Speech Recognition using Deep Neural Networks

Jungwon Seo
j.seo@stud.uis.no
University of Stavanger
Stavanger, Norway

ABSTRACT

This paper presents a speech recognition system that classifies the single word audio file. The system has built after testing the several deep neural network model such as feed forward neural network, convolutional neural network and recurrent neural network. The audio file is transformed into STFT spectrogram to reformulate the task as an image recognition process. Additionally, several data augmentation has applied including a state-of-the-art spectrogram augmentation. The dataset is from Kaggle speech recognition competition, and the accuracy scored from Kaggle has considered as final validation. Using Resnet18 and several optimizations achieved 78.234% of Kaggle accuracy.

KEYWORDS

speech recognition, deep neural networks, convolutional neural networks, data augmentation

ACM Reference Format:
Jungwon Seo. 2019. Speech Recognition using Deep Neural Networks. In *Proceedings of Data-Mining Project (DAT550).* UiS, Stavanger, Rogaland, Norway, 6 pages. https://github.com/uis-dat550-spring19/Jungwon-Group

1 INTRODUCTION

One of the accomplishments of Deep learning technology is that it has made more science industry more accessible, even if people are not experts in a particular field. Especially, acoustic speech recognition has been significantly improved over the conventional approach [6].

Furthermore, deep learning framework like Tensorflow and Pytorch has increased the accessibility for developers to tackle the various types of real-world problems. In 2017, Google introduced the audio processing functionality in Tensorflow and wanted to encourage developers to use it by hosting Kaggle speech recognition competition [1].

To build the speech recognition model, we manually implement five different types of deep neural networks.

- Feed Forward Neural Network (FNN)
- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
- Long Short Term Memory (LSTM)
- Bi-directional Long Short Term Memory (Bi-LSTM)

In this paper, we tackle the speech recognition task without using advanced signal processing technology. Moreover to find the most suitable approach we attempt to apply the various types of a deep neural network model. The paper flows with describing our step-by-step method to build a speech recognition model.

2 BACKGROUND

2.1 STFT spectrogram

In the signal processing area, there are several characteristics to analyze or process the signal such as amplitude, time and frequency. The Fourier transform (FT) which decomposes a signal into its frequencies level have escalated the signal processing research area. However, in speech recognition, standard Fourier transform has a critical limitation, because speech is not a constant signal; it changes with time [10]. Therefore, we need a different method that contains both time and frequency information; this is the reason that we decide to transform the audio file into The short-time Fourier transform (STFT) which contains both time and frequency domain [2].

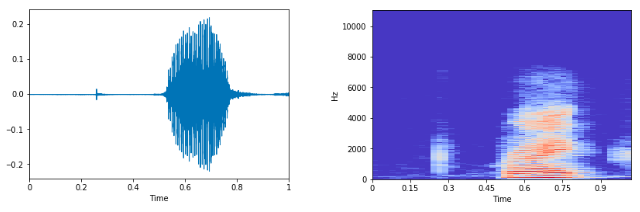


Figure 1: Left: amplitude-time, Right: STFT

As shown in Figure.1, there are two ways to visualize the audio signal. The left plot is showing the amplitude-time domain signal, and the right plot is showing STFT spectrogram. Both decomposed features can be used in this project; however, the amplitude does not always guarantee the steady behavior for the similar sound (or speech); it is more related to the volume of the sound. Therefore, we discard the wave plot option.

2.2 Deep Neural Network

The deep neural network (DNN), more precisely the multi-layer feed-forward neural network (FNN) published a couple of decades ago [11], starts to receive significant attention in recent years. Not specifically original DNN itself; however, since all modern the-state-of-art artificial intelligence is using deep learning to solve the real world problem, it is understandable to acknowledge the effect.

