LIPS: Learning Inaudible Processed Speech

John C. Merfeld
Department of Computer Science
Boston University
Boston, MA 02215
jcmerf@bu.edu

Allison Mann
Department of Computer Science
Boston University
Boston, MA 02215
ainezm@bu.edu

Abstract

While translating speech to text and back again has become a household technology over the past decade, many situations do not lend themselves to such audio translation. Automated lip reading provides an alternate avenue for translating speech to text. In this paper, we introduce 1) a video-processing pipeline that prepares raw video and text data for predictive modeling, 2) a deep-learning-based word classifier model on the processed data, and 3) evaluation of our model on several different vocabularies. All of this adds context to the still-burgeoning field of automated lip reading, suggesting questions for future study.

1 Introduction

Lip reading is the process of identifying spoken words purely from lip movement without any audio data. It is notoriously difficult, even for humans, but it plays a crucial role in human communication. This is highlighted by the McGurk Effect, where one phoneme's audio is transplanted on top of another phoneme's video, causing a third phoneme to be perceived [1]. Hearing-impaired people especially rely on this method to understand speech without audio. Even the most skilled lip readers are able to achieve an accuracy of around 30% when identifying isolated words. They rely on context, body cues, and hand gestures to fill in the gaps [2]. Commonly, humans also use lip reading in cases where discreet communication is necessary or in noisy environments where audio signal is not available.

With recent machine learning advances, attention has been drawn to this problem due to its many practical applications. Lip reading has promising applications in assisting hearing-impaired people. With automated lip reading, video annotation and translation become possible. Lip reading also has applications in surveillance and analysis of forensic video.

In this paper, we propose *Learning Inaudible Processed Speech* (LIPS), a method for lip reading via deep learning. Our method treats lip reading as a word classification problem. After extensive video processing, we use convolutional neural networks to exploit spatio-temporal features from input video data.

2 Related work

In recent years, there have been many advances in lip reading. Preceding work can be organized into two groups: classification systems and sequence prediction systems.

2.1 Classification Systems

The majority of work done on this problem has been in the development of classification systems. The classification system approach performs classification over words or phonemes rather than full

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

sentence sequence prediction. Ninomiya et.al. used Hidden Markov Models for classification of words or phonemes [3]. Sui et.al. approach the lip reading problem with Boltzmann Machines [4] and Chung et.al. propose a deep neural network framework for word classification. Their method can effectively learn and recognize hundreds of words [5]. Thabet et. al. propose novel video processing methods and show experimental results over multiple classifiers including Multi-Layer Perceptron, Naive Bayes, Support Vector Machine and Logistic Regression. Their results were very impressive with accuracy of 65% over 5 distinct words [6].

2.2 Sequence Prediction Systems

Comparatively, much less work has been done on sequence prediction systems for lip reading. These systems aim to translate end-to-end sentences from visual data rather than single word classification. Notable work in this category was done by Graves et.al who use deep learning with recurrent networks for end-to-end speech recognition [7]. Their work, while not strictly lip reading, influenced following sequence prediction for lip reading such as the work of Assael et.al. with their system LipNet. LipNet uses convolutional neural networks to provide a distribution over sequences of phonemes and demonstrates high accuracy, especially with respect to distringuishing phonemes [8].

3 Method

Transforming video footage into a lip-reading prediction requires a significant processing pipeline. In this section, we discuss the training dataset, how it was prepared for modeling, and the overall architecture of our model.

3.1 Data

The Oxford-BBC Lip Reading Sentences 2 (LRS2) Dataset was selecting for training. Details of its creation can be found in [9]. A brief summary of the data will follow.

