https://doi.org/10.15407/csc.2024.01.038
УДК 621.39

V.O. KHOLIEV, Professor Assistant at Electronic Computers Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0002-9148-1561, vladyslav.kholiev@nure.ua

O.Y. BARKOVSKA, Assoc. Professor at Electronic Computers Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, ORCID: https://orcid.org/0000-0001-7496-4353, olesia.barkovska@nure.ua

IMPROVED SPEAKER RECOGNITION SYSTEM USING AUTOMATIC LIP RECOGNITION

The paper is focused on the relevant problem of speech recognition using additional sources besides the voice itself, in conditions in which the quality or availability of audio information is inadequate (for example, in the presence of noise or additional speakers). This is achieved by using automatic lip recognition (ARL) methods, which rely on non-acoustic biosignals generated by the human body during speech production. Among the applications of this approach are medical applications, as well as processing voice commands in languages with poor audio conditions. The aim of this work is to create a system for speech recognition based on a combination of speaker lip recognition (SSI) and context prediction. To achieve this goal, the following tasks were performed: to substantiate the systems for recognizing voice commands of a silent voice interface (SSI) based on a combination of two neural network architectures, to implement a model for recognizing visemes based on the CNN neural network architecture and an encoder-decoder architecture for the LSTM neural recurrent network model for analyzing and predicting the context of a speaker's speech. The developed system was tested on a chosen dataset. The results show that the recognition error in different conditions averages from 4.34% to 5.12% for CER and from 5.52% to 6.06% for WER for the proposed ALR system in 7 experiments, which is an advantage over the LipNet project, which additionally processes audio data for the original without noise.

Keywords: *SSI; ALR; AV-ASR; silent speech interface; automatic lip recognition; RNN; LSTM; speech recognition.*

Introduction

The effectiveness of speech recognition today is the result of decades of research by hundreds of scientists and engineers working in statistics, linguistics, semantics, prediction algorithms, and audio processing. Most speech recognition researchers, who realized limitations such as computing power, subsequently abandoned neural networks to use generative modelling approaches until the recent deep learning revival that began around the 2010s. As of 2016, major tech companies including Google, Apple, Amazon, Microsoft, and Baidu are using LSTM networks as the basis for their new products.

The main problem for researching recognition systems from technology companies is closed ac-

Цитування: Kholiev V.O., Barkovska O.Yu. Improved Speaker Recognition System Using Automatic Lip Recognition. *Control Systems and Computers*, 2024, 1, 38—49. https://doi.org/10.15407/csc.2024.01.038

cess to the code and details of the algorithms, as they are trade secrets, but there are other solutions from scientific communities, enthusiasts, or non-profit organizations. Examples of popular and publicly available recognition systems include CMUSphinx, HTK, which use a Hidden Markov Model (HMM) and were created by US universities [1] or the Julius system, which is an improved version of HTK with the use of trigrams and support for the Japanese language [2]. Developers also often use Kaldi, an open-source speech recognition toolkit based on deep neural networks with waveform pre-processing. Another set of speech recognition tools is RWTH ASR, which was created by a group of scientists from the University of Aachen. It consists of tools for developing acoustic models and decoders, as well as components for adaptive, unsupervised, or discriminative learning [3].

A modern speech processing system is a silent speech interface (SSI), which is implemented through automated lip reading (ALR), audiovisual automatic speech recognition (AV-ASR), or subvocal recognition (SVR). These methods use other devices that differ from the microphone and record the movements of the vocal apparatus (lips, tongue, larynx with vocal cords, etc.). Such systems also use a convolutional neural network (CNN) to extract image features.

SSI relies on non-acoustic bio-signals generated by the human body during speech production to enable communication when verbal one is not possible or desirable due to noise [4]. SSI devices can process a variety of bio-signals to enable silent communication, such as electrophysiological recordings of neural activity, electromyographic (EMG) recordings of vocal tract movements, or direct tracking of articulator movements using imaging techniques. Depending on the situation, some signal processing methods may be better suited than others for collecting speech-related information. For example, EMG and imaging methods work for laryngectomy patients who have lost the ability to speak after the vocal cords have been removed, but are not suitable for patients with severe paralysis [5]. From these signals, the intended message is decoded using speech recognition or synthesis algorithms.

Related Works

SSI systems are implemented in two ways — through the use of special medical devices that are usually attached to the skin and through the use of an optical camera that can analyze video images of a person speaking. The goal of the LipNet project is to understand natural speech, which is mainly made up of phonemes, making it necessary to use a speech corpus, a database containing words, phrases, and models that can work with them effectively. The shapes created by a person's lips or facial expressions can be processed and then converted into sounds, a group of which is compared to words. This technology has been successfully used to analyze old videos that did not have a soundtrack or to recognize people talking in noisy environments [6]. In ALR, there is a fundamental performance limitation due to homophones, which are sets of words that sound different but include the same lip movements, so they cannot be distinguished in an image alone. The solution to this problem is to use neural networks for content processing, which can clarify the meaning of these words or combine it with audio processing.

