IM4WAV: I HEAR YOUR TRUE COLORS IN STEREO

Jinghao Miao, Akshat Katoch, Chengqian Zhang
Georgia Tech
{jmiao38,akatoch3,czhang657}@gatech.edu
IM4WAV GitHub Repo

1. Abstract

This paper introduces a novel redesign to an image-to-audio model designed to enhance a current multi-modal technique improving experiences in computer vision by converting images into stereo audio. Current research in this field predominantly focuses on text and image interpretations, but significantly lacks in auditory dimensions. Our model addresses this gap by enabling the translation of visual content into immersive and engaging sound experiences. Our proposed model, inspired by the "I Hear Your True Colors: Image Guided Audio Generation" paper from Roy Sheffer and Yossi Adi, extends the IM2WAV by incorporating stereo audio to simulate a more realistic and immersive auditory environment. Our model redesigns the processing of images to identify objects, analyze their positions and depths with DINOv2 [10] to do sound localization, encode the segmented image into latent space with pre-trained CLIP models [11], generate a corresponding audio file with VQ-VAE [12], and then apply the sound localization to the outputs. For instance, it can differentiate and spatially represent sounds from varying sources within an image, like conversations from a crowd on one side and car noises on the other, in a stereo format. For the audio generation part, we built upon the basis of IM2WAV's VQ-VAE structure. Eventually, we successfully outperformed IM2WAV for the stereo audio generation performance by testing left and right channels separately. This is a model for stereo audio generation while only requiring one single image, making it probably the easiest and the most interesting model to use for image-to-audio generation.

2. Introduction

In the current computer vision multimodal research, images are frequently interpreted as text or other images: there are text-to-image, image-to-text, image-to-image, etc. However, there is one lacking human sense which has not been researched deeply: sound. An image-to-audio model is important because incorporating sound with visual content can enhance the overall experience for all users: making applications, websites, and media more engaging and immersive, empowering sound designers with more useful references, and so on. For a simple instance, in educational materials, adding sound to images can make learning more interactive and memorable. Therefore, we propose an image-to-audio model that takes in an image as input and outputs a stereo audio that corresponds to the features of the image. In this report, we will introduce the model inspired by the paper "I Hear Your True Colors: Image Guided Audio Generation" by Roy Sheffer and Yossi Adi. Honestly, the IM2WAV proposed by them is already pretty great, but there's one thing missing: human beings usually have two ears. If we want to fully experience the sounds from an image, we have to make ourselves feel like we are actually in that environment. This is where the stereo setting becomes important. To fulfill all the possible utilization mentioned above, making sound stereo is a necessary step. After the literature survey, we did find some useful papers. "Audiogen: Textually guided audio generation" by Felix Kreuk provides us with intuitive ideas about how to generate audio from CLIP outputs; "Fréchet Audio Distance: A Metric for Evaluating Music Enhancement Algorithms" by Kevin Kilgour et al. provides us with a widely used evaluation metric;

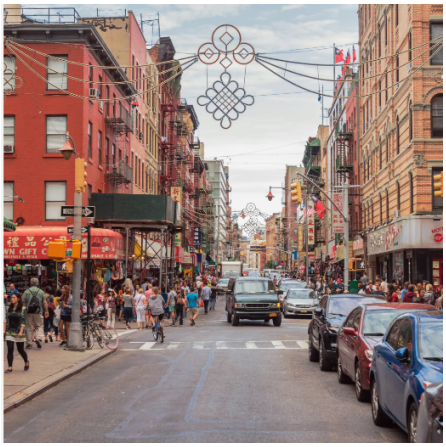

Figure 1. Example of image including crowds and cars

"DINOv2: Learning Robust Visual Features without Supervision" by Maxime Oquab et al. provides us a strong backbone model to produce volume factors necessary for stereo sound production [4,5,7,10,13]. Our image-to-audio model will receive a random image as input, identify the objects included in the image, analyze the position and depth of the objects, and finally produce a stereo audio file that corresponds to all the content shown in the image. For example, if there are crowd talking on the left sidewalk and cars on the right side of the road, our model will produce a stereo sound which contains the sound of the crowd talking and the car driving through (Fig. 1).

