Speech Recognition Based on Tensorflow Speech Commands Datasets

Pengluo Wang
ECE department
pew067@eng.ucsd.edu

Zhipeng Yao
Mathematics department
zhy212@ucsd.edu

Yue Yang
ECE department
yuy287@eng.ucsd.edu

Yuyi Tan
ECE department
yut080@eng.ucsd.edu

Abstract

One prevalent approach to achieve sequence to sequence learning is the recurrent neural network (RNN), which maps an input sequence to a length-changing output sequence. Noticed that computations of CNN over all elements can be fully parallelized during training, which makes it more compatible with GPU. Also, compared to recurrent neural networks, convolutional neural networks is much easier to optimize since the number of non-linearities is fixed and independent of the input length. Besides, it builds a hierarchical representation of the input words which enables it to capture long-range dependencies in a much easier manner, similar to the tree-structured analysis of linguistic grammar formalisms. Hence, we built our architecture entirely on CNN instead of RNN. We also testified that using gated linear units (GLU) will lead to better performance. One of the advantages is that it eases gradient propagation.

1. Introduction

1.1. Background

Sequence to sequence learning has been successfully applied in many engineering fields such as machine translation, speech recognition and etc. The dominant approach to achieve sequence to sequence learning is using bidirectional recurrent neural networks on both input and output sides. The input sequence is encoded with a series of bi-directional RNNs and a variable length output is generated by another set of decoder RNNs. Both set of RNNs interface via a soft-attention mechanism.

Convolutional neural networks are not as common used as recurrent neural networks in sequence to sequence learning field. Compared to recurrent neural networks, convolutions represent contexts with fixed size. In contrast to recurrent neural networks, which maintains a hidden state to restore the entire past, convolutional neural networks do not depend on computations of the previous time step and thus parallelization is allowed over all elements within a sequence.

Recently, research has been done to apply convolutional neural network to sequence learning. Bradbury et al. (2016) introduced recurrent pooling between a succession of convolutional layers. Kalchbrenner et al. (2016) tackled neural translation without attention. However, neither of these two approaches demonstrated remarkable improvements over other state-of-art results on large benchmark datasets. Meng et al. (2015) explored gated convolutions for machine translation, but their model was learned via a small dataset and used in tandem with a traditional counting-based model. Gehring et al. (2016) built a partially convolutional architecture which was well-performed on large datasets, but the decoder still used recurrent networks.

1.2. Convolutional neural networks

Sequence to sequence learning has been successfully implemented in many tasks such as speech recognition. And the main approach is to use recurrent neural networks (RNN) to date encode the input sequence. However, here we construct an entirely convolutional neural networks model which has a lot of advantages compared to the RNN.

Convolutional neural networks allow parallelization over every element in a sequence since it doesn't depend on the computations of the previous time step. Inputs to a convolutional network are fed through a constant number of kernels and non-linearities.

Compared to the recurrent networks which model the chain structure, Multi-layer convolutional neural networks create hierarchical which provides a shorter path to capture long-range dependencies, the hierarchical structure representations over the input sequence in which nearby input elements interact at lower layers while distant elements interact at higher layers.

Inputs to a convolutional network are fed through a constant number of kernels and non-linearities, compared to up to *n* operations and non-linearities to the first word and only a single set of operations to the last word of recurrent networks. For example, a feature representation capturing

relationships within a window of n words were obtained by applying only $O(n/k)$ convolutional operations for kernels of width k, whereas recurrent neural networks need a linear number $O(n)$. Fixing the number of nonlinearities applied to the inputs also eases learning.

