Learning Spatial Features from Audio-Visual Correspondence in Egocentric Videos Supplementary Material

Sagnik Majumder[1,2] Ziad Al-Halah[3] Kristen Grauman[1,2]
[1]UT Austin [2]FAIR, Meta [3]University of Utah

In this supplementary material, we provide additional details about:

- Video (with audio) for qualitative illustration of our pretext task and qualitative evaluation of our model on the downstream tasks (Sec. 1), as noted in Sec. 4.3 in main.
- Spatial audio denoising results with varying noise levels (Sec. 2), as mentioned in Sec. 4.2 in main
- Evaluation of the impact of the channel masking probability r (from Sec. 3.3 and 3.4 in main) in our audio masking protocol (Sec. 3)
- Analysis of the effect of alternate audio masking choices (Sec. 4), as referenced in Sec. 4.2 in main
- Study of the effect of multi-level positional embeddings (Sec. 5), as noted in Sec. 4.2 in main
- Analysis of the effect of task-specific backbones (Sec. 6), as mentioned in Sec. 4.2 in main
- Comparison of model capacity and computational cost among different pretraining methods (Sec. 7), as noted in Sec. 4.2 in main
- Qualitative analysis of the visual attention maps for left and right audio channels separately (Sec. 8), as referenced in Sec. 4.3 in main
- Analysis of the impact of our finetuning strategy (Sec. 9), as noted after Sec. 4.2 in main
- Evaluation of the impact of our model parameter initialization on the downstream performance (Sec. 10)
- Additional dataset details (Sec. 11), as mentioned in Sec. 4 in main
- Additional model architecture and hyperparameter details for both self-supervised pretraining and downstream training (Sec. 12), as referenced in Sec. 3.4 in main

1. Supplementary video

The supplementary video provides a qualitative illustration of our pretraining task for learning spatial features from audio-visual correspondence in egocentric videos, and our proposed approach. Moreover, we provide video samples from the both EgoCom [14] and EasyCom [2] datasets to illustrate the unique challenges posed by the egocentric videos. Additionally, we demonstrate our model's prediction quality for both active speaker detection and spatial audio denoising, and analyze common failure models for our model on both tasks. Please use headphones to hear the binaural audio correctly. The video is available on `http://vision.cs.utexas.edu/projects/ego_av_corr`.

2. Denoising with varying noise levels

In table 2 in main, we evaluated denoising with 0 dB noise. Here, we analyze the effect of varying the noise level. Table 1 reports the results with 2.5 dB and 5 dB noise. We observe general similarity in performance trends across all noise levels. Whereas our model outperforms the baselines in the high-noise settings (0 and 2.5 dB), using 2.5D-VS [4]++ improves the separation quality for 5 dB, underlining that our features are especially important for tackling the more challenging high-noise settings.

3. Channel masking probability r

Here, we analyze the effect of the channel masking probability r in our audio masking protocol (Sec. 3.3 in main) on the downstream task performance. Table 2 reports the active speaker detection (ASD) results on the more challenging EgoCom [14] dataset, and table 3 reports the denoising results for different noise levels. We notice that the performance on both ASD and denoising, especially at the higher noise levels, declines upon increasing or decreasing the value of r from our choice of 20 $\%$ based on the downstream validation performance (Sec. 3.4 in main), which helps our model achieve a fine balance between the two complementary strategies of masking a complete channel and randomly masking audio tokens. Whereas randomly masking a channel of the binaural audio entails solving the more under-constrained and consequently complex binauralization task, thereby helping our model learn stronger spatial associations between vision and audio, randomly masking audio tokens helps with improving training stability.

4. Alternate audio masking choices

Here, we evaluate alternate masking choices, namely time, frequency, and time-frequency masking, in place of randomly masking audio patches as part of our proposed mask-

Model	2.5 dB SI-SDRi ↑	2.5 dB STFT ↓	5 dB SI-SDRi ↑	5 dB STFT ↓
No pretraining				
U-Net w/o vision	1.91	5.32	2.02	3.04
U-Net	2.04	4.72	2.05	2.85
U-Net w/ ImageNet features	2.04	4.66	2.24	2.74
Alternate pretraining methods				
U-Net w/ TLR [19] features	1.70	5.40	2.00	2.77
U-Net w/ 2.5D-VS [4] features	1.81	4.81	2.22	2.62
U-Net w/ 2.5D-VS [4]++ features	2.65	4.31	**2.79**	**2.48**
U-Net w/ SSR [16]++ features	2.25	4.63	2.21	2.80
U-Net w/ AV-MAE [5] features	2.46	4.60	2.14	2.93
Ours	**2.72**	**4.22**	2.46	2.70
Ours w/o pretraining	2.30	4.54	2.15	2.83
Ours w/ pretraining using monaural audio	2.58	4.38	2.31	2.81

Table 1. Audio denoising with U-Net [19] backbone for varying noise levels. All STFT distance measures use base 10^{-3}.

