Speech denoising using lip reading

Ryan Nett, Pierre Lucas, Yongxin Liu

March 22, 2019

Abstract

Our task is to denoise noisy audio using the video of the speaker. Specifically, we use the lips of the speaker to help a denoising autoencoder perform better than it would on audio alone. Both lip reading and audio denoising have been done in the past, but they haven't been combined except in one paper, which is similar to our starting point. More broadly, the use of closely related types of input to help classify or denoise some main input can be useful in other areas, for example, classifying pictures of animals based on the picture, and the sound that animal makes. We get good results, especially when adding a discriminator, and show that using the video contributes significantly to the resulting quality. We even show that we can successfully denoise an audio signal with 0.5 standard deviations gaussian noise added (the audio signals are normalized to 1).

1. Introduction

In a speech, the voice contains only a small part of the message that is being transmitted. Understanding what people say relies on other features, such as the lips. More generally, the facial expressions and the body gestures help to get the nature of the speech.

While speaking, lips don't contain more information than textual information. However, combined with the voice, they help to better understand what is being said. An easy way to verify this is to see how it is disturbing to watch a video where the audio and the video are delayed.

Denoising and enhancing speech is an open problem that can serve many applications. A main daily life application of it would be to denoise the audio from phone calls by using pictures from profile lips. Indeed, noises coming from the street, subways or other people make the understanding sometimes hard. Another application would be to improve the accuracy of the lipreading, which can especially serve robotics purposes. Finally, we could classify animals based on their sound and pictures to increase accuracy.

The challenges and the opportunities offered by audio denoising motivated us to develop an innovative model presented in this paper. Our model has been designed as two branches autoencoder that uses a discriminator for training. The results in *Figure 1* and presented later show the success and the ability of our model to properly denoise a person speaking.

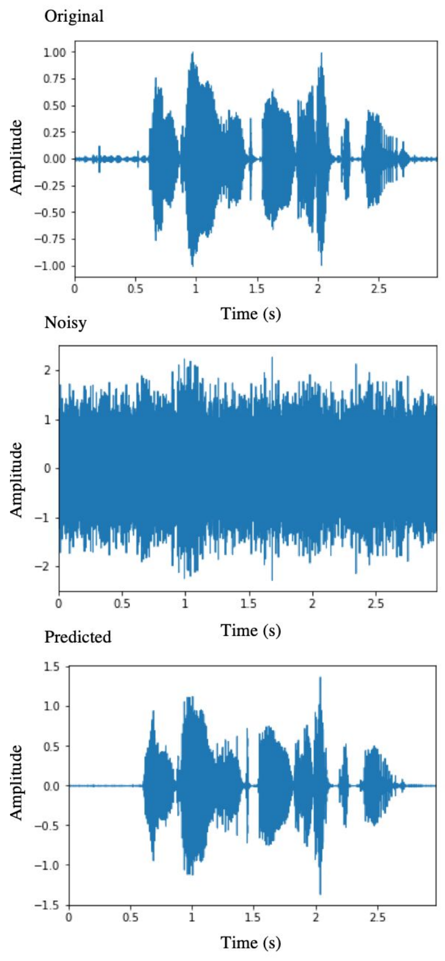

Figure 1. Time-domain graphs of noisy audio (middle), its clean version (top) and its denoised version (bottom)

1.1 Related work

Many works are currently being done on subjects related to audio-visual speech recognition.

Most of the work we found was done on lip reading, which is only of limited use to us. There were a few audio encoding/decoding methods, but not very many, and most were of dubious quality.

To help our development, we used sample code from two papers, one a Lipreading [1], the other on Visual Speech Enhancement [2].

1.1.1 Lipreading

Lipreading is an important field of research because it can serve many purposes, for instance in robotics as well as in medicine.

Decoding text from the lip movements is not our goal, but in order to achieve this task, a common work has to be done: catch the features from the lips. *LipNet* [1] is a model developed in 2016 that shows the best current performances, achieving 95.2% accuracy in sentence-level.

