FIT 3162 Computer Science Project
Summer Semester, 2021

Final Project Report
Due date: Friday 14 Jan 2022, 11.00 pm

Prepared by:

Joanne Ang Soo Yin	30513723
Lau Sin Lu	30764025
Nelly Tay Yee Ting	30312523

Word count: 7809

Contents

Introduction **3**

Background **4**

Outcomes **8**
Part I: EVP 8
Technical limitations of our code 8
Technical limitations of the EVP design and future work 8
Part II: Analysis of EVP-generated videos 9
Testing EVP results using FERs and MER 9
Limitations 12

Methodology **13**
Software and tools 13
Part I: EVP 14
Design: Overview of EVP architecture 14
Our contribution 15
Part II: Analysis of EVP-generated videos 17

Software deliverables **21**
Summary of software deliverables 21
lm2video module 23
vid2vid module 24
Summary of Software Qualities Handling 26

Critical Discussion **27**

Conclusion **28**

Appendix **29**
Sample source code 29
Annex 32

References **33**

Introduction

Our project revolves around Audio Driven Emotional Video Portraits (EVP), which is a novel facial video manipulation technique developed by Ji et al. Given an input audio and an input video, Ji et al. 's method enables the video to be altered, such that the subject speaking in the video shows the emotion detected from the input audio.

Our project aims are formulated as such:

I. Perform analysis on EVP-generated videos using micro-expression recognizers and facial expression recognizers

To achieve novelty, our team aims to analyse facial videos generated using EVP to investigate whether the facial expressions and micro-expressions detected from these videos correspond or conflict with each other.

II. Develop a library of code for facial manipulation software using EVP as base code

Our team aims to provide a library of code, such that those who want to work on completing EVP in the future can utilize our code.

Background

Literature Review

Deepfake technology has raised many concerns and distress worldwide over the years due the production of ingenuine but realistic visual and auditory media. As much as it is a problem, it is still needed in multimedia related fields such as digital human animation, telepresence and film-making. It is achieved through leveraging powerful machine learning and artificial intelligence techniques to produce the visual and audio content. These techniques include advanced image processing techniques and generative adversarial networks (GAN). Facial features and facial motion components are extracted and transplanted onto a target face using advanced image processing techniques, whereas GAN utilizes generative algorithm and discriminative algorithm to manipulate the source data (specifications). Given input labels, the generator side will create new instances of images or videos and the discriminator side evaluates the instances created.

The focus of the project is to build on the existing Emotional Video Portraits (EVP) algorithm proposed by Ji et al. (2021), with the aim of building a library of code for facial manipulation software. Apart from that, in order to evaluate the correspondence between the subjects' expressions from the generated facial videos and the desired emotion, we also perform tests on EVP-generated videos using state-of-the-art facial expression recognizers. We go one step further by performing tests on using a micro-expression recognizer to inspect whether the emotions detected using facial expression recognizers are the same with those detected using a micro-expression recognizer. This helps to uncover potential problems with the EVP technique which can be improved upon in the future to enhance the naturality of the facial videos generated.

According to Ji et al. (2021), "the task of audio-driven talking-head generation aims at synthesizing lip-synced videos of a speaking person driven by audio." The methods to complete this task are divided into two categories, image-based and video-based editing. In the case of image-based methods, an approach was proposed by Song et al. (2019) that incorporates audio and image in the recurrent unit in order to achieve a smooth transition for the movement of the face and lip. Two types of discriminators were constructed by them, namely spatial-temporals discriminators and lip-discriminators. The former is designed for video-realism and image-realism, while the latter is used to increase the lip synchronization's accuracy (Song et al., 2019). Although this approach does not require a video stabilization procedure or extra image deblurring work, and the network can be extended to model a more natural pose and expression for a video, the results are still lacking in animating head movements and expressions.

As for video-based editing methods, since the video portraits are framed from the person's face up to their shoulders, videos in this setting will definitely be more realistic but harder to construct. Most methods only edit the mouth areas due to difficulty in editing the face pose and shoulders as well. An audio to landmark RNN was trained to synthesize talking videos of Obama that have high potential to deceive (Suwajanakorn, Seitz and Kemelmacher-Shlizerman, 2017). Realistic talking videos can be achieved through such

techniques, but it is often difficult to manipulate the upper face and emotions using their models.

Ji et al. (2021) suggested that the emotion and content component entangled in the audio should be extracted independently into two latent audio spaces, namely the duration-dependent space and content agnostic-encoding. The former one is the speech content encoding of the audio and the latter is the emotion's content-agnostic encoding. This is unlike the previous method proposed by Wang et al (2020) where only a single representation is learnt from the audio signals. Pseudo training pairs are built and the cross-reconstructed training for the emotion entanglement is adopted. The training of the disentanglement network is done through leveraging the audio-visual dataset with different people speaking identical words but under different emotions states (Ji et al., 2021). The Mel Frequency Cepstral Coefficients (MFCC) and Dynamic Timing Warping (DTW) are used for the alignment of the uneven-length speeches.

When generating talking faces, the prediction of the landmark motions from disentangled audio embeddings is done by utilizing the audio-to-landmark network. Ji et al. (2021) has chosen to edit not only the mouth area of the target, but also the whole face in order to acquire a more realistic result. This adds complications to the generated facial videos due to two reasons: 1) There is little to none information for the pose in the audio clips. Due to that, predicted landmarks for the head movements may largely differ from the target video which will cause the misalignment of the head poses. 2) Blending the target video and the edited face seamlessly is a huge challenge. The 3D-Aware Keypoint Alignment algorithm is proposed by Ji et al. (2021) as a countermeasure to the aforementioned challenge. Landmarks that are generated will be positioned based on guidance landmarks, which is a reflection of the actual head movement seen in the target video. The difference between the two landmarks is computed for it to be useful. The fusion of the generated landmarks with the target image's edge map is made use to train Edge-to-Video translation networks (Ji et al., 2021).

The method used for the detection of the landmark on the target video was proposed by Wu et al. (2021). Their facial landmark localization method is achieved by defining the geometric structure of the human face using boundary lines. An edge detection algorithm proposed by Green (2002) is applied to extract the outer edges of the face from the target facial video. For our project, we follow the EVP methodology closely in writing the library of code with the open sourced version of EVP code as base code.

Facial micro-expressions refer to the involuntary muscle movements of one's face, which could reveal one's real emotion since they are uncontrollable. However, these expressions are obscure and only last for a very short duration thus human's naked eyes are commonly incapable of identifying micro-expressions. In recent years, the machine learning techniques are leveraged to develop automatic micro-expressions detection algorithms and many micro-expression recognition approaches today including hand-designed and learning-based methods have achieved significant results.

Sai et al. (2019), had proposed two 3D convolutional neural networks which extract both the spatial and temporal features from the input videos for classification. The first 3D-CNN model which is full face region based, MicroExpSTCNN, is constructed by different layers to

learn the information from faces in input video. As the researchers Duan et al. (2016), Wang and Hua (2016) discovered that eyes and mouth region contribute more to an efficient expression recognizer, the authors proposed another MicroExpFuseNet with two versions: Intermediate MicroExpFuseNet and LateMicroExpFuseNet that only detect the muscle movements of eyes and mouth portion and fused the features together to predict the expression. The features of eyes and mouth are separately processed by two 3D-CNN networks, and two versions of MicroExpFuseNet models concatenate the output at different levels. In the experiments, the performance of three models had obtained similar or even better accuracies than state-of-the-art methods.

Among the three models, MicroExpSTCNN had attained best performance and by analyzing the saliency maps, the author suggested that facial features apart from eyes and mouth play an important role in classifying micro-expression. In our work, we trained the MicroExpSTCNN with the SMIC dataset and obtained a training accuracy of 61.86%. It is then used to test the EVP results and it was observed that the majority of the positive and surprise emotions are predicted as negative emotions. A few examples are taken from the SMIC dataset and EVP output to figure out the reason behind this result. It was then discovered that the training dataset consists of more features such as slightly raised eyebrows and slight curve at the corner of the mouth as compared to the EVP output videos. Therefore, it can be concluded that the EVP has a limitation where it is unable to generate videos with micro-expressions accurately.

Emotions can be reflected through movement of subcutaneous muscles which can be called as facial expression (Shi et al., 2021) . According to Wen et al., “Facial expressions are direct and fundamental social signals in human communication” (2021). Nonverbal emotional cues can be conveyed through facial expressions as well as when they are conveyed along with other gestures (Wen et al., 2021). Facial expression recognition (FER) as the name suggests, is a technology that allows the automatic recognition of facial expressions through computers (Wen et al., 2021).

