Deeply Visual Microphone

Nischal B Krupashankar, Arnaav Anand, N C Sathya, Bhanuprakash Narayana

Luddy School of Informatics, Computing & Engineering,
Indiana University. Bloomington

Abstract

Roughly a decade ago, there was a breakthrough in the field of computer vision where a team at MIT demonstrated a research wherein they utilized high-speed video footage of sound interacting with an object's surface and captured minuscule vibrations to partially reconstruct the original sound waves. We are pioneering a novel approach aimed at determining whether replacing the traditional mathematical methods used in the primary study with deep learning techniques would enhance sound extraction. We use a pseudo-Siamese network with pre-trained ResNet50 model and a custom CNN architecture on the original data set augmented with our own data set to demonstrate a proof of concept. We further delve into feature pyramid networks to enhance the reasoning capability of the model.

1. Introduction

Sound waves represent fluctuations in pressure propagating through a medium. When sound encounters an object, it induces movement on the surface of that object. Depending on various factors, such as environmental conditions, the surface may either move in sync with the surrounding medium or deform according to its vibration modes. In either case, the motion pattern contains valuable information that can be harnessed to recover the original sound.

The vibrations induced by sound in an object often generate sufficient visual signals to partially reconstruct the sounds that initiated them, using nothing more than high-speed video footage of the object. It is feasible to recover discernible speech and music from the visual data captured even from ordinary objects like a bag of chips.

Building upon this insight, the team at MIT proposed a passive approach for sound recovery using video, which involved visually detecting small vibrations in an object in response to sound and converting these vibrations back into an audio signal, effectively turning commonplace objects into potential microphones. The team recovered sound from objects by recording them with a high-speed video camera, extracting local motion signals, aligning and averaging these signals into a single 1D motion signal, and then denoising it to produce the recovered sound.

Our approach to this existing methodology entails a deep learning approach to supersede the mathematical models used in the original approach. We rebuilt the entire Matlab code from scratch in Python in an attempt to stick as close to the initial proposed methodology as possible. We downloaded the originally recorded data set and even tried attempting to record our own data set at home, both of which were sub-sampled to fit our use case. We normalized the frames by the mean and standard deviation of the training dataset. We also normalized the audio samples to bring the range of values down from (-32k, 32k) to (-1, 1).

We then propose a Siamese-like network to simultaneously generate feature maps for the reference frame and the current frame. We concatenate these feature maps and feed it into a fully connected network which approximates the sound sample for the current frame. We used two different types of CNNs to extract features from the image. The first CNN is ResNet50 with pre-trained weights and the other is a custom CNN designed to extract low level features through a variation of successive 3x3 kernels.

The loss metric for training the network is Sum of Squared Differences, which is essentially Mean Squared Error with the reduction set to sum. The metric for understanding the performance of the model is Root Mean Squared Error, which gives us an idea about the average error across all our samples.

2. Background & Related Work

Davis et al. [1] produced the original functionality of passively recovering sound from a video at MIT in 2014. Their data set and code lay the foundation of our augmentative hypothesis. The team was able to derive a method for sound recovery, which does not require active lighting or additional sensors. They captured three high-speed videos (1kHz-20kHz) of sound hitting objects and extracted local motion signals from the recorded footage, which are then processed to produce the recovered sound. Objects that

can reflect the motion property need to have high damping but also very light to move easily with changes in air pressure. To this end, they used a bag of chips and a potted plant as their objects of study. They compared various camera set-ups in a controlled set-up during the video-capturing phase. The quality of the recovered sounds was evaluated using intelligibility and signal-to-noise ratio (SNR) metrics. The mathematical model was built on the complex steerable pyramid concept [7], which was used to generate scale and orientation invariant feature maps.

The results of the study show that the proposed method can partially recover comprehensible speech and music from visual input alone, demonstrating the potential of everyday objects as visual microphones. The approach was very meticulous as the changes in pixel intensities from one frame to another were reflected within a fraction of a fraction, approximately 1/1000th, of a pixel. Our motivation was to try and increase the efficacy of this approach with deep learning to produce a more amplified version of the input sound wave, or something better than what the proposed method was able to accomplish.

More recently, Long et al [5] created a research application called Side Eye, which revealed how smartphone camera hardware, including rolling shutters and movable lenses, unintentionally modulates ambient sound onto camera images, creating an optical-acoustic side channel for eavesdropping. This side channel, amplified by CMOS rolling shutters and movable lenses, enables eavesdropping without line of sight or objects in the camera's field of view, with experiments demonstrating high accuracies in recognizing spoken digits and speakers.