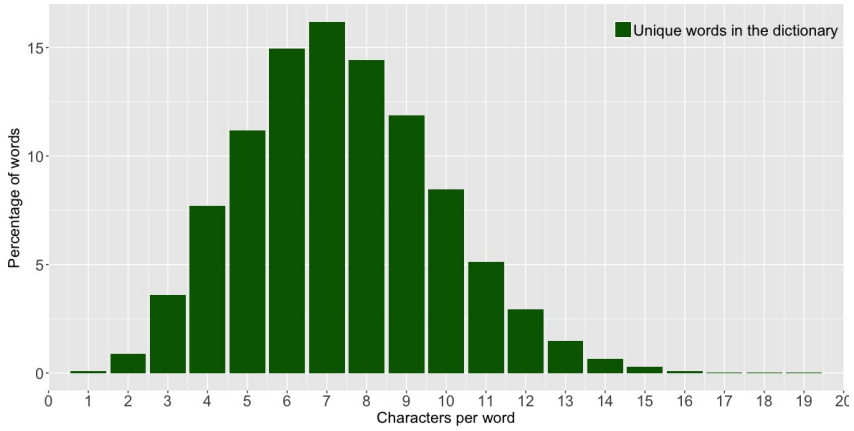

Figure 1: A dictionary D containing an entry for each unique word in the dataset would have a modal character count of 7. Longer words tend to have more distinct visemes than shorter ones.

The atomic units of raw data are video files, each around ten seconds long. The video clips come from BBC news and talk shows, but each frame's viewing window is shrunk to track the face of an individual speaker. A single speaker may span multiple clips, but LRS2 was designed specifically to include as many different faces as possible. Accompanying each video is a like-named metadata text file; this file contains an ordered list of words spoken in the clip along with timestamps indicating when the speaker starts and stops saying each word. These times are precise to the hundredth of a second.

We did not use the entirety of LRS2; we limited our study to the "pre-train" segment because the "main" segment's text files do not include word timestamps. Still, the model's potential training examples encompassed over 2 million utterances of 41,427 unique words; 47GB of data in all.

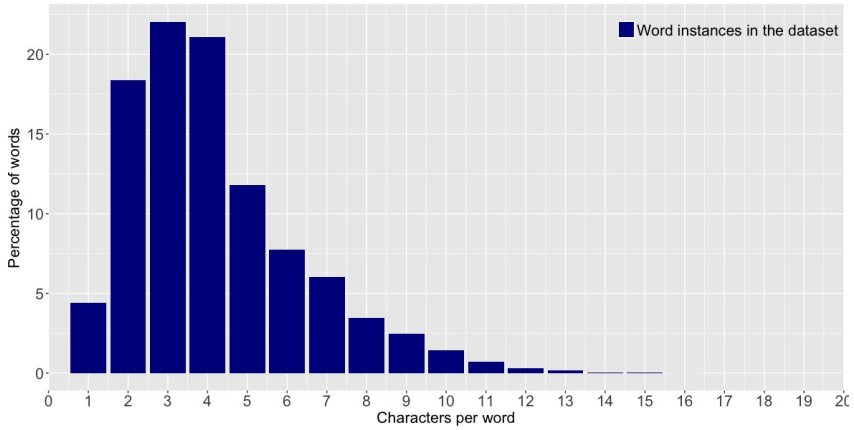

Figure 2: Although there are more unique words with a higher character count, shorter ones appear much more often in the data.

3.2 Feature Processing

Consider an example pipeline whose output is a model that, given a video clip, predicts whether the speaker in that clip is saying the word "yes" or the word "no." So as to act within the realm of computational feasibility, we assign a parameter N_{files} as an approximator for the size of this model's training set.

First, we prepare a dictionary mapping strings to unique integers. We formally define this as a set $D = \{\text{'YES'} : 0, \text{'NO'} : 1\}$ Next, N_{files} text files are parsed word by word. If a word in the file is 'YES' or 'NO,' it is recorded, along with its beginning and ending timestamps.

Next, the videos corresponding to those N_{files} text files are loaded and converted to greyscale. If a clip contains a word belonging to D, every frame between that word's two timestamps is recorded into its own 2D array of integers between 0 and 255. Thankfully, the videos in LRS2 are already of a uniform height and width, so no resizing is necessary. Together, this list of 2D arrays composes a 3D object we call a **word clip**. Note that at this stage, audio information is no longer present in the data representing the video.

Of course, each word instance may be composed of a different number of frames, depending on how long the speaker took to say the word. The model, however, will expect every input to be of the same dimension. Thus, we find the median length in frames of every word clip already created. If a word clip's frame count is higher than the median, it is evenly downsampled. Word clips that are too short have some of their frames repeated until they reach the median frame count. At the end of this preliminary process, then, we have a list of data objects pairing a word with a 3D word clip object of uniform size.

