Research Article

Indonesian Lip-Reading Detection and Recognition Based on Lip Shape Using Face Mesh and Long-Term Recurrent Convolutional Network

Aripin and Abas Setiawan

1Department of Biomedical Engineering, Universitas Dian Nuswantoro Semarang, Semarang, Indonesia
2Department of Computer Science, Universitas Negeri Semarang, Semarang 50229, Indonesia

Correspondence should be addressed to Aripin; arifin@dsn.dinus.ac.id

Received 29 November 2023; Revised 1 April 2024; Accepted 9 April 2024; Published 18 April 2024

Copyright © 2024 Aripin and Abas Setiawan. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Communication through speech can be hindered by environmental noise, prompting the need for alternative methods such as lip reading, which bypasses auditory challenges. However, the accurate interpretation of lip movements is impeded by the uniqueness of individual lip shapes, necessitating detailed analysis. In addition, the development of an Indonesian dataset addresses the lack of diversity in existing datasets, predominantly in English, fostering more inclusive research. This study proposes an enhanced lip-reading system trained using the long-term recurrent convolutional network (LRCN) considering eight different types of lip shapes. MediaPipe Face Mesh precisely detects lip landmarks, enabling the LRCN model to recognize Indonesian utterances. Experimental results demonstrate the effectiveness of the approach, with the LRCN model with three convolutional layers (LRCN-3Conv) achieving 95.42% accuracy for word test data and 95.63% for phrases, outperforming the convolutional long short-term memory (Conv-LSTM) method. The proposed approach outperforms Conv-LSTM in terms of accuracy. Furthermore, the evaluation of the original MIRACL-VC1 dataset also produced a best accuracy of 90.67% on LRCN-3Conv compared to previous studies in the word-labeled class. The success is attributed to MediaPipe Face Mesh detection, which facilitates the accurate detection of the lip region. Leveraging advanced deep learning techniques and precise landmark detection, these findings promise improved communication accessibility for individuals facing auditory challenges.

1. Introduction

Speech is the most fundamental type of human communication that uses both visual and auditory elements. Vocalizations in the audio signal are represented by lip movements in speech. Although audio signals typically do a good job of conveying information, lip reading could be necessary in some circumstances, particularly in noisy areas where audio understanding might be compromised. Lip reading can interpret speech based only on visual cues and has recently attracted significant attention due to its possible uses in language identification [1], emotion recognition [2], and human-computer interaction [3].

Lip-reading applications that only identify lip movements are considered more respectful of individual privacy by not including speech during communication [4]. The impact may reduce concerns about the misuse of speech datasets. In addition to that, it is also very useable for deaf people during communication [5]. However, it may be difficult to interpret lip movements effectively, especially when there are similarities in the forms of lips of distinct words or when there are outside influences such as background noise [6].

In this regard, advances in technology present viable ways to interpret lips using visual information recorded by a camera [7, 8]. The goal of conventional machine learning methods for lip reading is to identify temporal patterns in data streams. Many researchers employ deep learning models for lip reading in addition to advancing machine learning into deep learning [9]. The key to achieving
a successful recognition system in lip reading is the precise
detection of the lip region and the subsequent classification of
utterances, a task influenced by factors such as language,
dialect, and individual lip structure. Although some studies
have developed language-specific lip-reading systems tai-
lored to distinct regions [7, 10–14], the diversity of lip shapes
and external noise poses challenges to effective detection and
classification algorithms.

This study addresses these challenges by proposing an
Indonesian lip-reading dataset, as well as a detection and
recognition system based on the diversity of lip shapes
considering the unique characteristics of eight types of lip
shapes [15]. The MediaPipe face mesh [16] was used for the
detection and elimination of surrounding noise, and the
long-term recurrent convolutional network (LRCN) [17]
for the classification of utterances.

The structure of this paper is organized as follows.
Section 1 provides an introduction to the background and
challenges of lip reading. Section 2 reviews related works and
highlights the contributions of our research. Section 3 de-
scribes materials and methods, including data acquisition,
detection algorithms, preprocessing techniques, and model
development. Section 4 presents the experimental results
and a discussion of various scenarios. Finally, Section 5
concludes our findings and discusses future directions.

2. Related Works

Lip-reading applications utilize image processing, machine
learning, and deep learning to understand spoken words
through lip movements. An eigenlip model has been pro-
posed that calculates the Euclidean distance between the
upper and lower lips, along with the hidden Markov model
(HMM), for word prediction [18]. In addition, neural
network models have been developed for the classification of
laughter speech, using limited audiovisual mapping [19]. Lin
et al. achieved an accuracy rate of 80% in predicting vowel
utterances [20], while bidirectional long short-term memory
(Bi-LSTM) models were used for visual speech recognition
[21]. However, distinguishing silent speech from whispered
speech remains a challenge. The bidirectional gated re-
current unit (Bi-GRU) extracts features for audiovisual
recognition but struggles in noisy environments [22].
Convolutional neural networks (CNNs) with the pretrained
VGG-16 model [23] and LSTM combinations with a histo-
gram of oriented gradients and a support vector machine
(HOG + SVM) have been proposed for spoken word
recognition [23].