The proposed FERs used in this project are the Distract Your Attention Network (DAN) and Frame Attention Networks (Emotion-FAN). Both of these are state-of-the-art FERs which are used to perform tests on the EVP results to obtain the confusion matrix and accuracy for analysis. Emotion-FAN, which was proposed by Meng et al., is a network that highlights certain discriminative frames automatically in an end-to-end framework (2019). It works in such a way that a video with variable number of face images is taken as an input to the network and a fixed-dimension representation can be produced. The two modules in the network are the feature embedding module and the frame attention module. Face images are embedded into the feature vectors with the help of the feature embedding module. Multiple attention weights are then learned by the frame attention module to form a single discriminative video representation through adaptively aggregating the feature vectors. The Emotion-FAN was trained on the CK+ dataset and is then used to test the EVP results for analysis.

The DAN is proposed with three key components which are the Feature Clustering Network (FCN), Multi-head cross Attention Network (MAN), and Attention Fusion Network (AFN) (Wen et al., 2021). Robust features are extracted using the FCN. In order to maximize the class separability, the FCN adopts a large-margin learning objective for the extraction. A

number of attention heads are then instantiated by the MAN, so that multiple facial areas can be attended to simultaneously and attention maps can be built on these regions. Before the fusion of attention maps into a comprehensive one, the attentions are distracted by the AFN to multiple locations. DAN was trained on the RAF-DB dataset and also used to test the EVP results to generate an output for analysis.

After the aforementioned datasets are trained on the FERs, it can be observed that the Emotion-FAN obtained an accuracy of 97.44% and DAN obtained an accuracy of 88.33%, which are both close to the accuracy of 99.69% and 89.7% respectively as stated in each of the papers. The FERs are then tested with EVP results. Based on the confusion matrices acquired, the surprise emotion was observed to have the highest predicted rate among all seven emotions labels, namely, happy, angry, disgust, fear, sad, surprise and contempt. After examining a few error subjects, it was found out that this result is due to the network predicting the emotion of the image to be surprised although they actually belong to other emotions. A reason for that is because the faces in the output videos of EVP tend to be exaggerated where the subjects' eyebrows are raised with widened eyes and their mouths opened, which are all features of a surprised emotion. From here, one is able to realize that the output videos produced by EVP have limitations where the subjects' faces become too exaggerated which can cause inaccurate predictions.

As a conclusion, we used the state-of-the-art micro-expression recognizer, MicroExpSTCNN and the two facial expression recognizer, Distract Your Attention Network (DAN) and Frame Attention Networks (Emotion-FAN) for testing the EVP results to analyze the quality of the video generated by the EVP. All of these are the most relevant and important literature found to aid the team in the project. Through the experiments on the EVP results, a few limitations of the EVP output was discovered including the EVP not being able to generate negative emotion accurately due to the exaggerated facial expressions in the videos and the EVP is not able to generate micro-expressions in the output videos. The library of code that our team has developed follows the original EVP methodology as closely as possible, and developers should be able to build on it to produce functional facial video manipulation software.

Outcomes

Part I: EVP

Initially, the EVP base code lacked these components:

- 106 facial landmark detection
- Dynamic Time Warping to align audio clips such that they are of the same length
- 3D facial modelling code, whereby 3D parameters can be recovered from 2D facial landmarks for pose alignment
- Input generation code for both the audio2lm module and the lm2video module
- Pretraining code for content encoder, emotion encoder
- Cross-Reconstructed training code for the decoder
- Training code for the audio2lm module
- Converting the final output images of the vid2vid module into videos

Among the missing components of EVP, we were able to complete these parts:

- Dynamic Time Warping
- 3D facial modelling code
- Input generation code for both the audio2lm module and the lm2video module
- Pretraining code for content encoder
- Training code for the audio2lm module
- Converting the final output images of the vid2vid module into videos

Users of our code library will be able to build on our work to develop a functional facial manipulation software much more easily and efficiently.

Technical limitations of our code

- Since the 106 landmark detection code that the EVP authors used is not open source, we opted for dlib's 68 landmark detection code, aiming to adapt EVP to work for 68 facial landmarks. Through correspondence with the EVP authors, we were informed that this approach is not recommended due to the different definitions of close key points. One should replace the current landmark detection code with a suitable substitute should they try to build on our work.
- Pretraining code for the emotion encoder as well as the disentanglement module is not complete due to the lack of time. To obtain good results, one should follow the EVP paper to supplement this part of the code.

Technical limitations of the EVP design and future work

Throughout our code development process, we found some limitations of EVP, including:

- Due to the limited generalizability of the video rendering network used to produce the output videos from the final adapted landmarks, an individual model has to be trained for each subject. It would take a significant amount of time to generate facial videos for every new subject.

To improve EVP in terms of scalability, one could investigate methods to make the video rendering network generalizable,

- EVP authors trained and tested EVP using the MEAD dataset. Upon inspection of the dataset, we discovered that the emphasis of emotion expression was mainly placed on the visual facial expressions. Though certain subjects were labeled as “disgust” or “contempt”, the intended emotion was not reflected well in the audio and it sounded more neutral. Given that EVP extracts the emotion information solely from the audio, usage of the MEAD dataset is not ideal, unless one takes extra care to filter out videos of acceptable standard.

  Training and testing EVP on other datasets to see how the performance differs from using the MEAD dataset could be part of future research.

Part II: Analysis of EVP-generated videos

Analysis and discussions on the emotions of videos generated by EVP have been carried out to evaluate the capability of EVP producing expected emotional state as well as the limitations of EVP.

Testing EVP results using FERs and MER

We have analysed and compared the results based on the confusion matrices generated by FERs and MER. Before the testing phase, we trained the FERs and MER to ensure that the performance of our models on specific dataset are close to the results stated in paper. DAN (Wen et al., 2021) was trained on RAF-DB (Li et al., 2017) while Emotion-FAN (Meng et al., 2019) was trained on CK+ (Lucey et al., 2010). Fig 1 shows the training confusion matrices of DAN and Emotion-FAN we managed to obtain, which are 88.33% and 97.44% in terms of accuracy respectively, which are both similar with the records stated in the papers, 89.7% and 99.69%.

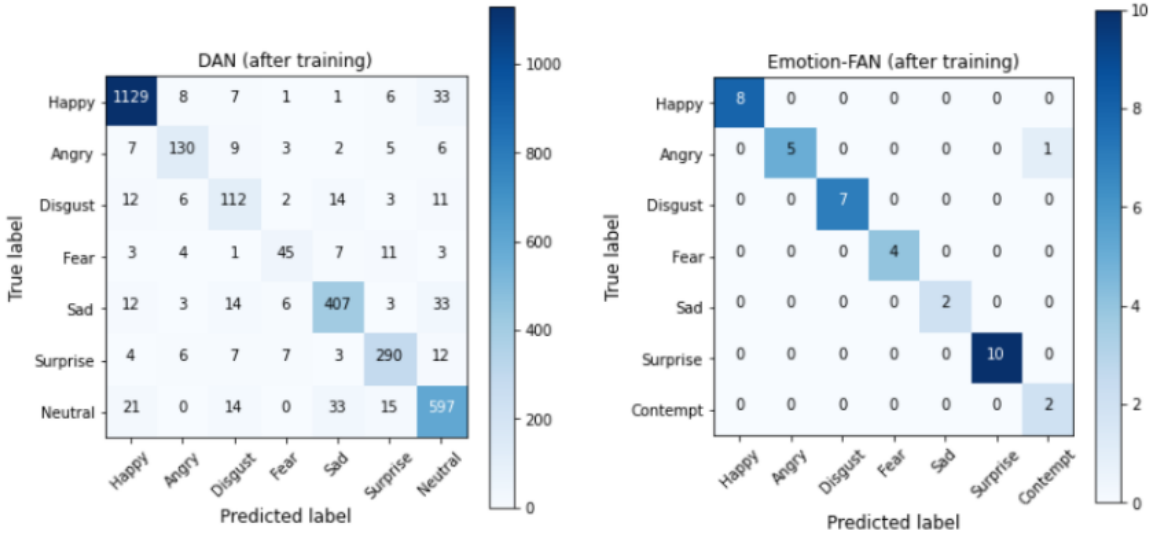


Fig 1. DAN (left confusion matrix) is trained with RAF-DB dataset and its accuracy is 88.33%. Emotion-FAN (right confusion matrix) is trained with CK+ dataset and its accuracy is 97.44%.