$r(\%)$	TalkNet [17] Val	TalkNet [17] Test	SPELL [12] Val	SPELL [12] Test
0	67.9	62.9	67.6	65.3
20 (Ours)	**68.7**	**63.9**	**68.4**	**65.6**
50	64.5	63.1	67.5	65.2
80	64.4	61.8	64.7	60.1
100	67.9	63.4	66.1	65.1

Table 2. Effect of r on the mean average precision (%) of our model for active speaker detection with two different backbones (TalkNet and SPELL).

ing strategy, in table 4 and 5. Our model outperforms the versions with these alternatives, showing that random patch masking when combined with channel dropping enables learning more useful features in our setup. This happens possibly because in random patch masking, dropping a full frequency band or time segment is highly improbable thereby allowing our model to extract useful information from the unmasked regions of all frequency bands and time segments of the audio spectrograms.

5. Multi-level positional embeddings

Here, we evaluate the impact of our multi-level positional embeddings by comparing our model with the ablation where positional embeddings are used only at the input level. See table 6 for results on ASD and table 7 for results on denoising with 0 dB noise. Our model improves over the ablation on both tasks, showing that using multi-level positional embeddings is crucial for remembering the spatial layout of the tokens at different stages in the model.

6. Task-specific backbones

Here, we study the impact of using task-specific backbones on our model performance by evaluating two baselines, with the same architecture but without task-specific backbones (Ours w/o B)—one is learned from scratch and another is pretrained. See table 8 for results on ASD and table 9 for results on denoising with 0 dB noise.Our pretraining scheme leads to better performance than a from-scratch model even w/o B (table 8 and table 9 top), and we get the best results when we combine our features with task-specific backbones. This shows that while our audio-visual features provide important spatial cues to downstream models, they are not intended to replace the face-specific features used in ASD or the mixed audio features used in denoising.

7. Pretraining model capacity and computational cost

Here, we report the model capacity (parameter count) and GFLOPs of all pretraining methods in table 10. Note that both the parameter count and GFLOPs of all transformer [18]-based methods (2.5DVS [4]++, AV-MAE [5] and SSR [16]++) are comparable to those of our model[1], re-emphasizing that our improvements in performance on the downstream tasks are solely attributable to our better model design.

8. Visual attention maps per audio channel

In Fig. 1, we show the attention maps of our model (similar to Fig. 4 in main) separately for the left and right channels, on the more challenging EgoCom [14] dataset. We notice that our model uses the left channel to focus more on areas to the left of scene image and vice-versa, indicating that our

[1]The parameter count and GFLOPs of AV-MAE are a bit lower owing to its modality-inpainting architecture design, where the modality being inpainted is dropped from the input, leading to a slightly smaller model.

$r(\%)$	0 dB SI-SDRi ↑	0 dB STFT ↓	2.5 dB SI-SDRi ↑	2.5 dB STFT ↓	5 dB SI-SDRi ↑	5 dB STFT ↓
0	2.17	6.60	2.57	4.38	**2.85**	**2.35**
20 (Ours)	**2.20**	**6.51**	**2.72**	**4.22**	2.46	2.70
50	1.92	7.19	2.30	4.60	2.09	2.80
80	1.82	7.55	1.98	5.05	1.68	3.30
100	2.11	6.60	2.65	4.31	2.79	2.48

Table 3. Effect of r on our model performance for audio denoising.

