Supplementary Material:
SynthVSR: Scaling Up Visual Speech Recognition With Synthetic Supervision

Xubo Liu[1,*], Egor Lakomkin[2], Konstantinos Vougioukas[2], Pingchuan Ma[2], Honglie Chen[2], Ruiming Xie[2], Morrie Doulaty[2], Niko Moritz[2], Jachym Kolar[2], Stavros Petridis[2], Maja Pantic[2], Christian Fuegen[2]

[1]University of Surrey [2]Meta AI

A. Multimedia Video

The multimedia video file `video.mp4` presents examples generated by the LAM-LRS3-AVoX-VSR model (as described in Sec. 4.3) with cropped lip images from CelebA and speech clips from Librispeech. We recommend turning on speakers to note the lip-syncing performance.

B. Architecture Details

B.1. VSR Model

Here, we describe the details of the VSR model (as referenced in Sec. 3.1 of the main paper), which is the same as that used in the previous work [1, 2]. The architecture of the VSR model is depicted in Fig. 1. The visual front-end consists of a 3D convolutional layer with a kernel size of $5 \times 7 \times 7$ followed by a ResNet-18 model. The visual features produced by the last residual block are aggregated along the spatial dimension by a global average pooling layer. Next, we use the Conformer encoder to model the visual features extracted by the front-end. Each Conformer block has a feed-forward module, a self-attention module, a convolution module, and a second feed-forward module stacked in order. We first use a linear layer to project the front-end embedding to a D-dimensional space, where D is the dimension of the Conformer encoder input embedding. The projected features are added with the relative position information and further passed through the Conformer encoder backbone. Then, we use the Transformer decoder to map text the visual representation to a distribution over word-piece tokens. The decoder is composed of an embedding layer and a stack of Transformer decoder blocks, each decoder block consists of a self-attention module, an encoder-decoder cross-attention layer, and a feed-forward layer. Layer normalization is added before each module. The prefixes from index 1 to l - 1 are projected to embedding vectors, where l is the length of target tokens. The absolute positional encoding is added to the embedding. The decoder generates the token sequence $y = (y_1, y_2, ..., y_l)$ in autoregressive manner by factorising the joint probability distribution:

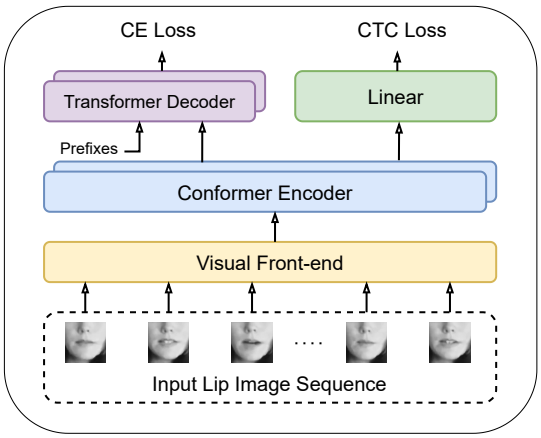

Figure 1. The VSR model used in this work is based on a Conformer encoder, a 3D ResNet visual front-end, and a combination of CTC and attention-based decoder.

$$P_{Decoder}(y|z_e) = \prod_{i=1}^{l} P(y_i|z_e, y_{1:i-1}), \quad (1)$$

where l is the length of the target token sequence and z_e is the features extracted by the Conformer encoder.

We use a combination of CTC loss and attention-based CE loss as the training objectives of the baseline VSR model. A linear projection layer is used to map the high-level visual feature sequence into the output probabilities to compute the CTC loss. The CTC criterion assumes conditional independence between the output predictions and the estimated sequence posterior has the form of $P_{CTC}(y|x) \approx \prod_{t=1}^{T} p(y_t|x)$, and the CTC loss is defined as $\mathcal{L}_{CTC} = -\log P_{CTC}(y|x)$, where x is the input video. The attention-based CE loss is calculated based on Equation (1): $\mathcal{L}_{CE} = -\log P_{Decoder}(y|z_e)$. The VSR training objective is computed as follows:

$$\mathcal{L} = \alpha \mathcal{L}_{CTC} + (1-\alpha)\mathcal{L}_{CE}, \quad (2)$$

*Work done during an internship at Meta AI.

where α controls the relative weight in CTC and CE losses. α was set to 0.1 in this work. In the evaluation, we use the model averaged over the last 10 checkpoints for decoding.

B.2. Speech-Driven Lip Animation

We describe the implementation details (as referenced in Sec. 4.3 of the main paper) of the proposed speech-driven lip animation model. Specifically, the image encoder is a 5-layer 2D CNN. Batch normalization and ReLU activation are used for the first four layers while the last layer uses tangent activation. The image encoder maps a 96×96 input image to a 512-dimensional latent representation. The speech encoder is a stack of six 1D CNN followed by a 2-layer GRU. Batch normalization and ReLU activation are used for the first five layers while the last layer uses tangent activation. After that, the encoded speech chunks are fed to the GRU layers, which produce a 256-dimensional latent feature. We use StyleGAN2 as the frame decoder. Instead of generating frames from a constant input, our StyleGAN2 decoder uses the penultimate layer of the image encoder. The frame discriminator is a 5-layer CNN that determines whether a frame is real or not conditioned on the target frame. Batch normalization and Leaky ReLU activation are used after each convolution layer except for the last layer. The sequence discriminator is a 5-layer spatial-temporal CNNs, followed by a GRU layer and a single classifier layer. Batch normalization and ReLU activation are used first four layers and the fifth layer uses tangent activation. The detailed configurations of the image encoder, the speech encoder, the frame discriminator and the sequence discriminator are listed in Tab. 1, Tab. 2, Tab. 3, Tab. 4, respectively. We refer to the configuration of a convolutional layer as *Conv[(kernel size), (stride), (padding) @ Channels]*, *BN* and *Tanh* indicates the Batch normalization and tangent activation, respectively.

