Learned Real-time Facial Animation from Audiovisual Inputs for Low-end Devices

Iñaki Navarro
Dario Kneubuehler
Tijmen Verhulsdonck
inavarro@roblox.com
dkneubuhler@roblox.com
tverhulsdonck@roblox.com
Roblox Corporation
United States of America

Eloi du Bois
William Welch
Charles Shang
Ian Sachs
ebois@roblox.com
wwelch@roblox.com
cshang@roblox.com
isachs@roblox.com
Roblox Corporation
United States of America

Victor Zordan*
Morgan McGuire†
Kiran Bhat
vbzordan@roblox.com
morgan@roblox.com
kbhat@roblox.com
Roblox Corporation
United States of America

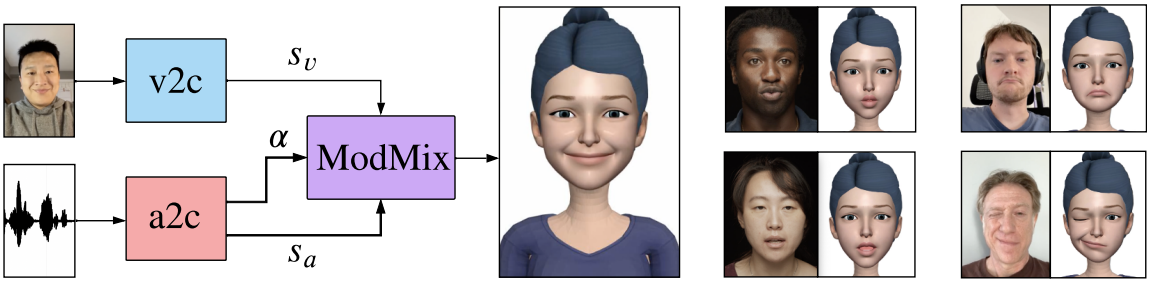

Figure 1: We synthesize facial animation in real time via two convolutional neural nets. On the left, the v2c network maps video input (s_v) while the a2c network maps audio input (s_a) and voice activation (α). These signals are fused and applied to a FACS encoded facial rig. Audiovisual inputs yield quality animation when both are present as well as robustness in the absence of either. We optimized the system for production on our target of operating on low-end mobile devices.

ABSTRACT

We present an approach for generating facial animation that combines video and audio input data in real time for low-end devices through deep learning. Our method produces control signals from audiovisual inputs separately, and mixes them to animate a character rig. The architecture relies on two specialized networks that are trained on a combination of synthetic and real world data and are highly engineered to be efficient in order to support quality avatar faces even on low-end devices. In addition, the system supports several levels of detail that degrade gracefully for additional scaling and efficiency. We showcase how user testing has been employed to improve performance and a comparison with state of the art.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → Robotics.

KEYWORDS

Facial tracking, neural networks, real-time character animation

ACM Reference Format:
Iñaki Navarro, Dario Kneubuehler, Tijmen Verhulsdonck, Eloi du Bois, William Welch, Charles Shang, Ian Sachs, Victor Zordan, Morgan McGuire, and Kiran Bhat. 2023. Learned Real-time Facial Animation from Audiovisual Inputs for Low-end Devices. In *ACM SIGGRAPH Conference on Motion, Interaction and Games (MIG '23), November 15–17, 2023, Rennes, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3623264.3624451

1 INTRODUCTION

Real-time face animation for virtual 3D characters has important applications such as AR/VR, interactive 3D entertainment, previsualization and video conferencing. Yet despite important research breakthroughs in facial tracking and performance capture, there are few commercial examples of real-time facial animation applications in the consumer market. Further, mass adoption requires real-time performance on commodity hardware and visually pleasing animation that is robust to real-world conditions without requiring manual calibration. We present a deep learning based framework for synthesizing facial animation from combined video and audio

*Also with Clemson University.
†Also with University of Waterloo.

MIG '23, November 15–17, 2023, Rennes, France

ACM ISBN 979-8-4007-0393-5/23/11.
https://doi.org/10.1145/3623264.3624451

that addresses most of these challenges. Our formulation is fast and produces visually pleasing animations even on low-end devices.

Our approach inputs audio and video for direct animation control for rigged faces in real time. To this end, we employ the Facial Action Coding System (FACS) [Ekman and Friesen 1978] to define a set of controls to deform a 3D face mesh rig. Our adoption of FACS is twofold. First, FACS controls are intuitive and therefore easy to use. And second, they are rather ubiquitous [Apple 2021] and transferable between rigs. The aim for our deep learning-based method is to take real-time video as input and output a set of FACS with minimal delay. To achieve this, we opt for a per-frame solution and rely on the network architecture to impose temporal continuity through causal convolution in order to regress FACS values that are smooth over time. Further, we propose an audio solution that specifically targets the mouth and jaw FACS controls. We combine the two in our multimodal approach using per-control blending values and the voice activity probability (see Figure 1).

The strength of using multimodal input includes not only improved "lip sync" through a specialized audio component, but also redundancy that aids in continuity during occlusion, poor lighting, and single viewpoint limitations. Furthermore, our formulation allows us to disregard either of the video or voice inputs and create FACS animations from the available signal, making it robust in real-world communication use cases. The system is designed to be immediately usable by a wide spectrum of users including those on low-end devices. Our contributions also include a method for extensive use of synthetically rendered frames in training. We harness the strength of these frames by linking the computer generated and real world images through facial landmarks. With this, our video network is trained on millions of carefully crafted and well-distributed image sequences. Notably, these contain consistent and controllable samplings of facial pose sequences which are hard to find in real-world data sets. In conjunction, the audio network is carefully trained on hybrid audio data from audio books and real-world speech in a manner that makes it robust to environment and voice variation, while maintaining sharp FACS production.

We posit that fused audiovisual signals are more robust and have higher visual quality than either used separately. We perform various ablation studies that reveal the contributions and weaknesses of the components of each network. We also showcase our meticulous study of the engineering highlighting optimizations aimed at real-time performance on low-end devices, and a method for user testing to improve overall visual fidelity of avatar animation and address known issues.

2 RELATED WORK

Recently, deep learning has been applied to the problem of face tracking and animation with great success. Several works use convolutional models to predict facial landmark locations for images and videos [Bulat et al. 2021; Bulat and Tzimiropoulos 2017; Chen et al. 2019; Kumar et al. 2020]. The results are precise, but these approaches are unsuitable for real time use on mobile devices given their proposed architectures and compute budgets.

In PFLD [Guo et al. 2019], the authors propose a lightweight parameterizable architecture based on MobileNetV2 [Sandler et al. 2018] for 2D landmark prediction. MobileNetV2 is a convolutional architecture optimized to run on mobile CPUs. It reduces the total compute and the size of its basic blocks by using separable convolutions: depth-wise convolutions followed by point-wise ($1x1$) convolutions, instead of regular 2D-convolutions.

Feng et al.[2018] use a single convolutional model to simultaneously generate 3D face alignment and 3D face reconstruction from a single image. Grishchenko et al.[2020] directly estimate a 3D face mesh from images using an attention mechanism to improve eye and mouth accuracy. Their architecture runs in real time on mobile GPUs. Laine et al.[2017] propose a real time deep learning method to predict vertex positions of the input face using a convolutional network. A drawback of that system is that the model must be be retrained for each user, unlike ours which generalizes. Wood et al.[2022] regress dense 2D landmark positions along with their certainty using a convolutional model trained on 100K realistic synthetic images, and use the outputs to fit a 3D model.

Vision transformers (ViT) have outperformed convolutional neural networks on several computer vision tasks. Recently, Xu et al. [2022] propose a set of ViT based networks to regress 2D body keypoints. The compute and memory requirements make these architectures less suitable for low-end mobile devices. Feng et al. [2021] train a model which regresses a parameterized face model given in-the-wild images, disentangling personalization and expression parameters by means of self-supervised training and differential rendering. The created avatars are animatable via these parameters. According to the authors, their system can fail due to extreme head pose and lighting, and is not robust to extreme occlusion. They also list as future work imposing temporal continuity on their tracking.

