Single Audio to Inferred Animation

PROGRESS REPORT 3

COMP 400

Kevin Junyang Cui (260984715)

supervised by

Joseph Vybihal

McGill University

School of Computer Science

August 16, 2023

ABSTRACT

The virtual avatar, Virtual YouTuber, or *VTuber*, is a phenomenon in recent cultural spheres in which an entertainer's motion or likeness is represented entirely by an animated character, either in 3D or 2D (e.g. *Live2D*[1]), for the purpose of both real-time streaming and animation playback. While the primary method for high-fidelity real-time motion-tracking has been via either video (such as with computer vision), infrared camera, or wearable devices, these methods prove to be a challenge both in terms of affordability and accessibility. Meanwhile, alternatives such as keyboard-based interfaces impose limitations on precision and practicality. Audio or speech-based tracking has become available in some tools such as *VTube Studio*, which employ rudimentary acoustic analysis for the purpose of optimising lip-synchronisation in addition to video facial tracking, based on attributes of the inputted microphone audio such as volume and phonemes detected in the frequency domain.[4]

Novel approaches such as *AlterEcho* have attempted to further infer movement such as non-verbal behaviour from audio input, including text-to-speech and acoustic analysis (which map to a repertoire of gestures).[8] This allows not only for a more affordable, accessible interface with no additional cognitive effort on the subject's part, but also opens the door to a more efficiently encoded input and added personalisation of the output via manually inputted static parameters through loosened coupling between the subject and the avatar. However, such novel methods are limited by factors such as language spoken and granularity of personalisation.

An extension to the conclusion yielded from AlterEcho proposes to infer overall body language for full-body animation from single audio speech by means of recurrent neural network, specifically bidirectional LSTM. The outcome of this experiment yields a model that produces a vague resemblance in motion of the torso of the subject trained on when presented with single audio speech.

PROJECT DESCRIPTION

0.1 PURPOSE & DESCRIPTION

While animation of the mouth can be easily inferred directly from acoustic analysis as in VTube Studio[4], the same is yet to be said for full-body animation. This project attempts to infer non-verbal behaviour in the torso for animation from a single audio input as an extension to AlterEcho. By empirically training on different architectures and hyperparameters, a model is found that produces an output that vaguely infers the body language in the torso of a subject as a virtual character, which can be theoretically used in complement to VTube Studio or AlterEcho's software.

0.2 WHAT WAS ACTUALLY BUILT

The project uses a bidirectional stacked LSTM with an attention layer to predict a set of blendshape weights and bone coordinates from a single audio speech recording that can be directly converted into 3D humanoid animation in a custom client via VMC protocol.

0.3 WHAT DOES IT DO

The model can produce a vague prediction for a single subject. The animation produced from the neural network pipeline can be directly played in the created client where there is a visible correlation between the movement of the torso and the inputted audio from just a qualitative assessment. Quantitatively, it has one of the lowest validation losses out of all the models with different hyperparameters and architectures trained, though it is not in a stage in which it is ready to compare against other solutions or real motion capture.