Recent advancements in lip reading involve deep
learning algorithms, such as convolutional neural networks
(CNNs). Martinez et al. improved word-by-word lip reading
using multiscale temporal convolutional networks
(MSTCNs) [24]. Kounparoulos and Potamianos introduced
efficient networks for lip reading, achieving high accuracy
levels in the lip-reading in the wild (LRW) dataset [25].
Visual speech recognition (VSR) models have also been
developed, surpassing previous methods in accuracy [26].
However, these studies predominantly focus on English-
language datasets.

Language-specific datasets are crucial for accurate lip
reading. Recent efforts include German, Mandarin, Turkish,
and Indonesian lip-reading systems. German lip-reading
system achieved an accuracy rate of 88% [27], while the
Mandarin system reached 61.18% accuracy using 3D-CNN
with DenseNet + LSTM model [28]. Atila and Sabaz de-
veloped a Turkish lip-reading system with a Bi-LSTM model
that achieves 85% accuracy for words and 91% for sentences
[29]. Indonesian lip-reading system, although limited to
only 50 sentences, reached an accuracy rate of 80% for
syllable classification using 3D-CNN and Bi-GRU
models [30].

Along with the related work, the lack of Indonesian
lip-reading datasets, especially for word and phrase levels,
encourages this study to be able to make datasets open
publicly available datasets for researchers. The new dataset
consisted of 10 words and four phrases considering eight
different lip shapes. To improve the detection and clas-
sification accuracy, this study also proposed state-of-
the-art detection algorithms and deep learning models to
close the gap. The MIRACL-VC1 dataset [7] with word
samples is also considered to test our algorithm frame-
work. In summary, this research has the following
contributions:

(1) This study presents the first open-ended dataset for
lip reading called IndoLR with an Indonesian lan-
guage data sample consisting of several words and
phrases, considering eight different types of lip
shapes.

(2) The MediaPipe Face Mesh [16] is used to obtain lip
ROL, which is then trained with the long-term re-
current convolutional network (LRCN) in the In-
donesian lip-reading dataset, which produces an
accuracy of more than 94% compared to the con-
volutional LSTM (Conv-LSTM) model.

(3) The proposed framework has also been applied to an
available public dataset called MIRACL-VC1 [31],
achieving an accuracy of 90.67% and an F1 score of 91
with the best LRCN model in the word-labeled class.
This performance showed a good result compared to
previous studies.

A preprint has previously been published and has not yet
been peer-reviewed [32]. The updated work used MediaPipe
Face Mesh to detect the lip and the LRCN architecture.
Subsequently, the proposed method was also evaluated in
the MIRACL-VC1 dataset [7] in this study.

3. Materials and Methods

In general, the proposed system is presented in Figure 1.
First, the data acquisition is carried out to collect the
isolated video data. Every video will be captured in a close-
up fashion (frontal). Second, lip detection is performed by
using MediaPipe Face Mesh [16]. Third, the video is
preprocessed by extracting the videos into image frames
for the training process. Fourth, building the LRCN model
and training were conducted to recognize the utterance
visually.
3.1. Data Acquisition. There are different types of human lips, so it is necessary to have a reference for the types of lip shapes common to humans. Reference to the shape of the lip from [15] has been adapted as shown in Figure 2. From the types of lip shapes, it is hoped that it can represent the overall shape of the human lips. Five women and three men participated in the production of these data. Each person represents each type of suitable lip shape. These types of lips are neutral, pointly neutral, thin, cupid’s bow, uni-lip, beestung, smear, and glamour.

Each person speaks ten words and four phrases. The ten words are “maaf” (sorry), “tolong” (please), “permisi” (excuse me), “halo” (hello), “mulai” (start), “berhenti” (stop), “lanjut” (next), “sakit” (hurt), “kembali” (back), and “awas” (be careful). Meanwhile, the four phrases spoken are “terima kasih” (thank you), “minta tolong” (please help), “saya minta maaf” (I am sorry), and “saya minta tolong” (I am asking for help). These words and phrases were chosen because Indonesians often use them. These words and phrases were chosen because they are often used in Indonesian language communication and reflect politeness.

Every word or phrase is recorded using a Logitech C525 camera with an 8-megapixel resolution and a standard PC to process the recording. The video captured is in MP4 format with a resolution of 480p (640 × 480) with a total frame rate of 30 FPS for the ten words and four phrases collected. The different settings were made due to the limitation of the machine to process each video. For every word sample, it takes 30 videos per person. Thus, the total data collected for the word dataset is 2400. In the phrases dataset only contains four phrases category, then the additional samples are gathered to 50 videos for each person. The total data collected for the phrase dataset are 1600. All these collected video samples are then called IndoLR (Indonesian lip-reading dataset).

The study of lip-reading was developed not only in one language. Each language has a different way of pronouncing the other, leading to further variations. So, some countries build their datasets, as shown in Table 1. The dataset taken from this investigation is also compared with another available dataset. Compared to some publicly available (or with limited access) datasets, IndoLR is the only publicly available dataset with the most data in Indonesia. In the research by Kurniawan and Suyanto [30], there are very few data samples due to the focus of the classification on syllables. In addition, the resolution provided in our dataset is also quite large compared to other studies. Although the number of data samples is not as large as in most recent studies (LRW [10], LRS2 [33], LRS3-TED [11], GLips [13], Turkish [29], CMLR [34], CN-CVS/Speech [35], and OLKAVS [14]), this study considers the shape of the lip type depicted in Figure 2. All speakers in IndoLR have sample representations of the previously mentioned lip-type shapes, with each type represented by one speaker.