The LipNet system is based on a bilateral convolutional neural network, LSTM, and feature fusion. The traditional method is combined with a deep learning method and a network structure is created to extract the spatio-temporal characteristics of the video from the lips. Lip reading decodes textual content according to the visual information about the movement of the speaker's lips. The sequence of steps for recognizing words in video is as follows:

- the rank fusion method is used to convert a video sequence of lips into an image and all information about the video frames is stored in memory. This approach can reduce the input dimensionality of the deep neural network and, using an acoustic model, the probability distribution of the uttered words is calculated for each frame;
- a bilateral convolutional neural network processes the forward dynamic image (DI) and the reverse DI, as it reduces the deviation in sudden movements;

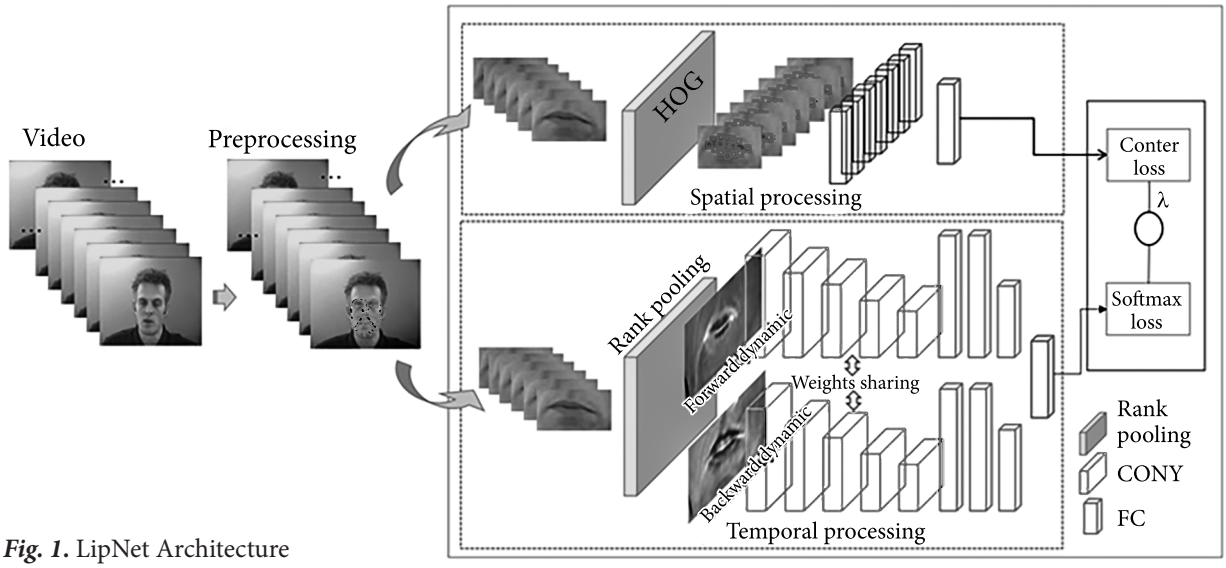

Fig. 1. LipNet Architecture

▪ the appearance shape function and depth function are combined, as this feature fusion method preserves spatio-temporal information and improves lip shape recognition depending on movements.

The architecture of LipNet consists of two parts: the first is classical feature detection, and the second is deep feature extraction, as illustrated in Fig. 1.

A histogram of oriented gradients (HOG) is used to detect the shape of lips in the frames. The information images obtained by the rank-based fusion are sent to a bi-recurrent neural network (BiRNN). All detected features are combined through a joint weight estimation function to obtain more significant features by training a convolutional layer to improve accuracy [7].

Alter Ego is a project developed in 2019 by MIT researcher Arnav Kapoor that effectively serves as a neural interface for working with a computer and requires only subtle stimulation of the speech muscles, without opening the mouth and without visible movements. [8]. The feedback to the user is provided through sound, using bone conduction, without disrupting the user's normal hearing perception. To use the headset to perform a simple task, such as searching for the weather on a computer, a query is first formed in the mind. The headset's sensors read bio-signals sent from the brain to the areas of the vocal apparatus that are used to speak out loud, such as the back of the tongue and the palate. The device then processes these signals as words and makes a request to a PC via a web connection. The researchers found that this system can understand its owner 92% of the time. The interface is being tested in a limited hospital setting, where it helps patients with multiple sclerosis communicate.

SpeakUP is an SSI-enabled project created by Varun Chandrashekhar in 2021 [9]. This device can be used by patients with speech impairments to communicate silently in English by simply speaking words or sentences in their mouths without making any sounds. SSI records EMG signals, which are then classified in real time using a trained machine learning model. The system has been found to provide 90,1% word recognition accuracy, is similar to Alter Ego, and is cheap to manufacture.

Ouisper is another way to get silent information through visualization [10]. Ultrasound is a non-invasive and clinically safe procedure that allows for real-time visualization of one of the most important articulators of the speech system, the tongue. Placed under the chin, the Ouisper ultrasound sensor can provide an overview of the tongue and is used for speech recognition, which in this case is called a silent vocoder.

Although researchers from Microsoft and other organizations have achieved human-level speech recognition based on the Switchboard results [11], there are still many unsolved problems. Switchboard's test is simple because each speaker is recorded on a separate microphone and different voices do not overlap in the same audio channel. People can understand several people speaking at the same time. A good speech recognition system should segment the audio based on who is speaking (diarization). Also of interest is the rate of semantic errors, which is rarely mentioned in system testing reports, although in practice users often encounter it in solutions from technology companies, for example, errors in recognizing voice commands from Google Assistant. In addition, the system should understand audio from multiple speakers (source separation), but such systems are still under development and have mediocre results. There are also problems with background noise for systems without SSI.