	TalkNet [17]				SPELL [12]			
	EgoCom		*EasyCom*		*EgoCom*		*EasyCom*	
Model	Val	Test	Val	Test	Val	Test	Val	Test
Ours w/ time masking	63.3	59.3	**60.9**	66.4	64.1	60.7	68.5	43.5
Ours w/ frequency masking	64.1	62.1	59.2	70.9	67.6	63.2	68.7	69.4
Ours w/ time-frequency masking	65.4	63.1	56.3	63.3	67.5	65.1	68.6	69.1
Ours	**68.7**	**63.9**	60.5	**71.8**	**68.4**	**65.6**	**68.9**	**70.2**

Table 4. ASD with our model when pretrained with other audio masking choices [9].

Model	0 dB SI-SDRi ↑	0 dB STFT ↓	2.5 dB SI-SDRi ↑	2.5 dB STFT ↓	5 dB SI-SDRi ↑	5 dB STFT ↓
Ours w/ time masking	1.82	7.41	1.98	4.88	2.07	2.80
Ours w/ frequency masking	2.05	7.04	2.33	4.85	2.25	2.79
Ours w/ time-frequency masking	1.91	7.12	2.14	5.15	1.81	3.13
Ours	**2.20**	**6.51**	**2.72**	**4.22**	**2.46**	**2.70**

Table 5. Denoising with our model when pretrained with other audio masking choices [9]. All STFT distance measures use base 10^{-3}.

	TalkNet		SPELL	
Model	*EgoCom*	*EasyCom*	*EgoCom*	*EasyCom*
Ours w/o multi-level PEs	59.2	70.2	60.4	65.6
Ours	**63.9**	**71.8**	**65.6**	**70.2**

Table 6. Effect of our multi-level positional embeddings on ASD.

Model	SI-SDRi ↑	STFT ($\times 10^{-3}$) ↓
Ours w/o multi-level PEs	1.30	7.88
Ours	**2.20**	**6.51**

Table 7. Effect of our multi-level positional embeddings on denoising with 0dB noise.

	EgoCom		*EasyCom*	
Model	TalkNet	SPELL	TalkNet	SPELL
Ours w/o B (from-scratch)	61.1		62.0	
Ours w/o B (pretrained)	63.1		65.7	
Ours	**63.9**	**65.6**	**71.8**	**70.2**

Table 8. Effect of task-specific backbones (denoted using 'B') on ASD.

Model	SI-SDRi ↑	STFT ($\times 10^{-3}$) ↓
Ours w/o B (from-scratch)	1.02	8.99
Ours w/o B (pretrained)	2.05	7.12
Ours	**2.20**	**6.51**

Table 9. Effect of task-specific backbones (denoted using 'B') on denoising with 0dB noise.

model can reason about the spatial properties of the scene using both audio and vision.

Further, to better portray the larger trend, we measure the percentage of cases in our test set where the left audio channel attends more towards patches on the left side of the scene image, and the right channel attends more towards patches on the right. This measure comes out to be 62.4% for the left channel, and 57.2% for the right channel, showing that our model uses the left channel to focus more on areas to the left of the scene image and vice-versa, across the whole test set.

	Model parameter #		GFLOPs	
Model	ASD	Denoising	ASD	Denoising
2.5D-VS [4]	61.2	18.2	79.5	33.2
TLR [19]	57.5	18.1	75.9	33.7
2.5D-VS [4]++	180.9	75.3	174.0	90.2
AV-MAE [5]	178.6	74.1	171.6	87.5
SSR [16]++	180.9	75.3	174.0	90.2
Ours	180.9	75.3	174.0	90.2

Table 10. Model parameter count (in millions) and GFLOPs of different pretraining methods.

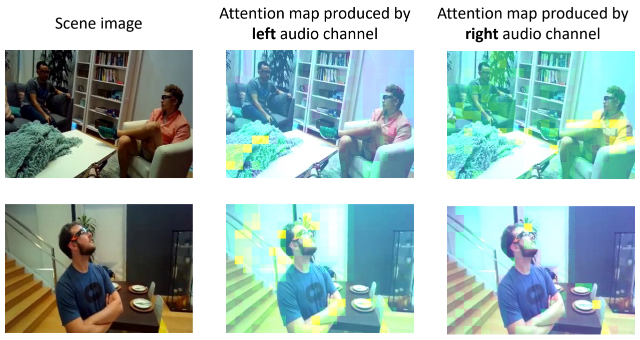
Scene image
Attention map produced by **left** audio channel
Attention map produced by **right** audio channel

Figure 1. Heat maps for left and right audio channels, similar to Fig. 4 in main. Interestingly, our model uses the left channel to focus more on areas to the left of scene image and vice-versa.