Several previous works have looked at producing facial animations from speech. Karras et al. [2017] propose a convolutional network to regress the per-vertex difference vectors from a neutral face. They input also an emotional state during training, which allows them to modify the emotional expression at inference time. Cudeiro et al. [2019] propose a mechanism to condition the convolutional model on the speaker identity during training, allowing the model to produce different realistic speaking styles. The model takes as inputs DeepSpeech [Hannun et al. 2014] features and the speaker identity, and predicts FLAME [Li et al. 2017] parameters allowing for generalization of both the input speaker and the animated target. MeshTalk [Richard et al. 2021] produces not only mouth region animation but also plausible animations in the rest of the face such us blinks or brow movements. The approach is based on a categorical latent space that disentangles audio-correlated and audio-uncorrelated information based on cross-modality loss.

Faceformer [Fan et al. 2022] produces very high quality 3D face mesh predictions taking raw audio as input by using a transformer-based autoregressive model with an audio encoder based on a self-supervised pretrained Wav2vec 2.0 [Baevski et al. 2020] model. Their solution is various order of magnitude larger in size, memory footprint and compute than ours, and not suitable for low end-devices. Codetalker [Xing et al. 2023] uses a very similar architecture as Faceformer, but their autoregressive encoder-decoder is pretrained to learn a discrete motion prior using a VQ-VAE codebook which enforces plausible facial expressions and motions. FaceXHuBERT [Haque and Yumak 2023] also leverages the power of self-supervised training this time using a pretrained HuBERT [Hsu et al.

2021] encoder. Their decoder, based on GRU layers, predicts vertex displacements. Although lighter than Faceformer and Codetalker is still computationally too expensive for low-end devices. These recent works predict mesh vertices which cannot directly used on any rig without a retargeting step.

The Snap [Chen et al. 2018] method generates facial animation from audio and video like ours, however that requires an explicit neutral expression calibration. That limits robustness and convenience for real-world use where a single user's camera angle, lighting, makeup, and facial hair may frequently change between sessions. Another method combining audio and video inputs is Abdelaziz et al. [2020] which, as with our method, outputs FACS to drive animation, and also runs on a mobile device. They train a network with independent trunks for audio and video which are concatenated at the very end to regress mouth related FACS, while the rest are regressed solely from the video input. A modality dropout during training forces the network to use information from both modalities.

3 OVERVIEW

Following the abstract schematic in Figure 1, multimodal input streams are taken as input and separately converted to two unique control inputs, s_v and s_a, for FACS from video and audio as well as α for mixing the two signals based on the predicted presence of speaking. Our system computes these values directly as audio and video regressed animation parameters, as opposed to geometric values such as 2D landmarks or the 3D face mesh. Each of the multimodal channels is processed in trained deep learning based systems which are both uniquely designed to be efficient on low-end hardware and robust to environment. The core elements of each is described along with our proposed network architecture and our method for training each, for video (Section 4) and for audio (Section 5). Our approach and considerations for combining these signals is next (Section 6). Special data preparation and use is key to our approach, we describe this in detail (Section 7). Further, we showcase that we are able to develop the system to correct issues through addition of new animation training sequences and targeted user testing (Section 8). We present our LOD approach, levels and control next (Section 9). The highlights of our implementation and engineering optimizations follow (Section 10). Finally we describe our results (Section 11), discussion and conclude (Section 12).

4 VIDEO TO ANIMATION

For video to animation (v2c), our aim is a real-time tracking system that can convert a video sequence taken as (frame-by-frame) input and output FACS controls and head pose for each frame. Further, our goal is to make this functionality available to a broad audience, implying a range of compute capabilities. To this end the proposed architecture shown in Figure 2 supports levels of detail (LODs) which enables it to trade off animation quality for lower computation cost. The resulting system supports smooth operation and makes real-time facial animation accessible on a wide range of devices.

In its lowest compute mode we rely on a lightweight multi-task model that jointly predicts face probability, landmarks, head pose and FACS weights. By lightweight model we mean that the model can produce outputs at least at 30fps on mid-range mobile devices. We call this model BaseNet.

With sufficient compute budget the system switches into running extra compute for more fidelity. In this operating mode the facial landmarks predicted by BaseNet are used to get a tight crop alignment of the face using procrustes which is fed to a larger network called HiFiNet that predicts higher quality FACS weights. BaseNet and HiFiNet share the same type of architecture based on MobileNetV2 [Sandler et al. 2018] but use different configurations (changing input resolution, number of residual blocks and layers) which are optimized for efficiency in the case of BaseNet and for quality and accuracy in the case of HiFiNet. Both consist of a multitask feature encoder that feeds several task specific sub models and are trained using real and synthetic data.

The input to BaseNet is a frame with an aligned face. In order to find a face within the full input frame coming from the camera we use the first two stages of MTCNN [Zhang et al. 2016]. The first MTCNN stage, called Proposal Net (P-Net), is in charge of finding face proposals, while the second, Refinement Net (R-Net), refines and filters the initial proposals providing faces in the form of bounding boxes. BaseNet is then run on these proposals to provide a face probability, landmarks, FACS and head pose. We are not limited to using MTCNN for the face proposal phase, other face detectors could be used.

To achieve real-time performance, and assuming we only track a face at a time, we take advantage of the correlation between consecutive frames. Namely, once an initial face is detected, we skip the MTCNN proposal network and directly run BaseNet by using the landmarks of the previous frame to align the input face. The face proposal network is only executed if the face probability predicted drops below a specific threshold. This results in an average speed up of 10x compared to running the face proposal network on every frame.

4.1 Network Architecture

Each of BaseNet and HiFiNet employ a feature encoder that feeds multiple task specific decoder models (Figure 2). We co-train BaseNet feature encoders (E_{vl}, E_{vh}) and landmarks (D_l) and face probability (D_p) decoders on a mix of synthetic and real images. We train all headpose (D_z) and FACS (D_{vl}, D_{vh}) solely on synthetic video.

Our task specific decoders for FACS (D_{vl}, D_{vh}) and head pose (D_z) are small, temporally aware neural networks that use high level features of the encoder as their input. In contrast to the encoder they are trained on continuous sequences of video data rather than single images and are therefore able to learn temporal relationships between frames. The architecture of the FACS prediction decoder (Figure 2) consists of 3 layers of causal convolutions with a kernel size of 2x1, followed by an LSTM layer and a fully connected layer. Our causal convolution layers are applied over the time dimension. This allows the model to implicitly learn filtering functions that are able to reduce jitter while maintaining responsiveness. Furthermore, using small temporal sub models allows us to reduce the size of the feature encoder even further without compromising quality because the availability of temporal information allows the decoders to compensate for the features coming from the smaller encoder.

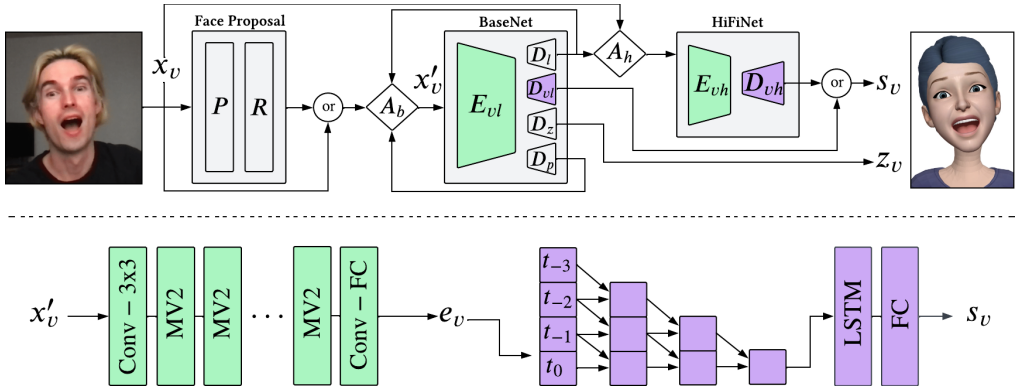

Figure 2: Overview of the video to animations system (v2c). The encoder models E_{vl} and E_{vh} provide high level information on a per frame basis which is then accumulated over time and processed by the FACS decoder models D_{vl} and D_{vh}. The BaseNet contains three additional decoder models for regressing landmarks (D_l), face probability (D_p) and head pose (D_z).

