UNIVERSITY OF SALERNO

COMPUTER SCIENCE DEPARTMENT

VISUAL SPEECH RECOGNITION

by Lipnet

Simone Benedetto

Salerno Daniele

2021/2022

Abstract

The goal of this project is to recognize words and phrases using lip recognition. The Convolutional Neural Network Lip-Net using the MIRACL-VC1 dataset will be used. It will then be explained the various experiments and changes made to ensure better performance and create a system capable of recognizing more skilfully the lip. In addition, we tried to create a dataset in Italian language to expand the possibilities of CNN and make it able to support an additional language.

1. Related Work

Lip recognition is very difficult for human beings. Over time we have tried, through the use of Artificial Intelligence, to facilitate this task, using machines to overcome this difficulty. The applications of this technology vary in different areas:

- It could allow a hearing impaired person to participate in conversations without using sign language, or participate in a video conference without difficulty. In addition, they could participate in social activities such as theater or cinema. It would break down social barriers and make life easier for a person with hearing loss.
- It could be used to generate subtitles, automatically, for any audio visual content.
- It could be used to recognize speech in noisy environments, where it is difficult to isolate the voice of a person you are interested in hearing what they are saying. In that case it could be used by intelligence services to intercept a conversation simply by using a video camera, and then doing so at a safe distance.

Automated lip-reading is difficult because it requires extracting spatiotemporal features from the video (since both position and movement are important).

In recent years, there have been several attempts to apply deep learning to lip reading. Such approaches perform word or phoneme classification. Approaches include learning multimodal audio-visual representations [1] [2] [3], learning visual features as part of a traditional speech-style processing pipeline (e.g., HMM, GMM-HMM, etc.) to classify word and/or phoneme combinations thereof.

Many of these approaches mirror early advances in the application of neural networks for acoustic processing in speech recognition [4].

Chung & Zisserman [5] propose spatial and spatio-temporal convolutional neural networks, based on VGGs, for word classification. The architectures are evaluated on a dataset consisting of BBC TV words (333 and 500 classes)

Chung & Zisserman [6] train an audio-visual matching model for learning pre-trained mouth features, which they use as input to an LSTM for 10-sentence classification on the OuluVS2, as well as a non-reading lip task.

Wand et al. [7] introduce recurrent neural network LSTMs for lip reading, but do not address either sentence-level sequence prediction or speaker independence.

Garg et al [8] apply a pre-trained VGG on faces to classify words and sentences from the MIRACL-VC1 Dataset [9], which has 10 words and 10 sentences. However, their best recurrent model is trained by freezing the VGGNet parameters and then training the RNN, rather than training them jointly. Their best model achieves only 44.5% accuracy in testing.

2. Proposed method

2.1 English Dataset

The English version of Lip-Net uses MIRACL-VC1 [9] dataset composed by “Train” of 2600 images for the training phase and “dev” of 200 images for the validation phase. The dataset consists of 10 phrases and 10 words which will be shown in Figure 1.

In the original dataset each folder was labelled with a numerical name (0 to 20) and when we trained the system, we found the next problem: each sample in each folder was assigned an incorrect class because the folders were processed in lexicographic order. To avoid this problem, we assigned each folder a letter (a to t), so that the labels provided in the dataset were correct (Figure 2).

ID	Words		ID	Phrases
1	*Begin*		1	*Stop navigation.*
2	*Choose*		2	*Excuse me.*
3	*Connection*		3	*I am sorry.*
4	*Navigation*		4	*Thank you.*
5	*Next*		5	*Good bye.*
6	*Previous*		6	*I love this game.*
7	*Start*		7	*Nice to meet you.*
8	*Stop*		8	*You are welcome.*
9	*Hello*		9	*How are you?*
10	*Web*		10	*Have a good time.*

Figure 1 - English Dataset label

Figure 2 - Dataset structure

Each image is composed of a sequence of 49 frames representing the lips of the speaker (Figure 3). By analysing the models made available in the project, we tried to use the model that achieves the best accuracy (52%).

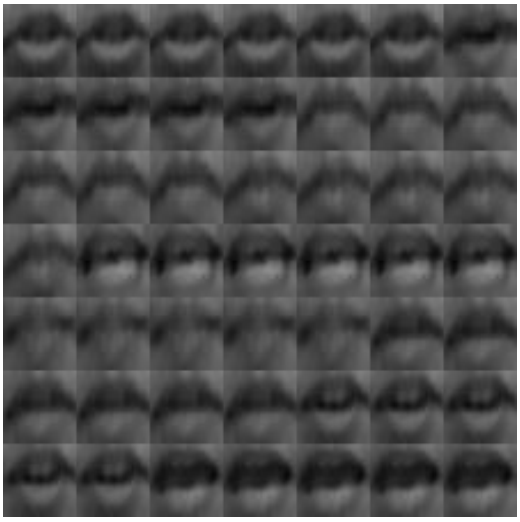


Figure 3 - Phrases "I am sorry"

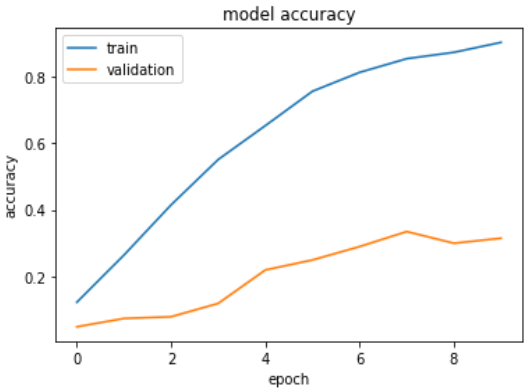
2.1.1 CNN with Adam optimizer

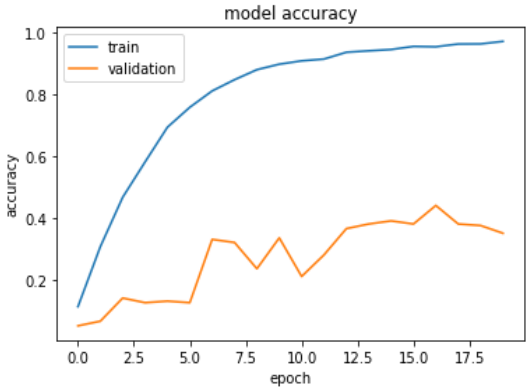
The model uses Adam as an optimizer and the Convolutional Neural Network used is shown in the next image.