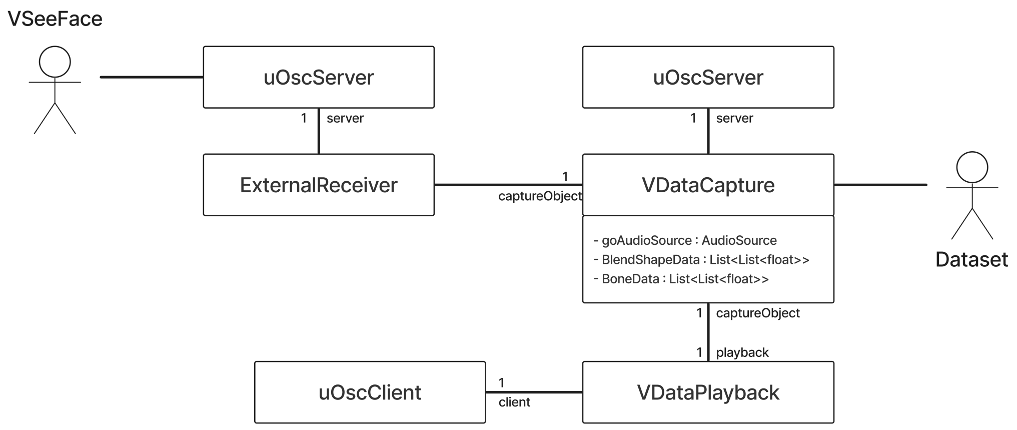


Figure 1: Domain model of interface. Data is captured locally via VMC protocol from a third-party motion capture VRM software such as VSeeFace by the uOscServer attached to the ExternalReceiver object, which feeds the data to a VDataCapture object that stores the data. The data can then be fed to the VDataPlayback object which sends it to the uOscServer attached to the VDataCapture via a separate port through its uOscClient.

0.4 DETAILED PROJECT DESCRIPTION

0.4.1 VMC Protocol

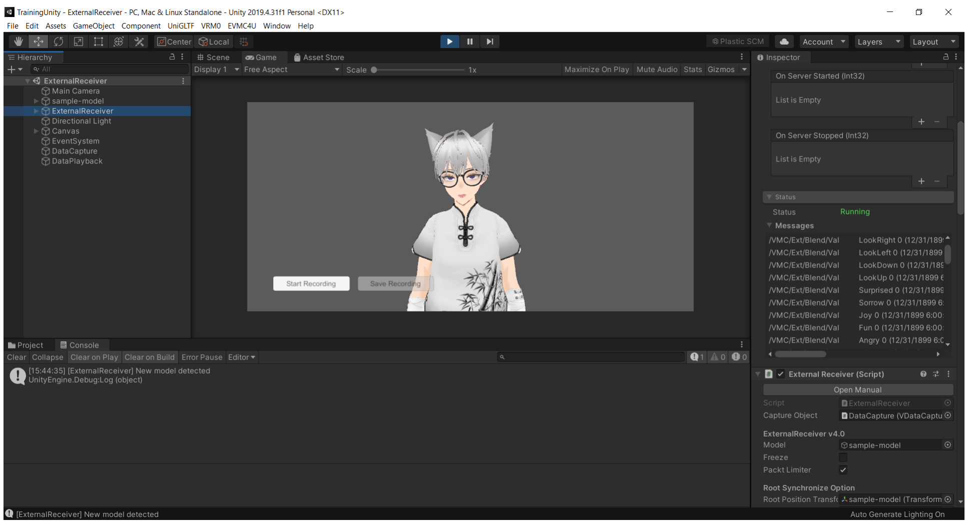
An interface was developed in Unity built from based on the VMC protocol for the capture and playback of blendshape and bone positional/rotational data. VMC protocol, or **Virtual Motion Capture Protocol**, is a protocol built from Open Sound Control (OSC) protocol and VRM for the communication of motion capture data, especially for avatar or virtual motion capture.

VMC data is received via Unity OSC server from an existing motion capture client such as VSeeFace (which uses OpenSeeFace) as such:

- **Bone Transform**
  /VMC/Ext/Bone/Pos (string)name (float)p.x (float)p.y (float)p.z (float)q.x (float)q.y (float)q.z (float)q.w

- **VRM BlendShapeProxyValue**
  /VMC/Ext/Blend/Val (string)name (float)value
  /VMC/Ext/Blend/Apply

and subsequently exported to CSV as a 396-dimension dataset (18 blendshape weights and 54 bones with a 3-tuple positional vector and 4-tuple rotational quaternion each, or $18+54(3+4)$) mapped to timestamps of a recorded audio at each call to */VMC/Ext/Blend/Apply*.