Until now, we have dealt with word clips of the speaker's entire face. However, the model's task is far simpler if it only sees the speaker's mouth. We use dlib's face recognition library to detect facial fearues on each frame of each video. Dlib is a C++ library that contains a toolkit of machine learning algorithms [10]. We primarilty used "Shape Predictor 68 Face Landmarks" which extracts coordinates of 68 landmarks of the face. These landmarks represent points on the mouth, nose, eyes and so forth. We take a standard size bounding box centered around the mouth to crop each frame. This results in all frames being normalized to a standard size without affecting the aspect ratio. This method is reliable becuase our dataset contains video data centered around the face of the subject, and the videos were normalized in a previous step. This ensures that the face of the subject is roughly the same size in each video.

Finally, we concatenate the frames of a video to make a long image. In the end, if each word is captured in n cropped frames with dimension h x w then the resulting feature vector will have dimension h x nw.

Thus, at the end of the pipeline, the model is prepared to be trained. To generalize this pipeline, we simply redefine D to include whatever words the model needs to distinguish between. To train

Figure 3: In the above example, both clips must be resized to 6 frames. The first clip has frames removed at regular intervals, while the second clip has frames duplicated at regular intervals. The result is an input object of uniform size.

models quickly, we can reduce N_{files} and temporarily store processed video data; to validate them, we can increase N_{files} to include the entire training set.

3.3 Model

Once our video data has been processed, we create label vectors via one-hot encoding. If the total dictionary has k words, then we create a vector of length k where each entry is 0 except at one location where it is 1. Each cell represents a word in the dictionary, and a 1 in that cell signifies a label of the corresponding word.

The model we use is a convolutional neural network (CNN) because it exploits spatiotemporal features in the input image by performing stacked convolutions over the image. Our model has two convolutional layers each with reLU activation followed by max pooling. Finally, we have a global max pooling hidden layer to flatten the features and an output layer of size k. We take the softmax of the output to obtain probabilities of each word, and we use categorical cross entropy to train the model. For testing purposes, we assign the predicted word to be the word associated with the max entry of the output.

4 Evaluation

4.1 Experimental setup

4.2 Results

4.3 Tuning hyperparameters

5 Conclusion

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

[1] McGurk H. & MacDonald J. (1976) Hearing lips and seeing voices. *Nature* **264** (5588): 746–8.

[2] Altieri, N. A. & Pisoni, D. B. & Townsend, J. T. (2011). Some normative data on lip-reading skills (L).*The Journal of the Acoustical Society of America* **130** (1), 1–4.

[3] Ninomiya, N. Kitaok &, S. Tamura & Y. Iribe & K. Takeda (2015) Integration of deep bottleneck features for audio-visual speech recognition. *In International Speech Communication Association*

[4] C. Sui & M. Bennamoun & R. Togneri (2015) Listening with Your Eyes: Towards a Practical Visual Speech Recognition System Using Deep Boltzmann Machines. *2015 IEEE International Conference on Computer Vision (ICCV)* 154-162.

[5] J. S. Chung & A. Zisserman (2016) Lip reading in the wild. *Asian Conference on Computer Vision* 87-103.

[6] Z. Thabet & A. Nabih, K. Azmi & Y. Samy & G. Khoriba and M. Elshehaly (2018) Lipreading using a comparative machine learning approach. *2018 First International Workshop on Deep and Representation Learning (IWDRL)* 19-25.

[7] Graves & N. Jaitly (2014) Towards end-to-end speech recognition with recurrent neural networks. *In International Conference on Machine Learning* 1764–1772

[8] Assael, Y.M & Shillingford, B. & Whiteson, S. & Freitas, N.D. (2016). LipNet: Sentence-level Lipreading. *CoRR*

[9] T. Afouras & J. S. Chung & A. Senior & O. Vinyals & A. Zisserman (2018). Deep Audio-Visual Speech Recognition. *IEEE transactions on pattern analysis and machine intelligence.*

[10] R.-L. Hsu & M. Abdel-Mottaleb & A. K. Jain (2002) Face detection in color images. *IEEE transactions on pattern analysis and machine intelligence* **24** (5), 696-706.