There are many options to choose for the classification task in supervised learning; however, unlike the simple image source such as handwriting digit images which is easily recognizable by human, the spectrogram does not show the distinct characteristic that we can instantly recognize. Therefore, we decide to use different types of deep neural networks which are showing promising performance in many pattern recognition tasks. However, to set the baseline model, we implement a simple feed-forward neural network(FNN) which consists of three fully connected layers with ReLU as a non-linearity between each layer. Cross-entropy loss is used as the loss function for all models, and the checkpoint of the model is saved every one epoch. To avoid the overfitting early stopping will be trigger when the higher validation loss than the lowest loss is calculated ten times for FNN and CNN and 20 times for the RNN models. Since we decide to use this FNN as a baseline, there is no hyper-parameter tuning for the simple FNN.

2.3 Convolutional Neural Network

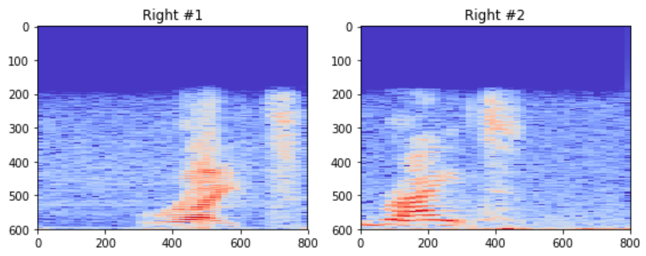


Figure 2: Two 'right' class spectrogram with different starting points

It is no exaggeration to say that the convolutional neural network (CNN) has advanced artificial intelligence decades ahead. The most well-known CNN was introduced by achieving significant accuracy from ImageNet classification in 2012 [8]. Regardless of CNN's popularity, there are several reasons that we decide to use CNN. Firstly, as we will explain later, the audio file is only one second long which means all the data can fit into similar image size. If the audio has a different length or longer length CNN might not be the strong candidate in this project. Secondly, even if it is short length speech, a one-word utterance can be located in many different timestamps as shown in Figure.2. Therefore, it is reasonable to choose CNN which is not affected by the location of the object as long as it sustains a similar pattern.

2.4 Recurrent Neural Network

RNN has proven to show promising performance over CNN, and sometimes it solved the problem which is difficult to be learned by CNN, especially, when the sequence or time is the crucial role in the data [4]. Even though the audio length is short, still time-domain information will be an essential role in this data. Therefore, we decide to test RNN as our candidate model which can learn meaningful information by feeding the frequency information into the network based on time. We test three different types of RNN. Simple RNN, LSTM, and BI-LSTM are tested having the same configuration of FNN mentioned above except the RNN's specific configuration.

3 EXPERIMENTS

3.1 Task Explanation

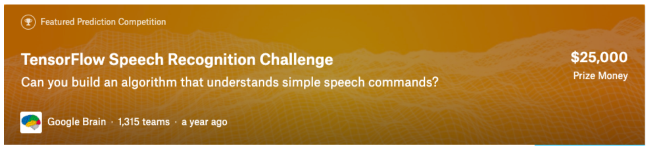

Featured Prediction Competition
TensorFlow Speech Recognition Challenge
Can you build an algorithm that understands simple speech commands?
Google Brain · 1,315 teams · a year ago
$25,000
Prize Money

Figure 3: Kaggle Speech Recognition Challenge (https://www.kaggle.com/c/tensorflow-speech-recognition-challenge**)**

Our task is classifying different types of utterance which is hosted by Google Brain Team in Kaggle shown in Figure.3. There are 12 classes that we have to classify: yes, no, up, down, left, right, on, off, stop, go, silence and unknown using offered audio dataset.

3.2 Dataset Description

Right	**Eight**	**Cat**	**Tree**	**Bed**	**Happy**
2367	2352	1733	1733	1713	1742
No	**Wow**	**Nine**	**Left**	**Stop**	**Three**
2375	1745	2364	2353	2380	2356
Bird	**Zero**	**Seven**	**Up**	**Marvin**	**Two**
1731	2376	2377	2375	1746	2373
Six	**Yes**	**On**	**Five**	**Off**	**Four**
2369	2377	2367	2357	2357	2372
Dog	**One**	**Down**	**Go**	**Sheila**	**House**
1746	2370	2359	2372	1734	1750
Background_noise					
6					

Table 1: The number of training audio files per class

The data is from [12], which consists of 64,727 audio files for training and 158,538 files for the validation. Unlike the target classes, there are more classes in the training dataset as shown in Table.1. All the audio files are 'WAV' format and most of them are approximately one second long except the background sound. In the background class, there are six audio files that the competition host recommends us to use to augment the data and to train for the silence class.

