Cantonese Lip-reading

St. Paul's Co-educational College

Steven Luo 4I (4)
Samuel Yau 4I (8)
Julian Chu 5D (6)
Woody Lam 5D (8)

Abstract
In this paper, we would present to you our method of data collection, data preprocessing, and evaluate the different models trained upon our dataset. Our work is divided into 3 main stages: 1) comparing CNN and LSTM models with both raw pixel data and dotpos data, 2) combining CNN and LSTM to improve model performance, and 3) building models using both dotpos data and raw pixel data at the same time to build a more robust model. This works serves as a direction for lip reading as well as video processing in general.

1. Introduction

In recent years, more and more people are using their smartphones for typing, yet the traditional way of typing with an on-screen keyboard is quite inconvenient. While voice typing is developed as an alternative, lip reading might be a better choice in noisy environments and occasions where we are not allowed to produce any noise (e.g. in a library). Lip reading is therefore a hot topic with well-developed models for English and Mandarin. While those models may have accuracy of over 70%, such model is non-existent for Cantonese, not to mention one with high accuracy.

As a pioneer project, we have selected 23 frequently used Cantonese words and created a dataset for our 4 different models. As a highlight of our project, the models were trained with not only raw footage but also preprocessed footage that went through feature extraction using the DLIB module. We used dlib's pretrained model to trace readers' lips using 20 dots and fed the (x, y) coordinates into our models to train. We reviewed the performance of different algorithms in classifying words from footages and evaluated the merits and demerits of each model.

As machine's lip-reading is getting better at a rapid speed, actual implementations of such technology may be seen in the near future. this technology is expected to be used as a means of typing, a tool of expression for the mute, or even a teaching utility to correct pronunciation of Cantonese words in the future.

2. Related works

2.1 Related works on machine lip-reading

Apart from voice recognition and facial recognition, lip recognition or lip-reading has also become one of the hottest topics of artificial intelligence. Vast improvements on lip-reading accuracy for both word-level and sentence-level predictions have been made over the past years, with the help of better feature extractions and deep learning.

Yannis et al.'s [1] work is probably the most popular among all lip-reading projects. Collaborating with Google Deepmind, they have created LipNet, a sentence-level model which performed better than previous word-level state-of-the-art accuracy. Unlike conventional models whose predictions are based on visemes detected, LipNet was able to predict on the sentence-level, which resulted in better results.

Besides deep learning classifications, conventional methods were also capable of lip-reading tasks. With the help of feature extractions or optical flow, conventional methods were also able to yield satisfactory results. Ayaz et al. [2] used optical flow (OF) to estimate lip motions and trained it with a support vector machine. Their SVM was trained to predict 14 visemes and obtained 95.9% classification accuracy.

It was only until lately, when lip-reading tasks were introduced to the Chinese area. In 2017, Chinese big data analysis solutions provider HyData has trained a Chinese-word model upon their 10-thousand-hours news footage. They have achieved 71% classification accuracy.

Currently, most state-of-the-art lip reading models are constructed with deep convolutional neural networks (CNN), recurrent layers (e.g. Long-short-term memory) or deep-layered neural network (DNN). Sainath T. N. et al. [3] investigated the performances of different combinations of the three networks and concluded with a best-model (CLDNN), which consists of all three networks. By combining the three layers, these models are able to extract features from the footages and find the correlations between different features or different frames, thus achieving good accuracies. In 2017, Vaswani A. et al. [4] developed a new "transformer" model, that has achieved better results than CNN + LSTM state-of-the-art models.

<u>*2.2 Brief illustration of our previous work*</u>

This project first started 5 months ago as a submission to join the Sensetime competition -- "The 1st International Middle School Students Artificial Intelligence Exchange Exhibition", which ended just about a month ago. Our project was ranked as the top 8 teams. In this part, we briefly explain our previous work for the competition and its connection with our current work.

For our previous competition, our project was divided into 2 stages: *1. Experimenting with conventional methods 2. Further exploration with deeper models.* In the first stage, statistical models were trained to investigate the effects of using gray-scaled videos and the significance of performing Dlib's data preprocessing. It provided us clues for our second stage of experiments where we trained our better models. In the second stage, we build deeper models upon our CNN networks from stage 1. Long-short-term memory layers were added and encouraging results were obtained. Our *best-model* was concluded to be a 3D-CNN + LSTM Network.

In our current project, we focus on making adjustments to our CNN + LSTM model to decrease its size and improve its performance. We also worked on a bilinear model which trained upon both raw-pixel values and Dotpos data. This model will be our final *best-model* with testing accuracies of 93%.

3. Data Collection:

<u>*3.1 Human lip-reading data collection*</u>

As the basis of our project lies on the hypothesis that visemes contain useful features to predict Cantonese words, we are curious to know about the performance of human brains in terms of processing the visual-only visemes footage so as to determine the corresponding words.

