Computer Speech & Language 65 (2021) 101131

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

LIS-Net: An end-to-end light interior search network for speech command recognition

Nguyen Tuan Anh*[a], Yongjian Hu[a], Qianhua He[a], Tran Thi Ngoc Linh[b], Hoang Thi Kim Dung[c], Chen Guang[d]

[a] School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, PR China
[b] Faculty of Electronic Engineering, Thai Nguyen University of Technology, Thai Nguyen, Vietnam
[c] Faculty of Civil and Environment, Thai nguyen University of Technology, Thai Nguyen, Vietnam
[d] R&D Department, 9-11 Kelin road, High-tech Industrial Development Zone, Guangzhou 510640, PR China

ARTICLE INFO

Article History:
Received 11 November 2019
Revised 29 April 2020
Accepted 27 June 2020
Available online 17 July 2020

Keywords:
Deep neural network
Speech Command Recognition
SCR
Keyword spotting
KWS
Light Interior Search Network
LIS-Net

ABSTRACT

With the rapid development of deep learning techniques, speech-based communication is getting more practically to be embedded into smart devices such as Alexa echo, TV, Fridge, etc. In this work, we have developed an efficient yet accurate Speech Command Recognition (SCR), that is particularly appropriate for low-resource devices. To this aim, a novel neural network, called Light Interior Search Network (LIS-Net), is presented that works with raw speech signal. LIS-Net is structurally composed of a sequence of parameterized LIS-Blocks, each of which is a stack of LIS-Cores, exploring the feature-map inheritance to learn highly distinctive and lightweight footprint of speech patterns. The proposed network is validated on Google Speech Commands benchmark speech datasets, demonstrating a significant improvement of accuracy and processing time in comparison with other state-of-the-art techniques.

1. Introduction

Nowadays, applying Deep Learning to smart devices has become an integral part of smart systems. Internet of things (IoT) is one of the leading systems in integrating intelligent control technologies to better serve human needs. Recently, the application of artificial intelligence in industrial and IoT devices has become a popular trend (Bhayani et al., 2016). The goal is to interact with people, increase responsiveness and reduce energy consumption. Puri et al. (2019) has introduced the methods of using AI in electronic equipment systems. For wide application, a system running on smart devices, in which speech is the most important signal, need to minimize energy, expand the ability to interact with people through audio and video. In the study Liua et al. (2019), the power dissipation of electronics is minimized by optimizing algorithms with specialized hardware. Zhou et al. (2019) has tried to solve the problem of dividing computational tasks of equipment systems in the network to improve the responsiveness of the system. If problems that require large amounts of data processing or tasks that require powerful artificial intelligence to process cannot be performed at the terminal, those systems will be processed in the cloud (Samie et al., 2019). From those strong trend, there are many applications of AI in practice with the goal of human-machine communication, such as Internet of Things for Enabling Human-Thing Cognitive Interactivity (Zhang et al., 2019), human-machine interaction using

*Corresponding author.
E-mail addresses: nt.anh@tnut.edu.vn (N.T. Anh), eeyjhu@scut.edu.cn (Y. Hu), eeqhhe@scut.edu.cn (Q. He), ngoclinhcnpm@gmail.com (T.T.N. Linh), hoangthikimdung85@gmail.com (H.T.K. Dung), cguang1@grgbanking.com (C. Guang).

https://doi.org/10.1016/j.csl.2020.101131
0885-2308/

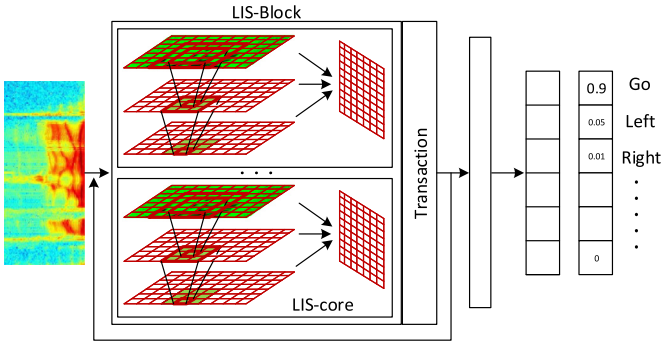

Fig. 1. Architecture of LIS-Net. Stacked parametric LIS-Blocks, in which each LIS-Block contains many LIS-Cores to find suitable parameterized information.

speech signal (Audhkhasi et al., 2017), and keyword spotting (Li et al., 2019; Liu et al., 2018; Yusuf et al., 2019; Liua et al., 2019) which have gained growing attention of researchers and manufacturers. Smart speech-enabled devices allow us to experience Hands-Free Speech Recognition by detecting key phrases to command or interact. The industrial devices are also controlled by voice commands with the basic keywords such as stop, up, down to lift the machine arms. The industry of SmartHome and Smart devices is growing aggressively with the ability to turn on/off the devices by keywords and/or voice commands. Only two keywords (yes/no) or 12 keywords ("yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", "unknown", and "silence") are currently not enough to use. Therefore, the voice command devices controlled with more keywords have become the trend. In 2017, Google expanded this field, so the Google Speech Commands dataset is proposed (Warden, 2018), including 35 commands such as left, right, up, and so on, which currently creates opportunities and challenges for Deep Learning models. The goal of a SCR model is to get high accuracy and small foot-print. Current models with high accuracy often have a large foot-print or vice versa. So it is necessary for developing a model with smaller foot-print and higher accuracy in order to solve a variety of problems in practice

To meet the expectation of a simple, powerful model which can solve the limitations of existing ones, a new model is studied and designed with the goal of a small number of parameters, fast running and high accuracy, inspired by such the architectures with high accuracy results in many areas today as Residual neural network ResNet (He et al., 2016), DenseNets (Huang et al., 2016), Inception (Zeng et al., 2016; Szegedy et al., 2014) and Xception (Chollet, 2017). These models are designed for image data. In this study, because the characteristics of command data are short. The 1D speech signal is transformed into 3D spectrum image to inherit the efficiency of CNN and ResNet while remaining the information on a wide range to have an overview on both the frequency and time axis, as well as the characteristics of the frequency spectrum in speech signals. In order to compare the results of the models easily, the literature and ours on the same Google Speech Commands dataset on sets of 12, 20 and 35 keywords are trained and compare the results in terms of accuracy, number of parameters, and inference time. The name so called "Light Interior Search Network (LIS-Net)" comes from the idea of searching the necessary features inner 3D space of a sample and concatenate internal stacked layers features to get better results. The word "light" means "light" in terms of the number of parameters and/or number of flops of the LIS-Net model, compared to the baselines structure of the network that LIS-Net inspired to create (ResNet, ResNext, DenseNet, Xception). From Fig. 9, it can be seen that to achieve the same or higher effect, LIS-Net has a smaller number of parameters. On the other hand, LIS-Net has a much smaller number of FLOPS than most baseline models with higher accuracy test results, shown in Fig. 10, so the authors called "light". The "interior search" part of the proposed name really comes from how a convolutional filter looks "within" its filter map. In each layer of Lis-Core, it has 3 different core layers that look and inherit useful feature in the internal layer. This idea will be clearly designed and explained in Section 3.

The model was designed to have a simple architecture by stacking blocks (called LIS-Blocks). It makes the implementation of the algorithm simpler and easier to understand. Besides, the number of parameters is also smaller.

