Supplemental Materials
Probabilistic Speech-Driven 3D Facial Motion Synthesis: New Benchmarks, Methods, and Applications

Karren D. Yang, Anurag Ranjan, Jen-Hao Rick Chang, Raviteja Vemulapalli, Oncel Tuzel
Apple
{karren_yang, anuragr, jenhao_chang, r_vemulapalli, ctuzel}@apple.com

A. Supplemental Video

Please see the **accompanying video** for an overview of our work and qualitative examples from our model.

B. Overview

The following sections provide additional methodology and/or results that were not included in the main paper.

Section C provides more details on the metrics, including (i) limitations of maximal lip vertex error for evaluating probabilistic models and (ii) details of the pretrained models used for evaluation.

Section D provides the complete table of benchmark results corresponding to Figure 2 of the main paper, a discussion of CodeTalker [11], and additional ablation results for our model.

Section E discusses *efficiency* of our method, including (i) a knowledge distillation strategy for amortizing the sampling strategies, and (ii) diversity *vs*. efficiency trade-off that can be achieved by sampling fewer codes from our autoregressive model at inference time.

Section F provides implementation and training details that were deferred from the main text.

Section G discusses the limitations and ethical considerations of this work.

C. Metrics

C.1. Limitation of Maximal Lip Vertex Error

Maximal lip vertex error (ℓ_{vertex}) is a metric that measures the maximum difference in lip vertices between the ground truth mesh and a mesh generated from the model. This metric is used as a proxy for lip articulation quality in existing works, but as a standalone metric, it has limitations for evaluating probabilistic models. As shown in Supplemental Figure 1, a probabilistic model (row 2) can generate lip articulation that is more similar to the ground truth (row 1), but due to variations between samples, have larger ℓ_{vertex} compared to a deterministic model (row 3) that generates an over-averaged result. Our proposed lip vertex metrics, ℓ_{cover} and ℓ_{mean}, address this limitation and provide a more complete picture of performance. Overall, there is a need to look across multiple metrics (sync score, FD score, lip vertex error) when evaluating speech-driven 3D facial motion synthesis.

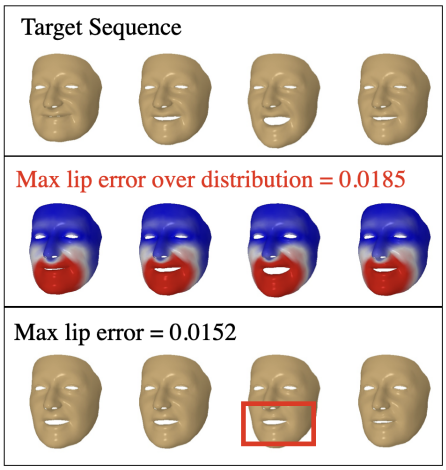

Figure 1. **Limitations of ℓ_{vertex} as a metric.** Probabilistic models (row 2) generate 3D facial motions with diversity, as shown by the color map of standard deviation. They may achieve worse ℓ_{vertex} compared to deterministic models (row 3), despite being able to generate a mesh sequence that matches the ground truth sequence better (row 1). See text for details.

C.2. Audio-Mesh Synchronization Networks

The audio-mesh synchronization networks are trained using InfoNCE contrastive loss [9] with a batch size of 64, *i.e.,* for each 3D mesh sequence, we sample 63 negative audio examples that are either semantically misaligned (taken from a different clip) or temporally misaligned (taken from a different time point of the same clip). Supplemen-

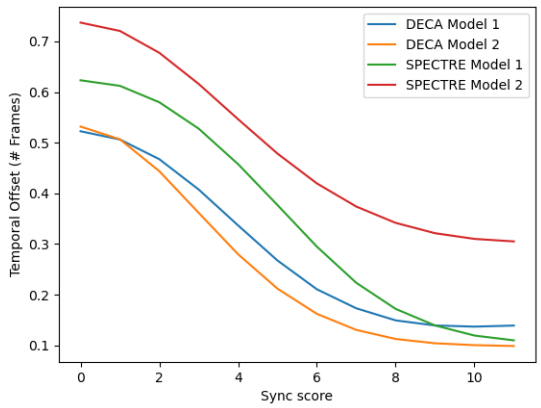

Figure 2. **SyncNet Evaluation** All of our synchronization networks can detect individual frames of temporal shift between audio and 3D facial mesh sequences.

tal Figure 2 shows plots of the models evaluated on held-out ground truth audio-mesh pairs with increasing temporal misalignment. The results indicate that all the pretrained networks are sensitive to individual frames of audio-mesh misalignment.