3. Technical Approach

In our model, the same image is sent into two DINOv2 models in parallel. The two DINOv2 models are connected to two different fine-tuned heads: the former performs the depth estimation of the image, and the latter separates the original images into segments. The output of depth estimation and segmentation are sent to the physical-modeled sound localization layer in Figure 2, which translates the depth information and the position information into the left and right channel volumes of the sounds created by the objects. The next portion of the model utilizes the CLIP model which is a pre-trained model developed by OpenAI that uses contrastive learning to match images to corresponding texts. The segmentation outputs are cropped and sent into CLIP models in parallel: each segmentation is sent into one CLIP separately. At the same time, the full original image is also sent into one specific CLIP layer to get general background information from the image. The CLIP model is given all the classes in our dataset and then matches our input with the most appropriate class. After getting the tokens from the CLIP models, we will put them into parallel VQ-VAE Audio Encoder-Decoder models [12] to generate audio outputs, and all the outputs will be summed together as the output of the model after being multiplied by the volume factored from the output of the sound localization layer. Fig. 2 shows the procedure diagram of our model.

3.1. DINOv2, Depth Estimation and Segmentation

DINOv2 is a model that simplifies the use of images in any system by producing high-performance visual features that can be directly employed with classifiers as simple as linear layers on a variety of computer vision tasks [10]. The output of DINOv2 can be connected to different types of linear heads for different computer vision tasks. In our model, we used 2 pre-trained heads of DINOv2: depth estimation and image segmentation. Depth estimation will predict the depth of each pixel in the image in order to get the distance information of objects, while image segmentation will segment objects in an image for further sound prediction per object. In our model, we are using the pre-trained heads of both depth estimation and segmentation downloaded from the DINOv2 GitHub page. We add a filter to the outputs of the segmentation model by checking the size of each segment. If a segment is too small, then we directly ignore it to keep the validity and efficiency of our model. We hoped the combination of the background image and the information about the object itself, would allow us to improve the performance of the audio generator.

3.2. CLIP Model

After the DINOv2 layers, the inputs are sent into the CLIP model separately. CLIP (Contrastive Language-Image Pre-Training) is a pre-trained neural network trained on a variety of (image/video, text) pairs to map the vision information to text tokens. During our training process, we utilized the video CLIP model to further train the VQ-VAE in our second training stage. More information can be found in the training section.

3.3. Sound Localization

To localize the sound source in the image, we want to consider both the distance and position information. Fortunately, because we are building a stereo audio model, only left and right will matter but not up and down. So, We first get the relative positions of the segmented images to the original image and then calculate the x coordination of the center of the segmented images. Next, we project the positions of the objects to the x-axis and calculate the percentage of the segment's horizontal offset. After that, the position information will be combined with the depth estimation result with a physically modeled formula where we can tune the weights, as closer objects should sound louder, and finally become the factor of volume for each object in separated left and right channels.

For example, based on the image we use in Fig. 2, we detected 6 segments along with their distance and position information. By combining these two types of information, we calculate each segment's left and right channel volume factors, as shown in Fig. 3. The left channel volume factors are shown in blue circles, while the right ones are shown in pink, with their radius showing the factors' numerical magnitude linearly. We can easily see that all the segments on the left side have larger blue circles, and all the segments on the right side have larger pink circles, showing the horizontal position relationships. In addition, for instance, since the human being, which is segment 3, is much closer than the plants behind, which is segment 2, she has an obviously larger volume factor. Same as all the segments. All this information can be very useful in predicting the correct sound positions in a stereo manner.

