Reprogramming DeepSpeech Model for Lyrics Transcription Task

Jiarui Xu[1], Alexander Lerch[2]
[1]Georgia Institute of Technology, Atlanta, GA 30332 USA

ABSTRACT Music Information Retrieval (MIR) is a particular research area of great interest because there are various strategies to retrieve music. One of the strategies is to transcribe lyrics from music. The aim of the task is to accurately transcribe lyrics in audio to text. This paper discussed using reprogramming DeepSpeech model (a speech-text transcription model) to achieve this goal. To do reprogramming, DeepSpeech model was treated as a pretrained model and we added some carefully designed noise on the input of the model to make this speech-text transcription model fit the music-text transcription task. This paper also discussed the evaluation of the results of experiments.

INDEX TERMS **DeepSpeech, Reprogramming, lyrics transcription, MIR, DAMP**

I. Research Statement

Transcribing lyrics from musical audio is a challenging research problem which has not benefited from many advances made in the related field of automatic speech recognition[1,2,3,6], owing to the prevalent musical accompaniment and differences between the spoken and sung voice.

However, one aspect of this problem which has yet to be exploited by researchers is that reprogramming DeepSpeech model for lyrics transcription[9]. In this paper we will investigate how this reprogramming can be leveraged to form transcription with improved consistency and accuracy.

II. Motivation

The motivation for this research is:

1. Lyrics Transcription and Speech Model are two topics that I am very interested in, because this project can help me understand deep learning and nature language processing better.
2. Reprogramming is a new technique in lyrics transcription and it will have great potential because creating singing dataset is time consuming and labor consuming, and reprogramming needs smaller dataset than other deep learning model; therefore, reprogramming provides a solution for insufficient data[7].
3. This application of reprogramming will lay the foundation for future work based on reprogramming and speech model becuase reprogramming will help DeepSpeech model robust. At present, DeepSpeech can only transcribe speech. However, with reprogramming, DeepSpeech can extend its application to transcribe singing and other sound with phonetic variation. When DeepSpeech model has the ability to transcribe singing, it will also have the ability to transcribe dyspronunciation which will be useful application for transcripition in other areas, such as helping the disabled.
4. Lyrics transcription has certain practical value because it can be applied on music software and other commercial software to do lyrics transcription.

III. Research Context

Singing speech recognition receives little attention in the field of speech research. The lack of research in this area is partly due to the less immediate importance of speech recognition applications, but it may also be due to a lack of readily available data. This is unfortunate because, apart from its application in music retrieval and indexing, singing speech recognition is itself a challenging problem worthy of investigation. When singing, for example, articulation is usually secondary, so singing sounds can share many other features of speech signals that are difficult to understand (e.g., dyspronunciation). It can be used as a stress test for state-of-the-art acoustic modeling (AM) techniques. Until recently, singing speech recognition has relied on small data sets, often mixed with speech and musical accompaniment (in which case the acoustic modeling problem becomes secondary to the more severe source separation challenges)[1,2,3,4,5].

The earliest notable work on singing speech recognition was done by Mesaros and Virtanen[1], most recently in 2010. Based on the materials in CMU Arctic speech database, the monophonic Gaussian mixture model (GMM)- hidden Markov model (HMM) model is trained. The work used just

30 minutes of hand-annotated monophonic singing recordings, divided into 49 segments (19 males and 30 females), each between 20 and 30 seconds in length. When music fragments were used to constrain the adaptive maximum likelihood linear regression (CMLR), Bigram achieved more than 87% word error rate(WER, we will discuss WER in evaluation part) and trigram achieved more than 100% error. More recently, Kruspe[2] trained a fully connected DNN-HMM model with three hidden layers on the DAMP Multiple Performance[6] solo-singing dataset. The data set was released without transcription or timing information, an error-prone process that requires automatic alignment of lyric text on Smule's website, with a phoneme error rate of about 77% reported on large test sets. Other recent work[3] investigated a state-of-the-art time-delayed neural nets-bidirectional Long Short-term memory (TDNN-BLSTM) model that trained 110 hand-annotation vocal singing recordings from YouTube using 39 Mel frequency cepstrum coefficients (MFCC) plus delta features.