1.1.2 Audio Generation

Recent works have been done to generate speeches. *WaveNet* [3], a deep generative model of raw audio waveforms, is one of the latest models with the best performance on this field. The success of their model is the quality of the voices generated, which sound natural and close to reality.

The quality of the generated voice matters to us since our goal is to denoise speeches. *Wavenet* paper has not been used in our current model but might be used for former implementations.

1.1.3 Audio Denoising

Previous work has been done on audio denoising. *Visual Speech Enhancement* paper [2] share the same ambition of denoising speeches using an audio-visual autoencoder. Our model is different from theirs due to two main modifications. On one hand, a discriminator has been implemented in our model, improving the accuracy with a high noise level. On the other hand, the video decoder takes a whole video file as input instead of single frames.

Yet, we did make any comparison between these two models. Nevertheless, our model should theoretically show better performance.

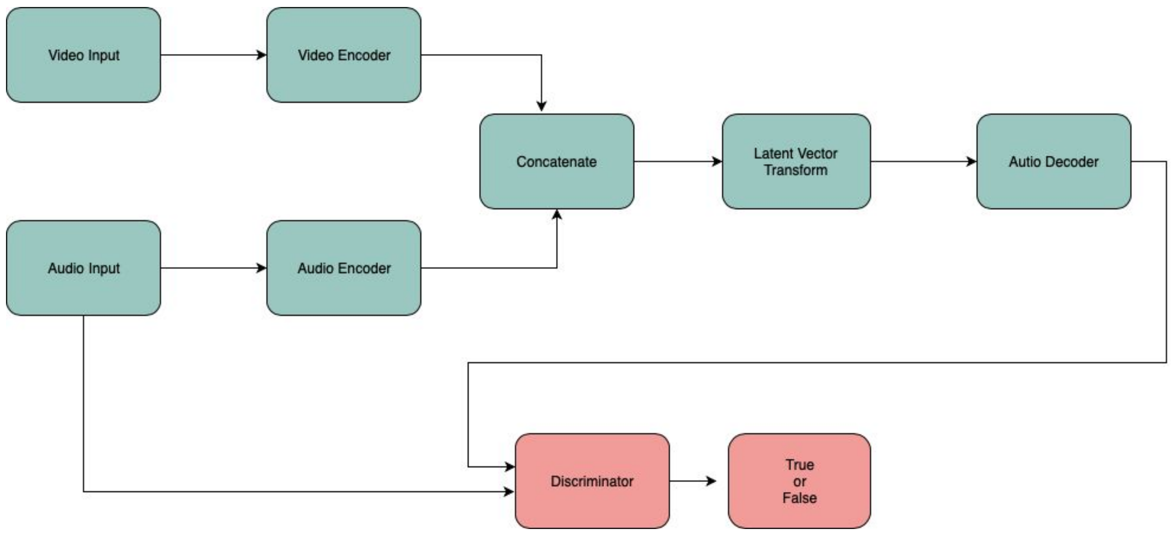

Figure 2. Flow chart of the whole architecture

2. Architecture

The architecture of our model has 5 parts: the audio encoder, the video encoder, the latent vector transforms, the audio decoder, and the discriminator (shown in *Figure 2*).

We extracted the audio from the video dataset and added the different level of Gaussian noise on it. Meanwhile, the lip movement will be cropped from each frame of video. These two types of data will be fed into audio encoder and video encoder respectively. The output latent vectors from two encoders will be concatenated together and then we have implemented latent transforms on it, which will be discussed in detail soon. To reconstruct audio from the latent vector, we fed it into our audio decoder. Finally, we use the discriminator to force the reconstructed audio close to the original clean audio.

The whole diagram is shown in *Figure 2* and the autoencoder parts are shown in *Figure 3*.