The following formula is the physically modeled sound localization algorithm. Because the sound intensity has a squared relation with the distance, we can get

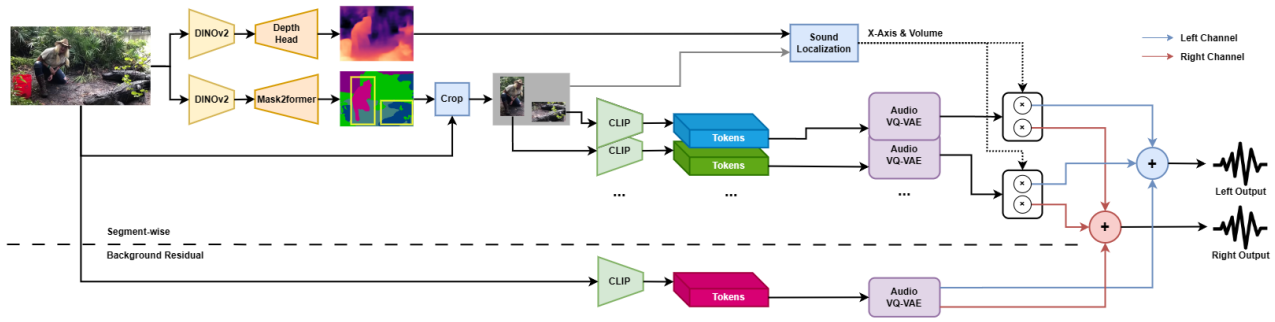

Figure 2. Procedure Diagram of IM4WAV

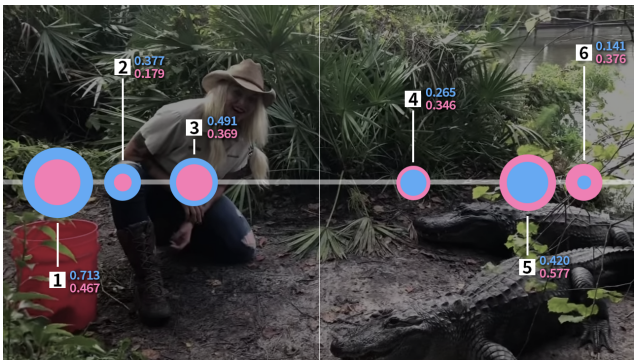

Figure 3. Volume Factor Visualization

$$\text{Volume Factor}_{\text{Depth}} = \text{Depth}^2_{[0,1]}$$

Because the standard angle of cameras and eyes is around $50°$ ($\frac{5\pi}{18}$ in radian), we can have the following equation

$$\alpha = \tan^{-1}\left(\frac{2x}{y} \cdot \tan\left(\frac{5\pi}{36}\right)\right)$$

where x is the distance between the object on the screen and the central point, and y is the width of the screen.

We make sure that α is always non-negative.

There will be two situations. If the object is on the left half of the screen,

$$\text{Volume Factor}_{\text{Horizontal, LeftO, LeftC}} = \cos\left(\frac{\frac{\pi}{2} - \alpha}{2}\right)$$

$$\text{Volume Factor}_{\text{Horizontal, LeftO, RightC}} = \sin\left(\frac{\frac{\pi}{2} - \alpha}{2}\right)$$

If the object is on the right half of the screen,

$$\text{Volume Factor}_{\text{Horizontal, RightO, LeftC}} = \cos\left(\frac{\frac{\pi}{2} + \alpha}{2}\right)$$

$$\text{Volume Factor}_{\text{Horizontal, RightO, RightC}} = \sin\left(\frac{\frac{\pi}{2} + \alpha}{2}\right)$$

Then, we get the final sound localization volume factor by combining these two factors,

$$\begin{aligned}\text{Volume Factor}_{\text{LeftC}} = {} & w_1 \times \text{Volume Factor}_{\text{Depth}} \\ & + w_2 \times \text{Volume Factor}_{\text{Horizontal, LeftC}}\end{aligned}$$

$$\begin{aligned}\text{Volume Factor}_{\text{RightC}} = {} & w_1 \times \text{Volume Factor}_{\text{Depth}} \\ & + w_2 \times \text{Volume Factor}_{\text{Horizontal, RightC}}\end{aligned}$$

Where w_1 and w_2 are weights that can be tuned but they always add up to 1. In our implementation, both weights are set to 0.5 after testing the validity.