TrainingUnity - ExternalReceiver - PC, Mac & Linux Standalone - Unity 2019.4.31f1 Personal <DX11>

```
ExternalReceiver
  Main Camera
  sample-model
  ExternalReceiver
  Directional Light
  Canvas
  EventSystem
  DataCapture
  DataPlayback
```

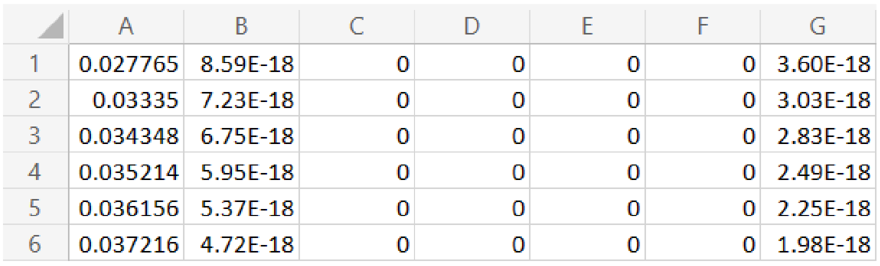
```
[15:44:35] [ExternalReceiver] New model detected
UnityEngine.Debug:Log (object)
```

```
On Server Started (Int32)
  List is Empty
On Server Stopped (Int32)
  List is Empty
Status
  Status        Running
  Messages
  /VMC/Ext/Blend/Val    LookRight 0 (12/31/189
  /VMC/Ext/Blend/Val    LookLeft 0 (12/31/1899
  /VMC/Ext/Blend/Val    LookDown 0 (12/31/18
  /VMC/Ext/Blend/Val    LookUp 0 (12/31/1899 6
  /VMC/Ext/Blend/Val    Surprised 0 (12/31/1899
  /VMC/Ext/Blend/Val    Sorrow 0 (12/31/1899 6:
  /VMC/Ext/Blend/Val    Joy 0 (12/31/1899 6:00
  /VMC/Ext/Blend/Val    Fun 0 (12/31/1899 6:00
  /VMC/Ext/Blend/Val    Angry 0 (12/31/1899 6:0
External Receiver (Script)
  Script                ExternalReceiver
  Capture Object        DataCapture (VDataCaptu
ExternalReceiver v4.0
  Model                 sample-model
  Freeze
  Packt Limiter         ✓
Root Synchronize Option
  Root Position Transfc sample-model (Transform
```

[ExternalReceiver] New model detected

Auto Generate Lighting On

Figure 2: Interface using stock VRM model for VMC capture/playback



	A	B	C	D	E	F	G
1	0.027765	8.59E-18	0	0	0	0	3.60E-18
2	0.03335	7.23E-18	0	0	0	0	3.03E-18
3	0.034348	6.75E-18	0	0	0	0	2.83E-18
4	0.035214	5.95E-18	0	0	0	0	2.49E-18
5	0.036156	5.37E-18	0	0	0	0	2.25E-18
6	0.037216	4.72E-18	0	0	0	0	1.98E-18

Figure 3: Sample snippet from captured dataset

An OSC client is also implemented in Unity to convert the dataset back to scheduled VMC calls to replicate playback of captured or predicted values, synchronised to an audio player.

0.4.2 Audio Feature Extraction

It can be hypothesised that the most relevant features in audio for animation (assuming speech in an arbitrary language) will be phonetic and tonal. That is, all features would be derivable from the Mel-frequency cepstrum coefficients (MFCC) of the target audio.[7][9] It can be noted that LPC is another possible avenue of feature extraction, but is observably inferior.[9]

Furthermore, as the model may be working with live production audio, it may be beneficial to implement some noise reduction filtration to isolate vocals. A libsox effect can be applied if applicable

0.4.3 Dataset & Data Loader

The data is imported from CSV to a Torch dataset where each VMC window is loaded with the corresponding time frame from the previous time of capture to the current time