The design of sketching the idea of LIS-Net is shown in Fig. 1. One-dimensional audio time series signals are converted to frequency domain, which holds the spectral value over time, from which the two-dimensional array is formed called input feature map which is extracted by one parameterize layer, then it is processed by layers in the stacked LIS-Block for searching the proper information to give out the most appropriate result shown in Fig. 1, in which, the system is designed as blocks (LIS-Block), which emphasizes the technique of extracting multi-layer, multi-region information or the combination of them. This technique exhibits the difference between LIS-Net and other model structures.

To maximize computational speed, 2D Separate convolution has been used in the core of LIS-Core to replace traditional 2D convolution. For an input image of size ($HxWxD$), to do 2D convolution (stride=1, padding=0) with Nc kernels of size ($hxhxD$). This transform the input layer ($HxWxD$) into the output layer $(H-h+1)x(W-h+1)xNc$. The overall multiplications needed is

$$Ncx(hxhxD)x(H-h+1)x(W-h+1) \tag{1}$$

Now with depthwise separable convolutions, with the input layer ($HxWxD$), it is divided into two phases:

- In the first phase, the filter ($hxhxD$) is split into D filter ($hxhx1$), after the transformation, the output will be $(H-h+1)x(W-h+1)xD$.

- In the second phase, to extend the depth, convolving the $(H-h+1)x(W-h+1)xD$ input image with Nc times $(1x1xD)$ convolutions, the output size will be $(H-h+1)x(W-h+1)xNc$.

The multiplication needed for an image $(HxWxD)$ is

$$\begin{aligned} &Dx(hxhx1)x(H-h+1)x(W-h+1)\\ &+Ncx(1x1xD)x(H-h+1)x(W-h+1)\\ &=(hxh+Nc)xDx(H-h+1)x(W-h+1) \end{aligned} \tag{2}$$

The ratio of multiplications between depth-wise separable convolution and 2D convolution is now:

$$1/N_c+1/h^2 \tag{3}$$

When $(Nc > > h)$, then the expression (3) reduces to $1/h^2$. That is why depth-cut convolution is chosen for LIS-Core for the purpose of high speed and small power.

Some main contributions of this study as follows:

- A new Deep Neural Network called LIS-Net was proposed, which achieved new state-of-the-art results, about 24% higher than the current state-of-the-art.
- The LIS-Net network was proposed with a simple architecture, which is suitable with speech command data with a smaller footprint and faster predictions than those of very powerful current models.
- LIS-Core architecture was proposed with feature-map inheritance to minimize the number of calculations, which help to reduce footprints and prediction time as well as install in applied embedded systems easily.
- LIS-Net network with parameterized LIS-Block and LIS-Core to support Auto Machine Learning for future application problems.

The rest of this paper has been structured as follows. Section 2 reviews the main approaches for speech keyword spotting and recognition. Section 3 presents in detail the proposed method. Section 4 describes the datasets, evaluation metrics and protocols, and experimental results. Finally, Section 5 concludes the paper and gives several possible lines of future studies.

2. Related work

Recently, the end-to-end model based on Convolutional neural network (CNN) is increasingly popular for SCR problems (Fourniols et al., 2018), such as “Convolutional Neural Networks” (Sainath and Parada, 2015; Andra and Usagawa, 2017), “Deep Neural Network” (Chen et al., 2014) and ResNet (Tang and Lin, 2017a). Computer Vision is one of the strong points of CNN because of the fast computing ability and is widely used in many fields including speech recognition. Many networks have achieved very high results such as CNN-BiLSTM, DeepSpeech (Speech and Group, 2016), Transformers (Jaderberg et al., 2015; Zhou et al., 2018), CNN-Attention (Chorowski et al., 2014; Tachibana et al., 2018). Another structure is Recurrent Neural Network (RNN) designed specifically for time series data like speech audio (Sun et al., 2017; Hwang et al., 2015), but due to sequential architectures of data, the calculation speed of RNN is much slower than that of CNN. Recently, there has been much progress in conquering accuracy results, and more state-of-the-art results have been continuously improved. In 2017, R. Tang *et al.* implemented a type of DNN, achieved 91.97% on 35 keywords (Tang et al., 2017). This model has focused on reducing the electrical energy consumption, but its results are not so high. The study (Tang and Lin, 2017b) has achieved the results with accuracy of 95% on 12 keywords; however, this model is quite small and is not strong enough to train a large keyword set, which requires the large memory capacity of the network. In the study (Zhang et al., 2017), the authors focused on optimizing the model to be small and fast with the accuracy of 95.4% on 10 keywords. Their model is a combination of CNN and RNN called Convolutional Recurrent Neural Network (CRNN). However, the accuracy of 95.4% is still the number needed to improve and increase the number of keywords. The study “Deep residual learning” (Tang and Lin, 2017a) gained state-of-the-art, outperforms Google’s previously-best CNN (Sainath and Parada, 2015) (95.8% vs 91.7% in accuracy) on a set of 12 classes; In the study of “Effective Combination of DenseNets and BiLSTM for Keyword Spotting” (Zeng and Xiao, 2019), the obtained results achieved state-of-the-art with accuracy of 96.6% on 20 command sets, but the authors said that “One disadvantage of CNN and ResNet is that they cannot get the dependency term on speech audios well”, and they use Google Speech Commands with 1 s in length. It is clearly seen that for the SCR problem, the signal is usually short, so the image dimension is not too long when converting the speech signal to the audio spectrum image. Hence, long term dependency is not really necessary and will be proven by experiments. Through these studies, it can be seen that in order to be practical, it is necessary to improve the accuracy and takes advantage of the fast computing power of embedded devices.

3. The proposed model

The architecture of LIS-Net network is illustrated in Fig. 2. The input layer for 16 kHz raw wave data using to create spectrogram image (Kim et al., 2018), the next numbers of blocks, called the Light Interior Search block (LIS-Block), and a classification block for creating the number of output classes (N_c) are stacked together. Each LIS-Block is stacked by number of LIS-Cores (core

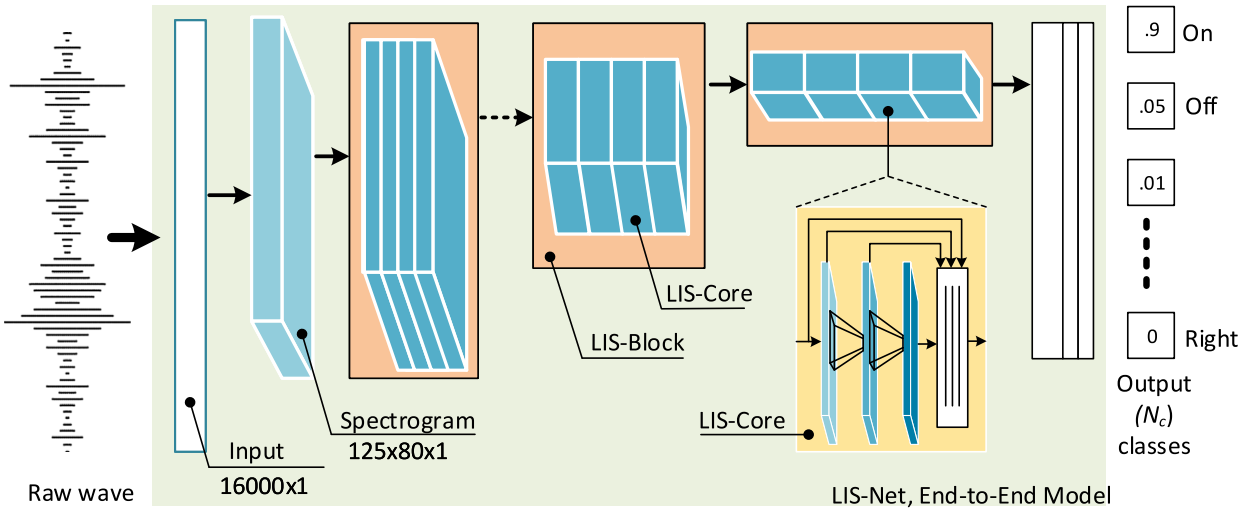

Fig. 2. Overview of System diagram. End-to-end architecture is used from the wave-form input layer to the classified labels layer. In which, LIS-Net contains a layer for convert wave form to spectrogram and stacked LIS-Blocks that contain LIS-Cores to extract features. N_c refer to number of the output classes.