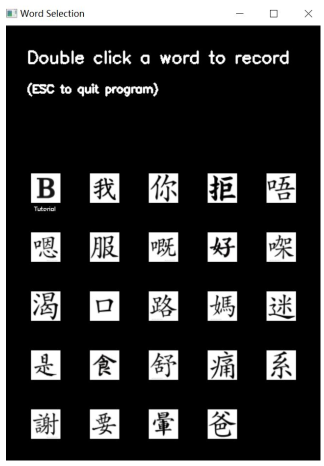
Such survey is conducted by an online questionnaire through Microsoft forms which not only answers but also total time taken to complete are recorded for our comparison between human beings and our trained models.

After sampling with some 30 people, the average accuracy score obtained is 46% while an average person would need 21 seconds to determine what word the video represent, regardless of correctness. While a significant portion of answers is "Unable to predict", we would conclude that CNN is able to perform considerably better than human brains in identifying the features of visemes and in a much more efficient manner. Such discovery is also in line with the general belief that CNN is capable of identifying more useful features from the same data than human beings when conducting classifications.

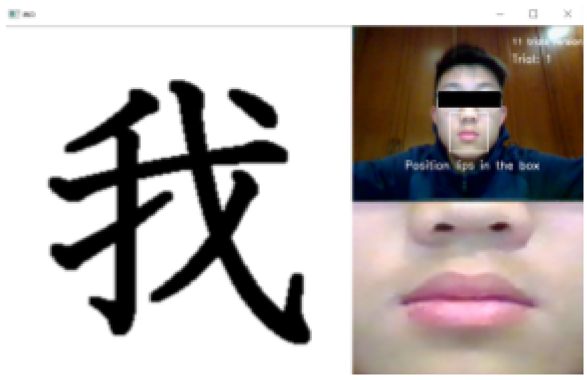
<u>*3.2 Training data collection*</u>

We created our own dataset of 23 different Cantonese words, with 200 samples per word in the first stage, and further expanded the size of the dataset in stage 2. The 23 words were chosen according to their importance in daily lives. Words such as "wo", "ni" are common daily words and "tong" or "yun" are words used during emergencies.

The dataset consists of two kinds, footage with lip only and footage with the entire face. We separated the data into two kinds to train our models upon different algorithms, which would be discussed later. We created a program that allows readers to select a word out of the 23 and records them saying the word for 20 times.



(Fig 3.1: Word selection page)



(Fig 3.2: Recording)

The program provides a 2-second interval for readers to say the word and records two footages, a 100 x 100 lip footage and a 640 x 480 full footage. This lowers the workload on data preprocessing afterward, as the size of each input instance was kept constant, with 40 frames per sample (20 fps) and constant frame size throughout. Footages were recorded by laptop webcams with the lightings of the environments made sure to be bright enough for lip features to be detected. 10 readers recorded 20 times per word, creating a dataset of 200 samples per word and a total of 4600 samples. We divided the dataset into 3 parts: training, validation and testing, each consisting of 150, 30, 20 samples per word respectively. Validation set was used to evaluate the models with accuracy scores and confusion matrices, which in turn would assist us with the fine-tuning of respective models. Testing set was untouched until the final analysis of models.

4. Data preprocessing:

Generally speaking, we used two different sets of footages to train our models: 1. Only-lip footage 2. Face-footage. For lip-only footage, raw pixel values are being used to train the models, making every instance a 400000-dimensional array after grayscaling. For face-footage, lip edges are traced with 20 dots and the coordinates of each dot are saved. Coordinate values are normalized and resized by applying our own algorithm to the raw coordinate values. Resulting instance is a 1600-dimensional array.

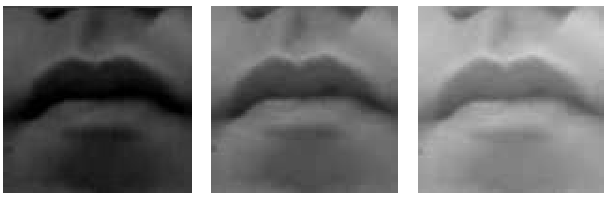
For lip-only footage, grayscaling was applied to reduce the size of each input instance and the rationale of which will be explained in section 5. Each sample size is decreased by 3 times, from (40, 100, 100, 3) to (40, 100, 100, 1). Then, the sample matrix is resized into a 1D matrix of (1, 400000).

For face-footage, we reduced the training time and the dimensionality of each instance, by using coordinates of lip features instead of raw pixel values to train our second model. To do so, we used the pre-trained feature detection model "dlib_shape_predictor" to outline the lip with 20 dots. This model is an implementation of the "One Millisecond Face Alignment with an Ensemble of Regression Trees" by Kazemi and Sullivan (2014) and can be used for real-time detection and achieves satisfactory quality. Coordinate values of each point were then fed into our algorithm to be normalized and standardized.

Data Augmentation

After the first stage of experiments and evaluation, we decided to expand our exploration in the direction of constructing deeper and more complex neural networks. However, the training of deeper neural networks relies even heavier on larger amounts of training data, as each additional layer of network adds huge number of trainable weights to the model. Thus, we had to expand our dataset. After a new series of data collection, our dataset has increased from having 200 footages per word to having 428 footages per word (except for “en”, which has only 409 footages, due to video corruptions). Newly added data have the same format as all previous data, i.e. 40 frames per footage and consists of both only-lip and full-footage.