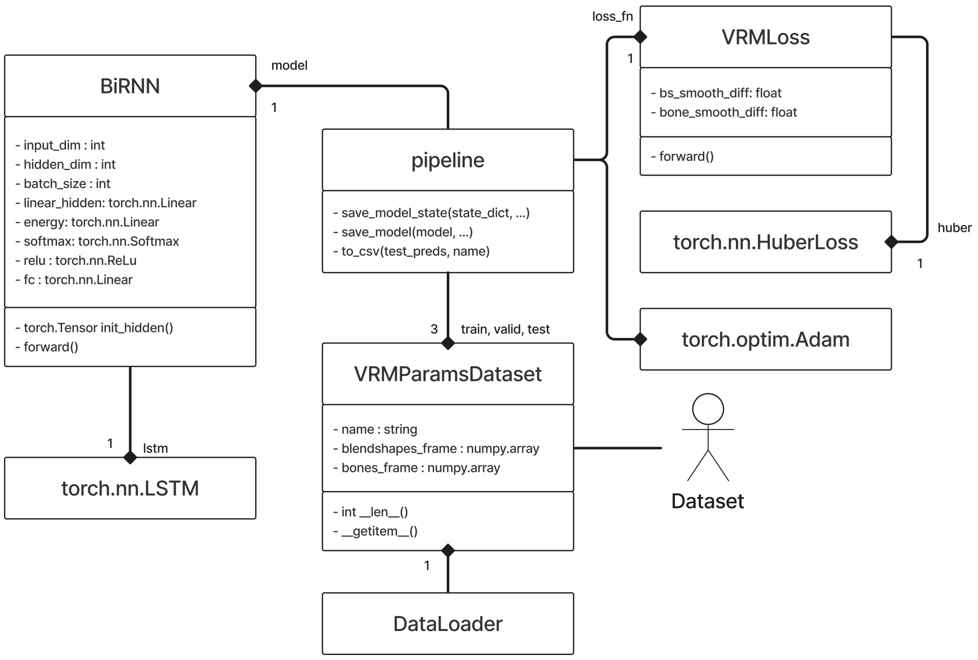


Figure 4: Domain model of LSTM training pipeline. BiRNN is trained on a DataLoader containing a VRMParamsDataSet with Adam optimizer[6] and VRMLoss, which uses HuberLoss.

of capture. The time frame between VMC capture points is set to a static value to little detriment to the data, as each VMC window can remain for several frames of audio. The training data consists of 30 minutes of mono audio at a 48000 sample rate for a total of 86400000 frames and 95449 VRM captures of 396 animation parameters, each set to a static interval of approximately 0.01886s or 905 frames. Of the 396 parameters, 12 are trimmed out due to irrelevance, leaving 384 parameters with 6 blendshape weights and 54 bones of 7 orientational values. These 12 are manually set to 0 in the final output (except for the **Neutral** blendshape weight which is set to 1). The remaining data is normalized to prevent the network from prioritising values with larger quantities such as bone coordinates as opposed to blendshape weights.

A short-term Fourier transform is computed on the discrete time window of every 56 frames, computed as $\lfloor 905/16 \rfloor = 56$ where 905 is the number of frames per VRM capture, such that there are 16 FT frames between every capture, or 32 frames per capture with 2x overlap or 16 frame past and future samples.[9]

From each fourier transform frame, 39 MFCCs are computed. The dataset is processed in a data loader with a batch size of 100 which returns each VRM capture with its $39 \times 64 = 2496$ corresponding MFCCs. Two other datasets and data loaders are also initialised as the the validation and test data, each with 1 minute of recorded audio and