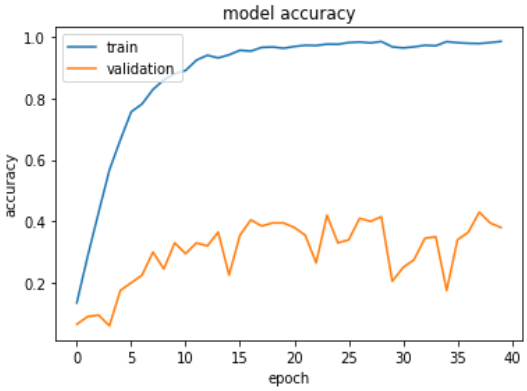
```
# Initialising the CNN
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Convolution2D(32,(2,2),input_shape = (224,224,1), activation = 'relu',strides=2,name='convo1'))
classifier.add(Convolution2D(64,(3,3), activation = 'relu',name='convo2'))
# Step 1 - Pooling
classifier.add(MaxPooling2D(pool_size = (2,2)))

# Step 2 - Convolution
classifier.add(Convolution2D(64,(3,3),activation = 'relu',name='convo3'))
# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2,2)))

# Step 3 - Convolution
classifier.add(Convolution2D(64,(3,3),activation = 'relu',name='convo4'))
# Step 3 - Pooling
classifier.add(MaxPooling2D(pool_size = (2,2)))

#Step 4 - Flattening
classifier.add(Flatten())
classifier.add(BatchNormalization())
classifier.add(Dropout((0.5)))
classifier.add(Dense(1024, activation = 'relu'))
classifier.add(BatchNormalization())
classifier.add(Dropout((0.4)))
classifier.add(Dense(20, activation = 'softmax'))
```

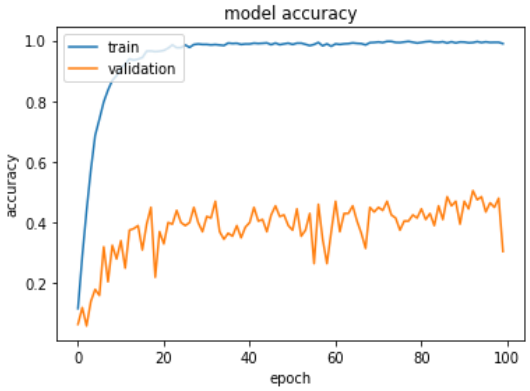
We trained the model with the following parameters:

NUMBER OF EPOCHS	VALIDATION ACCURACY
10	0.3150
20	0.3500
40	0.3800
100	0.3050









2.1.2 Create images for prediction

Our method to extract frame and plot image

There was no script for frame extraction, so we created own version.

We receive a video as an input and then work on it to get the final image, which is composed, as we showed before, of 49 frames. For each frame we take the facial landmarks and extract the mouth. Then we save it as a JPG.

```
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
```

The 68 facial landmarks it's a pre-trained landmark detector identifies 68 points ((x,y) coordinates) in a human face. These points localize the region around the eyes, eyebrows, nose, mouth, chin and jaw. The points from 49 to 68 represents the mouth.

```
vidcap = cv2.VideoCapture(video_path)
success, image = vidcap.read()
count = 0

# create new directory for frame
dirname = 'frame'
os.mkdir(dirname)

while success:

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # detect faces in the grayscale image
    rects = detector(gray, 1)
    for (i, rect) in enumerate(rects):
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        for (name, (i, j)) in face_utils.FACIAL_LANDMARKS_IDXS.items():
            if name == 'mouth':
                # h,w modify
                (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
                max_value = max(h, w)
                y = y - max_value // 3
                roi = image[y:y + max_value, x:x + max_value]
                roi = cv2.resize(roi, (250, 250))

    cv2.imwrite(os.path.join(dirname, "frame-%d.jpg" % count), roi)  # save frame as JPEG file
    success, image = vidcap.read()
    print('Read a new frame: ', success)
    count += 1
```

After this operation, we select 49 representative frames from the collection of frames. To perform this selection, we used a SSIM (Structural Similarity Index) method that measures how similar the images are by returning a value between 0 and 1 (the higher this value, the more similar the two images are). We calculate the SSIM between the image and its next for all the frames obtained, and then sort all the couples by SSIM value in descending order, so that we have the most similar couples of frames at the top.

```
dir_name = 'frame/'
# Get list of all files in a given directory
list_of_files = glob.glob(dir_name + '*')
# sort all files numerically
list_of_files.sort(key=lambda var: [int(x) if x.isdigit() else x for x in re.findall(r'[^0-9]|[0-9]+', var)])
img_n = []

for currIndex, filename in enumerate(list_of_files):
    if not os.path.exists(list_of_files[currIndex]):
        print('not exist', list_of_files[currIndex])
        break
    img = cv2.imread(list_of_files[currIndex])
    img1 = cv2.imread(list_of_files[currIndex + 1])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    s = ssim(img, img1)

    img_n.append([list_of_files[currIndex], list_of_files[currIndex + 1], s])
    currIndex += 1

    if currIndex >= len(list_of_files) - 1:
        break
```

We then select the last 48 pairs and delete the first frame in those that precede them. In this way we obtain 49 representative frames.

```
data = np.array(img_n)
data = data[np.argsort(data[:, 2])[::-1]]
print(data)
data = data[: -48]

for current, file in enumerate(data):
    delete(data[current][0])
    current += 1
```

Then we plot the images sorted numerically in ascending order to obtain the following image.

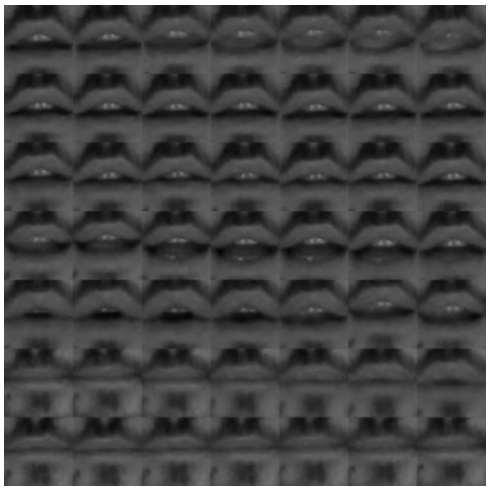


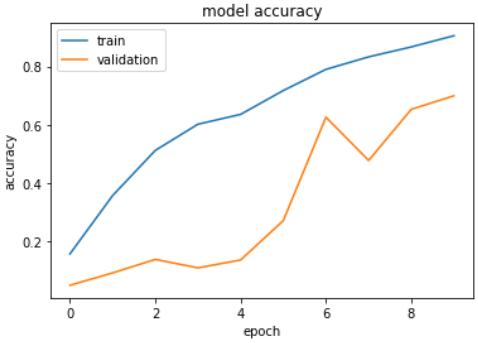
Figure 4 - Word "Hello"

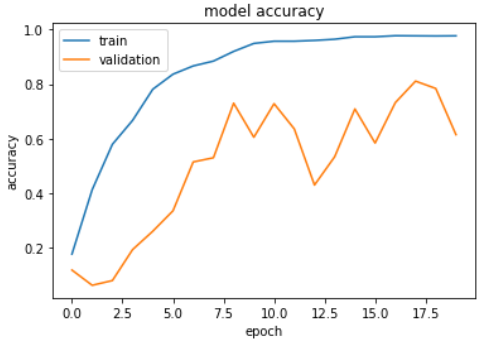
After obtaining images to test Lip-Net we did the prediction for each image, but they were not recognized.

