Audio to Robot Facial Landmark Lip Synchronization In The Wild

Yunzhe Wang
Department of Computer Science
Columbia University
yw3737@columbia.edu

Abstract: Given a piece of speech waveform and a static face image, lip synchronization matches the person's lips/facial movement to make it seems like the person is delivering the speech. Accurate and natural lip movement is a crucial component for better human-robot interaction and communication. In this paper, we present a generative pipeline that outputs a sequence of facial landmarks with facial structures matching the given audio. The pipeline was trained purely from human speech visual-audio data. The generated landmark can be further used for downstream tasks such as synthesizing photo-realistic video or set as input to drive a robot face. Several evaluation metrics were used to compare various model architectures. We conclude that our pipeline at the current stage can successfully lip-sync to relatively simple speech audio. We identified several modifications and directions that could be made regarding data processing, pipeline design, and deployment for improved performance.

Keywords: Lip Synchronization, Robot Learning, Representation Learning

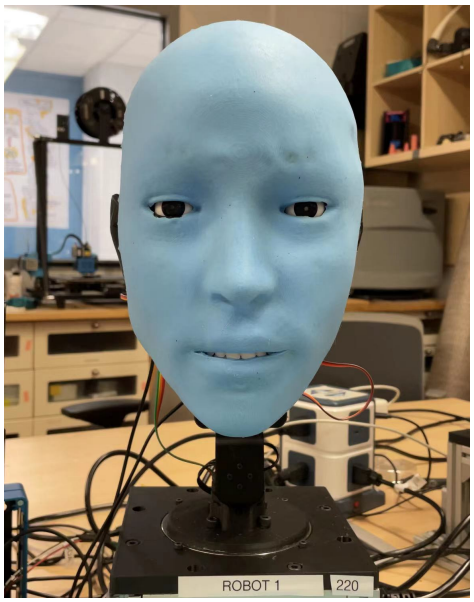


Figure 1: The robot face *Eva* that we are going to lip-sync on

Contents

1 Introduction

1.1 Lip Synchronization

Lip synchronization is the technology that matches a person's lips movement with a given piece of speech audio, the topic can sometimes get generalized to also sync up with related facial expressions. Common application includes creating talking animations of virtual characters, fabricating a speech video of a person (Deep Fake)[1], or creating a humanoid robot face that speaks with realistic lip movement [2]. Old methodologies that create realistic lip synchronization usually involve a large amount of hard-coded lips/facial commands, making them unscalable to new speech data. Whereas existing scalable lip synchronization methods usually oversimplify lips movement preserving only mouse opening and closing.

Fortunately, the technology has become increasingly mature in the computer graphic field where human speech patterns and facial movements could be learned directly from large-scale speech visual-audio datasets with end-to-end neural network models. Prajwal's Wav2Lip model [3] utilizes a GAN architecture with loss from a pre-trained lip-sync expert. Lahiri et al. [4] utilizes a video-based learning framework to animate 3D talking faces from audio. The end-to-end models coupling the speech-related and identity-related model such as the implementation of the recurrent model combines the frame-to-frame and context information well [5]. Based on that, implementations such as the Wav2Lip and Zhou's model [6] decouple the identity and speech information and focus on the capability to generate arbitrary-subject speaking with high fidelity. Nonetheless, much of the techniques focus purely on lips while ignoring the natural control of other facial features.

As for lip-syncing in the humanoid robotic face, due to the lack of dedicated robot datasets or a missing pipeline to transfer lip-sync knowledge from existing human speech audio datasets onto the robot face, much of the current methods still require hard-coded expression. Chen [7] proposed a two-stage framework with a generative network that synthesis robot facial expression images and an inverse network that actuates robot face to express the facial expression in previously generated images. In this paper, we present an audio-to-landmark generator, as an extension/modification to Chen's framework, that eventually matches robot face and lips movement to a given piece of audio through trajectories purely learned from human speech visual-audio dataset.

1.2 Facial Landmarks

In many of the above-mentioned models that synthesize talking face animation or photo-realistic videos, facial landmarks are usually predicted first from audio and then used as an intermediate blueprint that guides downstream photo or animation synthesis with the matching facial expression. A facial landmark is an array of 2-dimensional or 3-dimensional points that marks the position of facial key-points such as eyes, eyebrows, nose, lips, and oval face contour. Off-the-shelf toolkits, such as OpenPose [8, 9] (2D landmarks with 140 key-points) and MediaPipe FaceMesh [10] (3D landmarks with 468 key-points) could be used to extract facial landmarks given subjects' photos or videos.

A complete face landmark contains information including head rotation, head position, and facial contour, whereas the facial contour further contains expression and identity information. For example, a smiling face is different from a sad face, and different people may have different eye distances or lips thickness. Since we are mapping human faces to robot faces 2.6), it would be important if we can decouple different types of landmarks information and keeps only the facial expression. Wang [11] proposed a landmark feature extractor pipeline that learns face appearance, canonical

key-points, head pose, and head motion with 4 separate neural networks, such features significantly decreases bandwidth needed in Nvidia video conference. MediaPipe applied similar concept using regular methods, see section 2.1.3. To make sure we uses the same terminology (see image example in Appendix A), we define the following terms:

Canonical Face: A static face of an identity with no expression. See Figure 7.

Face Translations: Position of the face in the frame

Face Rotation: Rotation of the face

Face Perturbation: Landmark displacements from the canonical face that encodes facial expression

Metric Face: Canonical Face + Face Perturbation. See Figure 8 (right).

Face Transformation: Face Rotation + Face Translation

Screen Face: Metric Face + Face Transformation. See Figure 8 (mid).

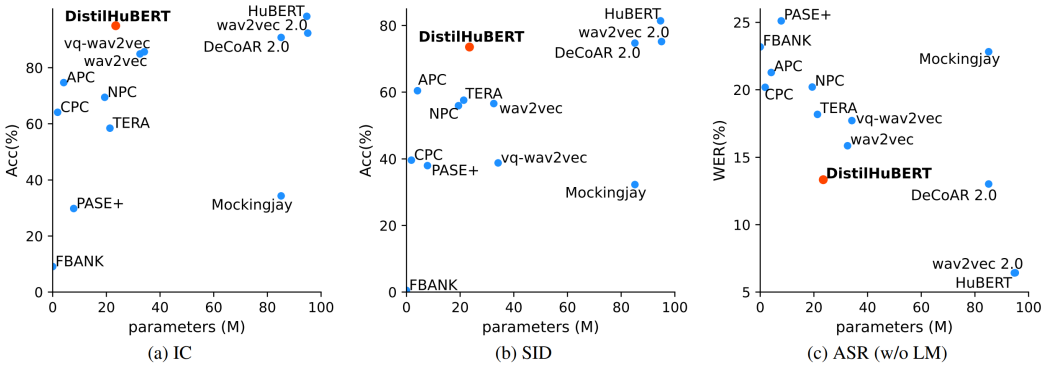
1.3 Self-Supervised Representation Learning

Representation learning or feature learning is a set of approaches used in deep learning to automatically discover from raw data the representations needed for downstream tasks such as classification. It usually replaces manual feature engineering and is shown to achieve better performance in areas such as Natural Language Processing (BERT text embedding [12]) and Computer Vision (Inception [13] and SEER [14] image embedding). In a neural network, a representation of input usually refers to a high dimensional vector that captures a concise meaning of the input thereby representing it. The process of acquiring such representation usually comes from the hidden layer of another pre-trained neural network. With clever design in pre-training tasks, the pre-trained net could acquire a high-level understanding of the input data. Such training is usually self-supervised, meaning that no label to the input is needed.

Common self-supervised pre-training methods include predicting missing parts from the whole. For example, predicting items in the next timestep when providing the previous one, or predicting randomly masked out pieces given a complete sequence. Other common methods include Siamese networks [15], which present two slightly distorted versions of the same object and ask the network to produce the same embedding; and contrastive learning which presents a model with two incompatible objects while asking it to output representation with a high difference.

Apart from images and text embedding, there is also a research direction in creating representations for audio datatype, aiming to achieve the highest downstream performance with the least amount of model parameters (see Figure 2). Related models include Autoregressive Predictive Coding (APC) [16] that predicts the Mel-spectrogram of the next piece of audio given the previous piece in an auto-regressive manner with Gated Recurrent Unit; Transformer Encoder Representations from Alteration (TERA) [17] that uses transformer encoders [18] to reconstruct acoustic frames by altering along the time, frequency, and magnitude dimension; and Distillation of Hidden-Unit BERT (DistilHuBERT) [19] which not only uses a CNN and a transformer encoder to classify randomly masked frames to pseudo labels, the same method as previous model HuBERT [20], but also employs knowledge distillation techniques [21] to significantly reduces network size while preserving its accuracy. Considering both downstream performances and model size, we picked the APC model for speech representation in our work.

Furthermore, face representation that emphasizes the ability to distinguish between different identities is commonly used nowadays for identity verification in face recognition systems [22]. Similarly,



(a) IC (b) SID (c) ASR (w/o LM)

Figure 2: Comparisons of pre-trained speech representation model size vs. downstream performance on three tasks: Intent Classification (IC↑), Speaker Identification (SID↑), and automatic speech recognition (ASR↓). Figure taken from Fig.2 in the DistilHuBERT paper [19]

Vemulapalli [23] showcased that face embedding, when trained on a large-scale annotated dataset targeting facial expression regardless of identity, could nail at applications such as expression retrieval, photo album summarization, and emotion recognition. Wiles and Koepke [24] proposed facial attributes representation, acquiring impressive performance in downstream tasks such as head pose regression, expression classification, and facial landmarks regression. The embedding model FAb-Net was trained in a self-supervised manner by merely presenting the model unlabeled videos, given the task to predict future frames. Nonetheless, most facial representation models use face images as input rather than landmarks. It would be interesting to ask if we can train embeddings models using landmark data to assist the transition from human face to robot face, see section 2.6 and 4.5.

1.4 Goals and Motivation

Our goal for this project is to design and implement an algorithm that takes the sequence of the audio as the input and outputs the corresponding sequence of facial landmarks with natural lip synchronization and facial expression. The generated landmarks will then be mapped from the human space to the robot space to eventually drive the robot face using an inverse model. This will be one of the modules of the entire software composition of the robot face, which enables it to speak naturally like a human being. Given that most face robot uses hard-coded facial expression with trivial lip synchronization, we are motivated to find deep learning solutions that teaches the robot speech-related facial movements from human videos in the wild.

2 Method/Technical Approach

2.1 Landmarks Acquisition from VoxCeleb2 and Preprocessing

To build a generator model that learns speech-related facial patterns from humans, a large-scale audio-visual dataset is needed. We targeted the VoxCeleb2 dataset [25] due to its abundance and the fact that most previous successful lip-sync and facial deep-fake models were built upon this dataset. Each data instance is a video clip in mp4 format of a celebrity speaking with an accompanying audio file in m4a, where the face landmarks could be extracted using MediaPipe FaceMesh [10], an open-sourced software that detects 3D face-mesh landmarks. After acquiring the landmarks we need, we

cleaned the data by removing videos with low bit-rate, instances where the speaker is not looking at the camera, or with too much head movement.

2.1.1 VoxCeleb2 Visual-Audio Dataset

The VoxCeleb2 dataset [25] is a speech visual-audio dataset containing over 1 million utterances for 6,112 celebrities, extracted from YouTube videos. Each utterance is a video clip of a celebrity speaking, usually in an interview or public speech. Each video has a fixed resolution of 224x224, a fixed frame rate of 25 fps, and an average duration of 7.95 seconds. The dataset is fairly gender-balanced, with 61% of the speakers male. The speakers span a wide range of different ethnicities, accents, professions, and ages. Videos quality ranges from professionally shot multimedia to crude hand-held devices, and speakers showcase various head poses and movements. Face detection and tracking techniques are used so that the face always appears at the frame center. To make sure the landmark we acquire has high quality, we used the top 40K videos with the highest bit-rates for further processing.

2.1.2 MediaPipe FaceMesh

MediaPipe is an off-the-shelve tool that offers machine learning solutions to various detection tasks in streams. One of it's sub-module, MediaPipe FaceMesh [10], is extremely suited for our 3D face landmark detection task on VoxCeleb2 videos. In comparison with OpenPose [8, 9], the tool we used previously, we found out that MediaPipe processes significantly faster and simpler such that inferences can reach +30 FPS on a single CPU whereas OpenPose stuck at around 3-5 FPS. By parallel computing on all 32 processors of AMD Ryzeb 9 5950X CPU, we acquired all needed landmarks data in about a day. The same amount of work is no way near possible for OpenPose.

We believe there are two main reasons that MediaPipe runs faster. First, MediaPipe FaceMesh is a dedicated model just for facial landmarks detection whereas OpenPose is designed more suited for holistic body landmark detection including face, body, and hand. Since face detection is a sub-task for holistic detection, a dedicated tool is expected to work faster. Second, when the input data is a video, OpenPose would treat individual frames as standalone images and process them independently, while MediaPipe FaceMesh enables face-tracking across those frames to reduce inference time and improve accuracy. Another major difference between MediaPipe and OpenPose is how they deal with low confidence data. Given a partially covered or poorly recognized face image, OpenPose outputs all detected landmark points with related confidence scores, so that unrecognizable parts have near-zero confidence. Whereas, MediaPipe hallucinates missing points based on the current data distribution without a confidence score. In our previous model with OpenPose landmarks (Appendix B), we add the confidence score as a part of the input, but it was removed in the MediaPipe model version. The effect of this change needs further investigation. One limitation we found about MediaPipe is that the detected face contour isn't as stable as that of OpenPose from frame to frame with shakiness even if the subject is not moving at all. To counter this shakiness artifact, we smoothed the landmarks by calculating a moving average with a fixed window along the time dimension, but this also reduces the maximum range of landmarks' movement especially the opened-mouth area (see section 4.2 for a possible solution).

2.1.3 Metric Face Alignment

Since we need to map facial landmarks from human distribution to robot distribution without head movements, we need to align the generated landmark such that it is constantly facing forward without head movement. We previously used Affine transformation, which applied a shear movement on

2D landmarks so that the left eye, right eye, and low jaw appear in the same location for all faces. However, the method couldn't handle well on side faces. Instead, we used the face transform module and face geometry module provided by MediaPipe. The method linearly maps the screen landmark at each frame onto a fixed canonical face assuming that the input video frames are observed by the virtual perspective camera. The resulting landmark is named the metric face. An image example of the screen face and metric face can be seen in Appendix A, figure 8.

2.1.4 Landmark Data Cleaning Through Centerness and Shakiness

To ensure high-quality landmarks, we picked the top 40K videos with the highest bitrate for further processing in MediaPipe. Some videos may have frames that contain multiple or no faces. We remove all such videos in preprocessing. Since we don't want side faces that contain little to no lips information or faces with too much head movement such that the detected landmarks are heavily distorted, we came up with two metrics scores: centerness and shakiness. Given the metric landmarks and its related pose-transform matrix, we can first extract the pose-rotation vector from the pose-transform matrix and turn the rotation vector into a rotation matrix using Rodrigues' rotation formula [26] `cv2.Rodrigues`, and then compute the Euler angle of the frontal vector (tip of the nose pointing outwards) from the rotation matrix by using QR-decomposition [27] `cv2.RQDecomp3x3`. Based on the value p (position) of this front angle and its change-of-value (velocity), we can define centerness c and shakiness s score of a video as:

$$c = \frac{1}{T}\sum_{t=0}^{T} p_t \tag{1}$$

$$s = \frac{1}{T}\sum_{t=1}^{T} p_t - p_{t-1} \tag{2}$$

where t indicates frame index and T is the total number of frames. A video with high centerness tends to contain mostly side faces, and videos with high shakiness tend to have large head movements during speech. Figure 3 shows the distribution of centerness and shakiness score of 17K videos. We want to keep videos with low centerness and low shakiness. For convenience, we manually set the centerness threshold as 30 and the shakiness threshold as 2. Videos with c and s values below the threshold are kept. After all landmark data cleaning processes, we are left with 265K VoxCeleb2 videos with relatively high bitrate, a sole face detected throughout the video, more centered facing, and fewer head movements.

2.2 Audio Slice and Mel-Spectrogram Processing

We directly download the original audio file from the VoxCeleb2 Website. Each file is written in m4a format extracted from the corresponding video. The audio dataset has the same filename and directory structure as that of the video dataset.

To obtain the audio training data, we extracted the Mel-Spectrogram of each of the m4a files that has corresponding landmark data in the processed matrix using the `feature.melspectrogram` function in Librosa library. The parameter `n_mel` – the number of Mel bands – controls the height of the matrix and the parameter `hop_length` – the number of samples between successive frames – controls the length of the matrix. The relationship between the hop length h, sample rate f_s, and frame per second fps is that $f_s/h =$ fps. So, to make sure the width of the output Mel-spectrogram

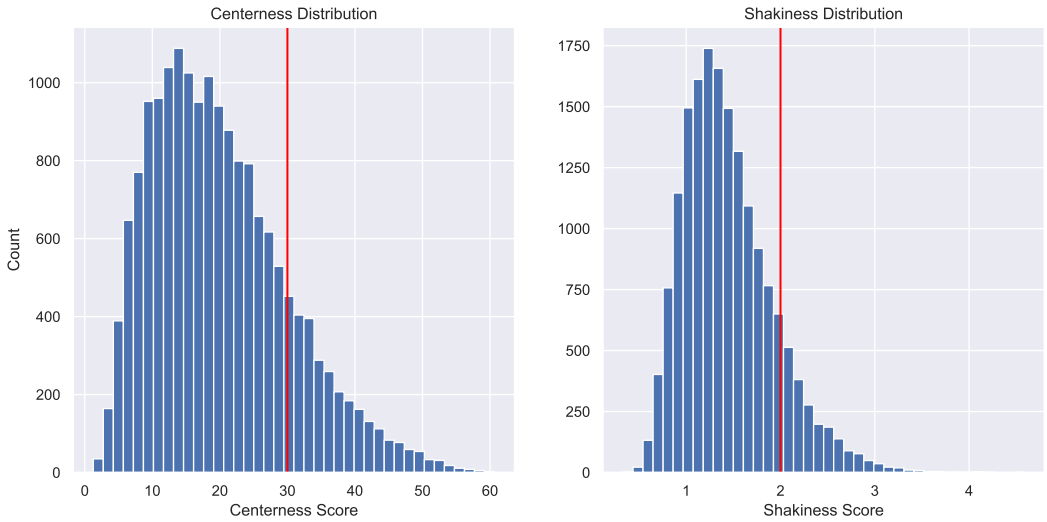


Figure 3: Centerness and shakiness score distribution of 17K videos. The red vertical line indicates the chosen threshold for each metric. Videos with centerness and shakiness value below the threshold are kept.

matches with the number of video frames, we can set the hop length h as that expressed in equation 3 where t is time in seconds.

$$h = \frac{\text{fps}}{f_s} = \frac{\text{fps} \cdot t}{f_s \cdot t} = \frac{\text{waveform length}}{\text{expected number of frames}} \tag{3}$$

To be specific, we read each audio file with a sample rate of 16000, and compute Mel-spectrogram with 80 Mel bands, hop length $16000/50 = 320$, window length $16000/25 = 640$. This gives us a spectrogram matrix with dimensions exactly $2n \times 80$, where n is the number of frames to the matching video. We also set a low-pass frequency filter at 90, and a high-pass filter at 7600 since that frequency range contains most human speech information. After acquiring the Mel-spectrogram, we convert it into log-scale using the `power_to_db` function in Librosa. The first 0.04 seconds of audio information are repeated to expand the matrix and the whole matrix is then sliced into overlapping pieces with a length of 13 frames (roughly 0.5 seconds) along the time axis. The number of audio slices matches exactly the number of frames of the video. The maximum input sequence length is capped at 500 frames (roughly 20 seconds). Each slice is then fed into the APC model.