C.3. Speaking Style Recognition Network

The style recognizer is trained on 3D facial motion (deformation between animated and neutral face meshes). The facial motion encoder uses a similar architecture as our RVQ encoder with standard 1D convolutional blocks instead of causal 1D convolutional blocks. We use angular margin loss [4] to maximize the cosine similarity between embeddings from the same speaker while minimizing cosine similarity with other speakers. The performance of the models for DECA and SPECTRE are shown in the GT line of Table 2 in the main paper.

D. Additional Results

D.1. Complete Table - Main Figure 2

Supplemental Table 1 shows the full results corresponding to Figure 2 of the main paper. Our approach outperforms the existing methods across the board. Importantly, while deterministic methods (*i.e.,* Faceformer+Style) achieve good lip synchronization and lip vertex error, they suffer in diversity/realism as highlighted in red.

Sync score Existing deterministic methods (FaceFormer, Faceformer+Style) achieve better synchronization than existing probabilistic methods (Meshtalk, MeshTalk-ND, MeshTalk+Style, MeshTalk-ND+Style). Our probabilistic method achieves the highest sync scores out of all methods, particularly when we use our sampling strategies to trade off diversity for greater speech fidelity (Ours+Avg100).

Frechet distance Deterministic methods (VOCA, Faceformer, Faceformer+Style) suffer on this metric, as highlighted in red, suggesting that the generated facial motions are unrealistic. The probabilistic methods perform better on this metric, with our method outperforming MeshTalk on three out of four cases.

Maximal Lip Vertex Error Deterministic methods (VOCA, Faceformer, Faceformer+Style) achieve lower ℓ_{vertex}, which measures the maximum vertex error between the ground truth and one synthesized mesh sequence. However, this does not take into account the diversity of probabilistic methods. When we compute the maximal vertex error between the ground truth and the average of many synthesized sequences (ℓ_{mean}), our approach matches Faceformer+Style and outperforms the others. We also achieve the lowest coverage error (ℓ_{cover}), suggesting that the ground truth sequences are closest to our sampling distribution. Finally, when we use sampling strategies (Ours+Avg100), we are able to trade off the coverage of our model (ℓ_{cover}) for improved precision (ℓ_{vertex}).

D.2. Discussion of CodeTalker

CodeTalker [11] extends Faceformer [5] using a vector-quantized (VQ) autoencoder to learn a discrete 3D facial motion prior. While Faceformer uses an auto-regressive transformer to directly regress 3D mesh deformations, CodeTalker uses an auto-regressive transformer to regress the embeddings of the ground truth meshes in the latent space. Their training loss consists of a combination of regression errors over the embeddings and the original 3D mesh deformations after decoding, and training occurs in a teacher-forcing manner. During inference, the predicted embeddings are projected to the nearest codes in the VQ codebook before being decoded to produce 3D facial motion. The motivation is that the projection to the VQ codebook selects a mode in the distribution of 3D facial motions, whereas Faceformer regresses to the conditional mean of motion and produces over-smoothed outputs that do not correspond to any mode. Importantly, while their auto-regressive model selects codes from a pretrained codebook, it is deterministic and selects a code that is nearest to the regressed latent embedding.

We trained the original implementation of CodeTalker by the authors on our data, as well as our own re-implementation using our RVQ codebook and auto-regressive architecture. While training the VQ codebook produced good reconstructions of 3D facial motion, we found that training the auto-regressive model using the combination of regression losses failed to converge to a reasonable result on our data. This is likely due to the large scale and diversity of our dataset compared to VocaSet [3] and BIWI [6], which leads to a high-variance, multi-modal dis-