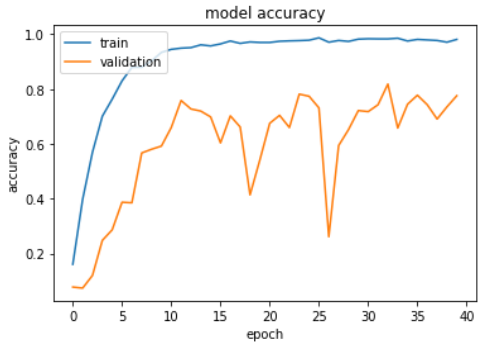
2.1.3 K-Fold

Checking the dataset, we realized that many images were blurred, especially those used in the validation phase, and that overall, they were not of excellent quality. So, we use K-Fold cross-validation using 5 splits to have 80% of the dataset for training and 20% for dataset validation

NUMBER OF EPOCHS	VALIDATION ACCURACY
10	0.669
20	0.811
40	0.818







After training, we tried prediction on the dataset "dev" with weights of 20 and 40 epochs and we obtained the following results:

- **20 epochs**: 39% accuracy;
- **40 epochs**: 41% accuracy;

It is important to underline that the images in the dataset “dev” are totally different from the images in the train dataset.

We then used the weights of 20 epochs to try and predict our images and again none were recognized.

So, we noticed that our images are clearly superior in terms of quality, and this could be a problem, also our method of image creation might not match the one used by them. For this reason, we decided to try to build an Italian dataset using our method.

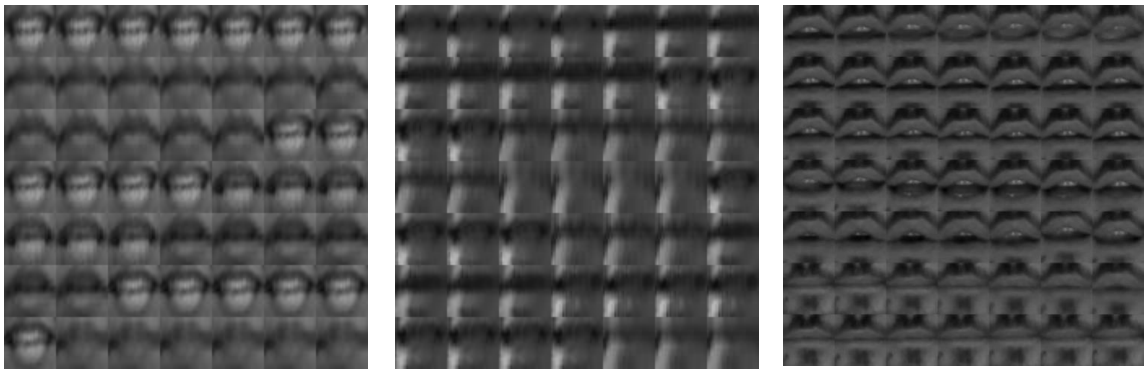


Figure 5 - Difference between original dataset images and our images

2.2 Building Italian Dataset

The Italian dataset is composed of 5 words and 5 phrases that are:

ID	WORDS		ID	PHRASES
a	Buongiorno		f	Come stai
b	Automobili		g	Tutto bene
c	Regali		h	Dobbiamo studiare
d	Telefono		i	Andiamo a casa
e	Università		j	Prendiamo un caffè

The words and phrases have been chosen to obtain different lips movements in the final images.

18 male and female individuals contributed to the construction of this dataset. To expand the size of the dataset and to include low or high light scenarios we made two variations for each original image. In addition, a high contrast copy was created for each original image.

Now we create the dataset in this way:

- Train contains 608 images
- Test contains 69 images composed by:
  - A part of images already contained in train
  - A new part of image different from train obtained by 3 new speakers

We did this because we had few images and consequently the test dataset was too small.

2.2.1 Training with ADAM

We used the same model explained in chapter 2.1.1, and then we trained it and we obtained, with 50 epochs, a 68% in validation accuracy.

It’s not a good result because we have, in “Test” dataset, some images contained in “Train” dataset.

2.2.2 K-fold

As we showed we have a little dataset, so we try to use K-Fold with 20 epochs and 5 splits, obtaining 19% validation accuracy.

2.2.3 Beard creates some problems

When we tested the model with Adam for prediction, we noticed that it recognized new female speakers more easily, and one of the male speakers who participated in the building of the training dataset. We therefore noticed that the beard creates some problems and perhaps it is possible to avoid this issue by enlarging the dataset. Since we did not have this possibility, we searched a method to remove the beard.

2.2.3.1 Colours recognition with K-means

Because the beard is the darkest part of the image, we decided to recognize the colours of the images using K-Means and replace the darkest pixels with a lighter colour, however any imperfections or shades in the image are composed of many colours, some of which are dark. This methodology proved unsuccessful at first because the images were completely distorted.

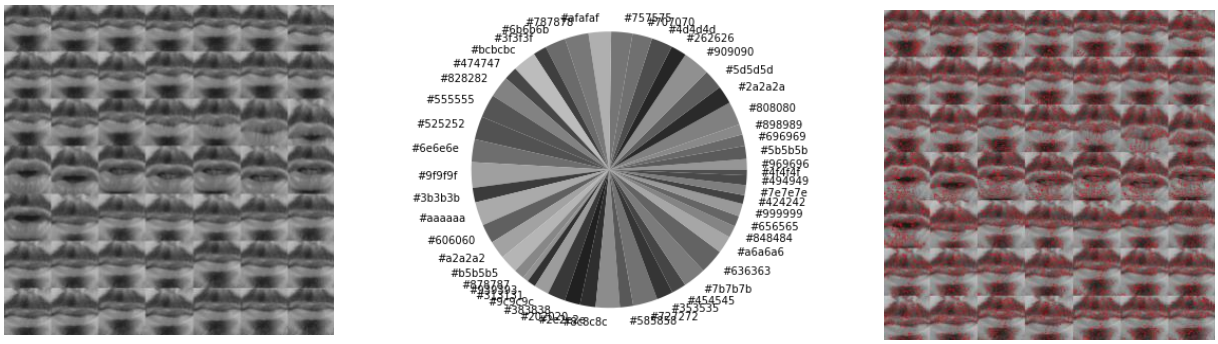


Figure 5 – Method to replace darkest pixel (the red colour is for illustrative purposes only)

2.2.3.2 Dataset with people with no Beard

Given these impediments, we deleted all bearded samples and used the new dataset to perform training and later validation. The problem is that the dataset is now down by 50%.

Using model with Adam on 50 epochs we obtained 51% validation accuracy.

2.2.4 Improve frame capture

Another problem we have encountered is that the images of the same class are not enough similar. This is because it depends on the moment when the speaker talks during the recording of the video.

For this reason, we thought of detecting the first frame in which the speaker starts talking and delete the previous frames in which he is silent.

2.2.4.1 Lip movement detect [10]

A simple RNN based detector that determines whether someone is speaking by watching their lip movements for 1 second of video. The problem however is that by analysing the video for each 25 frames, it does not return exactly the first frame in which they are speaking, but in which sequence of 25 frames they are speaking. Furthermore, on some videos the prediction was not very accurate. We therefore decided to use another method.