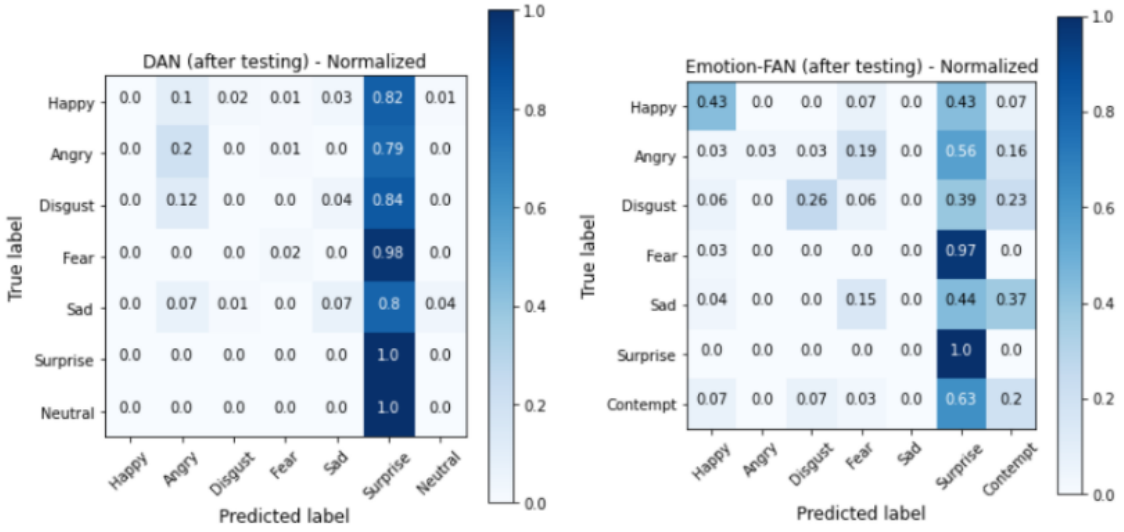


Fig 2. The normalized confusion matrices of testing results of using DAN (left) and Emotion-FAN (right) on EVP's results.

Fig 3. The error outputs which are predicted by FER as the surprise emotion. Their true emotions (from left to right) are anger, disgust, neutral, fear and happy respectively.

Based on the normalized confusion matrices in Fig. 2 generated by DAN and Emotion-FAN after testing on EVP results, we observed that surprise emotion has the highest predicted rate in facial expression detection. The examples of error subjects are examined to discover the reasons for abnormal detection rate. Fig. 3 listed some input frames that are predicted incorrectly by at least one FER as the surprise emotion although they actually belong to other emotions. From these examples, we could find that most of them hold at least one key feature of surprise emotion (adapted from), such as raised eyebrows or eyelids, widened eyes or open mouths. These features may affect the performance of facial expressions recognition since the features extracted by model could determine the classifications of facial expressions. Also, since surprise emotion has the highest detected rate, we may conclude that EVP tends to transform to a new face that consists of surprise's key features.

Apart from that, we have analyzed the confusion matrix produced by MicroExpSTCNN testing on the micro-expressions of EVP results. The model is trained with SMIC dataset and the training accuracy is 61.86%. Figure 4 shows the confusion matrix (left) of the training phase while another confusion matrix (right) is the testing results of the testing phase. The negative emotion (anger, sadness, disgust, fear) has the highest true positive rate. However, the majority of the positive (happy) and surprise emotions are predicted as negative emotions.

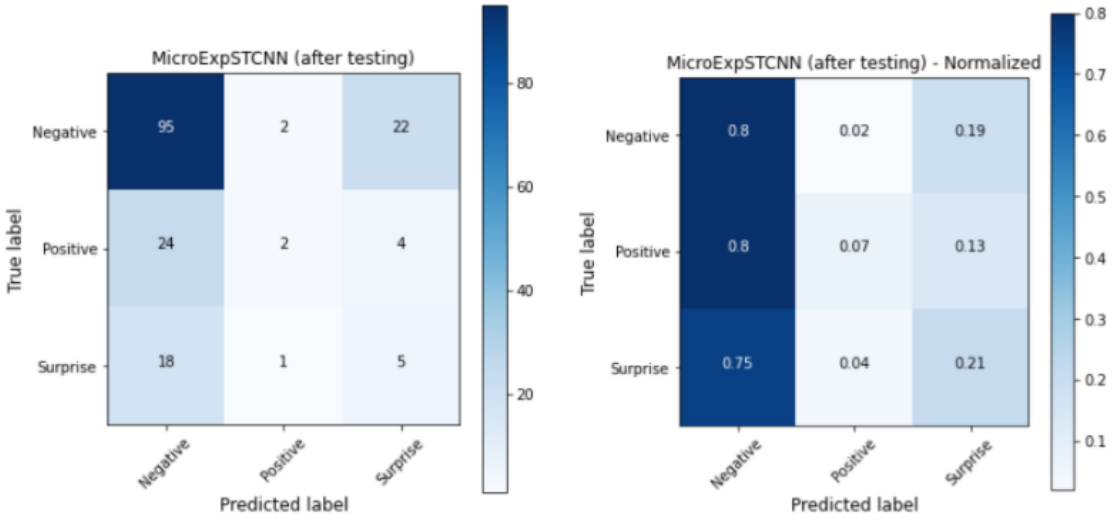


Fig 4. MicroExpSTCNN (left confusion matrix) is trained with SMIC dataset and the training accuracy is 61.86%, which is similar to the accuracy of 68.75% stated in the paper. The right confusion matrix shows its testing results on EVP

The examples of testing output and the videos from SMIC micro-expression dataset are evaluated in order to figure out the reasons for incorrectly predicting positive and surprise videos. It is undeniable that features extracted from frames during the training process directly affects the performance of the model. In the augmentation part of MER, only the first 18 frames of each input video are fed into the model for training usage. Therefore, the features of 18 frames should contain the data of vital muscle movements of micro-expression which could be learned by model to classify the emotions. However, the video lengths of the SMIC dataset are much longer than all EVP-generated videos, which means that the most significant part of SMIC's videos that shows the key features or movements of micro-expression may not be extracted.

From figure 5 and figure 6, we discovered that the negative subjects in the training dataset have more features such as raised eyebrows or eyelids, widened eyes compared to the positive and surprise subjects in SMIC, and these features are almost shown in the error output of EVP results. Therefore, this is the possible reason that most of the EVP videos are predicted as negative emotion, which is due to the features extracted from training dataset that causes bias in the emotions classification part.

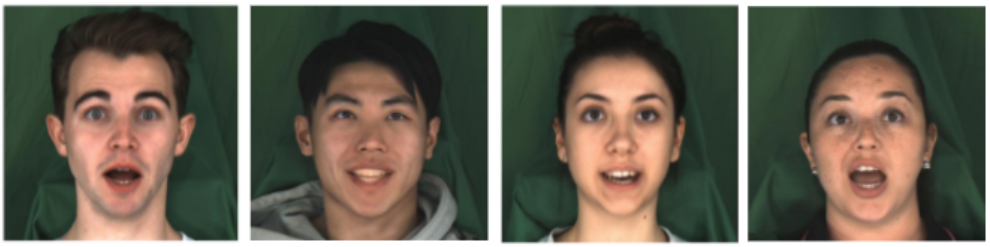


Fig 5. The examples from EVP results which are wrongly predicted as negative emotion.

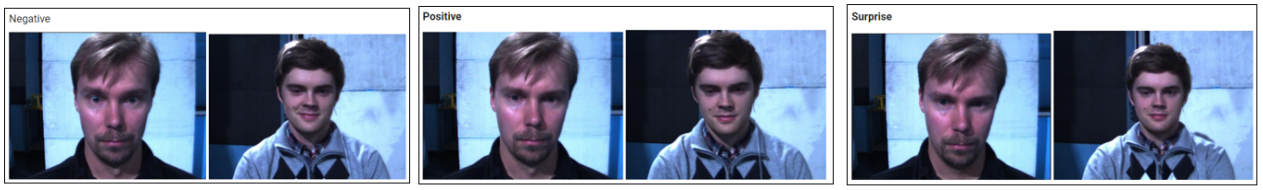


Fig 6. The frames of different emotions from the training dataset. The faces of negative emotion (left box) have the features of widened eyes and raised eyebrows. The positive (middle box) and surprise (right box) emotions does not have any significant muscle movements.

Limitations

Based on the analysis above, we discovered some limitations of EVP, including:

- EVP could not generate videos with negative emotion very accurately, as all have been predicted as surprise emotion.
- The amount of frames extracted by MicroExpSTCNN may not be enough for extraction of key features from the videos.
- EVP could not precisely produce the micro-expressions in the result videos.

Methodology

Software and tools

The following are all of the software and tools used for our project.