3.4. Volume Factor Multiplication

After the audio files are generated, there is no factor that takes the volumes into account. Thus, we retrieve the output from the sound localization result to solve this problem. For each object in the image, the output from sound localization is a tuple that contains two float factors for the left and right channels respectively. The output generated from the VQ-VAE Audio Encoder-Decoder models will be multiplied by the volume factors in amplitude separately so that the objects' sounds have the correct portion in the channels.

3.5. VQ-VAE Audio Encoder-Decoder

VQ-VAE Audio Encoder-Decoder is a one-dimensional hierarchical VQ-VAE architecture. [12] This architecture is

composed of multiple levels of detail (2 in our case) to capture features with different sizes, which can be helpful in producing detailed sound quality. It will first be trained singly to map audio to latent spaces, which uses the same code book as our text tokens, then be trained with the video CLIP model to learn the code book for conditional generation, and eventually be used as a whole to produce audio. More information and utilization can be found in the Training section.

3.6. Stereo Output

Each VQ-VAE Audio Encoder-Decoder generates one soundtrack of one object which is then multiplied by volume factors into two channels and summed up together to get the final output of the model. We have one more generator which gains the information from the whole picture to produce the background sound. We first thought about adding the volume factors into the tokens output from the CLIP models in a way similar to positional encoding, but we realized that it would not work in a stereo manner. If we decide to use two separate audio encoder-decoders to produce left and right channel sounds, due to the probability-distribution-based property of transformers, we may not be able to get exactly the same sound for two channels, which is essential if we want that sound to be stereo. This assumption needs further research to prove. What we want the output to be like is two same sounds but with different volumes. Multiplying the different volume factors by the same sound generated by one single encoder-decoder will give us the ideal result.

3.7. Baseline Model and Evaluation Method

We compare the proposed method to the IM2WAV Model proposed by R. Sheffer and Y. Adi [13] as well as our own small custom base model. Compared to the IM2WAV, the model we proposed makes a difference in depth estimation and image segmentation. In the IM2WAV model, there is only an image classification model in the architecture, thus it can only produce sound for the whole image together without any information in volume. The custom base model utilizes the clip model to convert the images to a text class. The text class is then tokenized and then fed to a VAE audio decoder to learn the corresponding sound. Moreover, as it is a broad whole-image classification, the sound cannot be stereo but only a duplication of one soundtrack for both left and right channels. The evaluation metric we use is the Frechet Audio Distance metric. The metric extracts features for the generated and real audio through pre-trained models. Then we model each set of features with a Gaussian distribution, characterized by a mean and covariance. The FAD is the distance between these two Gaussian distributions, quantifying the similarity between the generated and real audio feature spaces. We also added a KL Divergence as another evaluation metric. More information can be found in the Results and Analysis section.

4. Data Collection Pipeline

We have a data collection pipeline for our training and testing dataset. We downloaded the VGGSound [1] dataset as our training and testing dataset. The videos in the dataset are downloaded from YouTube by using functions from the yt_dlp package. We first convert the videos with different types to mp4 and filter all the videos under 25 fps for dataset quality. Next, we picked video classes that contain more than 40 videos in order to get enough training and testing data for each class. Then the videos are clipped into images: for every video, we collect the frames from the 1st, 5th, and 9th seconds of the videos as the image input, while the audio of the videos acts as the expected output of our model in training and testing. The reason we did this was to diversify the images so they were all different. Till now, all the data are prepared for training and inferencing.

5. Training

The training process is separated into two stages. [13] Since some of our models are already pre-trained, such as the DINOv2 and the CLIP, so the part we need to train is the VQ-VAE audio generation model. The first stage is to train a general VQ-VAE. The main goal of this stage is to establish the latent space of audio information. In this stage, the input and output are both audios, meaning train the model to learn the internal lower-dimension distribution of the audio.