2.2.4.2 Mouth aspect ratio

This method allows us to detect if the mouth is open or not, by computing the distance between the two lips. In addition, it is necessary to establish a threshold, above which the mouth will be considered open. By performing some tests on different subjects, we noticed that the values obtained both when the mouth is closed and when it is completely open are different for each subject. For this reason, establishing a threshold was an obstacle.

We therefore decided to integrate this technique during frame extraction and lips detection. During these operations we calculate for each frame the mouth aspect ratio, and after processing the full video we perform the following operations:

- Calculate the minimum value and the maximum value between the detected values
- Calculate the average between the two values

- Calculate the mean error between all the values detected and the average
- Calculate the threshold by subtracting half of the mean error from the average.

Finally, we search the first frame in which the mouth aspect ratio is equal to or higher than threshold and remove all previous frames. We will probably have less than 49 frames to generate the final image. To solve this problem, we replicate the last frame of the sequence to

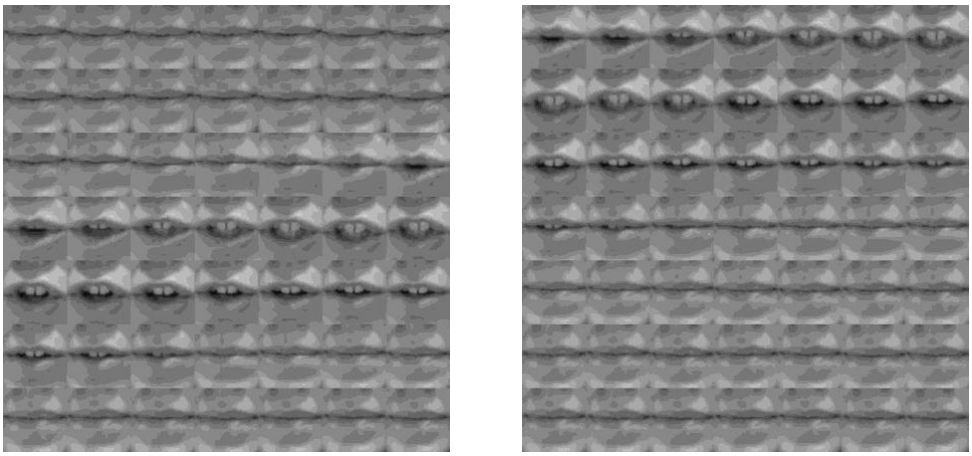


Figure 6 – "Buongiorno" generated with the old method (left) and the new method (right)

reach 49 frames. On the opposite, if we have too many frames, we apply the SSIM method to remove similar frames.

As we can observe, with the new method, we remove the initial frame in which the speaker is not talking.

With this method we created a new Italian dataset and to expand its size we made three variations for each original image:

- Low brightness
- High brightness
- High contrast

After we split the dataset in this way:

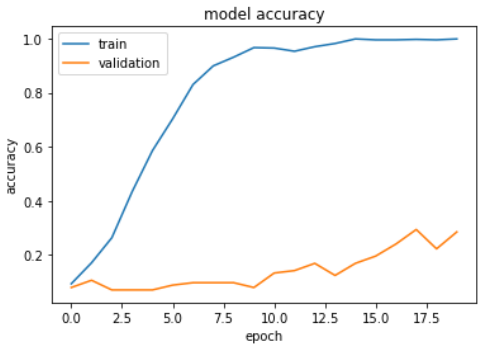
- Train composed of 594 images
- Test composed of 112 images.

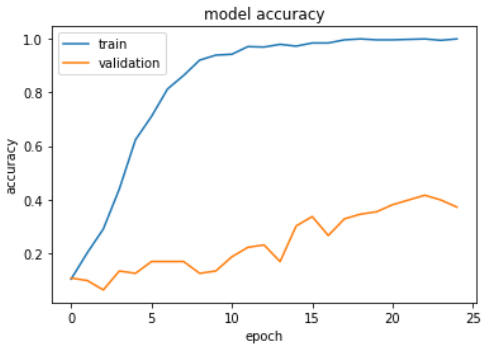
In this case in dataset "Test" we have 3 new speakers, 2 males with no beard and 1 female. In this way, the results obtained in the validation phase will be reliable.

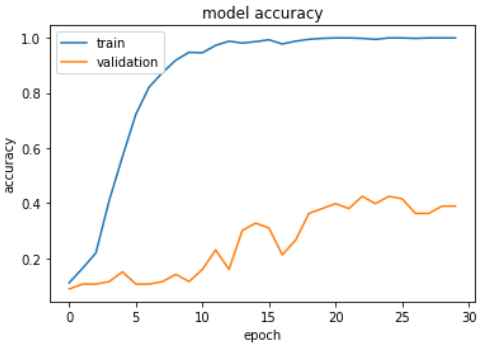
2.2.4.3 Training with ADAM

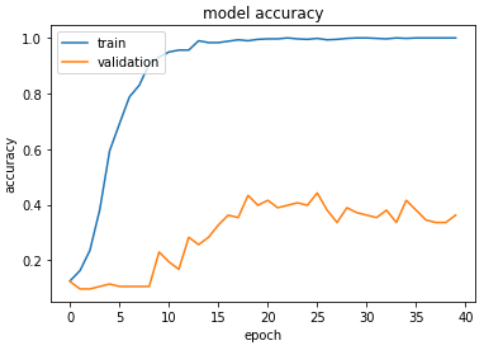
We used the same model explained in Chapter 2.1.1 obtaining the following results.

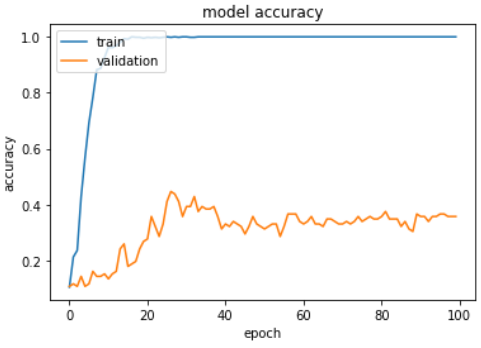
NUMBER OF EPOCHS	VALIDATION ACCURACY
20	0.2857
25	0.3717
30	0.3894
40	0.3628
100	0.3571







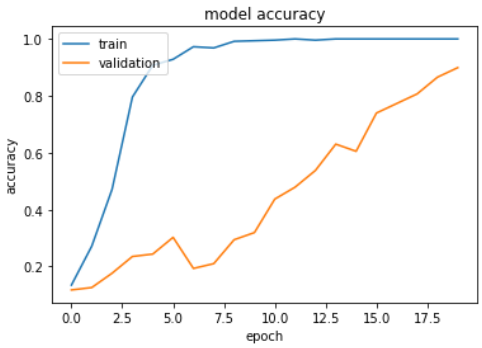


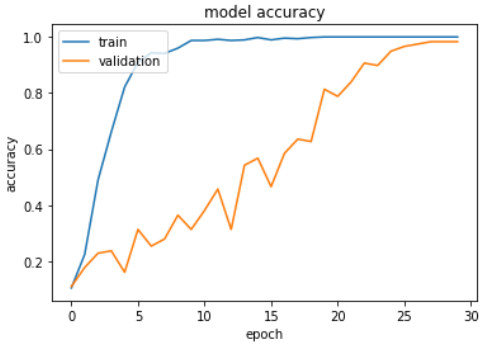


2.2.4.4 K-fold

We also tried K-fold with 5 splits, and we obtain the following results.