2.3 APC Speech Representation Model Pre-Training

After acquiring the Mel-spectrogram from each audio slice, one would then want to acquire acoustic features from the raw audio that encompasses temporal dependencies. A common approach is to directly apply 2d convolution to the Mel-spectrogram. Nonetheless, more recent evidence indicates that features learned automatically as speech representation using a self-supervised paradigm could often achieve higher performance in various downstream tasks [16, 17, 19]. Considering both accuracy and model size, since we will eventually deploy the model onto Jetson Board possibly without power GPU and CUDA memory, we picked Autoregressive Predictive Coding (APC) [16] as our speech representation model. The GRU-based APC model contains only 4M parameters achieving comparable performance while other transformer-based alternatives take 20-100M parameters.

Most of the speech representation methods above provide pre-trained models using the LibriSpeech dataset [28], which is a corpus of approximately 1000 hours of 16kHz read English speech derived from audiobooks reading. However, the dataset is relatively small, composed of few speakers, and limited to just English. With similar work, [29] pre-trained APC model using the Chinese sub-dataset of the Common Voice Dataset [30], which contains crowd-sourced audios from the general public. We found that the pre-trained APC trained by [29] evaluates better than the original pre-trained APC. We believe this is because the crowd-sourced Common Voice contains more speaker identities with recordings of various qualities, such that data distribution matches closer to the audios in VoxCeleb2 and thereby able to generalize better even with a different language. This encourages us to pre-train our own APC model.

The APC model contains an upstream part that creates speech embedding given audio log-Mel-spectrogram, and a post-net that predicts the next piece of log-Mel-spectrogram with a time gap specified by time-shift. The post-net are removed after pre-training, and we can directly use the speech embedding generated by the upstream model. To pre-train our own APC model, we directly used 500K audio instances in VoxCeleb2, trained with time-shift 3 and embedding dimension 512 for 170 epochs. Our APC gets the highest downstream evaluation score in our lip-sync task.

2.4 Audio2Landmark Pipeline

2.4.1 Model Architecture

Given a piece of speech wave and a frame of the static face, the Audio2Landmark pipeline animates the static face by predicting the lip-synced landmark displacement (architecture shown in figure 4). The pipeline first transforms the audio wave into a log-Mel-spectrogram and unfolds it into overlapping slices with a length roughly equal to 0.5 seconds (section 2.2), it then feeds each slice to the APC model to get a speech representation of the same length in time (section 2.3). After that, the speech representation is compressed along the time dimension in two ways: 1. feed through three LSTM layers [31] and take the output at the last position as the compressed embedding; 2. through direct pooling with max, mean, or sum operation. After obtaining the compressed speech embedding, we proposed two different architectures to predict the landmark displacement.

The first architecture directly concatenates the static face with the compressed speech embedding and predicts the displacement with 3 MLP hidden layers with ReLU activation followed by batch normalization. Note that the first architecture predicts each landmark purely based on the respective half-second audio information. The first architecture contains 6.3M parameters in total (APC 4M, LSTM 1.8M, MLP 0.5M).

The second architecture encompasses information over the entire audio dimension through an additional transformer encoder layer (multi-head attention with 16 heads) [18] before concatenating with the static face and going through a 2-hidden-layers MLP. Since predicting a landmark is essentially doing multi-dimensional regression for all landmark points along all axis, we use linear activation (no activation) in the MLP output layer. We also added absolute position embedding to the compact speech embedding for timing information in hoping to learn eye blink, which correlated with time. The second architecture contains 7.5M parameter (APC 4M, LSTM 1.8M, Transformer Encoder 1.3M, MLP 0.4M) when using LSTM compressing, and 7.7M parameters (APC 4M, Transformer Encoder 3.2M, MLP 0.5M) when using pooling.

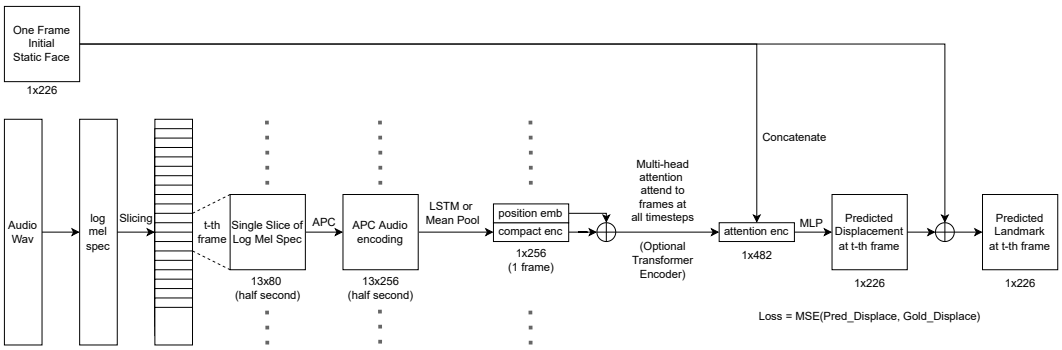


Figure 4: Audio2Landmark model architecture of version with the multi-headed attention (best performance)

2.4.2 Neural Network Training

To train the Audio2Landmark model, we first split the development set into train and validation sets in a 0.8:0.2 ratio. Given an instance of training data (sequence of paired audio slices and facial landmark), we first compute the ground-truth landmark displacement with regards to the first frame and then input the audio sequence and the first landmark frame to get the full landmark displacement sequence. We minimize the mean squared error of the predicted displacement and the ground-truth displacement and update model parameters with the Adam Optimizer [32] using a learning rate of 0.0001 and betas (0.9, 0.999) without weight decay. Since input sequences have varying lengths, the loss value for each sequence was divided by its respective length. The model was trained on an RTX3090 GPU for 30 epochs using batch size 64. Model parameters are clipped with max norm 1.

2.5 Robot Face Landmark Acquisition

We also did some experiments regarding acquiring robot face landmarks using MediaPipe FaceMesh. See robot face appearance in figure 1. We noticed that the landmarks extracted with RGB videos may sometimes contain frames of unrecognizable faces. By turning RGB images to grayscale, the number of missing faces decreases. By enhancing the grayscale image `cv2.equalizeHist`, the number of missing faces further decreases. Also, the robot landmarks detected by MediaPipe are relatively unstable (more shaky movement) compared with human landmarks. The inaccuracy and instability could largely be due to the difference between the appearance of the robot face and an actual human face. Future work may focus on the appearance design of the robot face e.g. adding makeup. Also, we believe that there exists an image pre-processing method (by changing image color, brightness, etc) to enhance the accuracy and stability of robot face landmark extraction. Finding the best pre-processing technique by itself is an optimization problem and needs further investigation.

2.6 Human to Robot Face Mapping

Since there could be a big difference between the robot landmark distribution and human landmark distribution, a key problem is how can we align landmarks from the human space to the robot face. One method to evaluate the facial expression separability of the data is by visually observing clusters after dimensionality reduction. Figure 5 shows the UMAP [33] embedding of the metric landmarks taken from facial images in the DISFA+ Dataset [34]. We randomly selected 15 images from each of the 24 categories combination (4 subjects each performing 6 different facial expressions). We

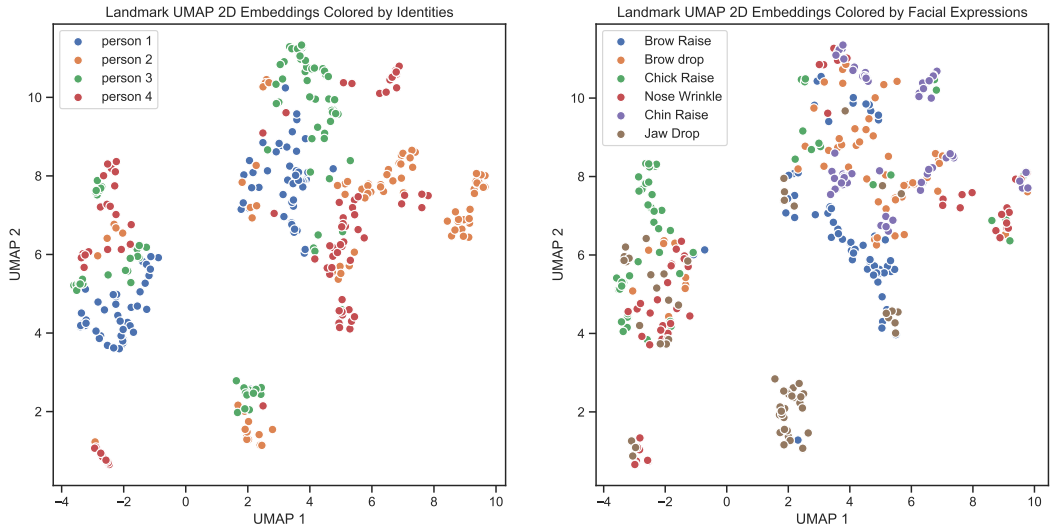


Figure 5: 2D UMAP embedding of the metric landmark extracted from randomly shuffle images of 4 subjects doing 6 different facial expressions (Brow Raise, Brow drop, Chick Raise, Noise Wrinkle, Chin Raise, and Jaw Drop) with 15 images from each category. A facial expression is triggered if the respective action unit intensity is above 3 (range 0-5).

discovered that the metric landmark extracted from MediaPipe already has a certain potential to differentiate facial expressions across different identities. For example, the small cluster at the bottom-mid location shows that images of the same Jaw-Dropping expression (brown) are clustered together even though it is performed by two different people (green and orange).

Nonetheless, the overall embedding does not show strong data separability with regard to facial expressions. Given that the robot should show the same facial expression matching a given landmark regardless of who performed the expression, we proposed a facial landmark attribute representation model to disentangle the identity information, see Section 4.5. We hypothesized that given a robust facial representation model M and metric landmarks l, the facial embedding $M(l)$ should demonstrate strong clustering separation across different identities in UMAP projection.

3 Experimental Results

3.1 Evaluation Metrics

Similar to the evaluation metrics as used by [6], we used 6 different metrics to evaluate the fitting of the generated landmarks and the ground-truth landmarks regarding the lips and jaw movements, eye blinks, and holistic facial expression.

Difference in Position of all landmarks (D-P): Computes the average euclidean distance between the predicted landmark positions and the ground-truth landmark positions.

Difference in Velocity of all landmarks (D-V): Computes the average euclidean distance between the predicted landmark velocity and the ground-truth landmark velocity, where the velocity is calculated as the first-order difference of landmark positions along the time dimension.

Difference in Position of Lips-and-Jaw landmarks (D-PL): Computes the average euclidean distance between the predicted landmark positions and the ground-truth landmark positions for just the lips and jaw regions.

	D-P ↓	D-V ↓	D-PL ↓	D-VL ↓	D-AL ↓	D-AE ↓
LSTM W/O Attention	29.642	6.506	17.462	4.374	0.781	0.389
LSTM W/ Attention	29.098	**5.858**	**17.030**	**3.871**	**0.714**	**0.380**
Mean Pool W/ Attention	**29.097**	5.884	17.095	3.893	0.742	0.381

Table 1: Model evaluation of different architectures regarding with/without multi-headed attention and pooling methods for getting compact speech representation.

Difference in Velocity of Lips-and-Jaw landmarks (D-VL): Computes the average euclidean distance between the predicted landmark velocity and the ground-truth landmark velocity for just the lips and haw regions.

Difference in Area of opened-inner-Lips landmarks (D-AL): Computes the absolute value difference between the predicted inner-lips area and the ground-truth inner-lips area. The inner-lips area is calculated through the Shoelace area-of-polygon formula.

Difference in Area of opened-Eyes landmarks (D-AE): Computes the absolute value difference between the predicted opened-eyes area and the ground-truth opened-eyes area. The opened-eyes area is calculated through the Shoelace area-of-polygon formula.

3.2 Model Performance

After training models of different versions, we evaluated their performance on the test set of the VoxCeleb2 dataset using the six metrics shown in table 1. We observe that models with multi-headed attention work strictly better than the ones without, and the model that gets compact speech representation using LSTM works slightly better than the one using mean pooling.

Besides number in the 6 metrics, we also visually evaluated the quality of predicted facial landmarks using the test set of VoxCeleb2, an emotion speech dataset called RAVDESS [35], and text-to-voice synthesized speech from online resources. We noticed that our model could do a decent job in lip-syncing with the voice for all three data sources when the input speech is simple and slow.

However, for fast speech or sentences with long words (usually speaks fast), our model produces a much higher error. We believe this is because our model does not have enough temporal resolution (see the proposed solution in section 4.1). Also, for text-to-speech synthesized voice, the opened-mouth area is relatively small (see the proposed solution in section 4.2). Another limitation is that the eye-blink movement is not fitted since there is no direct correlation between speech and eye-blink (see the proposed solution in section 4.3).

4 Discussion and Future Work

Given that 1. our current model could not predict well on fast speech; 2. sometimes the input audio produces an overly small opened-mouth area; 3. our model failed to capture eye-blink at all; 4. the model will eventually be deployed onto a physical robot and compute on an embedded computing board; 5. the robot face will interact with human beings primarily through conversation, we identified several improvement methods and additional network modules to better guide future direction.

4.1 Finer Temporal Resolution for Fast Speech Audio Handling

Through visual evaluation of generated facial landmarks, we observed an issue: when the input audio has a fast speaking speed or contains long words (usually speaks fast), the generated landmarks failed to match with the speech with a high chance. We believe this is because our model does not possess high enough temporal resolution while processing the audio information. In our current model design, we set the audio-landmark sequence length the same as the number of frames in a VoxCeleb2 video, where all video frame rates are 25fps. Increasing the audio resolution, by adjusting the window length and hop length setting when computing Mel-spectrogram, allow us to get a larger matrix for the same audio length that for sure encodes more audio information at a unit time. However, this buff in temporal resolution increases the number of audio slices with creates a mismatch to the face landmarks that is fixed at 25fps as defined by the dataset. One could then use interpolation methods to further increase the frame rate.

Another possible explanation is related to the window length setting when cutting audio into slices. Our current audio slices correspond to an audio length of around 0.5 seconds, whereas the duration of a phonetic syllable could be much shorter than 0.5 seconds especially those seen in long words. The optimal slice length hyperparameter needs further tuning. Also, since lip shape should be most correlated with the lip shape phonetic syllable, one may attempt to slice the audio based on those syllables with unequal length or add a sequence of syllable embedding into the input. Kamper [36] proposed a word/syllables segmentation method using dynamic programming and self-supervised scoring, which could be an extension to our current model.

4.2 Opened-Mouth Area

We noticed that the generated face landmarks are sometimes very small, especially when using text synthesized voice. Even if the lips movement matches with the audio input, if the opened-mouth area is overly small, the robot's speaking ability appears less convincing. Nonetheless, certain audio pre-processing hacks such as an increase in volume or pitch would lead to a bigger opened-mouth area in generated landmarks, we believe this is because when you speak with high volume or high pitch, you are expected to open your mouth bigger. Also, since we are predicting landmark displacement rather than position, we can multiple each lips displacement with a scaling factor to control the opened-mouth area. The best audio processing tricks or optimal scaling factors need further experiment.

4.3 Eye-Blink Fitting

Our current model failed to fit eye-blink at all. In fact, learning eye movement purely from human data is something that most audio-driven facial landmark generators would fail at, apparently because there is little to no correlation between speech and eye movement, and different speakers could have different eye-blink patterns. Previous models that showcase eye-blinking usually employ direct sampling from data [29], or the speech model was trained with a single identity [6] so that the eye-blink behavior is more predictable.

Since humans blink eyes once every five seconds on average, it is possible to learn eye-blink behavior based on timing alone. We previously thought the added position embedding in the Transformer encoder would help the eye movement fitting, but it did not. Future experiments on fitting eye movement could attempt using a separate module with time information without audio. One may also try adding timing information in other forms such as noting the time gap between two consecutive eye blinks, adding features like opened-eyes area, discretizing eye-blink to binary commands, or adding the landmark generated at previous k timesteps to input.