To compensate for the small amount of training data, the acoustic model was first pretrained on 100 hours of speech and then adjusted. However, the best WER score was 73.09%. Other attempts at singing speech recognition have resulted in similarly poor recognition performance [4,5].

The new way to do this task is to use reprogramming. Model reprogramming was firstly introduced in [7]. The authors show that one can learn a universal input transformation function to reprogram a pre-trained ImageNet model (without changing the model weights) for solving MNIST/CIFAR-10 image classification and simple vision-based counting tasks with high accuracy. It can be viewed as an efficient approach for transfer learning with limited data, and it has achieved state-of-the-art (SOTA) results on biomedical image classification tasks [8]. However, despite the empirical success, little is known on how and why reprogramming can be successful. Therefore, applying reprogramming on this new task is a worthy endeavor .

IV. DeepSpeech Model Description

The core of the system is an RNN, whose input is the spectrum of speech and output is an English string[10]. This RNN outputs a probability distribution, representing the probability of a character being output at this time. The character set also includes the 26 letters a-z and Spaces. For input x, we use h^l for layer l and h^0 for input. The first three layers are fully connected layers. For the first layer, the input at time t not only includes the feature x_t at time t, but also includes the features of C frames before and after it, with a total of 2C+1 frames. The first three layers are calculated by the following formula[10]:

$$h_t^l = g(W^l h_t^{l-1} + b^l) \tag{1}$$

Where g(z)=min(max (z,0),20) limits its maximum value on the basis of ReLU, so it is also called Clipped ReLU. $W^{(l)}$, $b^{(l)}$ are the weight matrix and bias parameters for layer l. The fourth layer is a bidirectional RNN[10]:

$$\begin{aligned} h_t^f &= g(W^4 h_t^3 + W_r^f h_{t-1}^f + b^4) \\ h_t^b &= g(W^4 h_t^3 + W_r^b h_{t+1}^b + b^4) \end{aligned} \tag{2}$$

Here, the most common RNN is used instead of LSTM/GRU to make the network structure simple and consistent and facilitate the optimization of calculation speed. In this bidirectional RNN, the parameters input to the implicit cell are shared (including bias), and the RNN in each direction has its own implicit cell - implicit cell parameter. h^f is calculated from time l to time t, while h^b is calculated from time t in reverse. Layer 5 takes as its input the two outputs of layer 4's bidirectional RNN. The last layer is a fully connected layer (no activation function) that uses SoftMax to turn the output into probabilities for each character.

With $P(c_t = k \mid x)$, we can use the connectionist temporal classification(CTC) loss calculation, and strives for the loss of parameters of gradient.

CTC[17] mainly converts network output into conditional probability distribution on tag sequence. This network can then be used as a classifier by selecting the most likely label for a given input sequence.

DeepSpeech has a word error rate (WER) of 12.6 on the standard Switchboard task, with the best resolution 10.4[10].

V. Dataset Description

Damp Sing! 300x30x2 (Sing!)[11] is the third karaoke performance database released by Smule2, available in DAMP repository. As with the previous two versions, the data comes from Smule's collaborative Karaoke mobile application. The first album, DAMP Vocal Performance, contains 34,620 interpretations (i.e., performances) covering 302 different songs. However, there is a large imbalance in the number of performances per song, and data is difficult to use due to a lack of timing information to match lyrics to performances. The second DAMP Vocal Sings (Balance) database[12] is an extension of the previous version, containing 24,874 explanations from 5,429 artists, but covering only 14 song arrangements. While this data set is acoustically rich, with only 14 songs, it does not have enough linguistic variability to train robust speech models. It is more suitable than speech recognition to study the variability of singers.

The latest Smule data release, Sing! Provides 18,676 translations for 13,154 singers, covering 5,690 songs, with the same number of translations by gender in the singer's country. In addition, it provides a lyric cue that is presented to the performer along with the cue time, so it is easier to align the lyrics with the signal. The data was collected and processed by Smule in the second half of 2017 and released in early 2018, selecting the two most popular artists (male and female) from the 300 most popular song arrangements in 30 countries. The arrangement of popular songs is explained by counting the number of popular songs, explained by the natural number of listen and vote Smule user application. It can be assumed that up-voted interpretations are well sung, good quality recording.