To further increase our data size, small degree of data augmentation was done. As video transposing would be unnecessary with the presence of FC-CNN layers, only the brightness of videos were adjusted (i.e. grayscale values). 50 units of grayscale value was added to and subtracted from the original videos, resulting in two additional videos that appear in different brightness. Values lower than 0 or higher than 255 after augmentation would be treated as 0 and 255 respectively. The difference between the three videos are set to be significant enough that each could be identified as individual footage. Thus, the resulting data size was tripled.


(Fig. 4.1: Data augmentation. Left: -50 grayscale value. Middle: Raw frame. Right: +50 grayscale value.)

5. Stage 0: Investigations from the previous project

Previously, we have experimented various conventional methods for our video classification task. Models such as random forest classifier, support vector machine, multi-layered perceptron. Accuracy scores of respective models trained with both raw pixel values and 20-dot lip-shape data are compared. In this part, we focus on the SVM investigation for data preprocessing, which assists us on the training of later convolutional models.

Investigating the difference in accuracy and training time between coloured and grayscale image

Coloured images are resized from (40, 100, 100, 3) matrices to 1D matrices of (1, 1200000) while grayscale images are reshaped to (1, 400000). In consideration of limited time for training the models, this stage is designed to determine whether greyscale images would bring significant differences to accuracy scores, or instead be capable of replacing coloured images in the following training so as to save time.

In various trainings with different sizes of training samples, it is observed that surprisingly both sets provide exactly the same accuracy scores given the same sample sizes. This reflects that the reduction in features and dimensions in grayscale images has virtually no significant effect on the effectiveness of the models. In contrast, in terms of time required for training, considerable differences in duration can be found, which are reasonable owing to the fundamentally tripled amount of information. Nonetheless, this stage clearly reflects that grayscale images are able to strike a balance between accuracy and time needed, and thus such method would be adopted in the following stages for all methods.

6. Stage 1: Convolutional neural network and long-short-term memory

In our previous project, we only experimented with the two CNN models using raw pixel values. In this project, we build an extra CNN model trained with dotpos data, as an attempt to create a smaller model and to further utilize our preprocessed dataset. This Dotpos-CNN ends up as our *best-model* in Stage 1, proving the positive effect of preprocessing the data.

Recent improvements in computer video classification rely largely on the integration of Long-Short-Term memory (LSTM) modules to the CNN layers. The combination of CNN and LSTM provides better spatiotemporal detections, as the

CNN layers extract important features at a certain time frame while the LSTM layers connect the output of each period and evaluates upon the temporal dimension. Thus, in this stage, we build individual LSTM models to evaluate their performance.

6.1. Convolutional neural networks

We built 3 CNN models: 3D CNN and 2D CNN using raw pixel data, and a CNN using dotpos data. The CNN for dotpos is newly built for this project.

6.1.1 2D Convolutional neural network using raw pixel data

Data Preprocessing

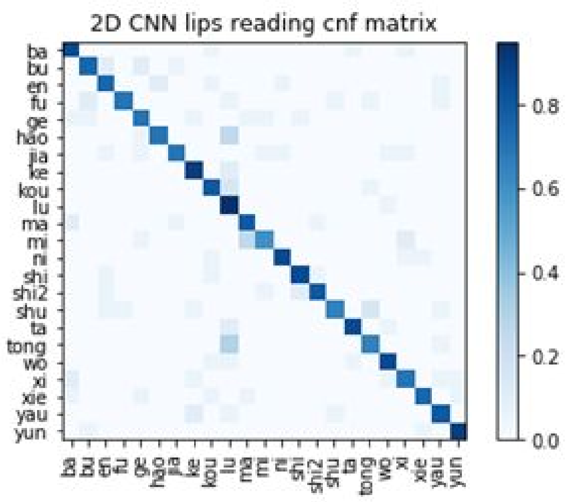
Frames of a video are concatenated into one long picture, resulting in 2D picture with dimensions of 100x4000 (the last frame is cut due to some faulty footages in the dataset, assuming that this produces insignificant impact on the experimental results).

Model Architecture

The model first feeds the data through a layer of 2D convolution, with a filter size 3x3, then a 2x2 max pooling layer, and then another layer of 2D convolution and max-pooling with the same parameters as the former. Lastly, the output is converted to a 1-dimensional array and is fed into a fully connected layer that maps the results into 23 units, each corresponding to a word class. The word class that has the highest value is the predicted word. The batch size is set to be 4 due to computational constraints and the learning rate is set to be 0.00001 (because of the small batch size).