The second stage incorporates video CLIP models. We first use video CLIP models to encode all the video visual information into latent features and then integrate the video visual information together with the extracted audio information as the actual inputs. In this stage, we train the VQ-VAE to learn the matching of the audio latent space and the video visual latent features in order to implement the image-guided audio generation process in the inference stage. One key point of this training stage is two separate models with different detail levels. This is one of the characteristics of VQ-VAE, using a lower-level model to generate more abstract audio tokens and a higher-level model to generate more details on the foundation of the previous model. [12]

The loss functions we used are the same loss functions from the IM2WAV paper. There are three loss functions, the reconstruction loss, the commitment loss [12], and the STFT spectral loss [2]. The reconstruction loss measures the difference between the original and reconstructed audio in the time domain. The commitment loss is the distance between the output of the encoder and the selected codebook vector. It encourages the encoder to produce outputs that are close to the desired codebook vector. The spectral loss is the difference between the reconstructed and original

audio in the frequency domain.

However, with limited computational units, after we trained the model for 50 epochs, it still failed to converge, which negatively affected the output results and caused total failure for the second stage (unstable and increasing loss). We have tried many groups of possible hyperparameters, using GCP as the training platform, and shrinking the size of the training dataset, but it still failed to produce the ideal model for further training and inference. We realized that handling a video model is too much work for our computational units. Therefore, we finally decided to use the pre-trained first-stage VQ-VAE for further training. In our second training stage, the model is trained on 500 videos which are divided into 25 classes, meaning 20 videos per class. By training 50 epochs for each level, we got an ideal audio generation model that will work for both IM2WAV and our IM4WAV.

One thing worth mentioning is that even though our second training stage worked as expected, it didn't reach its full capability. We can tune the batch size, the learning rate, and the epochs during training. Without enough GPU memory, we can only pick 5 as our batch size, reaching the limit we have. More information will be given in the Result section. However, this doesn't affect our improvements because our main innovations go into the image processing part instead of the audio generation part. This means that we can use the same audio generation model to test and compare the performances.

6. Inference

The inference process contains our main innovation. As physically defined, sounds all have their origins. So, the first step is to get the positions of those sound origins and then generate audio for each object. We first input the image into our sound localization and segmentation pipeline introduced above. Then we get segments for each large enough object in the image along with their calculated volume factors for left and right channels separately. The next step is to overlap all the audio of objects within one image after multiplying by their relative volume factors. We have tested the stereo effect, which sounds great.

With the limit of computational resources, we have to decrease the batch size while generating audio due to the relatively small GPU memory we have. This largely restricts our inference efficiency. We designed this model to be able to generate audio for all the segments in parallel, but this requires large GPU memory.

7. Results and Analysis

(Several audio output examples have been submitted as supplementary materials.)

As shown in Figs. 4 and 5, we are able to get pretty decent depth and segmentation results from the state-of-the-art image featuring model DINOv2 [10].

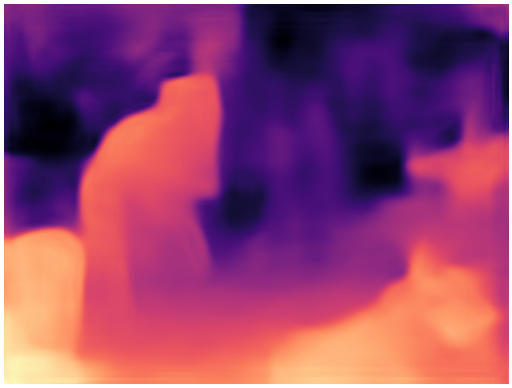

Figure 4. Image Depth Visualization

Figure 5. Image Segmentation Visualization

Model	$FAD_1 \downarrow$	$FAD_2 \downarrow$	$FAD_3 \downarrow$	$FAD_4 \downarrow$	KL↓
IM2WAV	128.80	33.66	0.0053	1.82	1.92
IM4WAV	**125.73**	**33.51**	**0.0044**	**1.75**	**1.68**

Table 1. Main results tested with Left and Right channels separately

For evaluating our model, we use the same metrics proposed in the im2wav paper but in a stereo manner, which means that we evaluate left and right channels separately. [13] Frechet Audio Distance is proposed by Kilgour et al [5], which measures the distance between the generated and real distributions. The generated and real audio files are modeled as multivariate normal distributions.

$$FAD = ||\mu_r - \mu_g||_2 + tr(\Sigma_r + \Sigma_g) - 2\sqrt{\Sigma_r \Sigma_g}$$