BaseNet and HiFiNet encoders differ in the input resolution, number and size of MobilenetV2 blocks. The decoders or temporal FACS models have identical architectures but different trained weights. In Table 3, we present some of the architecture and performance details for each of the architectures. The final architectures were obtained by doing systematic ablation studies looking at jitter and positional error in FACS as well as model size, while keeping the compute as low as possible for a desired given quality.

4.2 Training

Training a FACS model in a supervised way directly from synthetic data is not straightforward, because the domain gap between our synthetic data and actual real faces is large. This means that the feature representations learned by a network trained on a synthetic dataset are different from ones required for real data and the model is not able to generalize.

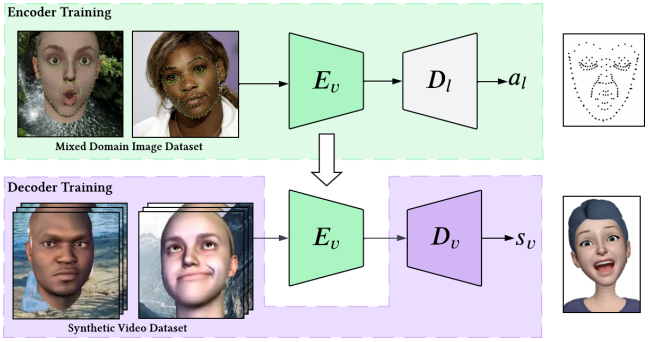

Figure 3: We train our video to animation models (BaseNet and HiFiNet) in two phases. In the first phase we train a feature encoder by regressing 175 facial landmarks. In the second phase, we freeze the weights of the encoder and train a FACS regression temporal model.

To solve this problem we split the training up into two phases. Firstly, we train our feature encoder network on regressing landmarks from both synthetic and real data. Although the landmarks are not used to regress FACS this helps the model to learn a representation that is valid for both real and synthetic data. The result of this training phase is a feature encoder that is able to encode high level features of faces, which can be used as inputs to subsequent processing stages.

To jointly train the encoder E_v and the landmarks sub model D_l, we linearly combine two different loss terms:

- L_{LMK}, Positional Loss on landmarks We use the RMSE of the regressed positions.
- L_{CON}, Consistency Loss on landmarks. This encourages landmark predictions to be equivariant under different transformations [Honari et al. 2018]. It allows to use pairs of real images without landmark annotations.

In the second training phase we train the actual FACS decoder. We use the previously trained feature encoder, weights frozen. By doing so we can make sure that the encoder retains its capability to work on real data, while the FACS regression model is trained on synthetic data using the high level features of the encoder as its input. To train the FACS decoder, we linearly combine several different loss terms:

- L_{POS}, Positional Loss on FACS. We use the MSE of the regressed FACS.
- L_{VEL}, Velocity Loss. We reduce jitter using temporal losses over synthetic animation sequences. A velocity loss inspired by [Cudeiro et al. 2019] is the MSE between the target and predicted velocities. It encourages overall smoothness of dynamic expressions.
- L_{ACC}, Acceleration Regularization Loss. In addition, a regularization term on the acceleration is added in order to reduce FACS weights jitter (its weight kept low to preserve responsiveness).

5 AUDIO TO ANIMATION

Our deep learning based audio to animation (a2c) component takes a raw audio sequence as input and outputs a subset of speech-related FACS controls (s_a, e.g., jawOpen, mouthPucker, mouthStretchRight) and also voice-activity likelihoods (α) for use in the Audio/Video

mixing of Section 6. The audio to animation system needs to generalize well across different types of voice, be robust to noisy environments, and be computationally efficient in order to deploy on low-end mobile devices. This is accomplished with separately trained encoder and decoder stages. As shown in Figure 5, the encoder is trained using audio from many speakers, with the goal of finding a speaker-independent latent representation of the speech. The output animation decoder is then fine-tuned using a smaller set of FACS labeled videos.

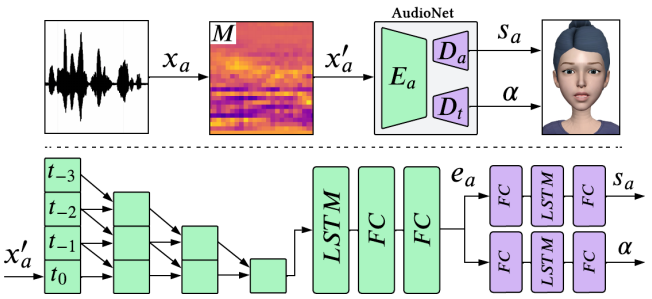

Figure 4: Overview of the audio to animations system (a2c). The audio-to-animations system consists of an encoder, composed of causal convolutional layers, LSTM layers, and a fully connected layer. This encoder (E_a) generates a semantic embedding of the input audio, which is then fed into FACS and VAD decoders (D_α, D_t) to generate the final animation outputs.

5.1 Network Architecture

Our audio to animation network (AudioNet) includes two stages: an encoder (E_a) which takes the audio feature (x'_a) as input and encodes it into a shared phoneme-aware embedding space, and decoders (D_α, D_t) which are appended after the embedding to estimate task specific outputs, shown in Figure 4.

For audio feature extraction, given a raw input audio clip, we extract 26 MFCC features at a hop length of 16ms and a window length of 16ms. Five consecutive frames of MFCC features are packed together to form the final input to the encoder, which are reshaped flat before the input. Note that packing consecutive features is the only systematic latency (40ms, which is calculated as the middle point of 5 windows of 16ms) introduced into the a2c sub-system.

As a trade-off between representation and computational cost, we use a simple network architecture to encode the input to a latent embedding. This network includes: (1) a stack of causal convolutional layers followed by batch norm [Ioffe and Szegedy 2015] and activated by leaky-ReLU [Maas et al. 2013]. The causal convolutional layer expands the receptive field without introducing any additional latency since it only looks at series values from the past; (2) an LSTM layer that works along the time dimension to retain a temporal consistent embedding. This choice of encoder provides a simple yet effective embedding for downstream tasks and enables the audio to animation system to run in real-time on low-end mobile devices. This causal convolution followed by an LSTM pattern used in AudioNet follows the same structure as the v2c FACS decoder described in Section 4.

Our audio to animation system outputs are structured as multiple tasks off the encoder. We use one decoder for each task to estimate the corresponding output from the embedding. These decoders share a same architecture except for dimension of the output layers, including a fully connected layer to reduce the embedding dimension, followed by an LSTM layer and a fully connected output layer. Since the first layer reduces the feature dimension (e.g., 64-d), these decoders run very fast.