The classification task can be reconsidered as shown in Figure 4. Among 31 classes only ten classes are directly trainable, and the other 21 classes should be used to predict either 'unknown' or 'silence' class. We decide to train each of the classes to avoid the data imbalance that can occur by merging those 21 classes as one class. And also we assume the 'silent' class will be similar to 'background_noise' class.

3.3 Data Setting

As shown in Figure.5, we use STFT spectrogram as our default input source. We generate 600 by 800 pixels spectrogram image per audio file to transform the task from speech recognition into image recognition problem. During the training process, the test data set is randomly chosen from the training dataset. 10% of the training data has

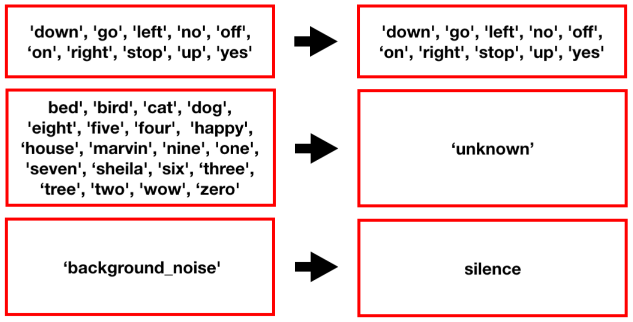

'down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes'
'down', 'go', 'left', 'no', 'off', 'on', 'right', 'stop', 'up', 'yes'
bed', 'bird', 'cat', 'dog', 'eight', 'five', 'four', 'happy', 'house', 'marvin', 'nine', 'one', 'seven', 'sheila', 'six', 'three', 'tree', 'two', 'wow', 'zero'
'unknown'
'background_noise'
silence

Figure 4: Actual classes for submission

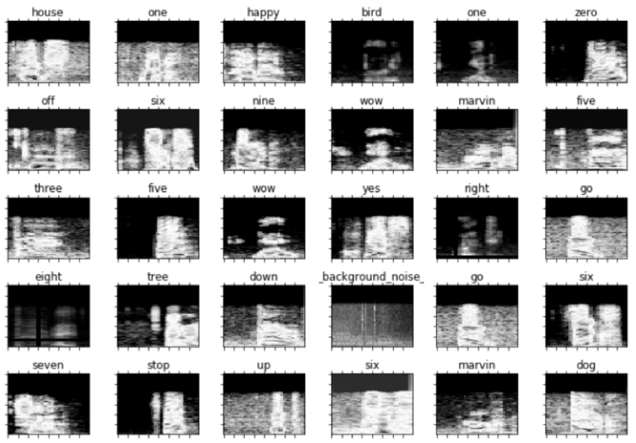


Figure 5: Sample spectrograms for each class

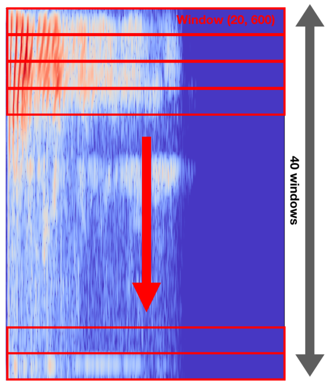
used for the validation. For the parameters to decompose the audio into STFT, we used default setting from the 'librosa' library which are $n\_fft = 2048$, $hop\_length = win\_length/4$, $win\_length = n\_fft$. These parameters are adjusted when we balanced the number of datasets later.