NUMBER OF EPOCHS	VALIDATION ACCURACY
20	0.8991
30	0.9830





So, we used the weights of 20 epochs on "Test" dataset and we obtained 0.2743 of accuracy. Instead, with weights of 30 epochs we obtained 0.2566 of accuracy.

2.2.4.5 Training with RMSPROP

We used the same model explained in chapter 2.1.1 but with different optimizer and we obtained the following results.

NUMBER OF EPOCHS	VALIDATION ACCURACY
20	0.2920
100	0.3363

2.2.5 Remove details

Such poor results may result from too much information in a single image. For this reason, we have applied different method to the images in the dataset. These methods will be applied separately on different copy of original dataset.

Resize images

The first resizing was to bring the images from 224x224 to a size of 100x100. The second resizing was to bring the images from 224x224 to a size of 50x50.

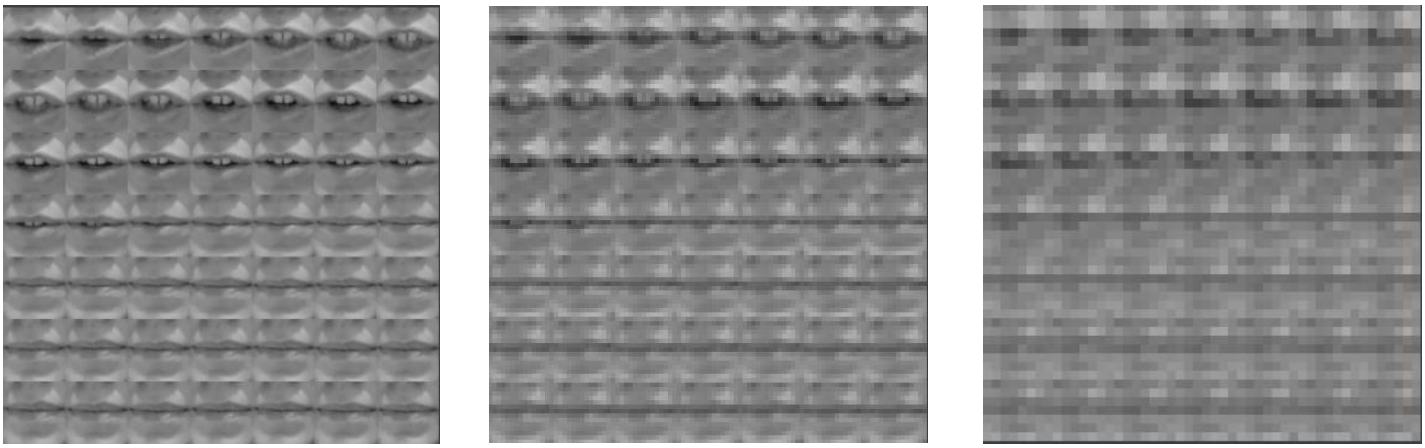
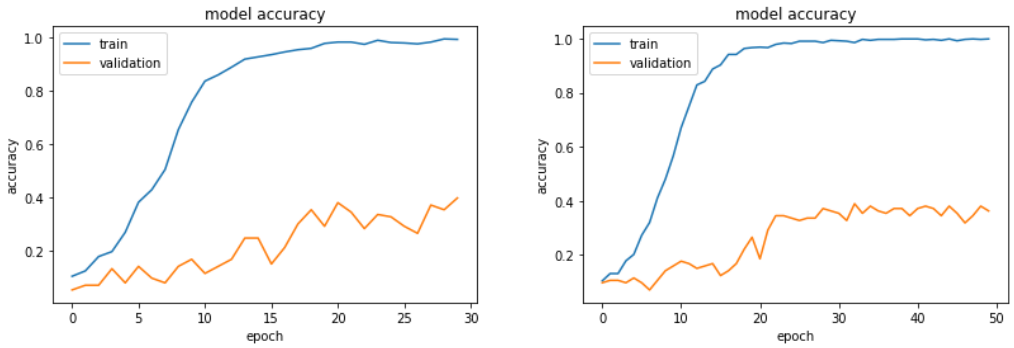


Figure 7 - (left) 224x224, (center) 100x100, (right) 50x50

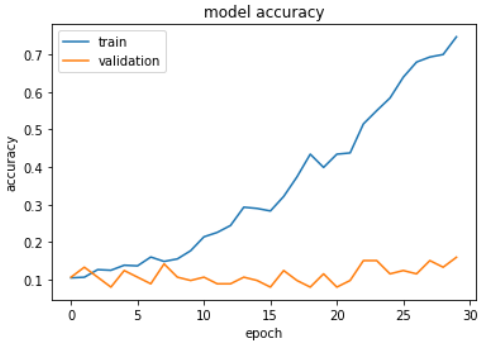
We used the same model explained in Chapter 2.1.1 obtaining the following results on images 100x100.

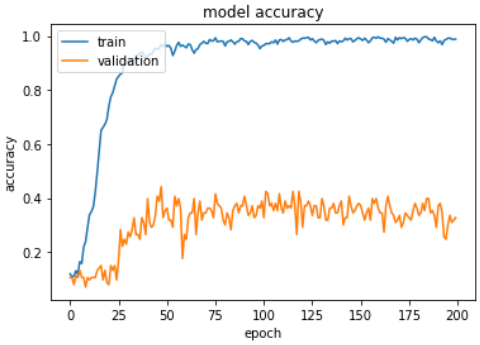
NUMBER OF EPOCHS	VALIDATION ACCURACY
30	0.3982
50	0.3628



We used the same model explained in Chapter 2.1.1 obtaining the following results on images 50x50.

NUMBER OF EPOCHS	VALIDATION ACCURACY
30	0.1593
100	0.3274





Gaussian Blur

We then used the filter Gaussian Blur. Two variations were made: radius 3 and radius 5.