VI. Baseline

The baseline's[9] results obtained using a GMM-HMM and a factorised TDNN[14] state-of-art acoustic model. The acoustic features used are 13 MFCC plus delta, delta and energy, frame length is 25 ms, and overlap time is 15 ms. Baseline first trains a GMM-HMM three-tone speaker with GMM on top of fMLLR. The model is used to apply a cleansing process (standard Kaldi5) to the acoustic training set to remove bad pronunciations from the training data (e.g., filter out those incorrect transcripts).

This process removes about 10 percent of the training speech. Using "clean" data, a factorised TDNN with lattice free MMI[15] was trained.

In an initial experiment, the acoustic model was trained on clean-100 LibriSpeech training acoustic material(100h annotated audiobooks)[13] to test the performance of the recognizer when trained on well-labelled but extratronic speech data. Baseline compares this with Smule Sing! and previously described K song DSing training data sets. Performances in terms of WER is 23.33%.

VII. Experiment Description

A. Without noise

First of all, I transcribed the songs directly using the DeepSpeech preconfigurated model -- deepspeech-0.9.3. I started by converting the json in the Damp data to txt format and converting the m4a audio file to wav to match the required DeepSpeech type. Next, I transcribed the audio and text using the DeepSpeech model as a pretrained model.

To be more specifically, a loop was used to load every audio file in order with its corresponding ground truth lyrics file. The audio file was processed by DeepSpeech model. The transcription was saved and compared with the ground truth using WER(WER will be discussed in evaluation part).

After all the files were processed, I took the arithmetic average of the transcribed results of these documents to get the final transcribed accuracy.

In the following experiments, I still used the DeepSpeech preconfigurated model -- deepspeech-0.9.3. This time I tried to reprogram the input by adding Gaussian noise, Gamma noise, and const noise separately on it.

Since I add the same parameters to each point on the spectrum, I can directly add the noise to the audio waveform for training. In the training, I used mean-square error(MSE) as loss to measure the difference between the training value and the ground truth, and used Adam Optimizor for optimization and gradient calculation. The reason to use MSE is that MSE is the most intuitive means to represent loss.

B. Gaussian noise

Gaussian noise is a kind of noise whose probability density function obeys Gaussian distribution (i.e., normal distribution, with three parameters: mean value, variance and amplitude).

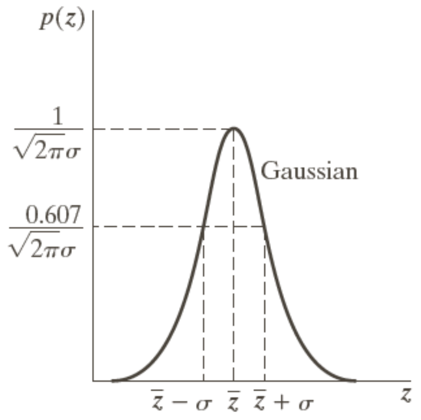

FIGURE 1. Gaussian Distribution.

C. Gamma noise

Gamma noise is a kind of noise whose probability density function obeys Gamma distribution (with three parameters: shape parameter b, scale parameter a and amplitude).

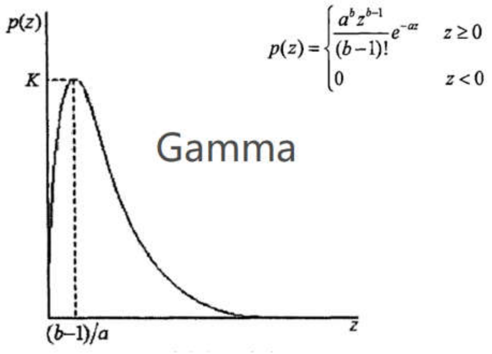

FIGURE 2. Gamma Distribution.