There is another disadvantage with publicly available solutions (Kaldi, Julius, etc.), as they cannot process words in context — past conversations, facial expressions, knowledge about the person speaking. Also, many solutions require a constant connection to the network, which makes them useless for many places due to the computing power or the number of trained models that need to be moved outside the recognition device to separate servers (for such solutions as Google, Microsoft, the Alter Ego project, Kaldi, and HTK software).

Most works on speech recognition from EMG or lip reading consider their own dataset, often with a very limited set of words or even independently recorded specifically for the paper (Table 1). In addition, speech analysis based on visual features is much more sensitive to changes in the user, different word sets, and accents. However, video-based speech recognition is generally more difficult than analysing audio signal processing projects.

Human speech contains about 50 phonemes (the smallest discernible unit of audio), while lip-reading can distinguish 10—15 visemes (groups of visually indistinguishable phonemes). Thus, the sequence of vizemes may often not correspond to a specific word and the accuracy of lip reading is highly context-dependent. In addition, even among people speaking the same dialect, the correspondence between lip movements and pronounced visemes can vary greatly, making it almost impossible to build a universal recognition model without information about the forms of lip movement or other organs. Another drawback for such systems is the problem with underexposed images in low light, although there are projects that use deep neural networks to improve illumination and remove noise in the image.

Also, authors often report the accuracy of their models trained on audio or visual features separately, but it is impossible to compare this with models that incorporate both. Therefore, it is more difficult to compare systems with each other than with audio information. For example, in the pro-

Table 1. **Comparison of speech recognition systems**

System	SSI	Technologies	Dataset	Command type	Accuracy, %
Kaldi	—	GMM/HMM	Switchboard	Sentence.	~80
Sphinx	—	HMM	EUSTACE	Sentence.	71,99
Microsoft	—	RNNLM	Switchboard	Sentence.	93,8
Ouisper	+	Ultrasound	Own	Word	~70
SpeakUP	+	EMG	Own	Phrase	80,1
Alter Ego	+	EMG	Own	Phrase	91,2
Fu et al (2008)	+	Cochlear implant	AVICAR	Numbers	37,9
Gergen et al (2016)	+	ALR	GRID	Word	86,4
LipNet	+	ALR	GRID	Sentence.	95,2

ject, the researchers managed to achieve 79% accuracy using only visual features [12].

Aims and Tasks of the Work

The aim of this work is to create a system for speech recognition based on a combination of speaker lip recognition (SSI) and context prediction. To achieve this goal, the following tasks need to be accomplished:

- to substantiate voice command recognition systems for silent speech interface (SSI) based on a combination of two neural network architectures;
- implement a model for recognizing visemes based on the CNN neural network architecture;
- implement the encoder-decoder architecture for the LSTM recurrent neural network model for analyzing and predicting the context of a speaker's speech;
- test the implemented system on the selected data set and analyze the results.

Results and Discussion

The speech recognition system uses a neural network-based architecture for speech processing, which maps sequences of video clips of variable length into text sequences.

1. ***Application of CNN.*** The input data in the form of video files is used to solve the problem of face analysis using convolutional neural networks (CNN), namely, to recognize facial features and visem.

Convolutional Neural Networks (CNNs) are an artificial neural network architecture aimed at efficient image recognition (Fig. 2). The convolutional kernel is a small weight matrix that is "moved" across the entire processed layer of the input image, generating an activation signal for the next neuron in the layer with the same position after each shift. To organize a convolutional neural network, the following is used:

- convolution (CONV);
- pooling (POOL);
- ReLU activation function;
- fully-connected layer (FC).

The image convolution operation is an operation between the image matrix and the convolution kernel (filter), when each element (pixel) in the output image is the sum of the product of the kernel element value and the value of the corresponding element of the input image matrix. An example of a convolution operation is shown in Listing 1.

The convolution process is iterative. First, a filter is applied to a section of the input image and the output value is recorded. Then the filter is shifted by one position if the step is 1, or by several positions if the step is set to a larger number, and the process is repeated until the convolutional function is completed.

Listing 1. An example of adding a convolution layer to a model

```
  model.
add(Conv2D(filters=64,kernel_size=
(3,3),strides=(1,1),
  padding='same',activation='relu',
name='2D-Convolutional-Layer'))
```

Multiple convolutional layers need to be configured to improve the network. The benefits come from the fact that subsequent convolutional layers reveal additional complexity in the image. The first layer in a deep convolutional network (DCN) tends to find low-level features (e.g., vertical, horizontal, diagonal lines...). Subsequent deep layers can identify higher-level features, such as complex shapes that are visually evident when certain sounds are pronounced, such as a hidden lower lip for /f/ or /v/ phonemes.

Usually, the aggregation layer is added after the convolution layer. Its purpose is to reduce the size of the convolved objects, increasing computational efficiency. It can also help eliminate noise by preserving the strongest activations.

In addition, a dropout layer is applied, which randomly sets the input units to 0 depending on the seed provided. This means that random inputs (features/nodes) will be zeroed out to prevent the network from overtraining and significantly increases the learning rate.