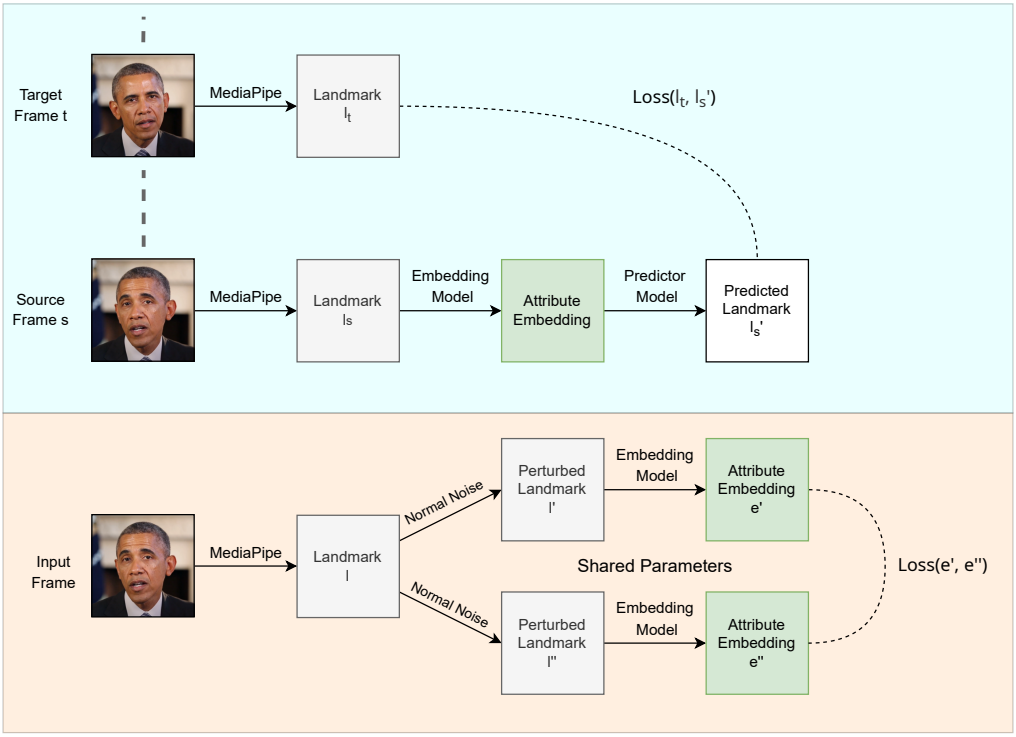


Figure 6: Proposed training paradigm for self-supervised facial landmark attribute representation. Top: Masking method that predicts landmarks at a future frame. Bottom: Siamese architecture that learns the same representation for two perturbed versions of the same landmark.

4.4 Deployment on Jetson Board with Small Latency and Model Size

Since our audio-to-landmark generator model will eventually run on an embedded computing board inside a physical robot, latency and model size are two crucial considerations. In particular, we will be using Jetson Xavier NX, which consists of a modern CPU and GPU with 8GB of shared memory. Given an audio Mel-spectrogram, our current model could infer the synchronized landmarks instantaneously on a modern computer. However, the timing is not yet accessed on the Jetson board. Furthermore, much of the time will also be spent on audio file loading and the conversion from audio to Mel-Spectrogram. Based on benchmark statistics regarding the loading speed of various python libraries on various audio file formats, the fastest loading audio file format is wav (less than 0.01 sec) while Librosa is the fastest library in converting audio wave to Numpy array. Regarding deployment, one may consider first converting the PyTorch model to NVIDIA TensorRT through ONNX for optimized inference performance.

4.5 Facial Landmark Attribute Representation

Mainly inspired by FAb-Net [24] and motivated to solve the human-robot face mapping problem (section 2.6), we proposed a self-supervised learning approach to acquire face attribute representation from landmarks. The representation should decouple identity information so that, when applying to the inverse model that drives robot faces from landmarks, errors coming from the identity noise could be maximally reduced. Figure 6 shows two proposed training paradigm.

References

[1] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2019.

[2] K.-G. Oh, C.-Y. Jung, Y.-G. Lee, and S.-J. Kim. Real-time lip synchronization between text-to-speech (tts) system and robot mouth. In *19th International symposium in robot and human interactive communication*, pages 620–625. IEEE, 2010.

[3] K. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 484–492, 2020.

[4] A. Lahiri, V. Kwatra, C. Frueh, J. Lewis, and C. Bregler. Lipsync3d: Data-efficient learning of personalized 3d talking faces from video using pose and lighting normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2755–2764, 2021.

[5] Y. Song, J. Zhu, D. Li, X. Wang, and H. Qi. Talking face generation by conditional recurrent adversarial network. *arXiv preprint arXic:1804.04786*, 2019.

[6] Y. Zhou, X. Han, E. Shechtman, J. Echevarria, E. Kalogerakis, and D. Li. Makeittalk: Speaker-aware talking-head animation. *arXiv preprint arXic:2004.12992*, 2020.

[7] B. Chen, Y. Hu, L. Li, S. Cummings, and H. Lipson. Smile like you mean it: Driving animatronic robotic face with learned models. *arXiv preprint arXiv:2105.12724*, 2021.

[8] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[9] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[10] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. *arXiv preprint arXiv:1907.06724*, 2019.

[11] T.-C. Wang, A. Mallya, and M.-Y. Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10039–10049, 2021.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[14] P. Goyal, M. Caron, B. Lefaudeux, M. Xu, P. Wang, V. Pai, M. Singh, V. Liptchinsky, I. Misra, A. Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[15] S. Becker and G. E. Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992.

[16] Y.-A. Chung and J. Glass. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501. IEEE, 2020.

[17] A. T. Liu, S.-W. Li, and H.-y. Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366, 2021.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] H.-J. Chang, S.-w. Yang, and H.-y. Lee. Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7087–7091. IEEE, 2022.

[20] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[21] G. Hinton, O. Vinyals, J. Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[22] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. *Advances in neural information processing systems*, 27, 2014.

[23] R. Vemulapalli and A. Agarwala. A compact embedding for facial expression similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5683–5692, 2019.

[24] O. Wiles, A. Koepke, and A. Zisserman. Self-supervised learning of a facial attribute embedding from video. *arXiv preprint arXiv:1808.06882*, 2018.

[25] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[26] R. W. Brockett. Robotic manipulators and the product of exponentials formula. In *Mathematical theory of networks and systems*, pages 120–129. Springer, 1984.

[27] G. G. Slabaugh. Computing euler angles from a rotation matrix. *Retrieved on August*, 6(2000): 39–63, 1999.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[29] Y. Lu, J. Chai, and X. Cao. Live speech portraits: real-time photorealistic talking-head animation. *ACM Transactions on Graphics (TOG)*, 40(6):1–17, 2021.

[30] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

[31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[32] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] L. McInnes, J. Healy, and J. Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[34] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh, and J. F. Cohn. Disfa: A spontaneous facial action intensity database. *IEEE Transactions on Affective Computing*, 4(2):151–160, 2013.

[35] S. R. Livingstone and F. A. Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[36] H. Kamper. Word segmentation on discovered phone units with dynamic programming and self-supervised scoring. *arXiv preprint arXiv:2202.11929*, 2022.

Appendix

A Landmark Figures

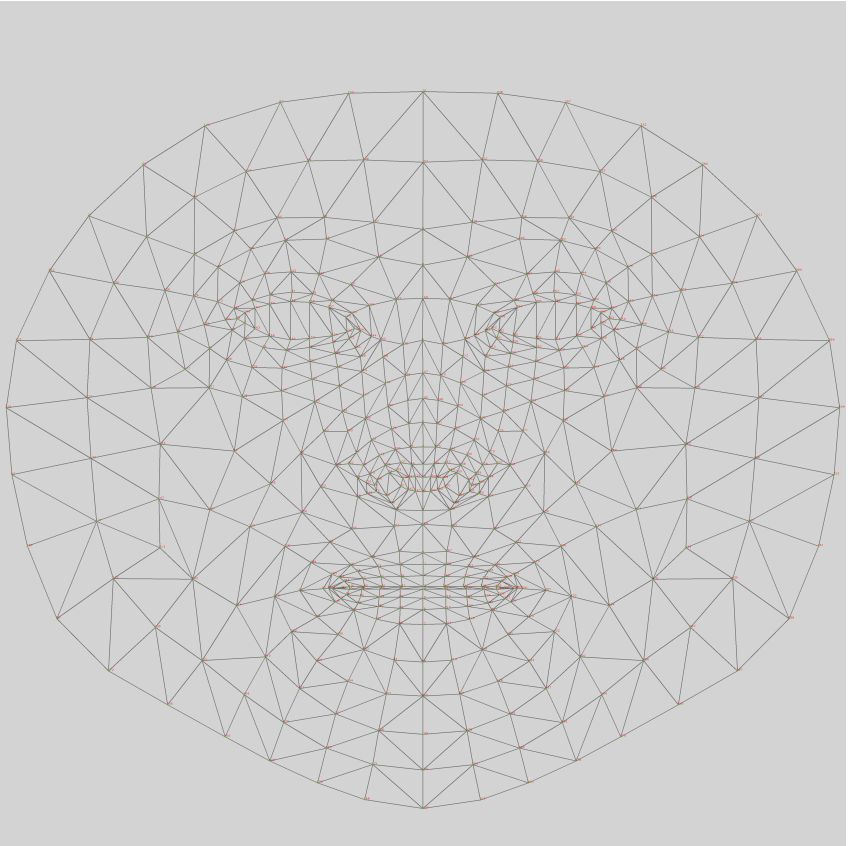


Figure 7: The static canonical face used by MediaPipe. The number (red) on each node shows the landmark indices. A Screen face is normalized onto the canonical face to get a metric face.

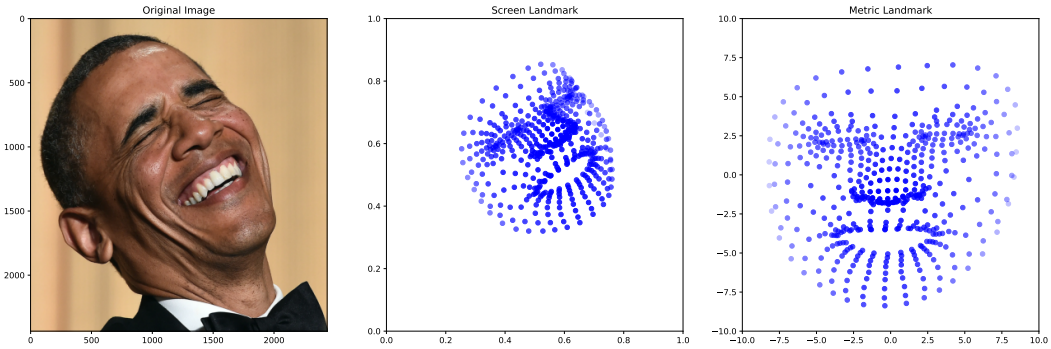


Figure 8: Example of a screen landmark (mid) and a metric landmark (right) extracted from a given face image (left). The screen landmark has facial key-points positions that overlap exactly with that in the original image. The metric landmark is the screen landmark normalized onto the canonical face in figure 7. The metric landmark removes rotation and translation information with a face always looking forward, showing just the facial expression.

B Previous Model Trained with OpenPose Landmarks

Prior to our current model trained with MediaPipe landmarks, we have another working audio-to-landmark generator model that uses OpenPose landmarks (architecture shown in figure 9). However, due to OpenPose's limitation in processing speech, we weren't able to acquire enough training data so the trained model does poorly on test data and overfits fairly quickly. We ran multiple instances on Google Colab Pro GPUs for a week and were only able to process 18K videos. For comparison, we processed 400K videos using MediaPipe on a single computer.

Model details are shown here for documentation purposes. Similar to the current model version, we used APC speech representation for the audio feature, Recurrent Network (GRU) to acquire long-term dependencies across the time dimension, and predict landmark displacement. However, there are also some major differences. First, we did not slice the audio into overlapping slices. Instead, the model predicts the landmarks at each frame based on audio information of the same length (0.04 seconds) and all audio information at previous timesteps preserved through recurrent units. This gets us low-quality landmarks for the first few frames. Second, since OpenPose provides a confidence score for each landmark point, we add the confidence as part of the input such that the learned landmark distribution is conditioned on the confidence. To generate high-quality landmarks after training, we set a fixed high confidence input score. Third, when predicting displacement at any given timestep we add the displaced landmark at the previous timestep as part of the input, which adds a short memory on the previously generated landmark as it also goes into the recurrent unit. However, this self-feeding recurrent network is unfriendly to batch processing and forces a for-loop in implementation, which significantly slows down the training efficiency. Therefore, this design is removed in the later version. Fourth, we used l1 loss instead of MSE because we empirically found that l1 loss works better at that version.

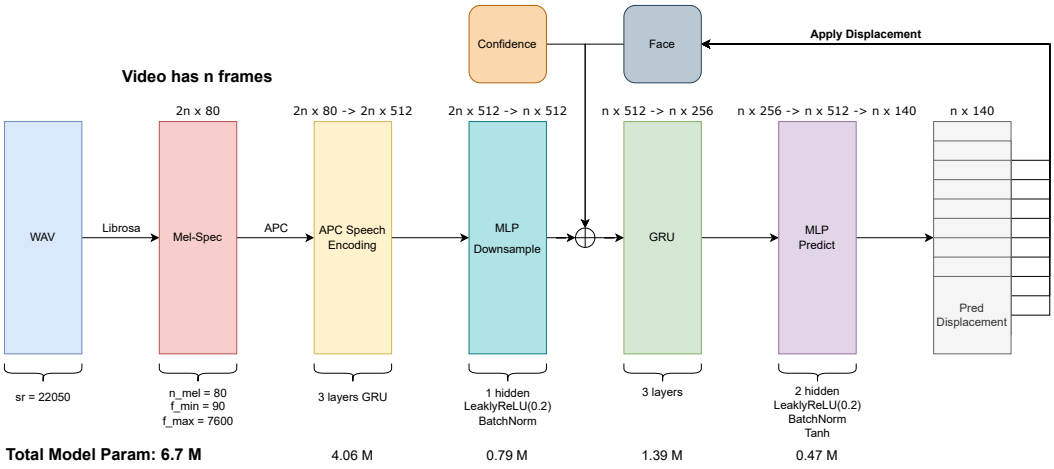


Figure 9: Previous Audio2Landmark model architecture build upon OpenPose landmarks. Audio is not cut into overlapping slices. The model employs a self-feeding recurrent network that preserves time dependency.

C Optional methods

Besides the prior model that uses OpenPose landmarks as discussed in Appendix B, there are two more methods related to data standardization and loss function design that may or may not improve the Results. They are marked here as 'Optional' for documentation purposes.

Landmark Parts Weight Loss: Even though we are fitting an audio-to-landmark generator that predicts holistic facial expression, the major prediction should still mainly focus on lips and jaw, which have the highest correlation with input audio. To encourage such behavior, we came up with a weight loss function that adds coefficients to different facial parts as shown in equation 4. One could set a higher coefficient to lips to encourage lip fitting. However, since this adds extra hyper-parameters, we did not include it in the latest version.

$$Loss = aL_{lips} + bL_{eyes} + cL_{nose} + dL_{eyebrow} + eL_{oval} \tag{4}$$

Displacement Standardization: A data distribution centered near 0 with roughly equal variance across different dimensions usually makes training more consistent in machine learning. We used `StandardScaler` in Scikit-Learn to standardize ground-truth landmark displacement. Since this creates shifting and scaling in data distribution, we applied an inverse transform on generated displacement to map it back to the original distribution. However, since the effect of this standardization process is not obvious, it is marked as optional here.

ANIMATORNIC ROBOT FACE SIMEXPRESSION AND LIPS SYNCHRONIZATION
2022 SUMMER RESEARCH REPORT

Yingke Wang, Yunzhe Wang, Jiong Lin
School of Engineering and Applied Science
Columbia University
New York, NY 10027, USA
{yw3821,yw3737,jl6017}@columbia.edu

ABSTRACT

Self-modeling enables robot to create a future self-image before making actions. In this project, utilizing the concept of self-modeling, we demonstrate two functionalities of our animatronic face robot: lips synchronization and sim-expression. The former takes a piece of speech audio and animates facial keypoints as if the robot is delievering the speech. The later mimics human facial expression with the ability to foresee expression changes thereby realizing simultaneous expression mimicing (simexpression). Both pipelines creates an 'imagined' version of the face robot represented in facial landmarks that are then actuated through an inverse model.

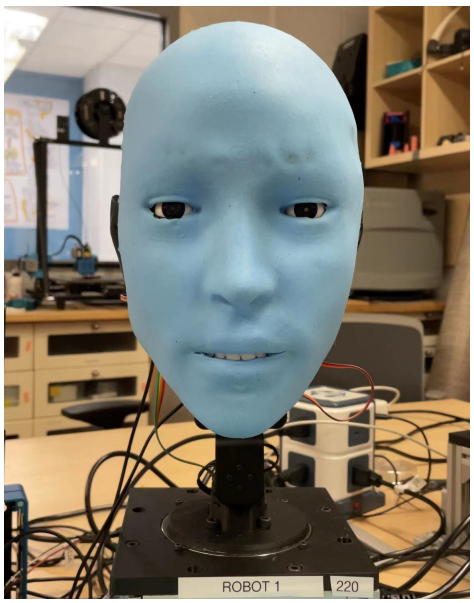


Figure 1: Animatronic Face Robot *Emo1.0* with simexpression and lips synchronization ability

CONTENTS

1 INTRODUCTION

1.1 ROBOT FACE SIMEXPRESSION

ScientistsHara & Kobayashi (1996)Kobayashi & Hara (1997) from University of Tokyo and University of Zurich put up with a new face robot. Their aim is to make the robot recognize the human facial expression and produce its own facial movements. Their are using 3-layers fully connected neural network to learn a 6-classes classification task. Takuya Hashimoto1 and Sachio Hitramatsu1Hashimoto et al. (2006) also put forward a face robot called SAYA. They are using the pneumatic actuators to simulate human muscle movements. Karsten BernsBerns & Hirth (2006) introduced a realistic humanoid robot head "ROMAN". The face and neck mechanical design were given in the their paper.

As time goes on, many new face robots become more iconic than realistic. Daniel BazoBazo et al. (2010) were using a hybrid face robot called BERT2 as their social robot platform, where the eyes movements were realized by a LCD display. Yurii VasylkivVasylkiv et al. (2021) introduced the tabletop robot Haru for the routine behaviors recommendation system. They are using screens to represent a robot expressions. Alessandro RonconeRoncone et al. (2016) form Yale and Matthew PanPan et al. (2020) from Disney Research mainly focused on the gaze movements of the humanoid robot.

In our project, we proposed a new facial expression robot that can predict human expression during interaction, and therefore produce natural and agile facial expressions. We have illustrated the hardware upgrades in the report last semester. This summer, we mainly focus on the two learning models for the face simexperssion, the inverse model and the predictive model.

1.2 LIPS SYNCHRONIZATION

Lip synchronization is the technology that matches a person's lips movement with a given piece of speech audio, the topic can sometimes get generalized to also sync up with related facial expressions. Common application includes creating talking animations of virtual characters, fabricating a speech video of a person (Deep Fake)Nguyen et al. (2019), or creating a humanoid robot face that speaks with realistic lip movement Oh et al. (2010). Old methodologies that create realistic lip synchronization usually involve a large amount of hard-coded lips/facial commands, making them unscalable to new speech data. Whereas existing scalable lip synchronization methods usually oversimplify lips movement preserving only mouse opening and closing.