9. Finetuning strategy

We mask audio tokens during finetuning primarily to reduce the computation overhead. Since our pretraining also involves token masking, our model can learn strong audio-visual features even when all audio tokens are not available during finetuning. However, to quantatively evaluate the effect of our finetuning strategy, we also finetune our model with all audio tokens and report the results in table 11 for ASD and table 12 for denoising. We don't see a significant change in performance when using all the tokens compared to masking when finetuning, but using all tokens is 1.7 times slower on average.

10. Model parameter initialization

To evaluate the effect of random parameter initialization on our model, we train our model on both tasks and datasets with 3 different random seeds. Across all runs, our standard errors are less than 0.01 on all metrics, showing that our model is robust to different random parameter initializations, and the improvements in performance are significantly larger than these small variations from randomness.

11. Dataset details

As discussed in main (Sec. 4), we use two public datasets containing egocentric videos with binaural audio, EgoCom [14] and EasyCom [2], for our experiments. For EgoCom, we follow the authors and split the data into train/val/test comprising 30.3/2.4/5.8 hours of data. For EasyCom, we randomly generate train/val/test splits with 4.5/0.38/0.39 hours of data, such that there is no overlap in conversation participants between any two splits. Next, we extract 1 second long clips from both datasets, where the video and binaural audio are sampled at 5 frames per second (fps) and 16 kHz, respectively. The frame resolution is 240×352 for EgoCom, and 198×352 for EasyCom. Furthermore, we choose audio channel 5 and 6 (corresponding to the in-ear microphones) as our binaural audio channels for EasyCom.

12. Model architecture and training details

In addition to the provided details in Sec. 3.4, 4.1 and 4.2 in main, we provide here extra model architecture and training details for both pretraining and finetuning on downstream tasks, for reproducibility. We perform all training using 8 NVIDIA Tesla V100 SXM2 GPUs. We will release all code and data.

12.1. Pretraining

We described our model architecture and pretraining details in Sec. 3.4 in main. Here, we provide additional details about our model's input preparation, architecture, parameter initialization, and training .

Input preparation. We sample the video clips at their original resolution, normalize them using the per-color means and standard deviations computed using ImageNet [7], and generate a total of 330 and 286 visual tokens for EgoCom and EasyCom, respectively, by splitting the clips into non-overalapping tubelets containing a sequence of 5 patches, where each patch is 16×16 in size (Sec. 3 in main). We represent the binaural audio as two-channel Kaldi-compliant [15] spectrograms with 98 temporal windows and 128 Mel-frequency bins, which we compute by using the binaural audio normalized to $[-1, 1]$, a window length of 25 ms and a hop length of 10 ms. We normalize the spectrograms by computing the mean and standard deviation of the Mel-spectrograms generated from all audio clips in each dataset. We next generate 392 audio tokens per spectrogram channel by splitting it into non-overlapping patches of size 2×16.

Architecture. All hidden layers in each transformer block [3] emit features that are four times as long as the embedding size for the block. We always use LayerNorm [1] after every output of a transformer block unless it's a direct input to another transformer block.

	TalkNet [17]				SPELL [12]			
	EgoCom		*EasyCom*		*EgoCom*		*EasyCom*	
Model	Val	Test	Val	Test	Val	Test	Val	Test
Ours w/ finetuning all audio tokens	65.3	**64.1**	58.9	**71.8**	**68.4**	65.5	68.7	70.1
Ours	**68.7**	63.9	**60.5**	**71.8**	**68.4**	**65.6**	**68.9**	**70.2**

Table 11. ASD with our model when all tokens are used in downstream training.

	0 dB		2.5 dB		5 dB	
Model	SI-SDRi $\uparrow$	STFT $\downarrow$	SI-SDRi $\uparrow$	STFT $\downarrow$	SI-SDRi $\uparrow$	STFT $\downarrow$
Ours w/ finetuning all audio tokens	2.18	**6.47**	2.50	4.49	**2.58**	**2.52**
Ours	**2.20**	6.51	**2.72**	**4.22**	2.46	2.70

Table 12. Denoising with our model when all audio tokens are used in downstream training. All STFT distance measures use base 10^{-3}.