2.1 Audio Encoder

The audio encoder is made up of 6 2D-convolution layers, followed by a reshaping. Each convolution layer was followed by a BatchNormalization layer and then a LeakyReLU activation.

The basis for this design came from the *Visual Speech Enhancement* paper.[2]

Filters	Kernel Size	Strides
64	(5, 5)	(2, 2)
64	(4, 4)	(1, 1)
128	(4, 4)	(2, 2)
128	(2, 2)	(2, 1)
128	(2, 2)	(2, 1)
128	(2, 2)	(2, 1)

2.2 Video Encoder

The video encoder is made up of 3 3D-convolutions and poolings, followed by a bidirectional GRU layer with 256 units. The basis for this design came from the *LipNet: End-to-End Sentence-level Lipreading* paper (the 2nd model) [1].

Filters	Kernel Size	Strides
32	(3, 5, 5)	(1, 2, 2)
64	(3, 5, 5)	(1, 1, 1)
92	(3, 3, 3)	(1, 1, 1)

Each convolution layer was preceded by a zero padding layer, and followed by a batch normalization, ReLU activation, and spatial dropout, and then a MaxPooling3D layer with a pool size and stride of (1, 2, 2) (on every layer).

2.3 Latent Transformations

The encoded audio and video are reshaped to have 75-time steps (one for each frame, although the audio is not precisely that), and then flattened down to a shape of (75, 896). It then goes through a single dense layer with 384 units, with batch normalization and a LeakyReLU activation. It is then reshaped to (3, 75, 128) for the audio decoder.

2.4 Audio Decoder

The audio decoder is the inverse of the audio encoder, using Conv2DTranspose layers with the same filters, kernel size, and strides, just reversed (e.g. the first layer has 128 filters and a kernel size of (2, 2)). This is followed by a Conv2DTranspose with 1 filter and kernel size and stride equal to (1, 1), a cropping layer, and reshaping.This outputs a (80, 299) spectrogram, which is the same as the audio input.

To convert the spectrogram back into an audio signal, you can use librosa's methods, but we also implemented a tensorflow inverse STFT that is compatible with librosa (the provided tensorflow implementation isn't). This allows us to use it in a lambda layer and reconstruct the audio as part of the model, which makes testing slightly easier. It's not worth doing just for testing, but we created it when experimenting with a residual network design.

2.5 Discriminator

The discriminator is just two convolutional layers with kernel sizes of 4 and strides of 2. The first has 32 filters, the second 63. This is flattened and fed into a dense layer with 1 unit and sigmoid activation to give us our true/false output.

2.6 Progression

When first creating our model, we started with the audio encoder and decoder from *Visual Speech Enhancement* [2]. While *Visual Speech Enhancement* uses a video encoder, their setup split the video up into single frames, which ours does not, so we weren't able to use it. We then added the video encoder from *LipNet: End-to-End Sentence-level Lipreading* [1]. To further increase the audio quality of our results, we added a basic convolutional discriminator. As shown in our **Experiment and Evaluation** section, adding video and the discriminator increased the quality of our results significantly. We also tested using a residual network with addition or multiplication, but neither worked well.

