Utrecht University

Department of Information and Computing Science

FaceDiffuser

Speech-Driven Facial Animation Synthesis Using Diffusion

Author:

Ștefan Stan

2176327

Supervisor:
Dr. Z. (Zerrin) Yumak

Daily Supervisor:
K.I. (Kazi) Haque MSc

Second examiner:
Prof. dr. E.L. (Egon) van den Broek

Acknowledgements

I would like to thank my supervisors, Dr. Zerrin Yumak and Kazi Haque, for their invaluable guidance, support and feedback. Working under their mentorship has been an absolute pleasure. My thanks to Prof. dr. Egon van den Broek who showed interest in this thesis and agreed to evaluate it.

Additionally, I would like to thank Michel Maasdijk, who facilitated my access to the computational resources of the Department of Information and Computing Science, allowing me to do a lot more experiments than I would have been able to do on my laptop alone.

I am also grateful for my family, and for the constant love and support they showed me. Thanks to all my amazing friends, both old and new, and to all the incredible people I met in The Netherlands.

Abstract

Speech-driven facial animation synthesis has been a notable area of research in recent years, with new state-of-the-art approaches constantly emerging. Deep-learning techniques have demonstrated remarkable results for this task, clearly outperforming procedural methods. However, we notice a relevant scarcity of methods for generating rigged character facial animations that could be seamlessly integrated into animation pipelines. There is also a lack of non-deterministic methods that can produce a high variety of animations. In this paper, we present FaceDiffuser, a deep-learning model able to generate expressive and diverse facial animation sequences based on speech audio input. To the best of our knowledge, we are the first to employ the diffusion mechanism for the task of 3D facial animation synthesis, leveraging their non-deterministic nature for a better and more expressive facial animation generation. We use a pre-trained large speech representation model, HuBERT, to extract speech features, as it has proven to be effective even in noisy audio cases, making our model more robust to noisy settings. We show that our model is robust to noisy audio and can be used to animate 3D vertex facial meshes as well as rigged characters.

We utilise 4D facial scan datasets as well as datasets containing rigged character animations, such as our in-house dataset UUDaMM, along with the recently released BEAT blendshape-based dataset. The results are assessed using both subjective and objective metrics in comparison to state-of-the-art results as well as the ground truth data. We show that our model performs objectively better than state-of-the-art techniques, our model producing lower lip vertex error than the competitors. In terms of the qualitative evaluation, we show by means of a user study, that our model clearly outperforms one of the state-of-the-art methods, while being rated similarly or slightly worse than the other 2 competitors. Furthermore, ablation of the diffusion component shows better performance over a variant of the model that does not use diffusion, strengthening our intuition over the benefits of the diffusion process.

Contents

1 Introduction

Figure 1: Series of frames of a mesh animated with FaceDiffuser

Believable facial animations are essential to many fields in the entertainment industry including but not limited to video games, animated movies, or even the computer-generated imagery (CGI) used in big-budget productions. The characters transmit emotions through their facial expressions, the animation quality being essential for the experience of the viewer. This, however, can be a costly and time-consuming process. Because of this, finding ways to automate it is an attractive field of research. Specifically, we have seen solutions for gesture generation, hand movement animation, and speech-driven facial animation.

Animating 3D characters is often a labouring process that falls in the hands of experienced animators who must manually pose the characters for each frame of an animation sequence. While this workflow enables a high level of control and ensures quality results, the process of completing the animations is tedious and requires hours of work and a great level of expertise for maybe just seconds of animation. A good example in this regard is the Walt Disney Animation Studios which in the 100 years since its beginnings has produced a total of 61 animated feature films. [1] While some of those films are 2D and do not require the creation of 3D models, the most recent releases are frequently computer-rendered 3D animations. Nearly all of the character animations produced by the studio need the intervention of experienced artists, either through key-framed animations, or post-processed motion-captured animations. This means that for every frame of a movie, an artist manually needs to adjust the characters in the scene. For the latest production at the moment of writing this thesis, Strange World, this means a total of $128,349$ [2] frames that needed creation. We may confidently assume that character animation alone takes a large amount of time out of the 3-5 year average animation movie production period.

Another way of acquiring facial animation is through performance-driven methods, where a performer is required to act out a scene while wearing motion and facial capture equipment that will translate the actions and expressions of the actor onto a virtual 3D character. This method has the benefit of obtaining more realistic human-like results as the performance of the actor can be translated directly onto the target character. Furthermore, it reduces the work of animators, who now just need to post-process the recorded sequences instead of creating them from scratch. This method, however, is not ideal and relies on several processes that need supervision to ensure the quality of the outcomes. Moreover, the high cost of the equipment makes this method usable only by big studios that could afford the expense.

Another difficulty encountered in animation production, especially in the case of movies targeted at a younger audience, is represented by the fact that dubbing a movie in multiple languages requires extra effort from the whole team and not just from the actors. New speech sequences will thus require new facial animations, in particular, new mouth animations that need to align with the new sounds that are being made by a character. While the best results would be obtained by the artists going back to the studio to regenerate the new animations, the process would take time and consume additional resources. Even in the case of facial capture, the actor would still need to reenact the scenes in different languages, therefore introducing new risks for errors if the target languages are not known by the actor.

As we can see, none of the aforementioned solutions is ideal, as each requires trade-offs between quality and cost. We consider that a solution to this problem, and many others, would be to use an automatic language-independent facial animation generator. By employing such a model, new

[1] https://disneyanimation.com/films/
[2] https://disneyanimation.com/process/

animations could be obtained fast and with minimal cost and ideally with minimal intervention from the artists. Furthermore, if the model can generate animations encoded as a series of facial rig parameters values or blendshapes, these could be integrated into animation workflows, enabling artists to tweak and adjust the results to their liking.

In recent years, plenty of studies have been focusing on automating this task and generating facial animation by employing different modalities such as audio, text, emotion or others. Some methods are based on procedural approaches [16, 55] which define sets of rules to drive character rigs. While robust, these methods are deterministic and will result in repetitive animations, failing to generate new and emotive expressions. Video manipulation and editing is another area of research that can benefit from facial animation synthesis. To this end, several approaches take on the task of animating 2D talking heads [60, 22, 50, 36]. While not the focus of our research, we consider these approaches to be pertinent for our research, mainly because of the speech modalities used for generating animations.

Deep-learning approaches are the most popular ones at the moment, as we can see from the works that aim to animate a 3D character that is either represented by a rig [62, 3, 9, 48] or a vertex point-cloud [44, 19, 12, 8]. Several modalities are considered the driving inputs for the animations, the most common being audio speech and text. However, it is necessary to note that these methods are inherently data-hungry and the quality of the results heavily relies on the quality of the training data itself. Most of the frequently used datasets, such as the ones collected by *Fanelly et al* [20] and *Cudeiro et al* [12], consist of a limited number of subjects and spoken sentences. While the authors of the datasets claim that the sentences were carefully chosen to capture a full range of available English phonemes, we argue that this is not enough to capture the full spectrum of facial expressions a person can make. We can therefore argue that for the generation of expressive and realistic animations, the ground truth animations must be themselves expressive.

To overcome the limitations of existing datasets, and to be able to learn as much as possible from the ground truth data, we propose an approach that includes the use of diffusion models. Diffusion models have shown remarkable results in the fields of content generation, in particular, we have recently seen the surge of "AI-generated" images all over social media. Most of those images are a result of the use of diffusion models, which can generate novel and realistic content. We leverage the non-deterministic nature of the diffusion process to help our deep-learning model to generate more diverse and new animations. Since facial expressions have a high variety, the same emotion being able to be conveyed in many different ways on a person's face and also across distinct people, we consider it critical for our model to capture the latent expressions space of the ground truth and generate new expressions each time it is inferred, while also maintaining accurate lip-sync across different generations. Furthermore, to tackle the small size of these datasets, we employ a pre-trained self-supervised language model, HuBERT [27]. This model represents the state-of-the-art in terms of language representations and has proven robustness in effectively encoding speech features, even when the input sequences are of low quality. By utilising this complex model as our speech encoder, we ensure the extraction of relevant features from the input audio speech. This, in turn, will aid our model in learning to translate the input speech features into accurate lip shapes.

Animating full faces based on speech input is a one-to-many problem, the upper part of the face, in particular, being only weakly correlated with the speech signal. Due to this fact, the previously proposed approaches often suffer from the problem of over-smooth results with little to no motion in the upper part of the face. This is caused by the model learning to generate an average expression and converging to it. Because of the many-to-many mapping between inputs and outputs, the model will try to minimise its defined loss by always outputting a smooth and mostly static animation rather than exploring a higher range of outputs.

Solutions for this issue were proposed by [44, 57], who both use a way of learning the latent space of facial expressions from the dataset to be able to later add it to the generated animations. They do this by making use of a 2 stage training approach, first employing an autoencoder, which is then used in the second stage. The first stage of the training is inherently a one-to-one learning goal since the model needs to learn how to correctly encode, and then decode the actual animation frames. In doing so, they ensure that the decoder, which is then frozen and employed in the second stage, learns a higher range of motions. While they have delivered remarkable results in terms of more expressive facial poses, with CodeTalker being the current state-of-the-art for this specific task, they still have the problem of being completely deterministic, CodeTalker, in particular, showing very low diversity in the generated animations. As a consequence of their deterministic character, once the models are trained, the same animations will be generated for a specific audio signal, essentially turning them into methods that one-to-one map the audio signal to the face. We consider this to not be ideal since there are facial movements, such as eye blinks, that are independent of speech or time constraints. Such facial motions could occur at any point during a sequence increasing the realism and naturalness of the results. Because of this, we would ideally like to replicate a similar outcome with our generative model.

To address this issue, we harness the power of diffusion models and include them in our model with the goal of generating more diverse animations with a distribution of motions closer to that of the ground truth. Based on the recent developments and fascinating results obtained by utilising diffusion models, such as novel and hyper-realistic images, we aim to integrate them into our model and analyse the potential benefits of such an architecture for a facial animation pipeline.

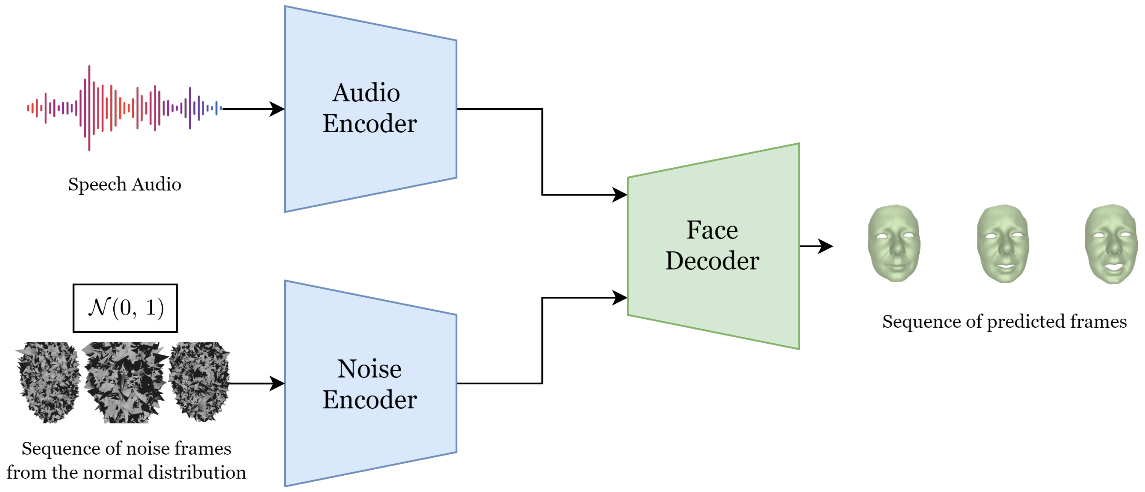

Audio Encoder
Speech Audio
Face Decoder
Sequence of predicted frames
$\mathcal{N}(0, 1)$
Noise Encoder
Sequence of noise frames from the normal distribution

Figure 2: High-Level overview of FaceDiffuser. An end-to-end model that generates facial animation based on only audio speech and randomly sampled noise

By utilising diffusion, we aim to address the one-to-one mapping these other approaches are facing and generate non-deterministic animations. While doing this, we also keep in mind that the generated animations need to be accurate in terms of lip-sync. A high-level overview of our approach can be seen in Figure 2, where 2 encoders are used for extracting relevant features from the audio input and the noised ground truth animation, which are then used for the facial decoder that outputs the generated animations. The audio input will condition the generation, ensuring that accurate lip movements are predicted, while the diffusion part will ensure the diversity of the possible results.

To the best of our knowledge, at the moment of writing this thesis, no proposed method employs diffusion models for 3D vertex motion synthesis. Probably closest to our research is the gesture generation field, which generates bone motions based on joint positions and rotations. Based on the success of these models [13, 49, 51], seeing more expressive and diverse animation sequences can be obtained by introducing diffusion into the training loop. By doing so, we expect to produce more expressive animations that would ideally not have the problem of over-smoothing the results and will include more motion in the upper part of the face.

To achieve this goal we make use of both rigged character-based datasets [37, 34] as well as 3D vertex datasets [20, 12, 56]. By following previous research and employing widely used 3D face datasets, such as BIWI [20] and VOCASET [12], we aim to place our work in relation to state-of-the-art developments. With the addition of the Multiface dataset, we additionally aspire to show that our proposed method is robust to different facial topologies. Secondly, by using rig-based datasets we seek to demonstrate that our method has the potential of being integrated into animation pipelines and that the resulting animations can be transferable across different characters. We show that such animations are easily editable in modern rendering engines such as Blender and that a wide range of characters can be animated with our approach, provided they have defined the 51 ARKit blendshapes.

The key contributions of this project can be summarised as follows:

1. **Incorporating diffusion into a facial animation synthesis deep learning model to address the one-to-many mapping between speech and expression.** With the help of

diffusion, we are capable of generating more expressive animations which are closer to the ground truth distribution. Furthermore, we show that our approach tackles the issue of over-smoothing and generates a lot more movement in the upper part of the face. Finally, we show that our model learns a stronger style embedding, the animations produced with different styles being more diverse than the ones of other state-of-the-art methods.

2. **Employing a pre-trained speech representation model to capture the particularities of speech.** By employing the HuBERT model, we effectively encode a large variety of speech features, which will help the decoder to make more accurate predictions of the character's expression.

3. **Developing a deep-learning algorithm for animating language-independent facial expressions for a rigged character driven by speech.** We propose an extra version of our model with only a few changes in terms of architecture that can be trained and used for generating facial animations encoded as rig control or blendshape values. With this, we show the potential for our model to be integrated into character animation workflows.

1.1 Research Questions

By taking into account the previous work along with the data we have available for training we aim to answer the following main research questions:

- Does integrating diffusion models into a speech-driven animation pipeline improve the resulting animations?
- Do our results beat state-of-the-art techniques?
- Can we develop a similar model that generates speech-driven animations for rigged characters?

Therefore, the main goal of this project is to make use of diffusion models for generating facial animations for 3D vertex mesh characters and analyse their benefits through objective and subjective metrics. Furthermore, we compare our results to those obtained by other state-of-the-art techniques, allowing us to position our work. Secondly, we want to develop a model that is capable of generating animations that would drive a rigged character. We consider that animating characters based on facial controls or blendshape values instead of vertex displacements would allow our method to be integrated into animation pipelines, therefore facilitating the work of animators. Furthermore, we consider that those types of animations would be more easily transferable between different characters.

1.2 Report Roadmap

Based on these ideas, we will expand and provide a more in-depth look into our chosen approach during the rest of the report, which is structured as follows. In the next section (Section 2), we present the state-of-the-art concerning speech-driven facial animation synthesis. We start with a quick history of the different methods of automating the facial animation process and continue by presenting works more similar to our approach. In Section 3, we give more details of the theory behind the methods employed in our approach. We provide a high-level overview of audio encoding and machine learning techniques as well as present the diffusion process. Next, we define the questions and sub-questions we aim to answer with our contribution in Section 1.1.

In the second half of this report, we begin by presenting in detail the datasets used to train our models Section 4 and continue in Section 5 with the methodology for developing and evaluating our model. In Section 7, we evaluate our results based on objective and subjective metrics. Finally, we discuss the possible directions in which our research can be expanded in the future in Section 10 and summarise our results along with a brief discussion of the key findings in Section 11.

2 Related Work

In this section, we present an overview of the different methods used for generating audio-based facial animation. To begin, we provide a summary of the modalities that are employed in the task of animating virtual humans, focusing on audio-driven methods.

We explore methods based on 3D point-cloud vertex character models along with ones based on rigged characters. Furthermore, we highlight the work done in the field of talking-head generation.

We compare different deep-learning approaches for facial animation synthesis and use the outcomes to guide our decision for the finalised architecture. Finally, we present some recent solutions using diffusion models for various generative tasks. To the best of our knowledge, there does not yet exist a method for generating 3D facial animations that makes use of diffusion. To this end, we take inspiration from the closest fields to facial animation generation, namely gesture generation and talking head synthesis.

2.1 Driving Modalities

Text and audio are the most used inputs when it comes to generating facial animation. Because of that, it is important to choose appropriate methods to encode them and extract relevant features such that the model can learn to make good predictions.

Some previously proposed methods have attempted to incorporate additional modalities to drive the animations and improve the overall realism and accuracy of the generated results.

Richard et al. [43] combine audio and gaze features into a multi-modal deep learning model based on variational autoencoders (VAE) that will dynamically decide which parts of the face of a 3D codec avatar will be driven by each modality. The head pose is also relevant when creating realistic facial animation, head rotation and movement carrying significant non-verbal cues in conversations. This is especially true in the case of dyadic conversations where the listener can participate in the communication with verbal and non-verbal cues. For instance, nodding or shaking one's head can convey agreement or disagreement with the speaker while showing that one is actively listening and paying attention to the other person. In Learning2Listen[36] and "Let's face it" [29], the authors use head rotation as a modality, in addition to speech, to learn motions for a listener character based on the speaker. Both Meshtalk [44] and CodeTalker [57] propose a 2-stage training strategy that aims to create more expressive animations by first learning the latent space of expressions included in the training dataset. The success of their approach is based on using the ground truth visual frames in the first stage, ensuring that the animation decoder will learn a higher range of possible motion. By employing diffusion models, we essentially are doing a similar thing, but faster, considering our method is end-to-end. With the inclusion of the diffusion process, we are able to learn the latent space of facial motions, and then sample it during inference.

2.2 Speech-Driven 3D facial animation

2.2.1 Procedural methods

The earliest work for automatic facial animation involves designing a set of rules that can be used for finding the best mapping between phonemes and visemes. A phoneme is the smallest unit of speech in linguistics, encoding a specific sound that a person can make, whereas a viseme is the visual representation of this sound translated visually as the mouth shape a person makes when pronouncing that specific phoneme. Due to the fact that human speech is a complex process and most of it happens hidden in the throat, a one-to-one mapping between phonemes and visemes is impossible. The same viseme can be attributed to multiple sounds, and since people speak fast, during the flow of speech, visemes often overlap each other, creating new visual representations in the process. This is known as co-articulation, and together with the many-to-one mapping between phonemes and visemes, makes the task of automating speech animation to be a complicated one.

The problem of co-articulation was addressed by *Cohen et al.* [10] who proposed dominance functions as a solution for mapping the relationship between sequences of phonemes. This was later used as a base for the method proposed by *Albrecht et al.* [2] who define a pipeline for generating facial animation from textual inputs. Their system uses the MARY [3] text-to-speech (TTS) system to gener-

[3]http://mary.dfki.de/

ate audio speech from text. Dominance functions are then used to compute facial muscle parameters, which are synchronised with the audio and rendered into an animation.

There are multiple methods using text as input combined with a TTS system to synthesise 2D talking-head animations [23, 11]. By using a high-quality speech model, such as HuBERT [27], to extract and encode features from the audio, we consider that this use case is also possible for our method, which can produce accurate lip movements for all kinds of speech signals, including artificially generated ones. Furthermore, with the recent boom in generative models and the increased quality, we consider that animating a character could become a completely automated process, by combining multiple generative models. This could be especially useful for animating background characters or NPCs in video games.

A rule-based approach was proposed by *Maatman et al.* [35] to model the behaviour of a listening agent based on the speech and posture of the speaker. Although their list of rules is incomplete, their findings serve as a foundation for future research and demonstrate that listening behaviours are influenced not only by the interlocutor's speech but also by their gestures and expressions.

More recently *Wang et al.* [55] developed a method based on a set of rules for the most common sequences of phonemes that require co-articulation adjustments. While their model is not complete, since co-articulation might vary from subject to subject, they suggest that their method is easily customisable and allows for the introduction of additional rules into the system.

In 2016, *Edwards et al* introduce JALI [16], a rigged model based on jaw and lip anatomical actions that represent a solution for the multi-mapping between phonemes and visemes. Their method puts a focus on lip-synching animation. Realism and easy customisation represent goals for the generated results. Unlike previous methods, JALI uses a different parametrization of facial actions for visemes, enabling a more realistic and accurate representation of actual speech. The method is effective in producing animator-ready lip-synchronised frames from audio, with animators able to fine-tune the final results using animation curves.

2.2.2 2D Talking Heads

For talking head generation approaches, the task is usually to produce photo-realistic videos corresponding to an input and a target. Based on the driving source of the animation, these can be categorised into text-driven, audio-driven, video-driven, or a combination of multiple modalities.

This field of research has the advantage of large amounts of readily available data, using "in-the-wild" videos being common practice when it comes to 2D talking head animations. Nevertheless, since the algorithms are often based on pixel values rather than actual geometry, most of these methods do not apply well to 3D virtual characters. Despite this, we still consider these methods relevant for our research since some also take audio as input and need to extract relevant features from it in order to succeed in accurate lip-sync animation generation.

Moreover, some methods also make use of 3D features, by extracting this information from the videos. This has proved to give better-looking results, 3D information being important when movements, such as head rotation are to be rendered in the resulting videos. To this end, multiple proposed methods first use techniques for 3D head reconstruction, making use of state-of-the-art methods like DECA [21] and FLAME [33], which can build mostly accurate 3D head representations from even just one monocular image. Another option is extracting facial 2D facial landmarks from the video frames and using their coordinates as labels for training a deep-learning model [17, 18]. Using "in-the-wild" monocular videos aims to avoid the pitfall created by the lack of readily-available 3D datasets by utilising the large amount of data published on the internet with a large variety of speaking styles, spoken languages, conversation types, and so on. However, the quality of these results is bounded by the accuracy of the 3D reconstruction methods used, which cannot be perfect and might introduce biases in the training pipeline. Furthermore, in the case of animatable characters, additional steps would be required to transition from mesh to rig, thereby increasing the risk of errors straight from data collection. In contrast, we aim to avoid this limitation by using an in-house dataset created with a focus on 3D rigged characters. By using rigged 3D characters which can be posed by adjusting controls or blendshape weights, we significantly reduce the number of target parameters that need to be learned during training to drive the animations. Furthermore, by using a conventional blendshape representation such as the set of 52 ARKit [4] blendshapes used to record the facial animations in the

[4]https://developer.apple.com/documentation/arkit

BEAT dataset, we make sure that our method can be easily integrated into an animation pipeline for a variety of different characters, the only condition being to define the same blendshape expressions for the characters. We believe this to be a trivial step for an experienced artist, considering the number of facial animations that can be automatically obtained afterwards.

By taking into account the goal of these papers, we can also make the connection with this thesis, the difference laying mostly in the outputs. While these methods can produce photo-realistic 2D videos, we, in turn, want to be able to produce animations for 3D characters.

A method for editing talking head videos based on text inputs was proposed by *Fried et al.* [22]. Their novel method allows editing videos by changing sections of the annotated transcript. It is important to note that the suggested solution is not a learning problem, with deep learning employed only in the final phase to render the videos. However, by giving the algorithm both a video input and a transcript, a user can customise the animation by making little changes to the transcript, such as deleting words, changing the order of words, and adding new words. Their method, like some of the other procedural methods discussed, is based on a set of rules. In this instance, though, the rules are not pre-defined but rather learned from the original video. A completely audio-driven talking-head animation method is proposed by *Thies et al.* [50]. They employ a 3D head reconstruction latent space in their pipeline, their method being able to generalise across different speaking subjects. Furthermore, by including the 3D reconstruction phase, this method becomes more similar to our task of virtual character animation. A pre-trained audio encoding network is also used to extract audio features, namely, the pre-trained DeepSpeech [24] model. While the results for ”clean” data are of high quality, the proposed pipeline still has limitations when it comes to noisy audio or multi-party conversations. To the best of our knowledge, our chosen audio encoder, HuBERT, [27] does not have these limitations, proving state-of-the-art performance even in the case of very noisy audio.

Generative adversarial networks (GANs) have also proved to be successful in talking head generation, the end-to-end method proposed by *Vougiokas et al.* [53] being able to animate a still image based on audio input. To this end, 2 different discriminators are used: one that would tell if the generated image is real or not and the other focusing on the sequence of frames and discriminating between real and fake videos. In a follow-up work [54], they introduce a third discriminator that focuses on lip-audio synchronisation, to further improve the realism of the generated videos. While an ablation study suggests that the frame discriminator is the one that improves the resulting realism the most, the authors do notice an improvement in adding a synchronisation discriminator. Since we consider accurate lip-synch to be the most important part of the generated animations, we will also consider putting more weight on lip-synch loss. Additionally, we plan to use RNNs, which by their very nature will be able to establish temporal connections between subsequent frames of the animation, to prevent the potential of jittery results.

2.2.3 Deep Learning for 3D character animation

The methods presented in this subsection are closest to our project since they also employ deep-learning methods for animating 3D characters. *Karras et al* [30] aim to animate virtual characters by employing a relatively simple end-to-end deep learning architecture. They use just 5 minutes of performance-capture data from one subject and train a neural network comprised of multiple convolution layers. The data is divided into frames, so each audio frame input, obtained by extracting LPC features from the raw audio, will be used to predict the corresponding visual frame. Despite the lack of temporal awareness caused by this design, their method still produces acceptable results. We can attribute the apparent quality of the animations to the inclusion of a velocity loss which ensures a smoother transition between consecutive frames.

Taylor et al. [48] propose a deep-learning method based on audio and text inputs. Unlike the previous method, they do not extract audio features from audio but rather use phonetic labels, each frame in the training dataset being manually annotated with a specific phoneme based on the audio. By using a very compact and generalised face parametrization based on landmarks, their method proves to be also successful in animating unseen face models and even more stylised characters. A retargeting step is also included in the pipeline, making it suitable for rigged characters.

Zhou et al. [62] build on the successful JALI rig model introduced by *Edwards et al.* [16] and present Visemenet, a deep-learning approach that aims to be a better solution than the previously presented procedural method. This approach is based on the previous findings from JALI and focuses on producing editable animations. In doing so, they aim to introduce a useful tool for animators that could streamline the process of facial animation creation.