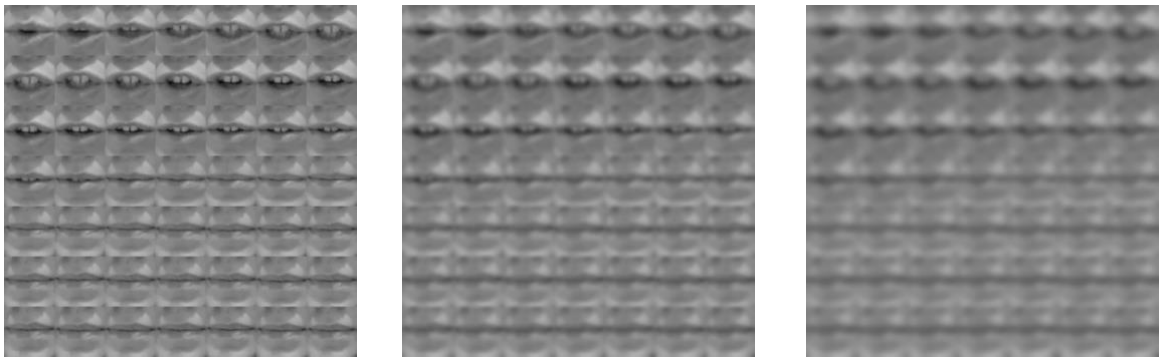
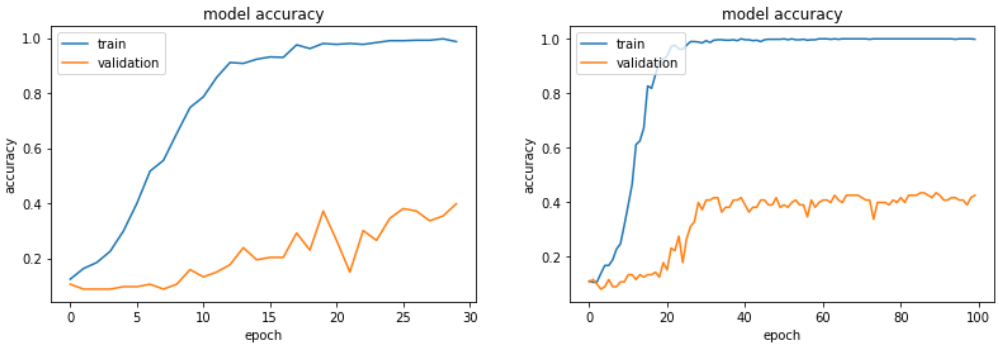


Figure 8 - (left) original, (center) radius 3, (right) radius 5

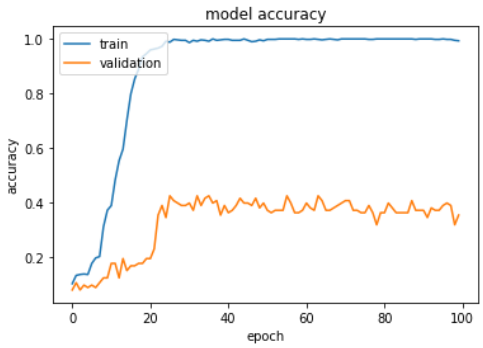
We used the same model explained in Chapter 2.1.1 obtaining the following results on blurred images (radius 5).

NUMBER OF EPOCHS	VALIDATION ACCURACY
30	0.3982
100	0.4248



We used the same model explained in Chapter 2.1.1 obtaining the following results on blurred images (radius 3).

NUMBER OF EPOCHS	VALIDATION ACCURACY
100	0.3540



Negative images

As a last method we used the negative of the images. This choice allowed us to remove information from the images (such as the beard) while highlighting the lips.

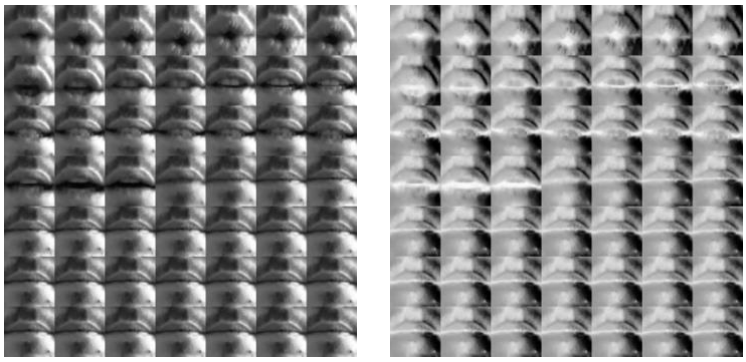
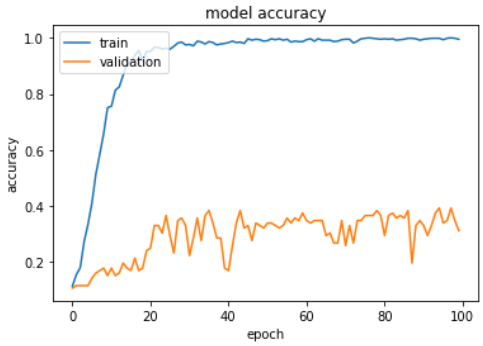


Figure 9 - (left) original, (right) negative

NUMBER OF EPOCHS	VALIDATION ACCURACY
100	0.3125



2.2.6 Merge Dataset

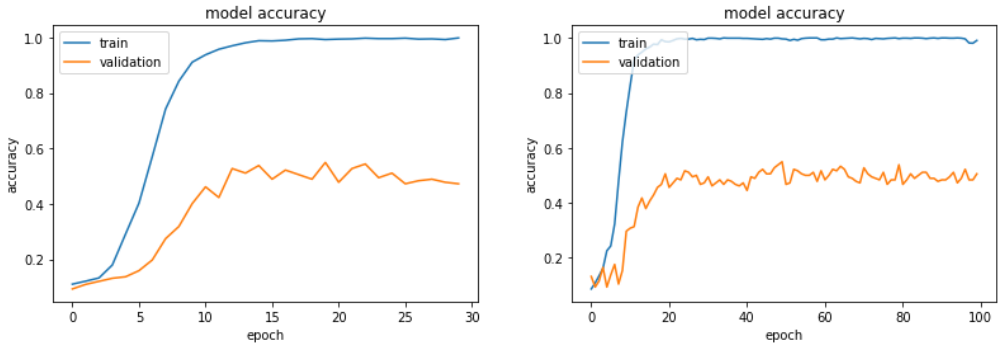
As a final experiment we tried to create a new dataset generated by joining the dataset created in Chapter 2.2 and the dataset created in paragraph 2.2.4.2. The reasons for this choice are:

- larger size of the dataset. “Train” is composed now by 1202 images and “Test” by 182 images;
- by analyzing the dataset created with the ratio method, we realized that some images have errors, and the initial frames chosen do not correspond to the exact moment in which the speaker is speaking;

So, we decided to merge the two datasets because many of the images in the original dataset have the same shape.

We used the same model explained in Chapter 2.1.1 obtaining the following results.

NUMBER OF EPOCHS	VALIDATION ACCURACY
30	0.4725
100	0.5055



2.2.7 LipNet: End-to-End Sentence-level Lipreading

As a final test, given the unsuccessful results of using CNN and the impossibility of expanding the dataset, we decided to use a further version of LipNet [11]. LipNet is an end-to-end model that performs sentence-level sequence prediction for visual speech recognition.

We then made predictions with both their videos and ours by uttering the following sentences.