2. ***Application of LSTM.*** A recurrent neural network is typically used to find a regular pattern, such as words in a sentence, and an LSTM network

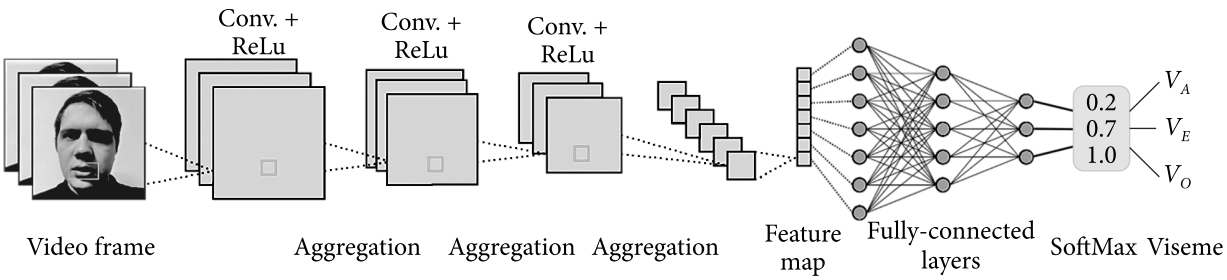

Fig. 2. CNN architecture for recognizing visemes

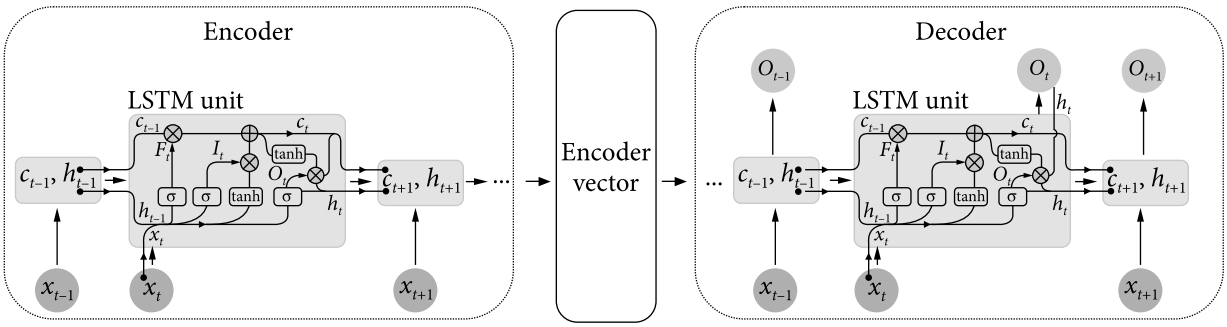

Fig. 3. LSTM encoder-decoder layout

is adapted to learning on tasks of classification, processing, and predicting time series in cases where important events are separated by time delays with uncertain duration [12]. To better detect and utilize long-term dependencies on sequence data (video or audio), the memory cell remembers related information that needs to be stored in a long sequence and removes some of the unnecessary information.

Any recurrent neural network has the form of a chain of repeating neural network modules. In a conventional RNN, the structure of a single module is very simple, for example, it can be a single layer with a hyperbolic tangent activation function, while the modules for LSTMs have a different form. Fig. 3 shows the key component of LSTM — the cell state — a horizontal line that runs along the top of the LSTM unit and participates in linear transformations, as well as generates the result.

Recurrent neural networks RNNs work by iteratively updating the hidden state *h*, which is a vector that can be of arbitrary size. They also apply shifts to h_t and y_t. It should be noted that at any given time *t*:

Voice control command recognition

- the next hidden state h_t is calculated using the previous one h_{t-1} and the next input x_t;
- the next outpu y_t is calculated using h_t.

In this case, the recurrent neural network consists of three weight parameters and two shifts, an example of such a cell is shown in Listing 2. Matrix multiplication is used for the weights, and then the vectors are added to the final result. Finally, a hyperbolic function or sigmoid is applied for the activation.

Listing 2. LSTM cell model

```
def RNN(x, weights, biases):
    x = tf.reshape(x, [-1, n_input])
    x = tf.split(x,n_input,1)
    rnn_cell = rnn.BasicLSTMCell(n_
hidden)
outputs, states = rnn.static_rnn(rnn_
cell, x)
    return tf.matmul(outputs[-1],
weights['out']) + biases['out']
```

In this paper, we propose to implement an encoder-decoder architecture for LSTMs, since a

sequence to sequence conversion process takes place. The input to the decoder is the visemes from the previous recognition unit, and the result of the encoder is a sequence of characters.

Technically, the input to an LSTM encoder-decoder can only use real numbers. The way to convert a phoneme or viseme to a number is to assign a unique integer to each element depending on its frequency of occurrence, so two dictionaries need to be created to prepare the training data, where each key is one of the 11 visemes and the value must be a number, and vice versa to decode the LSTM result. The decoding process involves feeding the state vectors and the sequence of characters to the decoder to make predictions for the next character and repeat the process until the end of the sequence character is encountered or the last character in the sequence is reached.

3. *Loading Files From the GRID Set*. When building predictive models, the original data is usually divided into training and control samples [13]. The GRID corpus consists of video and audio data recorded by 18 men and 16 women in a controlled environment with a video resolution of 720×576 pixels and 25 frames per second. Each pair of video and audio is also accompanied by a corresponding transcription, examples of which are shown in Table 2.