Model	**Sync score ↑**		**Frechet distance ↓**		**Maximal Lip Vertex Error ($\times 10^{-3}$) ↓**		
DECA	**Model 1**	**Model 2**	**Model 1**	**Model 2 ↓**	ℓ_{vertex}	ℓ_{cover}	ℓ_{mean}
VOCA	0.137	0.271	22.0	2.94	10.0	10.0	10.0
FaceFormer	0.348	0.361	13.9	2.44	9.9	9.9	9.9
MeshTalk	0.262	0.174	5.2	0.48	11.1	6.9	10.3
MeshTalk-ND	0.286	0.284	1.3	0.40	13.2	8.6	10.5
FaceFormer+Style	0.369	0.441	13.2	1.89	**7.9**	7.9	**7.9**
MeshTalk+Style	0.286	0.203	4.7	0.64	8.4	6.3	8.1
MeshTalk-ND+Style	0.298	0.302	1.0	0.34	11.9	7.6	9.5
Ours	0.463	0.464	**0.9**	**0.23**	10.8	**6.0**	**7.9**
Ours+Avg100	**0.684**	**0.600**	7.1	1.18	8.3	7.1	8.2
SPECTRE	**Model 1**	**Model 2**	**Model 1**	**Model 2 ↓**	ℓ_{vertex}	ℓ_{cover}	ℓ_{mean}
VOCA	0.357	0.290	524.9	66.8	15.5	15.5	15.5
FaceFormer	0.393	0.423	449.5	70.4	15.6	15.6	15.6
MeshTalk	0.309	0.302	227.9	37.3	17.7	12.4	16.1
MeshTalk-ND	0.327	0.436	49.1	7.4	20.3	13.1	16.0
FaceFormer+Style	0.438	0.576	351.0	40.5	**12.8**	12.8	**12.8**
MeshTalk+Style	0.331	0.372	178.5	12.7	13.9	10.2	13.2
MeshTalk-ND+Style	0.325	0.474	43.2	**6.9**	17.9	11.8	14.5
Ours	0.444	0.520	**40.9**	8.4	18.2	**10.0**	13.0
Ours+Avg100	**0.565**	**0.591**	199.3	30.3	13.9	11.9	13.7

Table 1. **Benchmark Results** corresponding to Figure 2 in the main paper. Best results in each column are **bolded**, while second best results are underlined. ℓ_{vertex}, ℓ_{cover}, and ℓ_{mean} denote the maximal lip vertex error, coverage error, and mean estimate error respectively and are computed with $|\mathcal{S}| = 100$. See Section 4.2 of the main text for descriptions of the metrics.

tribution in the latent space that is difficult to regress.

While we are unable to converge to a reasonable result with their original loss, we note that conceptually, taking the *expectation* of the code embeddings sampled from our model at each time point would produce an equivalent result to performing regression in the latent space. In other words, the expected output of CodeTalker can be achieved by performing code averaging as in Section 3.3 with an infinite number of codes. Therefore, we expect the performance of CodeTalker to be the limiting case of the trend of the green points in Figure 2(a-b) of the main text.

D.3. Additional Ablations

Choice of Temporal Model We show preliminary results of our model trained with different temporal models in Supplemental Table 2. We found that using a transformer as the temporal model in the absence of a reference style clip improves both the synchronization and realism/diversity of the outputs, as measured by sync score and Frechet distance respectively. However, the results were more varied when we provide additional information through a reference style clip. Use of a transformer for the temporal model may improve diversity at the cost of synchronization.

Temporal Model	Uses Ref. Style?	Sync Score ↑	Averaged FD ↓
Transformer block	no	**0.45**	**0.51**
Ours	no	0.43	0.73
Transformer block	yes	0.43	**0.36**
Ours	yes	**0.44**	0.54

Table 2. **Choice of Temporal Model** Comparison of transformer *vs*. masked convolution blocks for the temporal model on the DECA meshes. See text for details.

Audio Encoder	First Code CE ↓	Average Code CE ↓
Wav2Vec 2.0 [1]	2.09	2.54
Ours (trained from scratch)	**1.97**	**2.50**

Table 3. **Choice of Audio Encoder** Comparison of Wav2Vec 2.0 [1] and our audio encoder trained on scratch on the DECA meshes. CE: cross-entropy loss on held-out set (lower is better). See text for details.

Choice of Audio Encoder Several recent works, namely Faceformer [5] and CodeTalker [11] use a self-supervised and pretrained wav2vec 2.0 speech model [1] as the audio encoder. While this may prevent overfitting of the audio en-

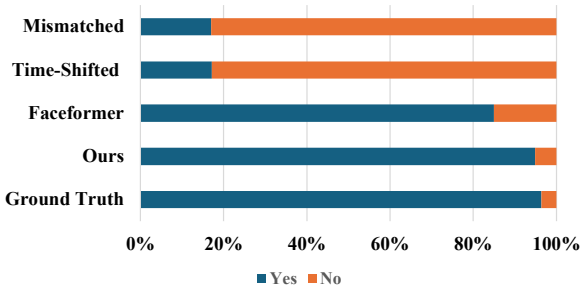

Figure 3. **Perceptual study:** Users rated the synchronization between the motion in the videos and the audio (yes/no). See text for details.

coder on small datasets as VocaSet [3] and BIWI [6], we found that using a pretrained audio encoder was not necessary for a large-scale dataset like VoxCeleb2 [2]. As shown in Supplemental Table 3, in our preliminary experiences, we found that using the pretrained speech model did not improve results.