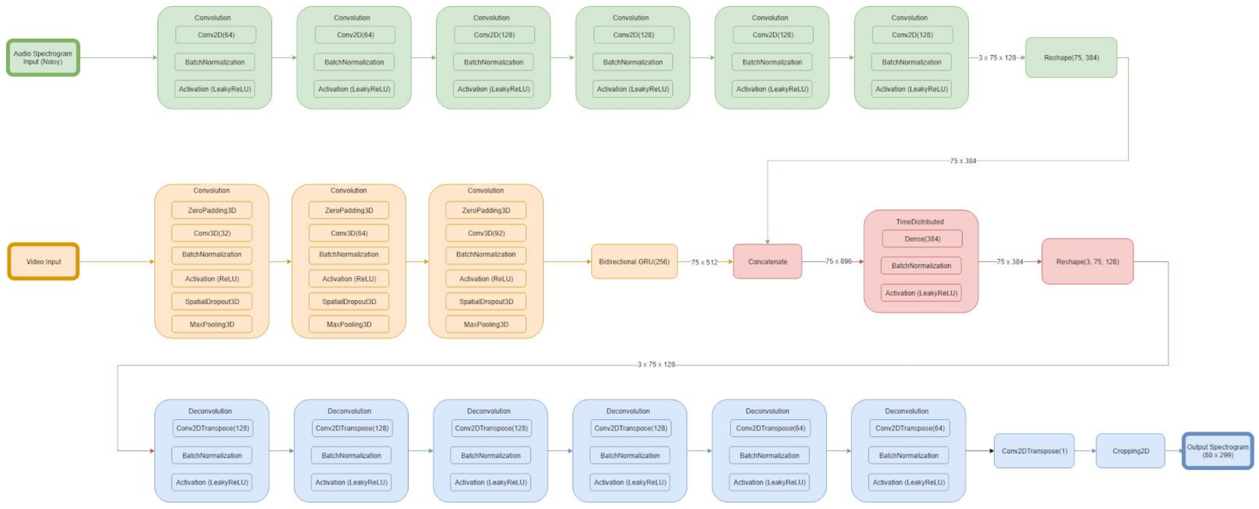

Figure 3. Autoencoder parts

3. Training

3.1 Dataset

The GRID dataset has been used to train and to experiment with our model. GRID is an audio-visual sentence corpus containing video and audio recordings of 1000 sentences spoken by 34 different talkers: 18 men and 16 women. We only used 3, two to train, and one to test how our model performed on an unseen speaker.

Designed for research, the sentences spoken are constructed as sextets: [verb, color, preposition, letter, digit, adverb]. For instance, “put white in G2 soon”.

This dataset has been used for many research and also in challenges, such as the First Speech Separation Challenge.

More complex and large audio-visual dataset exist, such as *Lip Reading in the Wild Datasets* (LRW, LRS2, LRS3), containing sentences spoken by speakers from the BBC. This dataset could be used to improve the results of our model and to make it more robust to all kinds of situations.

3.2 Data preprocessing

The architecture of our neural network requires to preprocess the data. The three followings inputs are needed:

- A spectrogram of noisy audio to feed the audio-autoencoder;
- A spectrogram of the clean version of the noisy audio to feed the discriminator;
- A set of cropped mouths frames from the spoken audio to feed the video-audio encoder.

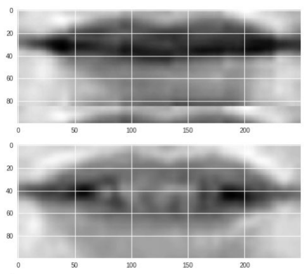

Figure 4. Cropped mouths sampled from a video

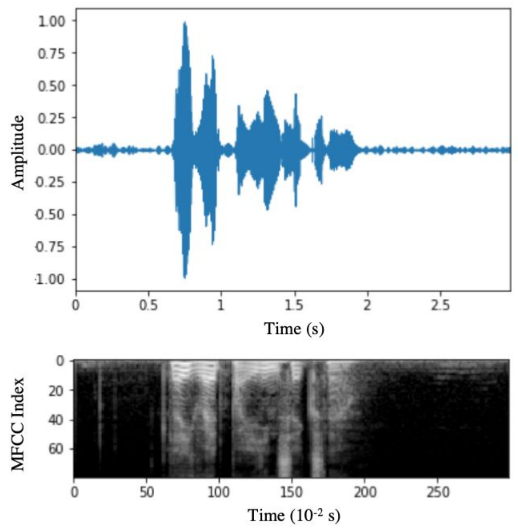

Figure 5. Time-domain graph of a sentence (top) and its associate spectrogram (bottom) extracted from our dataset.

3.2.1 Noise

The dataset used offers only clean audio. A Gaussian noise, with an amplitude going from 0.05 to 0.5 has been added to each audio.