In this study, the MIRACL-VC1 dataset [7] with word samples is considered to test our algorithm framework. MIRACL-VC1 is an openly available dataset with two sample types: color and depth. In this study, the total number of word data is 1500 utterances with word labels such as begin, choose, connection, navigation, next, previous, start, stop, hello, and web. Several researchers have also benchmarked the dataset to compare it with lip-reading studies.

3.2. Detection and Preprocessing. Detecting the position of the lip on the face of a person using computer technology is not easy. This difficulty occurs because the human lip is a small part of the human face that is considered to resemble the eyes and nose. There are many ways to detect faces, such as traditional machine learning [36] and deep learning [16, 37], which can detect human faces effectively. This study tried to use one of the most effective methods, the Haar cascade, HOG-SVM, or MediaPipe, to recognize the lip. In early-stage experiments, using the Haar cascade method, the intention was to detect the lip but sometimes not only that.
region but also small objects such as eyes, nose, and neck folds. Haar cascades can recognize faces effectively, but small things, such as the lip, are difficult to detect. Error detection was also proven in the study by [38], where by using the Haar cascade method, it was difficult to find and obtain ROI from the eyes. However, when it is collected in the dataset, it will cause noise.

There are other methods for detecting the lip more accurately: King [36] and MediaPipe [16]. These two methods can provide information on lip landmarks taken from facial landmarks. The Dlib uses the HOG-SVM algorithm to provide 68 landmark points in the facial image. In addition, the MediaPipe Face Mesh can estimate 468 3D landmark points on the face. Moreover, MediaPipe performance is better than Dlib when it detects local or small features of the face, including the lip. MediaPipe is also faster than Dlib in detecting the landmark of a facial image [39]. Moreover, in the study by Ishmam et al. [40], MediaPipe has better performance than Dlib in the isolation of lip from various face conditions such as angle, appearance, and lighting. In this case, the MediaPipe Face Mesh was considered as a method to detect the lip region.

The MediaPipe Face Mesh can track the lips and details of the tongue, teeth, and gums. The final image was cropped only for the lip region because there are some noises such as whiskers, chin, beaver, and nose which are close to the lip. There are three steps to detect the lip. The first is collecting the 40 landmark points from the 68 facial landmarks. Every landmark point LP_{x,y} has the x and y positions in 2D space. Unfortunately, the detected landmark points are not ordered and must be ordered. The second is finding the coordinate points within the dimension of the image. The calculation of the relative source point is RPs_{x,y} = LPs_{x,y} \cdot Imgwidth_{x,y} \cdot Imgheight_{y}, where the LPs_{x} and LPs_{y} are the landmark source point as well as Imgwidth and Imgheight are the width and height of the image, respectively. Subsequently, a similar calculation is also measured for the relative target point RPt_{x,y} = LPt_{x,y} \cdot Imgwidth_{x,y} \cdot Imgheight_{y}. Thus, the routes between the source and the target point can be stored to find the edge of the lip. The third step is to extract the region of interest on the lip by creating a boundary box around the border. The boundary box can be calculated by using the
minimum and maximum indexes of the route to mask the lip. Since it is spatially impossible for a lip region image to produce the exact width and height dimensions, the gaps will be filled with black color.

The preprocessings are performed before the data enter as input to the network. First, each frame image is resized to 80 × 80. Second, the sequence length is determined. The sequence length determined for the word dataset is 30 frames, while for the phrase dataset, it is 40 frames. The image frames are not taken from frame index 0 but from the middle. Clipping of frame images from the middle index in isolated videos is carried out by considering the presence of stillness at the beginning of the video and at the end of the video, which can cause bias in the training process. If the number of frames in a video is more than the sequence length, then the silence at the beginning and end can be eliminated so that the focus is on the situation where the lips move to speak. Meanwhile, if the speaker speaks too fast, the number of frames will be less than the sequence length, resulting in a lack of image samples. This can be circumvented by adding an image that contains a fully black-colored or a black-padded image. Applying black-padded images as blank images preserves the temporal structure of the original sequence of frames and prevents information loss during training. It also ensures that all sequences are of the same length, maintaining uniformity in sequence processing. This straightforward process does not require complex processing steps (which is possibly computationally inefficient), making it accessible to operate in a fixed-length sequence when trained later on. Figure 3 depicts an illustration of the frames with the clipping sequence in the middle. It is hoped that this strategy can focus more data on the situation when the lips are speaking and will not be too affected by the speed of speech. Third, pixel frame normalization is performed to reduce the computation. Normalization produces a range of 0–1, dividing each value of pixels by 255.

After the data were preprocessed, it was split into three parts, training set, validation set, and test set for words and phrases. The compositions for each part are 80:10:10 percent. Every single class in the word or phrase datasets for each part contains every person sample. This is necessary so that the data can be distributed evenly.

