths into batches of the same size, it is necessary to unify the video length of the data set. In this project, various methods of uniform video length were tried.

4.3.1 Use the first 90 frames

This method is used as a baseline. To unify the lengths of all the videos, an heuristic approach is to take the same number of frames from each video to form a clipped video representing the original one. In this dataset, all videos are longer than 90 frames, thus, when loading the dataset, the baseline method only loads the first 90 frames in each video, therefore the common video length in the dataset is fixed at 90.

4.3.2 Upsampling

This method upsample all video to the maximum video length of the dataset. First, scan the dataset and get the longest video length as the target common length of the data set. Then, for all videos shorter than the target common length, one frame is randomly extracted and one copy of it is inserted into its next position. Repeat this operation until the video length reaches the target common length.

4.3.3 Dimensional compression and replication

After a video of length n passes through the convolutional layer in the CRNN model, the output of the entire layer will have a size of $300 \times n$, where d is the length of the vector after each frame passes through the convolutional layer, i.e. the output size of the last fully-connected layer connected to the end of the ResNet-152 layer, and n is the number of frames. Because the length of each video is different, that is, n is indeterminate, the output size of the entire convolutional layer is uncertain.

This method compresses the time dimension of the output to one by averaging after the video passes through the convolutional layer, i.e. compressing the output size of convolutional layer from $d \times n$ to $d \times 1$. The averaged frame is then copied a fixed number of times m to form a fixed size matrix of $d \times m$.

4.3.4 Zero-padding

This method pads each video shorter than the maximum video length L of the dataset with the padding value to L , and instructs the convolutional layer and the recurrent layer to ignore these padding values when updating, therefore CRNN can accept videos in variable length without losing any information or adding any noise. However, due to the time limitation of this project, this part has not been completed. The specific situation is elaborated in Chapter 6.

4.4 Summery

In this chapter, the experimental methodologies are introduced. In the next chapter, the detailed result is given.

Results

In this chapter, the experiment results are given.

5.1 Overview

Table 5.1 gives the experiment results. Experiment 1 is the baseline for the entire project, using only basic data pre-processing intro introduced in sub section 4.2.1, and unifying the video length by intercepting the first 90 frames of each video. In other experiments, the performance of various methods was evaluated by changing the methods applied and comparing the results.

5.2 Data pre-processing

According to table 5.1, compared with the basic pre-processing only, the additional application of facial detection, alignment and cropping has a slight improvement on the accuracy of the model. This is because this method aligns the positions where significant features (such as eyes, mouth corners, etc.) appear on each image appear, thus, the convolution layer can extract important visual information more accurately, and it also removes that irrelevant to facial expressions, such as backgrounds, clothes, screens, numbers and color blocks.

5.3 Method of unifying video length

According to table 5.1, among the three different methods, the best one is upsampling, which slightly improves the accuracy of the model. This may be because upsampling preserves the chronological order of the video. Although some frames are copied multiple times, the appearance order of the frames is consistent with the original video. In contrast, the method of taking only the first 90 frames loses a lot of information. Many videos are much longer than 90 frames, and these videos may not even contain smiling frames in the first 90 frames. Dimensional compression and replication loses the time information of the video and makes the recurrent layer lose its effect. Therefore, the results obtained by these two methods are no different from blind guesses.

Table 5.1: The experiment results.

Experiment	Data pre-processing method	Unifying video length method	Accuracy
1 (Baseline)	None	Use only the first 90 frames	49.7%
2	Face detection, alignment, and cropping	Use only the first 90 frames	51.2%
3	None	Upsampling	50.4%
4	Face detection, alignment, and cropping	Upsampling	54.6%
5	None	Dimensional compression and replication	50.5%
6	Face detection, alignment, and cropping	Dimensional compression and replication	51.7%

5.4 Discussion

In this project, the final performance of the model was not satisfactory, and the best performance only achieved 54% accuracy. The reasons for the low accuracy may be multifaceted.

5.4.1 CRNN

It is difficult to believe that the low accuracy is attributed to the structure of CRNN. The same model was used to identify actions on the UCF-101 dataset [Soomro et al., 2012] before the model was officially used on UvA-NEMO Smile database to distinguish spontaneous and deliberate smiles. Similar parameter settings can achieve an accuracy of 85.68% on the UCF-101 dataset [HHTseng, 2019]. Therefore, it can be considered that the low accuracy is not due to a problem with the structure of the CRNN.