Listing 3. A snippet of the video frame download function

```
reader = cv2.VideoCapture(path)
ok, frame = reader.read()
if not ok:
break
else:
frames.append(frame[...,::-1])
return np.array(frames)
```

Table 2. **GRID dataset**

Command	Color	Preposition	Letter	Digit	Verb
Bin Lay Place Set	Blue Green Red White	At By In With	a—z (без w)	0—9	Again Now Place Soon

At the beginning of a Python project with an implementation, you need to set the path to the dataset so that the system can download the file. In the GRID dataset, there are 35 directories to be downloaded, but file number 21 is skipped because the video was corrupted for technical reasons. Once the dataset is loaded, functions from the OpenCV library are used to convert the video to a frame, as shown in Listing 3. You can also use a similar function to generate the results.

4. *Implementation Using CuDNN*. To speed up the process, you can use CUDA-enabled graphics card cores, which will significantly reduce the time it takes to train the neural network for real-time processing. Since the Python programming language was chosen, libraries with CUDA GPGPU support, such as Tensorflow and Keras, can be used to build high-performance layers and build models.

To speed up the process, you can use CUDA-enabled graphics card cores, which will significantly reduce the time it takes to train the neural network for real-time processing. Since the Python programming language was chosen, libraries with CUDA GPGPU support, such as Tensorflow and Keras, can be used to build high-performance layers and build models [14]. This library can also be used for CNN and LSTM, as it speeds up operations of combining, copying, element-wise calculation of hyperbolic tangent, matrix multiplication, addition, and other math functions.

The CuDNN-enabled model is not used by default in Tensorflow, so it is necessary to check the tf.device, which indicates the device used and switch from CPU to GPU.

It is also known that TensorFlow 2.0's built-in LSTM and GRU layers are suitable for using CuDNN kernels by default if a suitable GPU is available.

Keras has three packages for working with RNNs:

- SimpleRNN — fully connected RNN;
- GRU — managed recurrent blocks;
- LSTM — long short-term memory.

For LSTM, nested structures can be used because they store more information per unit of time. For example, a video frame can contain audio and video at the same time or consist of se-

veral vectors for each color. Such structures are used to modify an RNN cell or create your own architecture from different cells, an example of initialization is shown in the Listing 4.

Listing 4. Initializing a modified RNN cell

```
  NestedInput = collections.
namedtuple('NestedInput',
['feature1', 'feature2'])
  NestedState = collections.
namedtuple('NestedState',
['state1', 'state2'])

  class NestedCell(tf.keras.
layers.Layer):
    def __init__(self, unit1,
unit2, unit3, **kwargs):
      self.unit1 = unit1
      self.unit2 = unit2
      self.unit3 = unit3

      self.state_size = NestedState
(state1=unit1,
  state2=tf.TensorShape([unit2,
unit3]))
      self.output_size = (unit1,
tf.TensorShape([unit2, unit3]))

      super(NestedCell, self).__
init__(**kwargs)
```

The library for embedded RNN layers provides a cell-level API. Unlike RNN layers that process entire packets of input vectors, the RNN cell processes only one per iteration. Since the cell is located inside the loop, wrapping it with the tf.keras.layers.RNN layer makes it possible to process sequence packets, for example, RNN(LSTMCell(10)), which corresponds to LSTM(10). An example of using LSTM in Keras is shown in Listing 5. The cell abstraction together with the generic tf.keras.layers.RNN class allows you to modify the standard RNN architecture.

Listing 5. An example of LSTM application in Keras

```
  model = Sequential()
  model.add(Embedding(max_features,
256, input_length=maxlen))
  model.add(LSTM(output_dim=128,
activation='sigmoid', inner_acti-
vation='hard_sigmoid'))
  model.add(Dropout(0.5))
  model.add(Dense(1))
  model.add(Activation('sigmoid'))

  model.compile(loss='binary_
crossentropy',
   optimizer='rmsprop',
   metrics=['accuracy'])
```

Usually, the internal state of the RNN layer is reset with each new data packet (the data does not depend on the previous state). The layer will maintain the current state only for the duration of processing this element. However, if there are long sequences, for example, several hundred items, it is better to split them into shorter ones and gradually transfer them to the RNN layer without resetting the layer state. LSTM also uses the TimeDistributed function to encode the data content. Thus, the layer can store information about the entire sequence, although it processes only one subsequence at a time.

Results Analysis

Among the common ASR systems are Google Speech-to-text and CMU Sphinx [15]. Google STT is one of the most popular commercial ASR systems on the market [16], while CMU Sphinx has larger vocabulary databases that have been used by previous research in fields related to natural language processing and analysis [17], as the said speech-to-text system (CMU Sphinx) has a long history, starting in 1992 and ending with the latest update at the end of December 2023 [18].

Based on a study measuring the Word Error Rate metric [15], CMU Sphinx has a higher accuracy of recognizing input text — the average WER is 70.7%, while Google STT has an average WER of 86.2%.