Where μ_r and μ_g represent the means of the distributions and Σ_r and Σ_g represent their variances. We implemented the FAD scores based on the process described in the im2wav paper [13]. First, we found a model that extracts the features of the audio files [3]. After the extractions are completed, the outputs are in shape $(length, 128)$,

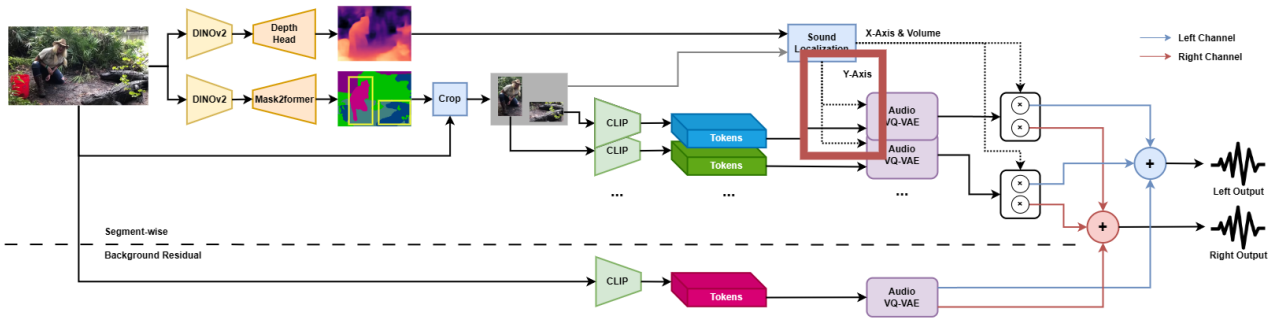

Figure 6. Future Procedure Diagram of IM4WAV with Y-axis Positional Information Encoding

where $length$ represents the number of seconds the audio file lasts. For each generated and original audio file the outputs are passed into the FAD calculation process that we implemented. What's more, we found an FAD implementation on this GitHub page. There are three audio transformation models we can choose from, so we tested our models on all three of them. As shown in table 1, $\text{FAD}_{2,3,4}$ represents the FAD scores calculated with the VGGish model, PANN model, and CLAP model respectively. We also implemented an FAD score ourselves by using a normal CNN architecture, and KL Divergence as parts of our evaluation metrics. One important point is that our scores are evaluated based on the models we trained ourselves. The models we trained are not the best among all existing ones, so the divergences might look pretty large. The reason it's important is that we can use these scores to compare the performance between IM4WAV and IM2WAV structures under the same circumstances.

For the test set, due to the low dataset quality, we manually picked 12 videos from the 25 classes we chose earlier with stereo sound available. It's important to pick videos with stereo audio because that's the key point of our IM4WAV model. Since IM2WAV is a mono audio generation model, we use the single channel for both left and right channel testing. The reason for manually picking videos is because of the imbalance of quality of the dataset we are using. This dataset heavily relies on YouTube videos, where the qualities and labels may vary. We have found videos with characteristics that make them bad testing data listed as follows:

1. **Mono audio:** These types of videos don't influence our training procedure because we don't require stereo videos to train. But they are useless during the testing stage.

2. **Stereo audio but incorrect:** There are videos with stereo audio available but coming from incorrect directions. For example, we can see a person on the left of the video but the audio is coming from the right. These types of videos are useless for testing.

3. **Background music:** There are many videos coming with background music which makes them useless for both training and testing.

4. **Visual information not matched with audio:** Many videos are clipped from TV shows or other similar origins. These videos might include very frequent camera switching without any change in audio. So it's hard to generate the correct audio while the visual information is changing quickly.

In addition, with the limit of computational resources, we can only use a limited number of testing image-audio pairs. In this situation, the quality of the testing set is even more important.

One interesting discovery is that, while we are evaluating both models on very obvious stereo videos, which were picked from the dataset manually, our IM4WAV model performed better than IM2WAV with a difference of 10 according to the FAD score with the VGGish model (the difference was 0.1 shown in the table above). However, since this is a relatively biased and unfair testing set, we only mention this here instead of treating it as a serious evaluation.

