GuNiLeo: An Attempt To Perform Lip Reading From Videos With STCNN

Beray Nil Atabey (*2045576*) Leonardo Biason (*2045751*) Günak Yüzak (*2048950*)

***Abstract*—Lip reading is a task that can have various usages as an accessibility feature, but it's also very complex to design: it requires a machine to be able to differentiate between the various words said by a speaker, and also to predict what the speaker said whenever words aren't spelled with a precise motion of the lips. With this paper, we propose a model based on a Spatio-Temporal CNN, capable of reading the words said by a speaker from a video clip of maximum 75 frames.**

I. Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

II. Implementation

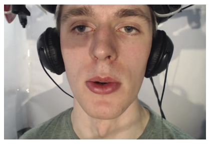
In order to create a lip reading model, the following steps have been undertaken:

1) creation and modeling of the dataset;
2) creation of the model;
3) training of the model;
4) evaluation of the model.

III. Creation and Modeling of the Dataset

The project adopted an already existing dataset from the University of Sheffield: such dataset is called LOMBARD GRID [1].

Fig. 1. Example of frame (on the left) and label (on the right)



```
{
    "video_name": [
        {
            "duration": 0.01,
            "offset": 0,
            "phone": "SIL_S"
        }
    ]
}
```

Frame Label

A. Structure of the Dataset

The dataset comprehends two fundamental parts: the data part and the labels. The data consists in a set of 5000+ videos, where each video records frontally a person saying a specific sentence. Each sentence is then transcribed in a `json` file, which has the following format:

- `video_name`: a list containing all the phonemes and the timing of said phonemes of a clip;
  - `duration`: the duration in seconds of the phoneme;
  - `offset`: the beginning of the phoneme, with respect to the beginning of the video;
  - `phone`: an encoding of the phoneme.

An example of data-label pair can be observed in Figure 1, where a frame of a sample video has been reported, alongside an example of possible label. The label can be interpreted as follows: there is a phoneme, the silent "s", which starts at $t = 0$ and lasts for $0{,}01s$.

B. Pre-processing of the Dataset

Since we want to read the lips, the only area that concerns us is the mouth and the close surroundings. The rest of the video can be cropped to avoid unnecessary memory-wise and computational power-wise waste. In order to locate the mouth, the `dlib` [2] library's face detection and landmark detection algorithm has been employed. The face detection is implemented in `dlib` itself, however for the landmark an external model had to be used. We now

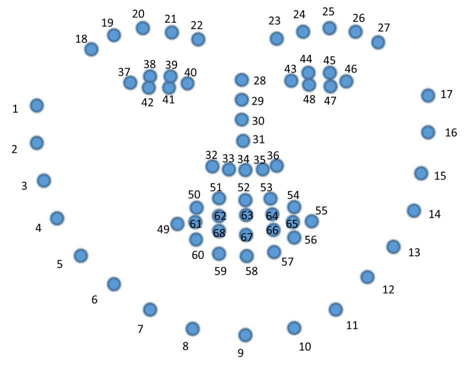


Fig. 2. Face landmarking points of the `dlib` library

proceed to describe the undertaken steps for the pre-processing of the data:

1) First, the face detection algorithm is ran on the video in order to find the coordinates of the face;
2) Such coordinates are given as input into the landmark model, which determines the important landmarks of the face via a sequence of 68 points. Points from 49 to 68 determine where the mouth is (such points are shown in Figure 2);
3) From these points, it's possible to find the points that are on the left,right, up and down most of the mouth, which correspond to the maximum coordinates that a mouth can take in a specific video;
4) From these coordinates a margin distance is added, and after that the video gets cropped around the mouth. Clearly, every person's mouth size is different from one another, so the resize procedure reshapes the video to a fixed size of 150×100 pixels;
5) Finally, the resized video is then grayscaled and normalized for a better performance of the model.

IV. Architecture of the Model

The model used for this project takes large advantage of the Spatio-Temporal Convolutional Neural Network concept [3] (from now on referred to as STCNN). The choice of using such type of neural network is because of its capabilities of retaining information in the long short-term, which is very helpful when it comes to analysing multiple videos and making predictions on the long run.

Here follows the detailed architecture of our model, with all its layers:

- STCNN and 3D Max Pooling $\times 3$;
- Bi-directional GRU unit $\times 2$;
- CTC loss function;
- Softmax activation function.

A visualization of our model can be found in Figure 3. As input, the model takes a clip of shape (100, 150, 75, 1), and it returns a tensor of shape (75, 37). The possible output, once encoded, is a list composed by letters from a-z, numbers from 0-9 or blank spaces "␣".