A set of tools and libraries for speech recognition that can be used to develop various applications (speech-enabled text entry systems, speech-enabled identification and verification systems, speech translation systems, etc.) are provided with

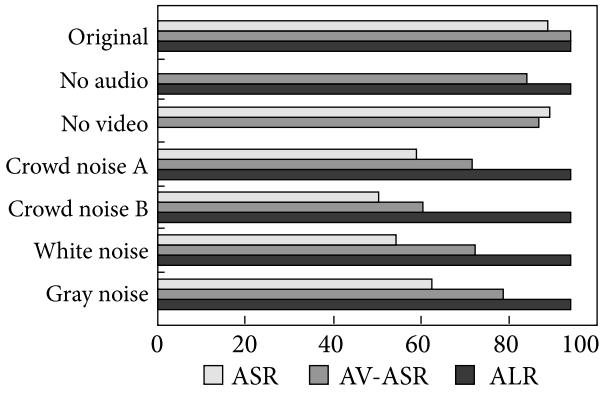

Fig. 4. WRR results in different conditions

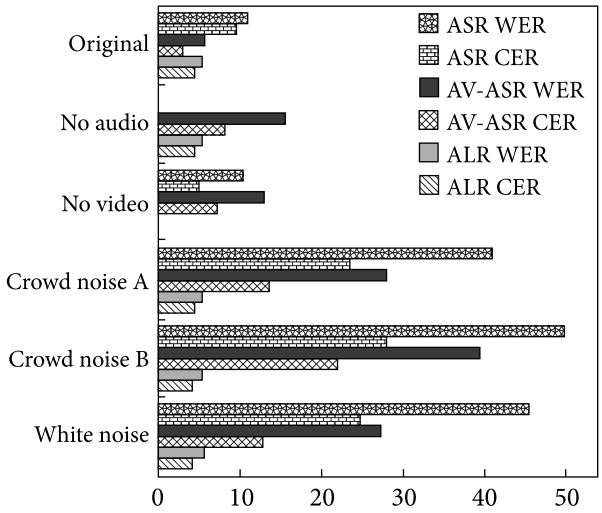

Fig. 5. CER and WER results in different conditions

the following CMU Sphinx advantages: **modularity** (CMU Sphinx consists of a set of modules that can be combined to create different speech recognition systems with different functionalities), **support for different languages** (CMU Sphinx supports speech recognition in different languages), **open source** (CMU Sphinx is an open system, which means that its source code is available for use and modification), **variety of models** (CMU Sphinx offers a variety of language and acoustic models that can be used to adapt the system to specific tasks and conditions)

Systems with noise were tested based on ALR, AV-ASR and ASR.

The ALR-based implementation can only recognize commands from video, so other approaches are used for comparison under different audio conditions: classical ASR and a combination of ASR and ALR — AV-ASR. The test sample is used from previous experiments, and a separate dataset of the GRID corpus without video is loaded to train LipNet based on AV-ASR and test CMU Sphinx, which cannot work with video files.

To test the systems, we also created several conditions that differed in noise (adding 10 dB of sound) for voice recognition (Table 3):

- white noise;
- gray noise;
- crowd noise A — simultaneous conversation of Ukrainian-speaking speakers;
- crowd noise B — simultaneous conversation of English-speaking speakers;
- no sound.

Table 3. **System performance in different conditions**

Condition	ALR		AV-ASR		ASR	
	Proposed system		LipNet		CMU Sphinx	
	CER	WER	CER	WER	CER	WER
White noise	4.34	5.79	13.04	27.4	24.9	45.59
Gray noise	5.12	6.06	11.38	21.46	22.17	37.42
Crowd noise A	4.52	5.52	13.74	28.18	23.6	41.11
Crowd noise B	4.43	5.65	22.2	39.52	28.04	49.86
No audio	4.58	5.73	8.34	15.8	—	—
No video	—	—	7.35	13.1	5.1	10.51
Original	4.6	5.54	3.1	5.8	9.67	11.14

When testing one-word commands, Table 3 shows the value of the WER metric measured as a percentage. It is WER%=100% — WRR% (for single-word commands), since WRR indicates the percentage of correctly recognized words compared to the total number of words in the text, which is useful for general evaluation of the recognition rate with larger text sets.

According to the results in Table 3, all command recognition systems perform well for the original based on the last three collections of the GRID corpus, however, CMU Sphinx (ASR) performs worse in terms of accuracy and error than the others. The recognition error under different conditions is 0.54% for the proposed ARL system in 7 experiments and is almost the same as for the LipNet project, which additionally processes audio data for the original without noise. We also did not measure the results for ALR in conditions without video and for ASR without audio, as these projects cannot use this data.

Error rates for ASR increase significantly when any noise is added, especially English speakers (crowd noise B), including for LipNet, which uses audio data and combines phoneme and viseme recognition. The reason for the deterioration in accuracy is that the system detects sounds and words that were previously recognized in the training set, such as "at", "zero", but LipNet's viseme processing improves the result over CMU Sphinx in these conditions. Also, ASR has better character and word error rates than LipNet in conditions without video. This may be due to the fact that the system is trained on video and waits for frames with people speaking.

LipNet showed the best result in terms of the number of character errors, recognizing phonemes (Fig. 4—5) that affected the final result, but the addition of noise worsened accuracy in other sound environments. Nevertheless, LipNet is able to work in conditions where there is no video, which is useful for recognition in cases where the speaker's lips are not visible in the dark or the camera has stopped working.