3.4 Approach and Implementation

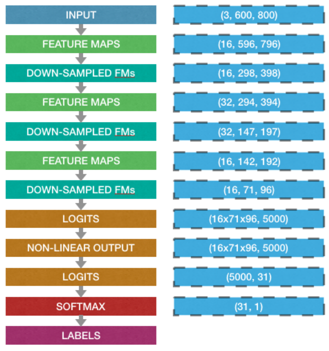
	FNN	RNNs
# of Layers	4	4
# of Input Nodes	800*600	600*20
# of Hidden Nodes	500	500
# of Outpu Nodes	31	31
# of Sequence	-	40

Table 2: Parameter information for each model

Our approach is building the baseline model first, and use an advanced version of the best performing model. Then we augment the dataset and tune the model parameters. We test five different deep neural network models and choose the best performing model in terms of training time and accuracy. We use Kaggle's accuracy to judge the real performance of the model.



(a) RNN window mechanism



(b) CNN architecture

Figure 6: Window mechanism for RNNs and the architecture of CNN

The architecture for both FNN and RNNs including LSTM and Bi-LSTM are similar as shown in Table.2 except for the sequence dimension. In RNN, the data has to be fed into RNN per time (in this case one pixel out of 800 pixels). However, there are two challenges while training RNN models. First of all, if we feed the data every 600 by 1 pixel the learning time is unacceptably slow, we could not observe any improvement in terms of accuracy. Therefore, we use a window mechanism for our RNN models which consists of 600 by 40 pixels. Secondly, if we do not carefully consider the orientation of the image and the way memory access the matrix data, it is often mistakenly fed on top to bottom. This behavior is not relevant to the FNN or CNN models since they are not considering time series; however, this is a crucial issue for RNN models. Therefore as shown in Figure.6a, before we feed into the model, we transpose the original image by changing time and frequency axis.

The CNN architecture shown in Figure.6b, consists of three convolution layers with filter size 5x5, 5x5 and 6x6 respectively. Additionally, two fully-connected layers are used. Our intention is only finding the most suitable network; therefore, there is no deeper consideration while deciding the layer parameter.

To implement all the models, we use Pytorch, and the training process is run in the cluster using the Tesla V100 Data Center GPU. All the source code can be found in the Github repository.

3.5 Training and selecting the best performing model

	FNN	CNN	RNN	LSTM	BILSTM
Training Acc. (%)	58	**87**	80	82	85
Kaggle Acc. (%)	48.7	65.2	58.5	63.7	**65.7**
# of Epochs	34	**6**	161	165	175
Training Time (hours)	**1.5**	1.6	14	15	17.7

Table 3: Result of candidate models

As shown in Figure.7, the FNN does not show enough performance to be considered. Three types of the recurrent neural network

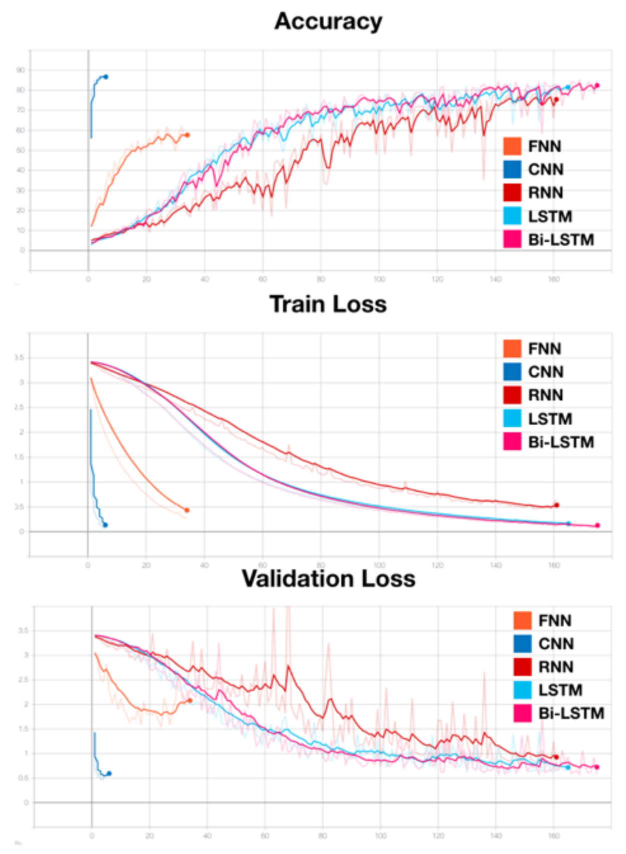


Figure 7: Accuracy and loss for each model

perform reasonably in terms of accuracy; however, the training time is much longer than the CNN while showing a similar or less accurate performance. We assume that in the longer speech recognition it is unavoidable to use RNN, also even in this task RNN models may perform better. However, due to the limitations mentioned above, we choose CNN to be our baseline model.