3.2.2 Spectrogram

The mel frequency cepstrum has been used as a representation of the audio signal, which is a particular representation of the short term power spectrum of the audio. Seen in most audio processing papers, this choice has been mainly retained because the MFC can easily be processed with convolutions. To implement it in our model, mel filter banks have been applied over the Short Term Fourier Transform of the signal. The results are shown in *Figure 5*. This kind of graph has been input to the audio-encoder.

The output of the audio-decoder is consequently a spectrogram. To actually re-create the audio signal from the output, our model also needs the phase of the audio. This reconstruction can be done outside of the model or inside of it.

3.2.3 Mouths

The mouths have been extracted from each frame of the video using an existing method based on facial landmark detection [4].

For now, our model is designed to work with a video containing only 75 frames, and the extracted mouths are resized to a specific size to be fed into the video-encoder.

3.3 Training

We trained our model on 200 videos out of 1000, in order to simplify the training's time complexity. Split into batches of 20, the model has been trained using 10 steps per epoch, which lets the model see all inputs on each epoch.

For each batch, we train the autoencoder to match the input, the discriminator to correctly discriminate, and then the autoencoder to fool the discriminator. We can vary the number of times each is trained, but early on found 3, 5, 1 (3 autoencoder repetitions, 5 discriminator repetitions, and 1 autoencoder to fool the discriminator repetition) to be the best.

There are several options of the optimizer. Comparing with the experiment results, we found that Adam optimizer with MSE loss is the best choice (see the experiments).

We ran most of our tests on 10 or 20 epochs because each epoch takes about a minute to run. Our results held as the epochs increased.

For our final test, we used 50 epochs. At this point, the loss has started to plateau, although it is still decreasing.

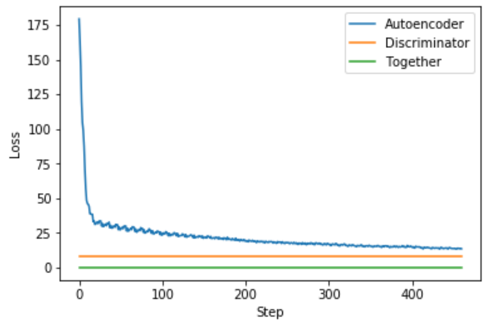

Figure 6. Loss for 50 epochs and 0.5 noise level.

We notice that each epoch has 10 steps, and the graph starts at through the first epoch as 0 (to avoid the high MSE hiding everything else).

4. Experiment and Evaluation

To evaluate our model, analysis on audio with the same Gaussian noise as the trained model have been done.

4.1 Evaluation criteria

Although our denoising results are applaudable , we still can not use them as the evaluation criteria to determine the parameters, such as optimizer and training epochs etc, we take Mean Squared Error(MSE) and Signal-to-noise Ratio(SNR) as criteria to do the evaluation.

- Mean Squared Error(MSE)

  MSE is used to calculate the mean squared error between the output denoising audio and original clean version. The lower of this value, the better of the result.

- Signal-to-noise Ratio(SNR)

  The signal-to-noise ratio computes a level of signal power to a level of noise power. In contrast to MSE, higher numbers generally mean a better result, since there is more useful signal(audio) than the unwanted noise.

4.2 Experimentation

Our experiments have been mainly separated into two parts: the parameters optimization and noise level tolerance.

4.2.1 Parameter Optimization

In the parameters optimization part. We set the training epoch to 20 for all the experiments. We used Adam with a learning rate of 0.005, and SGD with a learning rate of 0.02 and momentum of 0.9. The loss functions are only for the autoencoder, the discriminator always uses binary cross entropy.

The result is shown in *Table 1*.