Since our approach employed convolutional networks, we identified models capable of performing feature extraction across several sequential frames among a plethora of models suited for the task based on AlexNet [3] and LeNet [4]. The ResNet model [2] was trained on a similar task to extract image features with a good degree of accuracy.

3. Methodology

3.1. The Dataset

The data set is obtained from the publicly available supplementary material provided by the MIT team. In each example, a video of a material is taken, and either pre-recorded audio is played through a speaker or a male speaker speaks near the object.

There are five types of audio sources: Chirp, LiveSpeech, Mary, and RecordedSpeech. In the Chirp scenario, a five-second 100Hz to 1000Hz chirp is played through a speaker. In the LiveSpeech scenario, a male speaker speaks near the objects. In the Mary scenario, a MIDI recording of "Mary had a little lamb" is played. In the RecordedSpeech scenario, an audio clip from the TIMIT database is played, containing clips of English speakers of different sexes and dialects. One such experimental set-up to capture this can be seen in Figure 1, where a high-speed camera is placed behind a soundproof glass to record an audio playing over a bag of chips.

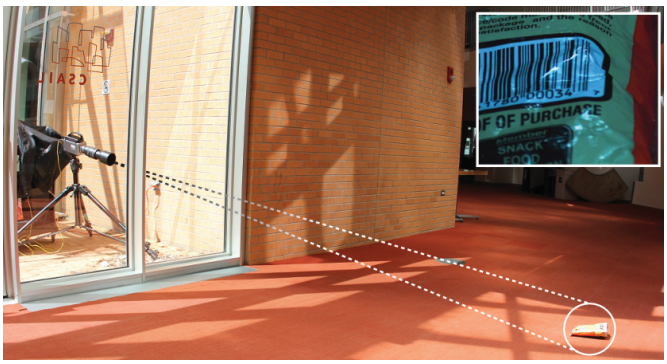

Figure 1. Speech recovered from a 4 kHz video of a bag of chips filmed through soundproof glass. The bag of chips on the floor is lit by natural sunlight only. The camera is positioned outside the room behind thick soundproof glass. A single frame from the recorded video (400 × 480 pixels) is shown in the inset. A speech of "Mary had a little lamb" was spoken by a person near the bag of chips.

Owing to the large file size of high-quality slowed-down videos, we decided to use only some of this data set–the MIDI recording of Mary, the RecordedSpeech scenario and the LiveSpeech scenario.

We also attempted to record our own data using a custom set-up at home, as shown in Figure 2. An aluminum foil exhibiting the suitable properties was placed on a table, with two phone flashlights producing the lighting and a 15-second audio clip of the Mary sample playing from a nearby speaker, with a OnePlus 9 Pro camera capturing the slow-mo video at 480fps and 720p.

Figure 2. Custom set-up of capturing high-speed video on camera using items found at home.

3.2. Mathematical Approach

Figure 3 illustrates a high-level overview of the process from start to end used by the team at MIT. The only difference between their approach and ours is the processing block (B) being replaced by deep learning instead of the mathematical model.

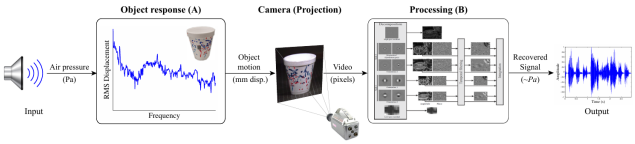

Figure 3. Basic flow of the entire process from capturing the video to regenerating the entire sound.

The frames are extracted using OpenCV. The first one is set as the reference frame. The complex steerable pyramid is applied to this frame, and the output feature map is stored as the reference feature map. A feature map that is scale and orientation invariant is generated by the complex steerable pyramid through the application of a series of filters. The phase information present in these feature maps is considered critical for extracting the sound from the frames.