3.3. Building the LRCN Model. Machine learning and deep learning are suitable methods that can be used as modern lip-reading techniques. In this study, the long-term recurrent convolutional network (LRCN) was used to train data on lip reading in the Indonesian lip-reading dataset. LRCN has been used successfully in action recognition, where each frame-frame video sequence used as a network input can be appropriately identified with the output activity associated with the video [17]. Related to this, we used LRCN to recognize what words or phrases are spoken, obtained from frame-to-frame sequence data from an uttered speech video. In LRCN, CNN and LSTM layers are combined in a single model. In this case, CNN will act as a spatial feature extraction from the frame, which will then be fed to the LSTM at each time step for temporal sequence modeling. Thus, direct training can be conducted to study spatiotemporal features end-to-end by producing a robust model.

Previously, the video data have been transformed into a sequence of image frames containing the lip area by cropping based on landmark detection using MediaPipe. After that, the image frames will be preprocessed to be a ready-to-train dataset. Every sequence of image samples with it is labeled and then collected into the words and phrases datasets. The LRCN model aims to bring those sequential inputs to static outputs that represent the word or phrase label \((x_1, x_2, x_3, \ldots, x_T, y) \rightarrow y\). Any data that have gone through preprocessing at a specific time frame \(x_t\) will be trained up to the length of the \(T\) frame of the time sequence, which is then considered as input. The static output is a single \(y\) label that contains the word class \(y \in \mathbb{R}^3\) or the sentence \(y \in \mathbb{R}^4\).

Each input \(x_t\), trained in CNN with three convolution layers, max pooling, dropout, and flatten, is wrapped by a distributed temporal layer. The isolation of each sequence of video frames is passed through the feature transformation \(\phi_{LSTM}(x_t)\). The details of the first convolution layer have 16 feature maps with 3 × 3 kernels and rectified linear unit (ReLU) activation functions [41, 42]. The ReLU function can produce nonlinear constraints on the input \(\max(0, f_z(x_t, W))\), where \(f_z(\cdot)\) is a linear function resulting from the input \(x_t\), and the weight parameter is \(W\). The first layer of the pool is the max pool with 2 × 2, followed by a dropout with a dropout rate of 0.25. The second convolution layer has 32 feature maps with configurations and is accompanied by the same pooling and dropout layers as the first. Likewise, for the third layer, it is also the same. The only difference is the number of feature maps in the third convolution layer, that is, 64. Subsequently, before entering the LSTM layer, there is a flatten layer to convert the spatial output values into vectors.

Then, there is one layer of LSTM with a total of 16 cells. Usually, in its general form, the LSTM model has a weight parameter \(W\) by mapping the input \(x_t\) and the previous time-step hidden state \(h_{t-1}\) with two outputs, namely, the nonlinear calculation output \(z_t\) and the updated hidden state \(h_t\). In Figure 1, the LSTM sequence learning will be carried out by passing the \(h_{t-1}\) output to \(h_t\). We calculate the first hidden layer sequence \(h_1 = f_w(x_1, 0)\), where \(h_0 = 0\) because there is no previous hidden layer output. Then, we calculate the second hidden layer sequence \(h_2 = f_w(x_2, h_1)\) and so on, so the hidden layer output at the current time step is \(h_t = f_w(x_t, h_{t-1})\). On LSTM to produce \(h_t\), it needs to calculate the following:

\[
i_t = \sigma(W_{wx}x_t + W_{hh}h_{t-1} + b_t),
\]

\[
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f),
\]

\[
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o),
\]

\[
g_t = \tan h(W_{xc}x_t + W_{hc}h_{t-1} + b_t),
\]
LSTM consists of an input gate $i_t \in \mathbb{R}^N$, forget gate $f_t \in \mathbb{R}^N$, output gate $o_t \in \mathbb{R}^N$, input modulation gate $g_t \in \mathbb{R}^N$, and the memory cell $c_t \in \mathbb{R}^N$, where $N$ is the hidden units. The input gate, the forget gate, and the output gate use the nonlinear sigmoid nonlinear function $\sigma(x) = (1 + e^{-x})^{-1}$. Meanwhile, in the modulation gate input and updated hidden state calculations, there is a nonlinear function hyperbolic tangent $\tanh(x) = e^x - e^{-x} / e^x + e^{-x}$. The $\odot$ operator symbol is the elementwise product of two vectors. Then, the dropout was performed [43] to minimize overfitting gaps that may occur during training. To classify the distribution of $P(y_t)$ in the output layer to the desired label results ($D \in \mathbb{R}^3$ on word labels and $D \in \mathbb{R}^4$ in phrase labels), many classes are classified, and the predicted distribution $P(y_t = d)$ uses the softmax function as

$$P(y_t = d) = \text{softmax}(\tilde{y}_t) = \frac{\exp(\tilde{y}_t, d)}{\sum_{d' \in D} \exp(\tilde{y}_t, d')}$$

(7)

The number of units in the output layer corresponds to the number of word and phrase labels in the two datasets. Therefore, there will be two LRCN architectures that have a different number of output layers. The word dataset has three units in the output layer, whereas the phrase dataset has four. The loss function used to evaluate this network is the categorical cross entropy $L$ with the calculation as follows:

$$L = - \sum_{i=1}^{M} \sum_{d=1}^{D} (y_{id} \ast \log(P(y_t = d)))$$

(8)

where $M$ is the number of data samples and $y_{id}$ is the output corresponding to the current data label. Adam optimizer [44] updates the weight parameter that stores the classification pattern. The standard neural network training cycle is used to perform forward propagation, calculate the loss function and backpropagation over time, and update the weight parameters.