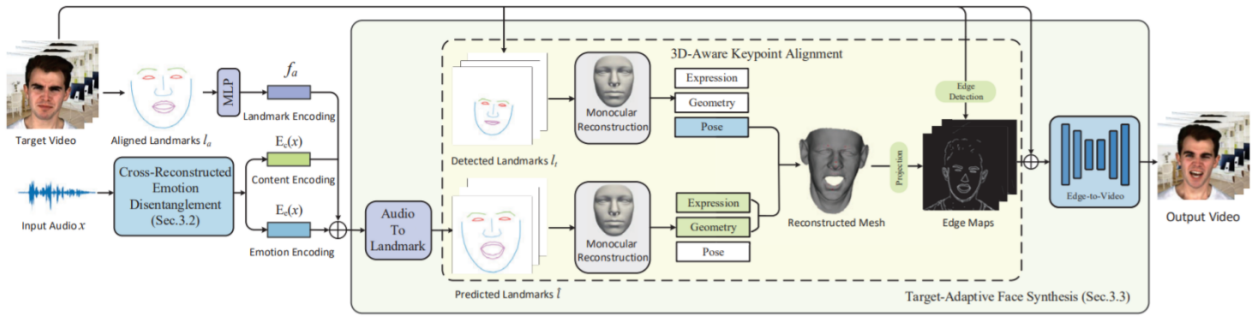
Name	Description
GitHub	GitHub is a version control system that integrates with Google Colab such that code repositories can be cloned in Google Colab conveniently.
Google Drive	Google Drive allows large datasets for the project to be stored. Also, it can be used together with Google Colab, as it can be mounted to a Google Colab notebook.
Google Colab Pro	Google Colab is an online cloud service that allows arbitrary python code to be written and executed through the browser. It provides real-time collaboration, GPUs and pre-installed libraries which ease us for developing code and running machine learning projects. We have subscribed to Google Colab Pro, which offers faster GPUs, more RAM and disk memory, longer runtimes and less prone to disconnecting than the free version of Google Colab.
Python 3.6	Python is an interpreted programming language with many libraries for machine learning available.
Software libraries	Software libraries that provide machine learning related functionality have been utilised in implementation.

Part I: EVP

Design: Overview of EVP architecture

Figure 7

Overview of EVP architecture



Note. An overview of EVP architecture. Adapted from "Audio-driven emotional video portraits," by Ji, X., Zhou, H., Wang, K., Wu, W., Loy, C. C., Cao, X., & Xu, F., 2021, *Proceedings of the 1EEE/CVF Conference on Computer Vision and Pattern Recognition*, 14080 - 14089 (https://openaccess.thecvf.com/content/CVPR2021/papers/Ji_Audio-Driven_Emotional_Video_Portraits_CVPR_2021_paper.pdf). Copyright 2021 by Ji et al.

We follow the design of EVP which you can see from the image above. EVP is composed of two main modules, namely audio2lm and lm2video. Finally, EVP authors also used the vid2vid technique developed by other researchers to produce the final output videos.

Given an input audio clip and a video clip, target video is manipulated to reflect the emotion detected from the input audio. The input audio is fed into the Cross-Reconstructed Emotion Disentanglement module to extract the content and emotion encodings, while aligned landmarks are extracted from the target video frames, from which landmark encodings are obtained. With the landmark, content and emotion encoding, the audio2lm module will be able to predict the facial landmarks.

The lm2video module obtains the detected landmarks from the target video frames. In order to preserve the pose of the target from the original video, we combine the pose parameters obtained from the 3D face model constructed from the detected landmarks, as well as the expression and geometry parameters obtained from the 3D face model constructed from the predicted landmarks. With that we have the reconstructed face mesh, whereby the landmarks can be projected to 2D to obtain the final adapted landmarks.

Finally, the vid2vid module can generate edge maps from the adapted landmarks. Using the edge maps and the target video frames, the final output video can be generated, whereby the subject in the video is now reflecting the desired emotion.

Our contribution

As we are building on the EVP authors' efforts, we will highlight the contributions that we have made to the code. It is important to note that we tried to follow the authors work as closely as possible by referring to the published paper and code comments, but the code base came with little documentation, therefore we were required to work out solutions to certain problems based on our own understanding.

Our approach to deal with these problems will be explained in detail, with care to mention what parts of implementation were made following the authors and others that were made based on our own reasonings.

1) Dynamic Time Warping

Paired audio clips with the same speech with different emotions and the same length are required to enable cross-reconstructed training for disentangling information extracted in the content and emotion encodings. Since obtaining such audio clips is a highly impractical task, EVP authors opt to obtain audio clips with the same speech content but different emotions, and use Dynamic Time Warping to make them equal in length.

Recall that our input to the software is an audio clip x and a video clip (with audio y), we choose to warp audio x to match the length of audio y. This is because there will be unavoidable distortions and loss of content in the warped audio. We choose to warp audio x to match audio y instead of the other way around. The goal is to preserve the speech content in audio y, since the emotional information loss in audio x will be less perceivable.

2) Audio2lm module input generation

The main input to the audio2lm module are the aligned landmarks, 16 principal components of the landmarks as well as reference displacements.

To obtain the aligned landmarks, the target video is split into a number of frames. The facial landmarks of the subject in the video are detected and we obtain the average value of landmarks across all video frames.

We used the inbuilt PCA function in sklearn library to compute the principal components of facial landmarks. The 68 landmarks detected are first standardized before fitting and transforming on the training data. In the PCA function, 16 principal components are kept as required for audio2lm.

Reference landmark displacements are used as the ground truth for training, whereby EVP authors try to minimize the difference between reference landmarks and the predicted landmarks.

Say we have a pair of videos for training, whereby the subjects are speaking the same speech with different emotions. Our goal is to alter video x with emotion m to show emotion n from video y.

The aligned landmarks and principal components are obtained from video x. On the other hand, the reference landmark displacements are obtained from video y, since we aim to reproduce emotion n shown in it.

3) Training code

EVP authors pretrained the content encoder on a lip-reading dataset and the emotion encoder through an emotion classification task prior to training the disentanglement module. We have contributed code to pretrain the content encoder.

EVP authors pretrain the content encoder on LRW (Chung & Zisserman, 2016) as it had little emotion to reduce the emotion information from content encodings. However, our team could not obtain the LRW dataset due to its data security restrictions. Instead, we trained the content encoder with a subset of neutral audio clips from the MEAD dataset, making sure to use subjects that were not chosen to train the disentanglement module.

Following the EVP authors work, the loss function was supervised with content loss, to ensure audio with the same speech should have similar content encodings. Our pretrained content encoder could produce zero loss consistently when tested with audios of the same speech.

Besides, our team has also contributed code for training the audio2lm module as a whole.

4) lm2video module input generation

The main input to the lm2video module are the target frames, detected landmarks, detected pose parameters as well as the parameters of the predicted landmarks.

Target frames are obtained by splitting the input videos into images, and a 68 facial landmark detector was used to detect the facial landmarks from these images.

To obtain the 3D parameters from the detected and predicted landmarks, we adapted open source 3D face reconstruction code (Smith, 2011) to fit a 3D face model from the 2D facial landmarks. With that, we were able to obtain the pose parameters corresponding to the detected landmarks and the expression, geometry and pose parameters from the predicted landmarks.

5) Image to video conversion

The output for the vid2vid module is images that make up the final output video of EVP. Our team supplemented code to convert these images into videos.

6) Code comments

Our team has added code comments based on our best interpretation of the EVP authors' intention wherever necessary to help users understand it more efficiently.

Part II: Analysis of EVP-generated videos

In the analysis, state-of-the-art facial expression recognition and micro expression recognition are applied on the output of EVP to verify the accuracy of emotions detected with its true label. We have utilized Frame Attention Networks proposed by Meng et al. (2019) and Distract Your Attention Network designed by Wen et al. (2021) for facial expression recognition whereas the MicroExpSTCNN models by Reddy et al. (2019) is used for detecting micro-expressions. The comparisons in terms of overall accuracies of FERs and MER as well as the accuracies on specific emotions recognition have been done to evaluate the performance and limitations of EVP.

The augmentation of training and testing data are done according to the ways given in each FERs and MER. Each recognition is trained with a different dataset: Emotion-FAN is trained with CK+ dataset, DAN is trained with RAF-DB dataset. The output of the training and testing parts of all recognition would be plotted as confusion matrices for usage of analysis and comparisons.