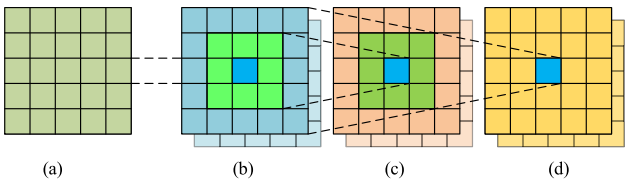

Fig. 3. Feature searching region of LIS-Core. The next layer extracts useful information based on the results of the previous layer, the search region is expanded on the input feature.

block of LIS network) and enclosed by two convolution followed by batch normalization and activation layers. It aims to increase the ability to learn parameters through intermediate layers. Each output of LIS-Block is transited by a max pooling block. Unlike ResNet, LIS-Net's architecture has the reduced width, height and the increased depth of feature tensor after each LIS-Block. In a block, the dimension of the LIS-Core's feature remains unchanged, but it is easy to change the number of cores. It leads to change of network depth easily and can use for different problems. The purpose of this design is aimed at optimizing the network for each further specific problem. Adjusting the width of the network between adjacent LIS-Blocks is done by two layers of convolution and max polling.

LIS-Core is optimally designed for calculation speed, the minimized number of parameters. The feature search regions, the areas that the three layers' filters mapped on input feature at one time, cover both frequency and time domains for speech data. Through the experiments with the LIS-Net network architecture, the search domain in LIS-Core has divided into three branches (1×1, 3×3, and 5×5) and four branches (1×1, 3×3, 5×5 and 7×7) according to the design method of Inception (Zeng et al., 2016) and InceptionResNet (Szegedy et al., 2014). It is found that the number of calculations is very large, and the number of parameters is significantly increased, but the efficiency is not different from the current LIS-Core method.

In LIS-Core, 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions, shown in Fig. 3 (a $\rightarrow$ b). Then, all features correlations are mapped to other spatial ones on the previous feature itself, shown in Fig. 3 (b $\rightarrow$ c). It makes the search area wider by frame via 3×3 convolution based on 1×1 channel-wise at the first step, In order to find the information of the 5×5 area on the original feature (Fig. 3a), the convolution based on the previous 3×3 area was performed (Fig. 3c $\rightarrow$ d). Finally, the features of 3 times convolution are concatenated together. The structure of LIS-Core block is shown in Fig. 5. For speech, it means that simultaneous interior search is taken by following frequency and time domain with three regions having different dimensions, giving more opportunities to find useful information. That is the characteristic of this model, so it is named Light Interior Search network (LIS-Net).

LIS-Core uses Residual Shortcut (He et al., 2016), so LIS-Block is no longer needed while still ensuring LIS-Net to avoid vanishing gradient. On the one hand, LIS-Core synthesizes useful features from 3 times of channel-wise 1×1, 3×3, 5×5 3D space mapping based on the previous layer. The features output has the same length and width and only increases depth following residual shortcuts, so it is possible to customize the number of blocks for any specific problem.

To understand the architecture of the network easily, LIS-Core is calculated as $LC(\cdot)$ function that getting output from $Z^{[6]}$, shown as Fig. 4, defined in (4) in which a infer to input feature:

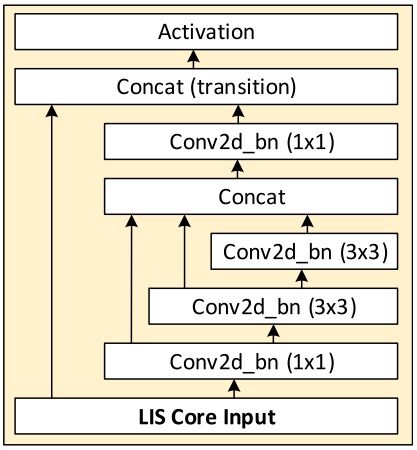

Fig. 4. LIS-Core structure.

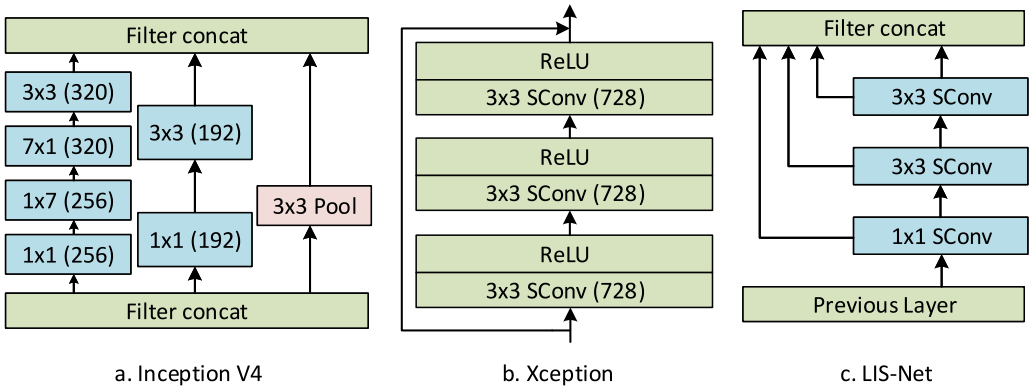

Fig. 5. Comparing core of the models. Sconv denotes for Separate Convolution, the numbers in parentheses are fixed feature size, mxn denotes for filter size.

$$\begin{aligned} Z^{[1]} &= \sigma(W^{[1]}a + b^{[1]}) \\ Z^{[2]} &= \sigma(W^{[2]}Z^{[1]} + b^{[2]}) \\ Z^{[3]} &= \sigma(W^{[3]}Z^{[2]} + b^{[3]}) \\ Z^{[4]} &= \sigma([Z^{[1]} : Z^{[2]} : Z^{[3]}]) \\ Z^{[5]} &= \sigma(W^{[5]}Z^{[4]} + b^{[5]}) \\ Z^{[6]} &= \sigma([a : Z^{[5]}]) \end{aligned} \tag{4}$$

where ": " denoted as concatenate features, σ is Relu activation. LIS-Block denote as LB function defined as (5) with input feature a:

$$\begin{aligned} g^{[1]} &= \sigma(W_{Lb}^{[1]}a + b_{Lb}^{[1]}) \\ g^{[k]} &= LC(g^{[k-1]}), k = 2..n \\ g &= pooling\left(\sigma(W_{Lb}^{[trans]}g^{[k]} + b_{Lb}^{[trans]})\right) \end{aligned} \tag{5}$$

where $LC(\cdot)$ is defined in (4), $g^{[1]}$ refer to feature map of the first layer in LIS-Block to enrich learning feature and layers, $g^{[k]}$ refer to feature map of stacked LIS-Core functions, n refer to maximum number of LIS-Core in the LIS-Block, g refer to feature map of transition layer, it also is the result of LIS-Block function. $W_{Lb}^{[..]}$ and $b_{Lb}^{[..]}$ refer to weight and bias in a LIS-Block, respectively.