4. Results and Discussion

The deep learning model used to test the IndoLR and MIRACL-VC1 dataset is not only compared with LRCN but also compared with convolutional LSTM (Conv-LSTM) [45]. Table 2 shows the architectural details of the three neural network models for the experimental scenarios. The three architectures are compared using the softmax activation function in the output layer and the Adam optimizer. The testing will be carried out on a test set that previously went through the same data acquisition process as the training set. The machine used to carry out the training is a consumer-grade CPU with an Intel i5 10th-gen processor with an RTX 3060 Ti GPU and 32 GB RAM.

Every video sample in the IndoLR dataset with the three applied network architecture scenarios was trained in 100 epochs. The hyperparameter settings for the overall experiments are the learning rate of 0.0005 and the batch size of 4. Since the training set is not large and there is a limitation on the consumer-grade computer to perform the training...
In the process, the batch size of 4 was chosen. In Conv-LSTM scenarios, max pooling 3D is added to reduce the complexity of the model. The number of convolutional layers in LRCN is limited to up to three layers. The consumer-grade GPU has 8–32 GB memory, but, in this case, it only used 8 GB on RTX 3060 Ti.

In terms of time complexity, LRCN is based on the operations performed in its convolutional and recurrent layers during both training and inference. Convolutional layers typically have a time complexity of \( O(N^2 \times C^2 \times X_{\text{in}} \times X_{\text{out}}) \), where \( N \times N \) is the image dimension, \( C \) is the size of the convolutional kernel, \( X_{\text{in}} \) is the number of input channels, and \( X_{\text{out}} \) is the number of output channels \([46]\). In the recurrent layer using the LSTM, it has a time complexity of \( O(M^2) \). The overall time complexity of Conv-LSTM for processing a sequence of length \( T \) is represented as \( O(T \times (N^2 \times C^2 \times X_{\text{in}} \times X_{\text{out}} + M^2)) \). As both architectures share similarities due to their integration of convolutional and recurrent layers, Conv-LSTM focuses on capturing spatial and temporal dependencies simultaneously within each time step, while LRCN typically processes sequences through separate convolutional and recurrent stages. Therefore, the exact time complexity may vary depending on factors, namely, the design of architecture, the characteristics of data, and implementation details.

Of the three architectures, Conv-LSTM requires a longer training time than the other two architectures. Conv-LSTM uses a special architecture to combine CNN and LSTM in recurrent steps. In LRCN, there is a TimeDistributed layer performed on every time slice for a warped certain layer. No recurrence process is going on in a TimeDistributed layer. Overfitting occurs in the word dataset with Conv-LSTM, where there is a significant gap between the accuracy of the

<table>
<thead>
<tr>
<th>Model</th>
<th>Words</th>
<th>Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv-LSTM</td>
<td>ConvLSTM2D (8)</td>
<td>ConvLSTM2D (8)</td>
</tr>
<tr>
<td></td>
<td>MaxPooling3D ()</td>
<td>MaxPooling3D ()</td>
</tr>
<tr>
<td></td>
<td>ConvLSTM2D (16)</td>
<td>ConvLSTM2D (16)</td>
</tr>
<tr>
<td></td>
<td>MaxPooling3D ()</td>
<td>MaxPooling3D ()</td>
</tr>
<tr>
<td></td>
<td>Flatten ()</td>
<td>Flatten ()</td>
</tr>
<tr>
<td></td>
<td>Dense (10)</td>
<td>Dense (10)</td>
</tr>
<tr>
<td>LRCN-2Conv</td>
<td>TimeDistributed (Conv2D (16))</td>
<td>TimeDistributed (Conv2D (16))</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (MaxPooling2D ())</td>
<td>TimeDistributed (MaxPooling2D ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Dropout ())</td>
<td>TimeDistributed (Dropout ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Conv2D (32))</td>
<td>TimeDistributed (Conv2D (32))</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (MaxPooling2D ())</td>
<td>TimeDistributed (MaxPooling2D ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Dropout ())</td>
<td>TimeDistributed (Dropout ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Flatten ())</td>
<td>TimeDistributed (Flatten ())</td>
</tr>
<tr>
<td></td>
<td>LSTM (64)</td>
<td>LSTM (64)</td>
</tr>
<tr>
<td></td>
<td>Dropout ()</td>
<td>Dropout ()</td>
</tr>
<tr>
<td></td>
<td>Dense (10)</td>
<td>Dense (10)</td>
</tr>
<tr>
<td>LRCN-3Conv</td>
<td>TimeDistributed (Conv2D (16))</td>
<td>TimeDistributed (Conv2D (16))</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (MaxPooling2D ())</td>
<td>TimeDistributed (MaxPooling2D ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Dropout ())</td>
<td>TimeDistributed (Dropout ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Conv2D (32))</td>
<td>TimeDistributed (Conv2D (32))</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (MaxPooling2D ())</td>
<td>TimeDistributed (MaxPooling2D ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Dropout ())</td>
<td>TimeDistributed (Dropout ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Conv2D (64))</td>
<td>TimeDistributed (Conv2D (64))</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (MaxPooling2D ())</td>
<td>TimeDistributed (MaxPooling2D ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Dropout ())</td>
<td>TimeDistributed (Dropout ())</td>
</tr>
<tr>
<td></td>
<td>TimeDistributed (Flatten ())</td>
<td>TimeDistributed (Flatten ())</td>
</tr>
<tr>
<td></td>
<td>LSTM (64)</td>
<td>LSTM (64)</td>
</tr>
<tr>
<td></td>
<td>Dropout ()</td>
<td>Dropout ()</td>
</tr>
<tr>
<td></td>
<td>Dense (10)</td>
<td>Dense (10)</td>
</tr>
</tbody>
</table>