D.4. Additional Perceptual Study

Supplemental Figure 3 shows the results of a user study that rates the synchronization of facial motion and audio. Ten participants were shown a total of ∼50 video clips. They were specifically asked whether the video was synchronized with the audio and labeled each file with Yes or No. In addition to our model's outputs, we also included negative controls (mismatched or time-shifted videos), positive controls (ground truth videos), and Faceformer as a baseline [5]. Almost all synthesized videos were rated as synchronized, comparable to the ground truth and outperforming Faceformer, while the negative controls received much lower scores.

E. Improving Efficiency

While the focus of the methodology and results in the main paper was primarily on the quality and diversity of the model, for certain applications (*e.g.,* real-time speech-driven 3D avatars), the efficiency of the method is also important. In this section, we elaborate on improving the speed/efficiency of our method.

E.1. Knowledge Distillation

In Section 3.3 of the main text, we described sampling strategies for trading off the diversity of the auto-regressive model for improved precision and fidelity. However, this strategy increases the inference speed, as multiple codes need to be sampled and aggregated. We propose a knowledge distillation strategy for amortizing this added sampling time. Recall that $\mathbf{j}$ denotes a matrix of codebook indices indexed by time t and depth d corresponding to the real facial motion $\mathbf{x}$. We obtain the new targets $\hat{\mathbf{j}}$ for the student by:

1. Computing the audio-visual context using the temporal model $\mathbf{h}_{av}[t]$ in a teacher-forcing manner, *i.e.,* inputting the ground truth $\mathbf{j}$ into Equation (2) in the main text.
2. Sampling codes from the depth model using Equation (3) without teacher-forcing. Namely, we use $\mathbf{h}_{av}[t]$ from Step 1 to compute v_{t2}, but use the sampled codes in place of the ground truth codes for computing $v_{td}, d \geq 3$.
3. Aggregating the sampled codes using strategies discussed in Section 3.3 of the main text and reprojecting them to the RVQ codebook to obtain new indices $\hat{\mathbf{j}}$.

The student model is trained in a teacher-forcing manner using both the ground truth codes $\mathbf{j}$ as well as the new targets $\hat{\mathbf{j}}$. Specifically, we use $\mathbf{j}$ as input to the temporal model, and we use $\hat{\mathbf{j}}$ as input to the depth model. We also use $\hat{\mathbf{j}}$ as the targets for optimizing the student model. As shown in Supplemental Figure 4, this enables us to distill the sampled and aggregated labels from a teacher model (blue) to a student model (yellow, '16') with improved inference time.

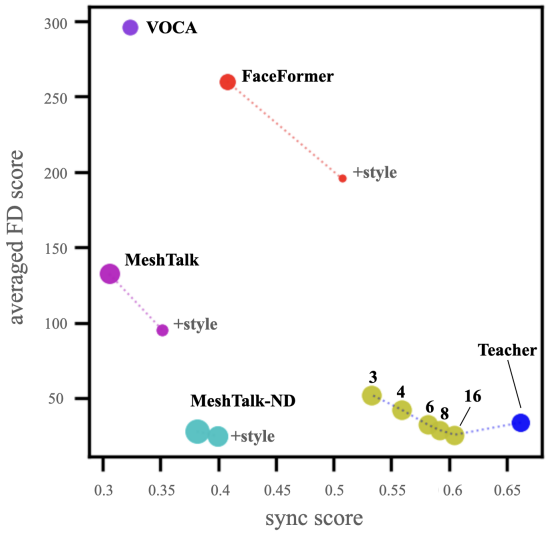

Figure 4. **Improving Model Efficiency** Results are shown for the SPECTRE meshes. See text for details.

E.2. Quality *vs*. Efficiency Trade-off

One of the advantages of the coarse-to-fine design of the RVQ codebook is the possibility of improving efficiency by predicting and decoding codes. We show that this can be done while still achieving high synchronization for as few as 3 codes (out of depth of 16). As shown by the yellow points in Supplemental Figure 4, reducing the number of codes yields a **quality *vs*. efficiency trade-off**, where we can achieve improved speed/efficiency at the cost of losing finer 3D motions. The frame rate of synthesis for a given number of codes is shown in Supplemental Figure 5.