3.6 Using Advanced Deep Neural Network

We decide to use advanced DNN model has been published. The reason we choose Resnet18 is which is consuming relatively less memory compared to other advanced models as shown in Figure.??. Moreover, since our approach is fast training and validating, we select the model that requires less resource in terms of memory and time.

After training using early stopping with 10 times of patient, the training accuracy reaches 96% and the Kaggle accuracy shows 78.042% which is 13% higher than the baseline model. [5]

3.7 Balancing the dataset

As shown in Figure.1, the number of dataset per class is imbalanced. Notably, the number of background noise dataset is only six therefor from our predicted file; it does not classify any test data as a silence as shown in table 4. Therefore, we decide to augment the data using different window length and hop length while we are generating the STFT spectrogram. We make all the class have 2,500 spectrograms with randomly selected audio set from each label. Therefore we can generate the pixel-level different spectrogram from the same audio file as shown in Figure.9.

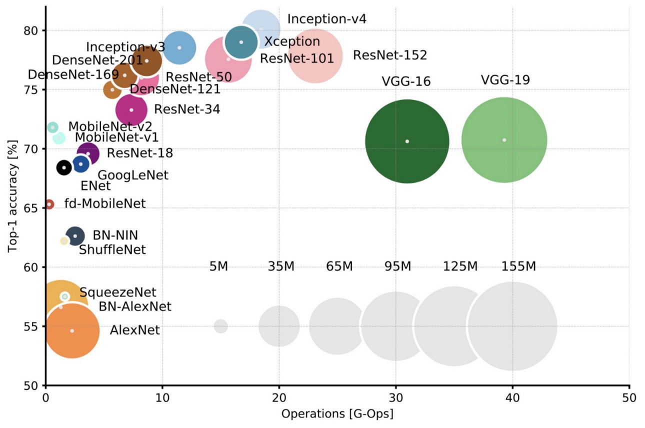


Figure 8: Top1 vs. operations, size ∝ parameters from [3]

	Imbalanced Dataset	Balanced Dataset
Down	4576	5731
Go	7093	6425
Left	5921	7752
No	9015	6805
Off	5907	6541
On	10443	7125
Right	6136	5551
Stop	6036	6314
Up	15450	8999
Yes	5562	5071
Unknown	82399	84645
Silence	0	7579

Table 4: Classification result before and after balancing the dataset

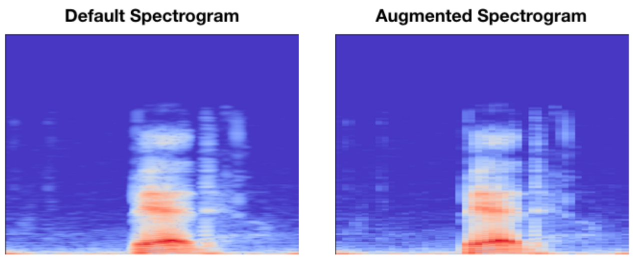


Figure 9: Newly generated spectrogram using different hop length and window length

Figure 10: Prediction result before and after

After train the same model with additional data, the Kaggle accuracy result shows 78.234% which is slightly improved score compared to the result using imbalanced data.

3.8 Parameter tuning

Since we are using an established network model (resnet18), instead of changing the model, we lower the learning rate to improve our model. We decrease the learning rate from 0.1 to 0.01 and over-train our network. It shows little improvement from the private score (79.278%) in Kaggle which is scored by part of the test data (50% in general). However, it shows the underperforming score from the public score (78.125%) which is scored by another part of the data. It is not a considerable difference, and we suspect that our model is overfitted into our training dataset.

3.9 Adding Dropout

As we mention above, we suspect that the reason that our model shows the big difference between the training accuracy and Kaggle accuracy is our model is overfitted into our training dataset. There are common ways to overcome this challenge. One is using more training dataset, and the other one is using 'dropout'.