Unlike LRCN, Conv-LSTM time complexity is determined by the operations performed within its convolutional and recurrent layers. At each time step, Conv-LSTM involves convolutional operations followed by recurrent operations. The time complexity of the convolutional layers in Conv-LSTM is the same as that of LRCN. Recurrent layers within Conv-LSTM must be LSTM units with the time complexity of \( O(M^2) \). The overall time complexity of Conv-LSTM for processing a sequence of length \( T \) is represented as \( O(T \times (N^2 \times C^2 \times X_{\text{in}} \times X_{\text{out}} + M^2)) \). As both architectures share similarities due to their integration of convolutional and recurrent layers, Conv-LSTM focuses on capturing spatial and temporal dependencies simultaneously within each time step, while LRCN typically processes sequences through separate convolutional and recurrent stages. Therefore, the exact time complexity may vary depending on factors, namely, the design of architecture, the characteristics of data, and implementation details.

Of the three architectures, Conv-LSTM requires a longer training time than the other two architectures. Conv-LSTM uses a special architecture to combine CNN and LSTM in recurrent steps. In LRCN, there is a TimeDistributed layer performed on every time slice for a warped certain layer. No recurrence process is going on in a TimeDistributed layer. Overfitting occurs in the word dataset with Conv-LSTM, where there is a significant gap between the accuracy of the
training data and the accuracy of the validation data, as shown in Figure 4(a). Meanwhile, the two LRCN architectures, namely, LRCN-2Conv and LRCN-3Conv, look more stable compared to Conv-LSTM. These models are shown in Figure 4(b) for the LRCN-2Conv model and Figure 4(c) for the LRCN-3Conv model. Furthermore, the training and validation data accuracy gap of the LRCN-3Conv model is smaller than that of the LRCN-2Conv model. The result shows that the accuracy of the LRCN-3Conv model is more stable than that of the LRCN-2Conv model.

Overfitting also occurs in the phrase dataset starting at the 10th epoch in the Conv-LSTM architecture. There is a significant gap between the accuracy of the training data and the validation data for the Conv-LSTM model shown in Figure 5(a). The gap between the accuracy of the training data and the validation data in the LRCN model, namely, LRCN-2Conv and LRCN-3Conv, appears to be more stable, as shown in Figures 5(b) and 5(c). In general, the gap between the validation accuracy of the training set and the validation set in the word dataset is better than the gap between the accuracy of the training data and the validation data in the phrase dataset, as shown in Figures 4 and 5.

When applied to the test set, the performance results of the deep learning models are shown in Table 3. The two LRCN models produce higher accuracy than Conv-LSTM, with a difference of 2.5–5% for the word dataset and 4.38–5.01% for the phrase dataset. LRCN achieved the highest accuracy with three convolution layers in the word and phrase datasets (LRCN-3Conv). The training time of the Conv-LSTM model is longer than that of the LRCN. It is around 10 times longer. It is because of the involvement of convolutional and recurrent operations at each time step rather than passing the convolution operation first and then followed by the recurrent operation. It also affects the recognition time for each video sample, where it needs a longer time than LRCN even in real-time situations. The best accuracy was achieved by LRCN with three convolution layers in both the word and phrase datasets better than LRCN with two convolutional layers and Conv-LSTM.

The receiver operating characteristic (ROC) and area under the ROC curve (AUC) were provided to evaluate the performance of the model. Due to the unknown cost of misclassification and class distribution during the training phase, this statistical metric is preferred. In this case, the ROC and AUC are applied to the multiclass classification problem. Therefore, the one-vs-rest (OvR) approach was used to distinguish between one class and other classes. The evaluation of ROC and AUC for each model scenario is presented in Figures 6 and 7 for the datasets of words and phrases, respectively. In either the word or phrase dataset, the model scenarios show a good performance of AUC, which is close to 1. There are no classes in all scenarios near 0.5, which means poor separability between classes. However, the LRCN performs better on separability than Conv-LSTM in both word and phrase datasets. After looking at the AUC result of the LRCN-3Conv in a word dataset, there is an interesting finding that in LRCN-3Conv, the words “permisi” and “berhenti” are successfully distinguished, although in the first sequence of image frames, it has a similar vowel sound of “p” and “b.” By conducting full training on the word instead of per syllable, the algorithm focuses not only on a certain frame but on the overall sequence of frames.