However, in most cases, deep-learning algorithms are data-hungry and require large amounts of quality data to produce good results. Thus the lack of readily-available datasets of audio-visual facial capture sequences represents a big problem in this field. To this end, some research efforts have also been directed toward gathering datasets for this specific task. *Cudeiro et al.* [12] introduce a new dataset of 29 minutes of speech collected from 12 subjects, along with a novel method for speech-driven facial animation generation. The dataset comprises 4D scans of the subjects' heads captured at 60 FPS, as well as 480 sequences of English speech, each lasting 3-4 seconds.

Due to a clear separation of the animation sequences by actor, they are able to introduce an identity encoding in their architecture, managing to separate speaking styles from the actual mesh, and making it possible to animate any facial mesh with the same topology as the meshes in the dataset. At inference time one of the training identifies can be chosen to condition the results, therefore allowing a certain diversity in the generated results. Despite the relatively high quality of the lip-sync animations generated by their method, the results look stiff and emotionless. Furthermore, at times, we can see some noisy results presenting themselves as jittery lip movements. This can be attributed to a lack of expressive animations in the dataset, upper face movements and eye blinkinkg being only scarcely available. To solve this issue, *Richard et al.* propose Meshtalk [44], which also uses a dataset of 4D facial scans but much larger than VOCA, amounting to a total of 250 subjects, each being associated with 50 sequences of speech. A categorical latent space is learned for disentangling speech-related expressions from non-audio-correlated expressions. Finally, their results are also improved by designing a loss that combines the errors for lip-synch, upper face movement and blinks.

By using deformation gradients to represent the motion for a 3D vertex character, *Chai et al* [8] manage to create a speaker-independent model that can better generalise to new facial topologies. They argue that deformation gradients are better than simple vertex offsets in synthesising distinctive local facial motions. Furthermore, they propose a lightweight method for speech encoding based on bidirectional LSTMs and spectral gathering. They do not manage to outperform state-of-the-art with this new speech-encoding, but the results are not much worse and succeed in proving that speech-encoding is an essential part of creating a robust facial animation pipeline. More recently, the HuBERT [27] model has proved to be very efficient as a speech encoder in an animation synthesis pipeline designed by the supervisors of this thesis *Haque et al.* [25] To this end, we also aim to integrate the HuBERT model into our architecture to encode the audio. The main difference between their approach and ours is constituted by the nature of the datasets. The previous method is used to animate 3D vertex characters, while we train our model on rigged characters. Furthermore, we aim to integrate a modality based on listener awareness into our architecture, the UUDaMM dataset containing dyadic conversations and allowing us to extract information about the listener's facial motion, as well as the listener's audio.

To tackle the problem of emotional animation, *Dahmani et al.* [14] create their training corpus by recording an actress reading 2000 sentences in a neutral tone, and an additional 500 of those in an emotional way. Textual data is used as input, while three conditional variational autoencoders (cVAE) are used to encode duration, acoustic and visual data. *Aylagas et al.* [3] propose a method that also uses conditional variational autoencoders. However, they use raw audio input and 3D mesh representations of a character along with their tongue. By including the tongue motion they argue that the results will look more realistic, the tongue being responsible for creating a majority of the visemes. Unlike previously presented work, they can generate speaker-independent animations without the need for conditioning on a specific identity. Furthermore, they include an additional step in their method, namely Mesh2Rig, that they train to translate the generated mesh and rig controls. Including the animation of the tongue, a perceptual study showed that the results were improved, with users preferring the characters that included tongue animations.

Recently, transformers have also gained popularity for the task of 3D character animation. Transformers leverage the attention mechanism to add context to each item in a sequence generation task, proving to sometimes be better than classic approaches of RNN. However, depending on the task they may be harder to train than their predecessors and based on the training technique might take longer to learn relevant information from the training data.

One of the methods based on transformers are Text2Gesture [5], which makes use of transformer-based architecture to generate emotive gestures for a 3D skeleton from text inputs. In terms of facial animation, *Fan et al.* have proposed [19]. By using a pre-trained speech model based on wav2vec [46] along with two biased attention mechanisms they manage to obtain results that surpass state-of-the-art in terms of realism. CodeTalker as well [57] has proposed a 2-stage learning technique, including a transformed-based VQ-VAE that learns the latent expression space which is then used in the second stage of training. In this thesis, we aim to show that we can produce quality results for this task with a more lightweight architecture based on GRU layers.

A transformer architecture was proposed as well in the task of blendshape animation generation by *Chen et al.* [9] who also use a transformer-based architecture but to animate a blendshape model instead of a point-cloud vertex model. To this end, they train a transformer-based model customised with an additional Mixture of Experts (MOE) layer, to learn to predict facial animations based on 32 blendshape controls. They find that increasing the number of audio features by combining MFCC with phonetic posteriograms (PPG) and prosody features improves the quality of the results. A user study suggests a clear preference for their results when compared to the state-of-the-art at the time LipSync3D [32]. Furthermore, by introducing the MOE layer in their architecture they notice a 17x speedup at inference time when compared to the Bidirectional LSTM method proposed by *Thies et al.* [50].

2.2.4 Dyadic Facial Animation

The generation of speaker-specific facial animation has seen extensive research in recent years, as seen from the number of works presented in the previous sections. However, the same cannot be said about dyadic facial animation, especially applied to 3D characters. The main problem in this research field lies in the scarcity of readily-available data that is suitable for such a task. As previously noted, many datasets do not account for multi-party discussions and instead focus on a single-speaking subject. Instead of generating animations based on a single input, they introduce a new modality, namely the expressions of the other participant in a conversation. If we try to find a similarity with our approach, we can identify the multi-input design of the methods. While they employ speaker audio and motion for listener motion generation, we make use of speaker audio and noised motion for animating the speaker themselves. We consider that including this extra modality will facilitate learning a broader range of facial motions. Furthermore, after training, the additional inp

A more complex, deep-learning method for generating facial expressions for one of the participants in a dyadic conversation, DyadGAN [28] was introduced by *Huang et al.*. Their method utilises the affective state of the other person, encoded from 2D landmarks processed from the frames of the videos. Employing a 2-step GAN architecture, first, a facial sketch based on landmarks associated with the expression of the listener is generated. After that, another GAN is used for translating a generated sketch representation to an actual photo-realistic image. Their method, however, has the disadvantage that it can only produce a singular frame, and using it for the video would not give good results since temporal factors were not accounted for when training. However, the authors revisit the method in a follow-up paper [38] in which they also introduce temporal constraints by feeding the network with previous frames from the listener. The results obtained by the authors show the existence of a strong correlation between the expressions of the speaker and the listener. It is therefore important to account for the interlocutor when producing animations for participants in a dialogue.

More recently, *Jonell et al.* have proposed a method for animating the listener actor in a conversation [29]. In their work, they use FLAME [33] parameterised avatars as the target characters for the generated animations. The MAHNOB Mimicry database [7] is used for training their model, which generates facial animations for a listener based on 3 modalities: 2 input audios from both the speaker and the listener, and the facial expressions of the interlocutor. They base their method on previous work for listener gesture generation [1], but instead focus on facial animation instead of body gestures. Another advantage is that their method is probabilistic and can consequently generate more realistic and variate expressions. To this end, they use normalising flows [39] which they claim have more advantages for this specific problem than GAN or VAE architectures previously used. While the results are not perfect and look jittery, they do manage to create somewhat accurate facial animation results for a listener. However, this was only appreciated positively by the users participating in the perceptual study when the audio was removed from the resulting videos, suggesting that the animations were not correlated with the audio, only relations between the speaker and listener animations being learned. The importance of the introduction of the interlocutor expression modality is also validated by the study, with users preferring correctly paired speaker-listener videos over randomly paired ones.

More recently, *Ng et al.* [36], reduce the number of modalities to 2 and propose a method that uses only the audio and expression from the speaker for listener animation generation, ignoring the audio of the listener. They employ a novel VQ-VAE model, being the first to use such a model in the task of motion synthesis. With this architecture, they learn a discrete latent space representation of listener motion that enables them to generate realistic animations for the passive participant in a conversation. Their results outperform other methods, managing to create smooth animations that are correlated with the speaker. This VQ-VAE approach is more recently exploited in the task of speaker animation, CodeTalker [57] showing state-of-the-art results by using the VQ-VAE facial autoencoder introduced by *Ng et al.* [36].

2.3 Diffusion models

The remarkable results obtained by employing diffusion models are undeniable, leading to the emergence of new methods for content generation in various fields. Most of these works focus on image data and employ diffusion for creating new images. This is made possible by learning a latent distribution of features from the training dataset through the diffusion process. By the model receiving a noised input and also the level of noise added to it, it will be able to learn the inverse function of denoising the input. After that, at inference time, pure random noise can be sent as input to the model, which will be able to find patterns it learned from the training data and generate new content. The sampling process in such generative models can be updated by introducing certain conditions during training, allowing for more control during inference. These conditions can be represented by one-hot-embeddings, text or even audio. In the case of our model, the condition is the speech itself, which will guide the generation to produce accurate lip movements.

Now looking at the field of animation generation itself, we can already see relevant works that put the benefits of diffusion models to good use.

Tevet et al. [49] propose Human Motion Diffusion Model, a model that can generate body animations based on text descriptions. They employ a transformer-based architecture and introduce the noised ground truth motions as an additional modality during training time. By doing this, they succeed in generating non-deterministic animations at inference time, since the noised animation modality can be different every time. Starting from the success of MDM, other works generate body animations based on audio instead of textual inputs [51, 58]. *Tseng et al.* train a model that can generate dance animations conditioned on music inputs, while *Yang et al.* use a similar approach for speech-driven gesture synthesis. Furthermore, the former work uses a style embedding that enables additional control during inference time. With this, we can also make the connection with our training, during which we use speech input, along with a style embedding determined by the different subjects in the dataset.

In terms of 3D facial animation, there has not yet been any development to successfully incorporate diffusion models, gesture generation being the closest task where we have seen this. Despite this, we see the potential of diffusion models and think that using them for facial animation is an inevitable step for research.

A talking-head speech-driven audio editing method is proposed by *Bigioi et al*[6]. By taking a template video as input along with a new speech segment, the authors generate new lip motions that accurately follow the target speech sequence. They introduce a diffusion mechanism into their network which learns to denoise videos in a frame-by-frame manner. By employing the diffusion mechanism, they can generate more believable lip shapes, outperforming the previous methods in the field. Their model is capable of generalising across different speaker identies and performs well even in the task of generating animations for new unseen subjects.

Closer to our task is DAE-Talker[15], which makes use of diffusion for generating talking head animation. They employ a diffusion autoencoder approach first introduced by *Preechakul et al.* [42] on images and extend it to generating videos. Different from the previous talking head method, they use a 2 stage training approach, first training a diffusion autoencoder that would learn the latent space of facial expressions from the training set. The first stage has no temporal awareness and only learns to reconstruct an image from its encoding and its noised representation. In the second stage, a transformer-based encoder-decoder architecture is used to encode the audio input and output frame embeddings. A particularity of this model is that instead of being trained on correct predictions of frames it is trained on correct predictions of frame embeddings. This constrains the second stage architecture to learning inputs that align with the embeddings learned during the first stage by the frame encoder. At inference time, the model trained in the first stage is used with the modification that the frame encoder is replaced by the network trained in the second stage. By doing this, the authors can generate higher quality videos, employing the decoder trained in the first step on ground truth frames. We have seen a similar approach in other works that do not use diffusion but have a 2-stage training, such as Meshtalk and Codetalker [44, 57]. In these works, learning a latent representation of the facial expressions played a relevant part in generating results with a higher level of perceived expensiveness. While not employing a 2-stage approach, we also aim to learn the latent ground truth expression space with the use of the diffusion process in an end-to-end deep-learning model.

2.4 Summary

In this section, we have seen a range of different approaches for generating facial animation based on speech and maybe even other modalities. Going by what the current best-performing methods are, we see a clear benefit of using data-driven deep-learning methods. Nevertheless, the quality of the results is bound by that of the training set, choosing an appropriate dataset with a large variety of entries being crucial in this task. However, this represents one of the main shortcomings in the field, datasets containing a wide range of speaking subjects being scarcely available. While this might be good enough for developing a method able to produce accurate lip movements, for the broader task of generating facial expressions it might not prove to be the best. We argue that specific expressions might be only captured in longer contexts and can develop over longer periods than just a few seconds captured in the widely employed datasets, BIWI [20] and VOCASET [12]. Furthermore, while we have seen huge interest in terms of generating 3D facial animation based on vertex displacements, these methods render to be character specific and not easy to transfer between different mesh topologies. Therefore, they render to be more useful in computer vision tasks, such as talking-head generation.

To this, end we propose an additional model configuration trained on a larger corpora of paired speech-animation sequences that aims to address these issues. Firstly, the 2 additional datasets we employ include longer sequences, participants being allowed to talk for longer and without a script. By doing this, the datasets include a more realistic expression space that our model can learn from. Secondly, to address the specificity problem, by using these blendshape-based datasets we effectively have the ability to generate animations that could be easily integrated into a regular animation workflow. As can be seen from our experiments, these animations can be added to render engines such as Maya or Blender in the form of keyframed sequences, an animator being then able to adjust the animation graphs to their liking.

3 Background Knowledge

In this section, we provide a more in-depth look into the necessary background knowledge needed for the completion and understanding of the project. Therefore, we start by presenting the audio and the different features used to represent it. We also present the HuBERT [27] model used in this thesis for audio encoding. Next, the theory behind rigged characters and blendshape posing and animation is illustrated. Finally, we explain how deep neural networks work, including some specific architectures that we will consider for the implementation of this project.

3.1 Audio

The goal of this thesis is to use speech audio as the main input modality driving the animations for a 3D character. While the animation can be driven by either vertex displacements or control rig values, the input audio is the same in both cases. Therefore it is essential to understand sound and extract relevant features so that the machine learning algorithm can learn to understand the different subtleties in human speech. Various sound embeddings are employed in the related research, going from extracting sound features in the form of manually crafted metrics, to more complex speech-language models which have already been trained on a large amount of data.

3.1.1 Mathematical features for audio representation

As stated by *Zhang et al.* [59], audio features can be categorised into physical and perceptual. While physical features are statistics and calculations that can be retrieved directly from the sound wave using formulae, perceptual features are more subjective and reflect sound characteristics that people can perceive. We continue by presenting some of these computable features that were successfully used in past work.

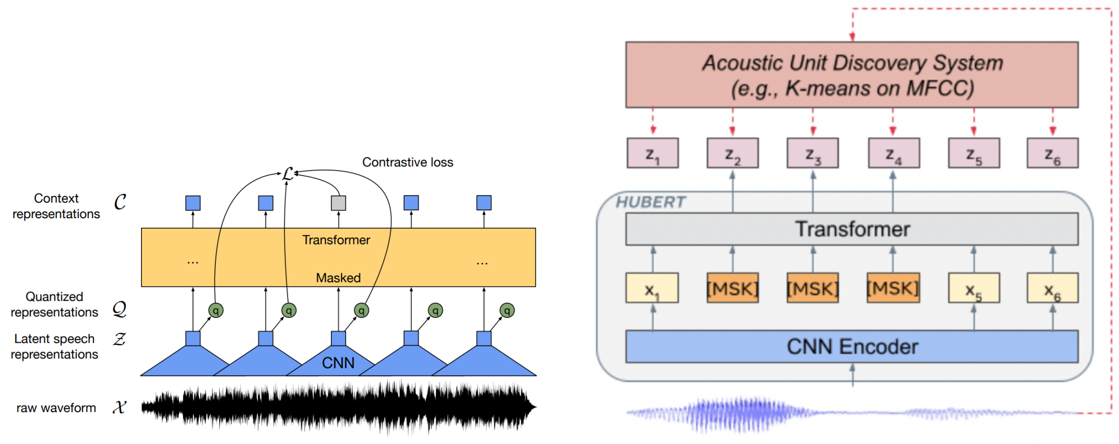
Prosody features are perceptual features and result from putting sounds together in human speech. They are usually not considered in studies since their computation is not as straightforward as other mathematical features that can be calculated from the audio signal parameters. Some of these features include the pitch, intonation, stress, and rhythm of sound. Such features were employed by facial animation research [9] together with transformers to further improve the expressiveness of the animations. Analysing their results from the accompanying videos, we can see that their model is able to produce accurate lip shapes and is robust to singing as well.

Mel-frequency cepstral coefficients (MFCC). To better understand these coefficients which are computable from sound, we first define the mel-scale. The mel-scale is a perceptual scale of sound pitches that is designed to accurately reflect the pitch distances as perceived by a human. It is formulated as a log function of the Hertz frequency representation, such that 1000 mels are equal to 1000 Hz. The mel name of this unit comes from the word melody, suggesting that this is a more subjective representation of the sound. MFCCs are a small set of features that together make up the mel-frequency cepstrum (MFC). The mel-frequency cepstrum itself is a representation of the short-term power spectrum of an audio signal.

Phonetic posteriograms (PPG). The phonetic posteriograms represent a mapping between sound and the previously defined phonemes. Therefore, if these features are successfully extracted, the speech sound will be represented as a series of phonemes together with the timestamp when the speaker pronounced each one. Considering the complexity of the problem, they are not easily extracted just by using some mathematical formula and it is usually a machine-learning task in itself as can be seen in previous work [61]. Therefore, for obtaining PPG features, the usual way to go is to use a previously trained network for automatic speech recognition (ASR). For example, *Zhao et al.* [61] implement a method for accent conversion, of which the first step is constituted by a network for PPG extraction.

Even though useful in multiple tasks involving audio, and even facial animation generation tasks, these mathematical features are not capable of encoding more specialised characteristics of the sound generated by speech. Relationships between different phonemes will therefore not be accurately represented, since semantic information is not accounted for when computing these features. We consider such semantic features to be crucial in the task of speech-driven facial animation, the long-term context being important to accurately depict expressiveness along with lip shapes.

To this end, we consider more complex methods for speech encoding and employ state-of-the-art methods in speech representation.


Contrastive loss
$\mathcal{L}$
Context representations
$\mathcal{C}$
Transformer
Masked
Quantized representations
$\mathcal{Q}$
Latent speech representations
$\mathcal{Z}$
CNN
raw waveform
$\mathcal{X}$
Acoustic Unit Discovery System (e.g., K-means on MFCC)
z_1 z_2 z_3 z_4 z_5 z_6
HUBERT
Transformer
x_1 [MSK] [MSK] [MSK] x_5 x_6
CNN Encoder

(a) wav2vec 2.0. Image source: [4]

(b) Hubert. Image source: [27]

Figure 3: Overview of large speech representation models architectures

3.1.2 Large speech models

In this section, we present some speech deep learning models that are often used in the task of automatic speech recognition. By making use of an already trained model that utilised a large corpus of diverse audio inputs, we make sure that our model can handle various speech settings as well as combat the data scarcity problem caused by the small size of our datasets.

DeepSpeech [24] is a state-of-the-art speech recognition method able to handle noisy environments with success. The model was initially developed for the task of translating audio to text, but since it is open-source, a pre-trained version of it was used for other various tasks. Since it was trained on a large corpus of data of paired audio and text sequences, it proves to be successful in the task of learning audio features from speech. Previous facial animation synthesis methods have used it as an encoder for raw audio signals, being integrated into a larger deep-learning architecture trained for generating 3D mesh facial animation [12].

Wav2Vec and Wav2Vec 2.0 [46, 4] The 2 versions released of the wav2vec model are speech recognition models used in various tasks like audio transcriptions. The architecture as it can be seen in Figure 3 is composed of a feature encoder component formed of multiple CNN layers. This component transforms the raw audio input signal into latent representations, that are then quantized and fed into the sequence of transformer layers. A difference between version 1.0 and 2.0 of wav2vec is that wav2vec 2.0 uses continuous speech representations instead of discrete ones and the self-attention mechanism captures dependencies across the entire audio sequence. Considering the large amount of data it was trained on, its pre-trained version can be used successfully in various tasks that require speech encoding. Another advantage of this is that the pre-trained model would be able to generate good speech representations even if the used dataset is very small as is the case of some of the datasets we use which contain only around 1 hour or less of speech sequences.

HuBERT [27] was designed specifically for the task of speech representation learning. It is a transformer-based model used for natural language processing. By using a masked prediction loss over speech segments, it can learn the sequential structure of speech and identify acoustic units. Besides acoustic models, HuBERT is also able to learn language models just from audio inputs.

As it can be seen from the model architectures of wav2vec 2.0 and HuBERT in Figure 3, they are very similar, both following pretty much the same general architecture, starting with a multi-layer CNN encoder and then several transformer layers that will generate the final acoustic representations. Some key difference between the 2 models is that HuBERT does not quantize the outputs of the CNN layers, and how the unit discovery is computed. For the latter, HuBERT uses simple k-means targets computed over MFCC features. To identify the different acoustic units, HuBERT uses a discovery system similar to performing K-means clustering on MFCC features. This will allow the network to learn discrete representations of phonemes and cluster them together accordingly. In the clustering step not just the correctness of the results is important, but also the consistency. Due to its flexibility and ability to adapt to even previously unheard languages, it is considered to be state-of-the-art in various

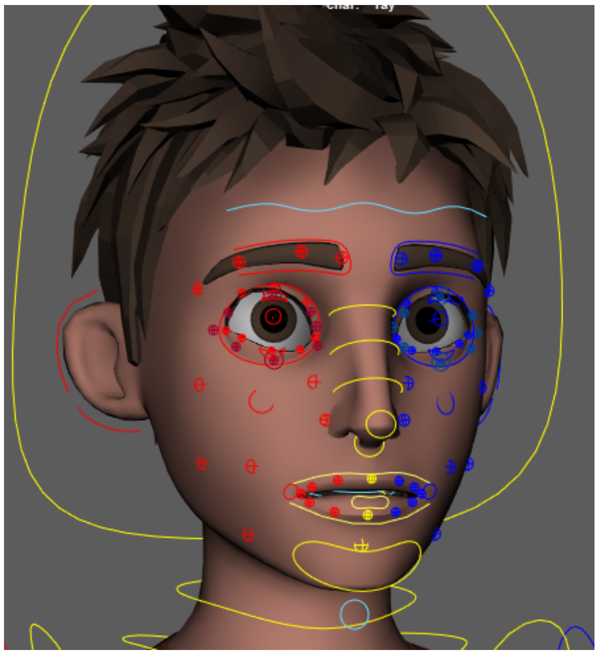


Figure 4: Ray model facial controls as seen in Maya

audio-based tasks. Since the model shows better performance than wav2vec, and it was previously successfully employed in the task of facial animation synthesis [25], we decide to also use it as the audio encoder in our model.

3.2 Facial animation for 3D characters

Animating virtual characters is a difficult task that typically requires the assistance of an experienced animator. Joints, blendshapes, or a combination of the two can be used to animate the face of a virtual character. In Figure 4 we can see the facial controls available for the Ray character that was used in the creation of the UUDaMM dataset. The character can be posed by individually selecting any of the control points and updating its position or rotation. This, in turn, will update the blendshape weights of the character, displaying the created expression. As noted by the authors of the dataset, only using blendshapes did not provide a full range of animation for the character used, actions like eye gaze or the mouth opening being animated by joints and not blendshapes. For this reason, instead of blendshape weights, joint controls were saved for every frame. Furthermore, using controls similar to those seen in Figure 4 makes animation much easier by allowing direct control over certain parts of the face. This is much more intuitive for an animator than manually updating blendshape weights.

A blendshape expression encodes an exaggerated facial pose that a character can make, a weight that takes values between 0 and 1 encoding the actual expression of the character. A weight of 0 would mean that the expression does not influence the final pose, while a weight of 1 would mean that the expression is maximally represented in the final pose. A blendshape pose is consequently created by combining all of the blendshape expressions that the artist has specified for a particular model. Each blendshape will be assigned a weight that specifies how much it will influence the final expression. Animators can create a wide range of facial poses by giving varying weights to the base expressions and mixing them in diverse manners after establishing a small collection of fundamental expressions. These fundamental expressions are typically based on the Facial Action Coding System (FACS). By using a blendshape representation, artists ensure an easier manipulation of the model, restricting the range of motion to only a combination of the allowed expressions. This makes it nearly impossible for users to create unrealistic-looking expressions. An example of blendshapes frequently used can be seen in Figure 32 where the subset of expressions necessary to create a full range of facial poses is presented.

A visual representation of how new expressions can be obtained can be seen in Figure 5 where the blendshape expressions for *Jaw Open* and *Mouth Pucker* are used and combined.

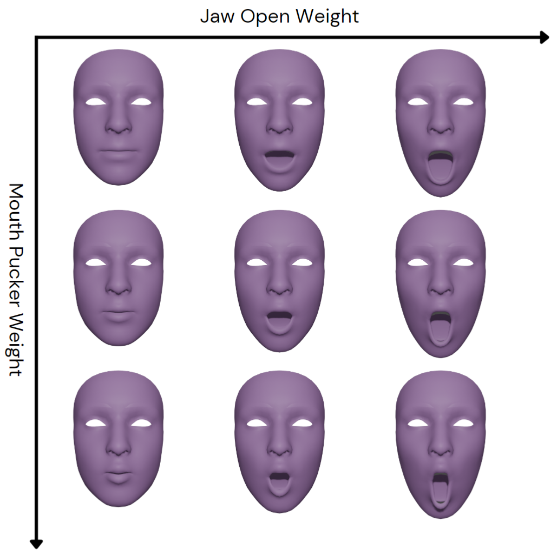


Figure 5: Representation of the weighted combination between two blendshapes

Even though the UUDaMM dataset is comprised of the facial control transforms rather than blendshape weights, at the core the final facial pose is still obtained by resolving the facial control parameters into blendshapes, so we can consider both the UUDaMM and BEAT datasets as being blendshape based. For this reason, we also consider that minimal adjustments will have to be made to the chosen architecture in order to train on one dataset or the other.

3.3 Deep Neural Networks

Initially inspired by the brain itself, deep neural networks have already led to many breakthroughs in a lot of different areas. We may divide the types of learning tasks into two broad categories, namely supervised and unsupervised learning, depending on the type of data we have available for training. While unsupervised learning uses unlabelled data, supervised learning makes use of labelled data that can be used to guide the model into making correct predictions. We can classify our work as supervised learning since our datasets contain pairs of input values in the form of audio features and ground truth values in the form of facial animation frames. Furthermore, we will mostly focus on methods used for generative tasks similar to ours.

Generally, we can define supervised deep learning algorithms as having two main steps: forward propagation (prediction) and backpropagation(learning). During the first step, the network will make predictions based on the input data and the parameters of the network (e.g. filter weights, and connection weights). This prediction will then be compared against the ground truth and a loss will be computed. The loss represents an error and quantifies how far off the prediction is from reality. Based on the loss, the weights of the network will then be updated, by propagating the error back into the network. These steps will be executed for several iterations defined by the designer, or until the algorithm has reached convergence.