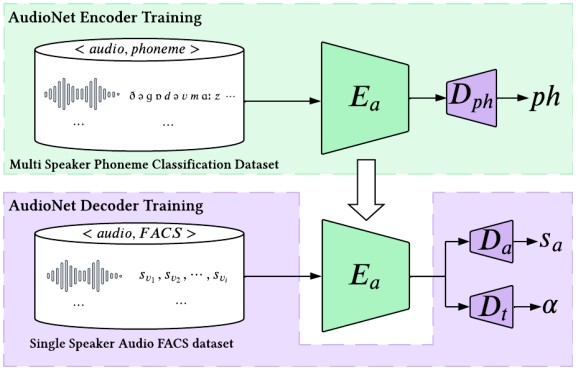

Figure 5: We also follow the two-phase training process in the audio to animation. In the first phase we train a audio encoder to output phoneme labels (ph). In the second phase, we add decoders on top the encoder to regress the FACS weights (s_a) and VAD signals (α).

5.2 Training

In the training of encoder and decoder portions of the network, audio samples are presented along with various time-aligned labels. The audio samples are augmented with a variety of randomly selected noise and pitch transformations. In order to improve the robustness of the model to different types of noise, we augment the input audio samples by adding randomly generated white noise or selecting pre-recorded ambient noise, such as street, restaurant, wind, or rain sounds. To simulate people speaking in different virtual rooms, we also use impulse response convolution to add reverberation to the input audio. Additionally, we have found pitch shifting, gain shifting, and speed changing to be useful techniques for further augmenting the audio samples.

Early end-to-end training experiments on the encoder+decoder network revealed that speech-related animation labels varied from speaker to speaker in ways that led to over-smoothed and muted FACS predictions. Much sharper results were obtained when training on a single speaker's motions. This led us to pre-train the encoder using only audio, from many different speakers, with the goal of learning a speaker-independent latent speech representation. The best results were achieved by pre-training the encoder on a phoneme prediction task. The intuition is that phonetic speech has a strong correlation to visual speech units (visemes[Zhou et al. 2018]) arising from the mechanics of producing the phonetic sounds. Training a phoneme recognition task thus favors an internal representation that is transformable to continuous analogues of viseme sequences, represented as FACS curves.

Specifically, for this phoneme training task, we append a simple decoder (D_{ph}), which is implemented as a fully connected layer activated by Softmax, after the embedding to estimate the phoneme labels. And the encoder along with the phoneme output layer are trained on phoneme transcripts of the LibriSpeech dataset [Panayotov et al. 2015]. To keep the pre-training simple, a sole CTC loss [Graves et al. 2006] is used without providing any time related information in the training. The encoder is trained for 100 epochs with an initial learning rate of 1e-3 and scheduled to reduce to 10 every 30 epochs. The model reaches to 23.6% word error rate on the validation set after pre-training.

After the encoder training, a phoneme-aware representation is fed into subsequent decoders, including the FACS decoder and a VAD (Voice Activity Detector) decoder as a binary (speech vs no-speech) classification. Once the encoder has been trained, its weights are frozen before the decoder training may proceed. This is done so as not to bias the encoder towards the single speaker's voice presented during the FACS decoder training. The FACS output task is trained using FACS-labeled audios from a single speaker [url 2017]. The VAD output task is trained concurrently using labeled audio examples from the VAD collection. We co-train the voice activity (D_t) and FACS (D_a) decoders by linearly combining three loss terms:

- L_{POS}, Positional Loss on FACS for audio, similar to that used in v2c training but using smoothed L1 norm [Girshick 2015].
- L_{VEL}, Velocity Loss, similar v2c and to [Cudeiro et al. 2019], but implemented as a smooth L1 loss, to stabilize the predictions over time.
- L_{VAD}, a cross entropy loss for the voice activity detection task.

6 MULTIMODAL MIXING (MODMIX)

Animation curves from audio and video sources are linearly blended. Mixed results s are derived from video to animation (s_v), and audio to animation results (s_a) by:

$$s_i = s_{v,i}(1-\alpha) + \alpha(w_{a,i}s_{a,i} + w_{v,i}s_{v,i}) \quad (1)$$

where α is a voice activity signal, $w_{v,i}$ and $w_{a,i}$ are mixing weights for video and audio results respectively, and i is the FACS index. If there is no speaking in the audio, α is 0 and only predictions from the video to animation pipeline are contributing to the final result. If voice activity is detected the α signal transitions from 0 to 1 and the output becomes a weighted combination of results from audio to animation and video to animation pipelines.

Note, with this rather simple approach, we did consider alternatives such as an additional network for mixing. However, the overhead of adding a third network to do this alone was deemed to be undesirable because the simple mixing technique described was far less costly (in network and computation) and did not degrade the quality output of the two input streams. Thus, we concluded that there is very little to gain from adding a more sophisticated mixing approach. The core role it plays is to provide a means to combine and switch the signals, which the simple solution we propose does well – with practically no cost in download or computation.

Figure 6: Synthetic data is generated for training. The value of synthetic data is key to our approach - to add precise FACS data pairing, control (especially extreme) lighting, reduce jitter, and increase individual subject variation, as well as offer a much larger dataset than otherwise possible.

7 DATA PREPARATION

The success of our approach relies in part on careful data preparation for both the v2c and a2c training. In both cases, we leverage an internal motion capture system for our ground truth data, though in substantially different ways.

7.1 Real and Synthetic Images

Our deep learning models for video need to generalize to a wide range of users and be invariant to face shape, age, ethnicity and gender. Furthermore, the model must be robust to challenging environmental conditions such as extreme lighting and occlusions. Training such a model requires a large amount of data. Typically, supervised learning methods rely on human labeling to generate training data. However, obtaining ground truth FACS labels for real data is non trivial because the mapping from facial expressions to exact FACS weights is ambiguous. Hence generating labels in such a way would result in a noisy dataset. For this reason we make use of synthetic data as this allows us to get an exact mapping between FACS labels and renders of a set of facial rigs. Furthermore, this also allows us to control the distribution of the dataset in terms of facial expressions, head poses, gender, age and ethnicity and avoid undesired biases.

We generate the training data (see Figure 6) in two phases. First, we generate FACS curves by an offline motion capture system on a video expression dataset and in a purely procedural fashion, and then augment both type of animations. Second, we use the generated animation curves to drive our face model and render animation sequences using various identities and lighting setups. This yields continuous sequences, which allows us to take the temporal domain into account when defining the model architecture. The FACS and head pose regression decoders are trained purely on synthetic data. We use a mix of real and synthetic data to train the encoder parts of both BaseNet and HiFiNet.

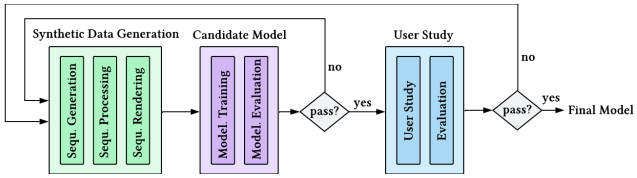

Figure 7: Visualization of our two phase iterative model selection process. We first manually evaluate candidate models by visually inspecting the performance on an internal dataset. If it passes, we conduct a user study. If any of the two verification steps fail we resort back to generating suitable training data for the relevant expressions.

7.2 Audio Data

For audio training, the a2c encoder is trained using examples from audio books in the Librispeech dataset [Panayotov et al. 2015]. The Montreal forced aligner [McAuliffe et al. 2017] was used to produce phoneme transcripts from the original text and audio clips. We retrained examples from over 2400 speakers for a total of 1000 hours of training data. Sequences are segmented into short (5-10 sec) clips along silence boundaries to create training examples from longer clips.

The a2c (FACS) decoder is trained using the audio from single speaker videos (Barack Obama's weekly addresses [url 2017]. FACS labels for these videos are derived using an in-house offline system. For the results in this paper, we assembled 13 hours of training data in this manner after filtering out non-frontal face clips. In addition to the FACS decoder, a voice activity detector is needed for use in audio/video mixing as described in Section 6. We used the AVA Spoken Activity Dataset [Roth et al. 2020] , which provided 45 hours of mixed speech/noise audio with speech and background labels for use in training. Finally, we augment audio examples with background noise during a2c network module training. There are two sources of noise: (1) convolution with impulse response functions, used to simulate audio environments. (2) background noise recordings from a subset of Audioset [Gemmeke et al. 2017].