Optimizer	Loss	MSE	SNR
SGD	MAE	0.00765	0.002559
SGD	MSE	N/A	N/A
Adam	MAE	0.1162	0.001086
Adam	MSE	0.0048306	0.0036631

Table 1. Evaluation of different combinaisons of optimizer and loss functions on the ficombinationsnal model

The experiment result of SGD with MSE is so bad we get infinite values in the audio output.

The lowest MSE and highest SNR of the combination of Adam and MSE will be our choice for the optimizer and loss function, and we will keep these parameters across the following experiment section.

4.2.2 Noise Level Tolerance

Once deciding the most suitable parameters to construct our denoising architecture, it’s time to exam the performance of this product.

In this section, not only we want to exam the upper limit noise level tolerance, but also to test the denoising performance without using discriminator and/or without using the video. The experiment and the results are shown in *Table 2*. We use this to show that the discriminator and the video input are both necessary to successfully denoise audio when the input is very noisy.

Surprisingly, using just the audio works well for low noise levels, but this may also be because the model has many fewer parameters and thus will train faster. As we increase the noise level, the video and discriminator become more and more important.

We also ran a test where we trained the model with a noise level of 0.2 and then tested it with a noise level of 0.4, with only 10 epochs. The several tests on different noise level are shown in *Table 3.*

Epoch	Noise Level		MSE	SNR
10	0.05	Everything	0.00616	0.00228
		No Video	0.02702	0.00209
		No Discrim.	1.08114	0.00207
		Just Audio	0.00551	0.00262
	0.2	Everything	0.00442	0.00170
		No Video	0.00946	0.00421
		No Discrim.	1.08805	0.00207
		Just Audio	1.09111	0.00205
20	0.2	Everything	0.00359	0.00
		No Video	0.00518	0.00045
		No Discrim.	1.09035	0.00200
		Just Audio	1.14478	0.00213
	0.5	Everything	0.00280	0.00074
		No Video	0.00496	0.00104
		No Discrim.	1.06589	0.00203
		Just Audio	1.09894	0.00207

Table 2. Evaluation of different architectures

While there is a noticeable drop in the quality of the output, it still successfully denoises the audio, and the resulting audio actually sounds like someone talking. This still true even when trained with a high noise level, and then given less noisy audio, although the difference isn’t very big. The results are in *Table 3*.

Training Noise Level	Testing Noise Level	MSE	SNR
0.2	0.2	0.00554	0.00199
0.4	0.4	0.00413	0.00152
0.2	0.4	0.05865	0.00170
0.4	0.2	0.01446	0.00141

Table 3. Testing and training on different noise levels.

4.3 Results

To test how good our best model is, we ran our model (with everything) for 50 epochs with 0.5 noise level. This took about 50 minutes. The results are shown in *Table 4* and *Figure 7*.

The audio is here.

Epoch	Noise Level	MSE	SNR
50	0.5	0.00558	0.00106

Table 4. MSE and SNR results of the final model trained on 50 epochs and a noise level of 0.5

4.3.1 Quantitative results

The model has run for around one hour with 50 epochs and 0.5 noise level. To our surprise, the performance of this architecture is pretty good even when the noise level has been increased to 0.5. Even though the output audio sounds a little bit distorted, we can still figure out what the speaker is saying and the more important thing is that the noise has been completely removed.

4.3.2 Qualitative results

By listening to the predicted audio, we can hear that the voice sometimes turned metallic, especially for a low level of noise.

This can be explained by the fact that our model outperforms, denoising not only the Gaussian noise but also the voice itself.