PHRASES (repository video)	RESULTS
set white with p two soon	set white with p two soon
lay blue by c two again	lay blue by c two again
bin blue at f two now	bin blue at f two now

PHRASES (our video)	RESULTS
set blue by m six again	bin blue a n nine again
set blue by m six again	bin red at d nine again
set blue by m six again	a blue by s six soon
set blue by m six again	bin blue I s six again
lay green at c three soon	bin red at k nine soon
lay green at c three soon	place green i b nine soon

Using their videos to make predictions, we got excellent results.

Using our videos we obtained a good result, because of our difficulty in pronouncing the sentences correctly.

2.3 Web Application

The goal of the work done was to integrate the Lip-Net into a web application that provides a user-friendly interface. The system gives the possibility to the user to start (Figure 7) and stop (Figure 8) the recording and wait for the result of the neural network. The system provides the possibility to delete the recorded video (Figure 9) and record a new one in case the user does not pronounce the sentence or word correctly. As previously described the datasets used were two, one in Italian and the other in English. Through a switch it is possible to choose the desired language.

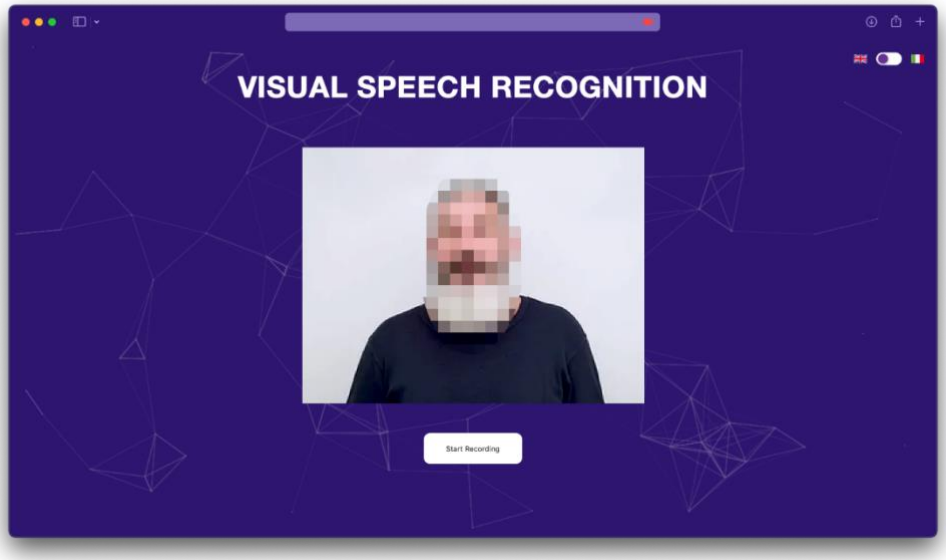


Figure 7- Start recording

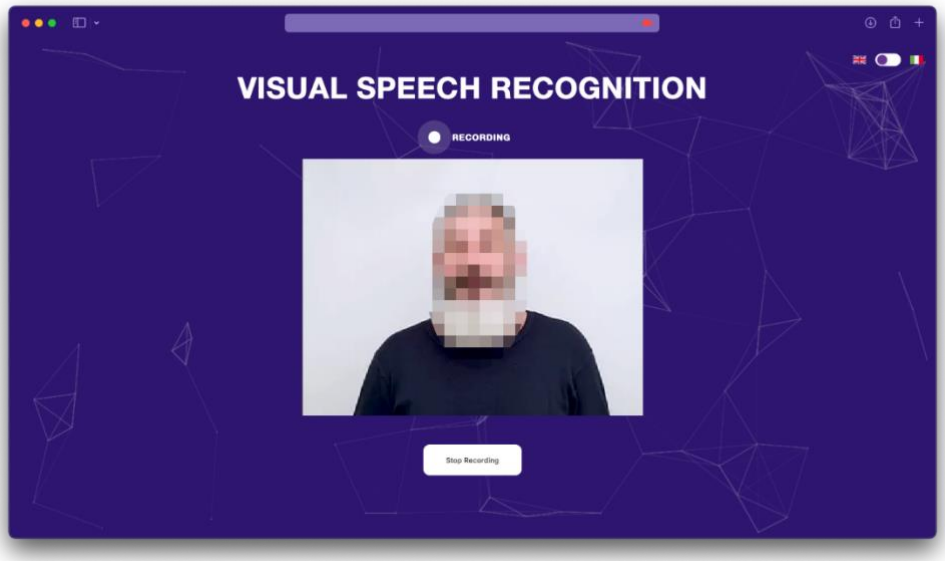


Figure 8 – Stop recording

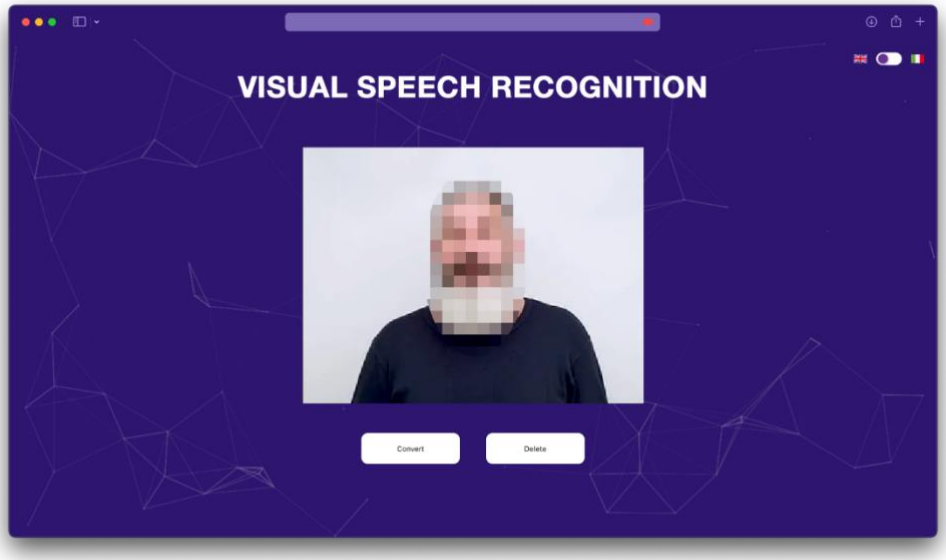


Figure 9: Convert or delete recording

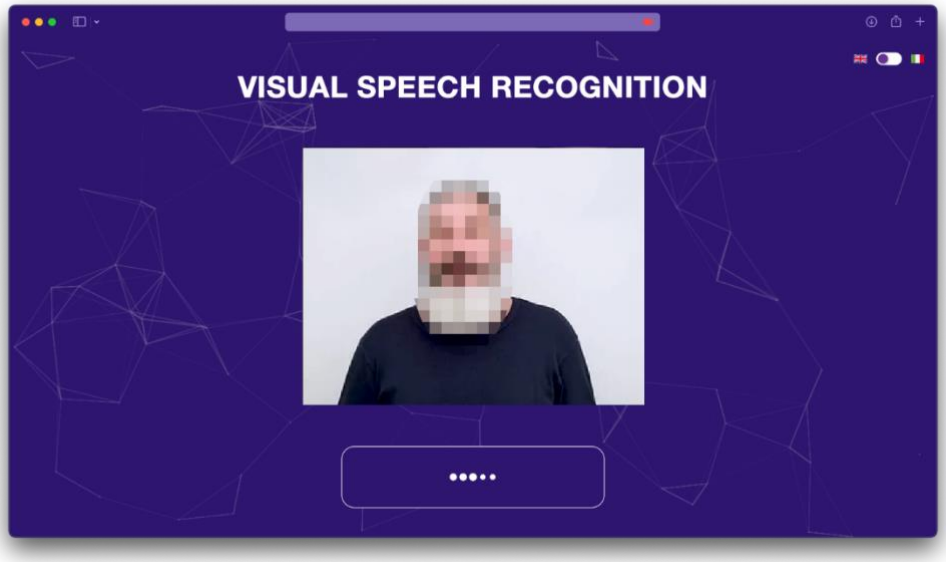


Figure 10 – Waiting prediction

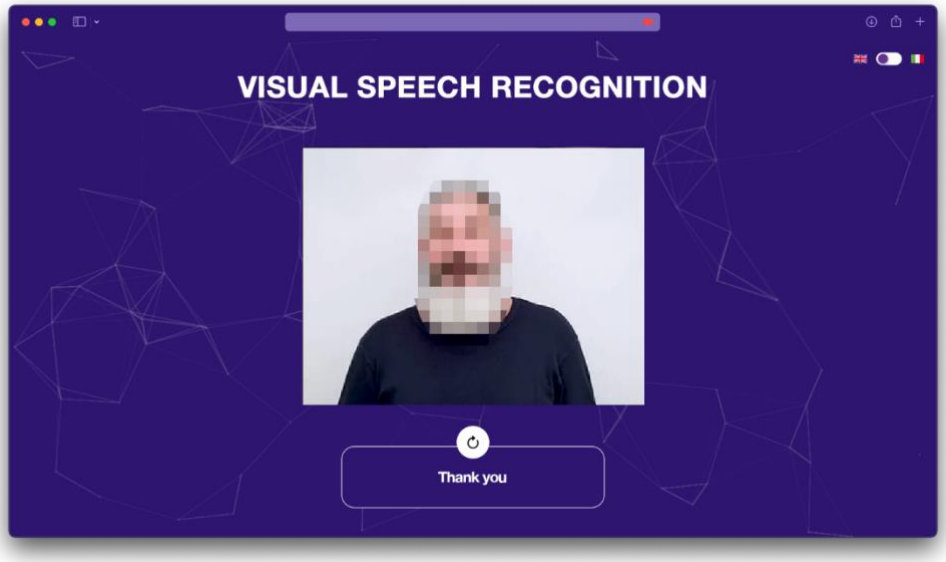


Figure 11 – Result

2.3.1 Architecture

They have been used the technologies HTML, CSS and JavaScript for the construction of the web page. As server has been used Flask on which are executed python scripts necessary for the functioning of the system.

After the video is acquired on the client side, using webcam, it's sent to the server, which will take care of frame extraction and lip detection. It will then take care of selecting the required 49 frames and plot the images. It will then take care of performing the prediction of the generated images and send the response to the client.

3. Conclusions

In conclusion, we first performed experiments on the MIRACL-VC1 dataset using the best model available in the project. After training the model, we created our own images to make the predictions with very negative results. We noticed that the images used for validation were not of the best quality and performed K-Fold Cross Validation and using the weights obtained our images were not recognized. We then noticed that our images were of higher quality than those used in the training, so we decided to create an Italian language dataset.

Using the same model and also the K-Fold methodology we did not get good results. When making predictions with new speakers we noticed that girls were recognized much better than men and therefore we thought that the beard might be a problem. We therefore tried to solve this problem by removing the beard using K-Means to recognize the colours in the image and replacing the darker colours (beard) with lighter colours in the image. This approach failed because the shadows and various shades in the image created problems. To test our hypothesis, we eliminated the samples with beards from the dataset, resulting in a halved dataset, and after training the model we obtained lower results than in the previous experiment.