Other concepts that should be taken into account when discussing machine learning tasks are the quality of fitting. By analysing how accurate the predictions for the training and test set are we can conclude if the model is: underfitting, overfitting or appropriately fitting the data. The training goal of an algorithm is to learn to make predictions based on the available data but also to generalise well to new data. Since our goal is to create a good generative model we need to construct a model that can generate novel animations and that can generalize to different types of speech, including noisy conditions and different languages. We say that a model underfits the data when it is not able to make good predictions either on the training or the test set. This might be caused by a variety of reasons like too little data, too simple architecture, or mislabelled data. Alternatively, when the model fits the training data too well, learning a very complex function, we say that the model is overfitting, making good predictions only for the examples it was trained on. These concepts will be relevant for us during the training of our model, especially when conducting the hyperparameter tuning. The goal is for the model to be able to fit well into both the training data, as well as unseen data. The quality of the fit will be supervised by plotting the training and validation losses after a certain number of epochs.

Convolutional Neural Networks (CNN). CNNs are a subclass of neural networks, typically used for tasks involving images or another type of multi-dimensional data. By including filters and convolutions, CNNs are not as prone to over-fitting as regular fully connected networks. Instead of computing connections between every neuron in consecutive layers, CNNs operate based on sliding a filter over the previous layer and computing the activation by convolutions. This way, the network is able to learn spatial and even temporal relations between different features, being able to identify patterns. The earlier layers of a CNN can extract more low-level features from the data, such as edges or textures, while the later layers will learn more high-level features, such as shapes or even more complex objects. Some versions of CNNs can even be used for single-dimensional data in the form of 1D convolutions. These can be used for inputs such as audio signals, but generally are not the best when it comes to encoding temporal information, with other methods like temporal convolutional networks (TCN) or recurrent neural networks (RNN) being preferred.

CNNs were used in the past for facial animations synthesis as was proved by *Karras et al.* [30] who computed linear predictive coding (LPC) features from the audio signal and reshaped the per-frame extracted features from 1D to 2D such that they can be processed by their CNN architecture. Due to the nature of the CNNs and their inability to learn temporal context, the authors had to design a loss specialised in ensuring the smooth transition between the predicted animation poses of the mesh. Similarly, one of the baseline methods we chose to compare our model with, VOCA [12], also makes use of a CNN network for animating 3D meshes, the task being very similar to ours. They also employ a velocity loss and experiment with an acceleration loss, to combat the problem of not taking into account the temporal context. While they manage to generate acceptable animations, their results are jittery and noisy. More about this will be elaborated in Section 7, where we show visual comparisons between VOCA and other methods, including ours. However successful, we consider this method to be outdated, and therefore we will not use CNNs when developing our model. Other methods, such as RNNs or transformers are much more suitable for dealing with sequential data, ensuring that temporal context is also learned without the need of designing additional losses.

U-Net. The U-Net type of architecture was first introduce by *Ronneberger et al.* [45] for the task of image segmentation. More specifically, their goal was to be able to correctly identify neuronal structures in biomedical images. This task is more complex than the simple labelling of an image since it requires extracting locality information from the image. Essentially, the network would need to be able to predict the class for every pixel instead of the whole image. To solve this issue, the authors propose a method with two consecutive paths: a contracting path for learning context and symmetric information, similar to the usual CNN networks, and an expanding path, which aims to reconstruct the image back to its initial size by up-sampling the image at each step in the algorithm. The 2 paths are symmetric and connected, the activation maps from the contracting path being copied to their corresponding expanding layer. This architecture has proven to be very effective for the task of image segmentation, only requiring small datasets to be able to give accurate results. This type of architecture proves to be a good choice for autoencoders as well, being able to reduce the dimensionality of the input data and then reconstruct it back to its original form. The skip connections specific to U-Net are usually missed in most autoencoders, but the general concept remains similar.

The U-Net architecture is also a popular choice when it comes to employing diffusion models for various computer vision tasks. Diffusion models have spiked in popularity due to the spread of "AI art" over social media. Through the use of the diffusion process, new high-quality instances of images, sound or other types of outputs can be generated from the latent space distribution of the examples in the training dataset. While these models have mostly been applied to image data as of yet, their power and potential for other fields cannot be ignored. Therefore, in this work, we aim to integrate them into our architecture with the goal of producing more expressive and diverse animations.

A UNet-type architecture can also be applied to the task of animation synthesis. *Vougioukas et al.* [53] have used it to add skip connections between a subject identity encoder and a frame decoder. It was also used for 3D character animation synthesis, but instead of images and pixel positions, 3D vertex meshes are being used as inputs. Meshtalk [44] has successfully integrated a UNet architecture as the decoder in their proposed method. Additional expression information learned in a prior stage is added to the middle layer, enabling the synthesis of more expressive animations based on non-speech-related facial expressions.

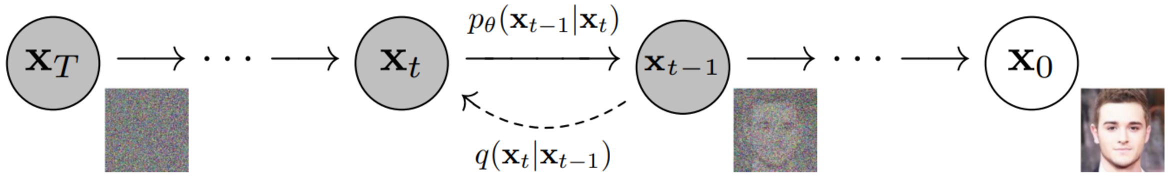

$\mathbf{x}_T$
$\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$
$\mathbf{x}_{t-1}$
$q(\mathbf{x}_t|\mathbf{x}_{t-1})$
$\mathbf{x}_0$

Figure 6: Diffusion process. Image source: *Ho et al.* [26]

Recurrent Neural Networks (RNN). Recurrent neural networks allow cyclical connections between nodes, with information from previous inputs being used to predict the current states. They are used for sequential data of variable length and therefore can be used for problems like speech recognition, natural language processing (NLP), music generation and others. Furthermore, the weights of the layers in an RNN are shared. In terms of actual training, RNNs differ from regular neural networks by making use of back-propagation through time (BPTT), meaning that the final loss will be determined by summing the generated losses for each of the timesteps. Most networks will also follow the encoder-decoder architecture type. While simple unidirectional RNNs allow the network to *"look-back"* and take information from previous outputs, some tasks, like language translation, might also require information from future outputs. For this task, bidirectional RNNs (BRNN) were introduced. The architecture of a BRNN is pretty straightforward, with two RNNs being used to parse the sequence from start to end and vice-versa, ensuring output conditioning in both directions.

Additionally, to ensure longer dependencies and not just on immediate consecutive segments, a different type of RNN unit was considered. The most popular units of this kind are:

- **Long Short-Term Memory (LSTM)**. LSTM is a specific kind of RNN cell designed to address the issue of preserving information over lengthy input data sequences. In contrast to conventional RNNs, LSTMs contain gates that regulate the flow of data in and out of the cell as well as a forget gate that decides which information from previous timesteps will be discarded and which will be used in the next steps. This makes LSTMs capable of learning long-term patterns in the data, thus making them suitable for tasks such as text translation or speech recognition.

- **Gated Recurrent Unit (GRU)**. The GRU cell is similar to LSTM, however, it only has 2 gates instead of 3. A reset gate controls the information flow into the cell, while the update gate controls the information flow outside of the cell. Since it is simpler than LSTM, and also has fewer parameters, GRU proves to often be more efficient than LSTM, architectures containing GRUs being trained much faster. In most tasks, they generally perform similarly to LSTM, thus making them an obvious choice if training times are taken into account.

Our project can also be defined as being a sequence-to-sequence task, with the audio input represented as a series of samples and the output as a series of animation frames. Therefore we will also make use of these types of sequence-to-sequence networks. To decide which one is most suitable for our task, we perform an analysis of the results obtained with different configurations.

Transformers. Similar to RNNs, transformers are also designed to handle sequential data, being suitable for the same range of applications.

However, they have the advantage of processing the entire sequence at the same time, allowing for more possibilities for parallelization, being in turn much more efficient. The transformer architecture was introduced by *Vaswani et al.* [52] and leverages the attention mechanism. Attention is a technique in machine learning that allows putting more weight on some parts of the data that are more important in order to increase their influence on the current predictions. By using multi-head attention and the benefits of parallelism, transformers have proved to be more efficient than regular RNNs.

3.4 Diffusion Models

Ultimately, we present the class of diffusion models and the concept behind how the process works. The concept of diffusion was introduced in 2015 by *Sohl et al.* [47] and is based on a concept taken from non-equilibrium thermodynamics.

A popular example given when diffusion models are explained is talking about a blob of ink in a glass of water. At first, when the drop of ink was first dropped in the water, we can still clearly distinguish it, but over time it *diffusses* in the water, making it impossible to know the initial structure of the drop of ink. With this example in mind, we can now explain how diffusion works in machine learning tasks. The idea behind it is to slowly noise an example in the dataset by gradually adding noise to it. The noising process is modelled as a Markov chain of operations between timesteps 0 and T. For ease of explanation, we will take the example of images. Let x_0 be a real image and x_T an image of the same size where all pixels have random values from the normal Gaussian distribution. We refer to the forward process of gradually noising x_0 as q-sample, while the backward process of gradually denoising x_T as p-sample.

The goal is to gradually add noise to it at each timestep, until timestep T, when the initial image is not recognizable anymore and the pixel values are completely random, representing a normal Gaussian distribution. At each intermediary step, during the forward process of noising the input, a little bit of Gaussian noise is added to the previous image. The process repeats iteratively until reaching timestep T. In Table 7 we can see this forward (bottom), and backward(top) process.

Now, if we go back to our own method of generating facial animations, we can define the diffusion process as follows. Let $x_0^{1:N}$ be a sequence of ground truth visual frames from the dataset with shape (N, C), where C is either the number of vertices, multiplied by the 3 axes, or the number of rig facial controls (or blendshape values). During training, we randomly sample timestep t, an integer value from the interval $(1, T)$ indicating the number of noising steps to apply to $x_0^{1:N}$, and to obtain x_t with the formula:

$$x_t^{1:N} = q(x_t^{1:N} | x_{t-1}^{1:N}) = \mathcal{N}(\sqrt{1-\beta_t} x_{t-1}^{1:N}, (\beta_t)\mathcal{I}) \tag{1}$$

In the above equation, β_t represents a constant that indicates the amount of noise added to the input sequence. The value β_t at each timestep is determined by the chosen variance schedule, which has to respect the rule: $0 < \beta_1 < \beta_2 < ... < \beta_T < 1$. The schedule can be either linear, cosine, quadratic or even more complex, indicating the spacing between the different betas. In our experiments, we have only used a linear schedule, meaning all of the beta values are at an equal distance from one another and the noising process is completely linear.

After we have executed the forward noising process, ideally we would want to be able to reverse it and go backwards from $x_T^{1:N} \sim \mathcal{N}(0,1)$ to $x_0^{1:N}$. Therefore we need to know the conditional distribution function $p(x_{t-1}^{1:N} | x_t^{1:N})$. However, this function is not yet known and we need to learn it. To this end, the proposed approach as it was introduced by *Ho et al.* [26] is to model a machine learning architecture that learns it, by learning the latent representation variance of the dataset. They define the training objective as learning to predict the noise ϵ that was added to the input x_0 in order to be able to extract it. As this approach would be favourable in the case where the task was to generate new unconditioned, or weakly conditioned outputs, similar to what we've seen in the image generation tasks, we think that for our specific task, the learning objective of predicting the correct x_0 would be more suitable.

Therefore, we follow in the footsteps of MDM [49] and EDGE [51] and design a deep-learning algorithm for generating visual frames from a condition (audio) and the learned latent representation of the dataset. By choosing this approach, our model is able to predict acceptable results even from the first denoising steps of the inference process, therefore, it allows us to reduce the inference time considerably. However following the full inference process would give the best results, the improvements are minimal after a threshold that is quickly reached.

Furthermore, we employ a simple loss for training as it was proved to be a good approach in both MDM and EDGE. More thorough experimentation was conducted by *Ho et al* [26], who also claim that utilising the simple loss for learning the variational bound proved to be both easier to implement and also to be advantageous for the quality of the sampling results. The formula for the simple loss is therefore:

$$\mathcal{L} = E_{x_0 \sim q(x_0|c), t \sim [1,T]}[\|x_0 - \hat{x}_0\|] \tag{2}$$

3.4.1 Sampling Process

Since we established how diffusion models are trained, now we will explain how the sampling process works. The idea is to employ the reverse process $p_\theta(x_{t-1}|x_t)$, that was learned during training, with θ being the trained configuration of the deep-learning model used to learn this function. Since we also want to sample the model based on an input condition, in our specific case audio, we define the inverse function as $p_\theta(x_{t-1}|x_t, c)$.

Sampling is thereafter an iterative process, starting from $x_T \sim \mathcal{N}(0,1)$ until we reach $\hat{x}_0$. It is important to note that at the first iteration, we would not have the actual ground truth input x_0 to diffuse and generate x_T, therefore x_T is just randomly sampled from the normal Gaussian distribution. This random initial sampling is the one that makes diffusion models to be non-deterministic and able to generate novel outputs each time they are sampled. Another important thing to note is that at each intermediary step, the model predicts the denoised output $\hat{x}_0$, which is then diffused back to timestep t and used in the following prediction. This noising step is important because the model was trained to generate outputs from noised inputs, and if it were to receive actual clean samples the outputs might be undesirable.

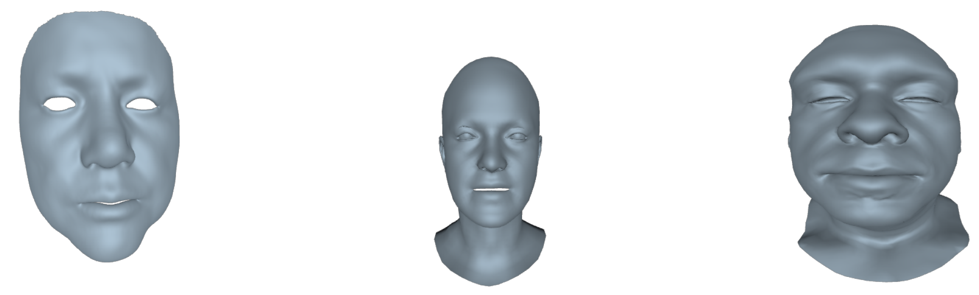

(a) BIWI Template (b) VOCASET Template (c) Multiface Template

Figure 7: Facial topology representations across the datasets

4 Datasets

The main requirement for a dataset to be usable in our architecture is to contain synchronised pairs of speech and facial animation sequences. By the format in which the animation sequences are stored, we can categorise the datasets as follows:

- 3D Vertex Datasets. An animation frame is represented by the positions of every vertex making up the mesh of the face.
- Facial Rig Controls. An animation frame is represented by properties (translation, rotation, scale) of rig controls. Handling these controls will result in creating different expressions on the face of the rigged character.
- Blendshape Based Datasets. An animation frame is represented by values stored for each blendshape defined for the face of a virtual character. Instead of manually moving controls like, in the case of the rigged characters, an expression can be created by adjusting the values of individual blendshapes and therefore creating new expressions.

In this section, we present the datasets used for training our model along with the particularities of each of these datasets. In our experiments, we mainly make use of the widely employed BIWI [20] dataset. By choosing a dataset that is already used by previous work, we have an easier time placing our work in comparison to other state-of-the-art approaches. Additionally, to show that our solution is not bound to only a specific type of dataset, we make use of the VOCASET [12], along with the recently released Multiface dataset [56].

Furthermore, our third research question is about our ability to automatically generate facial animations for rigged characters using deep-learning approaches. Therefore, we employ a slightly modified version of our model, adapted to deal with animations driven by rig controls instead of vertex displacements. To this end, we utilise our in-house UUDaMM dataset [37] to train and conduct experiments and ablation studies. Next, we also employ the facial animation sequences captured in the BEAT [34] dataset with the same goal of showing that our model can be adapted to more types of training data.

4.1 3D vertex-based datasets

These datasets include animations represented as vertex movements. At each frame, all the vertex positions of the mesh are known. They are therefore used in our training algorithm as such, representing the ground truth in our task of facial animation synthesis. While the topology of the meshes may vary across different datasets as can be seen in Figure 7, the format of the data is very similar and therefore no changes would have to be made in terms of the architecture.

4.1.1 BIWI

Following the steps of previous, we can see the trajectory of state-of-the-art works has most of the time used this specific dataset. We think this is because of its expressive animations along with the high-definition of the meshes, each face having a total of 23370 3D vertices. The chosen facial topology of the meshes only includes the face of the subjects, missing other features like eyelids or tongue. The dataset is comprised of 14 x 40 x 2 sequences of paired audio and animation. For the creation of these sequences, 14 subjects were asked to read and emote 40 different sentences, each sentence being read 2 times, one time with a neutral expression and one time with a more emotional one. Each sequence is on average approximately 5 seconds long and is captured at $25fps$. The face meshes include are very high-definition comprising 23370 3D vertices, despite only the front of the head being depicted. Compared to this, the other 2 vertex datasets which depict the whole head and neck, have a number of vertices equal to only 5023, and 6172 respectively. Based on previous work, we use the same data splits as in [19, 57] and only use the emotional subset of sequences. During training, only 6 subjects (3 female and 3 male) are used, along with 32 spoken sentences per subject. This amounts to a total of 192 sequences and represents the BIWI-Train dataset. From the remaining 8 sentences of these subjects, 4 are used for validations (24 in total), and 4 for testing (24 in total). We refer to this test set as BIWI-Test-A and will be used to compute objective metrics over our results. With the remaining, 8 subjects and their last 4 sentences, we form BIWI-Test-B which will be used in the user study for obtaining qualitative metrics. This dataset represents the main one we use during the model engineering phase and is the one our model was fine-tuned on. Furthermore, the hyperparameter tuning along with other experiments and the ablation study are mainly performed on this dataset.

4.1.2 VOCASET

This dataset is also frequently used for facial animation synthesis, however, is mostly used as an additional dataset rather than the main one. We think this is because of its non-expressive nature, including very few upper-face motions and is mostly focused on the lip part of the face. Despite this, we argue that it serves us well in analysing the lip-sync quality of our network without many distractions from other facial motions. The dataset was collected by *Cudeiro et al.* [12] and it accompanied their speech-driven facial animation solution.

The dataset is comprised of 480 sequences of audio-visual pairs which amount to a total length of just 29 minutes. The sequences are recorded at $60fps$ and the facial scans are translated onto the FLAME head topology [33] which is comprised of 5023 3D vertices. Unlike, the BIWI dataset, the mesh includes the whole head and neck of the person, including eyeballs and eyelids. Even though this would technically allow for more expressivity and possibly even eye blinks to be captured, these are scarcely available in the dataset. In terms of data split, we utilise the one proposed by the authors of the dataset and later used by other methods as well [19, 57], 8 of the 12 subjects along with all of their sentences are used for training (320 sequences), 2 subjects with 20 sentences per subject are used for validation (40 sequences), and similarly the last 2 subjects with the last 20 sentences are used for testing (40 sequences). This dataset is used mostly for obtaining objective and subjective metrics, that would make our comparisons with the state-of-the-art methods more complete. We do not use this dataset in the model design and hyper-parameter phase, and instead, just use it after the architecture is finalised.

Regarding pre-processing, the only step applied is to reduce the dimensionality of the data by training on a 30 FPS version of the dataset. This is done by dropping every other frame in the sequence.

4.1.3 Multiface

Despite being widely used, the previous datasets have their issues and we consider them to not be fully suitable for training a model able to generate diverse and expressive animations. The model will be able to learn expressions as much as the dataset includes them. For example, the VOCASET is mostly focused on lip movements, with the upper part of the face moving very little. Furthermore, none of these datasets includes eye blinks. Even though we are able to tell when the actor is blinking due to a deformation of the mesh in the eye region, the meshes in BIWI dataset do not have eyelids and even in the ground truth animations the "blinks" look more like glitches. Since this feature is missing, the model would not be able to learn it and generate it after.

For this reason, we include a third dataset in our research, namely the Multiface dataset, publicly released by *Wuu et al.*[56]. To the best of our knowledge, we are the first ones to use it for the task of facial animation synthesis besides its creators. Just like VOCASET, this dataset will only be used in the last part of our research to help us better compare our solution with other state-of-the-art methods.

In comparison to the other datasets, the face meshes included here are more complex in terms of features, containing attributes such as hair, eyelids and facial hair. Moreover, the dataset includes full 3D head scans, including the back of the head as well as the neck. We consider this dataset to be useful in training our diffusion model since it will have more chances of learning non-speech-related features, such as eye blinks or upper-face expressions.

The dataset contains a total of 13 subjects and it constitutes a subset of a larger dataset that was used for training Meshtalk [44]. Even though the full dataset is much larger, comprising a total of 250 subjects, it is sadly not available to the public. Nonetheless, the authors claim that the available subset is diverse enough to be able to use for facial animation synthesis. Considering the size of the other 2 datasets, we agree with the authors, Multiface being larger than both of them with a total of 650 sequences.

For each subject, there are a total of 50 spoken sentences available. Since the sequences are split by subject and sentence, it allows us to have a similar training technique, including the one-hot style embedding. The authors of the dataset share that the sentences were chosen in such a way that they are phonetically balanced ensuring a good generalisation across the possible phonemes.

Facial animations are encoded as a sequence of 3D face meshes captured at 30 frames per second. Each frame is represented by the full 3D face of the actor, with a total of 6172 vertices, including eyelids, neck, as well as different hairstyles for the subjects. An example of this can be seen in Figure 7, where we can see that the Multiface topology is closer to a real-life representation of a human head than the other 2 topologies from the other datasets.

Since there is no previous work to follow, we propose our own data split and use 9 of the subjects with the first 40 sentences for training (360 sequences) and call this subset Multiface-Train, the following 5 sentences of the same 9 subjects are used for validation (45 sequences), and the last 5 sentences make up Multiface-Test-A and are used for testing (45 sequences). With the other 4 subjects and their last 5 sentences that were unseen during training, we form Multiface-Test-B and use it for the subjective analysis.

Dataset	BIWI	VOCASET	Multiface
Training Set	• 6 subjects • 32 sequences per subject • Total = 192 sequences	• 8 subjects • 20 sequences per subject • Total = 160 sequences	• 9 subjects • 40 sequences per subject • Total = 360 sequences
Validation Set	• 6 seen subjects • 4 sequences per subject • Total = 24 sequences	• 2 unseen subjects • 20 sequences per subject • Total = 40 sequences	• 9 seen subjects • 5 sequences per subject • Total = 45 sequences
Test Set A	• 6 seen subjects • 4 sequences per subject • Total = 24 sequences	✗	• 9 seen subjects • 5 sequences per subject • Total = 45 sequences
Test Set B	• 8 unseen subjects • 4 sequences per subject • 6 conditions per sequence • Total = 192 sequences	• 2 unseen subjects • 20 sequences per subject • 8 conditions per sequence • Total = 320 sequences	• 4 unseen subjects • 5 sequences per subject • 9 conditions per sequence • Total = 180 sequences

Table 1: Summary of the dataset split for the 3 vertex-based datasets used in our experiments

4.1.4 Data Preprocessing

The ensure the data is suitable for training our deep-learning model, we apply some transformations to the raw data provided in the previously presented datasets.

Since there is already a clear split of sequences and previously employed train-test-validation splits, we do not take any steps in updating the data in this regard and simply use it as it is. For Multiface, which was not used for facial animation tasks as of yet, we propose our own split that can be seen in Table 1 along with the splits used for the other 2 datasets. To be consistent with the training format, we decided on a data setup previously employed by FaceFormer [19], FaceXHuBERT [25], and CodeTalker [57], namely each sequence will be encoded in a matrix of size N x $V \cdot 3$, where N is the number of frames in the sequence, and V is the number of vertices in the mesh.

Furthermore, an initial analysis of the data from the VOCASET, shows that the vertex data is already scaled to fit into a unit box, therefore we do not apply any preprocessing steps on it. For the

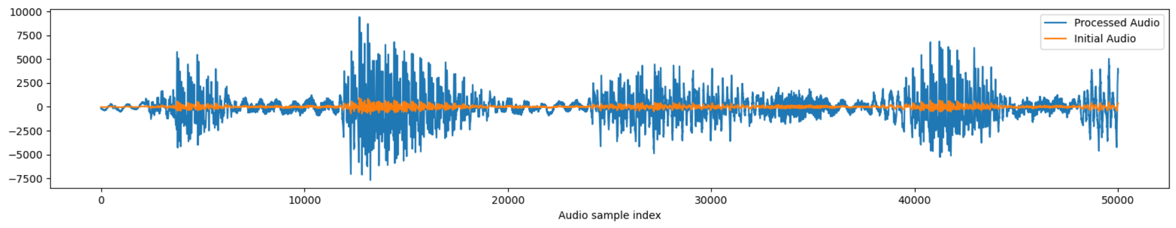


Figure 8: Example of an audio signal from the Multifacce dataset before and after processing.

BIWI set we use the same scaling that was used by CodeTalker, which employs a standard scaling of the vertices.

The only 3D-vertex dataset which requires some attention from us is the Multiface dataset. Firstly, the audio data is recorded in a 5-channel setting while the audio in the other 2 datasets is monophonic. This would be a problem since our model is able to handle monophonic data, and higher-dimensionality audios being reduced to just their first channel. To align with the audio format in the other datasets, we have chosen to convert the audio in Multiface to monophonic as well. We do this by taking the average intensity value across the 5 channels. Furthermore, an initial manual analysis of some of the sequences in the dataset revealed that the audio volume of the speech was very low and mostly inaudible by the human ear. While this might not constitute a problem for a robust speech encoder like HuBERT, we think it is important for the data to reflect a scenario closer to real-life usage, where the audios have an appropriate volume. Therefore, we have decided to increase the volume of all the audio in the dataset by a constant $d = 20db$. The value was chosen by comparing the average sound intensity in the BIWI dataset with the average sound intensity in the Multiface dataset. An example of one such audio adjustment can be seen in Figure 8.

Finally, the sequence data is stored as a series of *.obj* files, each file representing a frame in the animation. The first step we take to bring the data to the same format as the others is to transform the sequences into a matrix format. Additionally, we scale the vertex data to ensure the meshes fit into a unit box. In order for a machine learning algorithm to train well, the data needs to be normalised and all of the features should take values in similar ranges. Therefore, we apply min-max scaling on the vertex data of the dataset. We first compute the ranges of values over the 3 spacial axes: x, y, and z, and then scale each vertex coordinate accordingly to take values in the $[-0.5, 0.5]$ range.

4.2 Rigged character datasets

4.2.1 Utrecht University Dyadic Multimodal MoCap dataset (UUDaMM)

In this project, we also make use of the Utrecht University Dyadic Multimodal MoCap dataset (UU-DaMM) [37] previously collected by other Game and Media Technology students conducting their master thesis research under the supervision of Dr. Zerrin Yumak. The dataset consists of dyadic conversations between 2 actors. The dataset contains 9 hours and 41 minutes of recorded conversations between two speakers, of which 6 hours and 53 minutes represent active speech sequences, in which at least one of the two actors is speaking. The sequences were captured over 4 days, with 2 or 3 sessions per day. Five takes were recorded for each session so that each actor is equally represented in the dataset. To achieve this, a different type of conversation was assigned to each take. The purpose of these different types of conversation was to generate all the types of gestures that can occur during conversations: beat, iconic, metaphoric, and deictic. During this project, we are not using gestures, and only focusing on facial expressions instead, but we consider that the way the dataset was collected is also helpful for the task of facial animation, ensuring a balance in actor participation as well as a large variety of different sounds expressed during the almost 10 hours of conversations that were collected. A more detailed representation of the size of the dataset can be seen in Table 2.