Layers	Image Encoder
1	Conv2d[(4, 4) (2, 2) (1, 1) @ 64], BN, ReLU
2	Conv2d[(4, 4) (2, 2) (1, 1) @ 128], BN, ReLU
3	Conv2d[(4, 4) (2, 2) (1, 1) @ 256], BN, ReLU
4	Conv2d[(4, 4) (2, 2) (1, 1) @ 512], BN, ReLU
5	Conv2d[(6, 6) (1, 1) (0, 0) @ 512], Tanh

Table 1. Architecture of the image encoder.

Layers	Speech Encoder
1	Conv1d[(80,) (16,) (32,) @ 16], BN, ReLU
2	Conv1d[(4,) (2,) (1,) @ 32], BN, ReLU
3	Conv1d[(4,) (2,) (1,) @ 64], BN, ReLU
4	Conv1d[(4,) (2,) (1,) @ 128], BN, ReLU
5	Conv1d[(10,) (5,) (3,) @ 256], BN, ReLU
6	Conv1d[(5,) (1,) (0,) @ 256], Tanh
7	GRU @ 256
8	GRU @ 256

Table 2. Architecture of the speech encoder.

Layers	Frame Discriminator
1	Conv2d[(4, 4) (2, 2) (1, 1) @ 32], BN, LeakyReLU
2	Conv2d[(4, 4) (2, 2) (1, 1) @ 64], BN, LeakyReLU
3	Conv2d[(4, 4) (2, 2) (1, 1) @ 128], BN, LeakyReLU
4	Conv2d[(4, 4) (2, 2) (1, 1) @ 256], BN, LeakyReLU
5	Conv2d[(6, 6) (1, 1) (0, 0) @ 1]

Table 3. Architecture of the frame discriminator.

Layers	Sequence Discriminator
1	Conv3d[(7, 4, 4) (1, 2, 2) (3, 1, 1) @ 64], BN, ReLU
2	Conv3d[(1, 4, 4) (1, 2, 2) (0, 1, 1) @ 128], BN, ReLU
3	Conv3d[(1, 4, 4) (1, 2, 2) (0, 1, 1) @ 256], BN, ReLU
4	Conv3d[(1, 4, 4) (1, 2, 2) (0, 1, 1) @ 256], BN, ReLU
5	Conv3d[(1, 6, 6) (1, 1, 1) (0, 0, 0) @ 128], Tanh
6	GRU @ 512
7	Linear @ 1

Table 4. Architecture of the sequence discriminator.

C. Experimental Details and Results

C.1. VSR Pre-training in Low-Resource Setting

Here, we discuss the VSR pre-training details as referenced in Sec. 4.4 of the main paper. Because supervised training VSR models from scratch with long sequences often pose optimization problems, we first use 30 hours of LRS3 and 944 hours of Librispeech synthetic data to pre-train a SMALL VSR model with 12-layer Conformer encoder, 6-layer Transformer decoder, 256 input dimensions, 2048 feed-forward dimensions and 4 attention heads (encoder and decoder have same dimensions and attention heads). The SMALL model is further fine-tuned using 30 hours of LRS3 data. We pre-train and fine-tune the SMALL model for 75 and 25 epochs, respectively. The VSR WER after pre-training and fine-tuning are 58.9% and 52.6%, respectively, as shown in Tab. 5. The other training hyper-parameters are the same as we used in Sec. 4.3. As the SMALL model has the same visual front-end as the BASE model, we initialize the visual front-end weights of the BASE model from the fine-tuned SMALL model for the low-resource labeled data setting.

C.2. VSR Baseline in High-Resource Setting

In Sec. 4.6 of the main paper, we only consider the BASE VSR model trained with 438 hours of LRS3 and 2630 hours of pseudo-labeled data as the baseline system, since we found LARGE model suffers from some con-

VSR model	Training data	WER (%)
SMALL-pretrain	LRS3 (30 hrs) + LBS-Synth	58.9
SMALL-finetune	LRS3 (30 hrs)	52.6

Table 5. WER of pre-trained (SMALL-pretrain) and fine-tuned (SMALL-finetune) SMALL VSR models. WER is calculated without using the language model.

vergence problems during training, which was potentially caused by its much larger model size.

C.3. Experimental Results for Synthetic Video Data with Multiple Lip Images.

Here, we studied the effect of the scale of Librispeech synthetic data (LBS-Synth) generated from the LAM-LRS3-VSR-VL model. We conducted one additional experiment using the BASE VSR model under the LRS3 labeled data setting. In Table 6, we show that double the size of LBS-Synth leads to further improvement (WER 30.8% to 30.1%). This is done by synthesizing two videos per speech clip with two CelebA lip images. This experiment further proves the flexibility and scalability of our synthetic data generation pipeline, which in principle could lead to unlimited video data for scaling up VSR.

Training data	Hours	WER w.o. LM (%)
LRS3 + LBS-Synth x1	438 + 944	30.8
LRS3 + LBS-Synth x2	438 + 1,888	30.1

Table 6. Multiple lip images coupled with a single speech leads to better performance. WER is calculated without using the language model.

References

[1] Pingchuan Ma, Alexandros Haliassos, Adriana Fernandez-Lopez, Honglie Chen, Stavros Petridis, and Maja Pantic. Auto-AVSR: Audio-visual speech recognition with automatic labels. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023. 1

[2] Pingchuan Ma, Stavros Petridis, and Maja Pantic. Visual speech recognition for multiple languages in the wild. *Nature Machine Intelligence*, 2022. 1