Fortunately, the technology has become increasingly mature in the computer graphic field where human speech patterns and facial movements could be learned directly from large-scale speech visual-audio datasets with end-to-end neural network models. Prajwal's Wav2Lip model Prajwal et al. (2020) utilizes a GAN architecture with loss from a pre-trained lip-sync expert. Lahiri et al. Lahiri et al. (2021) utilizes a video-based learning framework to animate 3D talking faces from audio. The end-to-end models coupling the speech-related and identity-related model such as the implementation of the recurrent model combines the frame-to-frame and context information well Song et al. (2019). Based on that, implementations such as the Wav2Lip and Zhou's model Zhou et al. (2020) decouple the identity and speech information and focus on the capability to generate arbitrary-subject speaking with high fidelity. Nonetheless, much of the techniques focus purely on lips while ignoring the natural control of other facial features.

As for lip-syncing in the humanoid robotic face, due to the lack of dedicated robot datasets or a missing pipeline to transfer lip-sync knowledge from existing human speech audio datasets onto the robot face, much of the current methods still require hard-coded expression. Chen Chen et al. (2021) proposed a two-stage framework with a generative network that synthesis robot facial expression images and an inverse network that actuates robot face to express the facial expression in previously generated images. In this paper, we present an audio-to-landmark generator, as an extension/modification to Chen's framework, that eventually matches robot face and lips movement to a given piece of audio through trajectories purely learned from human speech visual-audio dataset.

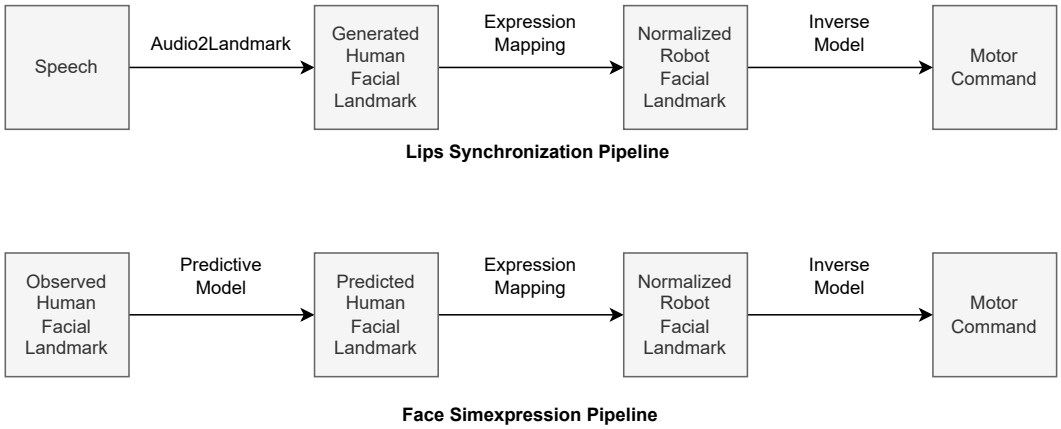


Figure 2: The two pipelines of the project: Lips Synchronization (top) and Face Simexpression (bottom)

2 Overall Pipeline

This project focuses on two aspects of the animatronic robot face we made: 1). robot lips synchronization and face simexpression. Each has its own pipeline with a total of 4 different modules: audio2landmark (section 3), predictive model (section 6), expression mapping (section 5), and inverse model (section 4) as shown in figure 2. The expression mapping and inverse model is used in both pipelines.

3 Audio2Landmark

Given a piece of speech wave (mp3, wav, m4a, etc) and a frame of the static face, the Audio2Landmark pipeline animates the static face by predicting the lip-synced landmark displacement. A detailed documentation (data processing, architecture, training, etc.) about the audio2landmark module can be seen in Yunzhe's Spring 2022 research report (request needed). This section mainly discuss updates since then.

3.1 Model Architecture

The pipeline first transforms the audio wave into a log-Mel-spectrogram and unfolds it into overlapping slices with a length roughly equal to 0.5 seconds, it then feeds each slice to the APC model to get a speech representation of the same length in time. After that, the speech representation is compressed along the time dimension by feeding through three Bidirectional-LSTM layers and then compute a (learnable) weighted context vector through the Luong attention mechanism Luong et al. (2015), and use the context vector as the compressed embedding.

3.2 Evaluation

We experienced with various net architectures, smoothing and audio slice length hyper-parameters. We also did ablation evaluation that removes the APC module. Similar to the evaluation metrics as used by Zhou et al. (2020), we used 6 different metrics to evaluate the fitting of the generated landmarks and the ground-truth landmarks regarding the lips and jaw movements, eye blinks, and holistic facial expression.

Difference in Position of all landmarks (D-P): Computes the average euclidean distance between the predicted landmark positions and the ground-truth landmark positions.

Difference in Velocity of all landmarks (D-V): Computes the average euclidean distance between the predicted landmark velocity and the ground-truth landmark velocity, where the velocity is calculated as the first-order difference of landmark positions along the time dimension.

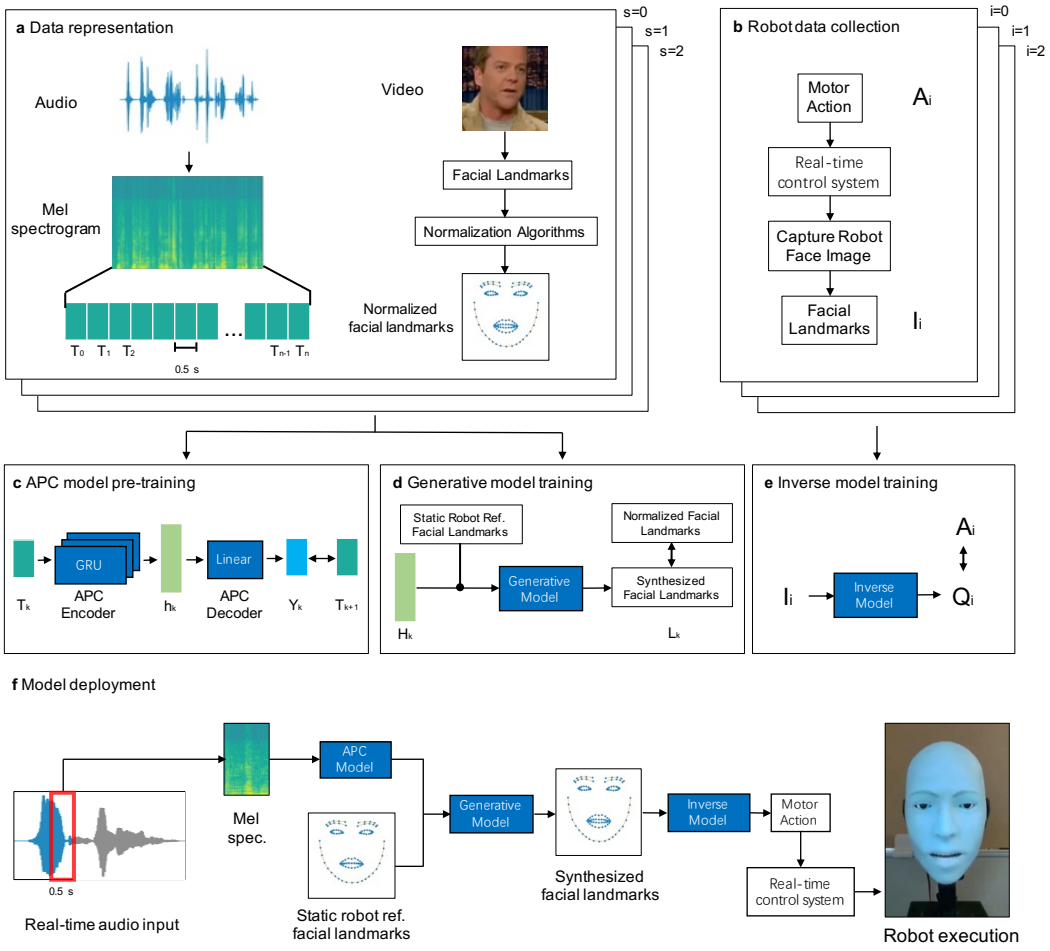


Figure 3: Overall Pipeline of Robot Lips Synchronizaiton

Difference in Position of Lips-and-Jaw landmarks (D-PL): Computes the average euclidean distance between the predicted landmark positions and the ground-truth landmark positions for just the lips and jaw regions.

Difference in Velocity of Lips-and-Jaw landmarks (D-VL): Computes the average euclidean distance between the predicted landmark velocity and the ground-truth landmark velocity for just the lips and haw regions.

Difference in Area of opened-inner-Lips landmarks (D-AL): Computes the absolute value difference between the predicted inner-lips area and the ground-truth inner-lips area. The inner-lips area is calculated through the Shoelace area-of-polygon formula.

Difference in Area of opened-Eyes landmarks (D-AE): Computes the absolute value difference between the predicted opened-eyes area and the ground-truth opened-eyes area. The opened-eyes area is calculated through the Shoelace area-of-polygon formula.

The evaluation result is shown in table 1. All models are evaluated at epoch 15 after trained on the test set of VoxCeleb2. We can see that the model with attention mechanism improves the overall accuracy. Also, the APC model helps with fitting the position of the landmarks but not the velocity. By visually observing the generated landmark animation, we discovered that the model APC+LSTM+Atten. better captures lips motion details so we select it as the finalized version. Lastly, even though we have a evaluation metric for eye-fitting, we observed that non of the models were able to capture eye blinks due to the little correlation eye blinks has with speech. We believes that a probabilistic generative module dedicated to eye behavior can better improves the performance.

When preparing demos of the audio2landmark module, we found that our model can generalize very well to unseen speakers and languages that never shows up in the training data. We believes this is because we make inference based on short audio segments, which in a way breaks down spoken languages into universal phonetic unites that has no boundary to any specific language. The pretrained APC module also gives model an ability to foresee future phonetics given the current speech segment thereby improves the robustness of speech representation.

Model	**D-P** $\downarrow$	**D-V**$\downarrow$	**D-PL**$\downarrow$
Random Baseline	156.699$\pm$44.80	193.688$\pm$54.11	84.856$\pm$24.09
APC+LSTM	38.018$\pm$10.51	14.917$\pm$3.61	22.317$\pm$6.32
LSTM+Atten.	38.101$\pm$9.62	**14.859$\pm$3.60**	22.356$\pm$5.60
APC+LSTM+Atten.	**37.942$\pm$9.61**	14.926$\pm$3.60	**22.245$\pm$5.55**
Model	**D-VL**$\downarrow$	**D-AL**$\downarrow$	**D-AE**$\downarrow$
Random Baseline	104.701$\pm$29.23	2.332$\pm$1.16	2.124$\pm$1.16
APC+LSTM	9.584$\pm$2.26	1.220$\pm$0.55	0.650$\pm$0.30
LSTM+Atten.	**9.547$\pm$2.25**	1.242$\pm$0.47	**0.643$\pm$0.28**
APC+LSTM+Atten.	9.613$\pm$2.25	**1.217$\pm$0.45**	0.652$\pm$0.28

Table 1: Different model architecture evaluated with 6 different metrics on the VoxCeleb2 test dataset. All models are trained with 15 epochs with audio slice width 25 (0.5 seconds) and no landmark smoothing.

3.3 REALTIME INFERENCE

Since the Audio2Landmark pipeline makes landmarks inference based on a tiny fragment (0.5 sec) of audio information where the time gap between is 0.04 sec (25 fps). We developed an inference pipeline that readsn speech audio from a microphone as stream to enable real-time inference. This could enable robot realtime conversation when long sentences are generate and speaks on the fly.

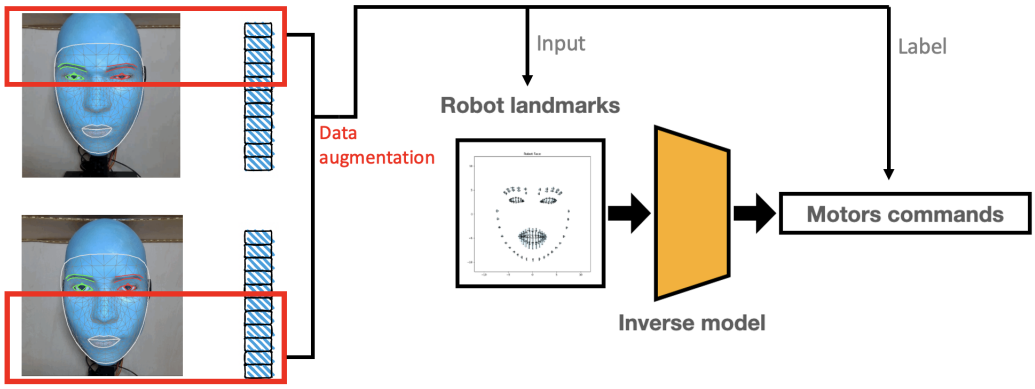


Figure 4: Inverse model training

4 INVERSE MODEL

4.1 DATA COLLECTION

Our goal for the inverse model is to enable the facial robot to mimic human expression in real time. The learning model was designed to map from landmarks to motor commands.
The process of data collection is this, sending random commands to the robots and recording the images frames of the robot's face by an RGB camera
Then, we did the landmarks detection using Mediapipe. The simplified matric face landmarks were used as the input data, and the corresponding motor commands were used as the labels.
Some methods were used to simplify this problem. For example, all the motor commands were normalized between 0 and 1. Also, we are using a streamlined version of landmarks given by Mediapipe, which is a set of 113 2d-array. During the data collection, the robot was doing symmetrical expressions, which we think can cover most of the situation and can reduce the size of the learning model.

4.2 DATA AUGMENTATION

The dataset was collected separately. One is the eyelids and eyebrows landmarks and motor commands, and the other is the mouth and jaws landmarks and motor commands. 1000 data has been collected for each of these two parts. After doing the combination, we can have more than 1 million data for the complete facial expressions and motor commands. Data splitting for the training and testing data should be done before the combination. For example, using 800 eyes data and 800 mouth data for the training dataset, we have 640000 training data and 40000 testing data. Otherwise, if the data splitting is done after the combination, the training data would contain the information of the testing data, which would make the testing loss lower than the training loss during training. The data augmentation and Training process are shown in fig 4.

4.3 NORMALIZATION

The inverse model takes in the robot expression frames and outputs motor commands. The robot face and human faces have different features. A normalization module has been designed to normalize different human faces to the robot face. The process is shown in fig5 and equation1. For each frame, we calculate landmark distances, multiply by a reduction rate α, then add the shrunken distances to the robot's static face. If the landmarks are still out of robot landmarks moving space, they will be restricted to the edges.

$$(Hframe_{(2\times113)} - Hstatic_{(2\times113)}) \cdot \alpha + RobotStatic_{(2\times113)} = RobotFrame_{(2\times113)} \quad (1)$$

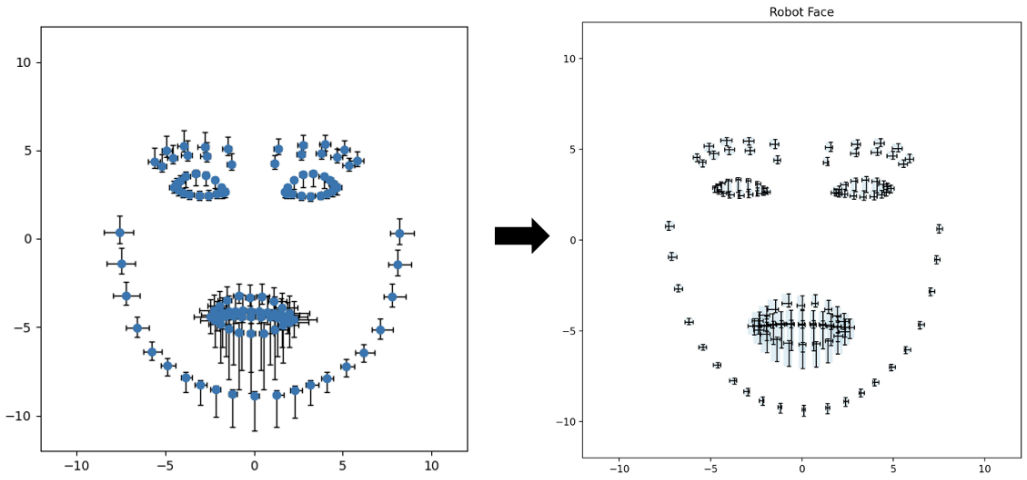


Figure 5: Normalization, human to robot landmarks

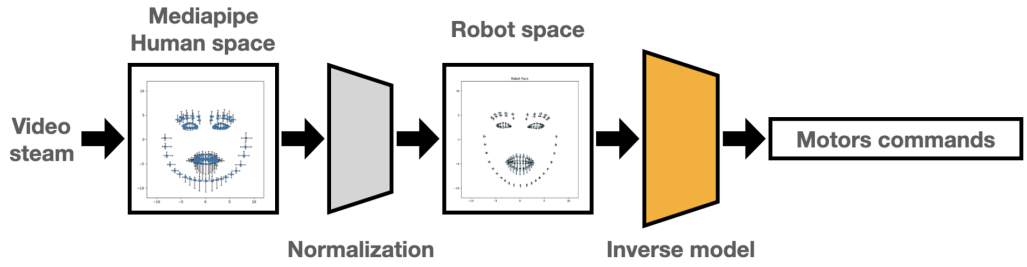


Figure 6: The mimic pipeline

4.4 Model architecture and the expression mimic pipeline

The learning model architecture includes three layers of fully-connected neural network. Batchnormal was used for the first layer. The input size is 226 (113 landmarks' x and y coordinates), and the output size is 11. Some other methods have also been used, including the residual neural network, which is not doing better than fully connected networks. The complete pipeline of the expression mimic pipeline is shown in fig6. Firstly, with the human expression frames from video steam, obtaining the landmarks using Mediapipe. Secondly, normalizing the human landmarks to fit into the robot landmarks' moving space. Thirdly, using the inverse model to get the motor commands with landmarks input. Finally, sending the motor commands to the robot hardware to produce the same expression as the human frames.

Compared with the last generation's inverse model, where the input is the robot facial expression images, this time we use landmarks as the representation, which makes the model much lighter, and can run in real-time interactions.