The dataset is comprised of multiple types of conversations aimed to cover a large variety of gestures. In terms of facial animation, we do not require this information and simply extract the sequences without keeping track of the type of conversation they were extracted from. Nevertheless, we present a quick summary of each of the conversation types along with descriptions of the capture settings.

Actor	No. frames	No. sequences	Audio length
1	4184510	2537	9 h 41 m
2	4184510	2537	9 h 41 m
Total	8369020	5074	19 h 22 m

Table 2: UUDaMM dataset size

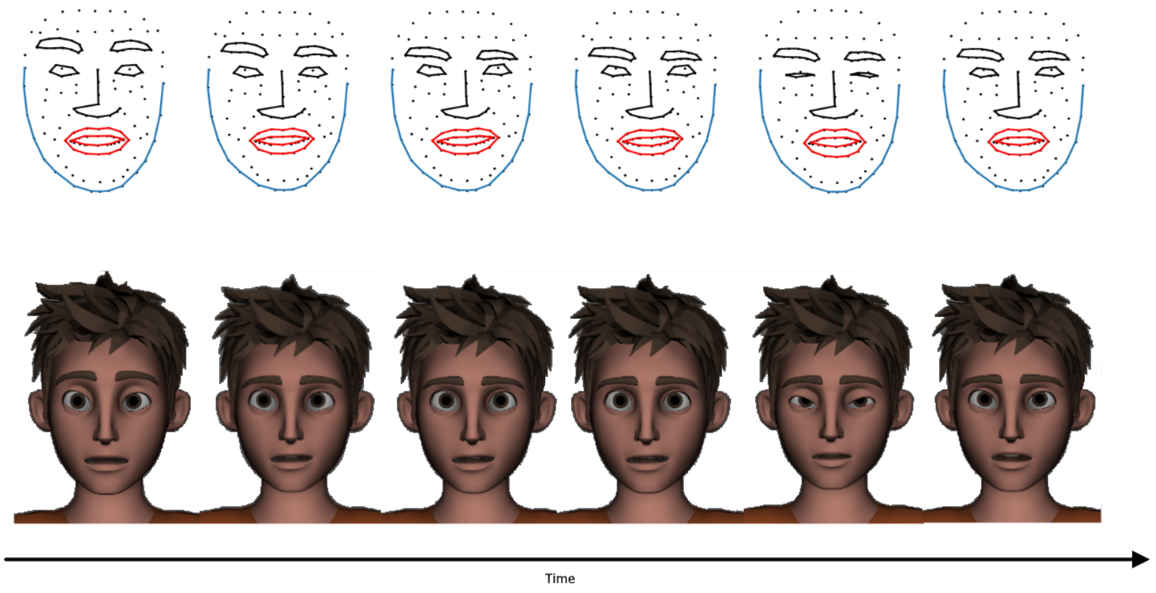


Figure 9: Series of frames from the dataset represented as landmarks (above) and character controls (below)

- **Description.** During these takes, mostly only one of the actors is speaking, the other just guiding the conversation and also acting as a listener. To this end, two types of tasks were presented to the actors. First, a description game in which one actor has an object in mind while the other actor has to guess it by asking questions about the object. Secondly, one of the actors was asked to describe something, such as an event in their life. The description task was therefore structured as a monologue, mostly only one actor speaking during these takes.

- **Free-Form.** These conversations had the purpose of most closely resembling natural random conversations that two people might have. To this end, the actors were presented with random starter topics and given the freedom into continuing a conversation from there. The actors were also allowed to deviate from the topic to ensure that the discussions remained natural. The ratio between the speakers is approximately 70% : 30%.

- **Debate.** These conversations ensured equal participation from both speakers. To ignite the debates, the actors were presented with either a question or a statement on a controversial topic and instructed to continue the conversation based on that topic.

The takes were categorised into discussion types as follows: description tasks constituted two of the five takes, free-form conversations represented two, and debate represented one. The roles of the actors were switched between the two takes for the first two types to ensure a balanced frequency of actor engagement.

The dataset includes motion capture data recording the gestures of the actors, as well as facial capture data in the form of facial controls, recording the expressions of the actors. Furthermore, the recorded audio files from the 2 speakers are provided in wav format, along with transcripts. The transcripts were produced manually by the authors of the dataset. The facial animations are saved as either 119 3D landmarks (Figure 33) or 192 facial control values for the Ray character (Figure 4). A representation of a short sequence of frames can be seen in Figure 9, where the recorded landmarks along with the rigged character expressions are presented. Since the included landmark data is 3D we can consider this to be a low-detail mesh representation of the face of a character. For this reason, landmarks were used first in the testing phase of the project, since we could more easily make the transition from vertex data to landmark data. After we managed to successfully train a model on the landmark data, we made the transition to the control-based data and trained a model able to generate accurate facial animations for the previously presented Ray character. Since we did not conduct any relevant experimentation on the landmark dataset, constituting merely a bridging step in our experimentation we will not present it further. Nevertheless, we thought it relevant to mention the existence of this other type of data encoding that could potentially be used for facial animation synthesis.

Since the captured sequences are quite long compared to the previous ones and can include moments of silence as well, we decided to split the data equally into 10-second clips. By choosing a length of 10 seconds, we ensure that in most sequences there is at least a significant part of speech. Even if a sequence does not include any speech from the actor, their expression is still recorded and therefore would prove to be useful in training a more generalised model.

Data Division, Cleaning, and Labeling The dataset contains separate audio inputs from both the actors, however labelling that will let us know the role of the actor does not exist. The authors of the dataset attempted an automatic division of the data based on aligning the audio and text with the use of Gentle[5] forced aligner. However, they note that the division is not perfect due to instances of words not being found in the audio or the transcript and suggest that a better division should be considered for future work.

The first step we take in terms of data processing is to downsample the visual frames from 120 FPS to 30 FPS. This is done by simply extracting the frames that are at a distance of 4. The 3 frames in between them are dropped and not considered. Since the human eye is not able to distinguish the difference at a frame rate higher than 24, we consider 30 to be an adequate choice when it comes to the generated animations. Furthermore, most modern animation engines have simple ways to increase the frame rate in post-processing. Since we do not take the role of the actor into account and only require paired sequences of audio and facial animation we decided to employ a basic sequence split, dividing the longer sessions into sequences of 10 seconds each, or 300 frames respectively. To keep the sequence size consistent, the last sequence from a session is dropped if it is shorter than 10 seconds.

Data Scaling The features that we will be using for training might take values in different ranges. This means that we need to scale the features to similar intervals, such that the training can assign similar importance to all the features. The features that we will use for training are represented by audio and facial features, the latter also being separated into landmarks and blendshape-based features.

The possible preprocessing that we could do for audio includes volume adjustment and equalisation, such that all the audio inputs are uniform. However, the HuBERT model has proved to be robust when it comes to noisy audio, so we consider that no further preprocessing will be necessary.

In the case of facial features, they need to be scaled to more appropriate ranges. For this, we simply compute the mean and standard deviation of each feature in the data and scale them based on the formula:

$$z = \frac{x - \mu}{\sigma}, \tag{3}$$

where

μ : mean
σ : standard deviation

Despite the disadvantage caused by the specificity of the dataset (only 2 participants and only one rigged character), we consider that a method developed for this dataset will be easily adjustable to animating other characters that are defined with the same parametrization. The results would of course be bound by the expressions portrayed by the 2 actors that were used for recording, but we consider it relevant to show the possibility of training such a model able to generate animations in the form of

[5]https://github.com/lowerquality/gentle

Dataset	DaMM	BEAT
Training Set	• 2 subjects • 2029 10-second sequences per subject • Total: 4058 sequences	• 4 subjects • 2175 10-second sequences
Validation Set	• 2 subjects • 254 10-second sequences per subject • Total: 508 sequences	• 4 subjects • 274 10-second sequences
Test Set	• 2 subjects • 254 10-second sequences per subject • Total: 508 sequences	• 4 subjects • 275 10-second sequences

Table 3: Summary of the dataset split for the 2 blendshape-based datasets used in our work.

facial rig controls. To counteract this issue, we employ a second similar dataset, the recently released BEAT, which includes 30 different subjects by comparison, along with multiple spoken languages.

4.2.2 BEAT

The second blendshape-based dataset we use is the BEAT dataset [34]. Just like UUDaMM, the dataset also contains body motion data along with facial capture, the difference being that the facial animations are encoded as sequences of blendshape weights instead of rig controls.

ARKit along with the depth camera of an iPhone 12 Pro is used to capture the facial movements of the actors, encoded as 52 blendshape weights defined in ARKit. The frame rate of the facial capture is 60 FPS. The dataset includes 30 participants, of which half are female and half male. Each participant was asked to read 118 predefined texts, each resulting in a one-minute recording, in various emotional ways in order to cover multiple variations of emotional speech. The 8 emotions captured during the collection are neutral, happiness, anger, sadness, contempt, surprise, fear, and disgust. An additional 12 recordings of 10 minutes each were captured for each participant in which they perform free-form conversations with an off-screen director.

Furthermore, the dataset contains sequences spoken in 4 different languages: English, Chinese, Japanese, and Spanish, also including native and non-native English speakers. The total size of the recordings amounts to about 76 hours of sequences. All these features make the dataset to be the most diverse out of the ones we considered, the only downfall being that dyadic conversations are not recorded. In our experiments, we employ a subset proposed by the authors of the dataset, BEAT-16h, including 4 different native English speakers.

In terms of pre-processing, we do not apply any transformations to the values of the features themselves, and merely split the sequences so that the data follows a similar format to that present in the other datasets. After splitting the provided sequences into 10-second segments, we obtain a total of 2724 that are used for training. In terms of data split, we employ an 80-10-10, training-validation-test split ensuring that each of the 4 subjects is adequately represented in the 3 splits. An overview of the used splits for both the UUDaMM and BEAT datasets can be seen in Table 3.

5 Methodology

As previously stated, the task for this thesis is to use deep neural networks to generate facial animation driven by audio. While extensive research has been done in the area of animating a 3D vertex mesh character, not the same can be said about the animation of a blendshape-based character. Furthermore, earlier approaches mostly focused on datasets including short spoken sentences, ignoring the conversational aspect of speech along with other particularities like pauses, stuttering or listening.

To this end, following the current state-of-the-art, we first define a deep-learning model trained on vertex data, employing widely used datasets such as BIWI [20] and VOCASET [12]. Since there are plenty of such methods previously proposed that were trained on these datasets, we have an easier time conducting a thorough analysis of our proposed method. Since there are not many methods trained on blendshape data, our second model analysis will be briefer, therefore it is essential to establish first how the vertex model performs.

In this section, we present the architecture design for the 2 proposed models and the methodology employed for choosing the best configurations for these models. Finally, we present the evaluation design including the metrics we aim to measure. For more clarity, we divide this section into 2 parts, starting with the vertex-based model and continuing with the blendshape-based one.

5.1 Problem Formulation

For the sake of clarity, we formulate our task as follows. Let A be an audio input associated with a sequence of ground truth frames $x_0^{1:N} = (x_0^1, x_0^2, ..., x_0^N)$, where N is the number of visual frames sampled at the specific dataset FPS. Each frame in the sequence x_0^n is represented as an array of vertex positions with the length V x 3, where V is the number of mesh vertices in the topology, and 3 is the number of spatial axes. In the case of the last datasets, x_0^n represents just a sequence of rig controls or blendshape values, having the length C, the number of controls running the facial rig. Based on only the audio input A, the goal of our architecture is to predict a series of animation frames $\hat{x}_0^{1:N}$ that resemble the ground truth frames $x_0^{1:N}$. Additionally, the predictions will be guided by a style S in the form of a one-hot vector with a length equal to the number of training subjects and noise $x_T^{1:N}$ sampled from the normal distribution $\mathcal{N}(0,1)$ and with the same shape as the ground truth sequence $x_0^{1:N}$. This noising process will be explained in more detail in the following sections. Worth mentioning is that instead of predicting vertex positions, V-FaceDiffuser generates animations in the form of vertex displacements in regard to a face template, the last step in the training process being to sum the network outputs with the input template as it can be seen in Figure 11. The second version of our model does not require this step, as the facial representations are not character-specific and can be transferred easily between different characters, provided they have a similar rig.

Finally, we can abstract our model with the following equation:

$$\hat{x}_0 = \text{FaceDiffuser}(\mathcal{A}, x_t, t) \tag{4}$$

where

$_0$: predicted animation sequence
A : input audio sequence
x$_t$: sequence $\hat{x}_0$ after t diffusion steps; when $t = T, x_t = \sigma^{1:N}$ drawn from $\mathcal{N}(0,1)$

5.2 Proposed Model Architecture

Since we have chosen to work with 2 different types of datasets, namely vertex and rig-based datasets, and since the way in which animations are encoded varies greatly, we were compelled to tune the models differently. Therefore, we have designed a general model that could be employed by both types of datasets, but with slight modifications in terms of hyperparameters. From here forward, we refer to the vertex-based model configuration as V-FaceDiffuser, and to the blendshape-based model as B-FaceDiffuser.

As it can be seen in Figure 11, the main difference that the 2 models have is that the B-FaceDiffuser does not include the *Noise Encoder*. This decision was made due to the data dimensionality, the vertex data being too large to be used as it is. We mention that through the noising process the input sequence $x_0^{1:N}$ keeps its initial shape (N, V x 3) after the noise was added to it and $x_t^{1:N}$ was computed.

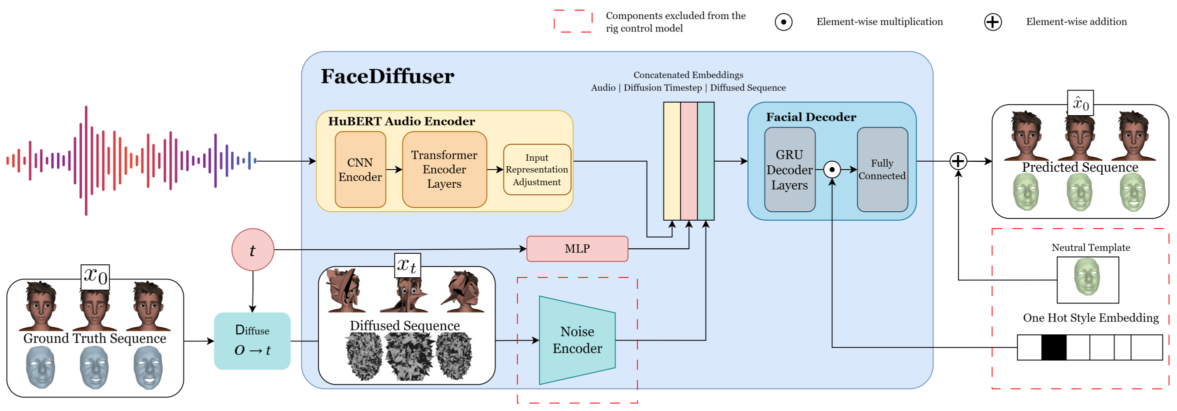


Figure 10: High-Level overview of the FaceDiffuser model architecture. Pictured here is the training overview; used inputs are paired audio-visual pairs; the visual frames go through the diffusion process, while the audio frames act as a conditioning term for the generated frames. Raw audio and noised ground truth animation frames are used as input for our model. Speaker animation frames are generated by the decoder ($\hat{x}_0$) and compared with the ground truth animation (x_0) to compute the loss and propagate it back into the architecture.

In the presented architecture we can identify the following main components that are included in both versions of the model:

Audio Encoder: HuBERT. The audio encoder is kept the same in both versions of the architecture. We employ a pre-trained version of the HuBERT architecture and use the released *hubert-base-ls960* version of it, which was trained on 960 hours of speech. Additionally, we mention that HuBERT uses the wav2vec 2.0 pre-trained processor to pre-process the audio inputs. This pre-processing is done before training and is not part of our algorithm.

Facial Decoder. The facial decoder component is responsible for producing the final animation frames. It is comprised of several GRU layers followed by a final fully connected layer that transforms the hidden states outputs of the final GRU layer from the GRU latent dimension to the dimension of the output sequence. During the decoding step, a style embedding can also be added in the form of an element-wise product between a learned style embedding vector and the hidden states output.

An overview of our proposed architecture can be seen in Figure 11. Just like in the case of animation encoded by vertex displacements, we aim to synthesise animation $\hat{x}_0^{1:N}$ of length N determined by the length of the input audio signal $a^{1:N}$. The training goal of our model is therefore to generate animation frames $\hat{x}_0^{1:N}$ that is as close as possible to the ground truth animation frames $x_0^{1:N}$. Each animation frame $x_0^n \in \mathcal{R}^{192}$ is represented by an array of facial control values, each value being responsible for controlling either the position or rotation of a specific rig control as can be seen in Figure 4

The pre-trained HuBERT model, which represents our *Audio Encoder*, will receive the raw audio input and process it. This will produce per-frame audio features at the frame rate of 50 FPS. Since our visual data might be recorded at a different frame rate, we perform an audit adjustment to align the 2 modalities.

Additionally, our model also receives as input t, a uniformly sampled timestep from the range $[1, T]$. Based on this sampled value, we gradually add noise to the ground truth animation frames during the diffusion process, and we obtain $x_t^{1:N}$, the noised ground truth sequence. Similar to the audio input, these 2 extra inputs are encoded.

The 3 obtained embeddings are then concatenated and sent to the *Facial Decoder*, which will predict animation frames $\hat{x}_0^{1:N}$ based on these inputs. The *Facial Decoder* is represented by a stack of 4 GRU layers with hidden size 1024 followed by a final fully connected layer that brings the output features back to the same dimension as the number of facial controls 192.

By including the diffusion process into our architecture we ensure that the predicted animations will be novel and non-deterministic for every new inference based on the same audio.

The inference is an iterative process, during which we go through all of the diffusion timesteps backwards from T to 0, gradually improving the prediction at each inference step. Since at inference time, we miss one of the inputs the model was trained on, namely the noised ground truth animation.

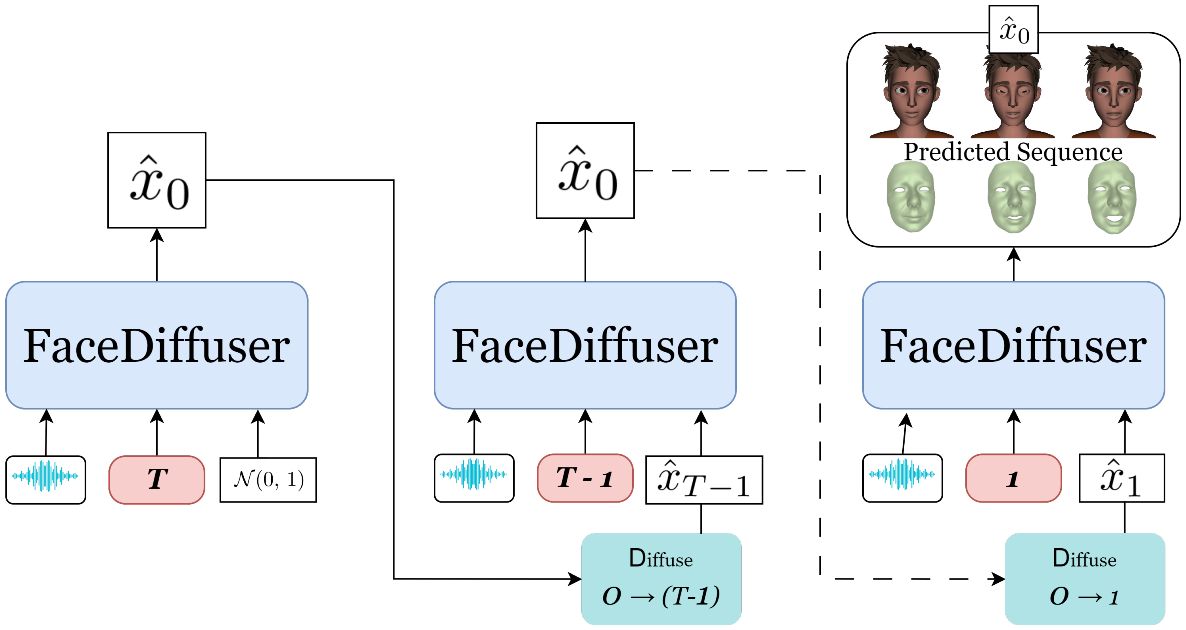


Figure 11: High-Level overview of the FaceDiffuser inference procedure. The inference is an iterative process, the first "noise" being represented by actual noise from the normal distribution $\mathcal{N}(0, 1)$. At each step, we provide the network with the audio and noised animation input. The predicted motion is then diffused again and sent to the next step. The model is able to predict concrete animations even from step T, but going through all of the steps will ensure the best quality.

At inference time T we provide the network we randomly sampled noised from the uniform distribution $\mathcal{N}(0, 1)$

The output of the decoder will then be compared to the ground truth animation paired with the audio, and the loss function will be calculated. Finally, the loss will be propagated back into the architecture, where it will be used to adjust the neural weights and guide the network to produce better predictions.

5.2.1 Loss Function

Using a simple loss error loss based on comparing the predictions with the ground truth has proved to be a good enough option for several previously proposed methods [41, 19, 17, 40]. Furthermore, we follow the work of the diffusion body animation models MDM and EDGE [51, 49], and train our model for predicting the actual control values instead of the added noise ϵ. We argue that in the case of more complex data that does not have a clear structure, like the animation controls, optimizing our model for predicting the real values is a much clear task.

To this end, we employ a Huber loss, that measures the error between the ground truth and predicted frames. The advantage of this loss formulation is that it combines an absolute error with a mean squared error, adapting the computation based on the size of the error. Therefore, this error is not influenced a lot by outliers and extreme values in the data.

$$L(x_0, \hat{x}_0) = \begin{cases} \frac{(x_0-\hat{x}_0)^2}{2} & , |x_0 - \hat{x}_0| < \delta \\ \delta * |x_0 - \hat{x}_0| - \frac{\delta^2}{2} & , otherwise \end{cases} \tag{5}$$

where

x_0 : ground truth sequence
$\hat{x}_0$: predicted sequence
δ : threshold that specifies when to change between L1 and L2 loss

5.3 Experimental Setup

Training Details All the models, along with the different configurations employed in the experimentation phase are trained on one NVIDIA A16 GPU, for a total of 50 (final) or 100 epochs (experiments). The training duration varies depending on the size of the used dataset along with the sequence representation dimension and is reported in the tables in Section 7. We make use of the Adam optimizer [31] and initialise the learning rate at $1e-4$. We use an online learning approach, updating the weights of the network after each training step.

5.3.1 Hyperparameter Tuning

A key component of creating a deep learning pipeline is hyperparameter tuning. By selecting the appropriate settings, the model will be capable to learn useful knowledge from the data. The objective is to create a model that can generate accurate predictions based on the training data while avoiding overfitting the training dataset and generalising well to unseen inputs.

The following hyperparameters were taken into account in this phase of the project:

- **Number of epochs.** Choosing an adequate number of epochs is important to ensure that the model has learned enough to be able to make good predictions. A low number of epochs may indicate that the training is incomplete, and the model may still update its weights to reduce the loss further. However, overfitting may result from prolonged training, which makes the model less generic to new data. Another issue with sequential models that was discovered during testing is that the model tends to learn an "average" output for the frames after a sufficient number of epochs, resulting in mostly stagnant animations. We shall take into account methods like early stopping in order to prevent overfitting. In machine learning, early stopping refers to terminating the training process before the predetermined number of epochs if the validation loss does not continue to decrease. To determine the number of epochs, we will plot the training and validation losses after each epoch and see their progression.

- **Decoder architecture.** We compare different decoder architectures including RNN, GRU and transformer and make a choice based on the quality of the results obtained with these versions of the decoder.

- **Number of layers in the decoder.** This defines the complexity of the decoder. The higher the number of layers, the more complex the model is. While a simpler model might not be able to learn all the correlations between the different features of the audio and the facial controls used as targets, a too complex model might quickly overfit the training data, choosing an appropriate value is therefore important.

- **Number of hidden state features.** Similar to the previous parameter, this also dictates the complexity of the model. We test different values and compare the results.

- **Noise encoder architecture.** Additionally, for the V-FaceDiffuser version of the model, we experiment with different architectures for the noise encoder, starting from a simple fully connected layer to a more complex design that links multiple layers together. Here, we also experiment with the use of single-dimensional convolutional layers.

Considering the nature of our task and the high training times, a fully realised grid search in the hyperparameter space would not be feasible. For this reason, we will perform hyperparameter tuning by training the network on just a small subset of the data. This is only valid for the blendshape datasets since they are significantly larger than the vertex datasets.

To this end, we will make use of a subset of 200 sequences for each of the 2 actors. We obtain the sequences by equally splitting the raw sequences into chunks of 10 seconds each with a frame rate of $30FPS$. This means that there will be a total of approximately 66 minutes of paired audio-visual frames of data used for hyperparameter tuning. To keep the data balanced, we first shuffle the full dataset and keep the first 200 sequences for each actor. We then choose an 80-10-10 train-validation-test split.

Furthermore, due to a high number of possible configurations of hyperparameter values, we do not perform a grid search in the parameter space. By doing this we reduce the number of tried

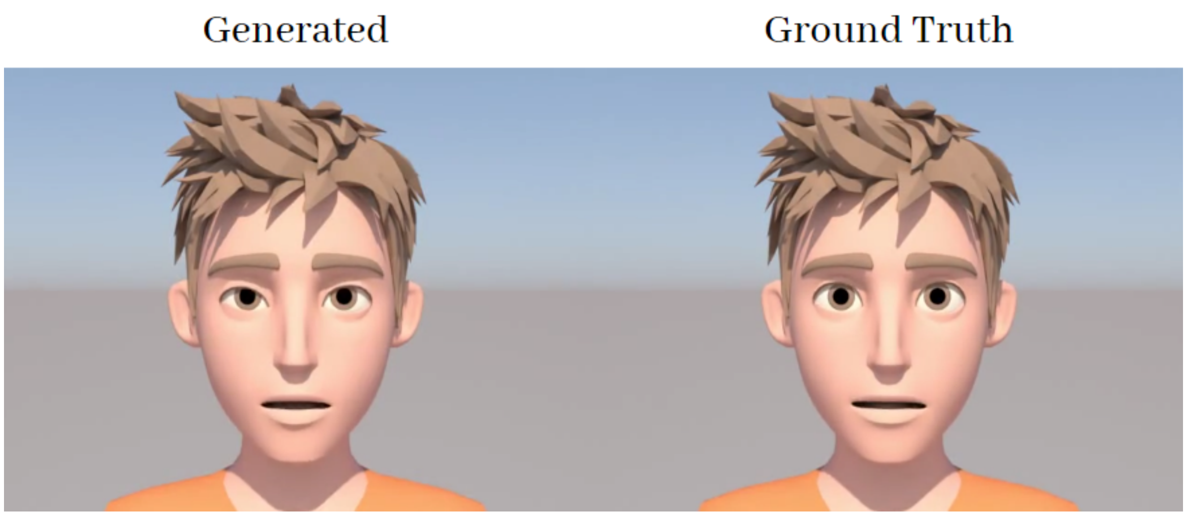


Figure 12: Side-by-side comparison between generated animation frames and ground truth frames

configurations to just a subset. We will randomly choose configurations and based on the results update the architecture. Then we move on to another hyperparameter and repeat the process.