EVP results preparation for testing

A dataset of EVP videos were generated using the authors' pretrained models for testing purposes. We chose 5 subjects (3 males and 2 females) of the Multi-view Emotional Audio-visual Dataset (MEAD) for video generation. For each subject, around 6 videos were generated for each of the 8 emotion categories, namely: happy, sad, disgust, angry, contempt, surprise, fear and neutral.

Following the method Emotion-FAN was tested on the CK+ dataset, we prepare the last three frames for each video to test it. Note that since Emotion-FAN does not recognize the neutral emotion, only the remaining 7 emotion categories were tested.

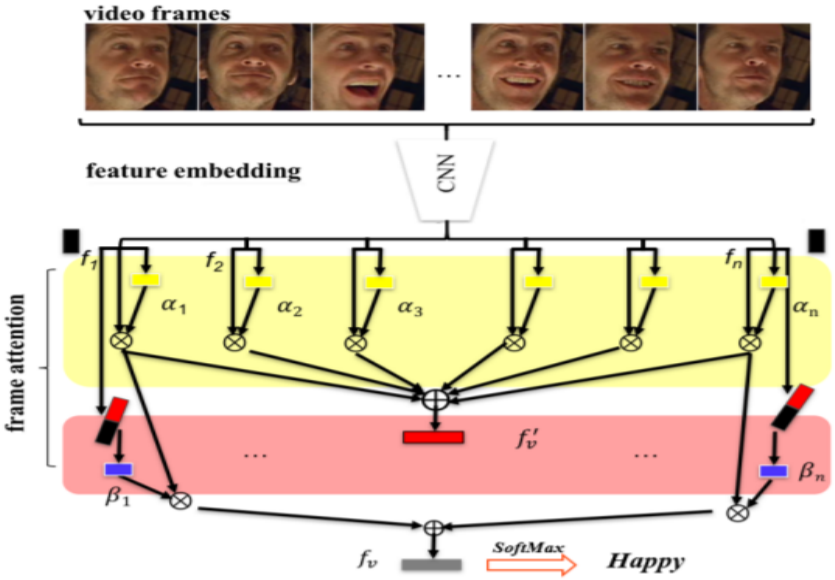
Since DAN accepts images as input instead of videos, we prepare the data for DAN in the same fashion as for testing Emotion-FAN whereby the last three frames of each video are used. The reasoning behind this was that we wanted to be as fair as possible and test the same frames for each video on both Emotion-FAN and DAN. Note that since DAN does not recognize the contempt emotion, only the remaining 7 emotion categories were tested.

For MicroExpSTCNN, the EVP videos were fed into it. Note that MicroExpSTCNN only recognizes negative, positive and surprise emotions, whereby angry, fear, disgust and sad were categorized as negative emotions.

Frame Attention Networks (Emotion-FAN)

Figure 8

Emotion-FAN architecture



Note. An overview of Emotion-FAN architecture. Adapted from "Frame Attention Networks for Facial Expression Recognition in Videos," by Meng, D., Peng, X., Wang, K., & Qiao, Y.,2019, *2019 IEEE International Conference on Image Processing (ICIP)*, 3866-3870 (https://doi.org/10.1109/ICIP.2019.8803603). Copyright 2019 by Meng et al.

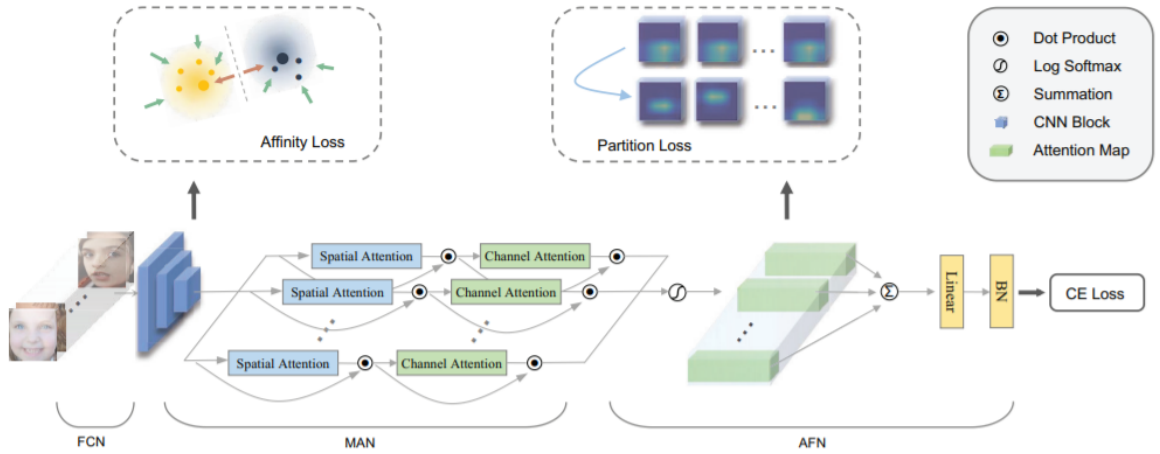
The image above shows the Emotion-FAN architecture. Emotion-FAN is constructed with two modules, feature embedding module and feature attention module. Feature embedding module is a convolutional neural network which embeds the face images into feature vectors. Feature attention module would then combine these vectors to output a single discriminative video representation. Before aggregating the vectors, the feature attention module learns self-attention weights which are assigned to each individual frame. The global representation features include all self-attention weights and are further used to refine new relation-attention weights, which will be learned by the feature attention module to output the final compact features and emotion detected.

Distract Your Attention Network (DAN)

The author of DAN suggested that attention mechanisms could improve the effectiveness of neural architecture by only concentrating on the most relevant part of input sequences, meanwhile achieving or even surpassing the performance of state-of-the-art approaches. Thus, they proposed a facial expression recognition algorithm that could capture multiple non-overlapping attentive features. DAN consists of three components: Feature Clustering Network (FCN), Multi-head cross Attention Network (MAN) and Attention Fusion Network (AFN).

Figure 9

DAN architecture



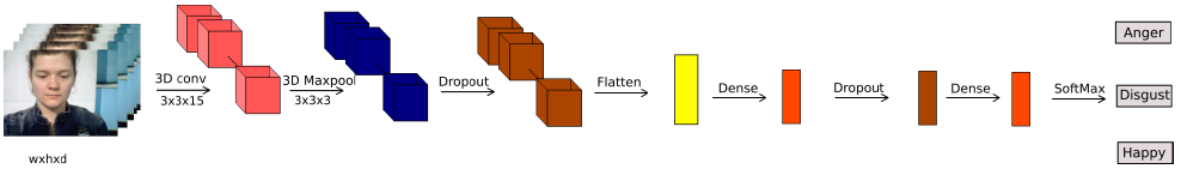
Note. An overview of DAN's architecture. Adapted from "Distract Your Attention: Multi-head Cross Attention Network for Facial Expression Recognition," by Wen, Z., Lin, W., Wang, T., & Xu, G.,2021, *arXiv preprint arXiv:2109.07270*. Copyright 2021 by Wen et al.

As shown in Fig. 8, FCN takes a batch of input frames to extract and cluster the basic features. At this stage, a discriminative loss function: affinity loss, is applied to maximize the inter-class distance at the same time minimizing the intra-class distance such that the model could learn discriminative features of different emotions. After that, a series of spatial attention units and channel attention units in MAN would extract the spatial and channel information from the output of FCN then incorporate these features into an attention map. The final sub-network, AFN would fuse these attention maps and focus on learning different crucial regions as well as avoid overlapping features by maximizing the variance among attention maps using partition loss. The predicted emotion and cross-entropy loss are computed after processing through these networks.

MicroExpSTCNN

Figure 10

MicroExpSTCNN architecture



Note. An overview of MicroExpSTCNN architecture. Adapted from "Spontaneous facial micro-expression recognition using 3D spatiotemporal convolutional neural networks," by Reddy, S. P.

T., Karri, S. T., Dubey, S. R., & Mukherjee, S., 2019, *2019 International Joint Conference on Neural Networks (IJCNN)*, 1 - 8 (https://doi.org/10.1109/IJCNN.2019.8852419).

MicroExpSTCNN is a 3D-CNN model which is sequentially formed by a stack of 3D convolutional layers, 3D Max Pooling layers, fully connected layers, activation functions and dropout layers. The 3D convolutional layers are used to extract spatial and temporal information of the full face region and the max pooling layer would then pick the best features from output of last layers. The dropout layer is used to improve the generalization capability of the network and prevent the overfitting of training data. Flatten layer helps to transform multi-dimensional input into the size which is required by dense layer while dense layers increase more non-linearity in the whole network in the form of hierarchical feature extraction. The final softmax layer evaluates the class scores for the classes being used by the dataset.