Table 1
The structure of LIS-Net model.

LIS-Net Layers	LC	G_R	B_P	Output size	LIS-Block
Wav data Input				16000	
Spectrogram Layer				125, 80, 1	
LIS-Block	1	1	48	63,40,48	conv2d_bn 1 × LIS-Core Transition MaxPooling2D
LIS-Block	2	2	48	32,20,96	conv2d_bn 2 × LIS-Core Transition MaxPooling2D
LIS-Block	3	4	48	16,10,192	conv2d_bn 3 × LIS-Core Transition MaxPooling2D
LIS-Block	4	8	48	8,5,384	conv2d_bn 4 × LIS-Core Transition MaxPooling2D
Classification block				N_c	MaxPooling2D conv2d_bn GAPooling2D Dense

Conv2d_bn: including 2D convolution, batch normalization and activation layers
LC: LIS-Core quantity
GR: Growth rate of block's feature size
BP: Base Parameters
Transition: Adjust filter to fit N_F
MaxPooling2D: 2D max polling to reduce feature size
GAPooling2D: 2D Global Average Pooling layer
N_c: numbers of output classes
Dense: fully connection to get output size of N_c

LIS-Net is stacked by LIS-Blocks together, and each LIS-Block has a specific dimension and the different number of LIS-Core. Parameters affecting the network include:

- N_{core}: the number of LIS-Cores in LIS-Blocks. If N_{core} is designated [1,2,4,8], it means that the network has 4 LIS-Blocks. Each LIS-Block in the order has the stacked LIS-Core numbers by 1, 2, 4 and 8, respectively.
- N_F: Number of filters, representing the feature size of a LIS-Block.
- B_P: The base parameter, the factor of feature size (N_F) with a constant value throughout the network.
- G_R: The Growth rate parameter to change the number of filters of LIS-Blocks. Similar to N_{core}, each element of G_R will be used to calculate N_F at the corresponding LIS-Block, seen in formula (7).

$$N_F{}^{(k)} = G_R{}^{(k)} * B_P \tag{6}$$

$$\begin{cases} G_R^{(0)} = 1 \\ G_R{}^{(k)} = 2 * G_R{}^{(k-1)} \end{cases} \tag{7}$$

Where: k refers to the kth LIS-Block with $k = 0, .., n-1$; n: the total number of LIS-Blocks available in the network. The deeper the network is, the more changes the features will be, so increasing G_R is selected to ensure that the most useful features are stored.

The network is configured by default as shown in Table 1.

To create output classes (N_c), the use of global average pooling thus allows networks to function with less computational power and have better generalization performance (Lin et al., 2013). This network architecture has two types of parameters which are the number of cores in each LIS-Block (LIS-Core) and the dimension of features map in LIS-Net (G_R) based on base parameter (B_P). These parameters can be easily modified even during training process to select the optimal network, and since then LIS-Net can support effectively for auto-searching parameters by AutoML method. In this study, *sparse categorical crossentropy* loss, *Adam* optimizer, and *sparse categorical accuracy* metric are used for all models.

3.1. Comparing structure of LIS-core with the core of other models

The goal of enhancements is to find ways to optimize model. Here, Inception, ResNet, DenseNets, Xception, ResNeXt baseline are chosen. There are many improvements that have been proposed. In each improvement, the scope of extracting information

of branches is changed to reduce the number of calculations and increase the scanning area. There are some major differences in module improvements in Inception from v1 to v4, the authors still keep the multi-branch architecture. The difference between versions is not much, here, version 4 is chosen for comparison, shown in Fig. 5a. This multi-branch designs make the module learn more features, but it is cumbersome. In the Fig. 5b, Xception core is a stacked design, in which each block is made up of Separate Convolution and activation classes. The advantage of this architecture is the improvement in speed compared to traditional convolution. However, due to the large size of the feature map, the footprint model is still large.

LIS-Net is designed to be completely different to minimize capacity and number of calculations. The number of branches has inheritance, and the latter branch inherits the result of the previous one in order to reduce the number of calculations that are needed to be performed, shown in Fig. 5c.

4. Experimental results

In this section, the method used in the experiments will be discussed. The dataset was used and compared with the results in published models. To compare LIS-Net with baseline models and other existing ones, the experiment was performed on the Google Speech Commands v1/v2 data set (Warden, 2018) including 12, 20 and 35 keywords. Each keyword in a wave file has 1 s in length. Input speech data is raw wave, and the program reads and saves to hard disk using NumPy format to increase data loading performance into Deep Learning Framework generator (de Andrade et al., 2018). During the experiments, raw wave input was processed by using a layer in the model to create spectrogram features. All models are similarly trained with setting random seed parameters to constant. Only the model is different to compare results easily in the same condition. The results are represented by testing accuracy for all the models.

4.1. Google speech command dataset

In order to use a model with an audio signal of any length, it should go through two stages, first, voice activity detection, split audio into short pieces, and second, speech command recognition. For convenience of comparing results, this study uses Google Speech Command dataset (Warden, 2018), it was introduced by Pete Warden consisted of 105,829 utterances of 12, 20 and 35 keywords in the training set. Each keyword is about one-second (or less) WAVE format file, made up of thousands of contributors. The Speech Commands Dataset was split into training, validation, and test sets, with 80% training, 10% validation, and 10% test by Google. When using the spectral feature, the raw wave is converted to a spectrogram with feature size of $125 \times 80 \times 1$.

4.2. Mel spectrogram parameters

To create a $125 \times 80 \times 1$ Mel-spectrogram image, the parameters of the *Spectrogram* layer were selected as follows:

- The number of DFT points:1024
- Hop length between frames in sample:128
- input shape (audio-channel, audio-length): (1, 16000)
- Padding strategies at the ends of signal: same
- The number of Mel bands: 80
- Minimum and maximum frequency to include in Mel-spectrogram: 40.0 and *sr*/2, respectively, where *sr* refers to audio sample rate

4.3. Hyper-parameter configuration

The coefficients of the experimental problem are optimized with the hardware used for model training with the same configuration for all models:

- Optimizer: Adam
- Momentum: 0.9
- Initial learning rate: 0.001
- Learning rate decay: Learning rate will be reduced to 10% compared to the current learning rate if val_loss does not decrease after 3 epochs.
- Dropout: It was not used. The reason is that Test Acc was significantly reduced after trial training for some models.
- Batch size: 64

4.4. Training infrastructure

All the trained models on systems have the following configuration: 4 NVIDIA GPU 1080ti, 2 CPU 8 thread each, 128 GB RAM, 2T Seagate HDD. Each model is assigned to use only 1 GPU and 4 CPU threads. During training process, parameters, such as model