While the results of training on just a portion of the data might not be perfect, we argue that they will be informative enough to guide the hyperparameter choice for the final network.

To compare the different parameter configurations we use both objective metrics, like the test loss, loss evolution graphs, or the lip-sync accuracy, as well as a subjective analysis, done by manually analysing the results. We argue that a subjective analysis is also important for this specific task since the model might just learn a mean animation that minimises the loss function, but in reality, the actual results are bad and uncorrelated to the input audio. In order to have a clear comparison of the outputs, we render side-by-side videos with the ground truth animation and different configurations as can be seen in Figure 12. This will help us in noticing differences between the results.

5.3.2 Evaluation

The evaluation of our proposed model, FaceDiffuser, is done in comparison with 3 other state-of-the-art methods, namely VOCA (2019) [12], FaceFormer (2022) [19], and CodeTalker (2023) [57].

For this, we employ the previously presented datasets, BIWI and VOCASET. All of these methods, including ours, require subject conditioning on one of the subjects used during training. For the seen subjects, we, therefore, condition the generation on the same subject. If the subject is unseen, we generate sequences conditioned on every training subject separately. For objective metrics, we use BIWI-Test-A, which contains only sequences conditioned on the same subject. Additionally, we also report the objective results on VOCASET-Test containing only unseen subjects. In this case, we make use of all of the generated variants of a sequence to obtain the objective evaluation.

To evaluate our rig-based model, we compare 2 different versions of it, a baseline model that does not use diffusion and our proposed model. We took this decision since, to the best of our knowledge, there are no publicly available end-to-end solutions for generating facial animations for rigged characters. Even though acceptable results might be obtained with the vertex-based approaches, we think such a comparison would be unfair and therefore decide not to do it.

5.3.3 Objective Evaluation

To objectively evaluate our results, we need to compute some metrics that could quantify the quality of the animations. We consider that employing a good objective metric is essential for accurately quantifying our results. From previous work [19, 12, 57], we have seen that vertex-based metrics are employed, measuring the error between the predictions and the ground truth animations. We consider these to be easy to compute, however, we consider that they only quantify a small part of the possible results. Since the problem in itself is a one-to-many problem, restricting the evaluation to a one-to-one computation is not ideal. Based on this, we follow previous work and employ similar metrics, while also proposing an additional diversity metric that does not take the ground truth into account and therefore does not suffer from the one-to-one issue the other ones introduce.

The most employed metric is represented by a reconstruction loss, where the predicted values are compared directly to the ground truth. While this might be a good metric when it comes to one-to-one problems in generation tasks like ours is not always the best choice. Facial expressions might differ from subject to subject even when the spoken sentence and the emotion are the same. A more suitable metric for this task might be an error based on the accuracy of the lip sync. While this is still not perfect, as there still exists some variation in lip shapes for the same phonemes, we consider the possible lip shape space to be much lower than the space of total expressions, therefore a metric based on lip-sync would be much more informative for the quality of the results than a general metric. Other metrics, such as the FDD metric introduced by CodeTalker [57], aim to compute a metric based on facial dynamics in a sequence. They argue that the general upper face dynamics should be close to those included in the ground truth data.

Even though these metrics have their flaws, at the moment of writing this thesis, they are the best choice as an objective metric, therefore we will also use them to analyse our results.

For the first model, based on vertices we employ the following objective metrics

- **Mean Squared Vertex Error (MVE)** We compute this error between the ground truth frames and generated frames. In the case of the blendshape datasets, we use the corresponding metric MCE (Mean Control value Error). The metric is computed the same in both cases, the only difference laying in the type of data it is computed over. To obtain this value, we compute the mean error of every frame in the predicted set and then get the mean value of all the errors.

- **Lip Vertex Error (LVE)** We compute the difference between ground truth frames and generated frames by taking into account only the mouth region. This will measure how our model performs compared to the others in the task of lip-synching. In the case of the UUDaMM dataset, we will only take into account the controls that directly influence the mouth area and compute LCE (Lip control error). Similarly, for the BEAT dataset, we will only consider the blendshapes that are highly correlated to audio-lip synchrony. These specific blendshape expressions can be seen in Figure 32, circled with a dotted line.

- **Upper Face Dynamics Deviation (FDD)** This metric is introduced by *Xing et al.* [57] and it aims to measure the motion variation between the ground truth and the generated sequences. In particular, the metric is only applied on the upper-face vertices, and it computes the standard deviation of vertex motions.

$$FDD(M^{1:N}, \hat{M}^{1:N}) = \frac{\sum_{v \in V_U} \sigma(M_v^{1:N}) - \sigma(\hat{M}_v^{1:N})}{|V_U|} \tag{6}$$

  where,

  $\mathrm{M}^{1:N} = x_0^{1:N} - x_0^{0:N-1}$: motions of the animation sequence
  V_U : list of vertices in the upper-face

  Since the metric is newly introduced, being only used before by its authors, we think it is important to have a deeper understanding of it before employing it in our experiments. Analysing the proposed Equation (6), we see that the metric does not have a lower or upper bound, being technically able to take any value in the domain, including negative values. A negative value would mean that the generated sequence is more dynamic, producing a higher variation of the upper-face motions.

- **Diversity**. We introduce an additional metric meant to measure the model's ability to produce diverse animations. With the introduction of diffusion, our goal was to develop a model able to generate a great range of motions and non-deterministic expressions for the regions of the face that are not correlated to speech. Based on similar metrics used in diffusion papers, we introduce a novel face diversity metric. Our model is non-deterministic and technically able to generate new animations at every new inference (provided the starting noise x_T is different), and we could compute this metric over different inferences with the same audio conditioning. However, since the other methods we compare ourselves with are deterministic, such a comparison would not be fair. Taking this into consideration, we define the diversity metric as follows. Let $\hat{x}_0^s$ be a generated sequence, conditioned on subject $s \in S$, where S is the list of training subjects. We compute the mean vertex difference between every 2 sequences conditioned on different subjects. We then take the mean of these differences and define the diversity of a sequence as follows:

$$Diversity = \frac{\sum_{i=1}^{|S|-1} \sum_{j=i+1}^{|S|} \|\hat{x}_0^i - \hat{x}_0^j\|}{\frac{(|S|-1)\cdot|S|}{2}} \tag{7}$$

where,

S : list of the training subjects
$\hat{x}_0^i$: predicted animation sequence conditioned on the ith subject from the list S

Since we see more diversity as a good thing, a higher value of this metric would be desired. Nevertheless, we stress that this metric should only be employed as a complement to the other objective metrics and not on its own as the results might be unreliable. Completely randomly generated sequences would produce high values for this metric but that does not indicate the quality of the results, as the produced animations might be noisy and not correlated with the audio. Considering this, a combination of low LVE and high Diversity would be ideal, indicating the model is able to produce accurate lip movements while maintaining high diversity between different styles.

We consider the first 3 metrics to be more subject-specific and consider that they would not give relevant information when applied to unseen subject sequences. This is because particularities such as the amplitude of the mouth opening might differ from subject to subject and would therefore generate a higher value for the LVE metric, while not necessarily quantifying whether the lip-sync is accurate or not. Therefore, good values for these metrics would be obtained if along with accurate lip movements based on the sound input, the model also learns to well reproduce the style of the subject. For this reason, these metrics will be computed over the A test sets. In the case of the VOCASET, for which we only have one test set of unseen subjects, we use all of the different versions of a sequence (conditioned on different training subjects) and average over it to compute these metrics.

That being said, our proposed diversity metric is more suitable for unseen subjects, conditioned on different training subjects, therefore we compute this metric over the B test sets.

5.3.4 Subjective Evaluation

We argue that objective metrics are not enough to quantify the quality of the results in the case of facial animation synthesis. The problem in itself is one-to-many and so far no quantitative metrics were introduced to take this into account. Even if we consider only the mouth part of the face, there could still be high variations in terms of style, such as the maximal lip distance. Moreover, if there are moments of hesitance or silence in the audio, these are also not taken into account by the previously presented metrics.

Consequently, we also employ a perceptual study of the obtained results. We adopt an A/B testing approach, meaning that we show pairs of results from our model and a competitor and ask the participants to choose the better result based on 3 criteria:

- **Lip-Sync Quality**. This is meant to measure the perceived accuracy of the lip movements per the provided audio.
- **Animation Realism.** It quantifies the perceived realism of the resulting animations. A more realistic animation would be closer to real-life facial movements.
- **Animation Appropriateness.** With this we aim to find out which of the resulting animations fit the provided audio overall. We ask the participants to account for everything they might find relevant to this criterion, such as perceived emotion from the audio.

Just like in the case of the objective metrics, we compare our results with the same state-of-the-art methods, also including a comparison with the ground truth. This results in 4 different comparisons. For this task, we employ BIWI-Test-B, along with the VOCASET. We leave the Multiface dataset out since we were not able to obtain acceptable results after training VOCA and FaceFormer on it.

We decided to split the comparisons on BIWI and VOCASET into 2 different surveys. We took this decision in order to keep the surveys short and prevent user tiredness. In each of the studies, a participant has to examine 12 pairs of animations, 3 for each of the 4 comparisons, and answer 3 questions for each of the comparisons following the criteria we previously presented. In BIWI-Test-B we have 32 sequences in total, 4 for each of the 8 unseen subjects. Since the subjects were unseen during training, we need to condition them on a different subject. For a fair comparison, this conditioning subject is randomly chosen for each of the sequences, ensuring a uniform spread of the conditions, i.e. each of the training subjects is used to condition $5-6$ of the test sequences. Similarly, with the 40 sequences from VOCASET-Test, we do a similar thing, ensuring that each of the 8 training subjects conditions 5 of the sequences in the test set. In doing so, we ensure a fairer comparison by employing a broad style diversity.

In addition to these 2 surveys, we design an extra one meant to act as a subjective ablation study of the diffusion component. We use sequences obtained from UUDaMM-Test and use the results of B-FaceDiffuser in comparison to its no diffusion version, B-Face. We also compare our results to ground truth animations to test whether the results are believable enough to be comparable to real facial motions. This survey has 10 comparisons where the same 3 questions as before are asked. The 10 comparisons are distributed as follows: 4 comparisons with the ground truth, 4 comparisons with B-Face using audios from the UUDaMM dataset, and 2 comparisons with B-Face using external audios. In order to obtain the pool of comparisons, we randomly picked 20 sequences from UUDaMM-Test, ensuring that there is enough speech in each of the sequences. 10 extra sequences are obtained by randomly selecting audio from the BIWI dataset and using them to generate animations from B-FaceDiffuser and B-Face.

To summarise, we designed 3 different experiments, 2 of which compare our model with state-of-the-art competitors on 2 different datasets, and an additional study meant to serve as a subjective ablation for the diffusion component. Further details about the surveys can be seen in Appendix A.

The list of the surveys and the data they include is:

1. Ours vs SOTA on BIWI-Test-B
2. Ours vs SOTA on VOCASET-Test
3. Diffusion Ablation Study on UUDaMM-Test

6 Experiments

In order to reach the final architecture configuration, we conducted several experiments to select different configurations of some of the components of our model, as well as the best hyperparameter values.

As it can be seen in Figure 11, the high-level architecture is very similar for both types of data and the model can be broken up into 3 main components:

1. Audio Encoder
2. Facial Decoder
3. Diffusion Model
4. Diffused Facial Encoder (Only in the case of the vertex-based model)

6.1 Audio Encoder Experiments

In terms of the audio encoder, there is no need for a lot of tuning. We harness the power of the pre-trained HuBERT model and use the audio encodings obtained from it. In an ablation study that is detailed in a follow-up section, we try different configurations of the audio encoder along with replacing it with wav2vec 2.0. In doing so we solidify our choice of the audio encoder.

6.1.1 Audio Input Masking

In order to encourage our model to learn the non-speech-related expressions from the latent expression space, we employ a random dropout of the audio condition. Therefore, with a dropout probability $c = 25\%$, we mask the audio features and provide the network with $\emptyset$ instead of the actual $\mathcal{A}$. The value for dropout probability was chosen based on experimentation with different values for it. In introducing this dropout we are later able to also generate unconditional face animations representing general expressions that are not constrained by audio input.

We try the following audio input masking strategies to compute the new $\emptyset$ value:

- **Fully Zero.** This means that the full matrix of audio features is set to 0 in 25% of the training steps.
- **Partially Zero.** This means that 25% of the audio features are set to 0 at each training step.
- **Fully Random.** This means that the full matrix of audio features is set to random values in 25% of the training steps.
- **Partially Random.** This means that 25% of the audio features are set to random values at each training step.
- **None.** We do not apply any dropout on the audio features and use them as they are.

In Table 4 we can see the results of employing these different audio dropout strategies. We can see that in 2 particular cases, of the Full Random and Partial Zero strategy we obtain the best results in terms of objective metrics. Interestingly enough, these versions also perform better in terms of lip-sync accuracy, seeming that the introduction of the dropout has also helped the model learn more accurate lip movements.

6.1.2 Discussion on unconditioned vs. conditioned sampling

By randomly masking the audio input during training, we effectively create a dropout mechanism, which has two main benefits: preventing the model from overfitting and encouraging the model to learn animation features that are not correlated to the speech. Since our model is able to also learn unconditioned features, we can employ a sampling mechanism frequently used in diffusion papers and combine the conditioned and unconditioned outputs into a final animation, using a weighting factor w_{cond}.

Strategy	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)
Full Zero	4.6300	4.4729
Full Random	4.3830	3.9557
Partial Zero	4.5156	3.7956
Partial Random	4.6499	4.3906
None	4.6289	4.2560

Table 4: Objective evaluation of different audio dropout strategies

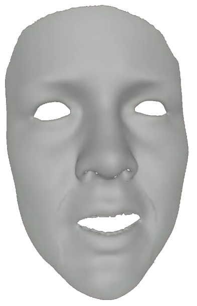


Figure 13: Visual artefacts produced by combined conditioned and unconditioned sampling

$$\text{FaceDiffuser}_w(x_t, t, a) = w_{cond}\text{FaceDiffuser}(x_t, t, \emptyset) + (1 - w_{cond})\text{FaceDiffuser}(x_t, t, a) \tag{8}$$

Despite the apparent benefit of this, we could not identify a correct setting for the weighting factor that could produce good results. Setting the value too low results in bad lip-sync, the model not accounting so much for the audio input anymore. The opposite case of setting the value too high proves to have little benefit as the unconditioned generation is overpowered by the conditioned one. Another approach we tries was to have a per-vertex combination of the two outputs, with the conditioned part having a higher weight on the lip region and the unconditioned part on the other vertices. This also has proved to not bring good results, as it produced visible artefacts between the regions driven by the different modalities as can be seen in Figure 13. Therefore, we decide to focus on only the conditioned sampling mechanism and leave this area as a possibility for future work.

6.2 Facial Decoder Experiments

One of the main components of our model is represented by the decoder. This part is responsible for taking the encoded audio features and other input modalities used during training and generating the actual animation frames. For both of our proposed models, we try different decoder types, of which we mention GRU, RNN, and transformer. Furthermore, we also experiment with different hyperparameter configurations. All of the models were trained for 50 epochs as the model starts to overfit if it is trained for too long, the validation loss not improving much, while the training loss continues to decrease.

RNN Decoder . We try a 2-layer RNN decoder with hidden-size 512. Analysing the results we can see that the animations produced are consistent with the audio, the mouth opening and closing at appropriate times. While the animations are mostly good, we do not notice moments of jitteriness that appear unnatural. Even though these moments are few and mostly occur in longer sequences and when there are moments of silence, ideally we would like to eliminate them.

Transformer Decoder . For the transformer, we try an implementation of a vanilla transformer with 1 layer and 4 attention heads. We base this configuration on the one proposed in the FaceFormer paper, also using the 2 learning strategies presented there. As we can see from the table, the teacher-forcing approach, while faster than the autoregressive strategy, produces significantly worse results. This is also true when we analyse the videos of these results. Effectively the transformer using teacher forcing was not able to train well, producing mostly static animations. While we can see the mouth opening at the correct times, this is not enough to create the illusion of accurate lip-sync and appears rather random. In the case of the autoregressive transformer, we see that it was able to train well and it produces quality animations. However, this is to the detriment of higher training times.

GRU Decoder . Finally, we test our proposed GRU decoder composed of 2 GRU layers with hidden size 512. This method has proven to be the best out of the test ones, producing the best results quality. Visually, the results are comparable to those obtained with the autoregressive transformer, however, when we take training time into account, GRU seems to be the better choice.

Decoder Type	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)	# Parameters	Training Time (h)
GRU	**6.9831**	**4.4696**	**3.8258**	121515870	0.90
RNN	7.0833	4.7870	4.0690	120727390	0.67
Transformer - Teacher Forcing	9.9767	10.1300	4.8587	194132574	1.34
Transformer - Autoregressive	7.0213	4.6941	4.3272	194132574	5.00

Table 5: Evaluation metrics computed over the test set for different facial decoder configurations

In Table 19 we can see the results obtained on different decoder configurations. Analysing them, we can see that overall, the best-performing model is the one that uses a GRU decoder. Furthermore, in terms of training time, GRU and RNN are the most efficient ones, only needing less than an hour for full training on the dataset. Compared to that, the autoregressive transformer model needs 5 hours for the same amount of that. Taking into account the iterative diffusion process, coupled with the fact that the autoregressive transformer model also requires an iterative frame-by-frame inference process, we consider the transformer to not be suitable for our goal. Moreover, we can see that the transformer model trained with a teacher-forcing scheme performs the worst out of the 4. Looking and the animation results we see that it does not manage to train well and it produces almost static frames that would not be considered realistic. Looking at the experiments of *Haque et el.* [25] we see a similar result for this decoder, therefore we also consider it to not be suitable. Finally, based on the obvious objective results, we chose the GRU as the decoder in our model.

Decoder Type	MCE ↓	LCE ↓	C-FDD ↓	# Parameters	Training Time (h)
GRU	**3.6791**	**3.5812**	1.8862	108395328	4.5
RNN	3.8654	3.9196	**1.8426**	88062784	2.92
Transformer	3.6881	3.7105	1.8533	126048064	4.58

Table 6: Evaluation metrics computed over the test set for different facial decoder configurations

B-FaceDiffuser Facial Decoder In the case of the rig-based architecture, we follow a similar experimental setup, testing different decoders.

The chosen configurations, in this case, are the same, the difference being in the values of the hyperparameters.

- **RNN Decoder**. First, we try an RNN decoder architecture with 4 stacked RNN layers and hidden size of 1024. Due to fewer correlations between the output features, we were forced to choose a more complex configuration for the decoder. Visualizing the results, we observe a similar performance as the RNN decoder in the vertex-based model. The results are acceptable, but slightly jittery. This is especially noticeable in the eye region, with the eyes moving consistently in an unnatural manner. The single advantage of this approach lies in the significantly shorter training time, however, we consider the loss of quality to be too noticeable for this to be a real advantage.

- **Transformer Decoder**. This time we try an 8-layer vanilla transformer decoder with 6 attention heads. We have trained the model in a teacher-forcing manner, therefore significantly lowering the training time. For the rig-based approach, this strategy has proved to be successful, the resulting animations being of good quality.

- **GRU Decoder**. Our proposed decoder architecture includes 4 GRU layers with a hidden size of 1024. This method has also proven to be the best out for this variant of our model. While training time is higher than RNN, the objective metrics as well as the resulting videos are significantly better than the RNN-based approach.

6.3 Diffusion Model Experiments

In terms of the diffusion model, we experimented with different diffusions step numbers as can be seen in Table 7. Analysing the visual results we conclude that the number of diffusion steps does not influence the model's capacity of producing acceptable animations, all of the configurations producing correct lip-sync animations. The differences are mostly noticed when it comes to the general expressivity of the videos. The results obtained with just 100 diffusion steps are generally the worst both objectively and subjectively, while we do not see a lot of differences between the results obtained with the other tested values. Since the number of diffusion steps does not generally influence the training times and we can see that acceptable results are obtained from the first inference diffusion steps, we decide to continue our experiments with a number of 500 diffusion steps.

Diffusion Steps	MVE $\downarrow$(x 10^{-3} mm)	LVE $\downarrow$(x 10^{-4} mm)	FDD $\downarrow$ (x 10^{-5} mm)	Diversity $\uparrow$
100	7.2391	4.7703	5.0705	0.8236
250	6.9618	4.5515	4.2407	0.8331
500	**6.8507**	**4.4364**	4.3212	0.8446
750	7.1290	4.6428	**4.1268**	**0.8725**
1000	7.0387	4.6897	4.5889	0.8617

Table 7: Evaluation metrics computed over the test results of different numbers for the diffusion timesteps

6.4 Hyperparameter Tuning

Hyperparameter	Tested Values	V-FaceDiffuser value	B-FaceDiffuser value UUDaMM	B-FaceDiffuser value BEAT
Number of epochs	25, 50, 75, 100	50	100	100
Diffusion Steps	100, 250, 500, 750, 1000	500	1000	1000
Input Embedding Dim	128, 256, 512, 1024, 2048	512	-	-
Number of GRU Layers	1, 2, 3, 4	2	4	2
GRU hidden size	256, 512, 1024	512	1024	256

Table 8: Hyperparameter values that were tested during tunning

Number of epochs. We train the model for 100 epochs and plot the loss graph as it can be seen in Figure 14. Additionally, we sample the model at every 25 epochs and compute metrics over the results. As can be seen from the loss graph, the model quickly starts overfitting, with the training loss going rapidly down without a lot of change from the validation loss. When we analysed the rendered results for each epoch we have seen that the animations sampled at epoch 25, while acceptable, were not perfect, the lip movements looking unnatural and stiff. For the other 3 epochs, we have not seen much change in the results and neither in the metrics computed over their results. With this in mind, we decided upon training our final model for 50 epochs. In the case of the rig-based datasets, we could still see improvements in the loss graphs even after epoch 50. Therefore for the B-FaceDiffuser, we have chosen to train the model for a total of 100 epochs.

Size of input embedding. We experiment with different sizes for the input embedding which is fed into the facial decoder. All of the input modalities are encoded to the same latent size, therefore the final input embedding will always have a size equal to 3 x $latent_size$. The values we experiment with for this latent dimension are 128, 256, 512, 1024, and 2048. In our experiments, a latent size of 512 has proven to produce the best results therefore it is the size we employ in our architecture as well.

In the case of the B-FaceDiffuser version, we did not conduct further experimentation with the input embedding size. Since we were not forced to encode the noised animation input, we decided on using it as it is and concatenate it to the audio features directly. Therefore, the final size of the input embedding for this model is equal to the sum of the audio embedding dimension, number of facial controls/blendshapes, and timestep embedding dim. The first 2 values are determined by HuBERT and the used dataset. HuBERT produces 768 audio features per-frame, sampled at 50 FPS. Since our visual data is processed at 30 FPS, we employ the input adjustment component of our model, by first upsampling the audio features to 60 FPS and then concatenating every 2 consecutive pairs, resulting in 1536 per-frame features at 30 FPS. More specifically, each visual frame will correspond to 2 audio frames, meaning a total of 1536 audio features. To this, we add the number of controls, 192 in the case of UUDaMM, and 51 in the case of BEAT. Finally, for the timestep embedding we use a simple multi-layer perceptron network that produces embeddings with dimensions equal to 128. This embedding is then copied for every visual frame.

Number of facial decoders GRU layers. For this hyperparameter, the values we decided to test were 1, 2, 3, and 4. Our results showed similar performance for the V-FaceDiffuser version of the model, therefore we decided that a 2-layer GRU architecture would be acceptable for our decoder. In the case of B-FaceDiffuser trained on UUDaMM, the model only started to produce acceptable animations when 3 and 4 GRU layers were used. We consider this to be caused by the more complex relationship between the facial controls and the audio features. Since there is a significantly lower number of controls than vertices, we think that the model had a harder time learning the correct expressions for each audio input. Furthermore, the facial controls used for training are in the end used for the computation of blendshape weights in Maya. This in turn adds another level of complexity. In the case of the BEAT dataset, blendshape weights are outputted directly by our model and we have observed that this is also handled well by a simpler architecture. In the end, we made the choice of employing a 4-layer GRU as our facial decoder for B-FaceDiffuser trained on UUDaMM, and a 2-layer GRU for the facial decoder of B-FaceDiffuser trained on BEAT.

Size of facial decoder hidden states. Additional to the number of layers, we also test different values for the hidden states dimension. Out of, the test values 256, 512, and 1024, a hidden state size of 512 has given the best results in the case of V-FaceDiffuser. The first 2 test values produced similar results, while in the case of 1024, the model starts overfitting very fast producing worse results on the test set. In the case of B-FaceDiffuser trained on UUDaMM, it seems that a more complex decoder produces better results, therefore we have chosen 1024 as the size of the hidden dimension of the GRU layers. For the model trained on BEAT, we have again observed that a simpler configuration produces better results, therefore the hidden dimension size used is 256.

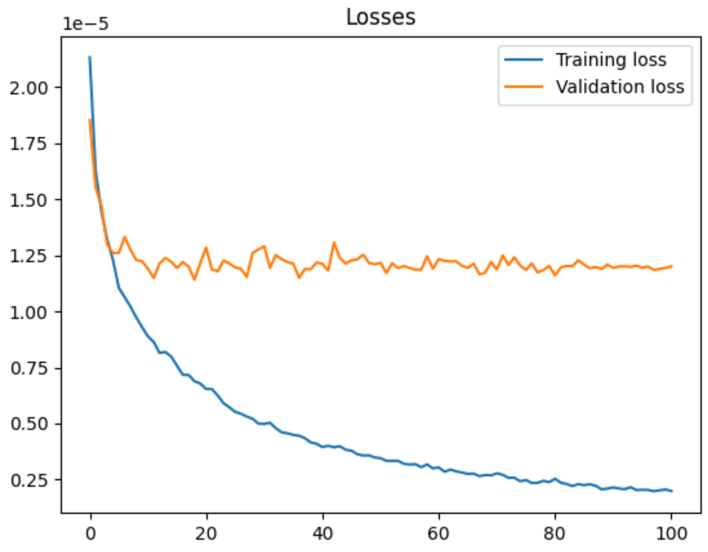


Figure 14: Loss graphic of training the model for 100 epochs. We see that the model quickly starts to overfit the data.