For the implemented ALR-based system, any conditions have no impact, since the changing audio signal is not an input in the recognition of phonemes or whole words, unlike the change of the face in the frames. This can be an advantage over other systems, since in the experiments conducted, a signal with noise degrades recognition accuracy for ASR and AV-ASR, but there are works that have improved the LipNet architecture for such cases [7], and noise reduction algorithms are used for ASR.

Conclusion

In this paper, we propose and describe a system for speech recognition based on a combination of speaker lip recognition (SSI) and context prediction. For this purpose, a model for recognizing the vizemes based on the CNN neural network architecture, a model of the LSTM neural recurrent network with the encoder-decoder architecture for the task of analyzing and predicting the context of the speaker's speech were implemented, after which these models were combined into a single system, in such a way, that the product of one is the input data of the next.

To achieve the acceleration, the CuDNN library was used, which allows training and running models on NVIDIA GPU accelerators, as it accelerates the operations of combining, copying, elementary calculation of hyperbolic tangent, matrix multiplication, addition, and other mathematical functions.

After practical experiments on a test set partially used from the previous experiments with the addition of a separate sets of data from the GRID dataset, the results show that all command recognition systems perform well for the original, but CMU Sphinx (ASR) performs worse in terms of accuracy and error than the others. The recognition error in different conditions averages from 4.34% to 5.12% for CER and from 5.52% to 6.06% for WER for the proposed ALR system in 7 experiments, which is superior to the LipNet project (on average 16,45%), which additionally processes the audio data for the original without noise. For the implemented ALR-based system, any conditions do not have an impact, since the changing audio signal is not an input in phoneme or whole word recognition, unlike the change in the face in the frames.

REFERENCES

1. Huang, X., Alleva, F., Hwang, M.-Y. and Rosenfeld, R. (1993). An overview of the SPHINX-II speech recognition system. *CiteSeer X (The Pennsylvania State University)*. doi: https://doi.org/10.3115/1075671.1075690.
2. Chung, J.S. and Zisserman, A. (2018). "Learning to lip read words by watching videos". *Computer Vision and Image Understanding*, 173, pp. 76—85. doi: https://doi.org/10.1016/j.cviu.2018.02.001.
3. Rybach, D., Gollan, C., Heigold, G., Hoffmeister, B., Lööf, J., Schlüter, R., Ney, H. (2009). "The RWTH aachen university open source speech recognition system". Proc. Interspeech 2009, pp. 2111—2114, doi: 10.21437/Interspeech.2009-604.
4. Tereshchenko, O.V., Barkovs'ka O.Yu. "Analiz vplyvu SSI-pidkhodu na produktyvnist' rozpiznavannya holosovykh komand". Materialy desyatoyi mizhnarodnoyi naukovo-tekhnichnoyi konferencii «Problemy informatyzatsiyi» (November, 24—25 2022) (In Ukrainian) [*Терещенко О. В., Барковська О.Ю.* Аналіз впливу SSI-підходу на продуктивність розпізнавання голосових команд. Матеріали десятої міжнародної науково-технічної конференції «Проблеми інформатизації». 24—25 листопада 2022 року].
5. Kapur, A., Kapur, S., & Maes, P. (2018). "Alterego: A personalized wearable silent speech interface". In 23rd International conference on intelligent user interfaces, Association for Computing Machinery, New York, NY, USA, pp. 43—53. https://doi.org/10.1145/3172944.3172977.
6. Orosco, E.C., Amorós, J.G., Gimenez, J.A., & Soria, C.M. (2019). "Deep learning-based classification using Cumulants and Bispectrum of EMG signals". IEEE Latin America Transactions, December 2019, 17(12), pp. 1946-1953. December 2019, doi: 10.1109/TLA.2019.9011538.
7. Zhang, T., He, L., Li, X. and Feng, G. (2021). "Efficient End-to-End Sentence-Level Lipreading with Temporal Convolutional Networks". Applied Sciences, 11 (15), p. 6975. doi: https://doi.org/10.3390/app11156975.
8. Hueber, T., Benaroya, E.-L., Chollet, G., Denby, B., Dreyfus, G. and Stone, M. (2010). "Development of a silent speech interface driven by ultrasound and optical images of the tongue and lips". Speech Communication, 52 (4), pp. 288–300. doi: https://doi.org/10.1016/j.specom.2009.11.004.
9. Mohapatra, D.R., Saha, P., Liu, Y., Gick, B., & Fels, S. (2021). "Vocal tract area function extraction using ultrasound for articulatory speech synthesis". In Proc. 11th ISCA Speech Synthesis Workshop (SSW 11), pp. 90-95. doi: https://doi.org/10.21437/ssw.2021-16.
10. Wand, M., Koutník, J., & Schmidhuber, J. (2016). "Lipreading with long short-term memory". In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vol. abs/1601.08188. pp. 6115—6119. URL: http://arxiv.org/abs/1601.08188.
11. Gonzalez-Lopez, J.A., Gomez-Alanis, A., Martin Donas, J.M., Perez-Cordoba, J.L. and Gomez, A.M. (2020). "Silent Speech Interfaces for Speech Restoration: A Review". IEEE Access, 8, pp. 177995—178021. doi: https://doi.org/10.1109/access.2020.3026579.
12. Yalkovskyi, A.Ye. (2009). "Problemy rozpiznavannya movy lyudyny". Problems of Informatization and Management, 3(27), pp. 163-166 (In Ukrainian). [Ялковський, А.Є. (2009). Проблеми розпізнавання мови людини. Problems of Informatization and Management, 3 (27). pp. 163—166. doi: https://doi.org/10.18372/2073-4751.3.570.
13. Kholiev, V., Barkovska, O. (2023). "Analysis of the training and test data distribution for audio series classification". Informatsiyno-keruyuchi systemy na zaliznychnomu transporti, 28. pp. 38—43. 10.18664/ikszt.v28i1.276343.
14. Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B. and Shelhamer, E. (2014). cuDNN: Efficient Primitives for Deep Learning. arXiv:1410.0759 [cs]. [online]. Available at: https://arxiv.org/abs/1410.0759.
15. Chen S.H.K., Saeli C., Hu G. (2023). "A proof-of-concept study for automatic speech recognition to transcribe AAC speakers' speech from high-technology AAC systems". Assistive Technology, pp. 1—8.
16. Del Rio, M., Delworth, N., Westerman, R., Huang, M., Bhandari, N., Palakapilly, J., McNamara, Q., Dong, J., Zelasko, P., & Jette, M. (2021). "Earnings-21: A practical benchmark for ASR in the wild". Interspeech, pp. 3465—3469. https://doi.org/10.21437/Interspeech.2021—1915.
17. Huh, J., Park, S., Lee, J. E., & Ye, J. C. (2023). "Improving medical speech-to-text accuracy with vision-language pre-training model". (arXiv:2303.00091). arXiv. http://arxiv.org/abs/2303.00091.
18. Shonibare, O., Tong, X., & Ravichandran, V. (2022). "Enhancing ASR for stuttered speech with limited data using detect and pass". Cureus, 14 (9). https://doi.org/10.48550/ARXIV.2202.05396.
19. GitHub. (n.d.). Release 5.0.3: Major bugfix release cmusphinx/pocketsphinx. [online] Available at: https://github.com/cmusphinx/pocketsphinx/releases/tag/v5.0.3 [Accessed 22 Mar. 2024].