Software deliverables

Summary of software deliverables

The software deliverable is a library of code that uses the EVP as the base code. Google Colab notebook with our work will be provided and the users can run the software from there. As we are focusing more on the analysis, the output of the state-of-the-art MER and two FERs after running them with EVP results are also an important part of the deliverable. These can also be run through the Google Colab notebook that we prepared that is dedicated to producing analysis output. The MER used is the MicroExpSTCNN while the two FERs used are the Distract Your Attention Network (DAN) and Frame Attention Networks (Emotion-FAN).

ii. Sample screenshots and description of usage (Note a more complete set of screenshots will be included in the separate User Guides assignment. The material here is only to provide a very brief idea of what can be expected)

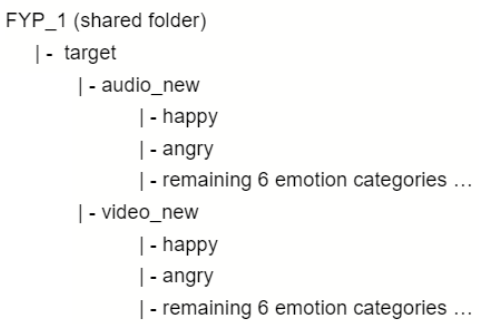
EVP code:

Many of the major components of the original code are missing, our team was unable to complete all the missing codes but we solved much of the existing problems with the code. To run the code, a few steps are to be taken:

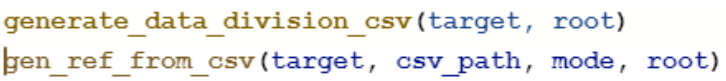
Set up: Users should have access to the MeSelves folder and all the cells in the Set up section should be run.

audio2lm module:
Generates the predicted facial landmarks based on the given input audios.

1. **Input preparation:** To prepare the input, users should set up a directory structure as shown in Sample Screenshot 1. All cells up to the “prepare input section” should be run and the gen_all_pca_mean function is called to generate the pca and mean of all target videos. Functions such as the ones shown in Sample Screenshot 2 are then called to prepare the csv files for training, validation and testing and also to generate the reference displacements.

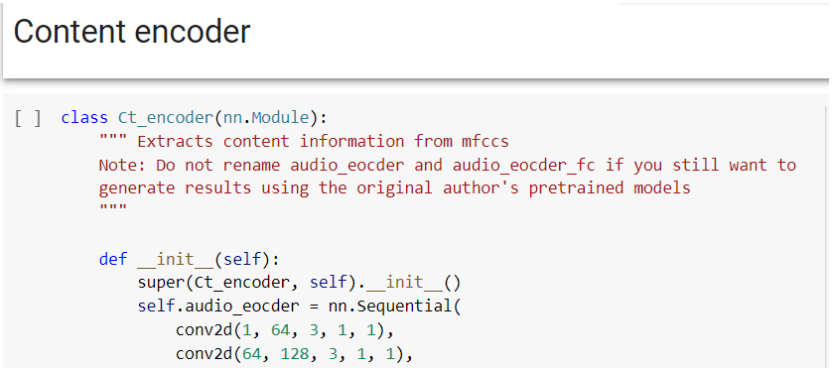


Sample Screenshot 1



Sample Screenshot 2

2. **Training**: All codes from the content encoder section as shown in Sample Screenshot 3 are to be run in order. One will have to prepare the emotion encoder for pretraining on their own as well as the decoder. After pretraining, all cells in the Train section should be run.



Sample Screenshot 3

3. **Testing**: All cells in the Test section are to be run and certain cells such the one shown in Sample Screenshot 4 should be uncommented and run.

```
# # Example for testing using original authors pretrained model M003
# # You will need to prepare your own yaml file if not using their provided models

# opt["audio"] = "/content/drive/MyDrive/FYP_1/EVP_datasets/audio2lm/data/M003/audio/angry/001.m4a"
# opt["emo_audio"] = "/content/drive/MyDrive/FYP_1/EVP_datasets/audio2lm/data/M003/audio/fear/001.m4a"

# with open("/content/drive/MyDrive/FYP_1/audio2lm/config/M003_test.yaml") as f:
#   config = yaml.load(f)

# test(opt, config)
```

Sample Screenshot 4

lm2video module

Generates the adapted facial landmarks and prepares target frames as input for vid2vid.

1. **Input preparation:** After generating results from the audio2lm module, input for the lm2video module should be prepared. The steps include preparing target frames, generating detected landmarks for target frames, generating detected pose parameters and generating predicted parameters from the predicted landmark.

```
prepare_target_frames("M007",
"/content/drive/MyDrive/MeSelves/FYP_1/M007/M007_train_set.csv",
"/content/drive/MyDrive/MeSelves/FYP_1")

!python gen_ldmk_from_img.py
/content/drive/MyDrive/MeSelves/FYP_1/EVP_datasets/lm2video/data_meselves/3DMM

!python gen_parameters.py detect
/content/drive/MyDrive/MeSelves/FYP_1/EVP_datasets/lm2video/data_meselves

!python gen_parameters.py predict
/content/drive/MyDrive/MeSelves/FYP_1/EVP_datasets/lm2video/data_meselves
```

Sample Screenshot 5

2. **Execute lm2video module**: Run all cells in filter1 and lm2map sections and uncomment cells as shown in Sample Screenshot 5 to start execution.

```
# Example using author's pretrained model

with open("/content/drive/MyDrive/FYP_1/lm2video/ldmk_idx.txt", 'r') as f:
    line = f.readline()

info = line.strip().split()
ldmk_idx = [int(it) for it in info]

params = {}
exp2ldmk = Exp2Ldmk(params)


target_name = "M030"
root = "/content/drive/MyDrive/FYP_1/EVP_datasets/lm2video/data"
filepath = os.path.join(root, target_name, "3DMM/test_results") # predicted parameters
target_path = os.path.join(root, target_name, "background") # detected pose parameters
txt_root = target_path # root for detected landmarks
image_root = txt_root

save_root = '/content/drive/MyDrive/EVP_complete/vid2vid/datasets/face/test_keypoints_' + target_name + '_pose_test/' # J: final ad
out_root = '/content/drive/MyDrive/EVP_complete/vid2vid/datasets/face/test_img_' + target_name + '_pose_test/' # J: target frames c

change_pose(filepath, target_path, txt_root, save_root, target_name, True)
prepare_image(save_root,image_root, out_root, target_name)
```

Sample Screenshot 6

vid2vid module

1. **Execute vid2vid:** All cells in the vid2vid section must be run and the function in the generate output section as shown in Sample Screenshot 7 to convert the images into a video, which is the final output video of EVP should be called accordingly.

```
# Example

# img_root = "/content/drive/EVP_complete/vid2vid/results/M003/test_latest/"
# output_dir = "/content/drive/EVP_complete/vid2vid/results/M003/M003_video/"
# generate_output_videos(img_root, output_dir)
```

Sample Screenshot 7

Analysis:

The output needed for the EVP analysis can be obtained through the following steps:

The EVP analysis can be accessed by first cloning the code from github to google drive.

```
[ ]  !git clone https://github.com/J-asy/DAN
```

```
[ ]  !git clone https://github.com/J-asy/Emotion-FAN
```

```
[ ]  %cd /content/drive/MyDrive/mer/smic
```

Sample screenshot 1

We can then start the training and testing of the FER and MER

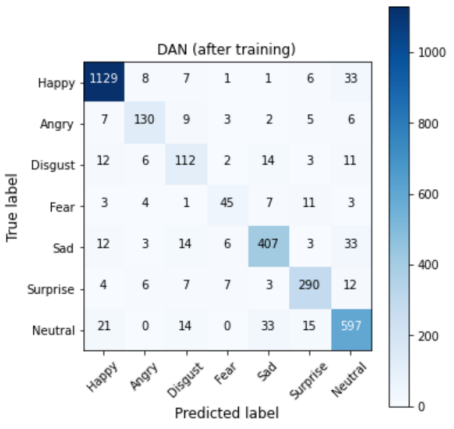
```
[ ]  # training on RAF
     %cd /content/DAN
     !CUDA_VISIBLE_DEVICES=0 python rafdb.py --mode train
```

```
[ ]  # testing on EVP using pretrained model
     !CUDA_VISIBLE_DEVICES=0 python rafdb.py --test_on evp
```

Sample screenshot 2

And generate the confusion matrix

```
# plot DAN's training confusion matrix
plot_confusion_matrix(DAN_train_cm, new_DAN_labels, title="DAN (after training)")
```