Specifically, each scale r and orientation θ is a complex image that can be expressed in terms of amplitude A and phase ϕ as

$$A(r, \theta, x, y, t)e^{i\phi(r,\theta,x,y,t)}. \quad (1)$$

The local phases ϕ are computed in this equation and subtracted from the local phase of a reference frame t_0 to compute the phase variations. The reference frame is typically the first frame of the video.

$$\phi_v(r, \theta, x, y, t) = \phi(r, \theta, x, y, t) - \phi(r, \theta, x, y, t_0). \quad (2)$$

The complex steerable pyramid amplitude A gives a measure of texture strength, and so we weigh each local signal by its (squared) amplitude:

$$\Phi_i(r, \theta, t) = A(r, \theta, x, y)^2 \phi_v(r, \theta, x, y, t). \quad (3)$$

For every subsequent frame in the video, the complex steerable pyramid is applied, and its feature map is generated. The phase of this feature map is subtracted from the phase of the reference feature map. The phase difference for all the bands in the complex feature map is then averaged to obtain what is referred to as the global motion signal. The global motion signal is finally,

$$\hat{s}(t) = \Phi_i(r_i, \theta_i, t - t_i), \quad (4)$$

which is then scaled and centered to the range $[-1, 1]$.

This global motion signal across the temporal dimension is correlated to the sound signal in the captured video. A low pass butterworth filter is applied to this signal to remove any high frequency noise. Spectral subtraction was electively performed on the generated sound signal to help remove some of the noise that might be introduced during the extraction process.

Spectrograms are utilized to display the waveform of the signals, showing the visible similarity between the recovered and original waveforms. There is also a striking audio resemblance between the original and recovered sound.

3.3. Deep Learning Approach

In the initial step, we employ OpenCV to extract frames from a video and concatenate them with the appropriate reference frame, typically the first frame of the video. This involves loading the video and extracting its initial frame to serve as the reference. Subsequently, each frame is connected with this reference frame. To maintain simplicity in the model's architecture, we resize frames to 224x224 for Dataset1 and 32x32 for Dataset2. This resizing results in a final size of 448x224 (WxHxC) for Dataset1 and 64x32 (WxHxC) for Dataset2. Each video's frames are then stored in individual folders, named after the respective number of frames in the video.

Regarding the decision to limit the number of frames, our aim was to investigate whether deep neural networks could approximate the mathematical model proposed in [1]. The original paper indicated that all necessary information for extracting the sound signal is present in both the reference frame and the current frame. Hence, we maintained a similar setup to explore the sufficiency of information for sound extraction using neural networks.

During experimentation, it became apparent that our hypothesis had limitations. While the feature maps generated by the complex steerable pyramid provided temporal data through phase, our method captured only the temporal relationship between the first and current frames. This lack of temporal positioning information within the video frames hindered the model's ability to extrapolate sound signatures. Providing additional temporal information could potentially enable models to extract sound even with limited spatial details, particularly in scenarios where sophisticated models like ResNet50 struggled to differentiate frames.

In the next step, we utilize scipy to manipulate audio files, pairing each frame with its corresponding amplitude value. The audio files, represented in 16 bits, have values ranging from -32k to 32k. To standardize these values within a tighter range, we normalize each sample by dividing it by 32767, resulting in values between -1 and 1. The audio is then segmented into equal parts corresponding to the number of frames in the video, serving as the training 'label' for the model. These values are stored in a .csv file for future reference.

For efficient data loading during model training, we cre-

ate a custom class by extending the functionality of the Dataset class in PyTorch. This custom class includes methods to facilitate data fetching without the overhead of running OpenCV concurrently with model training.

Finally, we preprocess the dataset by calculating the mean and standard deviation of the images and normalize them accordingly.

3.4. Experimentation

Our objective is to assess whether a neural network can effectively replicate specific mathematical techniques to approximate the same sound signal. To achieve this, we sought a network capable of extracting essential features for sound sample reconstruction. Figure 4 illustrates the Siamese-like architecture we employed to concurrently run the current and reference frame with the model. We explored three distinct approaches:

1. ResNet50 as a Feature Extractor: We opted to utilize a pretrained ResNet50 model initialized with ImageNet1k (v1) weights due to its reputation as a well-generalized feature extractor. This choice was primarily driven by the minimal variation observed in our training data regarding the types of objects recorded. Our goal was to employ a model capable of generating rich feature maps across a wide spectrum of objects. During training, we kept the ResNet model's layers frozen.

2. Custom Convolutional Network as Feature Extractor: To ensure compatibility with our dataset and eliminate any methodological gaps, we designed a small convolutional network from scratch. Given the mathematical model's reliance on smaller features such as edges, contrast changes, and textures, we opted for 3x3 kernels known for their effectiveness in extracting low-level features [6]. The architecture for this CNN is the primary depiction shown in Figure 4.