1.3. Gated linear unit

LSTMs enable long-term memory via a separate cell controlled by input and forget gates. Inspired by this idea, researchers came up with a new gate mechanism which possess solely output gates and allow the network to control what information should be propagated through the hierarchy of layers. The hidden layers are computed as:

$$h_l(\mathbf{X}) = (\mathbf{X}*\mathbf{W}+\mathbf{b}) \oplus \sigma(\mathbf{X}*\mathbf{V}+\mathbf{c}) \quad (1.1)$$

The output of each layer is a linear projection $\mathbf{X}*\mathbf{W}+\mathbf{b}$ modulated by the gates ($\mathbf{X}*\mathbf{V}+\mathbf{c}$). Similar to LSTMs, these gates multiply each element of the matrix $\mathbf{X}*\mathbf{W}+\mathbf{b}$ and control the information passed on in the hierarchy. This gating mechanism were dubbed Gated Linear Units (GLU). Figure 1 shows the architecture of the gated convolutional network for language modeling:

Gated linear units are a simplified gating mechanism based on the work of Dauphin & Grangier (2015) for nondeterministic gates, they can reduce the vanishing gradient for deep architectures by providing a linear path for the gradients while retaining non-linear capabilities. Through the experiments, we find gated linear units allow for faster convergence to better perplexities and gated linear units achieve higher accuracy and converge faster than the LSTM-style gating of Oord et al.

2. Data

2.1. Data set

We report our results on Tensorflow Speech Commands Datasets (TSCD for short), which consists of 65,000 one-second WAVE audio files of people saying thirty different words. Due to the limitation of GPU memory in the server, we only used the training data in TSCD, 4/5 of which is used as training set and 1/5 as testing set. Besides, we used 1/5 of the training set as validation set (in a cross-validation sense). We verified that this is enough to train a well-performed model.

Figure 2 shows some commands contained in the dataset and some wave audio files contained in label “five”. Most of the length of the wave audio files is one second, however we can see from figure 2 that there exist some audio files shorter than one second. This will be mentioned in later section.

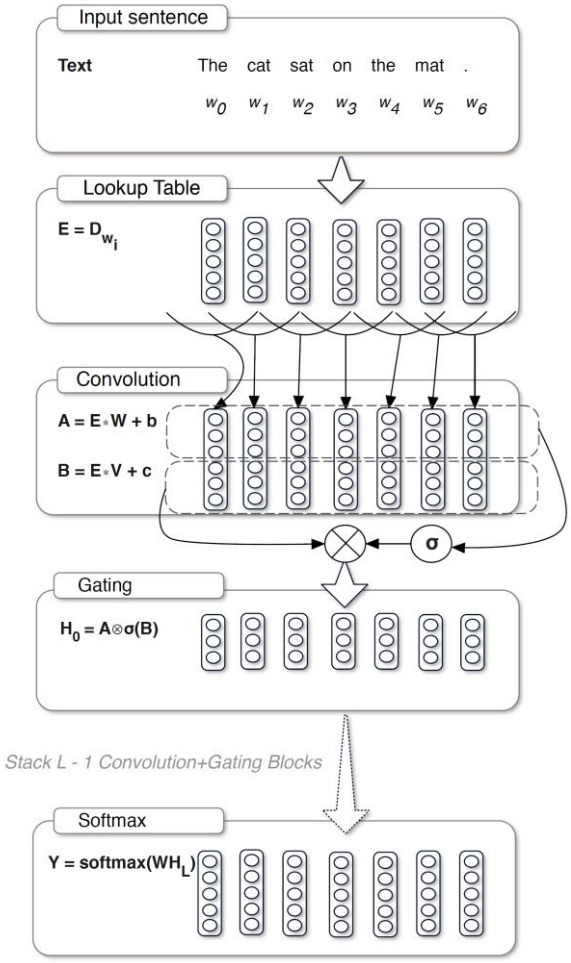

Figure 1. GLU diagram

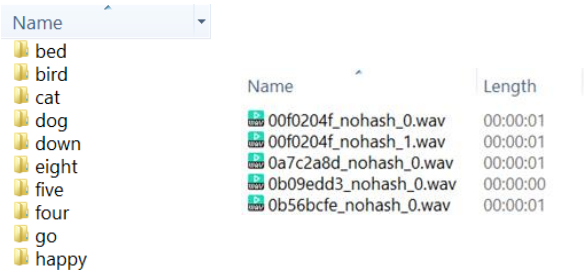

Figure 2. Data set directory (left) and waveform in directory “five” (right)