Evaluation

The model converges after 16 epochs and reached a validation accuracy of 85% after 20 epochs. It can be seen that the word 路(lu) is often confused with other words, while the world 迷(mi) is often confused with 媽(ma). (fig 6.1, accuracy plot of 2D CNN)



(fig 6.1, confusion matrix of 2D CNN)

6.1.2 3D Convolutional Neural Network using raw pixel data

Data Preprocessing

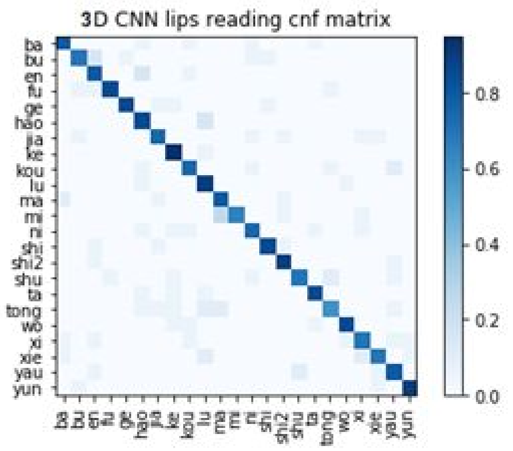
Each video sample is reshaped into a 3D object with dimensions of 100x100x40. This is done so that the same or nearby pixels across time frame is interconnected to each other and that the 3D kernel can capture features not only from the nearby pixels in a 2D picture but also the pixels in the consecutive time frames. By convolving nearby the time frames, the convolution neural network can analyze how the pixels change through time, and hence the movements of the lips.

Model Architecture

The model feeds the data through a layer of 3D convolution, with a filter size 3x3x3, then a 2x2x2 max-pooling layer, then another layer of 3D convolution and max-pooling with the same parameters as the forms. Then, the data is flattened and fed into a fully connected layer of 23 units. The word class that has the highest value is the predicted word. The batch size is set to be 4 due to computational constraints and the learning rate is set to 0.00001 because of the small batch size.

Evaluation

The model converges after 12 epochs and attained a validation accuracy of 86% after 20 epochs. Contrary to our expectation, the 3D convolution model is performing only slightly better (by 1%) than the 2D convolution. One thing to note is that the 3D CNN model is actually 5 times smaller in size than the 2D CNN, because the 2x2x2 max pooling layer is downsampling the sample two time more than the 2x2 max pooling layer. Similar to the 2D convolution, it is observed that the word 迷(mi) is often confused less with 妈(ma). However, 路(lu) is less confused with other words in this model than in the 2D convolution.



(fig 6.2, confusion matrix of 3D CNN)

<u>6.1.3 Convolutional neural network using dotpos data.</u>

Data Preprocessing

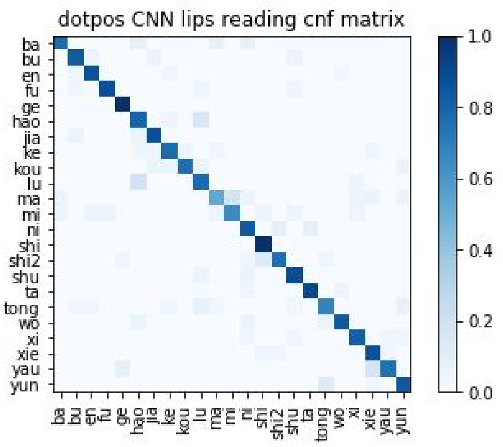
This model uses the dotpos data, which are the x and y coordinates of the 20 feature points on the lip predicted by the dlib shape predictor. The shape of each datum is 40x20x2, where 40 is the time dimension, 20 are the 20 points, and 2 are the x and y coordinates of each point.

Model architecture

The data first went through a 2D convolutional layer with 128 filters of shape 20x2 to convolve the 20x2 20 points coordinates in each time frame. The data, now with shape 40x1x1x128, is fed into a 1D convolution of kernel size 3 and then another 1D convolution of kernel size 2 to convolve the data along the time dimension. The output is then fed into the softmax output layer of 23 neurons. The batch size is set to be 20 and the training rate is set to be 0.0001.

Evaluation

The model converges after 140 epochs, and attained a training accuracy of 100, validation accuracy of 80 and a testing accuracy of 82. The model size is extremely small (2 orders smaller) compared to the models using raw pixel data. However, the model used to generate the dotpos from the raw footages(dlip shape predictor) is large.



(fig 6.3, confusion matrix of dotpos CNN)

5.2 Long-short-term memory models

Two LSTM models are trained. 1. Trained with raw-pixel values 2. Train with Dotpos data

<u>6.2.1 LSTM using the raw pixel data</u>

Data preprocessing

The data of shape 40x100x100 is reshaped to 40x10000 before being fed into the network

Model architecture

In this model, the data is directly fed into an LSTM with 512 neurons in the main cell, which gives it a total of around 500,000 trainable weights in the LSTM. The data is then fed into the output layer

Evaluation

<u>6.2.2 LSTM using dotpos data</u>

Data preprocessing

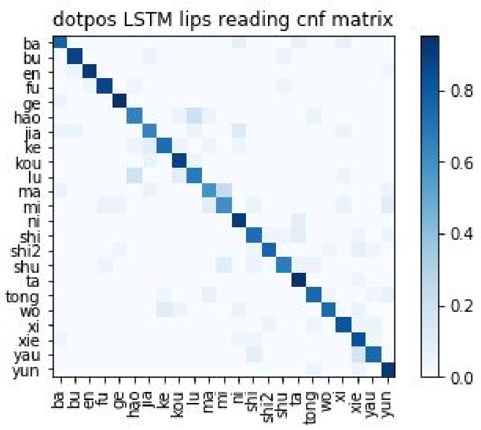
The data of shape 40x20x2 is reshaped to 40x40 before being fed into the network