Received 24.02.2024

В.О. Холєв, асистент кафедри "Електронно-обчислювальних машин",
Національний університет радіоелектроніки «ХНУРЕ», Харків, Україна,
ORCID: https://orcid.org/0000-0002-9148-1561,
vladyslav.kholiev@nure.ua

О.Ю. Барковська, кандидат технічних наук, доцент кафедри
"Електронно-обчислювальних машин",
Національний університет радіоелектроніки «ХНУРЕ»,
Харків, Україна, ORCID: https://orcid.org/0000-0001-7496-4353,
olesia.barkovska@nure.ua

ВДОСКОНАЛЕНА СИСТЕМА РОЗПІЗНАВАННЯ МОВИ
ЗА ДОПОМОГОЮ *AUTOMATIC LIP RECOGNITION*

Вступ. Робота присвячена актуальній проблемі розпізнавання мовлення за допомогою додаткових джерел, окрім власне голосу, в умовах, коли якість або доступність аудіо-інформації є недостатньою (наприклад, за наявності шумів або додаткових спікерів). Це досягається за допомогою методів *Automatic Lip Recognition* (*ARL*), що покладається на неакустичні біосигнали, які генеруються людським тілом під час відтворення мови. Серед застосувань такого підходу можна виокремити медичні застосування, а також обробку голосових команд у мовах з поганим звуковим середовищем.

Метою роботи є створити систему для розпізнавання мови на основі поєднання розпізнавання рухів губ спікера (*SSI*) та прогнозування контексту. Для її досягнення було виконано наступні задачі: обґрунтовано системи розпізнавання голосових команд безшумного голосового інтерфейсу (*SSI*) на основі поєднання двох архітектур нейронних мереж, реалізувати модель для розпізнавання візем на основі архітектури нейронної мережі *CNN* та архітектуру *encoder-decoder* для моделі нейронної рекурентної мережі *LSTM* з метою аналізу та прогнозування контексту промови спікера.

Результати. Розроблену систему було протестовано на обраному наборі даних. Тестування систем з шумом проводилися на основі *ALR*, *AV-ASR* та *ASR*. Реалізація на основі *ALR* вміє розпізнавати команди тільки з відео, тому для порівняння в різних звукових умовах застосовуються інші підходи: класичний *ASR* та поєднання *ASR* та *ALR — AV-ASR*. Тестова вибірка використовується з минулих експериментів, а також завантажується окремий набір даних корпусу *GRID* без відео для навчання *LipNet* на основі *AV-ASR* і тестування *CMU Sphinx*, який не може працювати з відеофайлами. Також для тестування систем було створено декілька умов, які відрізняються перешкодами (додавання звуку гучністю 10 дБ) для розпізнавання голосу.

Висновки. Результати показують, що похибка розпізнавання в різних умовах у середньому становить від 4.34% до 5.12% для CER та від 5.52% до 6.06% для *WER* для власної *ALR* системи у 7 експериментах що має перевагу над проєктом *LipNet*, який додатково обробляє дані з аудіо для оригіналу без шумів.

Ключові слова: *SSI; ALR; AV-ASR., RNN, LSTM, інтерфейс безмовного доступу, автоматичне розпізнавання рухів губ, розпізнавання мови, рекурентні нейронні мережі.*