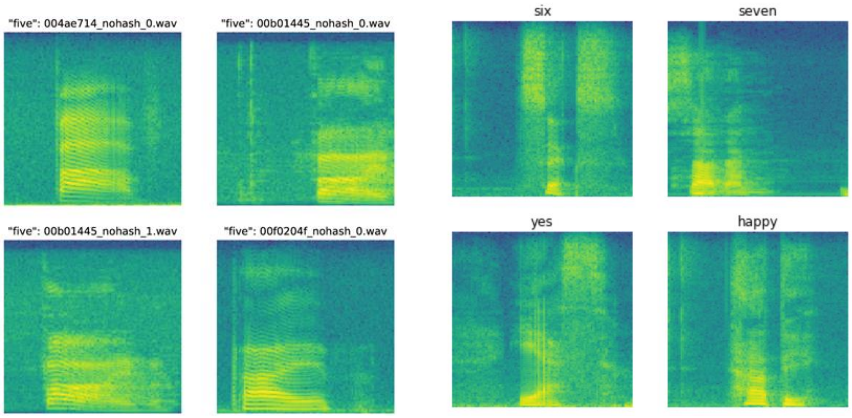

Figure 3. Spectrogram of input data "five" (left) and other four different commands (right)

As we can see from the figure 3 (left), spectrograms of the same command tend to have similar patterns, although the useful information is located in the different timeline (x-axis). Figure 3 (right) shows that different commands will result in different pattern of spectrograms. This is the basic principle why we can use CNN structure to train our model. By transforming the wave data to image data, we will not lose much information regarding to recognizing different commands.

2.2. Data preprocessing

As mentioned in the previous chapters, the preprocessing aims to transform waveforms in 1-d time domain into 2-d image frequency domain. This transform is implemented by calculating spectrograms of one-second wave audio files.

As we can see from the figure 2, spectrograms of the same command tend to have similar patterns, although the useful information is located in the different timeline (x-axis). Figure 3 shows that different commands will result in different pattern of spectrograms. This is the basic principle why we can use CNN structure to train our model. By transforming the wave data to image data, we will not lose much information regarding to recognizing unique commands.

2.3. Image reshaping

As mentioned in chapter 2.1, the audio files are not exactly one second long. This causes the fact that transformed spectrograms will not have the same shape. Typically, most of the spectrograms will be of shape (99, 161), however some may be (x, 161), where x can be smaller or larger than 99. So we introduce padding and intercepting techniques in order to guarantee all input data have same dimension.

① Padding 0 for short audio files.
② Intercepting for long audio files.
③ Finally, we get grayscale images of size (1 ×99 ×161).

3. Experiment and results

We propose an entirely convolutional architecture to model sequence learning. Our model is equipped with GLU (gated linear unit, Dauphin et al. 2016) and residual connections (He et al. 2015a). We evaluate our model using Tensorflow Speech Commands Datasets and compare its performance under several different situations.

3.1. Convolutional network structure

Figure 4 shows the network structure we used to train our model. There are four convolutional layers along with one linear layer used for classification. The third convolutional layer is used as a residual network. However since our model is not so deep, we did not observe any performance enhancement by using residual structure. We implement some popular tricks for convolutional networks such as batch normalization, leaky RELU, max pooling and average pooling. The parameters for each layer are shown in the above figures. And since it is a classification problem, we use cross entropy to calculate loss and then back propagate it.