5 Expression Mapping Network

For the future pipeline we design a new architecture that directly maps the generated human face landmarks to the robot face landmarks in replacement of the current normalization implementation. In this way the pipeline will be faster without collecting and calculating the target facial information (distance for calculating the reduction rate α), and enables a more flexible end-to-end mapping.

We explain the facial feature with separated elements such as face width, height, and gender, and compute corresponding feature vectors to generate faces that mimic the real-world face sample space.

5.1 DATASET

To obtain the distribution of real-world human facial features, we first use the CelebA dataset to generate different facial labels with the Baidu Cloud and Tencent Cloud Facial Attribute Detection API. With the label data we compute the corresponding direction vectors for each attribute and apply them to the original robot faces to mimic the real-world facial counterparts as the pair training data. Considering the application of the landmarks on motor commanders which are currently only relevant to the eye, eyebrow, mouth, and the cheek part, we use the simplified version of the robot face landmarks, where each frame of the face consists of 113 points with x and y position.

5.2 FACIAL FEATURE PROCESSING

We get the labels from the Baidu Cloud and Tencent Cloud Facial Attribute Detection API for the direction vector computation regarding each single facial features. We currently select 11 labels that impact the face contour identity information, including face width, height, gender, age, eye size, eyebrow thickness, eyebrow curve, eyebrow length, face shape, and lip thickness. Each label has different factor contribution to the overall landmark identity, which requires further adjustment on parameters. Labels such as width and age are quantitative and others such as gender and face shape are categorized with the corresponding possibility ranging from 0 to 1. Due to the time and quantity of image processing limits of the free API use, we select 10000 of the overall celebA dataset for label processing.

5.3 FEATURE VECTOR COMPUTATION

We compute the feature direction vectors with the inspiration from the linearity between the latent code and the corresponding image feature transformation of the StyleGAN model Karras et al. (2018). The latent code of the StyleGAN model, which is decoded to different attributes at different detail levels, can be linearly manipulated with certain direction vectors to enlarge or deduct the effect of certain facial features. As we are only working with the landmarks instead of pixels, we get rid of the complicated latent space of high dimensions and only focus on the operations of landmark coordinates. As the relative position of coordinates have a similar linear relation with the contour identity information, we conduct an iteration of 500000 ($n = 500000$) times to randomly select two facial vectors v_1 and v_2, composed by landmark coordinates v_{lm} and a single attribute label v_{lb} each time, to apply to the following equation2:

$$\vec{v_d} = \frac{\sum_{iter=1}^{n} \frac{v_{lm1} - v_{lm2}}{v_{lb1} - v_{lb2}}}{n} \tag{2}$$

The displacement and pose differences between faces can be neutralized with large numbers of iterations if the label is largely landmark-based. For example, the age attribute relies more on details pixel-wise and its relation with landmark coordinates may be trivial. Thus the displacement and pose angle differences may interfere the impact of relative position of landmarks, which lowers the accuracy of the direction vector (see "age" in fig 7). In contrast, the width of the face is highly related to the landmark coordinates and thus will generate a more accurate vector direction (see "width in "fig 7). After getting the computed direction vectors, we randomly arrange them through linear combinations multiplied with a feature weight β and apply the final vectors to each of the original robot face to generate faces mimicking the real-world randomized distribution.

$$\vec{v_{r'}} = \sum_{i=1}^{n} \vec{v_0} + \beta_i \cdot \vec{v_{d_i}} \tag{3}$$

Fig 8 is an example that shows through manipulating the gender and width vectors of the robot face, we get two faces with different contour identity without losing the expression information and treat those two as the real-world human face landmarks related to the original robot face.

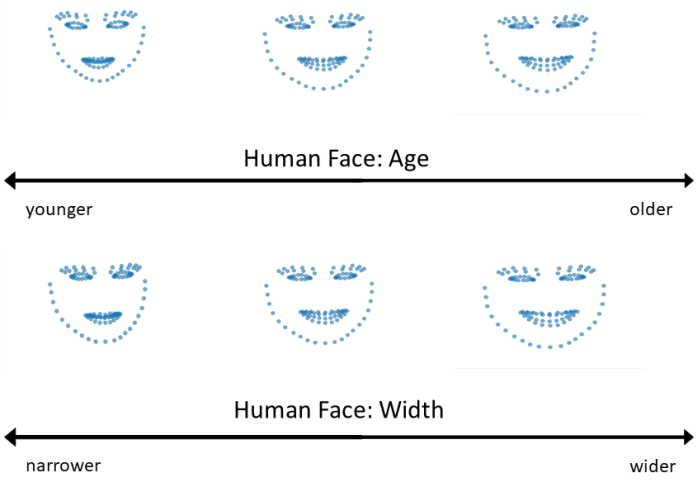


Figure 7: Human Landmark Age Transform

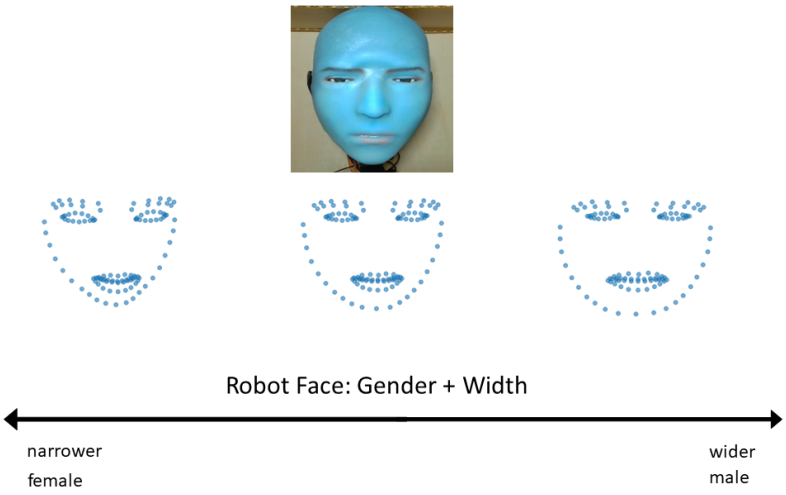


Figure 8: Robot Landmark Gender Width Transform

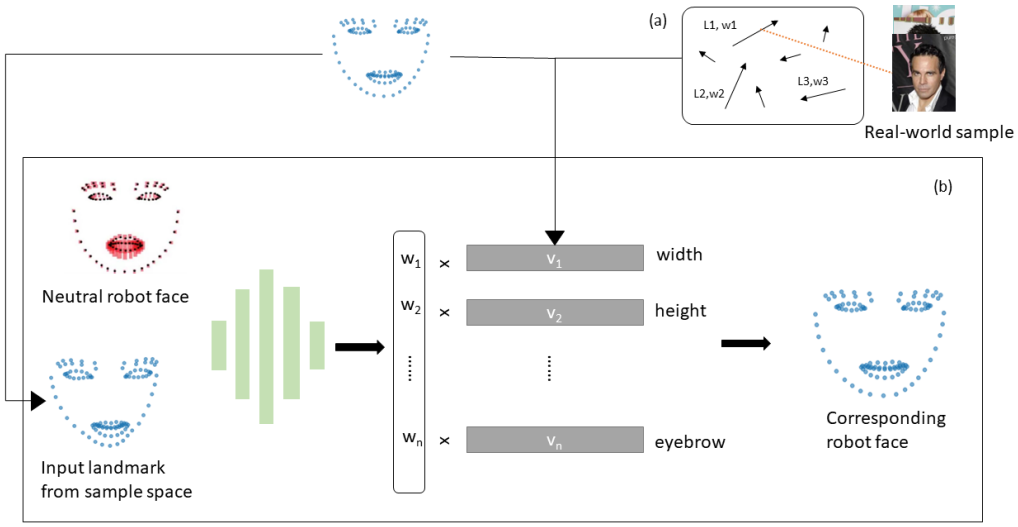


Figure 9: The Overall Pipeline for Expression Mapping Network

5.4 Network Architecture

With direction vectors generated (see part (a) in fig 9), we create a near-real-world sample distribution of the human faces corresponding to their robot counterparts. We then input the synthesized near-human face landmarks together with a neutral robot face landmarks ($2 \times 113 \times 2$) into a light-weighted network (see part (b) in fig 9) composed of multiple fully connected layers and batch Norm layers to learn the corresponding weight w (1×11) for each of the facial feature vectors. The objective is created by the corresponding robot landmarks from the original dataset with a comparison to the synthesized landmarks applied with linear combinations with the predicted weight:

$$f = \sum_{i=1}^{113} ||\vec{v_0} - (\vec{v_{in}} + w \cdot \vec{v_d})||^2 \tag{4}$$

6 Predictive Model

Enabled with the inverse model, our robot can imitate human facial expressions in real time. However, imitation is not a natural mode of human communication. A human can capture subtle facial expressions and regard each other's emotional changes accordingly. Therefore, we developed a predictive learning model to predict the target expression by capturing the facial landmarks' movements. With both the predictive model and the inverse model, our robot can interact with humans more naturally.

The complete pipeline of the facial expression robot includes the inverse model, the predictive model, and hardware control. We also reconstructed our project with ROS2 to make different modules of the robot run in parallel.

6.1 Predictive model training

The dataset we used is the MMI facial expressions dataset. This dataset includes 49 subjects with a total of 2387 expression frames. We hope to train a predictive model that can predict the expression landmarks before a human makes the actual expression. The ideal set of data should have a stable start and a clear change in expression. We filtered the data before training the model. Because the

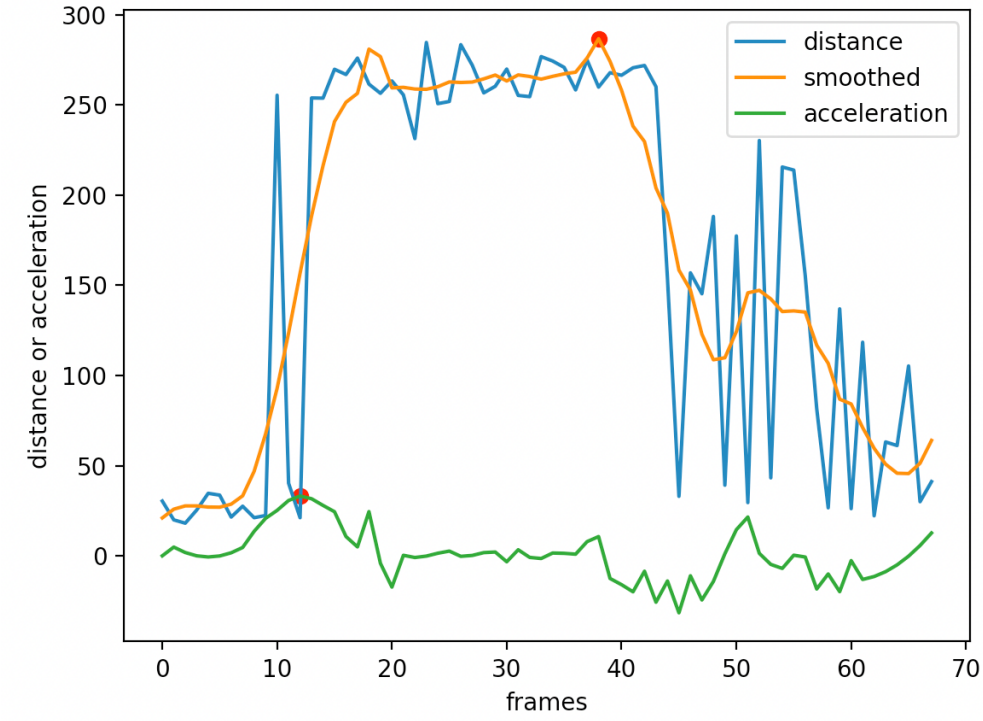


Figure 10: Distance and acceleration plot

original landmark data have problems such as data noise, and the acceleration peaks might come after the distant peaks.

We described a set of frames using the sum of the landmarks distance between each frame and the first frame. Then we smoothed the data using the moving average methods. Next, we can calculate the acceleration plots and find the frames that have the most dramatic expression changes. The ideal set of frames should have a distance plot and acceleration plot like the fig10.

We use the frames around the acceleration peak as the input and the frame at the distance peak as the label. The predictive model can be used as the pre-module of the inverse model. It predicts the target expression landmarks and then the inverse model maps the landmarks to the motor commands.

The combination of these two models works quite well. We did the questionnaires comparing the mimic pipeline with and without the predictive model, and most people could tell that the former had more natural interaction.

6.2 Model Evaluation

6.2.1 Inverse Model Evaluation

Comparing the inverse model output and the data labels, we can have the average loss on each command. As is shown in fig11 We can see that the average loss is 0.0779, and the max loss appears in the mouth motor commands. We also compared our methods with some baselines, such as the nearest neighbor, random input, and randomly generated commands (fig12). The mean loss of random methods is around 0.33. The mean loss of the nearest neighbor is 0.21. This method is 30 times slower than our inverse model and is not as accurate.

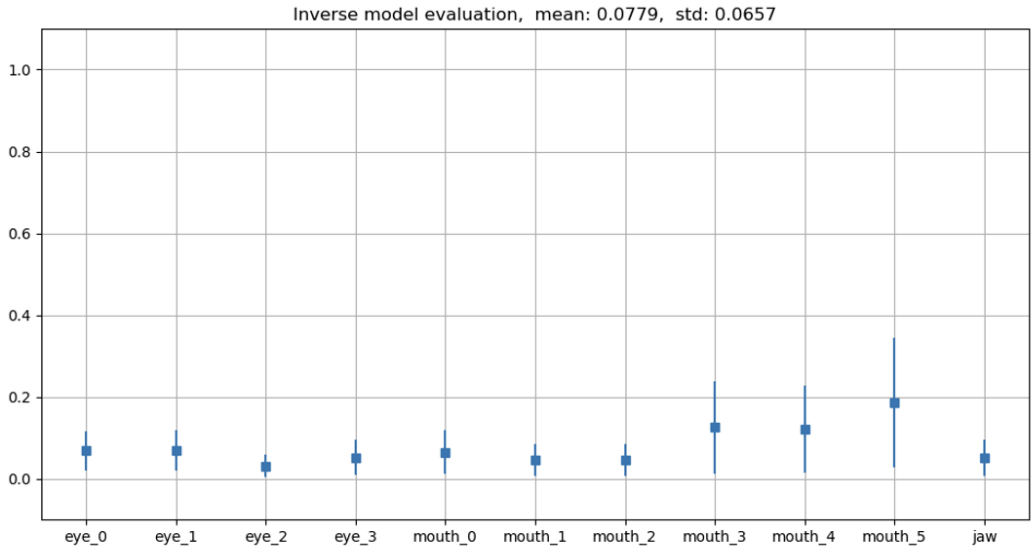


Figure 11: The average loss on each command

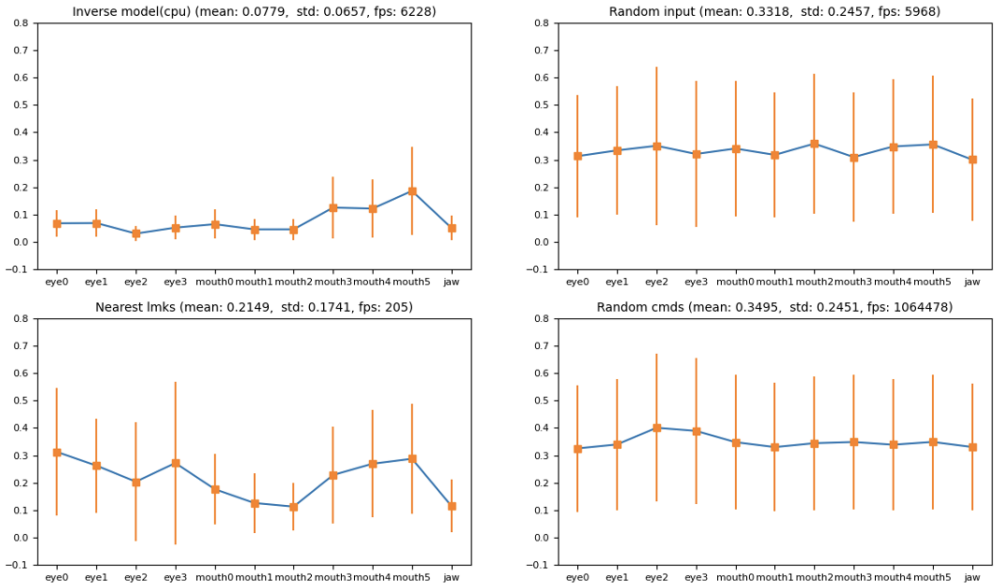


Figure 12: Comparing our method and the baselines

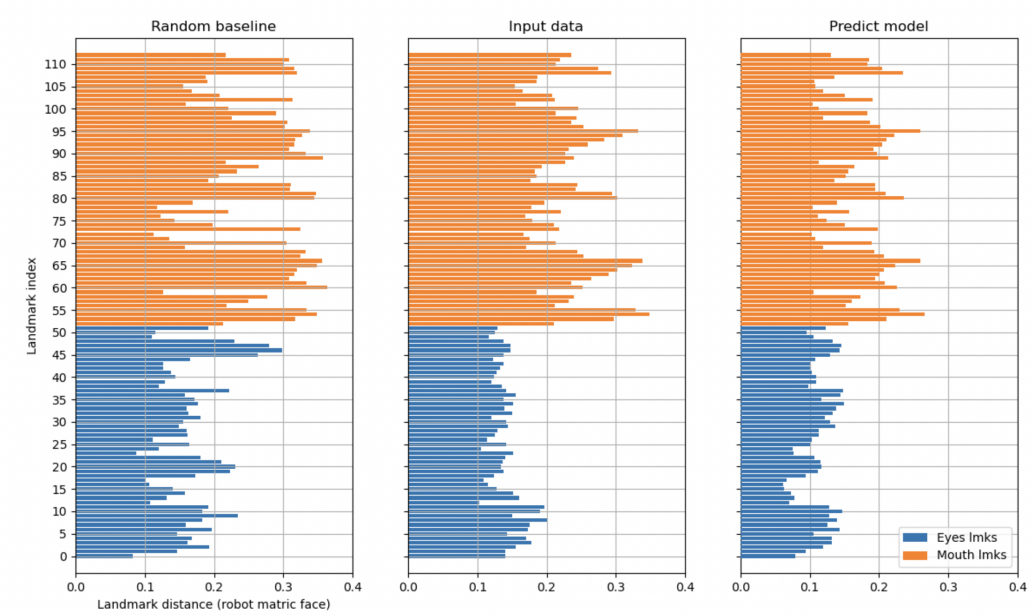


Figure 13: The average loss on each landmarks, comparing our method with the baselines

6.2.2 Complete pipeline evaluation

The complete pipeline was evaluated by comparing the landmarks of the output facial expressions and the target landmarks in the predictive model dataset. As is shown in fig13, we can see the distance on each of the 113 landmarks. The baselines are the landmarks of the randomly selected faces, and the landmarks of the input face. We can see that our predictive model is working better than both baselines.