5.4.2 Pretrained model

The most likely cause comes from the pre-trained model. In order to enable the convolutional layer to accurately extract the visual information in the images earlier, the ResNet-152 model pre-trained on ILSVRC-2012-CLS [Russakovsky et al., 2015] is used in implementation. However, since the dataset for pre-training is different from the content of the UvA-NEMO Smile database, it is likely that the ResNet-152 CNN cannot accurately extract important visual features related to distinguishing between spontaneous and deliberate smiles. It can be expected that training a ResNet-152 CNN from the scratch for the UvA-NEMO Smile database can effectively improve the feature extraction capability of the convolutional layer.

5.4.3 Video length

In this project, most of the methods to achieve uniform video length will cause different degrees of loss of video information. Although upsampling is the method causing the least loss among them, it still causes a loss of time information, because this is equivalent to slowing down the time of the shorter videos. The zero-padding method that was not implemented in this project can make CRNN compatible with video of various lengths without losing any information. It can be expected that the implementation of this method will effectively improve the accuracy of the model.

5.4.4 Face alignment

The face alignment method from dlib [King, 2009] does not perfectly align all faces.

5.5 Summary

Overall, the results of this experiment show that the accuracy of the model is not high enough. Most of the methods make the model produce accuracy similar to blind guessing, but only when using face detection, alignment, cropping and upsampling at the same time produce a slightly better result than blind guessing. However, this performance is not good enough to be practical in the real world. In this chapter, the reasons for the results are also discussed.

In the next chapter, the project is concluded, and the challenge the future works are given.

Conclusion

In this project, the application of end-to-end convolution recurrent neural networks is proposed to train the deep neural network model to automatically distinguish between spontaneous and deliberate smiles in the UvA-NEMO smile database. Compared with related studies, this method does not require manual input on the dataset, thus automating the processes from model training to prediction. Reasonable processes from data pre-processing to prediction was designed to implement most of the designed functions. Multiple sets of controlled experiments were performed to evaluate the performance of different methods, and the experimental results were explained and discussed.

In this chapter, section 6.1 gives the challenges faced in this project, and the future work after this project is given in section 6.2.

6.1 Challenges

6.1.1 Hardware limitation

The biggest challenge faced this project comes from hardware limitations. The UvA-NEMO Smile Database contains 1240 videos, each of which is between 100 and 700 frames in length, which means that there are huge frames to process. Moreover, CRNN is a deep neural network that includes ResNet-152, three LSTM layers, and four fully-connected layers, which consumes a lot of time for each iteration.

In the beginning, the project development environment was a laptop with an Nvidia GTX 980M GPU and an Intel Core i7-4720HQ CPU. Project development is done on Windows 10 and Windows Subsystem for Linux (Ubuntu). Models are typically trained on 5% of the dataset to get result earlier. Due to memory limitations, the batch size used for training can only be less than 3, otherwise the code will throw a memory leak. The training conducted in this environment consumes about 45 minutes per epoch on average. This is quite time consuming, and because CRNN training runs out of RAM to achieve maximum speed, it is impossible to do other work on the laptop at the same time.

To solve this issue, Google Colab is used as a free cloud computing service. Its ontology is a notebook software like Jupyter Notebook running in the cloud, but it provides a cloud Linux virtual machine and allows Google Drive to be mounted as a

virtual hard disk. Moreover, Google provides a free Nvidia Tesla T4 GPU with 16GB RAM for use on Colab. However, model training on this platform is still a challenge. Colab's cloud virtual machine is shut down after 12 hours from initialisation, and the high-performance GPU has a one-hour usage limit. Also, the training of the model requires loading data from Google Drive, and the transfer speed between the virtual machine and the virtual hard disk is quite slow. Therefore, although Colab provides a high-performance GPU, the time spent training model on it is actually much slower than training model locally.

In the last month of project development, I got an access to a GPU cluster server with four Nvidia GTX 1070 GPUs, so the entire project turned to development and training models on the GPU cluster. However, it is still time consuming to complete the training. In practice, each training epoch takes about 1 hour and 40 minutes when applying upsampling and training on 80% of the dataset with a batch size of 10, and it takes 5 to 6 days to complete a complete 80-epoch training. This speeds up the development of the project, but it is still not fast enough.