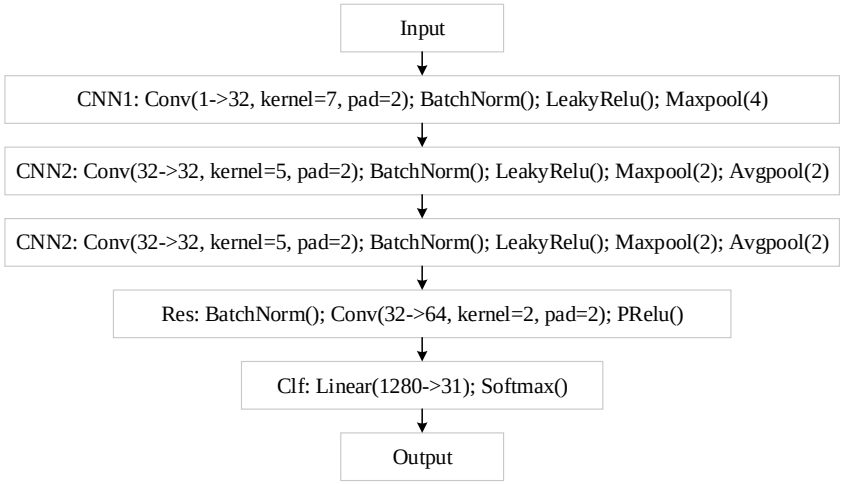

Figure 4. CNN structure

3.2. Different optimizers

First we train our model using different optimizers. Adam, SGD and RMSprop optimizations are tested on our model and plots of error and loss is shown below. The final classification error is shown in table 1. We can have the observation that bigger momentum values may lead to poorer performance. In both figures, we can see that when momentum = 0.9, the curve will oscillate a lot, making it harder to converge to the minimum. And for RMSprop optimization, best performance achieves when there is no momentum.

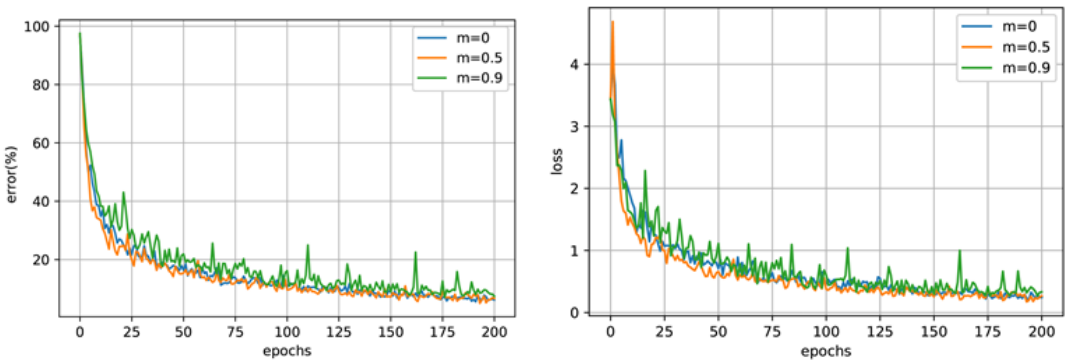

Figure 5. Classification error (left) and loss (right) vs. epochs for SGD optimization

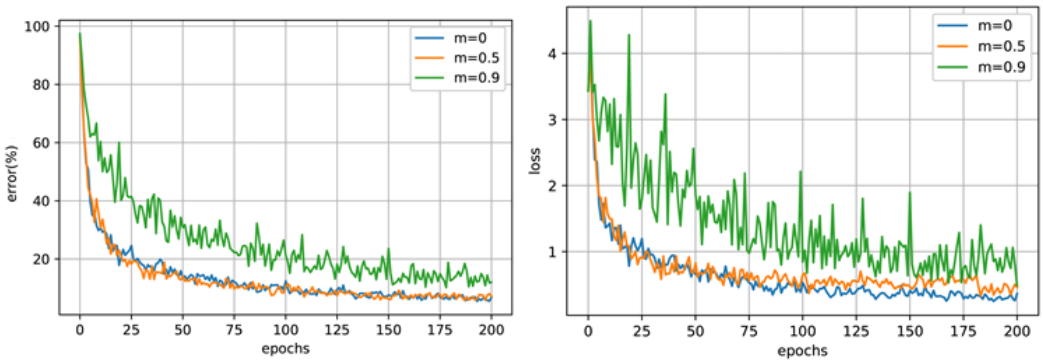

Figure 6. Classification error (left) and loss (right) vs. epochs for RMSprop optimization