D. Constant noise

Constant noise is to add a uniformly distribution to the input. (only have one parameter: amplitude)

E. Advanced Gaussian noise training

Next, I tried to change the value of each point in the spectrum and set a separate Gaussian noise parameter for each point. A 1-minute spectrum diagram has about 10,000 * 10,000 $=10^8$ parameters. So it's going to be a huge training.

Since for the preconfigurated model, only audio input can be received as input, and the spectrum map cannot be directly input to the model. So I had to rebuild the DeepSpeech model.

I learned a lot during this period. For example, this was the first time I understand how to read and use json files. By reading the code of the whole DeepSpeech model, exploring the core of the whole DeepSpeech, I learned the benefits of flags, which could batch summarize and process variable names and pathnames. And by reading code carefully, I also found the feeding.py file to add function properly to do reprogramming on the input spectrum.

During the construction of DeepSpeech model, I also overcame a lot of unexpected obstacles.

1. For example, in the source code, the author calls the function train.py in the transcribed code file. Perhaps due to code update delays, tensorFlow is used as version1 in train.py, which is different from Tensorflow version2 in the tranwritten code file. In tensorflow version2, the contrib package in tensorflow 1 was removed and the RNN package was also removed. However, rnn.LSTMBlockFusedCell() was called in the train.py file. So I had to read the source code of LSTMBlockFusedCell() myself and build a new LSTMBlockFusedCell() function in the version2 environment to meet the requirements of the overall code. Through this process, gained a deeper understanding of Long Short-Term Memory(LSTM)'s dropout mechanism and learned how to use dropout and cells to avoid overfitting of LSTM and Recurrent Neural Networks(RNNs) models in speech training[18].
2. Another tricky difficulty is the scorer's setup. After downloading the model, I found that it lacked a good scorer system to transcribe, so I consulted materials to learn how to build a scorer and built a scorer for the model by myself.
   a) The first thing to set up a scorer is to have a set of checkpoints. I also prepared a text file (in .txt), with one phrase or word on each line. The text file is used by the generate_lm.py script.
   b) Then I used the lm_optimizer.py with Damp dataset and a set of checkpoints to find optimal values of default_alpha and default_beta. The default_alpha and default_beta parameters are used by the generate_scorer_package script to assign initial weights to sequences of words.
   c) After that, I used the generate_lm.py script which is distributed with DeepSpeech, along with the text file, to create two files, called lm.binary and vocab500000.txt.
   d) Next, I used the generate_scorer_package to create a kenlm.scorer file.
   e) By using the kenlm.scorer file as the external_scorer passed to DeepSpeech.py, and used for the test phase. The scorer does not impact training; it is used for calculating word error rate.

In the whole training, I also used MSE loss and Adam Optimizor for optimization and gradient calculation.

VIII. Evaluation and Result

Word Error Rate (WER)[16,17] was used for evaluation. The number of errors is computed as the sum of substitutions (S), insertions (I), and deletions (D). If the length of the reference transcription is N, then the word error rate WER is computed as follows:

$$\mathrm{WER} = \frac{I+D+S}{N} \times 100. \qquad (3)$$

Finally, for experiments without noise, WER is 36.7%, which means the accuracy is (1-36.7%)=63.3%.

Here is an example of transcription:

ground truth:
on my own Pretending he's beside me
all alone I walk with him till morning
without him I feel his arms around me
and when I lose my way I close my eyes
and he has found me
in the rain the pavement shines like silver
all the lights are misty in the river

transcription:
uh my ol preturned ayese beside be
al low i wark with him till mordie
without hem i feel his arms arround me
and when i lose my way i gose my eyes
at he has found me
and the ry the pave and shines i silver
all the lights amisdearerriver

For Gaussian noise, Gamma noise, Constant noise:

1. Gaussian noise: mean, variance converge to 0, amplitude misconverges.
2. Gamma noise shape parameter b converges to 1, scale parameter's inverse 1/a converges to 0, amplitude misconverges.
3. Constant noise: amplitude converges to 0.

Therefore, the accuracy of the three training of reprogramming experiments all came to 63.3%, which is the same as the experiments without noise.