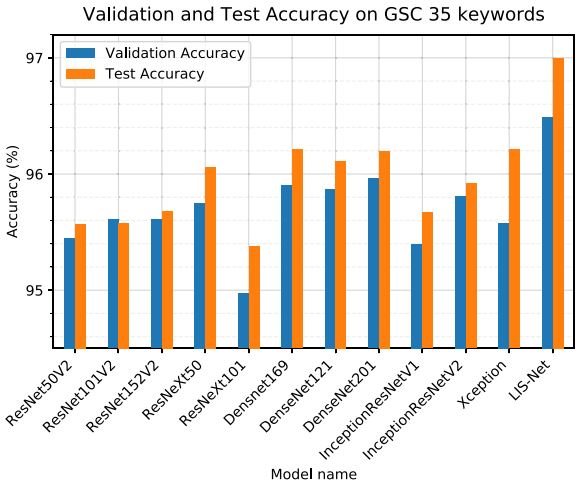

Fig. 6. Validation and Test Accuracy. Comparing the accuracy of models on the GSC v2 dataset on 35 keywords, sorting by model with similar architecture. Higher is better.

size (footprint), time of 1 batch training, all the number of epochs reaching to early stop, validation and test accuracy was carefully recorded for comparison.

4.5. Model training

To make it easier to compare results, the baseline models including DenseNets, ResNet, ResNeXt, Inception-ResNet, Xception and LIS-Net are trained on the same condition using different standard network sizes. The parameters of each model are recorded. For 12 and 20-command set, the proposed models are trained to get results and compare with the highest ones of published articles.

4.6. Results

To compare the results clearly and objectively, two types of comparison are implemented. The first type aims to compare the results with very powerful models today, which have architectures close to the proposed model. The other one compares with the models having state-of-the-art accuracy results in SCR.

4.6.1. Comparison with recently powerful models

The Keras framework is used to implement the models which are trained in the same condition without augmentation speech inputs. LIS-Net's results achieve new state-of-the-art in the Speech Command Recognition (SCR) problem seen in Fig. 6–8. LIS-Net model still has plenty of space to customize and refine, and it can be achieved to higher accuracy results. Among the investigated models, it can be clearly seen that LIS-Net achieved the highest results on both validation and test accuracy while still remaining a footprint small enough and shortest training time for each epoch, which strongly affects the prediction time on actual problems.

In detail, firstly with 35 commands set, all three standard ResNet sizes including ResNet-50, ResNet-101 and ResNet-152 give out the test accuracy results lower than 96%. Larger networks normally have bigger footprints but not always get better results. Compared with ResNet, the accuracy result of LIS-Net is higher (rising from 95.57% to 97%) while the total parameters are significantly smaller than those of others, seen in Fig. 9. Moreover, ResNet has the lowest test accuracy seen in Fig. 6. With ResNeXt 50 and ResNeXt 101, the highest test accuracy result is 96.06%, because ResNeXt has an improved architecture from ResNet. Compared to LIS-Net, the number of parameters is larger, but the test accuracy is still lower. For DenseNets having the special architecture, it inherits the feature map from all previous layers, so the results seem to be better with the highest value of 96.21%, in which DenseNets-121 with 8.16M parameters in total is slightly higher than those of LIS-Net; however, the test accuracy is still lower. For the improved Xception model inspired by ResNet and InceptionResNet, the test accuracy is higher than that of InceptionResNet, but still lower than that of LIS-Net. Secondly, with 12 keywords shown in Fig. 7, although all models archive high accuracy results, the result of LIS-Net is still the highest, reaching state-of-the-art in this dataset. And finally, comparing the accuracy results in the set of 20 keywords. Once again, LIS-Net showed remarkable ability when both validation and test accuracy performed better than baseline models. From the results shown in Fig. 6–8, LIS-Net has achieved new state-of-the-art performance of the convolution-based models on the Google Speech Command dataset with the smallest total parameters.

The comparison between Floating-point Operations Per Second (FLOPS) and test accuracy of models, seen in Figure 10, shows that the number of FLOPS of LIS-Net is approximately equal to that of DenseNet-121, but the accuracy is much higher. LIS-Net

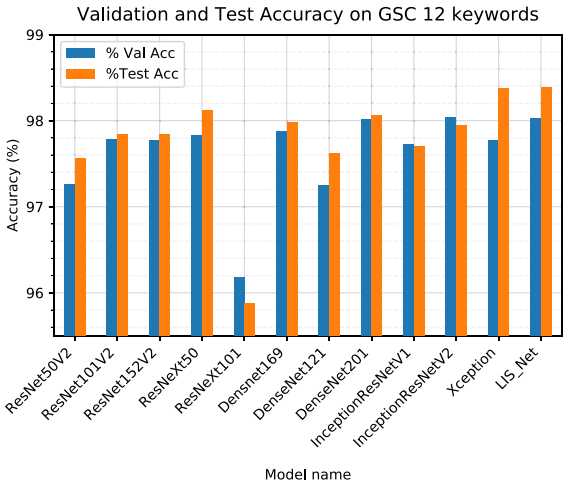

Fig. 7. Validation and Test Accuracy. Comparing the accuracy of models on the GSC v2 dataset on 12 keywords, sorting by model with similar architecture. Higher is better.

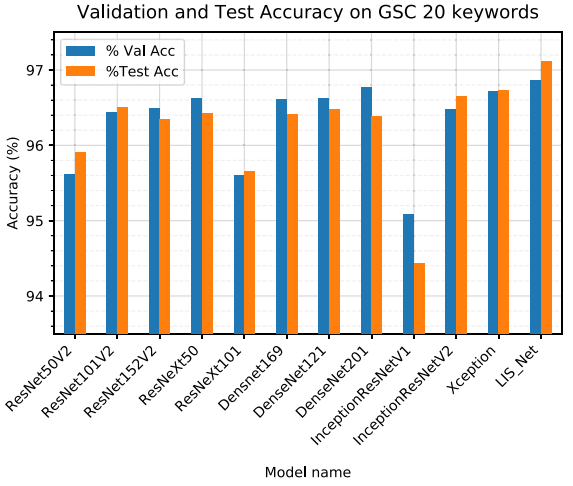

Fig. 8. Validation and Test Accuracy. Comparing the accuracy of models on the GSC v2 dataset on 20 keywords, sorting by model with similar architecture. Higher is better.

has a much smaller number of FLOPS compared to other models. Accordingly, it can be more preferable to use in embedded system devices than the other alternative models.

4.6.2. Comparison with state-of-the-arts models

Recently, there have been numerous speech data studies based on RNN, Attention, and combinations of those models (Zeng and Xiao, 2019). In Table 2, the models are compared on the Google Speech Command v1 dataset, which includes the ones with recent state-of-the-art results. In order to look up the results conveniently, the model's reference column is added. Similar architecture models are grouped together.

From Table 2, it can be observed that the highest result of the CNN group is 95.6%, slightly lower than that of ResNet and RNN (LSTM, GRU) (95.8%). Currently state-of-the-art results are in the combination model, DenseNets-BiLSTM3 (97.5%). LIS-Net model was born to create new state-of-the-art with superior results (98.1%), and the proposed model is proven to work very well with 12 keywords. With Google Speech Command of 20 keywords, M. Zeng (Zeng and Xiao, 2019) had implemented and gave the best results as shown in Table 3. DenseNets-BiLSTM is still the current state-of-the-art, but compared with LIS-Net, LIS-Net has achieved outstanding results, which once again prove that it achieves the new state-of-the-art performance on both Google Speech Commands dataset v1 and v2.