Sample screenshot 3

Summary of Software Qualities Handling

i. Robustness
We tried to apply testing functions and assertions in the EVP code to guarantee the output of modules are able to generate videos with desired emotion at the final stage. For the robustness of FERs and MERs, different types of loss while training the dataset will be processed and learned again by the model, in order to improve the performance of recognitions.

ii. Security
Since the base code of EVP, facial expressions recognition, micro expressions recognition and the dataset we used for training and testing are open-source, we do not apply security systems in our project.

iii. Usability
As to improve the useability of EVP code, we have prepared the user guide and the cell-by-cell code execution in ipynb for the ease of running the project. The libraries needed by EVP and our source code are able to be cloned or downloaded online, which means that the user could directly execute it without needing much effort on importing or uploading dataset since the model is pre-trained by EVP's author.

iv. Scalability
Due to the fact that there are parts missing in EVP, especially the 106 landmarks detection technique, we think that the performance of EVP is hard to be extended or improved unless the author publishes more techniques they used in EVP. Nevertheless, the performance of facial expression and micro-expression analysis may be further improved by training with different dataset and applying different hyperparameters to get better accuracies in EVP results.

v. Documentation and Maintainability
In our ipynb files which consist of the execution code of EVP, micro expression recognition and facial expression recognition, we have organized the sections and code according to different usage or modules. Furthermore, the function comments, block comments and in-line comments as well as additional explanations for important parts are included in most parts of our project source code. The clear user guide is also provided to help the user to execute the programs besides gaining better understanding on the implementation of the project.

To ensure that our project could be further maintained or improved on its performance, we have submitted complete modules, requirements, testing ways and dataset used so that the users could fix bugs or make alterations to the code especially under the help of explanation and comments in our project. However, due to some missing parts in EVP code, the modification on EVP may require additional efforts to research on the code of missing parts.

Critical Discussion

In the planning stage, our project focus was to develop a novel facial video manipulation technique using EVP as our base code. However, over the course of code development, we have discovered that there were many incomplete components of EVP.

The incomplete components were substantial, and together with the lack of documentation, it became impractical for us to follow our initial project proposal, as completing the base code itself would take a long amount of time. We would not be able to complete EVP and develop a novel variation within the deadline.

Hence, upon approval from our project supervisor, our project focus was shifted. Since solely attempting to complete EVP itself would mean we were simply following the original author's methodology, it would not bring much value. We decided that it would be best to produce a new novelty aspect, which is analysing EVP results using facial expression recognizers and micro-expression recognizers and transform the current EVP code that we had into a software library.

We showed good progress in our project development every week and managed to meet our newly formulated requirements well. With our efforts from Week 1 through the first half of Week 4, we have delivered a well documented library of code that will help future programmers to use to build a functional facial video manipulation software. In the remaining time left in the semester, we managed to deliver a novel analysis even though we changed the focus of our project relatively late in the semester. Hence, despite deviating from the original plan, our project is considered a success.

Conclusion

The whole project, Me Myself and I, had been successfully implemented during FIT3161 and FIT3162. In last semester, we had prepared the project proposal report and some code analysis, whereas in the current semester, we focused on developing missing parts of EVP code besides working on our novelty part, the analysis of EVP-generated results. For EVP, the missing components that we have achieved include Dynamic Time Warping, 3D face modelling, input generation code for modules, training of content encoder and audio2lm module, and finally the conversion of vid2vid's output images into videos. For the analysis of output videos, we leveraged the Distract Your Attention Network, Emotion Frame Attention Network to detect facial expressions while MicroExpSTCNN is utilized to detect the micro-expression.Results of testing the EVP-generated videos with these recognizers had been analyzed to evaluate the capability and the limitations of EVP. The future possible work for this project would be improving EVP by making it more scalable by generalizing the video rendering network. One could also consider training and testing EVP on other datasets to see whether better performance can be achieved compared to using the MEAD dataset.

Appendix

a) Sample source code

Sample source code for EVP

The following function is called to generate the predicted landmarks using the audio2lm module after training.

```
def test(opt, config):
    """ Generate predicted landmarks from AT_emotion based on input audios

    Args:
      opt: dictionary object that stores the input settings
      config: dictionary object parsed from yaml file that stores some file paths
    """
    os.environ["CUDA_VISIBLE_DEVICES"] = opt["device_ids"]

    pca = torch.FloatTensor(np.load(config['pca'])[:,:16]).cuda()
    mean = torch.FloatTensor(np.load(config['mean'])).cuda()

    encoder = AT_emotion(opt, config)

    if opt["cuda"]:
        encoder = encoder.cuda()

    # load the trained model
    state_dict = multi2single(config['at_model'], 0)
    encoder.load_state_dict(state_dict)
    encoder.eval()

    example_landmark = np.load(config['mean'])
    example_landmark = example_landmark.reshape(opt["num_landmarks"], 2)
    example_landmark =  example_landmark.reshape((1, example_landmark.shape[0] *
                                                  example_landmark.shape[1]))

    if opt["cuda"]:
        example_landmark = Variable(torch.FloatTensor(example_landmark.astype(float)) ).cuda()

    # get aligned landmarks
    example_landmark  = example_landmark - mean.expand_as(example_landmark)
    example_landmark = torch.mm(example_landmark,  pca)

    # load speech
    test_speech, sr = librosa.load(opt["audio"], sr=16000)
    clip = check_volume(test_speech,sr)
    emo_speech, sr = librosa.load(opt["emo_audio"], sr=16000)

    # Dynamic time warping to align audio clips
    wp = get_dtw_warp_path(test_speech, emo_speech, sr)
    warped_emo_speech = dtw_warp(wp, test_speech, emo_speech)

    # get MFCC features
    mfcc = get_mfcc(test_speech)
    input_mfcc = process_mfcc(mfcc)

    mfcc_emo = get_mfcc(warped_emo_speech)
    emo_mfcc = process_mfcc(mfcc_emo)
```

```
    print ('====================================')
    print ('Start to generate images')

    ind = 3
    with torch.no_grad():

        fake_lmark = encoder(example_landmark, input_mfcc, emo_mfcc) # landmark displacements
        fake_lmark = fake_lmark.view(fake_lmark.size(0)*fake_lmark.size(1), 16)

        fake_lmark = fake_lmark + example_landmark.expand_as(fake_lmark) # deduced landmarks
        fake_lmark = torch.mm( fake_lmark, pca.t() )
        fake_lmark = fake_lmark + mean.expand_as(fake_lmark)

        fake_lmark = fake_lmark.unsqueeze(0)
        fake_lmark = fake_lmark.data.cpu().numpy()
        fake_lmark = np.reshape(fake_lmark, (fake_lmark.shape[1], opt["num_landmarks"], 2))
        # fake_lmark shape (t, 106, 2) - t frames, x and y coordinate of 106 landmarks for each
        # frame

        fake_lmark = change_mouth(fake_lmark, clip)

        # generate a video to visualize predicted landmarks of each frame
        mouth_video_writer = VideoWriter(config['video_dir'], 256, 256, 25)
        mouth_img = []
        for i in range(len(fake_lmark)):
            mouth_img.append(draw_mouth(fake_lmark[i]*255, 256, 256))
            mouth_video_writer.write_frame(draw_mouth(fake_lmark[i]*255, 256, 256))
        mouth_video_writer.end()

        # save predicted landmarks as a text file (input for lm2video)
        temp = fake_lmark.reshape(fake_lmark.shape[0], fake_lmark.shape[1]*fake_lmark.shape[2])
        pred_lm_filename = config['lm_output_dir']
        np.savetxt(pred_lm_filename, temp*255, delimiter=' ', fmt='%d')

        add_audio(config['video_dir'], opt["audio"])
        print ('The generated video is: {}'.format(config['video_dir'].replace('.mp4','.mov')))
```

Here the function is being called to generate the predicted landmarks for target M003 that reflects the fear emotion.

```
# Example for testing using original authors pretrained model M003

opt["target"] = "M003"
opt["audio"] = "/content/drive/MyDrive/EVP_complete/EVP_datasets/audio2lm/data/" +
opt['target']+ "/audio/angry/001.m4a"
opt["emo_audio"] = "/content/drive/MyDrive/EVP_complete/EVP_datasets/audio2lm/data/" +
opt['target'] + "/audio/fear/001.m4a"

with open("/content/drive/MyDrive/EVP_complete/audio2lm/config/" + opt['target'] + "_test.yaml")
as f:
  config = yaml.load(f)
  config["lm_output_dir"] = os.path.join(config["lm_output_dir"], opt['target'], opt["target"] +
"_angry_fear_001.txt")
  print(config["lm_output_dir"])

test(opt, config)
```