Optimizer	Loss	Error
Adam	0.21	6.55%
SGD, m = 0	0.24	6.18%
SGD, m = 0.5	0.27	6.71%
SGD, m = 0.9	0.33	7.46%
RMSprop, m = 0	0.36	6.82%
RMSprop, m = 0.5	0.46	7.23%
RMSprop, m = 0.9	0.49	12.05%

Table 1. Final classification error and loss for different optimization techniques

3.3. GLU activation

From figure 7, we can have the observation that GLU activation will enhance the performance. It improves the classification accuracy and also converges a little faster than non-GLU models. Final classification error and loss for both networks are shown in table 2.

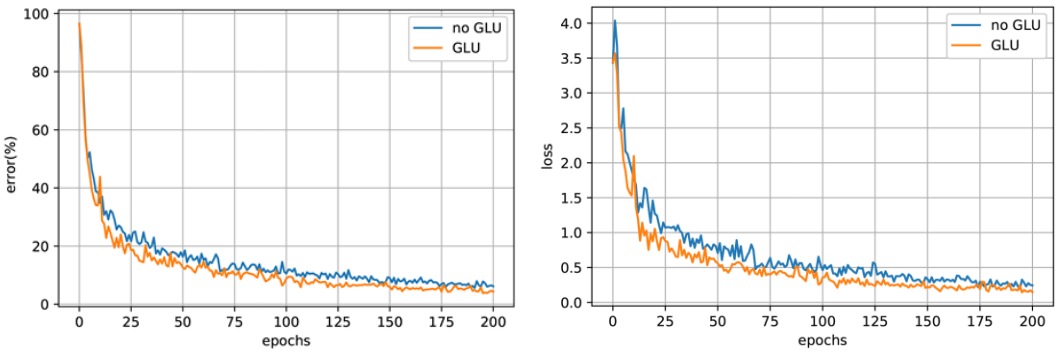

Figure 7. Classification error (left) and loss (right) vs. epochs for GLU and non-GLU network

Optimizer	Loss	Error
GLU	0.15	4.34 %
non-GLU	0.24	6.18%

Table 2. Final classification error and loss for GLU and non-GLU network

4. Analysis & Conclusion

We successfully implemented fully convolutional model for speech recognition tasks based on Tensorflow Speech Commands Datasets and modified the original model by adding a novel gating mechanism GLU (gated linear units) and achieved higher accuracy. The best error rate we achieved is 4.34%.

Our experiment proves that using convolutional model for sequence to sequence learning is indeed feasible. Furthermore, the GLU implementation allows the network to control what information should be propagated through the hierarchy of layers which improves the network performance.

5. Prospect

In the future, we will try to include separate attention module to achieve higher accuracy and acquire the ability to tackle with larger scale problems. Besides, we are going to apply our algorithm and method to more fields such as auto translation, video generation and etc.

Furthermore, we want to train a neural style transfer model which deals with video based on CNN. The foundation of neural style transfer work with images lies in the fact that the generic feature representations learned by high-performing Convolutional Neural Networks can be used to independently process and manipulate the content and the style of natural images. The success of implementing CNN to solve speech recognition problem proposes a promising method for neural style transfer with video files.

References

[1] Language Modeling with Gated Convolutional Networks, Yann N. Dauphin, Angela Fan and etc., arXiv:1612.08083v3 [cs.CL] 8 Sep 2017

[2] Convolutional Sequence to Sequence Learning, Jonas Gehring, Michael Auli and etc., arXiv:1705.03122v3 [cs.CL] 25 Jul 2017

[3] Three new graphical models for statistical language modelling, Mnih, Andriy and Hinton, Geoffrey., In Proceedings of the 24th international conference on Machine learning, pp. 641–648.ACM, 2007.

[4] Predicting distributions with linearizing belief networks, Dauphin, Yann N and Grangier, David., arXiv preprint arXiv:1511.05622, 2015.
[5] A neural probabilistic language model, Bengio, Yoshua, Ducharme, and etc., journal of machine learning research, 3(Feb):1137–1155, 2003.