Table 2
Accuracy comparison of Models on Google Speech Commands dataset.

Model	**Accuracy (%)**, (12 words)	Ref & implementation
ConvNet	90.5	(Sainath and Parada, 2015), *
Tpool2	91.97	(Tang et al., 2017)
TDNN	94.3	(Zhang et al., 2017)
DS-CNN	95.4	(Zhang et al., 2017)
HD-CNN	95.6	(Yan and Zhang, 2015), *
Res26	95.2	(Tang and Lin, 2017a), Google's implement
Res15	95.8	(Tang and Lin, 2017a), Google's implement
BiLSTM-2	93.6	*
BiLSTM-5	94.5	*
BiGRU-2	94.7	*
BiGRU-8	95.7	*
BiGRU-5	95.8	*
DenseNets-BiGRU	96.1	*
DenseNets-BiLSTM-2	96.2	*
Attention RNN	95.6	(de Andrade et al., 2018), *
DenseNets-BiLSTM-3	97.5	*
LIS-Net	**98.4**	Our model

*: best results, implemented by (Zeng and Xiao, 2019).

Table 3
Accuracy comparison of Models on Google Speech Commands dataset.

Model	**Accuracy (%)**, (20 keywords)	Ref & implementation
ConvNet	88.2	(Warden, 2018), *
C-1-G-2-BLSTM	90.9	*
Attention RNN	94.5	(de Andrade et al., 2018),*
DenseNets-BiLSTM-3	95.9	*
DenseNets-BiLSTM-2	96.6	*
LIS-Net	**97.1**	Our model

*: best results, implemented by (Zeng and Xiao, 2019).

4.6.3. Learning speed

To compare the training speed, all models are initialized at the same condition and trained on Google Speech Command v1. The speed of training models is recorded by early stop with minimum val_loss measurement, patience = 5, and batch size = 32. The training time per epoch of models is illustrated in Fig. 11. With the obtained results shown in Fig. 6, LIS model is chosen with average training time compared to other methods for best results to trade-off between speed and precision. It also affects the predictive time when the model is applied to the further applications.

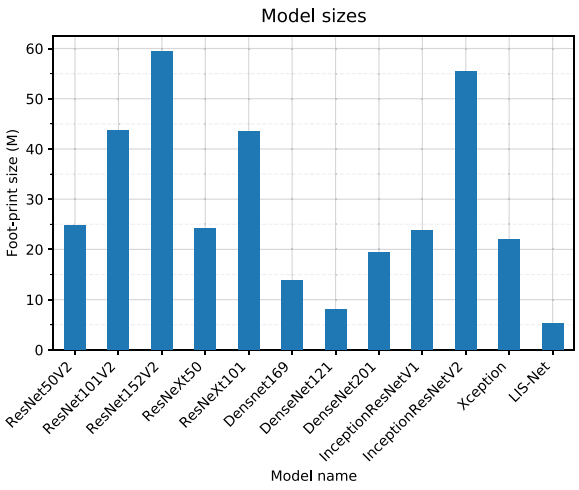

Fig. 9. Model sizes (foot-print size), smaller is better.

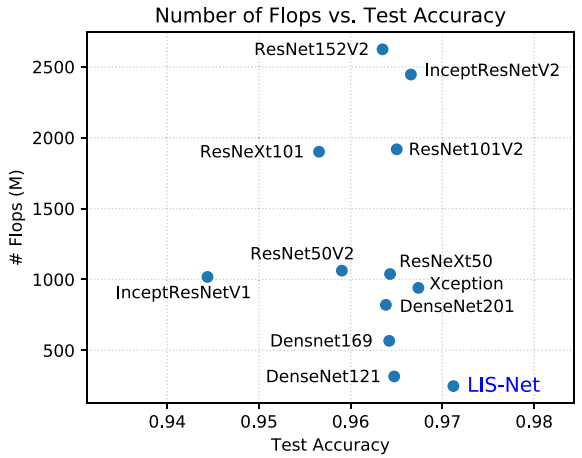

Fig. 10. Test Accuracy vs Number of Flops of the models, calculating on GSC-v2 20 keywords, in which #Flops lower is better, Test Accuracy higher is better.

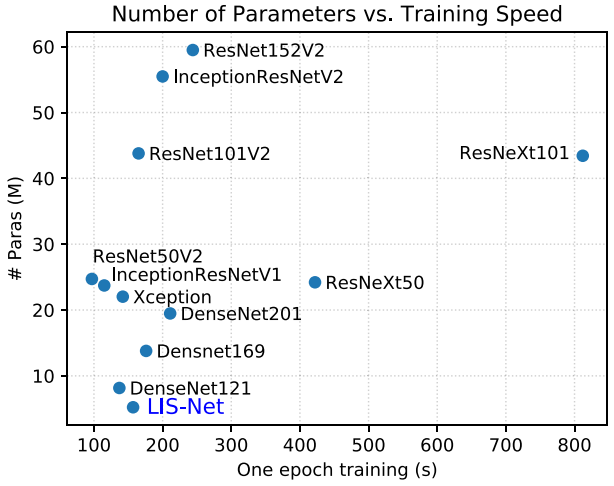

Fig. 11. Training speed vs Number of Parameters (#Paras(M)) of the models in GSC-v1 35 keywords in which lower is better

LIS-Net, DenseNets, Xception InceptionResnetV1 and ResNet 50 have nearly equal training time (Fig. 11). Due to the trade-off between test accuracy, training time and total parameters, high test accuracy is chosen to adjust the network.

4.6.4. Impact of different LIS-Net

The network is structured as shown in Fig. 2. The main component of LIS-Net is structured by stacking LIS-Blocks, and each LIS-Block is stacked by LIS-Cores with transition layer. The core size is determined by the filter parameters N_{core} with the number of filters given by the formula (6). The default G_R parameter is defined by the formula (7), but in case of variations or impact assessments of network parameters, this formula (7) may not be followed to get network structure on demand.

Conventional meanings in the comparative sections as below (Table 4 - 6):

- The first line is the selected configuration of default LIS-Net. The following lines are variations of hyper parameters.
- The column "#Params (M)" is the total number of parameters of the network with the corresponding configuration on that line.
- The "T" column is the time in seconds to complete an epoch training on Google Speech Command dataset v2, 35 commands (51094 images) using 04 CPU and 01 GPU 1080ti.
- Val_ACC and Test_ACC columns are validation and test accuracy respectively with the percentage values from 0 to 100.

a) Impact of Base parameters

Table 4
Impact of filters via B_P.

#	B_P	#Params (M)	T(s)	Val_ACC (%)	Test_ACC (%)
1	48	5.23	157	96.49	97.00
2	32	3.1	128	96.33	96.65
3	64	8.18	185	96.31	96.61

Table 5
Impact of G_R.

#	N_{core}	G_R	#Params (M)	T(s)	Val_ACC (%)	Test_ACC (%)
1	1,2,3,4	1,2,4,8	5.23	157	96.49	97.00
2	1,2,3	1,2,4	2.22	140	96.01	96.53
3	1,2,3,4,5	1,2,4,8,16	16.83	175	96.11	96.81
4	1,2,3,4,5,6	1,2,4,8,16,32	62.46	250	95.70	96.44

Table 6
Effects of different network structures based on N_{core}.