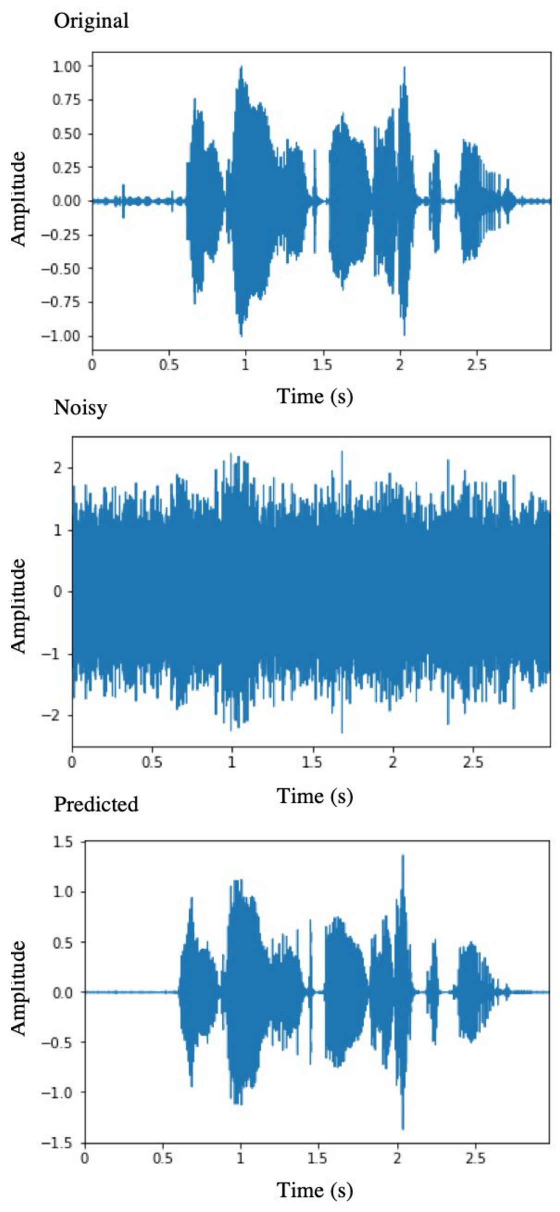

Figure 7. Time-domain graphs of noisy audio with noise level of 0.5 (middle), its clean version (top) and its denoised version (bottom)

It is also in part because our audio generation generates the spectrogram instead of real audio, and small errors in the deconvolution layers will cause distortion in the audio. A better audio decoder would likely fix this.

5. Conclusion

The results we obtained along our research are surprisingly good and very promising for future works on the subject.

We have already thought about many leads to make our model more robust and able to be used for many applications.

First of all, we intend to change the dataset we used to a dataset more like noise in the wild, more complex and closer to real life situations. For specific applications, such as denoising a phone call done with earphones, a dataset of profile lips speaking would be needed.

Following the same idea, we would like to change the type of noise added to the audio, more varied and closer to reality. For instance, noise generated by subways, streets, people, nature, etc.

Another way to improve our model would be to add a better audio generator/decoder, such as a WaveNet style architecture.

In addition, LSTMs or another type of recurrent layer seems like a good fit for processing the latent vector (it is split into time steps). We could not get them to produce good results, but none of us have any experience with RNNs either, so it isn’t very conclusive.

We could also experiment with using a different type of discriminator, such as one from WGAN or LSGAN.

Finally, we could improve our model that it can accept any kind of inputs, no matter the size of the video and its quality. This would mostly be done in pre-processing.

6. Citations

[1] Assael, Y. M., Shillingford, B., Whiteson, S., & De Freitas, N. (2016). Lipnet: End-to-end sentence-level lipreading. *arXiv preprint arXiv:1611.01599.*

[2] Gabbay, A., Shamir, A., & Peleg, S. (2017). Visual speech enhancement. *arXiv preprint arXiv:1711.08789.*

[3] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499.*

[4] Rosebrock, Adrian (2017, April). Detect eyes, nose, lips, and jaw with dlib, OpenCV, and Python. Retrieved from *https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python*

Appendix 1: Notebook

Open in Colab

https://colab.research.google.com/gist/rnett/dd4194417e3fb3e2bfaf051735c8cfa3/project.ipynb

https://gist.github.com/rnett/dd4194417e3fb3e2bfaf051735c8cfa3

Note that the code is not set up to run any particular test. It should be runnable with a little work, but is intended for inspection.