Model architecture

In this model, the data is directly fed into an LSTM with 128 neurons in the main cell, which gives it a total of 86528 trainable weights in the LSTM. The data is then fed into the output layer

Evaluation

The model only attains a training accuracy of 95 after 300 epochs. Its validation and testing accuracies are 75% and 78% respectively



(Figure 6.4, confusion matrix of dotpos LSTM)

6.3 Stage 1 Combined Evaluation

Model	**Training**	**Validation**	**Testing**
2D-CNN	100	85	77
3D-CNN	100	86	79
Dotpos-CNN	100	80	82
Pixel-LSTM	29	26	--
Dotpos-LSTM	95	75	78

(Table 1: Accuracy scores of all models trained in previous competition)

CNN (both 2D and 3D) attained a considerably high accuracy score for both validation (85 and 86 respectively) and testing (77 and 79). Contrary to our expectation, 3D CNN only perform very slightly better than the 2D CNN. This may be due to the limited training time for 3D CNN, as 3D CNN contains more trainable weights but is trained only for 20 epochs, which is the same for 2D CNN. The difference of test accuracy and validation accuracy is slightly larger than the other models. A possible explanation of this could be that regularization methods (e.g, dropout layers) are not used in our CNN model. Moreover, CNN generally requires larger datasets to obtain accurate and robust results.

To our surprize, the models using dotpos as data attained a better testing accuracy than the models using raw pixels as data. This is contrary to our findings in the previous works, which shows that dotpos models perform poorer than raw pixel models in all conventional models (including random forest, support vector machine, and multilayer perceptron).

The dotpos LSTM model yielded satisfying results, with similar performances as CNN models. This proved LSTM modules' ability to capture relevant representation of the video sequence. The great performances also encouraged us to develop deeper models in Stage 2, by combining the two models together into one. However, the raw pixel LSTM model performed poorly. This might be due to the fact that the raw pixel data contains not just sequences but also images, something LSTM is not good at analyzing.

Moreover, compared to the raw pixel models, the dot pos models shows a smaller difference between their validation and training accuracies. In fact, the testing accuracies of the dotpos models is higher than validation accuracies in some dotpos models. This might be due to the fact that the dotpos models use the coordinates of the 20 points as data, which by nature is much more regularized than the raw pixels data, while the raw pixel data might be easily affected by factors like lighting and color tones.

7. Stage 2: Combining CNN and LSTM

In Stage 2, we combine our CNN models and LSTM models to allow our models to capture both spatial and temporal features from the video sequences. Three models were trained, namely 2D-CNN + LSTM, 3D-CNN + LSTM and Dotpos-CNN + LSTM.

Initially, this three models were trained during our previous competition. However, new versions of the 3D-CNN + LSTM and Dotpos-CNN models, which has less trainable weights (for 3D-CNN + LSTM) and better accuracy scores, were trained in this competition.

7.1 2D-CNN + LSTM (Raw pixel values)

Architecture

In this model, each frame is fed into a 2D convolutional neural network of 128 filters with a kernel size of 3x3. It is then passed on to a max-pooling layer of 2x2, and then another convolutional layer and max-pooling layer with the same parameters as the formers. Then, The feature maps of each frame is flattened and fed into an LSTM cell, according to their time sequence. Lastly, the output of the 40x128 LSTM is flattened and fed into a fully connected layer 23 neurons, and softmax is performed to find the prediction.

The reason an LSTM layer is added after CNN is to allow the mode to analyse the sequence with sequential features across frames, instead of only spatial features with a single frame.

Performance

This model attained a training accuracy of 100%, validation accuracy of 87%, and a testing accuracy of 84%.

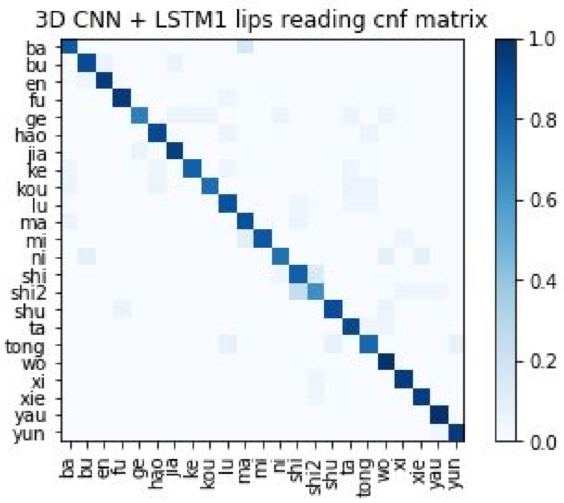
7.2 3D-CNN + LSTM (Raw pixel values)

Architecture

In this model, each frame is fed into a 3D convolutional neural network of 64 filters with a kernel size of 3x3x3. It is then passed on to a max-pooling layer of 1x2x2, and then another convolutional layer and max-pooling layer with the same parameters as the formers. Then, the 128 3D feature maps of each frame is flattened and fed into an LSTM per layer, according to their time sequence. Lastly, the output of the 40x128 LSTM is flattened and fed into a fully connected layer 23 neurons, and softmax is performed to find the prediction.