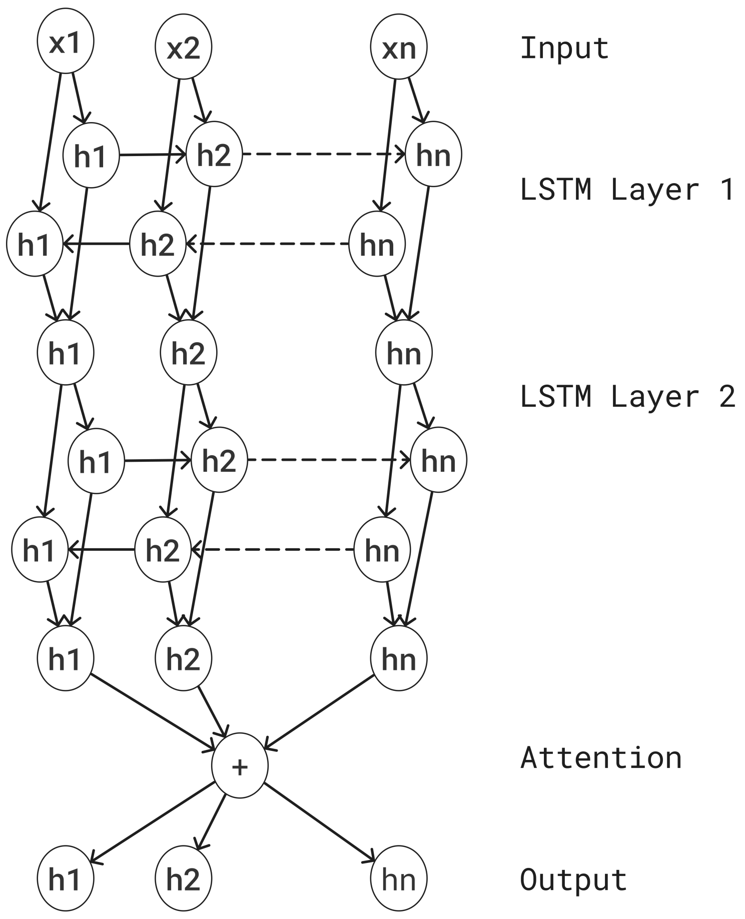


Figure 5: Architecture of Bidirectional LSTM with attention mechanism

VRM. Each set is recorded by the same actor, giving different monologues.

0.4.4 Training & Testing

A bidirectional long short-term memory (LSTM) network with an embedded attention mechanism has been shown to be effective in training for audio to facial animation, as it gives the network memory of past input audio features.[9] Therefore, it would follow that a similar model may be effective in inferring full-body animations. The LSTM has an input layer of 2496 corresponding to the MFCC of a single VRM capture along with two hidden layers each with a dimension of 512. This results in a 1024-dimensional output (due to the bidirectionality of the LSTM) which is fed into an attention layer, helping the network pay attention to relevant features. The output layer connects this to a 384-dimensional output representing the 384 blendshape weights and bone coordinates of the given audio window, which can be concatenated as a sequence and saved as a sequence of VMC calls.

The loss function, inspired from Audio2Face's approach[9] taking into consideration unit selection in concatenative speech[5], is a variation of the loss function containing two loss functions:

$$L = L_A(w_1^S, w_2^S, S) + L_A(w_1^B, w_2^B, B)$$

where w^S and w^B are static weights of the blendshapes and bones respectively and S is the input corresponding to blendshapes and B is the input corresponding to bones and

$$L_A = \frac{1}{n}\sum_{i=1}^{n} w_1 L_t(y_i, y_i^p) + \frac{1}{n}\sum_{i=2}^{n} w_2 L_s(y_{i-1}^p, y_i^p)$$

where y is the desired output and y^p is the output produced by the network, w_1 and w_2 are the weights of the loss functions, and L_t and L_s are the target Huber loss function and smooth "concatenative cost" loss function respectively, where

$$L_t(y_i, y_i^p) = L_\delta(y_i, y_i^p) = \begin{cases} \frac{1}{2}(y_i - y_i^p)^2 & \text{for } |y_i - y_i^p| \leq \delta \\ \delta|y_i - y_i^p| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

$$L_s(y_i^p, y_j^p) = |\theta - cos(y_i^p, y_j^p)|$$

where δ and θ are adjustable hyperparameters, with $\delta = 1$ and $0 \leq 0\theta \leq 1$ for this model. A Huber loss function is less sensitive to outliers in a dataset which may contain noise and converges faster than mean square error (MSE) loss and mean absolute error (MAE) loss to the minimum. L_s is a "concatenative cost" that ensures temporal smoothness between concatenated frames.[9][5] The absolute difference of this cost from θ is computed as a means to prevent the model from staying still, as motion from speech can be ambiguous and a $\theta = 0$ would prioritise lack of movement over movement, while movement can be assumed to be relatively continuous.