#	N_{core}	#Params(M)	T(s)	Val_ACC(%)	Test_ACC(%)
1	1,2,3,4	5.23	157	96.49	97.00
2	4,4,4,4	5.4	345	96.39	96.97
3	3,3,3,3	5.26	271	96.33	96.61
4	5,5,5,5	5.55	452	96.29	96.80
5	4,3,2,1	5.15	451	96.36	96.81
6	5,4,3,2	5.23	395	96.27	96.80
7	7,6,5,4	5.6	618	96.74	96.99

With $N_{core} = [1, 2, 4, 8]$, the size of the model will change when B_P varies, leading to the change of the result (Table 4). In the experiments, B_P is selected with values less than and greater than 48. For the case of choosing any value, the experiments are conducted as shown in Table 5. With the parameter $B_P = 32$, it can be seen that the network and the time T are smaller, and the result is slightly lower. With $B_P = 48$, the network grows, execution time is also slower, and the results are worse.

b) Impact of the network Growth rate (G_R)

In order to get different G_R and maintain the network architecture, the number of LIS-Blocks (N_{core}) must be changed. With $B_P = 48$, network filters are varied with the change of G_R, and the obtained results are shown in Table 5. Through the results, it indicates that the result is a little bit smaller by adding an LIS-Block. Adding more or reducing LIS-Blocks, the results of test accuracy decreases.

c) Impact of different network structure based on N_{core}

For $B_P = 48$ and $G_R = [1,2,4,8]$, the comparison of obtained results is shown in Table 6. It can be observed that with the parameters at line 7, the result of validation accuracy is better, and test accuracy at lines 2 and 7 is equivalent to the original LIS-Net. That means that there still has the space to optimize the network.

4.6.5. Discussion

For the specific problem of SCR using Google Speech Command dataset, LIS-Net has shown the good performance and achieves new state-of-the-art accuracy on all 12, 20 and 35 commands with fast prediction time and small total parameters. From the obtained results of this study, it is feasible to use CNN to train speech data by selecting the suitable model and architecture. The CNN architecture is selected to build the network because CNN can take full advantage of GPU parallel computing capability which is faster than the sequential RNN architecture. Furthermore, by evaluating the results, LIS-Net achieves higher test accuracy than baseline models in every variation case. Accordingly, it still has the space to improve and change the network by varying network parameters.

5. Conclusions

In this study, to solve the SCR problem a new network architecture, so called “LIS-Net”, is proposed for increasing accuracy and the predicting speed as well as minimizing the model parameters. In this design, LIS-Net inherits the simplicity of Standard Connectivity by stacking LIS-Block blocks together. In each LIS-Block, the stacked LIS-Core has a transition layer to change the size of the feature to make hyper-parameter customization. In specific problems, the model will be more flexible. In the design of LIS-Core, there is inheritance of ResNet shortcut in the style of Xception. It reduces the number of layers compared to Inception by inheriting a part of DenseNets design while remaining equivalent learning ability. The most outstanding of the network is the simplification of LIS-Core designed by 3 searching areas 1×1, 3×3 and 5×5 inherited for local features. From there, it can be

seen as cross-channel correlations and spatial correlations features mapping, thus reducing the number of parameters of the network while still improving the accuracy. On the other hand, the number of LIS-Blocks in the model, LIS-Cores in a LIS-Block, and the filter parameters in a LIS-Core are parameterized for easy customization for each specific application problem. LIS-Net has been applied to the SCR problem. Results of the study were performed on the Google Speech Command v1 / v2 dataset. To evaluate the model, the comparisons with powerful models of similar architecture such as ResNet, ResNext, DenseNets, InceptionResNet and Xception on GSC v2 on all 12, 20 and 35 keywords have been made, and LIS-Net has achieved state-of-the-art accuracy results (97% vs 96.2%), and LIS-Net's footprint is also the smallest of these models. To compare the model with specialized models designed for SCR, LIS-Net was compared with the CNN group (ConvNet, Tpool2, TDNN, DS-CNN, HD-CNN), RNN group (BiLSTM, BiGRU), and The group combines architectures (DenseNets - BiGRU, DenseNets - LSTM, Attention RNN) on GSC v1 and v2 (12 keywords). Through the investigation, LIS-Net has achieved state-of-the-art accuracy results (98.1% vs 97.5%). Besides, the effects of network parameters on the predicted results, such as Base parameters (B_P), the network Growth rate (G_R) and different network structures based on number of LIS-Cores (N_{core}), are also studied. The results are still higher than those of the baseline models with the same architecture. Moreover, the standard model still has the room for improvement. However, in this study, the results on other data sets for SCR, other areas such as speech recognition or image tasks such as object detection / recognition has not yet been invested or studied to comprehensively evaluate the new model. This suggests promising future works and it can be refined structures in automated ways on the standard model.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

In this article, we would like to especially thank to R&D department of GRG banking company, the National Nature Science Foundation of China 61571192 and Thai Nguyen University of Technology (TNUT), Thai Nguyen, Vietnam for supporting us in the experimental process.

References

Andra, M.B., Usagawa, T., 2017. Contextual keyword spotting in lecture video with deep convolutional neural network. 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS). IEEE, pp. 198–203. https://doi.org/10.1109/ICACSIS.2017.8355033.

de Andrade, D. C., Leo, S., Viana, M. L. D. S., Bernkopf, C., 2018. A neural attention model for speech command recognition. arXiv:1808.08929

Audhkhasi, K., Rosenberg, A., Saon, G., Sethy, A., Ramabhadran, B., Chen, S., Picheny, M., 2017. Recent progress in deep end-to-end models for spoken language processing. IBM J. Res. Dev. 61 (4/5), 2:1–2:10. https://doi.org/10.1147/JRD.2017.2701207.

Bhayani, M., Patel, M., Bhatt, C., 2016. Internet of things (IoT): in a way of smart world. Advances in Intelligent Systems and Computing, vol. 438, pp. 343–350. https://doi.org/10.1007/978-981-10-0767-5_37.

Chen, G., Parada, C., Heigold, G., 2014. Small-footprint keyword spotting using deep neural networks. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4087–4091. https://doi.org/10.1109/ICASSP.2014.6854370.

Chollet, F., 2017. Xception: deep learning with depthwise separable convolutions. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800–1807. https://doi.org/10.1109/CVPR.2017.195. arXiv:1610.02357v2.

Chorowski, J., Bahdanau, D., Cho, K., Bengio, Y., 2014. End-to-end continuous speech recognition using attention-based recurrent NN: first results. arXiv:1412.1602.

Fourniols, J.-Y., Nasreddine, N., Escriba, C., Acco, P., Roux, J., Soto-Romero, G., 2018. An overview of basics speech recognition and autonomous approach for smart home IoT low power devices. J. Signal Inf. Process. 9 (4), 239.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90. arXiv:1512.03385.

Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q., 2016. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv:1608.06993.

Hwang, K., Lee, M., Sung, W., 2015. Online keyword spotting with a character-level recurrent neural network. arXiv:1512.08903.

Jaderberg, M., Simonyan, K., Zisserman, A., Others, 2015. Spatial transformer networks. Advances in Neural Information Processing Systems, pp. 2017–2025.

Kim, T., Lee, J., Nam, J., 2018. Sample-level CNN architectures for music auto-tagging using raw waveforms. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 366–370.