Performance

This model attained a training accuracy of 100%, validation accuracy of 92%, and a testing accuracy of 87%.



(fig 7.1, confusion matrix of 3D CNN+LSTM)

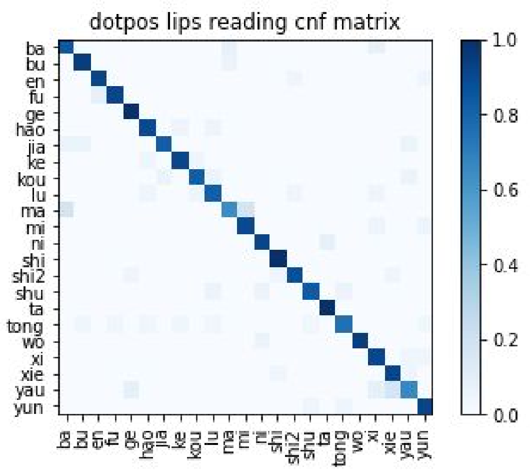
7.3 Dotpos-CNN + LSTM

Architecture

The x and y coordinates of the 20 dot positions of each time instance is first fed into a 2D convolutional layer with kernel size of 20x2 and 128 filters, where the first dimension are the 20 dots in the time instance and the second dimension 2 is the x and y coordinates of each dot. Then, the whole sequence of the 20 time instances of dot-position-convolution is fed into 2 1D convolution with kernel size of 3 and 2 respectively, outputting 128 filters. the output is fed into a many-to-one LSTM of 128 neurons. Lastly, the output of the LSTM is flattened and fed into a fully connected layer of 23 neurons, and softmax is performed to find the prediction.

Performance

This model attained a training accuracy of 100%, validation accuracy of 90% and a testing accuracy of 88%.



(fig 7.2, confusion matrix of 3D CNN+LSTM)

7.4 Stage 2 Combined Evaluation

	Train	Validation	Testing
2D-CNN + LSTM	100	87	84
3D-CNN + LSTM	100	90	87
Dotpos-CNN + LSTM	100	85	88

(Table. 2: Accuracy scores of CNN + LSTM models*)*

Comparing the new models and old models, the new models with the CNN+LSTM architecture has only a slightly higher validation accuracy but a much higher testing accuracy. This shows that the new models are more robust and are better at processing new unseen data.

Dotpos-CNN + LSTM model performs the best in Stage 2, with the highest testing accuracy of 88%. Moreover, Dotpos model has the smallest size among all, with only about 330,000 trainable weights which is several times smaller than that of the raw-pixel-value models. This means the training of Dotpos model is also several times faster than that of raw-pixel-value models. Assuming that we exclude the time required for preprocessing Dotpos dataset, Dotpos model is able to achieve better results with minimal training time, thus, we conclude that Dotpos-CNN + LSTM is our *best-model* in Stage 2.

8. Stage 3: Models using both raw pixels and dotpos data

In Stage 3, we combine 3D-CNN + LSTM and Dotpos-CNN + LSTM into one full model. We experimented two methods of combination: 1. Bilinear 2. Concatenation. [1]

8.1 Bilinear model

Architecture

This model takes in two inputs: the grayscale pixel values and dots' coordinates. The former input is fed into the 3D-CNN + LSTM model (without final Dense Layer), and the latter input is fed into the Dotpos-CNN + LSTM model (without final Dense Layer). The output of the two models are the flattened and concatenated, which is then fed into a fully connected layer of 23 neurons, where softmax is performed to get the prediction probabilities.

Data preprocessing

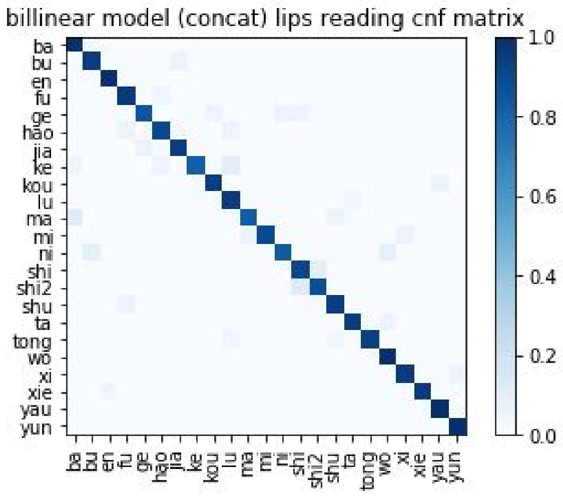
Both dotpos and raw pixel data are needed for this model. Since the estimated time for train the whole model together would need 20 hours per epochs, we trained the 3 parts of the models separately.