The hyperparameters of the model can be adjusted empirically, with the model quantitatively evaluated over the validation set using the same loss function or a different function such as the root mean squared error (RMSE) of parameters over all frames in a held-out test set.[9]

$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^p)^2}$$

The model is run on 50 epochs with the best model according to the validation loss selected for prediction.

0.4.5 Prediction

Once trained, the output of the model given some target audio input would be a 384-dimensional tensor which is inverse normalized into a set of VRM motion parameters. These are combined with the 12 static parameters and saved as a CSV file that can be played back on a VRM model via the same client. Additional physics and shading are applied automatically by the Unity game engine.

0.5 TECHNOLOGY STACK

The neural network is configured in PyTorch with gradients computed using the Adam optimiser[6]. The pipeline runs on CUDA with an RTX 2060. Version control is handled by Git, at the repository `https://github.com/kevinjycui/VSpeakMotion`.

The environment for the client is the Unity game engine using C# with Visual Studio Code. Unity plugins used include VRM4U, UniGLTF, uOSC, and a reverse engineered version of the Unity plugin EVMC4U.

```
VRMPlaybackClient
├─ Assets
│  ├─ EVMC4U
│  │  ├─ ExternalReceiver.cs
│  │  └─ ExternalController.cs
│  ├─ Prefabs
│  │  ├─ DataCapture.prefab
│  │  └─ DataPlayback.prefab
│  ├─ Scripts
│  │  ├─ SavCsv.cs
│  │  ├─ SavWav.cs
│  │  ├─ VDataCapture.cs
│  │  └─ VDataPlayback.cs
│  ├─ uOSC
│  │  └─ Runtime
│  │     ├─ uOscClient.cs
│  │     └─ uOscServer.cs
│  ├─ UniGLTF
│  ├─ VRM
│  ├─ VRMShaders
│  ├─ sample-model.prefab
│  └─ sample-model.vrm
├─ Packages
└─ ProjectSettings
```

Figure 6: Directory structure of important files in client

```
SingleAudio2InferredAnimation
├─ sampledata
│  ├─ Predictions
│  └─ Training
│     ├─ Audio
│     ├─ Blendshapes
│     └─ Bones
├─ pipeline.ipynb
└─ requirements.txt
```

Figure 7: Directory structure of important files in pipeline

FINAL REPORT ON DELIVERABLES

Table 1: Final status of deliverables from report 2

Date Completed	**Description of Task**
15 June	Develop interface and recorder for training dataset
22 June	Develop animation engine and base lip-sync
30 June	Collect data and create dataset of audio and parameters
6 July	Configure runnable base MFCC extraction and data loader
20 July	Configure trainable LSTM using loss function and optimiser
3 August	Train usable model for audio to animation inference
17 August	Train model which passes established test case(s)
19 August	Final project and report

TEST CASES

0.6 TESTING

The model is tested on various hyperparameters, notably the weights of the loss function, number of layers in the LSTM, number of epochs, number of features extracted etc. It is trained and tested against the test dataset, which has no overlap with the training or validation sets, and run in the client.