It is widely known that using relatively small data in a large neural network model, often occurs the overfitting problem. To solve this issue, we use the 'dropout' method which is randomly dropping half of the feature detectors [7].

```
model = models.resnet18(pretrained=False,
  ↪  num_classes=output_dim)
num_ftrs=model.fc.in_features
model.fc =
  ↪  nn.Sequential(nn.Dropout(0.5),nn.Linear(num_ftrs,
  ↪  output_dim))
```

Figure 11: Pytorch dropout adaption on Resnet18

The dropout functionality can be easily adapted in our current Pytorch implementation as shown in Figure.11. The dropout rate is set to 0.5.

3.10 Applying the-state-of-art Data Augmentation

Unlike the other image recognition problem, our speech recognition is not easy to use the common approach to augment the data such as scaling, rotating and cropping. One of the main reason is as long as we are using STFT spectrogram, there is no rotated, scaled and cropped spectrogram having the same label. That is why we decide to adjust the hop and window size to generate new spectrogram from the same audio data.

Recently, a new augmentation method has been introduced that can be easily applied to the spectrogram augmentation. Base on [9], spectrogram data can be augmented by warping in time direction (vertical) and frequency direction (horizontal) as shown in Figure.12. We apply this method to all the existing training spectrogram and double the amount of training dataset.

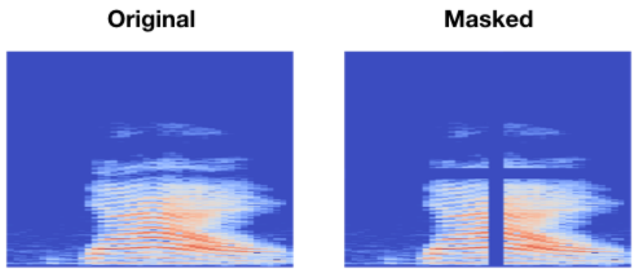


Figure 12: Augmented spectrogram using masks

The final submission achieve the 77.549% which is lower than the result before adding dropout and augmenting the data.

3.11 Discussion

Despite adding dropout and augmenting the dataset could not achieve the higher accuracy. Therefore, we suspect that there was a problem with the preprocessing.

Using advanced DNNs contributes, but is not a core factor in improving the accuracy. Based on other participants' discussion, there are several factors to achieve around 90% accuracy. For instance, there are many hundreds of miss-labeled data in the training dataset. One of the top-ten ranked participants manually relabeled or removed the mislabeled the incorrect dataset. Moreover, many participants who used spectrogram as an input source experienced higher accuracy by using high-resolution spectrogram over 4000x4000 pixels. Also, mixing audio files can improve detectability for the 'unknown' class. Lastly, training 'unknown' class after merged all the redundant classes may improve accuracy.

4 CONCLUSIONS

In this paper, we tested different types of deep neural networks to build the speech recognition model which was hosted in Kaggle competition. Specifically, we experimented Resnet18 with applying a state-of-the-art spectrogram augmentation method. Even though we could not achieve the satisfiable accuracy, we presented the step-by-step approach to improve the accuracy from 48.7% to 78.234% which is 639th result out of 1,315 teams.

REFERENCES

[1] 2017. Simple Audio Recognition | TensorFlow Core | TensorFlow. https://www.tensorflow.org/tutorials/sequences/audio_recognition

[2] Jonathan Allen. 1977. Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25, 3 (1977), 235–238.

[3] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. 2016. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678* (2016).

[4] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[6] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* 29 (2012).

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[9] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. *arXiv preprint arXiv:1904.08779* (2019).

[10] Ervin Sejdić, Igor Djurović, and Jin Jiang. 2009. Time–frequency feature representation using energy concentration: An overview of recent advances. *Digital signal processing* 19, 1 (2009), 153–183.

[11] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. 1997. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems* 39, 1 (1997), 43–62.

[12] Pete Warden. 2017. Speech Commands: A public dataset for single-word speech recognition. *Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz* (2017).