6.1.2 Zero-padding

The methods of upsampling and dimensional compression used in this project only need to add a few lines to the code, and there is no side effect on the project itself. Unlike them, development of zero-padding requires extensive modifications to the entire code, including padding with the padding value when assembling the batch, indicating that the convolutional layer and the recursive layer ignore padding values, etc.. This method was proposed in the middle of the project, but due to hardware limitations, it takes lots of time to debug, thus upsampling is applied instead of this method to get a usable result as soon as possible.

6.2 Future Work

In the future work, the method of exploring a better unified video length is an important direction. The first work is to implement a zero-padding method to make the CRNN model compatible with video of different lengths. Then, finding the peak moment of the smile in the video is a considerable direction, and finding a threshold of smile level and intercepting frames above this threshold as smile data for training is a data enhancement method that can be considered. It can be expected that this method can effectively improve the accuracy of the model when combined with zero padding. In addition, training a CNN model on the UvA-NEMO from scratch as the convolutional layer for visual feature extraction is also an optional direction.

These methods are all designed to achieve higher model accuracy. When the accuracy is high enough, which shows that this method performs well on the UvA-NEMO Smile database, extension of this method to other datasets is worth considering, and expecting to finally reach a generic model that can distinguish between spontaneous and deliberate smiles from arbitrary video.

Bibliography

- ALOM, M. Z.; TAHA, T.; YAKOPCIC, C.; WESTBERG, S.; SIDIKE, P.; NASRIN, M.; HASAN, M.; ESSEN, B.; AWWAL, A.; AND ASARI, V., 2019. A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8 (2 2019), 1–127. doi:10.3390/electronics8030292. (cited on pages 3, 4, and 5)
- AMOS, B.; LUDWICZUK, B.; AND SATYANARAYANAN, M., 2016. Openface: A general-purpose face recognition library with mobile applications. (2016). (cited on page 16)
- BENGIO, Y., 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 8 (03 2009), 292. (cited on page 3)
- BRADSKI, G., 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, (2000). (cited on page 16)
- BREZEALE, D. AND COOK, D., 2008. Automatic video classification: A survey of the literature. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38 (06 2008), 416 – 430. doi:10.1109/TSMCC.2008.919173. (cited on page 6)
- DARJI, M., 2017. A review of video classification techniques. *International Research Journal of Engineering and Technology*, 4 (06 2017), 1888–1891. (cited on pages 5 and 6)
- DIBEKLIOGLU, H.; SALAH, A.; AND GEVERS, T., 2012. Are you really smiling at me? spontaneous versus posed enjoyment smiles. *Proc. European Conference on Computer Vision (ECCV)*, (2012). (cited on pages ix, 1, 9, 10, and 13)
- DIBEKLIOGLU, H.; SALAH, A. A.; AND GEVERS, T., 2015. Recognition of genuine smiles. *IEEE Transactions on Multimedia*, 17, 3 (Mar 2015). (cited on pages 2 and 6)
- EKMAN, P.; FRIESEN, W. V.; AND SULLIVAN, M. O., 1988. Smiles when lying. *J. Personality Social Psychol.*, 54, 3 (1988), 414–420. (cited on page 1)
- ELMAN, J., 1990. Finding structure in time. *Cogn. Sci.*, 14 (1990), 179–211. (cited on page 5)
- FUKUSHIMA, K., 1988. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Netw.*, 1 (1988), 119–130. (cited on page 3)
- HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2015. Deep residual learning for image recognition. (2015). (cited on pages 2, 13, and 16)