Firstly, the two CNN+LSTM models using raw pixels and dotpos data respectively is trained, together with their softmax output player of 23 neurons, is trained separately until they are converged. Then, the output layer of the two models are taken away. The dataset is then fed into the two models (without the output layer) again, creating of new dataset for the final output layer, where each datum is the 10240 neurons outputted from the two CNN+LSTM models. This new set of data is fed into the fully connected output layer, which decodes the 10240 neurons into 23 neurons corresponding to the 23 word classes.

Performance

[1] All models in this stage is newly trained and is not from our previous works.

This model attained a training accuracy of 100%, a validation accuracy of 96% and a testing accuracy of 93%, showing significant improvements from former models. Moreover, as seen in the confusion matrix, while our previous best models shows obvious confusions in some particular words, the model has a similar accuracy for all words.



(fig 8.1, confusion matrix of 3D CNN+LSTM)

8.2 Concatenated model

Architecture

This model is similar to the raw pixel 3D CNN+LSTM model. However, the input is a four dimensional 40x100x100x2 object, where 40 is the time dimension, 100x100 is the spatial dimension, and 2 is the pixel value and the presence of dot respectively, where the former is a float value from 0 to 1 of the grayscale pixel value, and the latter is either a 0 or a 1 (if the dot is located in that coordinate, it is a 1, or else it is a 0). The 4-dimensional data is first fed into a 4D convolutional layer of 128 kernels of size 3x3x3x2, then a max pooling of 1x2x2x1, then another 4D convolutional layer and max pooling with the same hyperparameters. Then, the data is fed into an LSTM of 128 neurons in the main cell (41026048 trainable parameters in total) and lastly flattened and fed into a softmax output layer of 23 neurons.

Data preprocessing

The x and y coordinates of the 20 points of each time frame generated from the dlib shape predictor is converted into a 2D 100x100 array (matching our raw-pixel shape), where the locations of the 20 points are denoted as 1 and all other pixels as 0. Then, the 40 100x100 2D map of the dot positions is concatenated with the 40x100x100 raw pixel data, forming a 4D data of 40x100x100x2.

Performance

This model failed to converge. The validation accuracy of this model fluctuates between 10% and 20% during the training and do not show any improvements throughout the training.

8.3 Evaluation

Model	**Training**	**Validatio n**	**Testing**
Bilinear	100	96	93
Concatenated model	23	10-20	---

(Table 3: Accuracy scores of Bilinear and Concatenated models)

The bilinear model performs significantly better than the previous dotpos-only or raw-pixel-only models. Moreover, as seen in the confusion matrix, all words are predicted relatively accurately in the bilinear model, compared to having some specific words performing slightly worse in the previous dotpos-only or raw-pixel-only models. This might be due to the two types of data compensating each other for the words the other type of data does not do well in.

The concatenated model performed poorly. This is an unexpected result as we initially hypothesize that it should at least perform as well as 3D CNN+LSTM model, considering their similar structures. The poor performance might be resulted from the datum having too many parameters and data points, distracting the model from making important connection between certain data points and pixels. Possibly, it might also be caused by having an insufficient number of kernels, as the input in the model is a 4D tensor and might need more kernels to capture the different variations. Further investigation is needed.

In conclusion, the Bilinear model is our final *best-model* with the highest validation and test accuracies among all models experimented. A detailed model architecture may be seen in Appendix.

9. Future works

Due to time constraints, all models were trained only with the original dataset (i.e. without data augmentation). Despite the models' ability to yield representative results for meaningful comparisons and evaluations, we hope to train the models upon our full dataset (with data augmentation), especially for our current *best-model*: Bilinear Model. Models trained upon the full dataset are expected to have better robustness as they are trained with videos of different brightnesses.

Currently, the training for our Bilinear model has to be separated into three individual parts. That is, the two CNN layers and the final dense layer have to be back-propagated individually. This is done to save time as we could reuse the two CNN models trained in the previous stage, and we believe the backpassing of gradients from the final dense layer back to the separated CNN + LSTM layers may be very slow, due to problems like vanishing gradients. However, this is not a good solution in the long run. Training the layers separately is a time consuming process, as fine-tuning has to be done several times. It is also uncertain for the two layers to learn respective feature representations that are suitable for the concatenation in the final dense layer. Therefore, we hope to investigate on the methods to fully combine the CNN layers into one complete model, that can be back-propagated from end-to-end. This will largely benefit our future trainings of the bilinear model.

In this paper, we only selected 23 words for our research. As we've already achieved pleasing results on the 23-word dataset, we hope to expand our word-target to 50 or even 100 most-used Cantonese words, so as to raise the significance of our project, since classification of these words are far from enough if this technology is to be implemented into our lives. Our model is also lacking the ability to output full sentences from a series of input. Thus, we hope to extend our research to sentence-level models, which can receive an arbitrary length of video sequence and output all the respective words in series. This model will be more relevant for the realistic applications of machine lip-reading, such as mute typing or pronunciation correction models.