8 USER TESTING AND MODEL SELECTION

We performed extensive testing with subjects through the user study process outlined in Figure 7. As our models train within a few hours (see Table 1), we were able to run many experiments on our network architectures, loss weights, and input data. We started off with quantitative tools for selecting the preferred system. Namely, our initial model selection for v2c was done based on ground truth error per frame and a measure of jitter over the validation sequences. As well, for a2c, selection was based the word error rate on phoneme classification for the encoder while the decoder's selection was based on jitter and responsiveness measurements. But as we refined the system, the selection of new models was done by a combination of internal subjective testing of models, followed by significant user-in-the-loop testing.

We explain this process with an extended description of a specific example. Namely, during development, we found out that our models had learned a strong correlation between the mouth and the eyes: if the user performed a wink and opened their mouth the wink would not be predicted by our models. The issue was identified as a data distribution problem – we lacked training sequences in which winks were performed with the mouth open. We used our data generation system to create a better distribution of sequences which covered the failure cases, rendered them and trained several new models. We selected the best among these models looking at jitter and positional error metrics, followed by subjective visual inspection of predicted animations on a set of 100 videos which reflect different expressions, identities, poses, and lighting conditions as well as additional 'live' tests with the system running. Figure 8 provides a visualization of model performance before and after the tuning process described above.

Table 1: Training times in hour for the different networks on a Nvidia Titan Xp.

Network	Training time (h)
BaseNet encoder + landmark & face prob. decoders	5
HiFiNet encoder	8.5
BaseNet/HiFiNet FACS decoder	5
AudioNet encoder (phoneme pretraining)	12
AudioNet FACS + VAD decoders	3

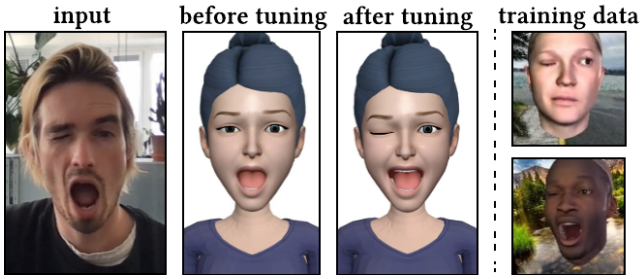

Figure 8: Model performance before and after the tuning process and exemplary training data used for tuning.

We then performed subject testing with about forty users in which we would do A/B testing between the previous system and the selected candidate. In the test, users ran two different versions with two devices (phones) placed side by side and performed a battery of facial expressions (neutral expression, winks, wink while opening mouth, large open mouth, frowning, etc.) while looking at the resulting animations. They were asked to choose the best for each expression. We gave the users detailed instructions with examples on how to perform the different expressions. Each user study replied with between 500 and 800 answers depending on the number of subjects and expressions evaluated. The supplementary video showcases examplary outputs from this methods.

9 LEVEL OF DETAIL (LOD)

We introduce a level of detail (LOD) strategy for the execution of our audio-video pipeline to selectively improve performance and ensure a smooth experience for the end user based on their

hardware and use conditions. We leverage the two-stage nature of our video pipeline to predict FACS from the BaseNet decoder (D_{vl}) or from the HiFiNet decoder (D_{vh}) when possible, to produce better quality outputs.

We vary the encoder *input* video rate (30 vs 15 fps), but always produce 30 fps *output*. For skipped input frames, we take as input the encoder's feature vector from the previous frame, which allows us to extrapolate the FACS outputs and generate results that are almost identical to that of running the encoders at 30fps (see Subsection 11.2). Our system supports this through its architecture, particularly the causal convolution blocks which provide a form of time filtering. The FACS video decoders are much smaller than the video encoders (see Tables 3), making our encoder subsampling strategy very efficient.

We support four LODs as defined in Table 2, which vary the encoder (BaseNet or HiFiNet) and subsampling. When FACS are predicted by HiFiNet, BaseNet is still run to predict head pose, face probability, and landmarks for alignment. For higher accuracy, we always run the very efficient a2c if audio input is available.

10 IMPLEMENTATION AND OPTIMIZATION

The implementation of our proposed algorithms included many careful and evidence based engineering choices and optimization techniques. Additionally our aim was to optimize our models for the largest range of possible devices so we explicitly optimized our models for deployment on devices without any kind of dedicated neural network inference hardware. We highlight some key choices that led to the real-time performance of our implementation.

10.1 Pre-training architecture optimization

In the implementation of the BaseNet/HiFiNet subsystem, we draw inspiration from MobileNetV2 [Sandler et al. 2018] but introduce critical optimizations for our application's demands. Namely, we changed the input resolution, number of layers, depth per layer, and included strided convolutions in the first layers. We also selectively use unpadded convolutions to decrease the feature map size. Using unpadded convolutions to reduce the featuremap size gives more control compared to strided convolutions, meanwhile we can maintain the residual by simply slicing the residual feature map to match the size of the unpadded convolution's output feature map.

Table 2: LOD scheme as name, encoder, and subsampling.

LOD	Network predicting FACS	Subsampling
LOD4	HiFiNet	None
LOD3	HiFiNet	HiFiNet only
LOD2	HiFiNet	All
LOD1	BaseNet	BaseNet only

Table 3: Architecture and performance information for the video and audio to animation models. The second column refers to the model size when stored in float16 precision, third is the input resolution of each model, and last is the Multiply-Accumulates (MAccs).

Architecture	Model size float 16	Input resolution	MAccs
BaseNet encoder	1.01MB	65x65	5.23M
HiFiNet encoder	1.85MB	129x109	25.22M
Base/HiFiNet FACS decoder	540KB	n/a	202K
AudioNet encoder	290KB	130x1	147K
AudioNet FACS decoder	78KB	n/a	39K
AudioNet VAD decoder	76KB	n/a	38K

10.2 Post-training architecture optimization

The next optimization step we incorporated was a post training architecture optimization step, here we take a trained model and improve the execution efficiency. This is primarily done by fusing layers, for example a convolution layer followed by a batch normalization layer can be fused into a single convolution layer by folding the batch normalization layer frozen variables into the preceding convolution. To achieve this we used the tensorflow Transform Graph tool [url 2022] and the NCNN Optimize tool [Tencent 2023]. Additionally we reduced the size of the frozen model 50% by reducing weights and biases from float32 to float16 precision at negligible accuracy loss.

10.3 Runtime execution optimization

We designed our algorithms to be executed on a large range of devices we focused on optimizing our models for inference on a CPU. Usually inference on a GPU is preferred for neural networks but we didn't choose this options for two reasons; The first being the fact that supporting inference on the GPU for a large range of devices is much harder than supporting inference on the CPU. The second reason was that due to our convolutional architecture being an order of magnitude smaller than most convolutional architectures such as MobilenetV2 [Sandler et al. 2018], we found that execution on the GPU often took more time compared to execution on the CPU. Our assumption is that this is due to our architecture being mostly memory bound as opposed to compute bound due to size and the abundant use of depth-wise convolutions.

Our most significant optimization adjusting featuremaps to blocks of 8 scalars; desktop AVX SIMD uses 8-element blocks and mobile Neon SIMD uses 4 at float32 and 8 for float16. This yields a net 50% increase in performance (SIMD execution is not perfectly efficient) [Zhang et al. 2018].

11 RESULTS AND EVALUATION

With our goal of real-time on low-end machines, we have done extensive performance evaluation of our architectures on different devices, Table 4 reports inference performance of the different architectures in our system using the NCNN framework [Tencent 2023]. Figure 9 shows that our system can map a wide range of input expressions to different characters via FACS.