The performance of the Conv-LSTM, LRCN-2Conv, and LRCN-3Conv models was also evaluated using precision, recall, and F1 score. Formulas to calculate precision, recall, and F1 score are presented in equations (9)–(11). Finally, precision, recall, and F1 score measurements are applied to the word and phrase datasets. A test result that accurately detects the existence of a condition is called a true positive (TP). True negative (TN) is an alternative term for a test result that correctly foretells the absence of a circumstance. A test result that falsely suggests the presence of an attribute is known as a false positive (FP). The result of a test that incorrectly implies that a particular circumstance is not present is called a false negative (FN).

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{10}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{9}
\]

\[
F1 \text{ score} = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{11}
\]

Precision is used to measure the level of accuracy between the actual value and the predicted value. Then, recall aims to calculate the ratio between TP and TP + FN. Meanwhile, the F1 score is used to calculate the average precision and recall. These additional performance calculations ensure that the model has feasible accuracy and sensitivity and are presented in Table 4 for the word dataset and Table 5 for the phrase dataset.

Table 5 shows that the average F1 score values for the LRCN-3Conv, LRCN-2Conv, and Conv-LSTM models are 91%, 95%, and 96%, respectively. The LRCN-3Conv model performs better than the LRCN-2Conv and Conv-LSTM models. Based on the comparison of Tables 4 and 5, in general, the model performs better on the phrase dataset than on the word dataset. The number of classes is taking a role because, in the phrase dataset, only four classes are compared with 10 classes in the word dataset.

The algorithm method proposed in this study is also applied to the MIRACL-VC1 public dataset only with the word-labeled data. The preprocessing stage and the LRCN use the same approach as applied in IndoLR. The MIRACL-VC1 dataset has fewer images for each class, as it is captured at 15 frames per second with a sequence length range of 4–27 image frames. In the experiments carried out, the length of the sequence frame determined is 12 frames.

ROC and AUC were also evaluated for each class in the MIRACL-VC1 dataset with two different LRCN models as shown in Figure 8. There is no significant difference between LRCN with two convolutional layers and LRCN with three convolutional layers. However again, the increased number of convolutional layers in LRCN has better separability as proved by the better result of AUC in most labels of classes. The accuracy performance results
Figure 4: Results of the training accuracy of the word dataset. (a) Conv-LSTM model. (b) LRCN-2Conv model. (c) LRCN-3Conv model.

Figure 5: Training accuracy result of the phrase dataset. (a) Conv-LSTM model. (b) LRCN-2Conv model. (c) LRCN-3Conv model.

Table 3: Performance results of IndoLR.

<table>
<thead>
<tr>
<th>Model</th>
<th>Val. acc. (%)</th>
<th>Test acc. (%)</th>
<th>Tt (s)</th>
<th>Rt (ms)</th>
<th>Val. acc. (%)</th>
<th>Test acc. (%)</th>
<th>Tt (s)</th>
<th>Rt (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conv-LSTM</td>
<td>94.7</td>
<td>90.42</td>
<td>7996.6</td>
<td>±88</td>
<td>95</td>
<td>90.62</td>
<td>9509.8</td>
<td>±105</td>
</tr>
<tr>
<td>LRCN-2Conv</td>
<td>95.83</td>
<td>92.92</td>
<td>600.3</td>
<td>±64</td>
<td>99.37</td>
<td>95.00</td>
<td>539.6</td>
<td>±68</td>
</tr>
<tr>
<td>LRCN-3Conv</td>
<td>97.92</td>
<td>95.42</td>
<td>727.3</td>
<td>±62</td>
<td>99.37</td>
<td>95.63</td>
<td>585.9</td>
<td>±66</td>
</tr>
</tbody>
</table>

*Tt, training time in seconds; Rt, average recognition time for each video sample in milliseconds.

Figure 6: The ROC and AUC for each class in the word dataset. (a) Conv-LSTM model. (b) LRCN-2Conv model. (c) LRCN-3Conv model.
obtained are also excellent compared to previous studies, which can be seen in Table 6. The detection method with MediaPipe Face Mesh is also the only one used to compare it with other studies. Detection with this method proved to be more robust, as it can more effectively and efficiently localize the lips compared to HOG+SVM and Dlib. The consistency of the results between accuracy and the $F_1$ score is also not far away. For the original dataset, MediaPipe+LRCN with 3 CNN layers has superior results (87%) compared to Inception V3 (86.6%) [48], CNN (52.9%) [48], VGG-16+LSTM [47] (59%), 3D-CNN [51] (70.2), and 3D-CNN+LSTM [52] (85%).