Parameter initialization. We use Xavier [6] uniform initialization for all network parameters. For the LayerNorm [1] layers, we initialize their weights to 1 and biases to 0. We use a truncated normal distribution with a standard deviation of 0.02 and a sampling range of $[-2, 2]$ to initialize the learnable modality and channel embedding tokens, and initialize the mask tokens from a normal distribution with a standard deviation of 0.02.

Training. We set the batch size to 104 and weight decay to 10^{-5} during pretraining.

12.2. Active speaker detection

In Sec. 4.1 in main, we outlined our feature fusion method for active speaker detection (ASD). Here, we provide additional architectural details for feature fusion, and also describe our finetuning process.

Pretrained feature fusion. Fig. 2 and 3 show our feature fusion methods for TalkNet [17] and SPELL [12] ASD backbones, respectively. The single-layer transformer decoder (Sec. 4.1 in main), which we use for fusing our pretrained features with the backbones (Sec. 4.1 in main), generates 128 and 512 dimensional embeddings for TalkNet and SPELL, respectively. Since SPELL doesn't train any audio-visual features when training its graph neural network (GNN), we first pretrain the the transformer decoder for SPELL by using it with the TalkNet backbone. Towards that goal, we feed the decoder features to a single linear layer that maps the 512 dimensional features to 128 dimensional features, and is followed by GELU [8] activations and LayerNorm [1], before fusing the 128 dimensional features with the TalkNet backbone. After pretraining, we append the 512 dimensional outputs of the decoder with the outputs of the two-stream audio-visual encoder (Sec. 4.1 in main) for training the GNN in SPELL.

Training. For TalkNet, we train using Adam [11] for 25 epochs optimizer with an initial learning rate (LR) of 10^{-4} for the backbone and 10^{-5} for the pretrained components, both of which we decay using a step LR scheduler by a factor of 0.95 after every epoch. We set the batch size to 400.

For SPELL, we first train the two-stream audio-visual encoder for feature extraction for 100 epochs using the cross entropy loss and Adam [11] with an initial learning rate of 5×10^{-4}, which we decay by 0.1 after every 40 epochs. We set the batch size to 320. For training the GNN of SPELL, we train for 70 epochs by using a batch size of 320 again and learning rate of 10^{-3}, while setting all other hyperparameters per the original paper.

12.3. Spatial audio denoising

Backbone architecture. Following [19], our U-Net backbone for spatial audio denoising (Sec. 4.2 in main) is an audio-visual model comprising an audio encoder, a visual encoder, and a decoder for predicting an estimate of the target audio. The audio encoder takes the log magnitude spectrogram of the mixed binaural audio as input, and uses a stack of 5 convolutional (conv.) layers to produce a multi-channel 2D audio feature map. Each conv. layer has a kernel size of 4, padding of 1, and stride of 2, and is followed by leaky ReLU [13] activations with a slope of 0.2 for negative inputs, and batch normalization [10]. The conv. layers have 64, 128, 256, 512 and 512 output channels, respectively. The visual encoder has a ResNet-18 [7] architecture that outputs a multi-channel 2D visual feature map without feeding it to the average pooling or any subsequent layers. We push the ResNet outputs through another conv. layer to match its height and width with the audio features. The conv layer has a kernel size of (1, 4), a padding of (0, 0) for EgoCom [14] and (1, 0) for EasyCom [2], and 128 output channels. Further, we remove the last feature column from the output of the conv. layer for all channels for EasyCom. We concatenate the per-frame features along the channel dimension

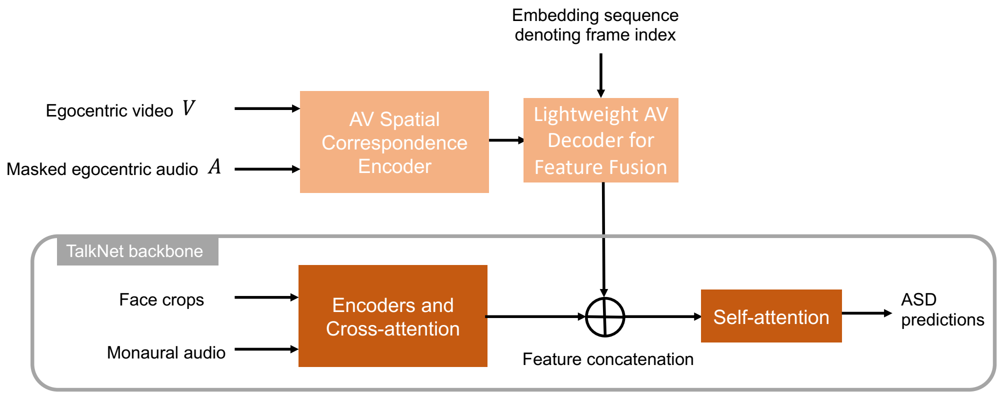
Embedding sequence denoting frame index
Egocentric video V
Masked egocentric audio A
AV Spatial Correspondence Encoder
Lightweight AV Decoder for Feature Fusion
TalkNet backbone
Face crops
Monaural audio
Encoders and Cross-attention
Feature concatenation
Self-attention
ASD predictions