3. Feature Pyramid Network-inspired Approach: We attempted to replicate the architecture of a model similar to the feature pyramid network. This involved extracting intermediate feature maps and concatenating them to form a larger feature map. The objective was to investigate whether this additional information would enhance the model's decision-making capabilities.

The concatenated image is disassembled into its constituent parts within the forward function of each network in as efficient a manner as possible. These parts are then forwarded through the respective feature extractors to obtain their corresponding feature maps. Although resembling a siamese network in structure, our architecture involves two branches of the same network processing two distinct inputs.

Subsequently, the feature maps from both branches are concatenated to form a larger feature map, which is then flattened into a one-dimensional array. This array serves as input to a fully connected neural network. Given our output range of -1 to 1, we opted to employ the hyperbolic tangent (tanh) activation function between subsequent layers of the network. The final layer consists of a single perceptron with tanh activation function. Given the relatively shallow nature of these layers, concerns regarding gradient vanishing or explosion are minimized.

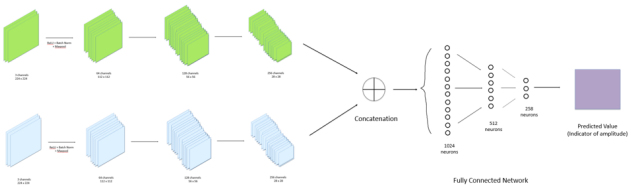

Figure 4. Siamese-like network with the custom CNN architecture represented as the twin sequences of blocks on the left for the current frame and reference frame.

For training the network, we utilized Mean Squared Error as the loss metric with reduction set to sum, effectively representing Sum of Squared Differences. To gauge the model's performance in assigning predictions, we employed Root Mean Squared Error as the evaluation metric. All models were trained using the following hyperparameters: Learning Rate: 0.001, Weight Decay: 0.0001, and Epochs: 10. The optimizer employed was the Adam optimizer.

4. Results

The mathematical model was successfully able to recreate the original sound to a certain degree, as seen on the spectrograms in Figure 5. The output audio file generated incurred significant losses in audio quality with a presence of noise, but we were able to audibly identify the output audio as that of the input.

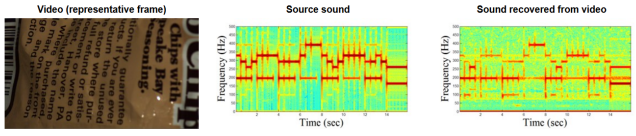

Figure 5. The mathematical model generating an output spectrogram on the right similar to the input spectrogram on the left, when the MIDI audio of Mary was played over a bag of chips.

However, our deep learning approach struggled to capture enough relevant information to recreate the sound in the output. Figure 6 shows the spectrograms for the three approaches attempted.

Table 1 shows a metric comparison of the RMSE and loss values for the individual approaches. The net loss value

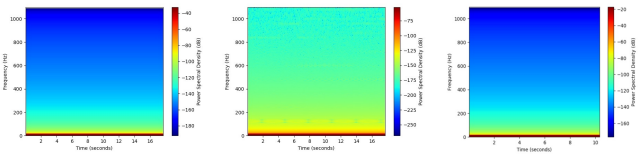

Figure 6. Spectrograms generated by the ResNet model, the custom CNN architecture, and the Feature Pyramid Network respectively.

incurred by both models appear to be very large. The Average RMSE suggests the predicted values and actual values are off by 0.48 and 0.37 in ResNet and Custom CNN respectively. This is not reflected in the output spectrograms because the variations in the actual values are so high that on average the values don't differ by a huge margin.

Table 1. Comparison of loss and RMSE values incurred by the three deep learning models in the approach.

Model	Epochs	Test Size	RMSE	Loss
ResNet	10	38,985 (39%)	0.489	19.143
CNN	10	38,985 (39%)	0.371	18.366
FPN	10	38,985 (39%)	1.070	146.429

It can also be seen that the FPN architecture has an abnormally high loss and RMSE. This is because it works with a larger number of distinct feature maps. Although it does a great job at differentiating more frames, it does not learn well with the given task and parameters.

5. Discussion

The fundamental principle underlying sound extraction hinges on capturing the subtle distinctions induced by sound waves in the recorded object. This necessitates a model capable of detecting even the most minuscule changes between any two given images, a task that poses significant challenges in design.