7 Results

7.1 Quantitative Analysis

We evaluate our proposed FaceDiffuser model in comparison with other state-of-the-art methods and on different datasets. To this end, we have chosen CodeTalker [57], Faceformer [19], and VOCA [12]. Furthermore, we will provide objective metric results computed over the results of these models based on what was presented in Section 5.3.3. Additionally, to these metrics, we also provide some visual aids for a better objective comparison. We render side-by-side expressions generated by specific key-frames related to relevant phonemes where the mouth should have a deterministic shape. Some of these phonemes include the /b/ and /p/ sounds, where the lips should be close together, or the /o/ and /u/ sounds when the mouth should be open and rounded. Furthermore, we provide graphs showing the distance between the lips and emphasize some specific sounds. Finally, we illustrate the general motion of the face in the form of heatmaps illustrating the mean motion and the standard deviation of each specific vertex on the face.

Since extracting the vertex values for the rigged models would be too much work, we decided to illustrate the general face motion by computing the optical flow over the rendered videos since we render the animations in the same conditions (camera centred on the face and directional lighting) we think this to be a good approximation, still showing us relevant information.

7.2 Vertex-FaceDiffuser Results

Method	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)	Training Time (h)
VOCA	8.3606	6.7155	7.5320	0.61
FaceFormer	7.1658	4.9847	5.0972	4.58
CodeTalker	7.3980	4.7914	4.1170	4.52
FaceDiffuser	*6.8088*	**4.2985**	**3.9101**	1.62

Table 9: Objective evaluation results computed over the BIWI-Test-A set. Best results are marked as **bold**

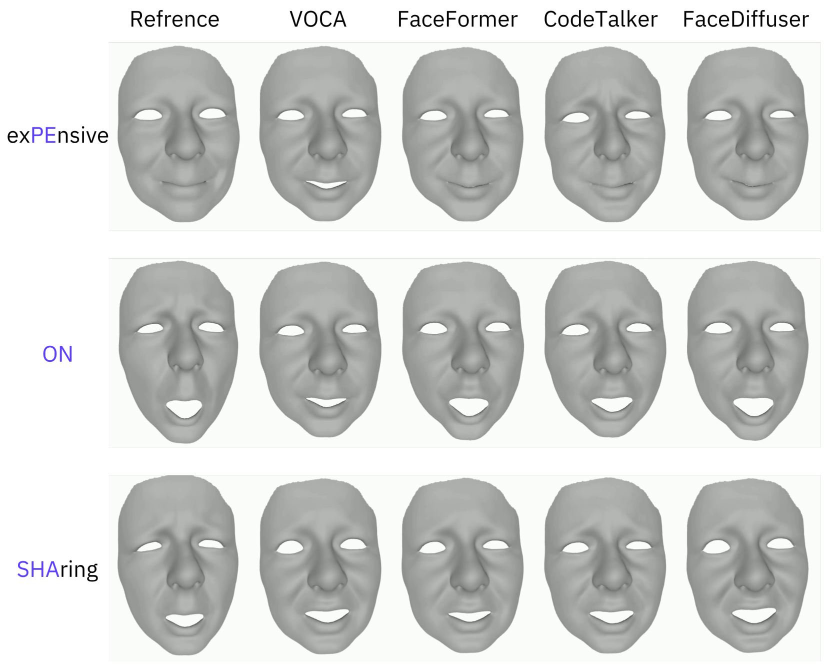


Figure 15: Visual comparisons with state-of-the-art methods. Sequences were sampled from BIWI-Test-B and each row is conditioned on the same training subject.

Method	MVE ↓(x 10^{-4} mm)	LVE ↓(x 10^{-5} mm)	FDD ↓(x 10^{-7} mm)
VOCA	9.1606	5.1076	3.7755
FaceFormer	9.1598	4.1564	3.4983
CodeTalker	**8.6666**	**4.1374**	3.3326
FaceDiffuser	9.5486	4.1652	**3.2375**

Table 10: Objective evaluation results computed over the VOCASET-Test set. Best results are marked as **bold**

Method	MVE ↓(x 10^{-2} mm)	LVE ↓(x 10^{-3} mm)	FDD ↓(x 10^{-5} mm)
CodeTalker	1.2170	2.0392	6.6857
FaceDiffuser	**0.7815**	**1.0034**	**4.9088**

Table 11: Objective evaluation results computed over the Multiface-Test set. Best results are marked as **bold**

Analysing the visual representation of the mean face motion in some randomly extracted sequences (Figure 16) we can see cases where our model performs better than state-of-the-art, but also cases where it performs worse. The quality of these results seems to be strongly correlated with the style condition that was used to generate the sequences. Our model seems to make stronger correlations to the style and does not generalise well across different styles. In our opinion this can be a good thing, our model learning to separate well between different styles and producing highly different motions when a different style embedding is used. However, there is also a downfall to this case, namely that if the ground truth animations for a specific style do not show a lot of movement in the upper part of the face, our model will also learn that for that specific condition, it should not produce a lot of movement. While we can easily shift to another condition and choose between the options on which the model was trained, in a perceptual evaluation like the one we designed, we might come short if the "bad" style embedding was chosen several times. A method like CodeTalker does not suffer from

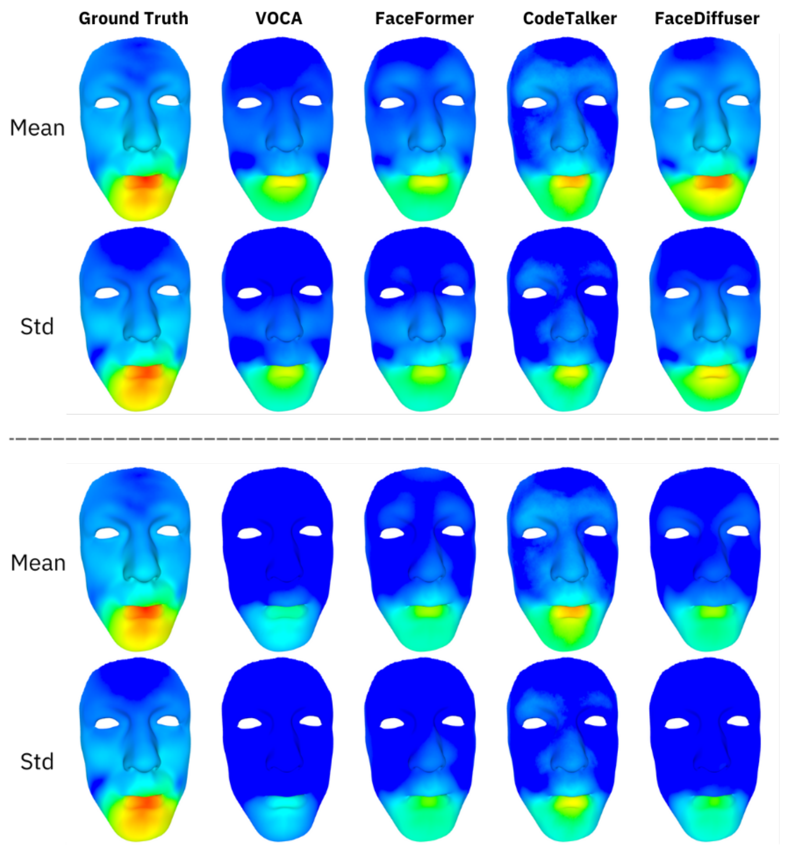


Figure 16: Visual comparisons with state-of-the-art methods. Sequences were sampled from BIWI-Test-B and each row is conditioned on the same training subject. We render the mean motion and standard deviation across the same sequence conditioned on different subjects. The colour scale goes from dark blue (low values) to bright red (high values).

this issue, producing more generalised results across different conditions, in the case of BIWI-Test-B the results for the same sequence are nearly identical to one another as it can also be seen in Table 12 where the diversity metric results are shown. There we can see that CodeTalker has a particularly low diversity score on the BIWI-Test-B set, meaning that the produced sequences conditioned on different training subjects are nearly identical to one another.

A possibility for this might be the introduction of diffusion. While it seems to help in generating more motion in the upper part of the face compared to a similar model that does not use diffusion, this seems to introduce extra subject-specific information into the training that gets correlated with the one-hot style embedding. This in turn will guide the model into learning a stronger subject-specific style instead of learning a more general expression space from the dataset, which seems to be the case with CodeTalker.

Looking at the results produced from the training on VOCASET, we can see somewhat similar results. Looking at Figure 17, we can see that our model produces average motions that are closer to the actual ones present in the ground truth data, this, in turn, translating into low values for the objective metrics as can be seen in Table 10. An interesting thing we can observe is that our model performs the worst in terms of the MVE metric. Since the MVE metric takes into account the whole set of vertices in the mesh, we think that this might be a direct result of our model generating motion even in regions that have little to no motion in the ground truth sequences. An example of this is the neck region, where we can see our model producing more motion than all competitors. We attribute this to the diffusion process as well and argue that it is generally a good thing considering that less movement would make the animations look robotic.

Another objective analysis we conduct is plotting the lip distance during a sequence sampled from

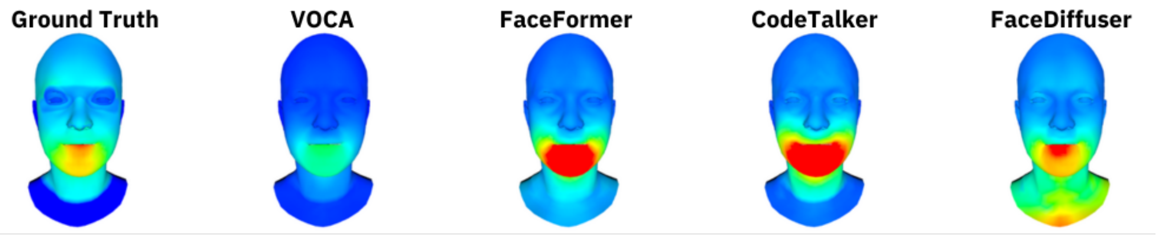


Figure 17: Mean motion across an animation sequence. Dark blue colors indicate little to no motion in the region, while bright red indicates a lot of motion. The sequence was randomly sampled from VOCASET-Test and the same condition subjects was used for all methods. We see that our model produces average motions that are closer to those in the ground truth data.

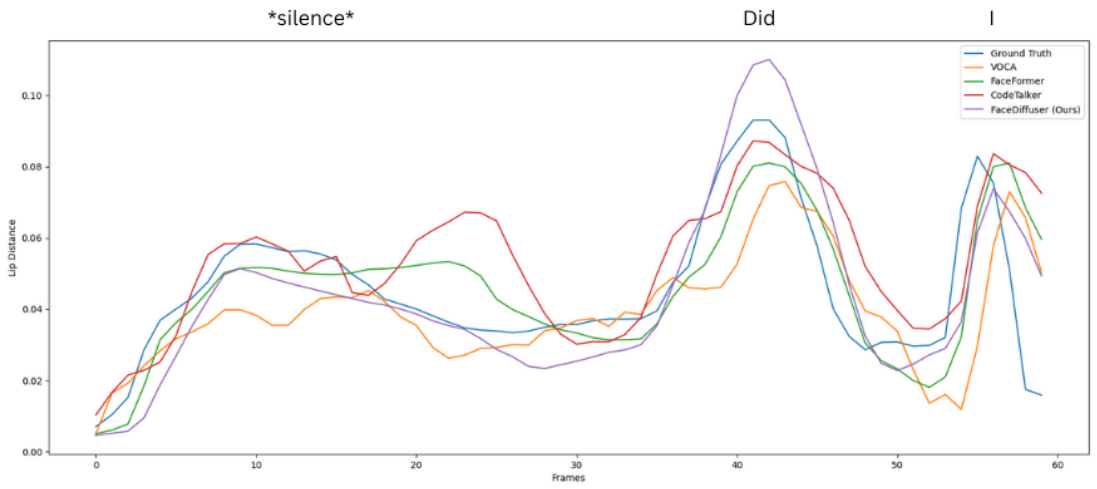


Figure 18: Lip distance evolution for the sequence starting with silence. We can see that our model is robust to noise and does not produce extreme lip movements when there is no speech segment

each of the models we compare ourselves with. We plot one such sequence in Figure 18. A sequence that also includes silence was chosen so that our model's robustness to noise can be visualized as well. If we render the sequence, we can see that CodeTalker in particular produces jittery results when it comes to noisy audio, the lips are unnaturally moving as can also be seen from the graph. On the other hand, if we focus on the curve representing our model, we can see a more smooth transition, with noise not having an impact on our results. We attribute this to the choice of using HuBERT as an audio encoder, which outperforms wav2vec 2.0 which is used by CodeTalker and FaceFormer. Another reason why CodeTalker performs worse in conditions of silence might be attributed to the architecture design and the scarcity of actual silence in the dataset.

Method	BIWI-Diversity (x 10^{-3} mm) ↑	VOCASET-Diversity ↑ (x 10^{-4} mm)	Multiface-Diversity ↑ (x 10^{-3} mm)
VOCA	7.8507	**6.5875**	0.0529
FaceFormer	5.9201	4.8669	1.4745
CodeTalker	0.0003	2.7711	1.3423
FaceDiffuser	**9.2459**	4.4235	**1.6080**

Table 12: Diversity metrics computed over the Test-B splits of unseen subjects. Best results are marked as **bold**

7.3 Blendshape-FaceDiffuser Results

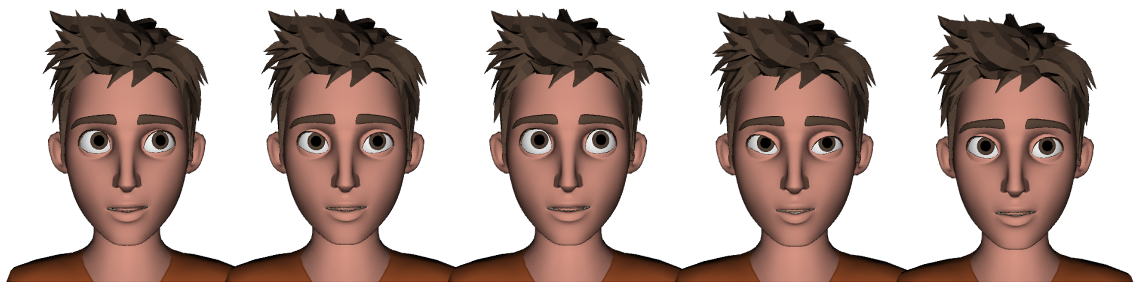
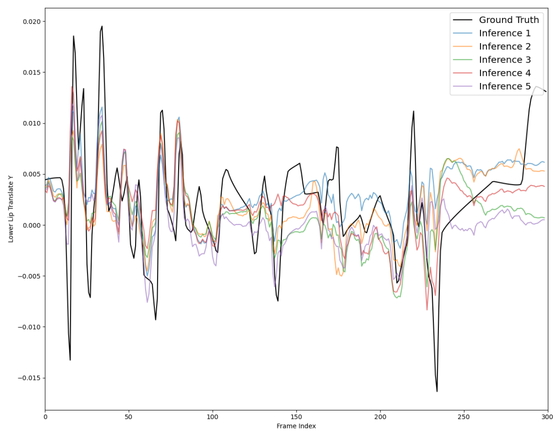
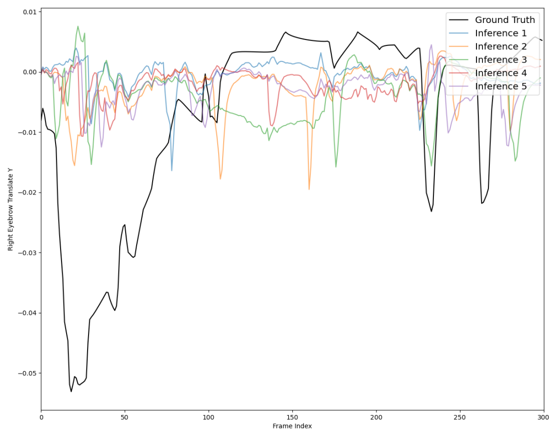


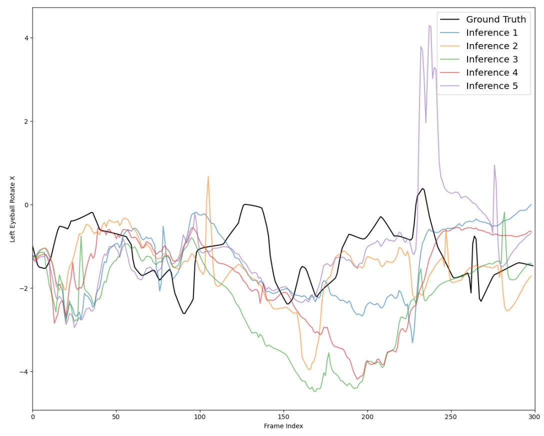
Figure 19: FaceDiffuser is non-deterministic and is able to generate novel animations every time. Pictured here is the same frame generated by our model. We can see that the mouth has a very similar shape, our model being consistent with the speech, while other non-speech related features, like gaze or upper face expression, differ between generations.



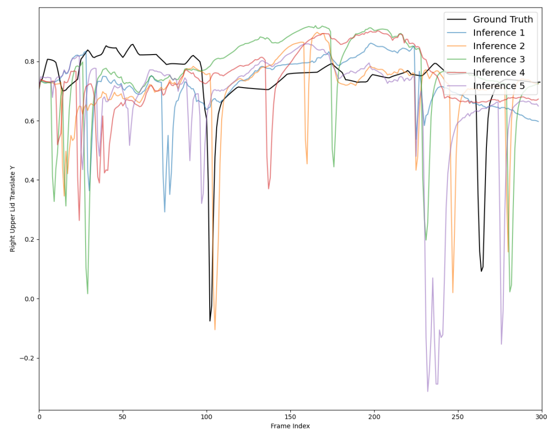
(a) Lower lip facial control evolution.



(b) Eyebrow facial control evolution.



(c) Eyeball rotation facial control evolution.



(d) Upper lid facial control evolution.

Figure 20: Animation graphs of some facial controls (i.e.- lower lip, eyebrow, gaze, upper eyelid) of the DaMM dataset. We synthesise animation data multiple times using the same audio and plot the graphs together with the ground truth. The black lines depict the ground truth whereas different coloured lines depict different inference results for the same input audio. It is evident that our approach produces lip control values similar to the ground truth as seen in Fig. 20a while encouraging diversity for the other facial controls as seen in Fig. 20b, Fig. 20c and Fig. 20d.

Method	MCE ↓	LCE ↓	C-FDD ↓
B-Face	**2.6963**	**1.5924**	2.0553
FaceDiffuser	3.4479	1.6671	**1.7752**

Table 13: Objective evaluation results computed over the Multiface-Test set. Best results are marked as **bold**

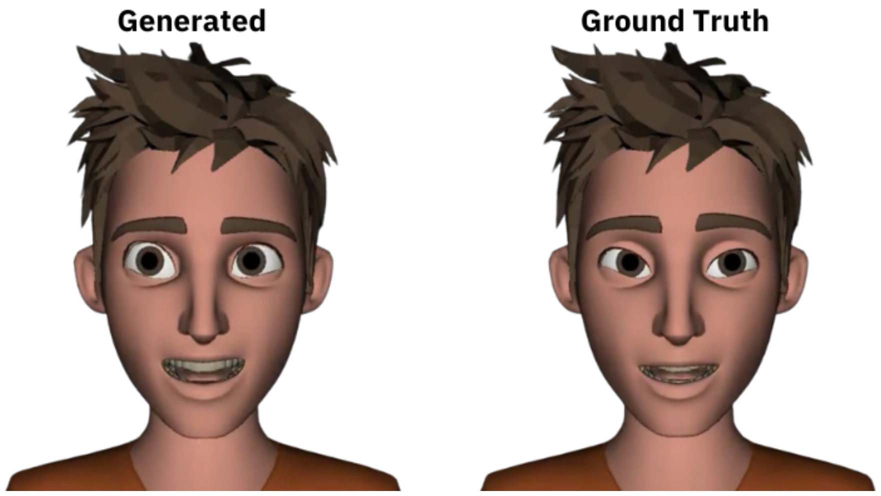


Figure 21: A frame sampled from laughing sequence.

In this subsection, we report the objective results of our second version of the model, B-FaceDiffuser. Since there are no state-of-the-art methods to directly compare our model with, we report the results obtained with 2 versions of the model, namely our finalised **B-FaceDiffuser** architecture and a version that does not make use of the diffusion process, which we call **B-Face**. The results can be seen in Table 13.

Analysing these objective metrics we can see that or diffusion model performs slightly worse on the metrics testing the similarity to the ground truth animation. This can be attributed to the non-deterministic nature of diffusion models. Expressions that are uncorrelated to speech might occur at any time during the generated sequence, increasing the differences between ground truth and prediction. A clear example of that can be the eye blinking. The chance of it occurring at the same time in both sequences is low, therefore increasing this error. An example of the diversity generated by our model can be seen in Figure 19, where we have rendered the same animation frame obtained by sampling the model 5 different times. As we can see, the upper face differs between different inferences, especially when it comes to the eyes. These results are further sustained by the graphs in Figure 20, where we can see the evolution of different control values during the same sequence. Just like before, these were obtained by sampling our trained B-FaceDiffuser 5 times with the same audio input, but different starting x_T. We can see that the lip control, which is highly correlated with the audio input, follows the trends of the ground truth and is consistent between different inferences, especially seen in the peaks and valleys in the graph. On the other hand, in the graphs for the controls that are not as closely related to speech, we can see a higher variation among the different inferences. This is especially clear in the upper lid graph (Figure 20d), where valleys corresponding to eye blinks occur at different times in all of the sequences.

Furthermore, we can argue that if the obtained results are more static, as is the case of our baseline model, the chance of having a lot of differences with the ground truth is also lower, translating to a lower error. On the other hand, we can see that the B-FaceDiffuser model produces animations that are more dynamic in the upper face while still maintaining accurate lipsync. Another advantage of the B-FaceDiffuser over the baseline is that it produces results that are non-deterministic, allowing the generation of a high variety of facial animation sequences from the same audio input.

7.4 Qualitative Analysis

Competitor	BIWI			VOCASET		
	Lip-Sync	Realism	Appropriateness	Lip-Sync	Realism	Appropriateness
VOCA	77.27 %	69.32 %	79.55 %	76.83 %	78.05 %	78.05 %
FaceFormer	31.03 %	34.48 %	37.93 %	49.38 %	46.91 %	51.85 %
CodeTalker	44.94 %	44.94 %	41.57 %	27.16 %	26.40 %	27.16 %
Ground Truth	38.71 %	34.41 %	36.56 %	23.81 %	33.33 %	27.38 %

Table 14: V-FaceDiffuser User Study Results. We report the frequency of users selecting our model in terms of the 3 criteria presented to them.

Competitor	Lip-Sync	Realism	Appropriateness
B-Face	63.77 %	66.67 %	65.94 %
Ground Truth	18.48 %	27.17 %	20.65 %

Table 15: UUDaMM User Study Results. We report the frequency of users selecting our model in terms of the 3 criteria presented to them

User Study Demographics In total, we managed to gather 83 responses spread among the 3 experiments as follows: 31 for the BIWI survey, 29 for the VOCASET survey, and 23 for the UUDaMM survey. An additional 3 responses were discarded due to the participants failing the sanity checks. The surveys were distributed online to different groups that voluntarily took part in the study without being remunerated, managing to collect 38 responses out of the total. The additional 45 responses were collected through the Prolfic [6] platform, which facilitates response collection by allowing participants to take part in surveys and be paid afterwards. We fairly compensated the participants by awarding them the equivalent of $9£/h$. Each survey was distributed to a different group, however, there might exist some overlaps of the same person taking part in multiple. Most of the participants are adults or young adults, with the following age distribution: 65.85% in the 18-25 age group, 25.61 in the 26-35 age group, and 8.54% in the 36-45 age group. Looking at the gender distribution, 36.59% of the participants identify as female, 54.88% as male, and 8.53% as non-binary/other.

In regards, to the user familiarity with the subject, we computed the average reported familiarity of the 3 areas that were questioned and we obtained 2.71 average familiarity with virtual humans, 3.52 average familiarity with 3D animated movies, and 4.06 average familiarity with video games.

In Tables 14 and 15 we report the frequency of favourable choices when our model was preferred over the competitor. As we can see from the first table, our model outperforms VOCA in both settings, while with the other methods, there is a closer battle.

When looking at the results obtained on the BIWI dataset, we can see that our model was chosen just slightly fewer times than CodeTalker, being more clearly outperformed by FaceFormer. In the case of VOCASET, our model performs nearly identically to FaceFormer, while there is a clear better preference for CodeTalker. By looking at the example provided in Table 9 of mean motion over a sequence conditioned on 2 different subjects we can that our model produces highly different results, with very little motion in one case, and more motion closer to the ground truth in the other case. When comparing this to the 2 sequences produced by CodeTalker, we can see that they are nearly identical. This works to the advantage of CodeTalker, by not having a strong style of conditioning, the model is able to always generate animations with a lot of movement, no matter what the conditioning subject is. On the other hand, our results are strictly bound by the latent expression learned from the specific subject. Since in the user study, we randomly pick the conditioning subject, it's highly probable that we generate some examples conditioned on a less expressive subject, ensuring our loss against the more vivid animation generated by the competitor. This can also explain why our model seems to perform better or similar to FaceFormer in one study, while in the other, the roles between FaceFormer and CodeTalker are switched.

[6] https://www.prolific.co/

8 Ablation Study

To analyse the effects of the different components of our final architecture, we test different configurations of it, by either removing or changing different parts. This gives us an insight into how our choices have influenced the final results and whether the chosen modules perform better than others used in similar works.

We will ablate on the 3 main parts of the architecture. Additionally, we also ablate the use of the dropout term introduced by randomly masking the audio inputs.

- **Audio Encoder.** We test the benefits of using the HuBERT audio encoder by trying different ways to encode audio. Based on previous work, we will replace the pre-trained HuBERT audio encoder with wav2vec and quantitatively assess the results.
- **Face Decoder.** The next element of our architecture we ablate on is the facial decoder. This part of the model is responsible for generating the animation frames based on the input modalities, which in our case are: audio, diffusion timestep, and diffused motion.
- **Diffusion Model.** Next, we ablate on our contribution of including a diffusion model into the architecture. To do this, we train a model with and without diffusion and analyse the results. Additionally, we also test different configurations for the noise encoder component.
- **Audio Dropout.** Finally, we try different ways of including the audio dropout in the architecture as well as eliminating it. We report the results in terms of objective metrics.

These different configurations are evaluated objectively based on the metrics defined in the previous subsections.