11.1 Single vs. multimodal frameworks

The supplementary video shows that when using audio input, we capture more inner-mouth details. For example the mouthPucker and mouthFunnel shapes have audio-based activations that exceed video by 30-40%. When only video is used, our method is able to capture expressions (e.g., furrowing the brow), but does not accurately reproduce detailed lip movements. By fusing the results

Table 4: Inference times in ms (mean and standard deviation) on several Android, macOS and Windows Intel based devices for our network architectures.

	BaseNet encoder		HiFi encoder		Base/HiFiNet FACS decoder		AudioNet	
Device	mean (ms)	std dev (ms)	mean (ms)	std dev (ms)	mean (ms)	std dev (ms)	mean (ms)	std dev (ms)
Samsung S6 (Exynos 7420 Octa)	3.8883	0.1647	8.2793	0.2933	0.4268	0.0160	0.6104	0.1081
Samsung Tab Lite S6 (Exynos 9611)	3.0019	0.0691	7.8778	0.1096	0.2491	0.0224	0.5063	0.0316
Samsung S8 (Qualcomm MSM8998)	2.6133	0.1543	7.0503	0.3659	0.2089	0.0261	0.4140	0.0848
Samsung Note9 (Qualcomm SDM845)	1.3564	0.1547	2.8330	0.3784	0.1547	0.0240	0.2726	0.0285
Samsung S9+ (Qualcomm SDM845)	1.3351	0.0504	2.6476	0.0641	0.1462	0.0178	0.2433	0.0149
iPhone 6S (A9)	1.0576	0.0648	2.4224	0.0876	0.1288	0.0134	0.2795	0.0368
Samsung S10e Europe (Exynos 9820)	0.8842	0.0947	1.5246	0.2062	0.0890	0.0087	0.1596	0.0152
Huawei Mate 40 Pro (Kirin 9000)	0.7945	0.0361	1.5726	0.0501	0.0850	0.0104	0.1814	0.0270
iPhone Xr (A12)	0.2836	0.0054	0.8075	0.0134	0.0513	0.0036	0.1245	0.0263
iPhone 12 (A14)	0.2105	0.0522	0.6160	0.0969	0.0371	0.0086	0.0849	0.0124
MBP 2019 (i9-9980HK CPU @ 2.40GHz)	0.4972	0.0531	1.4780	0.0765	0.0797	0.0083	0.1130	0.0304
Windows (i7-8850H @ 2.6GHz)	0.5859	0.0954	1.7332	0.1597	0.0879	0.0222	0.1478	0.0359
Windows (i9-10885H @ 2.40GHz)	0.4955	0.0313	1.5155	0.0692	0.0690	0.0114	0.1427	0.0150

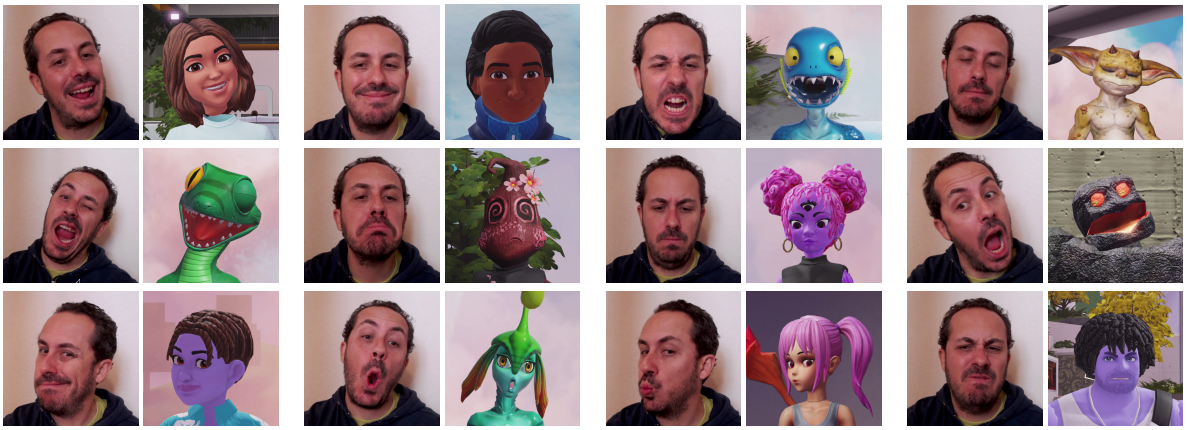

Figure 9: Our system performs well under a large range of input expressions. Because we use the standard FACS abstraction, these can be mapped to diverse avatar facial geometry.

of both modes we are able to track facial expressions while keeping good lip sync.

We investigated the impact of temporal information in our architecture by an ablation study on the v2c decoder. In this experiment, a baseline architecture replaced the temporal block consisting of causal convolutions and a LSTM with a fully connected layer. This resulted in low animation quality due to the high jitter, absent temporal information. We also experimented with adding more fully connected layers to give the model more capacity. This resulted in a slightly improved error metric but even higher jitter values. The causal temporal architecture refers to the architecture proposed in this paper. While keeping the FACS error constant, it greatly reduces the jitter. Finally, we experimented with a non-causal version of our architecture which predicts animations with a delay of one frame. By doing so, jitter was reduced even further without compromising the accuracy, but at the cost of introducing a lag. Since our system needs to be real time, we did not pursue this architecture despite it's favorable properties. Table 5 shows results for mean absolute FACS error and mean jitter over 10 training experiments, as well as corresponding standard deviations. FACS error is computed on an internal synthetic test dataset. Jitter is evaluated on a internal video expression dataset. We compute jitter by aggregating the FACS first derivatives when they differ in sign from the previous frame (change of direction in the movement) and dividing them by the number of frames and FACS.

11.2 LOD accuracy and performance

Our system supports four LOD levels which differ on whether HiFiNet is run or not, and whether BaseNet and HiFiNet encoders are run on every frame or on alternate frames (see Section 9). A lower LOD means less compute per frame is used, however a lower LOD also means the predictions are less accurate. To compare, we

compute the mean absolute error when running our video subsystem on a set of synthetic videos which exercise different facial expressions and head poses. The error on LOD1 is larger than in the rest of modes, which is expected since its FACS predictions come from BaseNet which has much lower capacity than HiFiNet used in LODs 2,3 and 4. The error difference between the last three modes is minimal with LOD2 having the largest error and LOD4 the lowest.

Figure 10 shows predicted animation curves (jawOpen, mouthPucker and eyeBlinkLeft) by the four LODs on a short input video. The differences of LOD2 and LOD3 with respect to LOD4 are minimal, almost not visible, indicating that the subsampling strategies of LOD2 and LOD3 do not degrade the quality of the predictions. We quantify the subsampling effect by calculating the prediction error of LOD2 and LOD3 when taking LOD4 as ground truth. We used an internal set of 50 videos and a total ~33k frames (representing different expressions, identities and lighting conditions) and calculated the mean absolute error across all the frames and predicted FACS, resulting in 0.0123 for LOD2 and 0.0059 for LOD3.

As seen in the figure, the main differences in the animation curves occur for LOD1 with respect to the other modes. This is expected since the predictions come from two different models that were trained separately, have different input resolutions and encoder capacities. Still the trend of each FACS control when comparing LOD1 outputs vs the rest is very similar.

In Figure 11, we show the performance of the LOD levels when run on different devices. We can see how LOD2 requires approximately half of the inference time on average per frame when compared to LOD4 for almost the same accuracy.

11.3 Comparison to deployed alternatives

We compare our video to animation sub-system against ARKit [Apple 2021], which generates equivalent FACS outputs taking as input only video. ARKit is only supported on iPhone X or newer devices, while our proposed solution can run on older devices such us iPhone 6s and non Apple devices.

We performed a quantitative comparison between the two systems by measuring jitter and expression asymmetry. Facial symmetry has been identified as a major contributing factor to perceived

Table 5: Ablation study on the v2c FACS decoder architecture. Error measures are averaged over 10 experiments.