Figure 2. Method to fuse our pretrained features with TalkNet [17] for ASD.

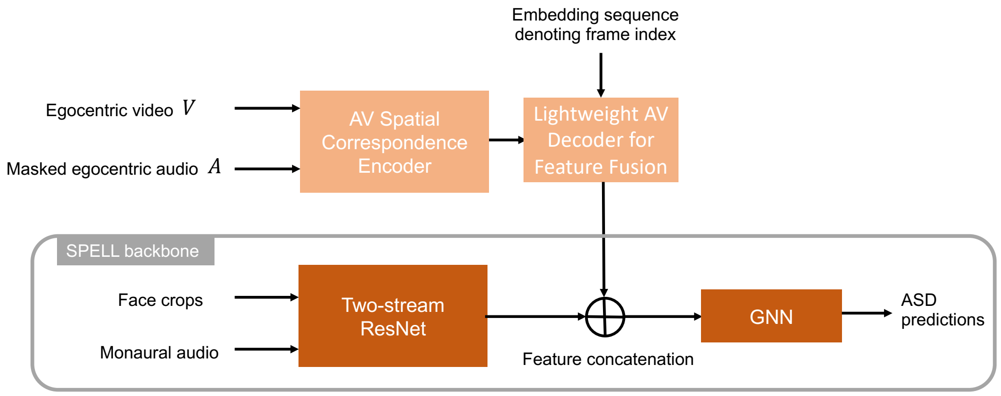
Embedding sequence denoting frame index
Egocentric video V
Masked egocentric audio A
AV Spatial Correspondence Encoder
Lightweight AV Decoder for Feature Fusion
SPELL backbone
Face crops
Monaural audio
Two-stream ResNet
Feature concatenation
GNN
ASD predictions

Figure 3. Method to fuse our pretrained features with SPELL [12] for ASD.

and generate the visual features. We then concatenate the visual features with the audio features channel-wise, and feed the concatenated features to the audio decoder, which predicts an estimate of the ratio mask [4, 19] for the target audio magnitude spectrogram. The audio decoder first uses a stack of 5 transpose convolutional (conv.) layers, which are connected to the corresponding encoder layers through skip connections. The transpose conv. layers have a kernel size of 4, stride of 2, and a padding of (1, 1), except for the last layer, which has a padding of (2, 1). The transpose conv. layers have 1152, 1024, 512, 256 and 128 output channels, respectively. Next, the audio decoder feeds the output of the transpose conv. layers to a conv. layer with 2 input and output channels, and a kernel size of (2, 1) to emit the predicted ratio mask.

Input preparation. To transform the audio waveforms into magnitude spectrograms, we first normalize them to [-1, 1] and then compute the short-time Fourier transform with a window length of 128, hop length of 64, and 512 frequency bins.

Pretrained feature fusion. Fig. 4 shows our feature fusion method for spatial audio denoising. We reshape the visual features from the outputs of our audio-visual encoder $\mathcal{E}^{AV}$ to form multi-channel 2D visual feature maps (Sec. 4.2 in main), such that the 2D raster order of the features matches that of the tubelets in the video clip, and feed the reshaped features to a convolutional (conv.) layer with a kernel size of (3, 4), stride of (2, 3), padding of (1, 2) and (2, 2) for EgoCom [14] and EasyCom [2], respectively, and 128 input and 784 output channels. We similary reshape the audio features, and feed them to another conv. layer with a kernel size of (1, 7), padding of 0, stride of (1, 6), and 128 input and 256 output channels. Both conv. layers are followed by leaky ReLU activations with a slope of 0.2 for the negative values, and batch normalization. Next, we concatenate the visual and audio features along the channel dimension, and further concatenate them with the audio encoder outputs channel-wise (Sec. 4.2 in main).