8.1 Audio Encoder Ablation

Audio Encoder	MVE $\downarrow$(x 10^{-3} mm)	LVE $\downarrow$(x 10^{-4} mm)	No. of Parameters	Training Time (min)
Wav2Vec2	7.1920	4.8462	203705704	68.34
HuBERT CNN encoder frozen	6.9063	4.4561	203705704	50
HuBERT trainable	7.0510	4.8203	207906152	50
HuBERT frozen	7.1136	4.7413	118256232	40
HuBERT (Ours)	6.9831	4.4696	189134952	47.5

Table 16: HuBERT audio encoder ablation study results

We have seen from previous work [25, 19, 57] that using large audio models has a clear benefit over extracting audio features such as MFC. Building upon that, we follow in the footsteps of *Haque et al*[25] and incorporated the pre-trained HuBERT model. Based on their extensive study of the use of different configurations of the model, we have also chosen to freeze the CNN feature extractor along with the first 2 transformer encoder layers in the HuBERT model. We keep all of the other layers trainable and update the weights accordingly during training. As was shown in the previous sections, our model was able to beat state-of-the-art techniques when it comes to lip-sync error so we argue that the introduction of the HuBERT model was beneficial for our architecture. To test if this was the case or if other choices have influenced our good results, we replace the pre-trained HuBERT model with a pre-trained Wav2Vec model, similar to what is used by both Codetalker [57] and Faceformer [19]. Additionally, we test different initialisations for the HuBERT audio encoder. The other parts of the architecture are kept exactly the same, as well as the chosen hyperparameters.

The audio encoder configurations we test are as follows:

- Wav2Vec2. [4] For this configuration, we adopt the wav2vec2 encoder as it was used in the Faceformer and Codetalker papers. Just like them, we also keep the layers of the encoder trainable with the exception of the CNN encoder layer.
- HuBERT frozen CNN encoder. We also try the same frozen configuration on our HuBERT encoder.
- HuBERT no frozen layers. In this case, we keep all of the layers of the model trainable.

- HuBERT all frozen layers. We freeze all of the layers of the model and don't update their weights during training.
- HuBERT with 2 frozen layers (Ours). This is our chosen configuration based on how it was proposed by *Haque et al.* [25].

Analysing the results in Table 16 we can see that the choice of the HuBERT model is better in all tested configurations against Wav2Vec2. Furthermore, we can see that by keeping the model fully trainable or freezing all of the layers we get the worst results. The best results we get are when we freeze the CNN encoder. Furthermore, by also freezing the first 2 transformer layers, we get very similar results to not freezing them. However, when we analyse the results visually, we see a very slight improvement in lip closures. That coupled with the fact that the chosen model has fewer parameters and therefore more compact and faster to train, we conclude our choice of the audio encoder to be the best out of the tested ones.

8.2 Facial Decoder Ablation

Decoder Type	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)	# Parameters	Training Time (min)
GRU	6.8088	4.2985	3.9100	189134952	≈ 1.12
RNN	7.0833	4.7870	4.0690	185985128	≈ 1.12
Transformer - Teacher Forcing	9.9767	10.1300	4.8587	194132574	≈ 1.34
Transformer - Autoregressive	7.0213	4.6941	4.3272	194132574	≈ 5.00

Table 17: Evaluation metrics computed over the test results of different facial decoder configurations

Next, we ablate on our choice of animation decoder. This part of the architecture is also important as it has the task of generating the final facial animations from the speech and noise distributions it received. For our results to be temporally stable, we have to choose a sequence-to-sequence architecture that has an awareness of past motions. We compare the results in terms of the pre-defined objective metrics as well as training time. The results of these different configurations can be seen in Table 19. As we can see, our chosen GRU decoder performs the best out of the tested configurations, resulting in a significantly lower LVE value.

We mention that these results were obtained before finalising the architecture and this ablation also acted as a method for choosing the final architecture for the face decoder. The final results of the chosen face decoder along with other choices we made can be seen in the state-of-the-art comparison table Table 9.

8.3 Ablation on the noise encoder

Noise Encoder	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)
MLP	7.1166	4.8283	4.7027
Conv1D + MaxPool (Ours)	6.8773	4.4353	3.9994
Conv1D + AvgPool	6.8675	4.4080	4.2663

Table 18: Results of different types of encoder for the noise

In the V-FaceDiffuser version of the model, we introduced a noise encoder with the task of reducing the high dimension of the noise input to a more condensed latent representation. Our final choice of noise encoder was represented by a sequence composed of a linear layer, 1D convolution and max pooling layer. In this subsection, we explore other configurations for this noise encoder and see whether they perform better or worse. Our chosen configurations are:

- **MLP.** A series of 2 fully connected layers.
- **Conv1D + MaxPool (Ours)** One fully connected layer followed by a single-dimensional convolution and a max pooling layer.

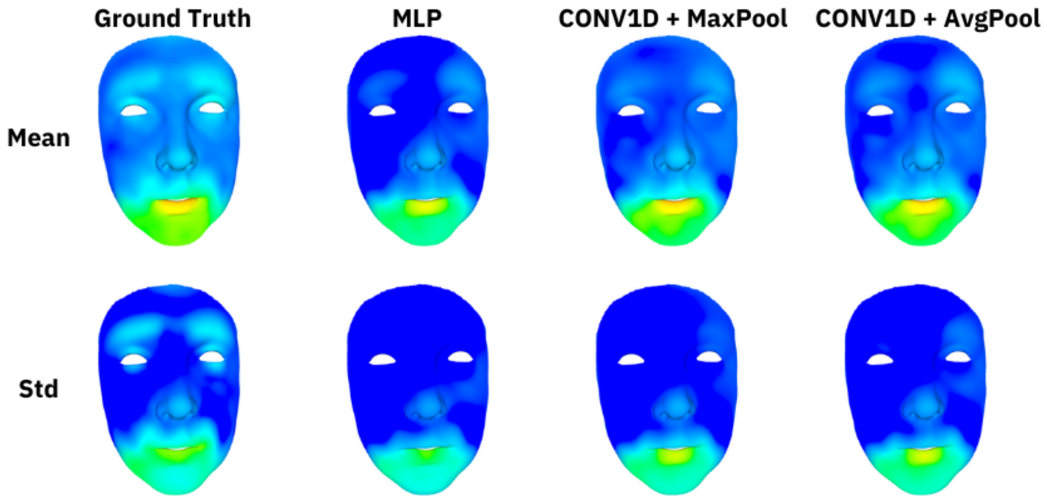


Figure 22: Mean motion maps for a sequence in the test set computed with 3 different configurations of the noise encoder. Results are plotted on a colour scale that goes from dark blue (no motion) to red (a lot of motion).

- **Conv1D + AvgPool** One fully connected layer followed by a single-dimensional convolution and an average pooling layer.

We consider this part of the architecture to be important since our model needs to be able to learn the latent expression space from the noise. If the noise encoding is not able to capture the particularities of the noise, the model will also learn nothing from it and produce animations that are highly conditioned by the audio input.

By analysing the results in both the table and the heatmaps we can see a trend in the expressivity of the animations, viewed as a higher activation of the face, can be seen the more complex the noise encoder is. Some more complex configurations were also tried, showing better results in terms of the mean motion of the face. However, this came at the detriment of ground truth fidelity, causing the objective metrics to have worse values. To this end, we decided to keep a simpler implementation of the noise encoder as it proves to be acceptable in producing quality results and we leave the engineering of potentially better noise encoder as a possibility for future work.

Out of the tested configurations, our choice performed the best. We can also notice that the Conv1D approaches are performing better than the MLP, both by looking at the results in Table 18 and the visual representation in Figure 22. Furthermore, the method using max pooling performs slightly better than the average pooling one. We can motivate this by the fact that more extreme relevant features from the input noise might get lost when an average is computed over them.

8.4 Ablation on the diffusion component

In order to test the benefits of adding diffusion to our model, we will exclude the diffusion part from the architecture and train a model on only speech features. This test will give us an idea of how the diffusion process influences the generated results. We evaluate these results both quantitatively and qualitatively.

To obtain these results we train an additional model in which we only use audio input as the features that are fed into the model and trained on. In Table 19 we report the objective results of this training, showing a noticeable improvement in our approach over the no diffusion one. Furthermore, these results are also solidified by the perceptual study we conducted. During the survey, we asked users to choose between pairs of animations that were generated with and without diffusion for the model trained on the UUDaMM dataset. As we can see in Table 15, there was a clear preference for our model over the baseline version of it.

Looking at the visual representations of the average motions over a sequence, we can also identify that our model generates more vivid motions. One clear difference in the case of the B-FaceDiffuser is its ability to generate more realistic-looking blinking and eye movement. While we can see the

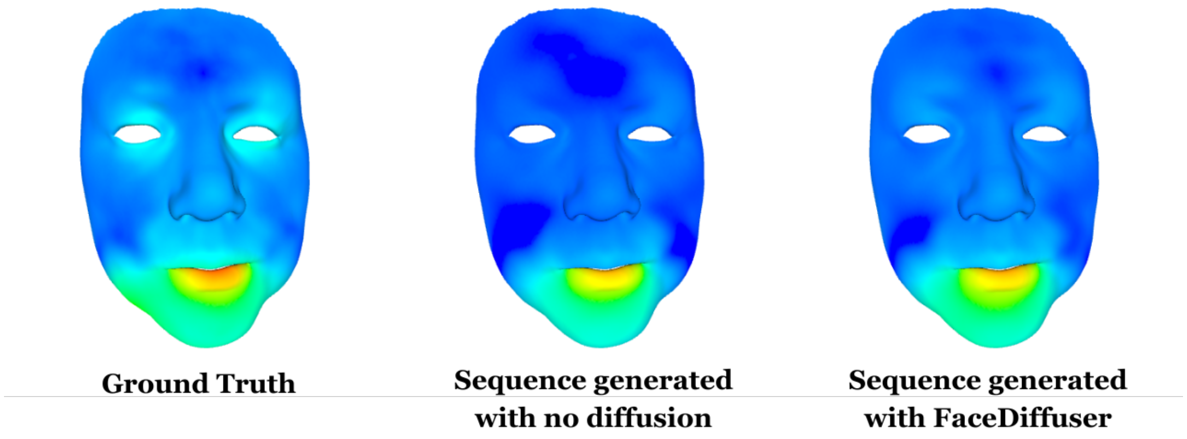

Ground Truth
Sequence generated with no diffusion
Sequence generated with FaceDiffuser

Figure 23: Mean motion maps for a sequence in the test set. Results are plotted on a colour scale that goes from dark blue (no motion) to red (a lot of motion).

Model	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)	# Parameters	Training Time (min)
w/o Diffusion	6.8833	4.5870	4.6690	185985128	≈ 67
FaceDiffuser	6.8088	4.2985	3.9100	189134952	≈ 67

Table 19: Evaluation metrics computed over the test results of different facial decoder configurations

eyes closing in the baseline version of the model, the process happens slowly and does not resemble actual human blinking. In the case of B-FaceDiffuser blinking happens more naturally along with more movements of the eyes. While these eye movements might generally be considered better than completely static eyes, we did notice that their frequency is significantly higher than in the same ground truth sequence. We think that this might have contributed to the low preference for our generated animations in comparison to the ground truth animations.

Additionally, we also show a visual comparison of the mean motion in a sequence from the BIWI-Test-B generated by our FaceDiffuser and a variant of it without the diffusion component. Here as well we can see that the generated animations are slightly more dynamic, the motions being closer to those that are present in the ground truth.

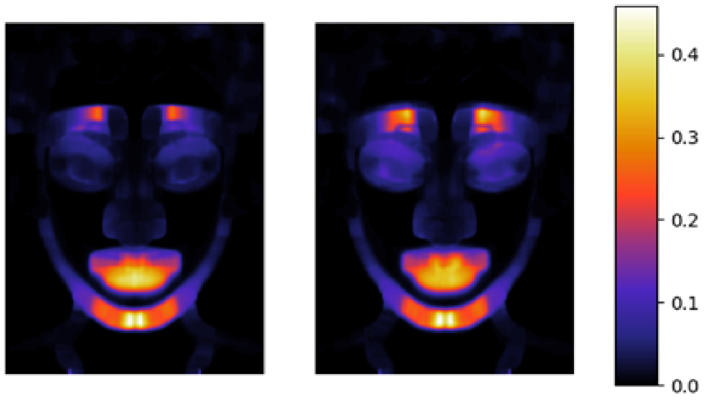


Figure 24: Mean motion maps for the same sequence generated without (left) and with (right) diffusion

8.5 Ablation on the audio masking component

Similar to other approaches using diffusion models, we randomly mask the generative condition (audio in our case) in order to help the model learn the latent expression space and not overfit the audio.

In this subsection, we present the results of excluding this dropout introduced by masking the audio condition. The results can be seen in Table 20.

We see that objectively our model performs better when this dropout mechanism is used. Furthermore, by analysing the animations produced with the 2 different approaches, we can see a noticeable improvement in the upper-face motion of the face, the model being able to better disentangle the speech-related expressions from the non-speech-related ones.

Condition Dropout	MVE ↓(x 10^{-3} mm)	LVE ↓(x 10^{-4} mm)	FDD ↓ (x 10^{-5} mm)
without	6.9070	4.5402	4.1132
with	6.8088	4.2985	3.9101

Table 20: Audio input dropout ablation results

9 Extra Use Cases

9.1 Animation Editing

By using a generalised animation encoding such as the ARKit blendshape expression space, we are able to easily edit the resulting animations to better emulate our desired results. This can be done by simply updating the animation curves as can be seen in Figure 25. A simple change like moving one of the curves upwards would change the entire sequence expression and could be used to generate various expressions. For example, an animator might choose to generate more obvious mouth movements by applying a filter over the *mouthOpen* blendshape, which would make mouth opens and closures more extreme. Considering that accurate lip movements are automatically generated by a machine-learning model like ours, the animators then have free-range customising those animations to better fit their desires.

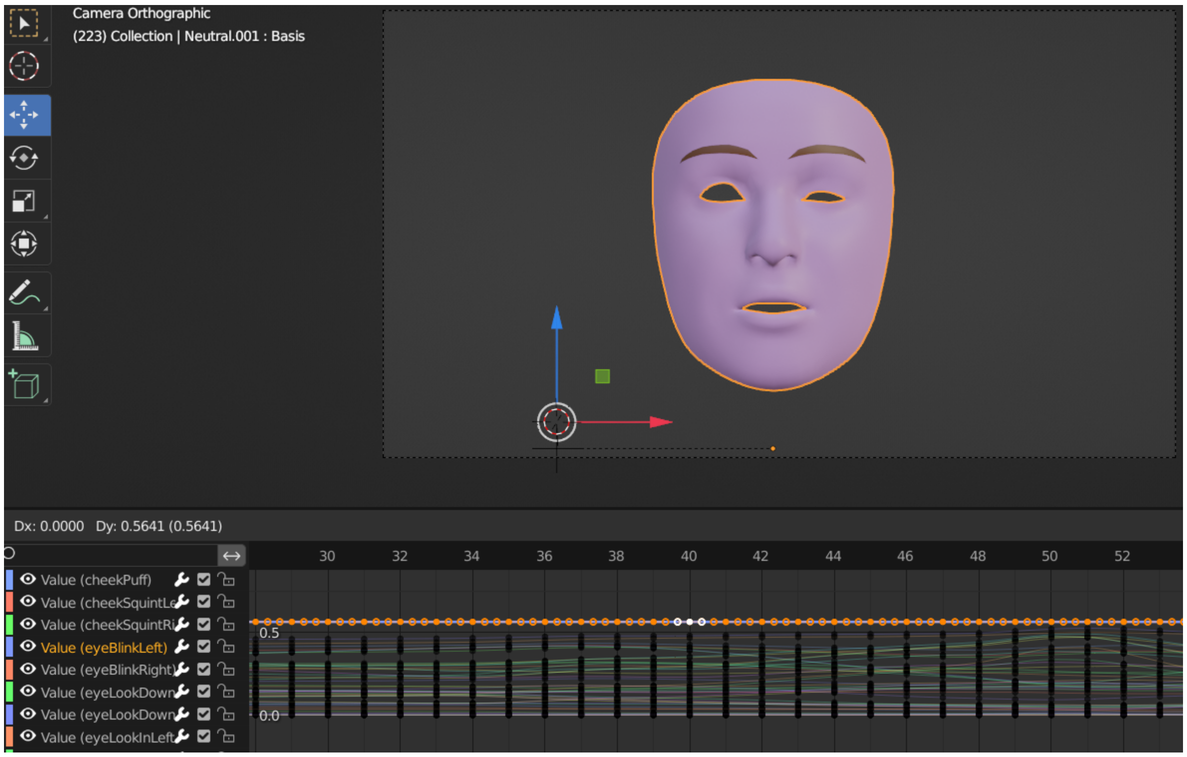


Figure 25: An example of how the generated animation can be edited by an animator after it was automatically generated

9.2 Animation Transfer

Another identifiable benefit of generating ARKit blendhshape animations is the transferability of such animations between different characters. The only requirement for the animation to be transferable is for the target character to also have defined the ARKit blendshapes. No additional retargeting steps are required and therefore the transfer is easy even for novice animators. This opens a wide range of opportunities as there is a vast array of different faces that can be animated by using our model.

Even more complex characters like the highly hyper-realistic Metahumans can be animated with this approach, as can be seen in Figure 26. However, we consider that the ARKit animations fare better with more stylized characters, as they can look unnatural on high-definition characters, causing the effect known as the uncanny valley.

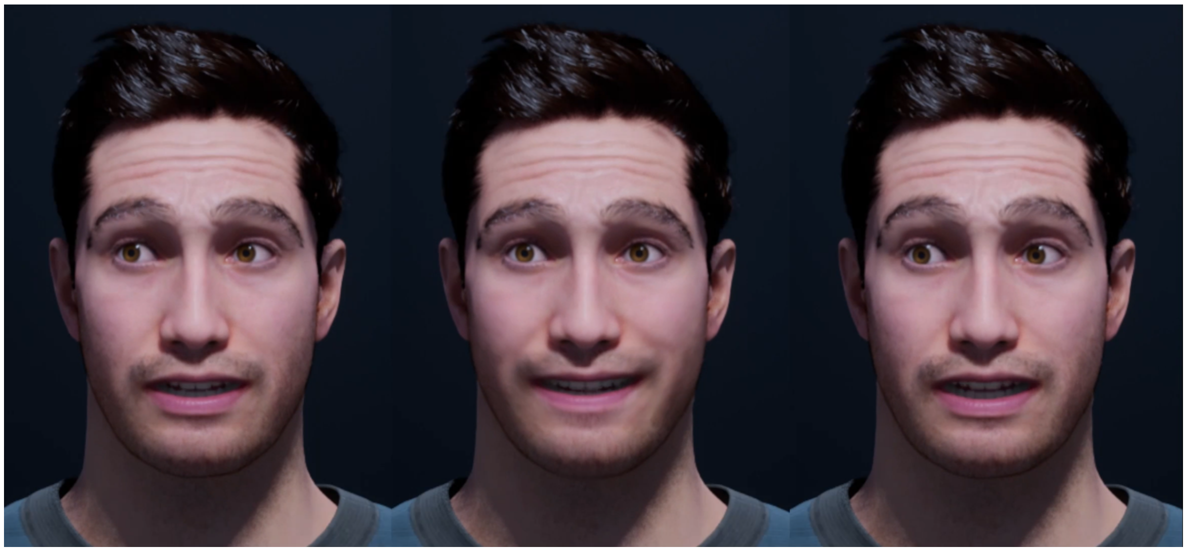


Figure 26: By using the BEAT dataset FaceDiffuser is able to generate ARKit blendshape animations that can be used to animate a large variety of 3D characters, including hyper-realistic Metahumans.

10 Limitations and Future Work

As we have seen from the previous sections, we were able to successfully train a deep-learning model for the task of generating facial animations, both as vertex movements and blendshape values. Even though our model performs objectively better than state-of-the-art techniques, the perceptual evaluation shows that there is still room for improvement.

One of the most significant limitations we can identify is the size of the vertex-based datasets. We argue that the sequences are too short to capture the full range of expressions a person can have, therefore our model cannot learn enough from the data either. It would be interesting to see how our model performs against state-of-the-art techniques when a larger dataset is used, such as the full Multiface dataset. Furthermore, a dataset that includes multiple takes of the same sentence being spoken would allow a better analysis of the diversity capability of the model. By being able to compute the diversity in the training set we would have a better understanding of how the facial expressions of a person can vary. In the BIWI dataset this is available in the form of 2 takes per sentence, one neutral and one emotional, but a dataset with a higher number of takes per sentence would be more interesting to employ.

Additionally, in this project, we have chosen to focus on the introduced diffusion component and designed our experiments in such a way that would facilitate our comparisons with state-of-the-art techniques. Even though available in the BIWI dataset, none of our competitors has made use of emotion embedding. By looking at the work of *Haque et al.* [25] we have seen that such a direction can prove to be fruitful. Furthermore, a broader experimentation with multiple emotions, such as the ones available in the BEAT dataset could also be a good direction for future work. Another limitation our solution has is the longer inference time caused by the iterative sampling process of diffusion models. While this time can be reduced by trading off between quality and the number of inference steps, our solution is still not suitable for real-time applications.

Finally, during this project we have started to experiment with a 2-stage learning approach, first learning the latent expression space and then in the second stage learning to predict motions based on speech. While in our proposed solution we already have a way of learning a broader latent expression space thanks to using diffusion, we consider that a comparison of these 2 approaches might be interesting to explore. During our short experimentation, we had some initial findings that give an intuition of the possibilities of this approach, however, we think that a more in-depth analysis needs to be made before drawing some relevant conclusions, as well as a better model design. To give a starting point for possible future work, we shortly explain and show the design of our 2-stage approach in Appendix B. We have seen the success of methods such as CodeTalker and we think it would be a nice area to explore, especially by employing a novel diffusion autoencoder.

11 Conclusion

To conclude, we first refer back to the research questions we introduced at the beginning of this report and one by one answer them based on the findings obtained through our experiments and evaluations.

The first question was: *Does the integration of diffusion models into a speech-driven animation pipeline improve the quality of the results?*. Based on the results presented in the previous section we can conclude that diffusion models show real potential for the task of facial animation. As we have seen, our FaceDiffuser model outperforms state-of-the-art methods in terms of objective metrics. Not the same can be however said about the subjective evaluation of our proposed model. As we have seen with the introduction of the *Diversity* metric, our model produces a higher variation of animations across different subject embeddings, and we can argue that our model better learns the latent expression of the subjects present in the training set. We can attribute this to the diffusion mechanism. Since noised ground truth visual frames were also used as inputs for our model, the model was effectively able to better extract per-subject information and learn a stronger correlation between certain expressions and their corresponding subject. While this proved to objectively be beneficial, as we have seen with better values for lip-sync error as well as FDD, in the case of the perceptual study such a high variance might not always be good. We can see that in the case of the BIWI set, our competitors all have lower intra-sequence diversity, the lowest being attributed to CodeTalker, which has a near-insignificant diversity value of just $0.0003 \cdot 10^{-3} mm$. With such a low diversity, we can conclude that the generated sequences with different style embeddings are nearly identical. On the other hand, when different subject conditionings are used, our model produces animations that are more diverse, ranging from more neutral to very expressive. Additionally, through one of the surveys we did a direct perceptual comparison between animations generated with FaceDiffuser and a baseline model without diffusion. The results showed a clear preference for the FaceDiffuser results, strengthening our claim that the inclusion of the diffusion method was beneficial.

Next, we aimed to answer the question of whether or not our model is better than state-of-the-art techniques. By analysing the results, we consider that this question does not have a clear definitive answer, as we see a contradiction between the objective and subjective evaluations. While we managed to beat every other competitor objectively, in terms of the subjective evaluation our method was not considered the best and fell somewhere in the middle. As a possible interpretation for this situation, we refer back to the previous paragraph and conclude that a different type of user study would have to be designed in order to be able to give a definitive answer. To summarise, our model managed to produce results with lower errors than state-of-the-art, producing more accurate lip-sync as can be concluded from the low lip-sync errors our results have. In terms of perceptual evaluation, we see a preference for our model against VOCA, while against FaceFormer and CodeTalker, our model is deemed to be similar or slightly worse.

Our final question was about our ability to design a data-driven model for the facial animation generation of rigged characters. This question we can happily mark as a success, being able to generate such animations for two types of rigged characters, namely by driving rig control values or blendshape weights. Additionally, we also show the non-determinism our model allows by generating different sequences from the same input audio. Furthermore, we show that this approach allows facile integration into animation pipelines, proving to be a method that indicates definite potential for the future of facial animation in both movies and video games.

To summarise our contributions, we have successfully integrated the diffusion process into a generative deep neural network trained to output facial animations conditioned on speech and style embedding. We showed that our model is able to learn a stronger style embedding, producing a higher diversity of motions between different style conditions than the competitors. With the help of a user study, we showed that our model outperforms one of the state-of-the-art methods while showing slightly worse performance than the other 2 competitors. We identified a possible cause for this in the stronger learned style embeddings, our method putting higher weight on the style. Furthermore, we also noticed a perceptual preference for the FaceDiffuser against a baseline that does not use diffusion, showing the benefits and potential of diffusion models for animation synthesis tasks. Moreover, through an extensive ablation study, we solidified our design choices, showing the benefits of different components of the architecture.

Finally, we presented extra uses for our proposed method, showing how it can be integrated into an animation pipeline and providing possible directions for future work.

References

[1] C. Ahuja, S. Ma, L.-P. Morency, and Y. Sheikh. To react or not to react: End-to-end visual pose forecasting for personalized avatar during dyadic conversations. In *2019 International conference on multimodal interaction*, pages 74–84, 2019.

[2] I. Albrecht, J. Haber, K. Kahler, M. Schroder, and H.-P. Seidel. " may i talk to you?:-)"-facial animation from text. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.*, pages 77–86. IEEE, 2002.

[3] M. V. Aylagas, H. A. Leon, M. Teye, and K. Tollmar. Voice2face: Audio-driven facial and tongue rig animations with cvaes.

[4] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[5] U. Bhattacharya, N. Rewkowski, A. Banerjee, P. Guhan, A. Bera, and D. Manocha. Text2gestures: A transformer-based network for generating emotive body gestures for virtual agents. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 1–10. IEEE, 2021.

[6] D. Bigioi, S. Basak, H. Jordan, R. McDonnell, and P. Corcoran. Speech driven video editing via an audio-conditioned diffusion model. *arXiv preprint arXiv:2301.04474*, 2023.

[7] S. Bilakhia, S. Petridis, A. Nijholt, and M. Pantic. The mahnob mimicry database: A database of naturalistic human interactions. *Pattern recognition letters*, 66:52–61, 2015.

[8] Y. Chai, Y. Weng, L. Wang, and K. Zhou. Speech-driven facial animation with spectral gathering and temporal attention. *Frontiers of Computer Science*, 16(3):1–10, 2022.

[9] L. Chen, Z. Wu, J. Ling, R. Li, X. Tan, and S. Zhao. Transformer-s2a: Robust and efficient speech-to-animation. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7247–7251. IEEE, 2022.

[10] M. M. Cohen and D. W. Massaro. Modeling coarticulation in synthetic visual speech. In *Models and techniques in computer animation*, pages 139–156. Springer, 1993.

[11] E. Cosatto, G. Potamianos, and H. P. Graf. Audio-visual unit selection for the synthesis of photo-realistic talking-heads. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No. 00TH8532)*, volume 2, pages 619–622. IEEE, 2000.