	FACS Error		FACS Jitter	
Architecture	mean	std dev	mean	std dev
Baseline	9.2e-2	0.63e-3	46.0e-4	1.9e-5
Causal Temp.	9.2e-2	1.1e-3	7.2e-4	1.4e-5
Non Causal Temp.	9.2e-2	1.5e-3	4.7e-4	0.78e-5

Table 6: Comparison of our system (LOD4) against ARKit in terms of jitter and asymmetry for our internal general and symmetric datasets respectively. Lower is better.

	FACS Asymmetry		FACS Jitter	
System	mean	std dev	mean	std dev
ARKit	13.0e-3	8.2e-3	8.3e-4	7.7e-4
Ours	3.6e-3	2.3e-3	3.1e-4	1.6e-4

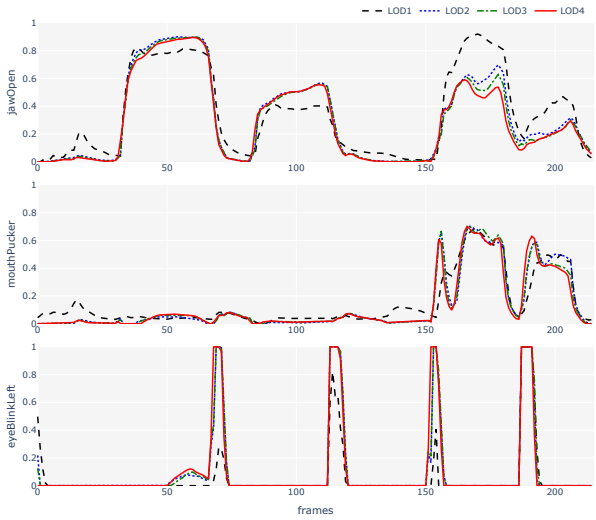

Figure 10: FACS Animation outputs (jawOpen, mouthPucker and eyeBlinkLeft) on the same input video sequence for the different LODs using only the v2c. LODs 2-4 use HiFiNet with different subsampling modes and yield similar results. LOD1 predictions are dissimilar due to its reliance on BaseNet.

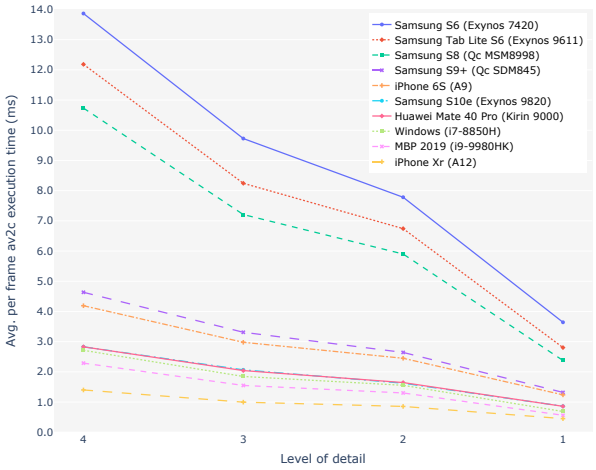

Figure 11: Mean inference times in ms on Arm and Intel devices for different LODs.

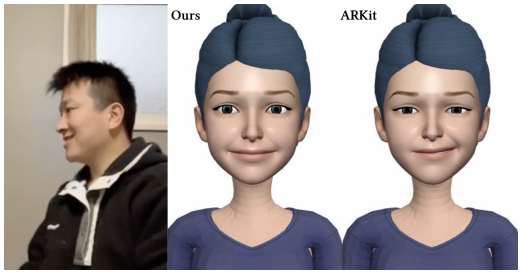

Figure 12: Input frame, our LOD4 result, and ARKit for self occlusion. We predict symmetry for the occluded region.

attractiveness [Perrett et al. 1999], while jitter on facial animations is very distracting and unpleasant to see. To measure jitter we use the metric described in Subsection 11.1. We measure asymmetry taking the mean absolute value of the difference between symmetric FACS control pairs, for instance between mouthStretchLeft and mouthStretchRight.

We recorded animation outputs and input videos from ARKit to then run under our system. We collected a first dataset to evaluate jitter. It contains multiple identities performing various expressions under different lighting conditions and head poses. We collected a second dataset to evaluate asymmetry, which contains the same variations but is restricted to symmetrical expressions such as smiles, puckers, mouth opens frowns and blinks. If the input expression is symmetric the asymmetry predicted by the system should be as low as possible. In Table 6, we show the mean and standard deviation for asymmetry metric on the symmetry dataset, as well as the jitter metric for the general dataset, comparing our video subsystem in LOD4 and ARKit. We show improvements in lower mean jitter and asymmetry than ARKit.

Fig. 12 compares our LOD4 predictions to ARKit for self-occlusion. Our system produces a more symmetric smile than ARKit. See the video supplement for more extensive comparison.

12 DISCUSSION AND CONCLUSIONS

In this work, we present a combined audiovisual approach for facial animation. We are able to achieve robust, real-time results tracking facial expressions from video, and complement this with a separate audio to animation model dedicated to estimating lip sync shapes from user's speech. Our ability to leverage speech audio also allows us to infer reasonable mouth movements under occlusion, and opens up more interesting productization possibilities.

We also propose a multi-tiered LOD system for facial animation. We can target maximum quality at the expense of higher compute by running both v2c and a2c together. We extend this LOD idea further by dividing the v2c model into BaseNet and HiFiNet, and by introducing a data-driven subsampling approach that allows us to avoid running our v2c model on every input frame. As demonstrated in our experiments, this data-driven subsampling deviates minimally from original, but requires only half of the compute time.

We describe a scalable system for directly regressing FACS weights from video by training on synthetic data. Using synthetic data gives us perfect FACS ground truth and allows us to train on unlimited identities in unlimited lighting conditions. Furthermore, it becomes trivial to sample the full space of facial expressions that we need for training in an iterative fashion. The key insight here, however, is that these benefits are only achievable by first bridging the domain gap by pretraining on a landmark detection task with a combination of natural and synthetic images. By directly regressing FACS from video, we avoid the pitfalls of landmark based approaches which struggle with the identity vs expression ambiguity. And because we regress FACS parameters rather than final geometry, our tracker can be used with any FACS supported rig.

There are several known limitations of our approach and directions for future work. Starting with audio lip sync, due to our training on clear speaking with limited noise augmentations, singing and jittery long vowels are not interpreted correctly. Further, in noisy environment, the current system may confuse background noises as speaking, especially sharp sounds, like knocking. Limits with the FACS rigs make it difficult to distinguish some vowels (i, e, a) and these rigs are also limited in their ability to control teeth and the tongue. As for video to animation, we implemented single face tracking, although support for multiple faces may be straightforward. Apart from running BaseNet (and HiFiNet on certain LODs) per found face, we would need to run a our Face Proposal module with a certain periodicity to incorporate new faces to the scene. Due to limitations in the input resolution, BaseNet produces a reduced set of FACS, including omitting gaze, compared to HiFiNet. Another limitation of the BaseNet, in contrast to the HiFiNet, is the possible loss of fast movements, such as blinks, under low lighting conditions due to the reduced temporal resolution. In addition, as we targeted symmetry, we in turn experience some loss of asymmetry.

As future work, we would like to extend the causal convolution/LSTM/FC decoder architecture for other real-time tracking problems such as body mocap. We also think that the LOD ideas described in the paper, such as temporal subsampling, are useful for improving performance of other real-time computer vision problems. The idea of using synthetic data to incorporate user preference followed by retraining is also applicable for other visual applications that use reinforcement learning with human feedback.

ACKNOWLEDGMENTS

We would like to acknowledge the help and support of additional fellow Robloxians that made this work possible. Vivek Verma, Michael Palleschi, Jihyun Yoon, Mahesh Ramasubramanian, Ron Griswold, and John Dodelson, thank you for your work on behalf of this effort.