Sample source code for analysis of EVP results

The following snippet shows the DAN and Emotion-FAN facial expression recognizers and MicroExpSTCNN being trained on other datasets and tested on EVP results.

```
# Sample source code - Analysis


##### 1. DAN facial expression recognizer #####

!git clone https://github.com/J-asy/DAN

# i) training DAN on RAF-db
!CUDA_VISIBLE_DEVICES=0 python rafdb.py --mode train

# read confusion matrixes from the training log file
DAN_train_log = '/content/drive/MyDrive/FER_complete/DAN/log/train_log_raf'
DAN_labels = ['Surprise', 'Fear', 'Disgust', 'Happy', 'Sad', 'Angry', 'Neutral']
DAN_train_cm, DAN_train_cm_norm, new_DAN_labels = obtain_confusion_matrices(DAN_train_log, 8,
DAN_labels, rearrange=True)

# plot the confusion matrix for DAN after training
plot_confusion_matrix(DAN_train_cm, new_DAN_labels, title="DAN (after training)")

# ii) testing DAN on EVP results using the trained model
!CUDA_VISIBLE_DEVICES=0 python rafdb.py --test_on evp

# read confusion matrixes from the testing log file
DAN_test_log = '/content/drive/MyDrive/FER_complete/DAN/log/test_log_evp'
DAN_test_cm, DAN_test_cm_norm, new_DAN_labels = obtain_confusion_matrices(DAN_test_log, 8,
DAN_labels, rearrange=True)

# plot the confusion matrix for DAN after testing
plot_confusion_matrix(DAN_test_cm, new_DAN_labels, title="DAN (after testing)")



##### 2. Emotion-FAN facial expression recognizer #####

!git clone https://github.com/J-asy/Emotion-FAN

# i) training Emotion-FAN on CK+
!CUDA_VISIBLE_DEVICES=0 python fan_ckplus_traintest.py --at_type 1 --fold 1

# read confusion matrixes from the training log file
FAN_train_log = '/content/drive/MyDrive/FER/emotion-FAN/saved/fan_ckplus_date_08-Jan-2022-15-39-
52.txt'
emotion_FAN_labels = ['Happy','Angry','Disgust','Fear','Sad','Contempt','Surprise']
FAN_train_cm, FAN_train_cm_norm, new_emotion_FAN_labels =
obtain_confusion_matrices(FAN_train_log, 8, emotion_FAN_labels, rearrange=True)

# plot the confusion matrix for Emotion-FAN after training
plot_confusion_matrix(FAN_train_cm, new_emotion_FAN_labels, title="Emotion-FAN (after
training)")

# ii) testing Emotion-FAN on EVP results using the trained model
new_model_path = "/content/drive/MyDrive/FER/emotion-FAN/saved/self_relation-
attention_2_97.4359"
!CUDA_VISIBLE_DEVICES=0 python evp_test.py --at_type 1 --fold 1 --load_model $new_model_path

# read confusion matrixes from the testing log file
FAN_test_log = '/content/drive/MyDrive/FER/emotion-FAN/saved/fan_evp_date_08-Jan-2022-16-07-
37.txt'
emotion_FAN_labels = ['Happy','Angry','Disgust','Fear','Sad','Contempt','Surprise']
FAN_test_cm, FAN_test_cm_norm, new_emotion_FAN_labels = obtain_confusion_matrices(FAN_test_log,
8, emotion_FAN_labels, rearrange=True)

# plot the confusion matrix for Emotion-FAN after testing
plot_confusion_matrix(FAN_test_cm, new_emotion_FAN_labels, title="Emotion-FAN (after testing)")
```

```
##### 3. MicroExpSTCNN micro-expression recognizer ####

# training MicroExpSTCNN on SMIC and testing on EVP results
!python3 MicroExpSTCNN.py

# read confusion matrixes from the training log file
train_log = '/content/drive/MyDrive/mer/smic/smic-train.txt'
train_cm, train_cm_norm = read_cm_from_txt(train_log, 4)

# plot the confusion matrix for MicroExpSTCNN after training
smic_labels = ['Negative','Positive','Surprise']
plot_confusion_matrix(train_cm, smic_labels, title="MicroExpSTCNN (after training)")

# read confusion matrixes from the testing log file
test_log = '/content/drive/MyDrive/mer/smic/smic-test.txt'
test_cm, test_cm_norm = read_cm_from_txt(test_log, 4)

# plot the confusion matrix for MicroExpSTCNN after testing
plot_confusion_matrix(test_cm, smic_labels, title="MicroExpSTCNN (after testing)")
```

b) Annex

Section	**Contributor**
Introduction	Joanne
Background	Nelly
Outcomes	Joanne (EVP), Sin Lu (Analysis)
Methodology	Joanne (EVP), Sin Lu (Analysis)
Summary of Software Deliverable	Nelly
Summary and discussion of software qualities handling	Sin Lu
Sample Source Code	Nelly
Critical Discussion	Joanne
Conclusion	Sin Lu
Appendix	Nelly
References	Joanne, Sin Lu and Nelly

References

Ang, J.S.Y., Lau, S.L., & Tay, N.Y.T. (2021, October). Monash FIT3161 Project Proposal.

Chung, J. S., & Zisserman, A. (2016, November). Lip reading in the wild. In *Asian conference on computer vision* (pp. 87-103). Springer, Cham. https://doi.org/10.1007/978-3-319-54184-6_6

Duan, X., Dai, Q., Wang, Y., & Hua, Z. (2016). Recognizing spontaneous micro-expression from eye region, *Neurocomputing, 217*, 27–36.

Green, B. (2002). Canny edge detection tutorial. Retrieved: March, 6, 2005.

Iwasaki, M., and Noguchi, Y. (2016). Hiding true emotions: micro-expressions in eyes retrospectively concealed by mouth movements. *Scientific reports, 6,* 22049.

Ji, X., Zhou, H., Wang, K., Wu, W., Loy, C. C., Cao, X., & Xu, F. (2021). Audio-driven emotional video portraits. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 14080-14089).

Joseph, G. (2019) FaceFitting [Source code]. https://github.com/Shade5/FaceFitting

Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00)*, Grenoble, France, 46-53. https://doi.org/10.1109/AFGR.2000.840611

Kendra C. (2021, April). The 6 types of basic emotions and their effect on human behavior. https://www.verywellmind.com/an-overview-of-the-types-of-emotions-4163976#citation-11

Li, S., Deng, W., & Du, J. (2017). Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2852-2861).

Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The Extended Cohn-Kanade Dataset (CK+): A complete expression dataset for action unit and emotion-specified expression. *Proceedings of the Third International Workshop on CVPR for Human Communicative Behavior Analysis (CVPR4HB 2010),* San Francisco, USA, 94-101. https://doi.org/10.1109/CVPRW.2010.5543262

Meng, D., Peng, X., Wang, K., & Qiao, Y. (2019, September). Frame attention networks for facial expression recognition in videos. In *2019 IEEE International Conference on Image Processing (ICIP)* (pp. 3866-3870). IEEE. https://doi.org/10.1109/ICIP.2019.8803603

Reddy, S. P. T., Karri, S. T., Dubey, S. R., & Mukherjee, S. (2019, July). Spontaneous facial micro-expression recognition using 3D spatiotemporal convolutional neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE. https://doi.org/10.1109/IJCNN.2019.8852419

Shi, J., Zhu, S., & Liang, Z. (2021). Learning to amend facial expression representation via de-albino and affinity. *arXiv preprint arXiv:2103.10189*.

Song, Y., Zhu, J., Li, D., Wang, X., & Qi, H. (2018). Talking face generation by conditional recurrent adversarial network. arXiv preprint arXiv:1804.04786.

Suwajanakorn, S., Seitz, S. M., & Kemelmacher-Shlizerman, I. (2017). Synthesizing obama: learning lip sync from audio. ACM Transactions on Graphics (ToG), 36(4), 1-13. https://doi.org/10.1145/3072959.3073640

Wang, K., Wu, Q., Song, L., Yang, Z., Wu, W., Qian, C., ... & Loy, C. C. (2020, August). Mead: A large-scale audio-visual dataset for emotional talking-face generation. In European Conference on Computer Vision, 700-717. Springer, Cham. https://doi.org/10.1007/978-3-030-58589-1_42

Wen, Z., Lin, W., Wang, T., & Xu, G. (2021). Distract Your Attention: Multi-head Cross Attention Network for Facial Expression Recognition. *arXiv preprint arXiv:2109.07270*.

Wu, W., Qian, C., Yang, S., Wang, Q., Cai, Y., & Zhou, Q. (2018). Look at boundary: A boundary-aware face alignment algorithm. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2129-2138.