7 Hardware control updates:Robot Operating System

In the last generation, we hoped the robot to control different modules in parallel. At that time, our robot had a more than 1-second delay doing the mimic pipeline. After upgrading the hardware and redesigning the mimic pipeline, our robot can run facial expressions in real time. This summer, we still tested running our robot in ROS2, hoping to enable the robot to do a more smooth control. We used the publisher and subscriber mode of communication. In this mode, if we set the camera as the publisher, the camera node always publishes the newest frame, and the tracking node and inverse model node, as the subscribers, only take in the frames when they finish processing the previous frames. We also set the other modules as independent nodes, such as the eyes motors control module and the mouth motors control module. The entire structure of the robot operating system is shown in the fig14.

While controlling different modules in parallel indeed smoothed the tracking demo of the robot, the learning model nodes were not giving a decent performance. One reason is the moving camera will have fuzzy frames which can cause unstable in the output of the commands. One possible approach is to set the camera closed when the robot is moving. If the learning models only take in frames when the images are steady, the output commands can have a good performance as we expected.

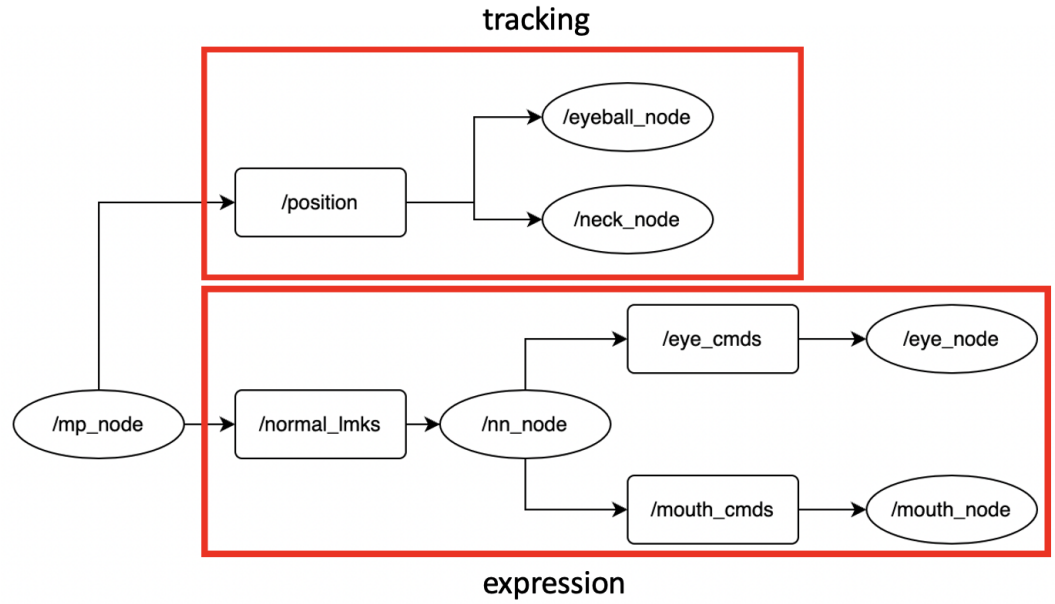


Figure 14: Structure of the robot operating system in our robot

REFERENCES

Daniel Bazo, Ravi Vaidyanathan, Alexander Lentz, and Chris Melhuish. Design and testing of a hybrid expressive face for a humanoid robot. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5317–5322. IEEE, 2010.

Karsten Berns and Jochen Hirth. Control of facial expressions of the humanoid robot head roman. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3119–3124. IEEE, 2006.

Boyuan Chen, Yuhang Hu, Lianfeng Li, Sara Cummings, and Hod Lipson. Smile like you mean it: Driving animatronic robotic face with learned models. *arXiv preprint arXiv:2105.12724*, 2021.

Fumio Hara and Hiroshi Kobayashi. A face robot able to recognize and produce facial expression. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*, volume 3, pp. 1600–1607. IEEE, 1996.

Takuya Hashimoto, Sachio Hitramatsu, Toshiaki Tsuji, and Hiroshi Kobayashi. Development of the face robot saya for rich facial expressions. In *2006 SICE-ICASE International Joint Conference*, pp. 5423–5428. IEEE, 2006.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. URL `http://arxiv.org/abs/1812.04948`.

Hiroshi Kobayashi and Fumio Hara. Facial interaction between animated 3d face robot and human beings. In *1997 IEEE international conference on systems, man, and cybernetics. Computational cybernetics and simulation*, volume 4, pp. 3732–3737. IEEE, 1997.

Avisek Lahiri, Vivek Kwatra, Christian Frueh, John Lewis, and Chris Bregler. Lipsync3d: Data-efficient learning of personalized 3d talking faces from video using pose and lighting normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2755–2764, 2021.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Thanh Thi Nguyen, Cuong M Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2019.

Kyung-Geune Oh, Chan-Yul Jung, Yong-Gyu Lee, and Seung-Jong Kim. Real-time lip synchronization between text-to-speech (tts) system and robot mouth. In *19th International symposium in robot and human interactive communication*, pp. 620–625. IEEE, 2010.

Matthew KXJ Pan, Sungjoon Choi, James Kennedy, Kyna McIntosh, Daniel Campos Zamora, Günter Niemeyer, Joohyung Kim, Alexis Wieland, and David Christensen. Realistic and interactive robot gaze. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11072–11078. IEEE, 2020.

KR Prajwal, Rudrabha Mukhopadhyay, Vinay P Namboodiri, and CV Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 484–492, 2020.

Alessandro Roncone, Ugo Pattacini, Giorgio Metta, and Lorenzo Natale. A cartesian 6-dof gaze controller for humanoid robots. In *Robotics: science and systems*, volume 2016, 2016.

Yang Song, Jingwen Zhu, Dawei Li, Xiaolong Wang, and Hairong Qi. Talking face generation by conditional recurrent adversarial network. *arXiv preprint arXic:1804.04786*, 2019.

Yurii Vasylkiv, Zhen Ma, Guangliang Li, Eleanor Sandry, Heike Brock, Keisuke Nakamura, Irani Pourang, and Randy Gomez. Automating behavior selection for affective telepresence robot. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2026–2032. IEEE, 2021.

Yang Zhou, Xintong Han, Eli Shechtman, Jose Echevarria, Evangelos Kalogerakis, and Dingzeyu Li. Makeittalk: Speaker-aware talking-head animation. *arXiv preprint arXic:2004.12992*, 2020.

A WHO DID WHAT

- Yingke Wang
  - Process and create a robot-face-based real human face sampling dataset
  - Construct the facial feature edition algorithm and generate direction modules
  - Construct the human-robot face mapping network based on the attribute vectors
  - Improve and design the model structure part of inverse model with Jiong
- Yunzhe Wang
  - Updated the Audio2Landmark model (APC+LSTM version) with attention mechanism (finalized version)
  - Experimented with effects of audio slice width, smoothing, and conducted ablation study that removes the APC model.
  - Experimented and prepared demo to evaluate the model's ability to generalize to different languages, speakers, breath sound, gibberish, and susceptibility to background noises.
  - Enables audio2landmark real-time inference
  - Literature review on probabilistic synthesis model aiming to fit eye blinking and head movement. Attempted speech prosody (emotion) feature disentanglement. Both leave to future work.
- Jiong Lin
  - Inverse model: data collection, training, evaluation
  - Predictive model: data collection, normalization, evaluation
  - Robot Operating System testing

A Two-step Generative Pipeline for Realistic Robot Face Lip Synchronization, Fall 2021, Dec 31

Yingke Wang
Department of Computer Science
Columbia University
yw3821@columbia.edu

Yunzhe Wang
Department of Computer Science
Columbia University
yw3737@columbia.edu

Abstract: Given a piece of speech audio and a reference person, lip synchronization matches the person's lips/facial movement to make it seems like the person is delivering the speech. Accurate and natural lip movement is a crucial component for better human-robot interaction and communication. In this paper, we present a two-step pipeline that could eventually lip-sync robot facial movement through knowledge purely learned from human speech visual-audio data. The pipeline consists of two separate modules: a landmark generator that generates face landmark based on given audio, and a robot image generator that produce authentic robot face images based on the previously generated landmark as well as the facial identity information. Several training experiments were conducted to evaluate the performance of our pipeline as well as the quality of generated landmark and robot images. We conclude that our pipeline at the current stage could already make relatively accurate predication on both the landmark generator and the robot image generator. Nonetheless, we identified several modifications and directions that could be made regarding data acquisition and pipeline design to further improve performance.

Keywords: Lip Synchronization, GAN, Robotics, HCI

1 Introduction

1.1 Background and Literature review

Lip synchronization is the technology that matches a person's lips movement with a given piece of speech audio, the topic can sometimes get generalized to also sync up with related facial expressions. Common application includes creating realistic animations of talking characters [1], fabricating a video of a person doing a speech he or she has never done (Deep Fake)[2], and creating a humanoid robot face that speaks with realistic lip movement [3]. Old methodologies that create realistic lip synchronization usually involve a large amount of hard-coded lips/facial commands, making them unscalable to new speech data. Whereas old scalable lip synchronization usually oversimplifies lips movement with only open/close mouse movement. Fortunately, the technology has become increasingly mature in the computer graphic field by learning human speech patterns and facial movements directly from large-scale speech visual-audio datasets with end-to-end deep generative models. Prajwal's Wav2Lip model [4] utilizes a GAN architecture with loss from a pre-trained lip-sync expert. Lahiri et al. [5] utilizes a video-based learning framework to animate 3D talking faces from audio. The end-to-end models coupling the speech-related and identity-related model such as the implementation of recurrent model combines the frame-to-frame and context information well [6]. Based on that, implementations such as the Wav2Lip and Zhou's model [7] decouple the identity and speech information and focus on the capability to generate arbitrary-subject speaking with high fidelity. Nonetheless, much of the techniques focus purely on lips while ignoring the natural control of other facial features. As for lip-syncing in the humanoid robotic face, due to the lack of dedicated robot datasets or a missing pipeline to transfer lip-sync knowledge from existing human speech audio

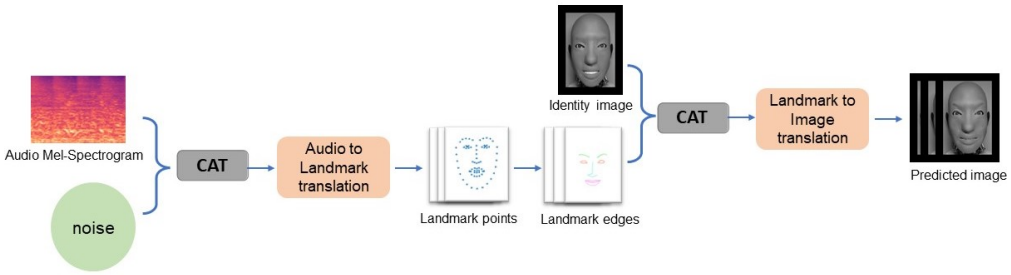


Figure 1: Overview of Our Algorithm

dataset onto robot face, much of the current methods still requires hard-coded expression. Chen proposed a two-stage framework with a generative network that synthesis robot facial expression images and an inverse network that actuates robot face to express the facial expression in previously generated images [8]. In this paper, we present a two-step generative pipeline, as an extension to Chen's framework, that matches robot face and lips movement to a given piece of audio through trajectories purely learned from human speech visual-audio dataset.

1.2 Goals and Motivation

Our motivation for this project is to design an algorithm that takes the sequence of the audio as the input and outputs the corresponding sequential images with natural lip synchronization and facial expression. This will be one of the modules of the entire software composition of the robot, which enables it to speak naturally like a human. The goal for this semester is to complete a baseline model that can translate the sequence of audio to corresponding landmarks and then translate the landmarks into robot face images.

2 Method/Technical approach

Our Lip Synchronization model breaks down to two separate modules that are linked sequentially: the landmark generator followed by the robot image generator. The landmark generator uses a Generative Adversarial Network (GAN) architecture that takes the concatenation of random noise and an audio signal, aiming to produce human face landmarks that capture lips movements as well as facial expressions while talking. The robot image generator uses a U-Net architecture that takes the concatenation of landmarks and identity image of the robot face, aiming to generate the target face which is based on the input landmark information and at the same time contains the face identity information from the input. Combining the two models, the whole architecture takes the audio as input and outputs the robot face image with the corresponding lip synchronization and facial expression as shown in 1.

2.1 Landmark Generator

As the first module of the pipeline, the landmark generator model aims to learn the natural lips movement and subtle facial expression change when humans speak. Such lips/facial movement could be represented as facial landmarks (x,y coordinates on a 2D plane). We decided to adopt a GAN architecture for its successful applications in various prior lip-sync methods and audio-visual deep fake technologies [4, 5, 2]. Our full pipeline diagram can be seen in figure 2.

2.1.1 Landmarks Acquisition with VoxCeleb2 and Openpose

To build a generator model that learns human facial patterns during a speech, a large-scale audio-visual dataset of human speech is needed. We targeted at the VoxCeleb2 dataset [9] due to its abundance and the fact that most previous successful lip-sync and facial deep-fake models were

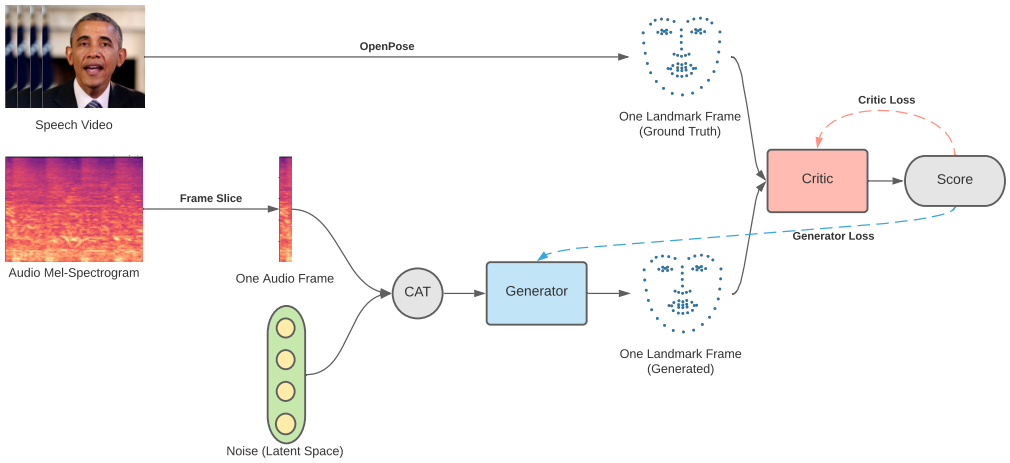


Figure 2: Full Pipeline of the Landmark Generator

built upon this dataset. Each data instance is a video clip in mp4 format of a celebrity speaking, where the face landmarks could be extracted using OpenPose, an open-sourced software that jointly detects human body, hand, facial, and foot keypoints on single images [10, 11]. The software runs relatively slow without high-end computing resources, so we adopted the Google Colab Pro GPU environment, and store acquired landmarks on Google drive, which could be later exported.

VoxCeleb2: The VoxCeleb2 dataset contains over 1 million utterances for 6,112 celebrities, extracted from videos uploaded to YouTube. Each utterance is a video clip of a celebrity speaking, usually in a interview or public speech. Each video has fixed resolution of 224x224, frame rate at 25 fps, and average duration of 7.95 seconds. Table 1 contains size details. The dataset is fairly gender balanced, with 61% of the speakers male. The speakers span a wide range of different ethnicities, accents, professions and ages. Videos quality ranges from professionally shot multimedia to crude hand-held device. The dataset also provides face detection and face-tracks where faces are normally centered, but with variations in head pose.

Video Stats	**Dev**	**Test**
# of speakers	5,994	118
# of videos	145,569	4,911
# of utterances	1,092,009	36,237
# of frames	∼215,958,859	7,200,945
File Size (zip)	248.62 GB	8.29 GB
Total Duration	∼2399.53 Hours	80.01 Hours

Table 1: VoxCeleb2 Dataset description statistics

Openpose: Openpose is a open-sourced software that jointly detect human body, hand, facial, and foot keypoints. In our case, we only need the facial data. Specifically, to extract facial landmarks given a video file 'input.mp4' and store the landmarks into json files in an output folder, we can run the following command:

```
./openpose.bin --video path/to/input.mp4
               --face
               --keypoint_scale 3
               --write_json path/to/output
```

In this command, the `--video` argument specifies the path to the input video file. The `--face` argument specifies that we want the output to include face landmarks. The `--keypoint_scale` argument with option 3 asks to scale the output coordinate within the range $[0-1]^2$. Finally, the `--write_json` argument ask the program to store the generated landmarks into a folder where each frame of the video writes a unique json file. The command can only process one video at a time. so an automate script is needed for batch processing. After running the command, the program creates a folder and populate it with json files under naming format `{input_filename}_{frame_number}_keypoints.json`. Inside the json file, you can locate all landmarks at the 'face_keypoints_2d' attribute, which is a size 210 list containing the position (x, y coordinates) and respective confidence score of 70 keypoints in an order of $(x_1, y_1, c_1, x_2, ..., c_{70})$. The "mean face" of face landmarks is plotted in the right image in figure 4. Note that output coordinates from OpenPose has origin at the upper left corner but the plot has original at the lower left corner.

Google Colab: Since the VoxCeleb2 dataset is too big and our computing resource is relatively slow and limited, processing the entire dataset with Openpose seems impracticable. So, we decided to only process a small portion of the data, and we will run the program using Google Colab. With Colab Pro subscription, we usually get allocated with a Tesla P100 GPU. For better model generalization, we aim to process videos for more unique speakers rather than more videos for few speakers. The data file needs to be uploaded to Google Drive before it can be used by Colab. To reduce the size for a single upload, we divide the dataset into small subsets where each one is a zip file containing a unique video for each speaker.