The ResNet model we employed exhibits remarkable generalization abilities, as evidenced by its consistent generation of identical feature maps for all images depicting the same object. This uniformity became apparent when examining the extracted sound samples, as the model consistently assigned the same sound sample to all frames. While this observation does not definitively confirm identical feature maps, it strongly suggests their similarity is sufficient to deceive the fully connected network.

Our custom-designed model demonstrated improved discernment capabilities compared to the ResNet model, albeit still falling short for this task. The notable difference in discernment capacity may stem from the fact that unlike the ResNet model, our custom model is permitted to learn and adjust its weights across varying frames.

Lastly, the model intended to emulate the feature pyramid network displayed the highest level of frame differentiation, despite underperforming in comparison to the other models. A hybrid architecture involving the FPN's frame differentiation ability could theoretically prove successful.

6. Conclusion

Our exploration into leveraging deep learning techniques to enhance sound extraction from visual data presents intriguing insights and challenges. While the mathematical model based on the complex steerable pyramid methodology showcased promise in partially recovering comprehensible sounds, our deep learning approaches, including ResNet50, a custom CNN architecture, and a Feature Pyramid Network-inspired model, struggled to replicate the same level of fidelity.

Despite significant efforts in data augmentation, architectural design, and training optimization, our deep learning models fell short in capturing the nuanced visual cues necessary for accurate sound reconstruction.

Moving forward, addressing the limitations observed in our deep learning approaches requires a holistic reassessment of model architectures, training methodologies, and data augmentation strategies. Additionally, exploring alternative deep learning paradigms such as recurrent neural networks (RNNs) or attention mechanisms may offer novel avenues for enhancing sound extraction from visual data.

Despite the current limitations, our study lays a foundation for future research endeavors aimed at bridging the gap between visual data and sound extraction through the lens of deep learning. As the field continues to evolve, advancements in both theoretical understanding and practical implementations hold the potential to reaching new frontiers in audiovisual processing and analysis.

7. Auxiliary Instructions

Our code is quite complicated to run and requires a lot of set-up work, which is frustrated even more by the large size of the data set. However, we traced these instructions to be functional to an external user who wishes to run our program. The contents can be found in the transform branch of our GitHub repository: https://github.iu.edu/nbangal/Deeply_Visual_Mic

To run the code replicated from the Visual Microphone paper, follow the below steps: 1. Install all the required dependencies using the requirements.txt file present in the main directory. 2. Run the videoToSound.py file with the required command line arguments.

To run our deep learning implementation, follow the below steps:

1. Install all the required dependencies using the requirements.txt file present in the main directory.
2. Download the videos from the Visual Microphone website and them into a Videos folder. Make sure to rename the videos as Video1, Video2 and so on.
3. In the Audio folder, access the metadata.csv file and add the required information for each video.
4. Switch to the DeepVM folder and firstly run the readWav.py file.
5. Next run the preprocessdata.py file.
6. To extract sound from the video, run the eval.py file.
7. Once the eval.py file has finished running, there should be a tensor_data_s.pt file if the model ran successfully.
8. Finally run the generate_audio.py file to generate a spectrogram.png image and a output1.wav audio file.

An alternative to these steps would be to use the preprocessed dataset available at: https://indiana-my.sharepoint.com/:f:/g/personal/arnanand_iu_edu/EkqHCHo5JsBBlaQ9sJZR9tsB6P2nGbvvMIoCHRv-jFvemg?e=VhSmaf Download and save this folder into the DeepVM folder with the name 'Dataset'. Following this, you should be able to skip steps 2,3,4 and 5.

References

[1] Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham Mysore, Fredo Durand, and William T. Freeman. The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 33(4):79:1–79:10, 2014. 1, 3

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 2

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 2

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[5] Yan Long, Pirouz Naghavi, Blas Kojusner, Kevin Butler, Sara Rampazzi, and Kevin Fu. Side eye: Characterizing the limits of pov acoustic eavesdropping from smartphone cameras with rolling shutters and movable lenses. pages 1857–1874, 05 2023. 2

[6] Dmytro Mishkin, Nikolay Sergievskiy, and Jiri Matas. Systematic evaluation of convolution neural network advances on the imagenet. *Computer Vision and Image Understanding*, 161, 05 2017. 4

[7] E.P. Simoncelli and W.T. Freeman. The steerable pyramid: a flexible architecture for multi-scale derivative computation. In *Proceedings., International Conference on Image Processing*, volume 3, pages 444–447 vol.3, 1995. 2