-
- HHTSENG, 2019. Hhtseng/video-classification. <https://github.com/HHTseng/video-classification>. (cited on pages 13, 25, and 37)
- HOCHREITER, S. AND SCHMIDHUBER, J., 1997. Long short-term memory. *Neural Computation*, 9(8) (1997), 1735–1780. (cited on page 2)
- HUNTER, J. D., 2007. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9, 3 (2007), 90–95. doi:10.1109/MCSE.2007.55. (cited on page 16)
- KING, D. E., 2009. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10 (2009), 1755–1758. (cited on pages 16 and 25)
- MANDAL, B. AND OUARTI, N., 2017. Spontaneous versus posed smiles—can we tell the difference? In *Proceedings of International Conference on Computer Vision and Image Processing*, vol. 460 (Singapore, 2017). Springer. (cited on pages 2, 6, and 7)
- McKINNEY, W., 2010. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, 51 – 56. (cited on page 16)
- OLIPHANT, T. E., 2006. *A guide to NumPy*, vol. 1. Trelgol Publishing USA. (cited on page 16)
- PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DeVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; AND LERER, A., 2017. Automatic differentiation in pytorch. In *NIPS-W*. (cited on page 16)
- PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; AND DUCHESNAY, E., 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12 (2011), 2825–2830. (cited on page 16)
- RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; AND FEI-FEI, L., 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115, 3 (2015), 211–252. doi:10.1007/s11263-015-0816-y. (cited on pages 17 and 25)
- SHI, B.; BAI, X.; AND YAO, C., 2015. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *Huazhong University of Science and Technology*, (Jul. 2015). (cited on page 12)
- SOOMRO, K.; ZAMIR, A. R.; AND SHAH, M., 2012. Ucf101: A dataset of 101 human action classes from videos in the wild. *CRCV-TR-12-01*, (Nov. 2012). (cited on page 25)

Appendix 1: Project Description

6.3 Project Title

Distinguishing genuine and posed smiles using computer vision deep learning approaches.

6.4 Learning Objectives

- Experience with face and emotion recognition
- Experience with computer vision for emotion recognition problems
- Experience with deep learning for face related computer vision tasks

6.5 Project Description

Most of the available techniques [1,2] of finding the difference between spontaneous vs. posed smiles are dependent on hand engineered facial landmark features. Moreover, established methods usually manually annotate the first frame as user input which essentially hampers the ultimate automation. We want to investigate whether an end-to-end deep learning framework can be employed to propose a fully automated solution for this problem. Recent developments of the combination of CNN and RNN can enable us to classify a video in an end-to-end manner [3]. We want to apply such approaches to classify spontaneous vs. posed smile videos. Our end-to-end architecture will incorporate all known knowledge about the discriminative marker but with no/minimum human input. We want to focus on UvA-NEMO smile database for our experiments due to its vastness and diversity.

[1]<https://ivi.fnwi.uva.nl/isis/publications/2015/DibekliogluTMM2015/DibekliogluTMM2015.pdf>

[2]https://link.springer.com/chapter/10.1007/978-981-10-2107-7_24

[3]<https://github.com/HHTseng/video-classification>

Appendix 2: Independent Study Contract



INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UniID:	u6343089		
FAMILY NAME:	Hou	PERSONAL NAME(S):	Liwei
PROJECT SUPERVISOR (may be external):	Zakir Hossain and Shafin Rahman		
COURSE SUPERVISOR (a RSCS academic):	Tom Gedeon		
COURSE CODE, TITLE AND UNIT:	COMP4560 Advanced Computing Project 12 units		

SEMESTER	<input checked="" type="checkbox"/> S1	YEAR: _2019_	<input checked="" type="checkbox"/> S2	YEAR: _2019_
PROJECT TITLE: Distinguishing genuine and posed smiles using computer vision deep learning approaches				

LEARNING OBJECTIVES:
Experience with face and emotion recognition
Experience with computer vision for emotion recognition problems
Experience with deep learning for face related computer vision tasks

PROJECT DESCRIPTION:
Most of the available techniques [1,2] of finding the difference between spontaneous vs. posed smiles are dependent on hand engineered facial landmark features. Moreover, established methods usually manually annotate the first frame as user input which essentially hampers the ultimate automation. We want to investigate whether an end-to-end deep learning framework can be employed to propose a fully automated solution for this problem. Recent developments of the combination of CNN and RNN can enable us to classify a video in an end-to-end manner [3]. We want to apply such approaches to classify spontaneous vs. posed smile videos. Our end-to-end architecture will incorporate all known knowledge about the discriminative marker but with no/minimum human input. We want to focus on UvA-NEMO smile database for our experiments due to its vastness and diversity. [1] https://ivi.fnwi.uva.nl/isis/publications/2015/DibekliogluTMM2015/DibekliogluTMM2015.pdf [2] https://link.springer.com/chapter/10.1007/978-981-10-2107-7_24 [3] https://github.com/HHTseng/video-classification