The MobileNet + LSTM and VGG-16+LSTM architectures [50] have an accuracy of more than 90%, which exceeds this study in the modified MIRACL-VC1 dataset. Modified means that the MIRACL-VC1 dataset is producing a new dataset which is similar to MIRACL-VC1. This is performed because the original MIRACL-VC1 has a lot of noise, such as part of the nose detected as a background, which can interfere with the training process. However, the value of the $F_1$ score in this study, compared to the results of the MobileNet+LSTM $F_1$ score, is 3% higher than the original MIRACL-VC1 [7]. This achievement is also inseparable from using MediaPipe Face Mesh to detect lips very well to avoid false information. Moreover, LRCN with three convolutional layers also gave

---

### Table 4: Results of the performance evaluation (precision, recall, and $F_1$ score) on the word dataset.

<table>
<thead>
<tr>
<th>Words</th>
<th>Conv-LSTM</th>
<th>LRCN-2Conv</th>
<th>LRCN-3Conv</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision (%)</td>
<td>Recall (%)</td>
<td>$F_1$ score (%)</td>
</tr>
<tr>
<td>Maaf</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>Tolong</td>
<td>85</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Permisi</td>
<td>96</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>Halo</td>
<td>91</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Mulai</td>
<td>89</td>
<td>71</td>
<td>79</td>
</tr>
<tr>
<td>Berhenti</td>
<td>89</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>Lanjut</td>
<td>95</td>
<td>75</td>
<td>84</td>
</tr>
<tr>
<td>Sakit</td>
<td>92</td>
<td>96</td>
<td>94</td>
</tr>
<tr>
<td>Kembali</td>
<td>88</td>
<td>92</td>
<td>90</td>
</tr>
<tr>
<td>Awas</td>
<td>85</td>
<td>96</td>
<td>90</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>90</td>
<td>93</td>
</tr>
<tr>
<td>Macro avg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>90</td>
<td>93</td>
</tr>
<tr>
<td>Weighted avg</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 5: Results of the performance evaluation (precision, recall, and $F_1$ Score) on the phrase dataset.

<table>
<thead>
<tr>
<th>Phrase</th>
<th>Conv-LSTM</th>
<th>LRCN-2Conv</th>
<th>LRCN-3Conv</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision (%)</td>
<td>Recall (%)</td>
<td>$F_1$ score (%)</td>
</tr>
<tr>
<td>Terima kasih</td>
<td>98</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>Saya minta tolong</td>
<td>86</td>
<td>80</td>
<td>83</td>
</tr>
<tr>
<td>Saya minta maaf</td>
<td>88</td>
<td>90</td>
<td>89</td>
</tr>
<tr>
<td>Minta tolong</td>
<td>90</td>
<td>93</td>
<td>91</td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>Macro avg</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>Weighted avg</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Figure 7: The ROC and AUC for each class in the phrase dataset. (a) Conv-LSTM model. (b) LRCN-2Conv model. (c) LRCN-3Conv model.
a significant result in accuracy. Therefore, the findings of this investigation can be accepted and used as a reference by looking at previous studies. The use of MediaPipe and LRCN is a potentially robust detection algorithm to support the performance of the neural network training model to detect lips correctly.

5. Conclusion

IndoLR has been successfully built for Indonesian lip-reading benchmarking. In this study, the LRCN architecture and MediaPipe Face Mesh have been proposed to recognize lip reading. The performance of the LRCN model has been tested under various conditions. Testing has been carried out on two types of test data, namely, the word and phrase datasets. The experimental results show that the LRCN with three convolutional models produces the highest accuracy in the word dataset and the phrase dataset than the LRCN with two convolutional layers and convolutional LSTM. Adding more convolutional layers can improve the performance of the algorithm.

The average F1 score values of the LRCN-3Conv, LRCN-2Conv, and Conv-LSTM models for the word dataset are 90%, 93%, and 95%, respectively. Meanwhile, the average F1 score values of the LRCN-3Conv, LRCN-2Conv, and Conv-LSTM models for the phrase dataset are 91%, 95%, and 96%, respectively. In addition, testing was also conducted on an open dataset available called MIRACL-VC1 in word-labeled classes. The LRCN with three convolutional layers also outperforms previous studies in the F1 score. The findings of this study show that it is possible to use MediaPipe to get lip ROI without any noise and implement LRCN in the frame-to-frame data. The types of lip shapes are also necessary, which may not be considered in other research studies.

For future work, it will be more robust if the dataset is enriched by involving more people with various lip types, poses, angles, and lighting conditions. The diversity of datasets can support emerging classification algorithms such as transformer or attention-based models because it has a “data-hungry” behavior. The larger data obtained can provide a better model performance. This study also has limitations, which are related to the efficiency of the computational time and memory used without losing the correlation between every frame sequence of the sample video. A more effective and efficient method is still needed, which can make lip-reading recognition perform better in real-world applications.

Data Availability

The dataset in MP4 format used to support the findings is deposited in the Kaggle repository and is available at https://www.kaggle.com/datasets/abasset/indolr. The working source code experiment is published at https://github.com/sukasenyumm/IndoLR.

Conflicts of Interest

The authors declare that they have no conflicts of interest.
Acknowledgments

The authors thank the Ministry of Education, Culture, Research, and Technology of Republic Indonesia, through the Directorate of Research and Community Service, for supporting the funding of this research through a competitive grant program with the Higher Education Excellence Applied Research scheme (Grant No. 182/E5/PG.02.00.PL/2023).

References

conference on information and communication technology (icolict), pp. 1–6, IEEE, Yogyakarta, Indonesia, June 2020.