For advanced Gaussian noise, there are still some problems to overcome because its error has not come down but continues to grow. The final transcription results were very unsatisfactory. The result of transcription would be meaningless:

ee et t e t t t i
i i i i i i i ar i
i i
i i i i
i i i i
i i i
i i i i

However, the WER is still 74.43% not 100%. Therefore, when using WER, the accuracy should higher than 25% because meaningless sentences will also have 25.57%

accuracy, which is due to the space between words are also counted in the length of the sequence.

As for the cause of the collapse of the model, I first excluded the possibility of inadequate epochs, because the loss of the whole model did not show a downward trend. I think the problem may lie in the model itself, because 10^8 parameters should be pre-processed, such as compression and weighted, before being put into the model. After this experiment, I will continue to improve the model.

IX. Conclusion

The innovation of this paper lies in using reprogramming to do lyrics transcription, and extending the function of DeepSpeech model from speech recognition to singing speech recognition and some pronunciation disorders recognition. However, the reprogramming model is still need to be improved to fit 10^8 parameters' training. During the whole model construction, I learned a lot and kindled my interest in the field of natural language processing. I will continue to devote myself to this project.

X. Reference

[1] A. Mesaros and T. Virtanen, “Recognition of phonemes and words in singing,” in 35th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2010, pp. 2146–2149.

[2] A. M. Kruspe, “Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing,” in 17th International Society for Music Information Retrieval Conference (ISMIR), New York, NY, USA, 2016, pp. 358–364.

[3] C.-P. Tsai, Y.-L. Tuan, and L.-S. Lee, “Transcribing Lyrics from Commercial Song Audio: The First Step Towards Singing Content Processing,” in 43th International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, Alberta, Canada, 2018, pp. 5749–5753.

[4] D. Kawai, K. Yamamoto, and S. Nakagawa, “Speech analysis of sung-speech and lyric recognition in monophonic singing,” in 43rd International Conference on Acoustics, Speech and Signal Processing, 2016, pp. 271–275.

[5] G. Roa, “Automatic Speech Recognition in Music,” Unpublished MSc. Dissertation, University of Sheffield, UK, 2016.

[6] Smule Vocal Performances (multiple songs) Dataset,“https://ccrma.stanford.edu/damp/,” in accessed July 2018.

[7] Elsayed, G. F., Goodfellow, I., and Sohl-Dickstein, J. Adversarial reprogramming of neural networks. In International Conference on Learning Representations, 2019.

[8] Tsai, Y.-Y., Chen, P.-Y., and Ho, T.-Y. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. In International Conference on Machine Learning, pp. 9614–9624, 2020.

[9] Dabike G R, Barker J. Automatic Lyric Transcription from Karaoke Vocal Tracks: Resources and a Baseline System[C]//Interspeech. 2019: 579-583

[10] Hannun A, Case C, Casper J, et al. Deep speech: Scaling up end-to-end speech recognition[J]. arXiv preprint arXiv:1412.5567, 2014.

[11] Smule Sing! 300x30x2 Dataset, “https://ccrma.stanford.edu/damp/,” in accessed September 2018.

[12] Smule Vocal Performances (balanced) Dataset, “https://ccrma.stanford.edu/damp/,” in accessed July 2018.

[13] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” in 40th International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 2015, pp. 5206–5210.

[14] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, “Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks,” in 19th conference of the International Speech Communication Association (ISCA – Interspeech 2018), Hyderabad, India, 2018, pp. 3743–3747.

[15] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur, “Purely sequence-trained neural networks for ASR based on lattice free MMI,” in 17th conference of the International Speech Communication Association (ISCA –Interspeech 2016), San Francisco, California, USA, 2016, pp. 2751–2755.

[16] Ali A, Renals S. Word error rate estimation for speech recognition: e-WER[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2018: 20-24.

[17] Graves A, Fernández S, Gomez F, et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks[C]//Proceedings of the 23rd international conference on Machine learning. 2006: 369-376.

[18] Zaremba W, Sutskever I, Vinyals O. Recurrent neural network regularization[J]. arXiv preprint arXiv:1409.2329, 2014.