10. Acknowledgements

This project was guided through technical and academic supports by Dr. N. Wong and The Department of Electrical and Electronic Engineering of The University of Hong Kong.

11. References

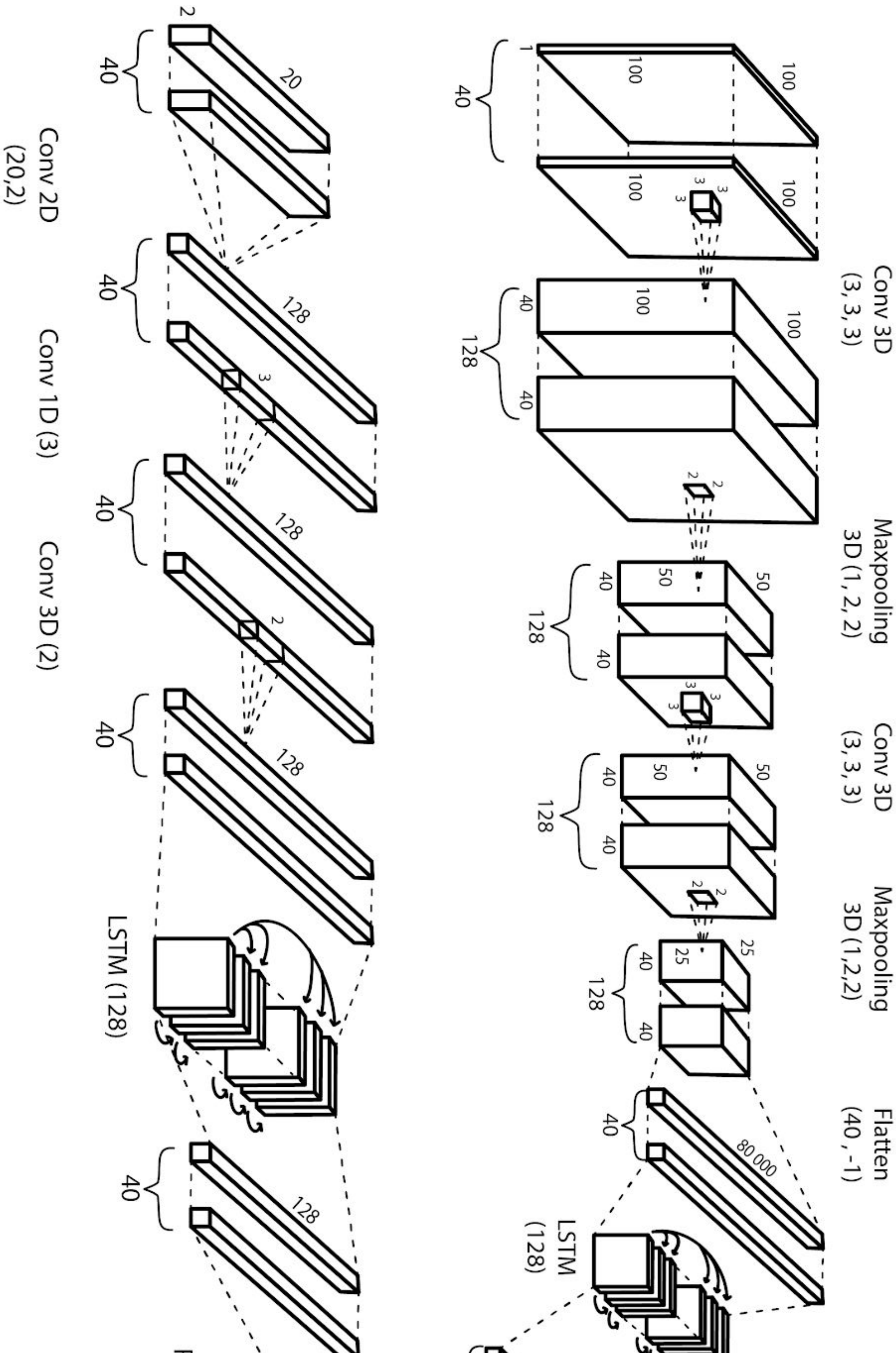
[1] Assael, Y. M., Shillingford, B., Whiteson, S., & De Freitas, N. (2016). Lipnet: End-to-end sentence-level lipreading. arXiv preprint arXiv:1611.01599.

[2] Ayaz A. Shaikh, Dinesh K. Kumar, Wai C. Yau, M. Z. Che Azemin & Jayavardhana Gubbi. Lip Reading Using Optical Flow and Support Vector Machines. In 2010 3rd International Congress On Image And Signal Processing (CSIP 2010)

[3] Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015, April). Convolutional, long short-term memory, fully connected deep neural networks. In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 4580-4584). IEEE.

[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*(pp. 5998-6008).

Appendix


Conv 3D (3, 3, 3)
Maxpooling 3D (1, 2, 2)
Conv 3D (3, 3, 3)
Maxpooling 3D (1,2,2)
Flatten (40 , -1)
LSTM (128)
Conv 2D (20,2)
Conv 1D (3)
Conv 3D (2)
LSTM (128)
80 000
128
40
100
50
25
20