The model supports 75 frames videos, which, in this case, it corresponds to a video of 3 seconds with a constant frame rate of $25f$. The videos that had more than 75 frames were discarded from the dataset, and the ones with less then 75 frames were extended with 0 matrices (so blank frames).

TABLE I
Values of the loss function $\mathcal{L}$ over the epochs

Epoch	**Value of $\mathcal{L}$**
1	2.0
2	3.0

V. Training and Evaluation of the Model

A. Hyperparameters

The model was trained with the k-fold cross validation technique, where $k = 5$, and with the following hyperparameters:

- 10 epochs;
- a learning rate of 10^{-4};
- a dropout of 0.5.

We opted for the AdamW optimizer, because of its computational efficiency. Given that the training of the model would've required various resources and time, it seemed a reasonable option.

B. Tentative Evaluation

During the training phase, we noticed that a major issue occured while computing the accuracy of the model: the `torchmetrics` package wasn't able to use the output provided by the model and the label correctly. In order to account for this issue, we decided to infer the accuracy thanks to the evolution of other parameters during training and testing time, one of which being the loss function $\mathcal{L}$. At the end of each epoch, a checkpoint would be saved, making us able to recover the status of the model over time. We are listing on Table I the values of $\mathcal{L}$ for the first x epochs.

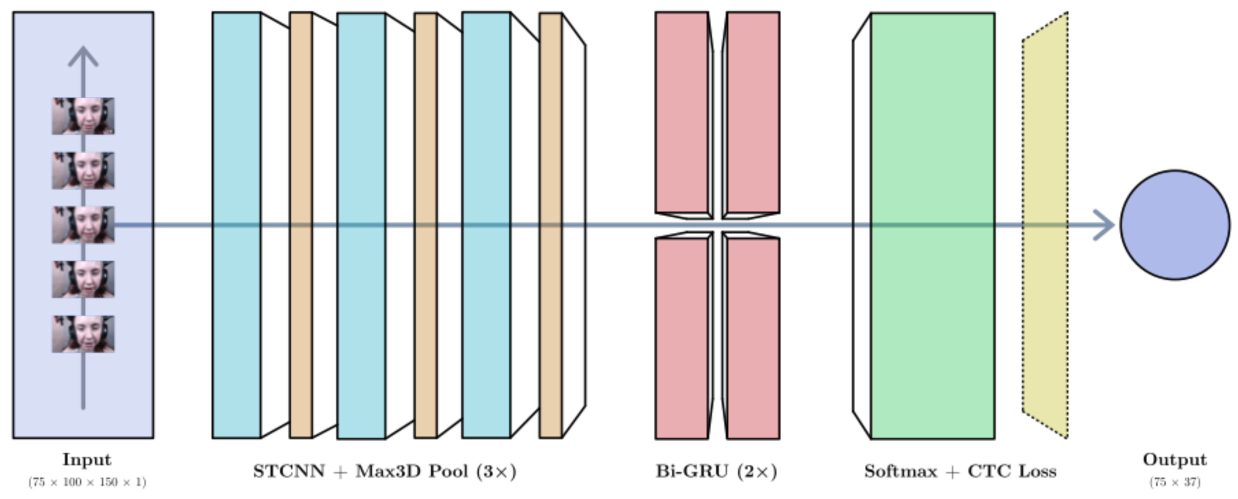


Fig. 3. Structure of the model employed

VI. Failure Reasons

Unluckily, this project didn't lead to the creation of an accurate model, but some conclusions can still be drawn. Why didn't it work? Why did some issues arise? We will thus try to answer to such questions.

First and foremost, why didn't it work as expected? This can be answered by looking at the values of the loss function over time: as we can notice, the value tends to increase over time, even for small amounts. This leads to think that, with enough time and resources, the model would eventually train well enough in order to have a satisfiable accuracy. We used a small number of epochs because of the necessity of this model, but the experiment could be easily replicated by using a wider number of epochs and by, possibly, augmenting the number of items in the dataset.

Another question that may arise is "*why did the loss function $\mathcal{L}$ assume negative values?*". We assume it's because

References

[1] U. of Sheffield, “The audio-visual lombard grid speech corpus,” last accessed 23 June 2024. Available here.

[2] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[3] Z. He, C.-Y. Chow, and J.-D. Zhang, “Stcnn: A spatio-temporal convolutional neural network for long-term traffic prediction,” in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, 2019, pp. 226–233.