Li, L., Wang, J., Li, J., Ma, Q., Wei, J., 2019. Relation classification via keyword-attentive sentence mechanism and synthetic stimulation loss. IEEE/ACM Trans. Audio SpeechLang. Process. 27 (9), 1392–1404. https://doi.org/10.1109/TASLP.2019.2921726.

Lin, M., Chen, Q., Yan, S., 2013. Network in network. arXiv:1312.4400.

Liu, B., Qin, H., Gong, Y., Ge, W., Xia, M., Shi, L., 2018. EERA-ASR: An energy-efficient reconfigurable architecture for automatic speech recognition with hybrid DNN and approximate computing. IEEE Access 6, 52227–52237. https://doi.org/10.1109/ACCESS.2018.2870273.

Liua, B., Wang, Z., Fan, H., Yang, J., Liua, B., Zhu, W., Huang, L., Gong, Y., Ge, W., Shi, L., 2019. EERA-KWS: A 163 TOPS/W always-on keyword spotting accelerator in 28 nm CMOS using binary weight network and precision self-adaptive approximate computing. IEEE Access 1. https://doi.org/10.1109/ACCESS.2019.2924340.

Puri, V., Jha, S., Kumar, R., Priyadarshini, I., Hoang Son, L., Abdel-Basset, M., Elhoseny, M., Viet Long, H., 2019. A hybrid artificial intelligence and internet of things model for generation of renewable resource of energy. IEEE Access 7, 111181–111191. https://doi.org/10.1109/ACCESS.2019.2934228.

Sainath, T.N., Parada, C., 2015. Convolutional neural networks for small-footprint keyword spotting. In: Proceedings INTERSPEECH, pp. 1478–1482.

Samie, F., Bauer, L., Henkel, J., 2019. From cloud down to things: an overview of machine learning in internet of things. IEEE Internet Things J. 6 (3), 4921–4934. https://doi.org/10.1109/JIOT.2019.2893866.

Speech, B., Group, T., 2016. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin Speech Interfaces : A New / Old Input Paradigm. Technical Report. https://doi.org/10.1145/1143844.1143891. arXiv:1512.02595v1.

Sun, M., Raju, A., Tucker, G., Panchapagesan, S., Fu, G., Mandal, A., Matsoukas, S., Strom, N., Vitaladevuni, S., 2017. Max-pooling loss training of long short-term memory networks for small-footprint keyword spotting. 2016 IEEE Workshop on Spoken Language Technology, SLT 2016 - Proceedings, pp. 474–480. https://doi.org/10.1109/SLT.2016.7846306. arXiv:1705.02411v1.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Inception-v4, Inception-ResNet and the impact of residual connections on learning Christian. Popul. Health Manage. 18 (3), 186–191. https://doi.org/10.1089/pop.2014.0089. arXiv:1409.4842.

Tachibana, H., Uenoyama, K., Aihara, S., 2018. Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 4784–4788.

Tang, R., Lin, J., 2017a. Deep residual learning for small-footprint keyword spotting. arXiv:1710.10361.

Tang, R., Lin, J., 2017b. Honk: a PyTorch reimplementation of convolutional neural networks for keyword spotting. arXiv:1710.06554. 10.1145/nnnnnnn.nnnnnnn

Tang, R., Wang, W., Tu, Z., Lin, J., 2017. An experimental analysis of the power consumption of convolutional neural networks for keyword spotting. arXiv:1711.00333.

Warden, P., 2018. Speech commands: a dataset for limited-vocabulary speech recognition. arXiv:1804.03209.

Yan, Z., Zhang, H., 2015. HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2740–2748.

Yusuf, B., Gundogdu, B., Saraclar, M., 2019. Low resource keyword search with synthesized crosslingual exemplars. IEEE/ACM Trans. Audio SpeechLang. Process. 27 (7), 1126–1135. https://doi.org/10.1109/TASLP.2019.2911164.

Zeng, G., He, Y., Yu, Z., Yang, X., Yang, R., Zhang, L., 2016. InceptionNet/GoogLeNet - going deeper with convolutions. Cvpr 91 (8), 2322–2330. https://doi.org/10.1002/jctb.4820. arXiv:1409.4842.

Zeng, M., Xiao, N., 2019. Effective combination of DenseNet and BiLSTM for keyword spotting. IEEE Access 7, 10767–10775. https://doi.org/10.1109/ACCESS.2019.2891838.

Zhang, X., Yao, L., Zhang, S., Kanhere, S., Sheng, M., Liu, Y., 2019. Internet of things meets brain-computer interface: a unified deep learning framework for enabling human-thing cognitive interactivity. IEEE Internet Things J. 6 (2), 2084–2092. https://doi.org/10.1109/JIOT.2018.2877786.

Zhang, Y., Suda, N., Lai, L., Chandra, V., 2017. Hello edge: keyword spotting on microcontrollers. arXiv:1711.07128.

Zhou, J., Wang, Y., Ota, K., Dong, M., 2019. AAIoT: Accelerating artificial intelligence in IoT systems. IEEE Wirel. Commun. Lett. PP (c), 1. https://doi.org/10.1109/LWC.2019.2894703.

Zhou, S., Xu, S., Xu, B., 2018. Multilingual end-to-end speech recognition with a single transformer on low-resource languages. arXiv:1806.05059, 2–6.

Nguyen Tuan Anh, He is a PhD student at the School of Electronic and Information Engineering, South China University of Technology (SCUT), Guangzhou, China. He received a Bachelor degree from the Thai Nguyen University Of Technology (TNUT, 2005), and a Master degree from Thai Nguyen University of Information Technology and Communications (ICTU, 2009), Thai Nguyen, Vietnam. His major research interests in the field of Speech Recognition, Keyword Spotting, Facial Recognition, Deep Learning, Embedded system, and Internet of Things (IoT). He is very eager to apply algorithms into real products

YONGJIAN HU, He received the Ph.D. degree in communication and information systems from the South China University of Technology, in 2002, where he is currently a Full Professor with the School of Electronic and Information Engineering. From 2011 to 2013, he was a Marie Curie Fellow with the Department of Computer Science, University of Warwick, U.K. From 2006 to 2008, he was a Research Professor with the Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), South Korea. From 2005 to 2006, he was a Research Professor with the School of Information and Communication Engineering, SungKyunKwan University, South Korea. From 2000 to 2004, he was with the Department of Computer Science, City University of Hong Kong, four times as a Research Assistant, a Senior Research Associate, and a Research Fellow, respectively. He has published more than 70 peer-reviewed papers. His research interests include information hiding, multimedia security, and machine learning. He is a Senior Member of the Chinese Institute of Electronics (CIE) and a Senior Member of the China Computer Federation (CCF).

Qian-Hua He, He received the Ph.D degree in communication engineering from South China University of Technology in 1993, where he is currently a Full Professor with the School of Electronic and Information Engineering, the B. S. Degree in physics from Hunan Normal University in 1987, and the M. S. Degree in medical instrument engineering from Xian Jiaotong University in 1990. From 2007.11 to 2008.10, he was with University of Washington in Seattle USA as a visiting scholar. From 1994 to 2001, he was with the Department of Computer Science, City University of Hong Kong, four times as a Research Assistant, a Senior Research Assistant, and a Research Fellow, respectively. His research interests include spoken term detection, audio event detection, speech coding, multimedia retrieval and digital audio forensic. He is a Senior member of IEEE.