Difficulty	Solution
Single data file is too large to upload	Extract a small subset of the data with one unique video for each speaker
Colab free version limits GPU usage for around 3 hours	Purchase Colab Pro for 9.99$/month
Colab will release instance when the tab is closed	Keep browser tab and PC always open
Colab will reach one-time limit after 12 hours	Keep note of where it was interrupted and manually resuming after limit
All data will lost at the Colab size when instance release	Directly store generated data onto Google Drive so data won't loss
Drive will be inaccessible when having larger number of files in a single folder	Export generated landmark data using Google Takeout

Table 2: Difficulties met and solutions found when running Openpose on Google Colab

Data Cleaning: The data acquired by OpenPose sometimes won't contain any key-points or the face landmarks values are all zeros. This is because OpenPose failed to recognize any person or any face in the input video. Such data instance are removed. Also, not every acquired landmarks might be legible for training. This can be shown from the confidence value of the landmarks. Usually, a face with low confidence is ill-recognized or turned sideways and could negatively impact the training result since the goal is to generate high-quality landmarks facing forward. To remove low confidence faces, we fitted a mixed-Gaussian distribution with three curves over the confidence distribution. Here, the confidence for one face refers to the sum of confidence of all 70 landmarks of that face. We set the threshold value for distinguishing good landmarks from bad landmarks as the mean of the highest confidence peak (57.834) minus one standard deviation (1.362). Any face with confidence below this threshold were removed, which leave us about 24.4% of all acquired landmarks.

All valid landmarks data (after removing unrecognized faces and low-confidence faces) are stacked into a matrix of shape 214980x210. A text file recorded the column indices of the matrix corresponding to each video frame after removing bad landmarks. All data files can be found in this Google Drive. A detailed description of landmarks can be found in this Google Doc.

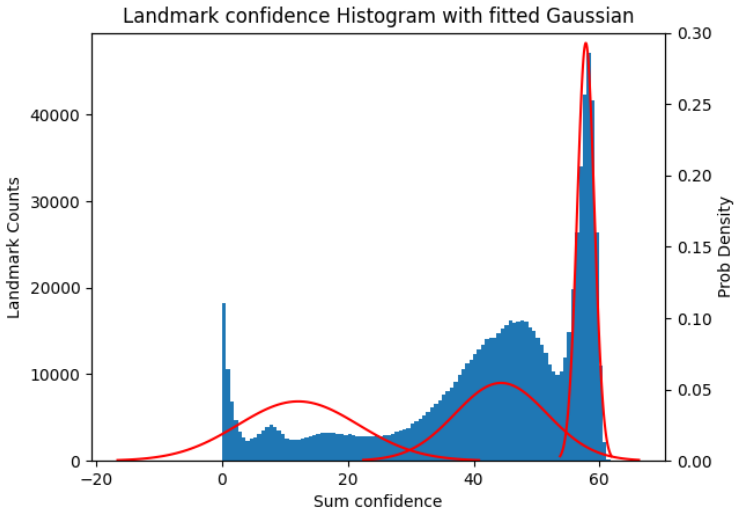


Figure 3: Fitted Gaussian distribution over face confidence distribution. The three fitted curves (right to left) has mean [57.834, 44.359, 12.110] and std [1.362, 7.308, 9.578]

2.1.2 Audio Mel-Spectrogram Processing

The original audio data files are directly download from the VoxCeleb2 Website. Each file is a m4a audio extract from the corresponding video. The audio dataset has the same filename and directory structure as that of the video dataset.

To obtain the audio training data, we extracted the Mel-Spectrogram of each of the m4a file that has corresponding landmark data in the processed matrix using Librosa. This would allows us to apply convolution on audio to capture features across some time overlap when making landmark prediction. Moreover, since the data is human speech, the Mel scale helps normalizes the importance of information at different frequency intervals that's more relevant for distracting speech information. When creating the mel-spectrogram, the parameter `n_mel` – the number of Mel bands – controls the height of the matrix and the parameter `hop_length` – number of samples between successive frames – controls the length of the matrix. The relationship between the hop length h, sample rate f_s, and frame per second fps is that $f_s \div h = \text{fps}$. So, to make sure the width of the output mel-spectrogram matches with the number of video frames, we can set the hop length h as that expressed in equation 1 where t is time in seconds.

$$h = \frac{\text{fps}}{f_s} = \frac{\text{fps} \cdot t}{f_s \cdot t} = \frac{\text{waveform length}}{\text{expected number of frames}} \tag{1}$$

To encompass broader audio information, we can set the hop length as integer multiples of this calculated value so that the output dimension will be multiplied by the same size without mismatching audio and video frames. One of the reason that we want the precise matching between audio and video frame is that some video frames are dropped due to bad quality. And to keep audio examples and video examples the same size, we need to drop the exact matching frames of the corresponding audio.

2.1.3 Wasserstein GAN Architecture and Training Practice

For the model to generate new face landmarks, we choose to build a Generative Adversarial Network due to its prior successful application in synthesizing fake face images and lip-syncing videos [4, 5, 2]. In particular, we choose the Wasserstein variant [12] of GAN for its stable training capability. Since the model should learn how human speaks, the input needs to be conditioned [13] on audio

signals (processed Mel-spectrogram). Figure 2 describes the overall model architecture. For the generator, a noise vector with an adjustable size representing the latent space will be concatenated with the audio Mel-spectrogram. At the first stage, we use a single slice of Mel-spectrogram with an information size equivalent to one frame of video (vector of size 128), and later we will use a longer audio segment. The concatenated vector gets feed into the generator, which produces a fake face landmark (vector of size 140). The discriminator/critic will then try to distinguish the fake landmark from the ground truth landmark by delivering a score.

Regarding the architecture, the Generator consists of two hidden linear layers activated by the LeakyReLU function, and the output is activated by the Tanh function. The Discriminator consists of one hidden linear layer that is also activated by the LeakyReLU function and then linearly project onto one dimension for the score.

We adopted several training practices suggested by the original WGAN paper [12]. In particular, we update the critic model more times than the generator each iteration e.g. 5. Using different learning rates for the generator and discriminator should have a similar effect. Second, we use the RMSprop optimizer rather than the Adam optimizer. Third, we clip weights of discriminator between -0.01 and 0.01 after each update. Fourth, we keep the learning rate relatively slow e.g. 0.00005. And fifth, we use the LeakyRelU activation function rather than ReLU.

2.1.4 Face Alignment

One of the assumptions for our AI Face robot is that it always talks facing the audience. Nonetheless, one issue with the VoxCeleb2 dataset is that speakers changes head pose constantly, making the acquired 2D landmarks occasionally skewed towards one side. To compensate for the head pose change, we tested several methods to align the face back to the center. Face alignment by itself is a difficult problem. Many modern techniques such as DeepFace [14] frontalizes the face to a 3D model and then apply 3D transformation so the image appears as if the face is looking directly towards the camera. OpenFace [15] uses a simple Affine transformation, where 3 key points (left eye, index 36; right eye, index 45; and nose tip, index 33) and three coordinates (corresponding keypoint coordinates on the mean face) are chosen as fixed points, and then, for each face, a transformation is calculated such that the transformation will make the chosen key-points move to the chosen coordinates while preserving colinearity (all points lying on a line initially still lie on a line after transformation) and ratios of distances (the midpoint of a line segment remains the midpoint after transformation).

For example, consider the linear system in equation 2, where x, y are the key point coordinates before transformation and u, v are the target key points coordinate after the transformation as defined by the mean face. We can then solve for M by multiplying the inverse of the 3×3 matrix to the right-hand side. Then, given the full original face landmarks, we can apply the transformation M as shown in equation 3.

$$M \underbrace{\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}}_{3\times 3} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix} \tag{2}$$

$$M \underbrace{\begin{bmatrix} x_1 & x_2 & \dots & x_{70} \\ y_1 & y_2 & \dots & y_{70} \\ 1 & 1 & \dots & 1 \end{bmatrix}}_{3\times 70} = \begin{bmatrix} u_1 & u_2 & \dots & u_{70} \\ v_1 & v_2 & \dots & v_{70} \end{bmatrix} \tag{3}$$

To show the effect of the Affine alignment, we can plot the face distribution by overlaying the landmarks of some randomly sample faces and observe how it change after the transformation. See figure 4 and note the three keypoints are just single dots after the Affine transformation.

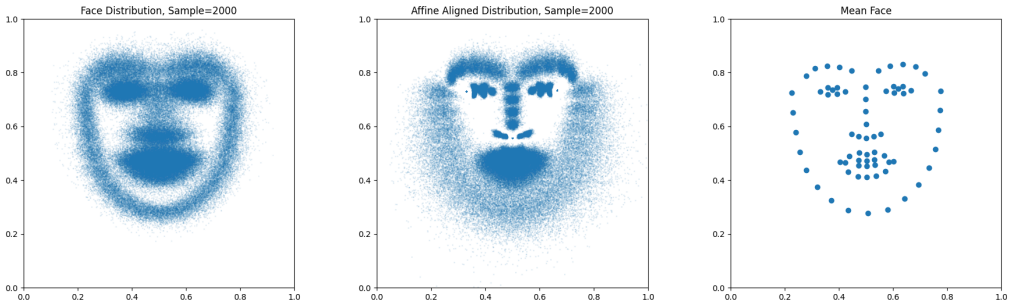


Figure 4: (left) Face distribution of 2000 randomly sampled original face landmarks. (mid) Face distribution of 2000 randomly sampled face landmarks after Affine transformation. (right) The mean face by averaging landmark coordinates over all 70 key-points. Used for key-point coordinate reference for Affine transformation

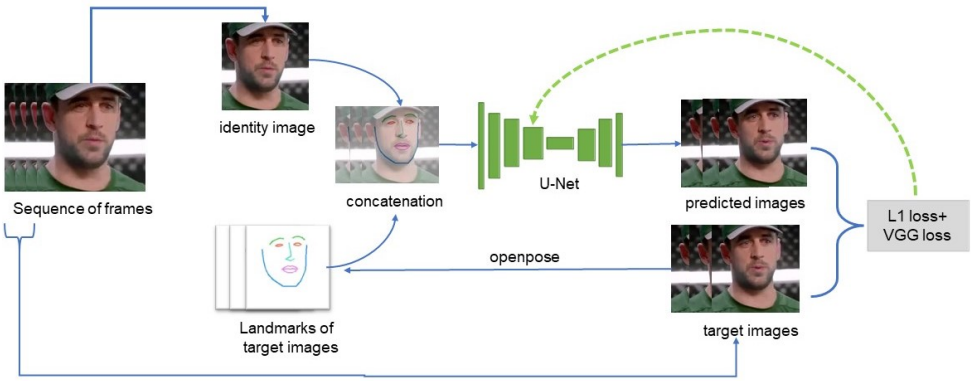


Figure 5: Full Pipeline of the Robot Face Generator

2.2 Robot Image Generator

The Robot image generator aims to generate the authentic robot face image based on the given face landmark information as well as the facial identity information. We use a U-Net architecture to learn the The full pipeline of the robot image generator can be seen in figure 5.

2.2.1 Landmark and Identify Image Processing

We use the the VoxCeleb2 dataset [9] and the Robot Face dataset collected by our lab for training of robot face generator. To translate the landmarks to target images based on identity images, we need to concatenate landmark information and the identity information as the input.

VoxCeleb2[9]

We transfer each of the video clip (an utterance) into sequences of images, and from each sequence of the images of 256*256*3 we select one as identity image and the rest as target images. We get the landmarks of the target images with OpenPose (to get data with good quality we remove the data with low landmark confidence score with OpenPose). The landmarks are recorded in dimension of 70*3 where each of the 70 rows represents a point of the face landmark and each of the 3 columns represents x-coordinate, y-coordinate, and confidence score repectively. Then we convert the landmark coordinates to images of 256*256*3 where the points are connected to draw the edge of the face. We concatenate the landmarks with the identity image into a matrix of 256*256*6 as the input and the corresponding target image of 256*256*3 as the ground truth for training.

Robot Face dataset

The robot face data set collected by the Creative Machines Lab consists of 16000 images of the robot faces with different facial movements taken at the same angle. For training, we use the sequences of hardwired facial movements to mimic the data collected from the sequences of muscle movements while talking. From the total 16000 robot face images we choose the first 1000 for validation, the middle 14000 for training, and the last 1000 for testing. We resize the robot face image of 420*580*3 into 210*290*3 for better training effect. From the training set we select one robot face image as the identity image and the rest as target images. We remove the 0-16 face landmark points (the jaw) and use the rest 5 points for training, as the jaw part doesn't work well on the normalization of the image. We removing the jaw part, we can approximately regard the face as a plane. Thus, without the first 17 landmark points, the landmark data is transferred from a 53*3 matrix into a 210*290*3 matrix. Then we concatenate the landmarks of the target images with the identity image into a 210*290*6 matrix as the input and use the corresponding 210*290*3 target images as the ground truth for training.

2.2.2 Model Architecture

Instead of using generative adversarial network model, we choose to apply an encoder-decoder structure for efficiency. At the first stage, we apply the baseline Encoder-Decoder architecture based on the model by [16] for training. The structure is a fully convolutional encoder and decoder architecture. Each layer of the encoder consists of two convolution layers and each layer of the decoder is a concatenation of a transposed convolutional layer and a convolutional layer followed by a transposed convolutional layer. For better training results, we change the baseline Encoder-Decoder architecture to the U-Net architecture [17]. The U-Net architecture is composed by a contracting path for capturing the image context and an expansive path for upsampling. The skip-connection of U-Net where feature map in each block of the extracting path will be concatenated to the corresponding block in the expansive path. It enables the combination of the features under different scales and improves the accuracy.

2.2.3 Loss function

The loss function here is for minimizing the pixel distance and the distance between the perceptual feature contents. For our predicted image $\hat{I}$ and the corresponding target image I, we apply the L1 loss and the VGG loss [18] which is based on the VGG feature map activation ϕ [19] and the L1 distance [7]. Instead of using both style loss and content loss from the VGG network, we choose the content loss only for our purpose. The content loss can well extract the low-level features of the images (e.g. contour) and the style loss extracts higher-level features (e.g. artistic style). After trying different loss in the Obama image over-fitting experiment (see 3.2.1), the combination of L1 loss and VGG loss proves to be the best.

$$Loss = 10 * L_1 + L_{vgg} \tag{4}$$

$$L_1 = \sum_{I,\hat{I}} ||I - \hat{I}||_1 \tag{5}$$

$$L_{vgg} = \sum_{I,\hat{I}} ||\phi(I) - \phi(\hat{I})||_1 \tag{6}$$

3 Experimental Results

3.1 Landmark Module Over-Fitting Experiment

We did two over-fitting experiments using part of the acquired data as a sanity check for our model architect design and implementation and also as a test regarding how well our model could perform

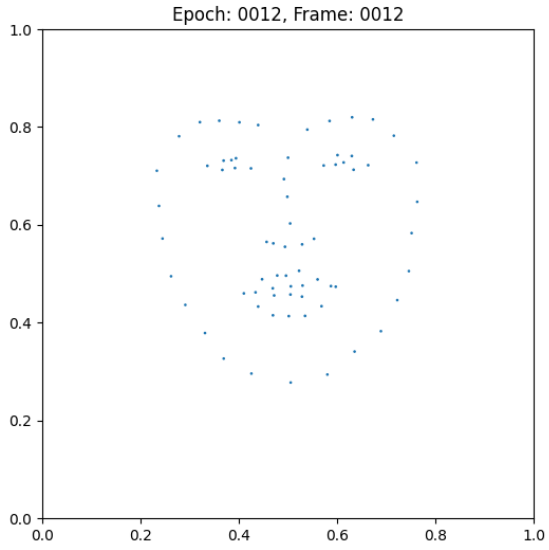


Figure 6: Generated face landmarks in episode 12 of the second over-fitting experiment using Affine transformed face images and noise vector of size 20. The landmarks then diverges abruptly.

at the simplest case. The first experiment was done over a single identity over just one frame of the video (1 training example), whereas the second experiment was done over 3000 identities using valid and high-confidence frames from over 6000 videos (214980 training examples).

Model evaluation method at the current stage still requires human judging the quality of generated landmark images. Before training, a uniform noise will be generated and an audio piece will be randomly selected. Both of them are stored as the fixed input. Throughout the course of training, the fixed input will be repetitively used for generating new face landmarks, which allow us to judge how the model quality changes over training time. With the goal of finding the best setup for 1. generating high quality landmarks, and 2. complete the first step quickly, our first over-fitting experiment could achieve model convergence model convergence in 3000 training steps (around 3 minutes training time) with the unbalanced training paradigm (5 updates of the discriminator followed by 1 update of the generator). We uses the same learning rate (0.00005) for both the generator and the discriminator. The training examples are not Affine aligned. A video visualizing how the generated face landmarks changes overtime by the fixed input can be seen here: Over-fitting experiment 1

Our second training experiment was performed on 214980 training examples, with the goal to 1. observe the quality of generated landmarks on single frame when training on a much more speakers, and 2. observe the quality of generated landmarks when providing a full audio. However, the second experimental goal is partly unfinished due to insufficient time left in the semester. As for the first goal, we noticed few interesting result: 1. The model could converge to a fairly decent quality in just 10 episodes (around 13 minutes of training time). 2. The model may still diverge quite abruptly even after it has achieved good quality. 3. The generated landmarks tend to look more like the mean face rather than the paired ground-truth landmark. A video visualizing how the generated face landmarks changes overtime by the fixed input can be seen here: Over-fitting experiment 2. Specifically, notice the generated images in the first half second of the video. Figure 6 shows the result of the generated image in episode 12. The training was done over Affine aligned examples, with generator learning rate 0.0001 and discriminator learning rate 0.0003. The noise vector has size 20.