8. Discussion

We have proved that our model does bring some improvements in generating stereo audio. There are several improvements that we haven't tried due to the time limit but are hopeful to improve the performance more:

1. **Better dataset:** We are using a dataset called VGGSound right now. It's a low-quality dataset with many videos not in stereo but mono. Also, there is a large portion of videos containing audio not related to the visual elements, so we're expecting lower-than-average results on this dataset. We found a better

dataset: RWAVS from another video-to-audio model called AV-Nerf [8]. So we do need more time to switch to this dataset after this semester.

2. **Better object detection model:** We found a better model for image object detection, which is called Grounded-SAM [6, 9], which might work better instead of Mask2Former.

3. **Y-axis positional information encoding:** As marked in the figure Fig. 6, we're considering the possibility of encoding the Y-axis positional information of the object into the VQ-VAE model in a positional encoding manner. This might help the model produce better spatial sound effects. Or even, if it's possible, we can encode both X-axis and Y-axis information into the VQ-VAE instead of multiplying the factors to the outputs, which will need further research.

4. **Better audio integration method:** If there's a better way to combine all the objects' sounds at the end of this model instead of simply adding them up.

5. **End-to-end sound localization:** Make the "Image to Sound Localization" pipeline end-to-end, which means train a newly designed header to take charge of it.

6. **More evaluation metrics:** There are some other available evaluation metrics, such as Clip-score and Accuracy.

There are some shortcomings of our IM4WAV model. For example, it's not able to detect if a sound is blocked or not, such as when a person is sitting inside a car with windows closed; it's not able to generate reliable audio if the image's visual information is messy because it will interrupt the segmentation results; it requires more GPU memory to go through the whole pipeline due to the segmented audio generation design; it's not able to generate audio for objects not in the training set, while it's good at generalization for objects trained. These are some directions in which future research can be implemented.

9. Team Contribution Table

Group Member	Contributions
Jinghao Miao	Implemented data collection and preprocessing, Designed IM4WAV architecture, Designed and implemented Sound Localization with DINOv2, Integrated Sound Localization into image CLIP model, Trained the Up and Low level VQ-VAE models, Inferenced with both IM2WAV and IM4WAV models, Proposed shortcomings and possible future improvements, Wrote the report, Attended group meetings.
Akshat Katoch	Design IM4WAV architecture, Implemented KL divergence evaluation function, Implemented the conversion of mono audio to stereo audio, Incorporated volume factors into the generated output, Helped implement the CLIP model, Helped create the final data for evaluation Wrote the report, Attended group meetings.
Chengqian Zhang	Implemented FAD evaluation function, Searched for the original paper for CNN architecture which extracts audio features, Cloned and revised the pre-trained feature extraction model, Implemented the Split Function of stereo audio to mono audio Designed and Wrote the poster, Wrote the report, Attended group meetings.

References

[1] Honglie Chen, Weidi Xie, Andrea Vedaldi, and Andrew Zisserman. Vggsound: A large-scale audio-visual dataset, 2020. 4

[2] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020. 4

[3] Shawn Hershey et al. Cnn architectures for largescale audio classification. *2017 ieee international conference on acoustics, speech and signal processing (icassp). IEEE, 2017, pp. 131–135.*, 2017. 5

[4] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models, 2022. 2

[5] Kevin Kilgour, Mauricio Zuluaga, Dominik Roblek, and Matthew Sharifi. Fréchet audio distance: A metric for evaluating music enhancement algorithms, 2019. 2, 5

[6] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 7

[7] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation, 2023. 2

[8] Susan Liang, Chao Huang, Yapeng Tian, Anurag Kumar, and Chenliang Xu. Av-nerf: Learning neural fields for real-world audio-visual scene synthesis, 2023. 7

[9] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. 7

[10] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 1, 2, 5

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[12] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 1, 2, 3, 4

[13] Roy Sheffer and Yossi Adi. I hear your true colors: Image guided audio generation, 2023. 2, 4, 5