ASSESSMENT (as per course's project rules web page, with the differences noted below):

Assessed project components:	% of mark	Due date	Evaluated by:
Thesis	45%		
Artefact	45%		
Presentation	10%		

MEETING DATES (IF KNOWN):

Weekly

STUDENT DECLARATION: I agree to fulfil the above defined contract:

.....*Lini Non.*.....
Signature Date 2019/02/27

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

.....*T.S. Gedeon*.....
Signature Date 28 February 2019

Reviewer:

Name:Penny Kyburz..... Signature:*[Signature]*.....

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

.....
Signature Date

SECTION D (Projects coordinator approval)

.....
Signature Date

Appendix 3: Code Description

6.6 Code Outline

The artefact includes 6 files:

- `NEMO_ResNetCRNN.py`
- `functions.py`
- `ResNetCRNN_check_prediction`
- `frame_extraction.py`
- `check_video_predictions.ipynb`
- `NEMOclasses.pkl`.

6.7 Copyright

The following codes are created by myself: `frame_extraction.py` and `NEMOclasses.pkl`. The following codes are based on HHTseng's code on GitHub [HHTseng, 2019] and modified by myself: `NEMO_ResNetCRNN.py`, `functions.py`, `ResNetCRNN_check_prediction` and `check_video_predictions.ipynb`. The parts that are original, modified, or others' work are clearly indicated in the codes.

6.8 Description

The file `functions.py` contains definitions of all the public functions, dataset classes, and models used in this project. The code for training models is included in `NEMO_ResNetCRNN.py`, and the code for prediction is included in `ResNetCRNN_check_prediction`. `NEMOclasses.pkl` contains object of the categories for classification and is loaded by `NEMO_ResNetCRNN.py` and `ResNetCRNN_check_prediction` automatically. `frame_extraction.py` is used to extract frames from the raw dataset. The code that illustrate the prediction result is included in `check_video_predictions.ipynb`.

6.9 Environment Prerequisites

The experiment was carried out on the GPU clusters with Nvidia GTX 1070 GPU and RedHat system. These codes should also work on Windows 10 and Ubuntu. If any er-

ror occurred, it is most likely due to insufficient hardware conditions. At this time, reduction of `batch_size` in `NEMO_ResNetCRNN.py` and `ResNetCRNN_check_prediction` should be considered.

- Python 3.6
- PyTorch 1.0.0
- Numpy 1.15.0
- Sklearn 0.19.2
- Matplotlib
- Pandas
- tqdm
- OpenCV 4.1.0
- OpenFace 2.2.0

Appendix 4: Readme

6.10 Environment Prerequisites

- Windows 10 or Ubuntu
- Python 3.6
- PyTorch 1.0.0
- Numpy 1.15.0
- Sklearn 0.19.2
- Matplotlib
- Pandas
- tqdm
- OpenCV 4.1.0
- OpenFace 2.2.0

6.11 How to obtain the data

The original dataset is available in: <https://www.uva-nemo.org/>

If you do not have access to the original dataset, the extracted frames are available in: <https://drive.google.com/open?id=17rgv6ucLS0vGARh6XNoQljfo1eFr8KYN>

6.12 Workflow

1. Prepare the data and environment.
2. Set the input and output locations in `frame_extraction.py` and run frame extraction with command `python3 ./frame_extraction.py`.
3. Set the The location of the data directory (`data_path`), the location of `NEMOclasses.pkl` (`smile_type_path`), the location where the trained model will be saved (`save_model_path`), the location of index of the videos randomly selected in the training set (`train_names_path`), and other parameters in `NEMO_ResNetCRNN.py`.
4. Run model training using command `python3 ./NEMO_ResNetCRNN.py`.

5. In `ResNetCRNN_check_prediction.py`, use the same configuration as that in `NEMO_ResNetCRNN.py`.
6. Run prediction using command `python3 ./ResNetCRNN_check_prediction.py`
7. To check the results, open `check_video_predictions.ipynb` with Jupyter Notebook, and run all cells one by one.