Training. We train using Adam [11] for 200 epochs optimizer with an learning rate (LR) of 5×10^{-4}. We set the batch size to 80.

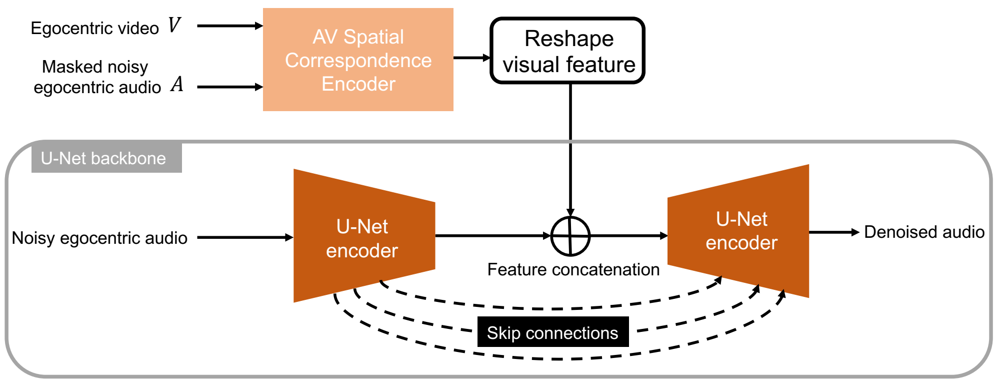

Figure 4. Method to fuse our pretrained features with U-Net [19] for spatial audio denoising.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4, 5

[2] Jacob Donley, Vladimir Tourbabin, Jung-Suk Lee, Mark Broyles, Hao Jiang, Jie Shen, Maja Pantic, Vamsi Krishna Ithapu, and Ravish Mehra. Easycom: An augmented reality dataset to support algorithms for easy communication in noisy environments. 2021. 1, 4, 5, 6

[3] Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *Advances in neural information processing systems*, 35:35946–35958, 2022. 4

[4] Ruohan Gao and Kristen Grauman. 2.5D visual sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 324–333, 2019. 1, 2, 4, 6

[5] Mariana-Iuliana Georgescu, Eduardo Fonseca, Radu Tudor Ionescu, Mario Lucic, Cordelia Schmid, and Anurag Arnab. Audiovisual masked autoencoders. *arXiv preprint arXiv:2212.05922*, 2022. 2, 4

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 2010. 5

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 5

[8] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016. 5

[9] Po-Yao Huang, Hu Xu, Juncheng Li, Alexei Baevski, Michael Auli, Wojciech Galuba, Florian Metze, and Christoph Feichtenhofer. Masked autoencoders that listen. *Advances in Neural Information Processing Systems*, 35:28708–28720, 2022. 3

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, Lille, France, 2015. PMLR. 5

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 5, 6

[12] Kyle Min, Sourya Roy, Subarna Tripathi, Tanaya Guha, and Somdeb Majumdar. Learning long-term spatial-temporal graphs for active speaker detection. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, pages 371–387. Springer, 2022. 2, 3, 5, 6

[13] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML 2010*, pages 807–814, 2010. 5

[14] Curtis Northcutt, Shengxin Zha, Steven Lovegrove, and Richard Newcombe. Egocom: A multi-person multi-modal egocentric communications dataset. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020. 1, 2, 4, 5, 6

[15] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011. IEEE Catalog No.: CFP11SRW-USB. 4

[16] Tomoya Sato, Yusuke Sugano, and Yoichi Sato. Self-supervised learning for audio-visual relationships of videos with stereo sounds. *IEEE Access*, 10:94273–94284, 2022. 2, 4

[17] Ruijie Tao, Zexu Pan, Rohan Kumar Das, Xinyuan Qian, Mike Zheng Shou, and Haizhou Li. Is someone speaking? exploring long-term temporal features for audio-visual active speaker detection. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3927–3935, 2021. 2, 3, 5, 6

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[19] Karren Yang, Bryan Russell, and Justin Salamon. Telling left from right: Learning spatial correspondence of sight

and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9932–9941, 2020. 2, 4, 5, 6, 7