REFERENCES

2009-2017. "Your Weekly Address". https://obamawhitehouse.archives.gov/briefing-room/weekly-address.

2022. Tensorflow Graph Transform. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/tools/graph_transforms.

Apple. 2021. ARKit Developer Documentation. https://developer.apple.com/documentation/arkit/arfaceanchor

Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems* 33 (2020), 12449–12460.

Adrian Bulat, Enrique Sanchez, and Georgios Tzimiropoulos. 2021. Subpixel heatmap regression for facial landmark localization. *arXiv preprint arXiv:2111.02360* (2021).

A. Bulat and G. Tzimiropoulos. 2017. How Far are We from Solving the 2D and 3D Face Alignment Problem? (and a Dataset of 230,000 3D FacialLandmarks). In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 1021–1030. https://doi.org/10.1109/ICCV.2017.116

Lisha Chen, Hui Su, and Qiang Ji. 2019. Face Alignment With Kernel Density Deep Neural Network. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 6991–7001. https://doi.org/10.1109/ICCV.2019.00709

Xin Chen, Chen Cao, Zehao Xue, and Wei Chu. 2018. Joint Audio-Video Driven Facial Animation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3046–3050. https://doi.org/10.1109/ICASSP.2018.8461502

Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael Black. 2019. Capture, Learning, and Synthesis of 3D Speaking Styles. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 10101–10111. http://voca.is.tue.mpg.de/

Paul Ekman and Wallace V Friesen. 1978. Facial action coding system. *Environmental Psychology & Nonverbal Behavior* (1978).

Yingruo Fan, Zhaojiang Lin, Jun Saito, Wenping Wang, and Taku Komura. 2022. Faceformer: Speech-driven 3d facial animation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18770–18780.

Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. 2021. Learning an Animatable Detailed 3D Face Model from In-the-Wild Images. *ACM Trans. Graph.* 40, 4, Article 88 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459936

Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. 2018. Joint 3d face reconstruction and dense alignment with position map regression network. In

Proceedings of the European conference on computer vision (ECCV). 534–551.

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. 2017. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 776–780.

Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. 369–376.

Ivan Grishchenko, Artsiom Ablavatski, Yury Kartynnik, Karthik Raveendran, and Matthias Grundmann. 2020. Attention Mesh: High-fidelity Face Mesh Prediction in Real-time. arXiv:2006.10962 [cs.CV]

Xiaojie Guo, Siyuan Li, Jiawan Zhang, Jiayi Ma, Lin Ma, Wei Liu, and Haibin Ling. 2019. PFLD: A Practical Facial Landmark Detector. *CoRR* abs/1902.10859 (2019). arXiv:1902.10859 http://arxiv.org/abs/1902.10859

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567* (2014).

Kazi Injamamul Haque and Zerrin Yumak. 2023. FaceXHuBERT: Text-less Speech-driven E (X) pressive 3D Facial Animation Synthesis Using Self-Supervised Speech Representation Learning. *arXiv preprint arXiv:2303.05416* (2023).

Sina Honari, Pavlo Molchanov, Stephen Tyree, Pascal Vincent, Christopher Pal, and Jan Kautz. 2018. Improving Landmark Localization with Semi-Supervised Learning. arXiv:1709.01591 [cs.CV]

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), 3451–3460.

Ahmed Hussen Abdelaziz, Barry-John Theobald, Paul Dixon, Reinhard Knothe, Nicholas Apostoloff, and Sachin Kajareker. 2020. Modality Dropout for Improved Performance-Driven Talking Faces. In *Proceedings of the 2020 International Conference on Multimodal Interaction* (Virtual Event, Netherlands) *(ICMI '20)*. Association for Computing Machinery, New York, NY, USA, 378–386. https://doi.org/10.1145/3382507.3418840

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

Tero Karras, Timo Aila, Samuli Laine, Antti Herva, and Jaakko Lehtinen. 2017. Audio-Driven Facial Animation by Joint End-to-End Learning of Pose and Emotion. *ACM Trans. Graph.* 36, 4, Article 94 (jul 2017), 12 pages. https://doi.org/10.1145/3072959.3073658

Abhinav Kumar, Tim K Marks, Wenxuan Mou, Ye Wang, Michael Jones, Anoop Cherian, Toshiaki Koike-Akino, Xiaoming Liu, and Chen Feng. 2020. Luvli face alignment: Estimating landmarks' location, uncertainty, and visibility likelihood. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8236–8246.

Samuli Laine, Tero Karras, Timo Aila, Antti Herva, Shunsuke Saito, Ronald Yu, Hao Li, and Jaakko Lehtinen. 2017. Production-Level Facial Performance Capture Using Deep Convolutional Neural Networks. In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '17)*. Association for Computing Machinery, New York, NY, USA, Article 10, 10 pages. https://doi.org/10.1145/3099564.3099581

Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. 2017. Learning a Model of Facial Shape and Expression from 4D Scans. *ACM Trans. Graph.* 36, 6, Article 194 (nov 2017), 17 pages. https://doi.org/10.1145/3130800.3130813

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. Atlanta, Georgia, USA, 3.

Michael McAuliffe, Michaela Socolof, Sarah Mihuc, Michael Wagner, and Morgan Sonderegger. 2017. Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi.. In *Interspeech*, Vol. 2017. 498–502.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5206–5210.

David I Perrett, D.Michael Burt, Ian S Penton-Voak, Kieran J Lee, Duncan A Rowland, and Rachel Edwards. 1999. Symmetry and Human Facial Attractiveness. *Evolution and Human Behavior* 20, 5 (1999), 295–307. https://doi.org/10.1016/S1090-5138(99)00014-8

Alexander Richard, Michael Zollhöfer, Yandong Wen, Fernando de la Torre, and Yaser Sheikh. 2021. MeshTalk: 3D Face Animation from Speech using Cross-Modality Disentanglement. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 1153–1162. https://doi.org/10.1109/ICCV48922.2021.00121

Joseph Roth, Sourish Chaudhuri, Ondrej Klejch, Radhika Marvin, Andrew Gallagher, Liat Kaver, Sharadh Ramaswamy, Arkadiusz Stopczynski, Cordelia Schmid, Zhonghua Xi, et al. 2020. Ava active speaker: An audio-visual dataset for active speaker detection. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 4492–4496.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CoRR* abs/1801.04381 (2018). arXiv:1801.04381 http://arxiv.org/abs/1801.04381

Tencent. 2023. NCNN. https://github.com/Tencent/ncnn.

Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Matthew Johnson, Jingjing Shen, Nikola Milosavljević, Daniel Wilde, Stephan Garbin, Toby Sharp, Ivan Stojiljković, Tom Cashman, and Julien Valentin. 2022. 3D Face Reconstruction with Dense Landmarks. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 160–177.

Jinbo Xing, Menghan Xia, Yuechen Zhang, Xiaodong Cun, Jue Wang, and Tien-Tsin Wong. 2023. Codetalker: Speech-driven 3d facial animation with discrete motion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12780–12790.

Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. 2022. ViTPose: Simple Vision Transformer Baselines for Human Pose Estimation. In *Advances in Neural Information Processing Systems*.

Jiyuan Zhang, Franz Franchetti, and Tze Meng Low. 2018. High Performance Zero-Memory Overhead Direct Convolutions. *CoRR* abs/1809.10170 (2018). arXiv:1809.10170 http://arxiv.org/abs/1809.10170

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR* abs/1604.02878 (2016). arXiv:1604.02878 http://arxiv.org/abs/1604.02878

Yang Zhou, Zhan Xu, Chris Landreth, Evangelos Kalogerakis, Subhransu Maji, and Karan Singh. 2018. Visemenet: Audio-driven animator-centric speech animation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.