0.7 RESULTS

Using the loss function as described previously, the model was able to achieve a validation loss of 2.3446098566 at epoch 38, which was the minimum out of 50 epochs. On the test dataset, the model achieved an RMSE of 0.0818010792 and produced a prediction. At a qualititative level, the prediction appears to have some correlation to the audio in the movement of the torso, though further testing with longer, more precise training data may improve results. The model is also able to make similar predictions for other actors not in the training set to varying degrees of quality. However, neither the blendshapes nor bones move significantly, leaving the model to be fairly still throughout. When tested on less epochs (e.g. 10 epochs), the character moves sporadically, especially in the rotation of the eyes (which are bone coordinates). When tested on more epochs (e.g. 50 epochs), the character does not visibly move.

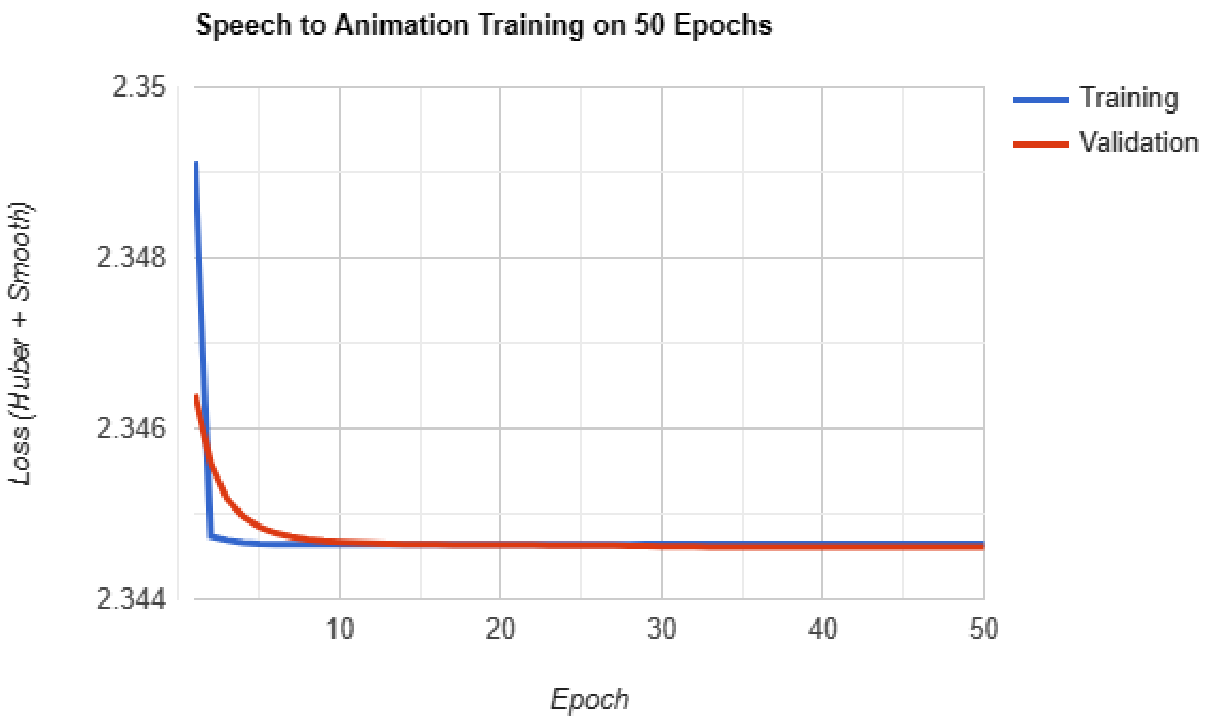


Figure 8: Training and validation loss of model on 50 epochs.