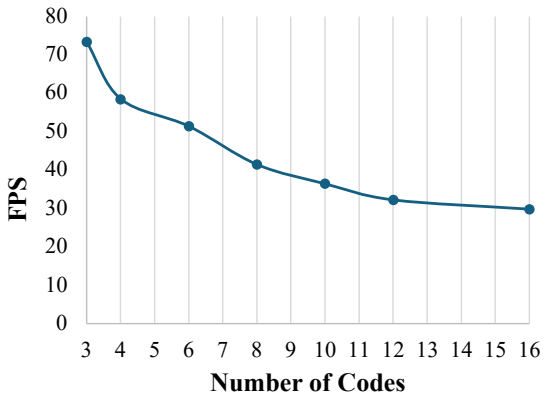

Figure 5. **Frame rate of synthesis for given number of codes.** Results are shown for the SPECTRE meshes. See text for details.

F. Implementation Details

RVQ Autoencoder The 3D facial motion encoder and decoder consist of 1D convolutional blocks. The inputs and outputs are 3D facial motion represented by mesh vertex deformations, *i.e.*, the difference between the mesh vertex positions for animated and neutral expressions. The encoder consists of a 1D convolutional layer with kernel size of 1 to aggregate information over mesh vertex deformations, then two 1D causal convolutional layers with kernel size of 3 to aggregate information over time. The decoder consists of the same blocks in the reverse order. For input size $\mathbf{x} \in \mathbb{R}^{T \times 3V}$, the size of the latent embeddings is $Z \in \mathbb{R}^{T \times N_C}$, where N_C is the dimensionality of the codes in codebook $\mathcal{C}$. In practice, we use a shared codebook [8] with $D = 16$, $|\mathcal{C}| = 256$ and $N_C = 128$.

Audio encoder Following [10], our audio encoder consists of 1D convolutional blocks operating over mel-spectrograms of 1s audio samples centered at each visual frame.

Reference Clip Encoder The reference clip is encoded using the same architecture as the RVQ encoder, except standard convolutional layers are used in place of the causal convolutions.

Two-Stage Auto-Regressive Model The temporal auto-regressive model consists of four masked causal convolutional layers [10] with kernel size of 2 and increasing dilation of $1, 2, 4, 8$ for gathering audio-visual context. The depth auto-regressive model consists of a masked transformer self-attention block with embedding size of 64.

Sampling Strategies For KNN-based sampling, we use $N = 100$ and $K = 3$. For SyncNet-based sampling, we take the top $1/2$ codes based on the synchronization score. For code averaging, we vary the number of codes averaged depending on the desired diversity vs. fidelity trade-off.

Training We train the RVQ autoencoder and two-stage auto-regressive model for approximately 150 and 200 epochs respectively with Adam optimizer [7] with learning rate of 10^{-4}. For knowledge distillation, we train the student for approximately 100 epochs. The two-stage auto-regressive model is trained in a teacher-forcing manner. We use both stochastic sampling and soft code targets [8].

G. Additional Discussion

Ethical Considerations. The datasets and models used in this work are intended for research purposes only. While meshes from this work can be used to render photo-realistic content, they should not be used to generate videos of individuals without their consent.

References

[1] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020. 3

[2] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018. 4

[3] Daniel Cudeiro, Timo Bolkart, Cassidy Laidlaw, Anurag Ranjan, and Michael 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. 2, 4

[4] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4690–4699, 2019. 2

[5] Yingruo Fan, Zhaojiang Lin, Jun Saito, Wenping Wang, and Taku 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. 2, 3, 4

[6] Gabriele Fanelli, Juergen Gall, Harald Romsdorfer, Thibaut Weise, and Luc Van Gool. A 3-d audio-visual corpus of affective communication. *IEEE Transactions on Multimedia*, 12(6):591–598, 2010. 2, 4

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[8] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022. 5

[9] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1

[10] Alexander Richard, Michael Zollhöfer, Yandong Wen, Fernando De la Torre, and Yaser 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. 5

[11] Jinbo Xing, Menghan Xia, Yuechen Zhang, Xiaodong Cun, Jue Wang, and Tien-Tsin Wong. Codetalker: Speech-driven 3d facial animation with discrete motion prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12780–12790, 2023. 1, 2, 3