We then saw that the images of the same class were not very similar, this was because of the frame capture and the moment when the speaker starts to talk. To solve this problem, we used a library that uses an RNN to detect when the speaker starts talking. However, this method did not return the exact frame in which the speaker started to talk but in which second he was doing it. Moreover, the prediction was not very accurate. Therefore, we decided to find the first frame in which the mouth is opening by calculating the distance between the two lips and setting a threshold. The first frames below this threshold will be deleted. We then created a new dataset using this new method and trained both the model with Adam and the K-Fold methodology without obtaining any improvements in the validation phase.

We then thought that too much quality was a problem, so we created several datasets using different methodologies for removing details.

We created both a dataset with images reduced from 224x224 to a size of 100x100, and a dataset with images of size 50x50. We obtained similar validation results to the other experiments with the 100x100 images only.

We created two other datasets with blurred images applying two different blurs (Radius 3, Radius 5) obtaining a small improvement in the validation phase compared to the previous experiment only with the blurred images using a Radius 5.

We created a dataset with negative images to try to bring out the lips and put in the background the beard and other types of details obtaining lower results than in the previous experiment.

We then created a new dataset combining the original Italian dataset with the one created with the Mouth Aspect Ratio method, with the intention of obtaining a larger dataset and to cover some errors generated by the choice of the threshold for the Mouth Aspect Ratio. We performed the training and obtained validation results in line with the previous experiments.

Finally, we used a new version of LipNet that makes use of a RNN for sentence-level prediction. Using their videos for prediction we obtained excellent results. With our videos we obtained discrete results, which can be attributed to the difficulty in pronouncing the sentences correctly.

After the experiments made using the Italian datasets, we think that to reach a high accuracy it might be necessary to increase the size of the dataset. By doing so the dataset will become rich in images that will cover various error scenarios.

Treating the sequence of frames as a single image with CNN is not very suitable because they will be seen as a whole and not as a sequence. It could happen that two images representing different words are very similar to each other thus causing a classification error.

As seen in section 2.2.7 the results obtained using a RNN are better, so we think that integrating a RNN inside our system can increase the performance.

Bibliography

[1] http://www.icml-2011.org/papers/399_icmlpaper.pdf

[2] https://www.isca-speech.org/archive_v0/interspeech_2015/papers/i15_0563.pdf

[3] https://dl.acm.org/doi/abs/10.1109/ICASSP.2016.7472088

[4] https://ieeexplore.ieee.org/document/6296526

[5] https://www.robots.ox.ac.uk/~vgg/publications/2016/Chung16/chung16.pdf

[6] https://www.robots.ox.ac.uk/~vgg/publications/2016/Chung16a/chung16a.pdf

[7] https://arxiv.org/pdf/1601.08188.pdf

[8] http://cs231n.stanford.edu/reports/2016/pdfs/217_Report.pdf

[9] https://sites.google.com/site/achrafbenhamadou/-datasets/miracl-vc1

[10] https://github.com/sachinsdate/lip-movement-net

[11] https://github.com/rizkiarm/LipNet