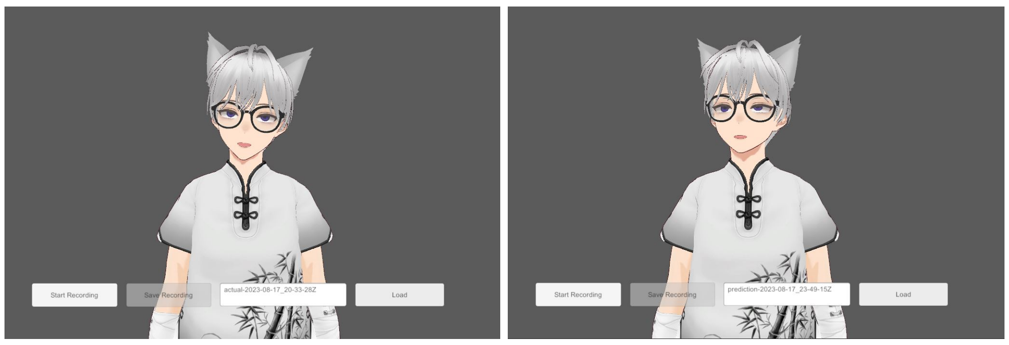


Figure 9: Left: Actual animation. Right: Predicted animation. Due to the ambiguity of speech to motion, the quantitative difference between the two is inconsistent.

PERSONAL REFLECTION

Research and testing was done efficiently, with the client and overall pipeline constructed in a timely manner. Tools used, such as Jupyter Notebook and Unity were easy to grasp and effective for their respective uses. Order of deliverables were successful logistically, as finishing the client completely before the pipeline proved useful in better understanding the output. However, further research in improvements beyond the architecture as inspired by Audio2Face for the specific case of speech to animation was not completed. It would be beneficial to begin research on such at an earlier stage. More focus on creating better training data as opposed to tweaking less and less significant hyperparameters on the model would have also been beneficial, as training data quality proved to be a bottleneck in output quality. It would have been beneficial to produce longer training data, and emphasised further on preprocessing feature extraction.

0.8 FUTURE WORK

It can be noted that the biggest issue in both the blendshape weights and bone coordinates is the lack of movement overall, in that motion is very slight and tends to be homogeneous for any input. As there is an inherent ambiguity in motion from speech, it would be beneficial to instead have the LSTM predict a sequence of means and standard deviations that could be fed into a Mixture Density Network (MDN)[3] to produce a probability density function on the values of the VRM parameters. This would allow for a degree of change in the sequence that would be more robust than the θ value in the custom smoothing loss function.

Another issue is the quality of the training data. Since the data is recorded as VRM parameters in VMC protocol, given the correct hardware such as a LeapMotion device[2], it would be easy to also capture arm and hand motion to the precision of individual fingers. Furthermore, the length of the recording and the diversity in actors can be increased as well to probable improvement of the model. This could also generalise the data, as some overly prevalent global variables tend to cause the model to fit to an unexpected pattern (e.g. if the actor tends to face left or right throughout their monologue, the model will cause the prediction to face that direction. If two datasets are combined in which the actor faces one way in one and another way in the other, the model will cause the prediction to switch back and forth between the two directions).

REFERENCES

[1] Live2d cubism. https://www.live2d.com/en/. Accessed 2023-05-30.

[2] Ultraleap. https://www.ultraleap.com/. Accessed 2023-08-18.

[3] Christopher M. Bishop. Mixture density networks, February 1994.

[4] DenchiSoft. Vtube studio settings. https://github.com/DenchiSoft/VTubeStudio/wiki/VTube-Studio-Settings, July 2022. Accessed 2023-05-30.

[5] Andrew J. Hunt and Alan W. Black. Unit selection in a concatenative speech synthesis system using a large speech database, May 1996.

[6] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, December 2014.

[7] R Raja Subramanian, Yalla Sireesha, Yalla Satya Praveen Kumar Reddy, Tavva Bindamrutha, Mekala Harika, and R. Raja Sudharsan. Audio emotion recognition by deep neural networks and machine learning algorithms, October 2021.

[8] Man To Tang, Victor Long Zhu, and Voicu Popescu. Alterecho: Loose avatar-streamer coupling for expressive vtubing, October 2021.

[9] Guanzhong Tian, Yi Yuan, and Yong Liu. Audio2face: Generating speech/face animation from single audio with attention-based bidirectional lstm networks, July 2019.