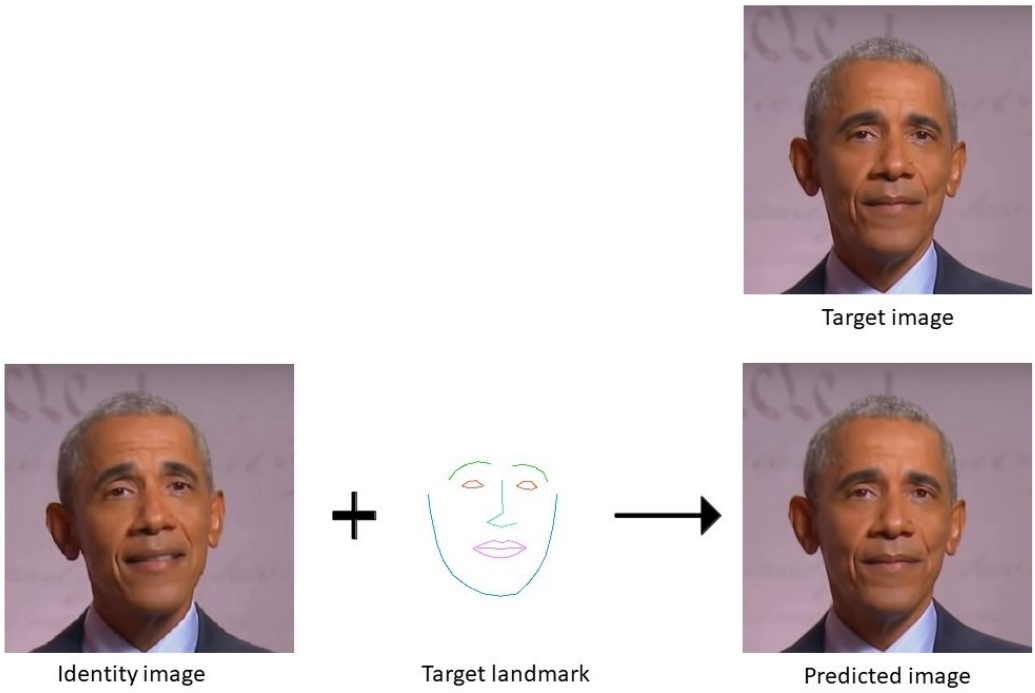


Figure 7: Obama Image Over-fitting Experiment

3.2 Landmark to Image Training Results

3.2.1 Obama Image Over-Fitting Experiment

For the first trial we apply an over-fitting experiment on a single portrait of Obama. We extract two image frames of 256*256*3 form an Obama speech video [20] and use them as the identity and target image respectively. The input is the channel-wise concatenation of the identity image and the landmark image extracted from the target image. The baseline Encoder-Decoder generator takes approximately 3min 21s to converge with a loss of 0.207 and the U-Net generator takes 10min 30s to converge with a loss of 0.090 7.

The baseline configuration is approximately 3 times more efficient than U-Net but with a much higher loss. Through zooming in the result images we notice the image generated by the baseline Encoder-Decoder have pixels in wrong color especially in the nose part (see 8). Thus we decide to use U-Net configuration for the following training. Through tuning the parameters of the learning rate and loss, we manage to deduct the training time of U-Net to 3min 30s (33.33% of the original training time).

We also experiment with different loss for the best training result. The model with both content loss and style loss takes longer time (30s) to converge with similar loss and output effect. Thus for efficiency we removes the style loss and only use the L1 loss and content loss based on VGG network for training.

3.2.2 VoxCeleb2 Training Result

We use a subset of the VoxCeleb2[9] for batch-training, which consists 204 clips with 29195 frames in total. We remove the frames with low confidence score and keep 18000 good samples for training. We first input clips from different speeches to train the model with different identity images at the same time. The training result doesn't appear to be good with a high loss value converging at 0.73, and the generated image as low fidelity9.

Figure 8: Compare of Results between Encoder-Decoder and U-Net

Figure 9: Results with Significantly Different Identity Inputs

We assume the reason might be the model isn't complicated enough to learn from significantly different identity information. For the second experiment we input the clips from the same speech, where the speaker's identity appears similar. However, we get a even higher loss value of 0.9510.

By visualizing the concatenation of landmarks and images, we found the potential reason: even though we get rid of the frames with low score, there are still many bad data remaining in the training set11. Since the VoxCeleb2[9] is an extremely large dataset with significant head movements of real human, the automatic landmark extraction through OpenPose might lack accuracy. We need to figure out a better way to detect the bad landmarks, as unlike the translation from audio to landmarks where the landmark information is stored as point coordinates, here we need to transfer the landmarks into face edges. This requires more focus on data cleaning. Considering generating the face image for the the robot face doesn't have the issue of bad landmark points (the data is collected by the lab with high resolution and high accuracy) and significantly different identity information (the robot face is of the same angle and identity information), experiments on these two problems will be our future steps.

Figure 10: Results with Similar Identity Inputs

Figure 11: Visualization of Bad Face Edge Concatenations

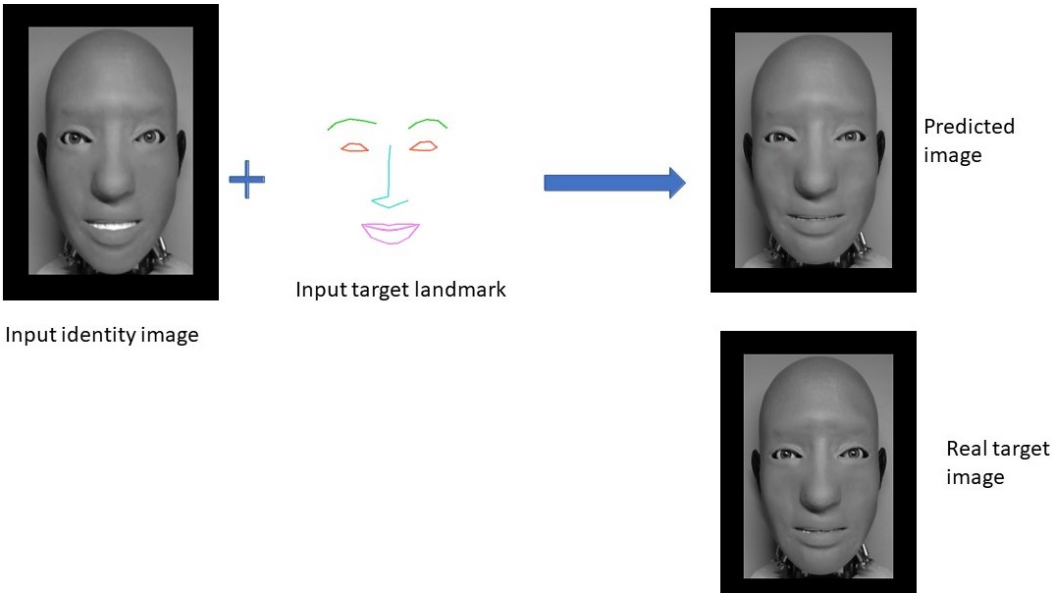


Figure 12: Training on the Robot Face Experiment

3.2.3 Robot Face Training Result

We finally train the model with the robot face data collected by the Creative Machines Lab. We select one robot face with relatively normal expression as the identity image and the rest 13999 images in the training set as the target images. Compared to the human face training on VoxCeleb2[9], the training on robot face dataset has s significant improvement on quality of the output, because of the fixed angle and identity information of the robot face, as well as the clean data. The model converges at a loss of 0.15 (20% of the loss of human data) and outputs images with high fidelity12.

We test the model with the 1000 images in the testing set and get a mean L1 loss of 0.0092 and a mean MSE loss of 0.0003. The compare between the part of the testing results and real target images are shown in figure 13.

From the testing result we can see the prediction with higher L1 and MSE loss is corresponding with a more exaggerated expression. The difference between the prediction and the real target image mainly concentrates at the degree of the eyelid opening and the detailed wrinkles14, as the eyelid movement is very trivial through the landmark and the wrinkles cannot be represented by the landmarks. For the wrinkles, we need to further experiment with the hardware group to see if such details will impact the inverse mapping to the motor driver. One potential way to better learn the

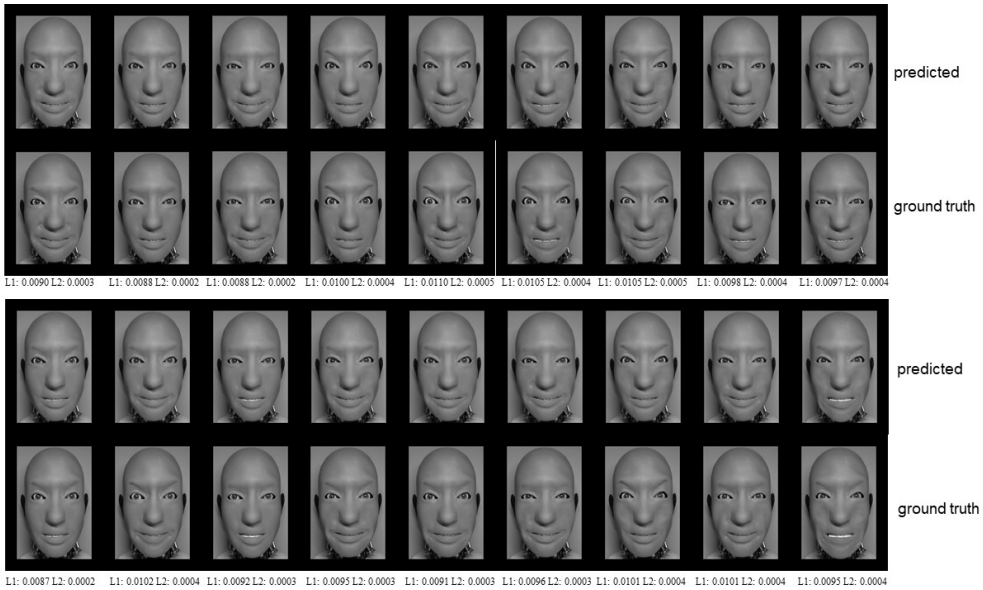


Figure 13: Testing Result with Loss

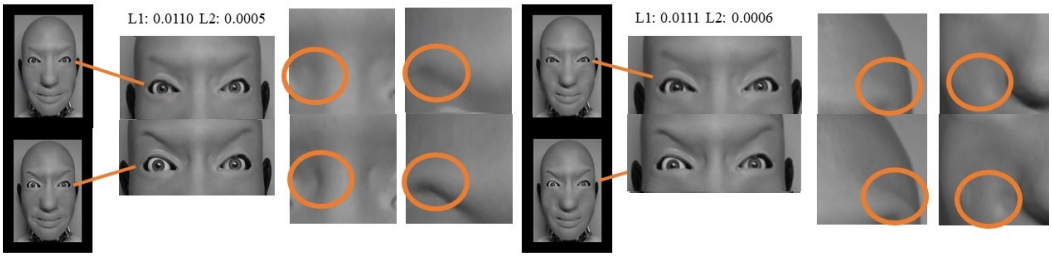


Figure 14: Detail Compare between the Prediction and the Target Image

details is to introduce the style loss for higher level of the image features. For the eyelid difference, we need to further experiment with the loss specifically focusing on that part.

4 Discussion and Conclusion

With structured processed training data for landmarks and audio Mel-spectrogram, Affine alignment, and our Conditional WGAN architecture, the current stage of the Landmark generator can create fairly high-quality human face landmarks with relatively fast training time. Nonetheless, several attempts/modifications could be made to potentially improve our model performance as well as open up new possibilities. First, even though most previous lip-sync models use a GAN architecture, most of them are also dealing with problems/tasks with generating. Nonetheless, our landmark generation task is relatively a simpler problem where only the coordinates of 70 points need to be generated. In our case, a simpler model architecture such as the variational auto-encoder could do the tasks at the same quality but with simpler training. Second, we currently use a frame to frame generation paradigm. However, to get smoother quality where audio info could be better reflected over a particular time span, we need to increase the input and/or output size by attempting segment-to-segment training. Third, even though we already have structured face landmarks and audio data for over 6000 videos it would still be a tiny portion of the full VoxCeleb2 dataset. More landmark acquisitions need to be done for more training examples.

The current stage of the landmark to image generator shows a relatively accurate prediction with the input robot landmark and the identity robot face image. Compared to the training result on the

robot image, the model shows an unsatisfied performance on the real human dataset with a loss of 0.73 (approximately 5 times the training loss of the robot face), majorly because of the messy data and the complexity of the variant human face features. Also, the model shows some weakness in generating t=detailed facial features such as the eyelid opening and the tiny wrinkles. For the next stage, we need to improve the model from three perspectives:

1. Dataset: For the implementation on human data, we need to find a better way to first clean the data of VoxCeleb2[9]. We can increase the threshold we set for the confidence score and remove the bad data based on the lines of the face edges converted from the landmark points. For the robot dataset, we can convert it to one channel data as the images are all converted to grey scale. In this way, we need to revise the model correspondingly to match the channels of the input and output data.

2. Accuracy improvement: The inaccurate prediction of the eyelid and wrinkles indicates that we need to further study the influence of detailed facial features on the inverse mapping to the motor driver. To achieve that we need to compare the accuracy of the hardware operation mapped from the prediction and the corresponding ground truth. We need to revise the loss function to see if the style loss will largely improve the model's ability of learning high level details of the robot face, since we only compare the style loss and content loss in the over-fitting experiment. We need to design a loss specifically focused on the eyelid movement (e.g. a pixel-distance loss concentrating on the eyelid area).

3. Model design: We haven't experimented on the whole architecture with the audio to the landmark module and the landmark to image module combined together. In our design the landmark to image module will take the output of the audio to the landmark module as the input. This means we need to align the output of the audio to landmark module to the identity image from the landmark to image module. We need to add an alignment layer and further experiment on the result of connecting these two modules.

References

[1] Y. Zhou, Z. Xu, C. Landreth, E. Kalogerakis, S. Maji, and K. Singh. Visemenet: Audio-driven animator-centric speech animation. *arXiv preprint arXic:1805.09488*, 2018.

[2] T. T. Nguyen, C. M. Nguyen, D. T. Nguyen, D. T. Nguyen, and S. Nahavandi. Deep learning for deepfakes creation and detection: A survey. *arXiv preprint arXiv:1909.11573*, 2019.

[3] K.-G. Oh, C.-Y. Jung, Y.-G. Lee, and S.-J. Kim. Real-time lip synchronization between text-to-speech (tts) system and robot mouth. In *19th International symposium in robot and human interactive communication*, pages 620–625. IEEE, 2010.

[4] K. Prajwal, R. Mukhopadhyay, V. P. Namboodiri, and C. Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 484–492, 2020.

[5] A. Lahiri, V. Kwatra, C. Frueh, J. Lewis, and C. Bregler. Lipsync3d: Data-efficient learning of personalized 3d talking faces from video using pose and lighting normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2755–2764, 2021.

[6] Y. Song, J. Zhu, D. Li, X. Wang, and H. Qi. Talking face generation by conditional recurrent adversarial network. *arXiv preprint arXic:1804.04786*, 2019.

[7] Y. Zhou, X. Han, E. Shechtman, J. Echevarria, E. Kalogerakis, and D. Li. Makeittalk: Speaker-aware talking-head animation. *arXiv preprint arXic:2004.12992*, 2020.

[8] B. Chen, Y. Hu, L. Li, S. Cummings, and H. Lipson. Smile like you mean it: Driving animatronic robotic face with learned models. *arXiv preprint arXiv:2105.12724*, 2021.

[9] J. S. Chung, A. Nagrani, and A. Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[10] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[11] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017.

[12] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[13] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[15] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[16] B. Chen, M. Chiquier, H. Lipson, and C. Vondrick. The boombox: Visual reconstruction from acoustic vibrations. *arXiv preprint arXiv:2105.08052*, 2021.

[17] O. Ronneberger, P. Fischer, and T. Brox. Convolutional networks for biomedical image segmentation. *arXiv preprint arXic:1505.04597*, 2015.

[18] K. Simonyan and A. Zisserman. Makeittalk: Speaker-aware talking-head animation. *arXiv preprint arXic:1409.1556*, 2014.

[19] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *arXiv preprint arXic:1603.08155*, 2016.

[20] N. News. Watch barack obama's full speech at the 2020 dnc — nbc news, 2020. URL https://www.youtube.com/watch?v=bps3m4eFTuE.

5 Who did What

Yunzhe Wang

- Design and development of the Landmark Generator Module
  - Landmarks Acquisition with VoxCeleb2 and Openpose
  - Developed a fully automated script to deploy our landmark acquisition tasks on Google Colab for faster processing.
  - Data cleaning and index alignment to remove ill-recognized face landmarks and low confidence landmarks with fitted Gaussian curves. Structured data are stored on Google Drive for easy access.
  - Audio Data acquisition in Mel-Spectrogram form. Aligning timing to the corresponding video in frame by frame precision.
  - Face landmarks alignment using simple rotate-shift-scale method and Affine alignment method
  - Designed and implemented training pipeline with our conditional WGAN architecture. Code can be accessed from Bo's Github repository.
- Training Experiments of the Landmark Generator
  - First training experiment and evaluation of generated landmark quality over just one speaker instance
  - Second training experiment and evaluation of generated landmark quality over more than 3000 speakers.
  - Hyper-parameter turning to find the best model setup. Can be find in the 'config' folder of code repository.

Yingke Wang

- Design and development of the Robot Image Generator Module
  - Transfer the landmark point coordinates into face edges and extract image frames into matrices from original video clips of the VoxCeleb2.
  - Data cleaning to remove the frames with low confidence score in each utterance clips and concatenate the selected identity images with target face edge images paired with corresponding target images.
  - Transfer the robot landmark point coordinates to face edges, extract the image information into matrices, and concatenate the identity robot face with the face edge paired with corresponding target robot face for training.
  - Separate the whole pipeline into two modules and design the landmark to image translation module with compare and experiments on two configurations: baseline Encoder-Decoder and U-Net.
- Training/Over-fitting Experiment of the Robot Image Generator Module
  - Compare the accuracy of the result and the training efficiency between different configurations (baseline Encoder-Decoder and U-Net) and different loss functions (style loss and content loss based on the VGG network).
  - Fine tune the hyper parameters of the model to improve the efficiency by three times (33.33% of the training time)
- Training Experiment on VoxCeleb2
  - Train the model with significantly different identity information from different speeches and compare it with the training results of similar identity information from the same speech.

  - Analyze the potential reasons for high loss value and find the problematic bad data samples in the training set.
- Training Experiment on Robot Face Dataset
  - Train the model with the robot face dataset collected by our lab and compare it with the training result of real human dataset.
  - Test the model with data samples in the testing set and record the L1 and MSE loss statistics of the testing results.
  - Quantatively and qualificatively analyze the testing results with higher loss and lower similarity with the target image and give potential reasons and solutions.