[12] D. Cudeiro, T. Bolkart, C. Laidlaw, A. Ranjan, and M. J. Black. Capture, learning, and synthesis of 3d speaking styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10101–10111, 2019.

[13] R. Dabral, M. H. Mughal, V. Golyanik, and C. Theobalt. Mofusion: A framework for denoising-diffusion-based motion synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9760–9770, 2023.

[14] S. Dahmani, V. Colotte, V. Girard, and S. Ouni. Conditional variational auto-encoder for text-driven expressive audiovisual speech synthesis. In *INTERSPEECH 2019-20th Annual Conference of the International Speech Communication Association*, 2019.

[15] C. Du, Q. Chen, T. He, X. Tan, X. Chen, K. Yu, S. Zhao, and J. Bian. Dae-talker: High fidelity speech-driven talking face generation with diffusion autoencoder. *arXiv preprint arXiv:2303.17550*, 2023.

[16] P. Edwards, C. Landreth, E. Fiume, and K. Singh. Jali: an animator-centric viseme model for expressive lip synchronization. *ACM Transactions on graphics (TOG)*, 35(4):1–11, 2016.

[17] S. E. Eskimez, R. K. Maddox, C. Xu, and Z. Duan. Generating talking face landmarks from speech. In *International Conference on Latent Variable Analysis and Signal Separation*, pages 372–381. Springer, 2018.

[18] S. E. Eskimez, R. K. Maddox, C. Xu, and Z. Duan. Noise-resilient training method for face landmark generation from speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:27–38, 2019.

[19] Y. Fan, Z. Lin, J. Saito, W. Wang, and T. Komura. Faceformer: Speech-driven 3d facial animation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18770–18780, 2022.

[20] G. Fanelli, J. Gall, H. Romsdorfer, T. Weise, and L. Van Gool. A 3-d audio-visual corpus of affective communication. *IEEE Transactions on Multimedia*, 12(6):591–598, 2010.

[21] Y. Feng, H. Feng, M. J. Black, and T. Bolkart. Learning an animatable detailed 3d face model from in-the-wild images. *ACM Transactions on Graphics (ToG)*, 40(4):1–13, 2021.

[22] O. Fried, A. Tewari, M. Zollhöfer, A. Finkelstein, E. Shechtman, D. B. Goldman, K. Genova, Z. Jin, C. Theobalt, and M. Agrawala. Text-based editing of talking-head video. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[23] U. K. Goyal, A. Kapoor, and P. Kalra. Text-to-audiovisual speech synthesizer. In *International Conference on Virtual Worlds*, pages 256–269. Springer, 2000.

[24] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[25] K. I. Haque and Z. Yumak. Facexhubert: Text-less speech-driven e(x)pressive 3d facial animation synthesis using self-supervised speech representation learning. *arXiv*, 2023.

[26] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[27] 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, 2021.

[28] Y. Huang and S. M. Khan. Dyadgan: Generating facial expressions in dyadic interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 11–18, 2017.

[29] P. Jonell, T. Kucherenko, G. E. Henter, and J. Beskow. Let's face it: Probabilistic multi-modal interlocutor-aware generation of facial gestures in dyadic settings. In *Proceedings of the 20th ACM International Conference on Intelligent Virtual Agents*, pages 1–8, 2020.

[30] T. Karras, T. Aila, S. Laine, A. Herva, and J. Lehtinen. Audio-driven facial animation by joint end-to-end learning of pose and emotion. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

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

[32] 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.

[33] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4d scans. *ACM Trans. Graph.*, 36(6):194–1, 2017.

[34] H. Liu, Z. Zhu, N. Iwamoto, Y. Peng, Z. Li, Y. Zhou, E. Bozkurt, and B. Zheng. Beat: A large-scale semantic and emotional multi-modal dataset for conversational gestures synthesis. In *European conference on computer vision*, 2022.

[35] R. Maatman, J. Gratch, and S. Marsella. Natural behavior of a listening agent. In *International workshop on intelligent virtual agents*, pages 25–36. Springer, 2005.

[36] E. Ng, H. Joo, L. Hu, H. Li, , T. Darrell, A. Kanazawa, and S. Ginosar. Learning to listen: Modeling non-deterministic dyadic facial motion. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[37] T. Nikolov, M. Tsakov, C. Gruter, and Z. Yumak. Utrecht university dyadic multi-modal motion (uudamm) dataset. 2021.

[38] B. Nojavanasghari, Y. Huang, and S. Khan. Interactive generative adversarial networks for facial expression generation in dyadic interactions. *arXiv preprint arXiv:1801.09092*, 2018.

[39] G. Papamakarios, E. T. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

[40] H. X. Pham, S. Cheung, and V. Pavlovic. Speech-driven 3d facial animation with implicit emotional awareness: a deep learning approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 80–88, 2017.

[41] H. X. Pham, Y. Wang, and V. Pavlovic. End-to-end learning for 3d facial animation from speech. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, pages 361–365, 2018.

[42] K. Preechakul, N. Chatthee, S. Wizadwongsa, and S. Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10619–10629, 2022.

[43] A. Richard, C. Lea, S. Ma, J. Gall, F. De la Torre, and Y. Sheikh. Audio-and gaze-driven facial animation of codec avatars. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 41–50, 2021.

[44] A. Richard, M. Zollhöfer, Y. Wen, F. De la Torre, and Y. Sheikh. Meshtalk: 3d face animation from speech using cross-modality disentanglement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1173–1182, 2021.

[45] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[46] S. Schneider, A. Baevski, R. Collobert, and M. Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[47] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[48] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews. A deep learning approach for generalized speech animation. *ACM Transactions on Graphics (TOG)*, 36(4):1–11, 2017.

[49] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[50] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, and M. Nießner. Neural voice puppetry: Audio-driven facial reenactment. In *European conference on computer vision*, pages 716–731. Springer, 2020.

[51] J. Tseng, R. Castellon, and K. Liu. Edge: Editable dance generation from music. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 448–458, 2023.

[52] 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.

[53] K. Vougioukas, S. Petridis, and M. Pantic. End-to-end speech-driven facial animation with temporal gans. *arXiv preprint arXiv:1805.09313*, 2018.

[54] K. Vougioukas, S. Petridis, and M. Pantic. Realistic speech-driven facial animation with gans. *International Journal of Computer Vision*, 128(5):1398–1413, 2020.

[55] A. Wang, M. Emmi, and P. Faloutsos. Assembling an expressive facial animation system. In *Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, pages 21–26, 2007.

[56] C.-h. Wuu, N. Zheng, S. Ardisson, R. Bali, D. Belko, E. Brockmeyer, L. Evans, T. Godisart, H. Ha, A. Hypes, et al. Multiface: A dataset for neural face rendering. *arXiv preprint arXiv:2207.11243*, 2022.

[57] J. Xing, M. Xia, Y. Zhang, X. Cun, J. Wang, and T.-T. Wong. Codetalker: Speech-driven 3d facial animation with discrete motion prior. *arXiv preprint arXiv:2301.02379*, 2023.

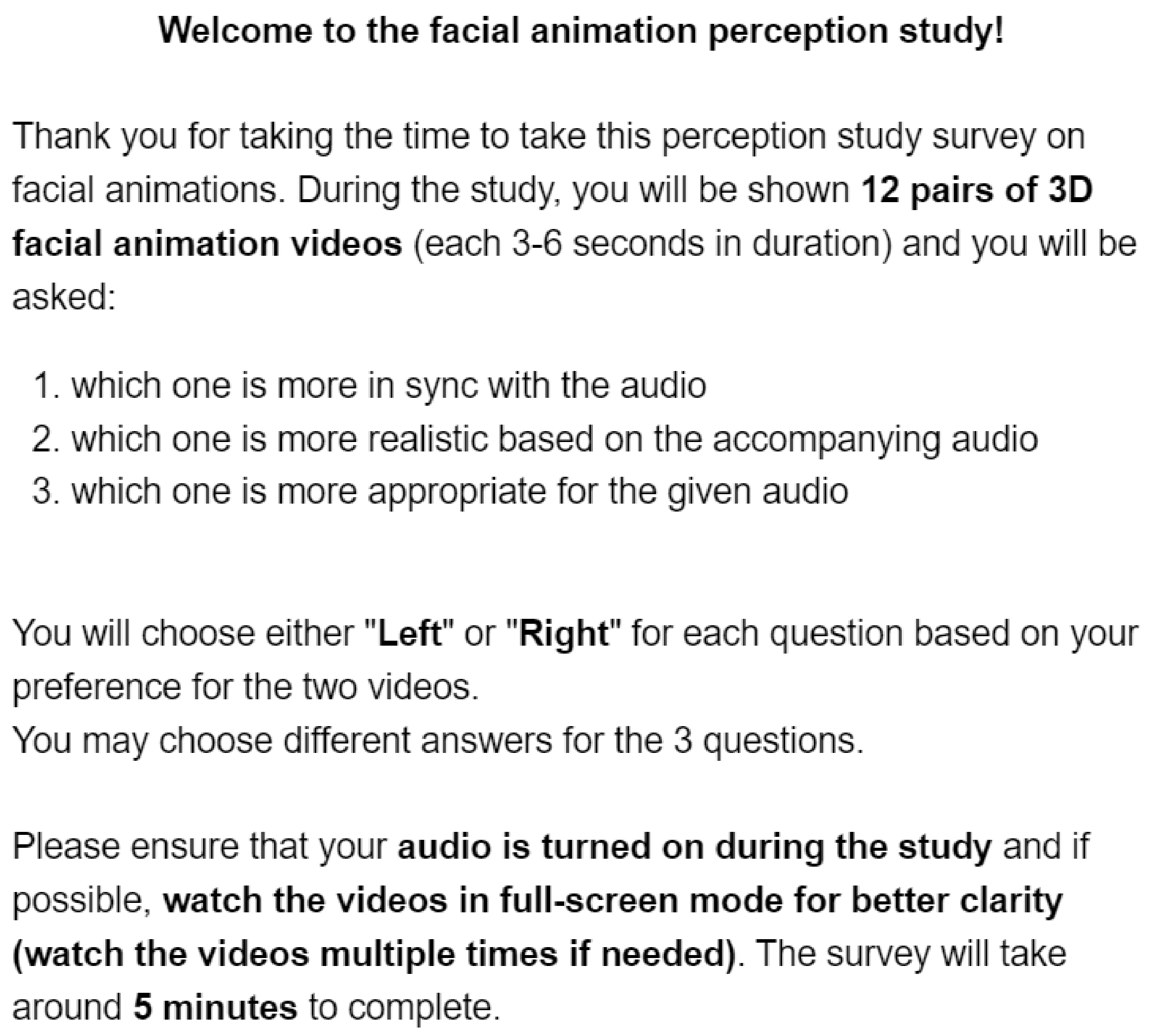
[58] S. Yang, Z. Wu, M. Li, Z. Zhang, L. Hao, W. Bao, M. Cheng, and L. Xiao. Diffusestylegesture: Stylized audio-driven co-speech gesture generation with diffusion models. *arXiv preprint arXiv:2305.04919*, 2023.

[59] T. Zhang and C.-C. J. Kuo. *Audio Feature Analysis*, pages 35–54. Springer US, Boston, MA, 2001.

[60] X. Zhang, L. Wang, G. Li, F. Seide, and F. K. Soong. A new language independent, photo-realistic talking head driven by voice only. In *Interspeech*, pages 2743–2747, 2013.

[61] G. Zhao, S. Ding, and R. Gutierrez-Osuna. Foreign accent conversion by synthesizing speech from phonetic posteriorgrams. In *INTERSPEECH*, pages 2843–2847, 2019.

[62] Y. Zhou, Z. Xu, C. Landreth, E. Kalogerakis, S. Maji, and K. Singh. Visemenet: Audio-driven animator-centric speech animation. *ACM Transactions on Graphics (TOG)*, 37(4):1–10, 2018.



Welcome to the facial animation perception study!

Thank you for taking the time to take this perception study survey on facial animations. During the study, you will be shown **12 pairs of 3D facial animation videos** (each 3-6 seconds in duration) and you will be asked:

1. which one is more in sync with the audio
2. which one is more realistic based on the accompanying audio
3. which one is more appropriate for the given audio

You will choose either "**Left**" or "**Right**" for each question based on your preference for the two videos.
You may choose different answers for the 3 questions.

Please ensure that your **audio is turned on during the study** and if possible, **watch the videos in full-screen mode for better clarity (watch the videos multiple times if needed)**. The survey will take around **5 minutes** to complete.

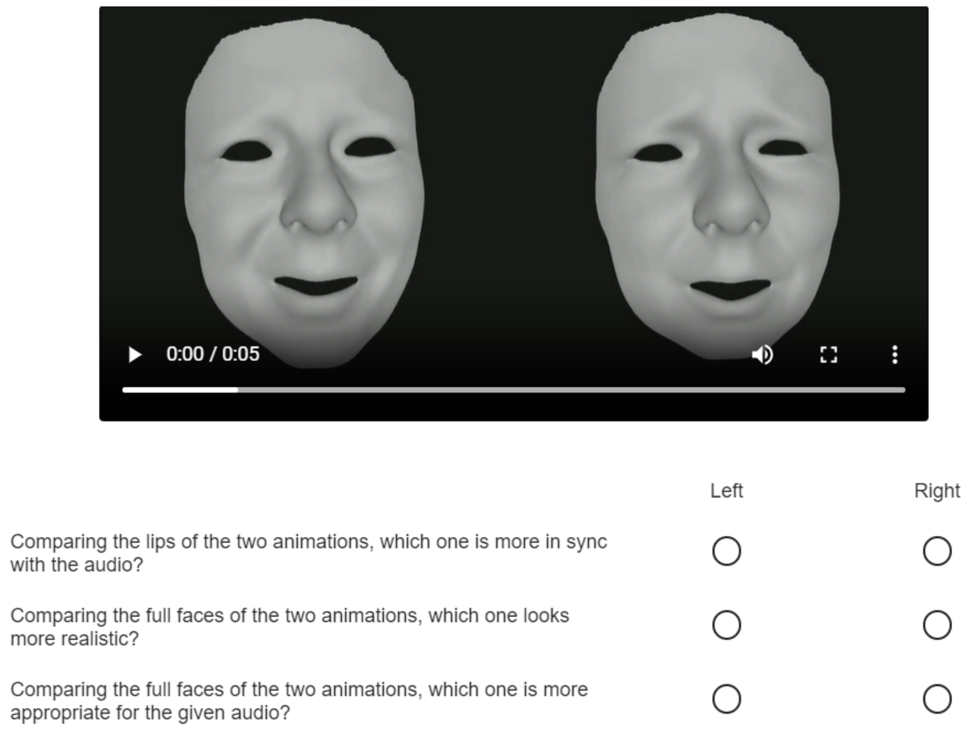
Figure 27: User study welcome screen where the instructions are presented to the user

A User Studies

To evaluate our models subjectively we conduct an analysis in the form of user studies to compare our results with other state-of-the-art approaches, namely VOCA [12], FaceFormer [19], and CodeTalker [57]. To avoid user fatigue caused by asking too many questions, we split our comparisons into 3 different tests as follows:

1. Ours vs. SOTA trained on BIWI
2. Ours vs. SOTA trained on VOCASET
3. Diffusion ablation study for our model trained on the UUDaMM dataset

The first 2 surveys will present users with 12 pair-wise comparisons. To avoid the impact of random selection by users, we randomly switch the side on which we show our model with our competitors. For each comparison, the user is asked 3 questions. For the first 2 questions we follow previous works [25, 19, 57] and ask the users about lip-synch quality and perceived realism of the animations. Additionally, we add an extra question asking about the animation appropriateness. An example of the interface can be seen in Figure 28. In an initial pilot study, we allowed the users to pick between 3 possible answers: **"Left"**, **"Right"**, and **"They are similar"**, however, we found that this lead to a majority of the users relying on the third option, making it hard to draw meaningful conclusions from



	Left	Right
Comparing the lips of the two animations, which one is more in sync with the audio?	○	○
Comparing the full faces of the two animations, which one looks more realistic?	○	○
Comparing the full faces of the two animations, which one is more appropriate for the given audio?	○	○

Figure 28: User interface of the first user study. The users will be shown 12 such side-by-side comparisons and asked to answer the questions underneath

the results. In the final study, we kept only the first 2 options, therefore forcing the users to make a more clear choice.

The questions the users were presented with are as follows:

1. Comparing the lips of the two animations, which one is more in sync with the audio?
2. Comparing the full faces of the two animations, which one looks more realistic?
3. Comparing the full faces of the two animations, which one is more appropriate for the given audio?

B 2-stage training approach

In the past, we have seen the benefits of a 2 stage approach in producing more expressive animations in papers such as Meshtalk, Codetalker or Learning2Listen. All of these approaches first train their model to learn the latent expression space. In this stage audio is not used, the first stage network only uses the visual frames. We experiment with a similar approach, in the first stage training a diffusion autoencoder. The facial decoder of this autoencoder is then kept frozen in the second stage when the model is trained to predict visual frames from the audio input. A visual representation of this model can be seen in Figure 31.

By employing this approach we have found that the model learns to produce more motion in the face, but worse results overall. This however happens at the detriment of the objective metrics, in all our experiments, the results being worse than our proposed model, as well as some of the other state-of-the-art techniques. Our intuition tells us that in this approach the model does not make such strong connections between style and motions, being able to generate more general animations. However, further work would have to be made to draw some more concrete conclusions.

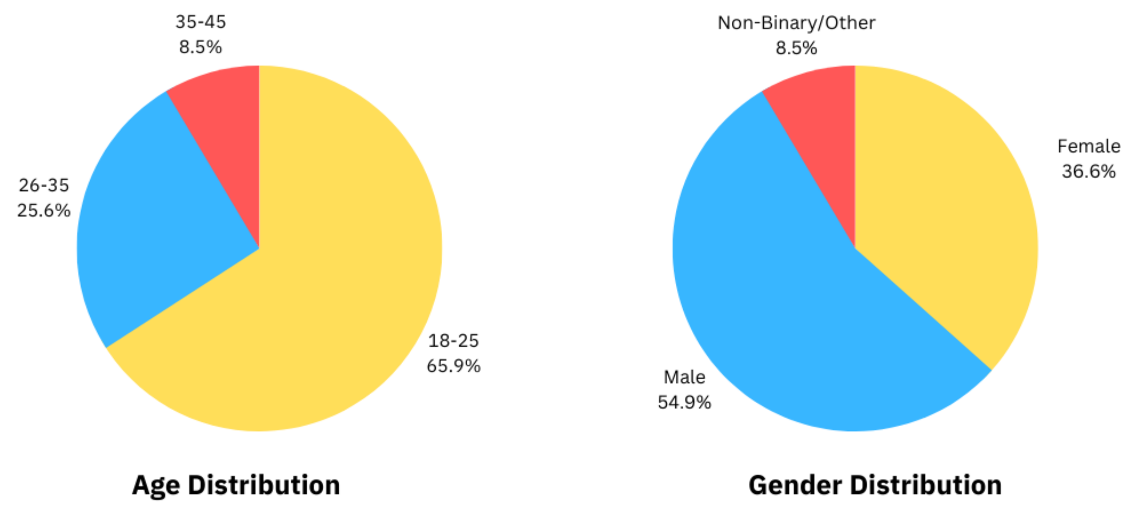


Figure 29: Demographic distribution of the user studies participants

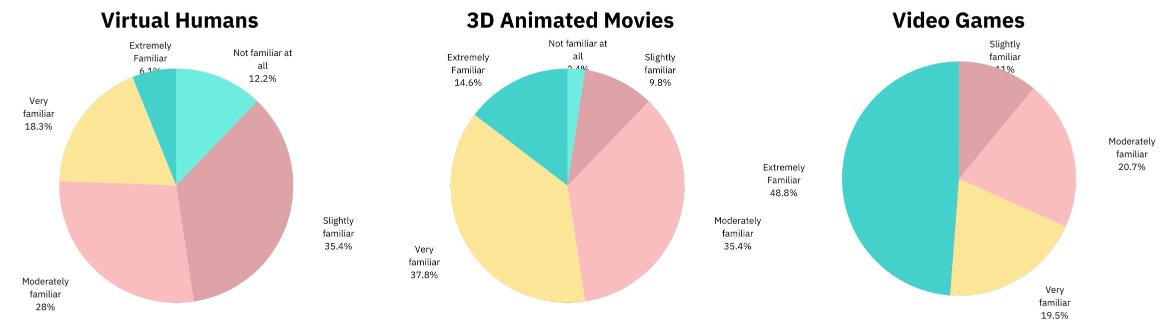


Figure 30: Particpants reported familiarity with different computer animation subjects

C Extra Figures

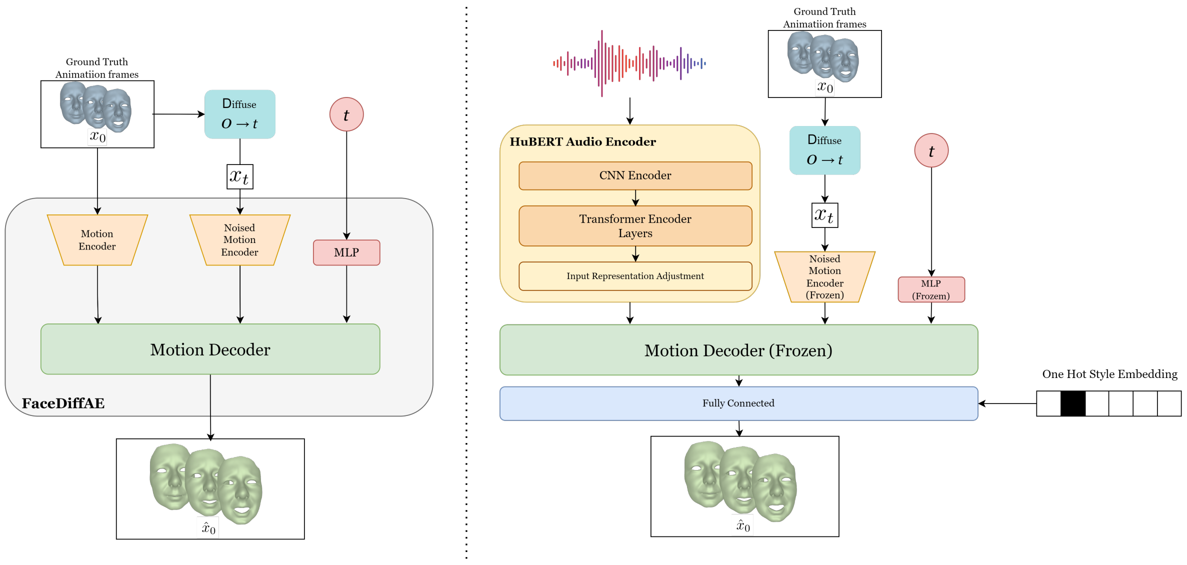


Figure 31: High-level overview of the 2 stage FaceDiffuser architecture

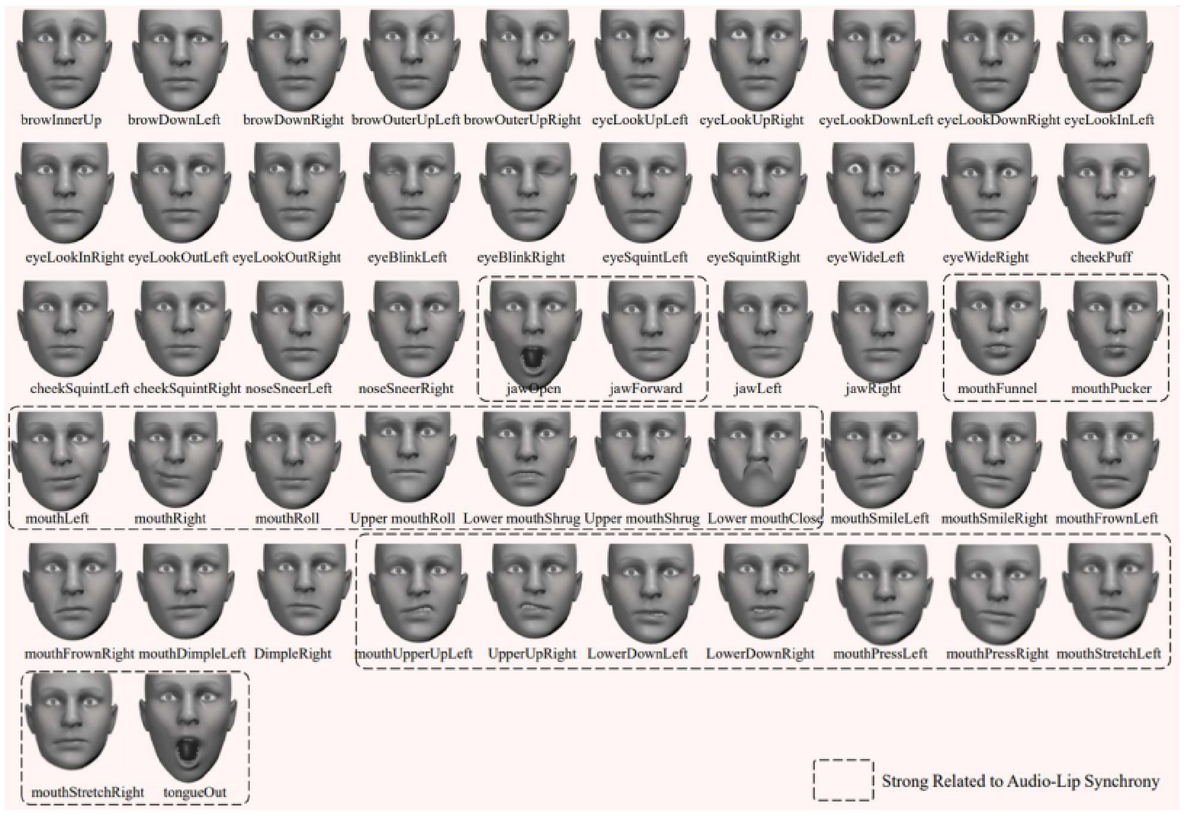


Figure 32: The blendshapes used for creating the BEAT dataset. Image taken from *Liu et al.* [34].

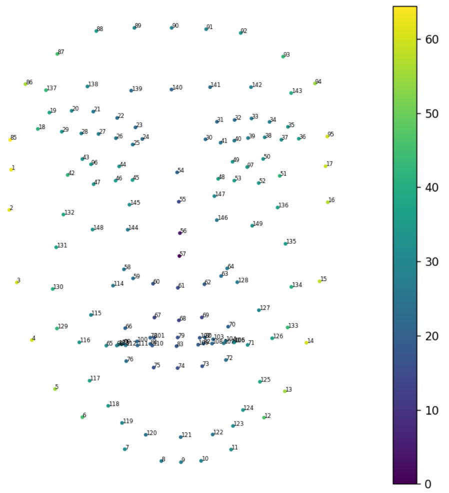


Figure 33: 3D landmark positions in the UUDaMM dataset. The colour of the landmarks represents the third dimension.