Research On How Heuristics Can Simplify Model Training For Automatic Speech Recognition

Zhenling Yang

MInf Project (Part 1) Report
Master of Informatics
School of Informatics
University of Edinburgh

2022

Abstract

In recent years, end-to-end speech recognition based on neural networks have become the mainstream of speech recognition, and two model structures based on connectionist temporal classification (CTC) and sequence-to-sequence are proposed. Compared with traditional ASR systems, such as HMM-GMM and hybrid HMM-DNN, end-to-end speech recognition greatly simplify the implementation and training process of ASR systems, and achieve better recognition results.

However, end-to-end speech recognizers require various heuristics such as Dropout, Label Smoothing and Scheduled Sampling to achieve a desirable performance. But little research has been done on why these heuristics improve model performance and what exactly these heuristics compensate for the shortcomings of speech recognizers. In this report, I will implement a joint CTC attention model as a baseline and design experiments to evaluate the defects of the model itself and study how Dropout, Label Smoothing and Scheduled Sampling compensate for these defects as well as study how they affect calibration, exposure bias, ability to stop of the model to explain why these heuristics can improve model performance.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Zhenling Yang*)

Acknowledgements

I would like to thank my supervisor Hao Tang for his support and guidance throughout the project. I am especially grateful to him for carefully designing the mini ASR course in the early stage to help me build up the background knowledge of speech recognition. I am very grateful to him for conducting small meetings every week to guide our project progress, and I am very grateful to him for the experimental suggestions he gave me every week. Hao Tang, thank you for your teaching and hard work!

Finally, I would like to thank my family and friends for their great support. Without them, I would not be where I am now. Mom, Dad, thank you for making me feel like you're always there for me even though we're thousands of miles apart. Thank you dear friends for your company, thank you.

Table of Contents

Chapter 1

Introduction

In recent years, the end-to-end speech recognition has become the mainstream in the field of speech recognition. Sequence-to-sequence models are one of the widely used types in the family of end-to-end models. Sequence-to-sequence models are models that take a sequence as input and output another sequence, and the main components are an encoder and a decoder network. The encoder converts the input data into corresponding hidden vectors containing its context, while the decoder uses the previous output and hidden vectors as input to predict the next output. The wide adoption of sequence-to-sequence models can be attributed to its simplicity in implementation while being competitive with the state of the art.

Some past ASR systems such as HMM-GMM and HMM-DNN required the integration of multiple modules to implement a complete speech recognition pipeline. They need to train the acoustic model, lexicon model and language model separately, and integrate each module through WFST to generate a complete speech recognition network, which is finally decoded through a graph search or beam search algorithm. The development of such a system often requires a huge amount of engineering code, especially the design and implementation of WFST will be very complex, and require very professional linguistic knowledge to build the lexicon model.

Sequence-to-sequence ASR systems no longer need to train each module individually; instead, the functions of all modules are entirely performed by a neural network. The acoustic model is replaced by the encoder, and the language model is replaced by the decoder. Therefore, sequence-to-sequence speech recognizers can directly match speech features with transcribed text through an attention mechanism, and do not require separate training of acoustic models, lexicon models and language models as in traditional ASR systems. Thanks to the development of deep learning frameworks in recent years, this model can be easily implemented with only a small amount of engineering code.

Although many studies have shown that sequence-to-sequence speech recognizers generally outperform HMM-GMM or HMM-DNN, many researchers still find that this type of model is hard to train. The main reasons are that the training of the model converges slowly, the mismatch between training-time and test-time algorithms, and the

model is prone to overconfidence in its own prediction results. Therefore, there needs to be a lot of heuristics to improve the training of the model.

Research in recent years has found that sequence-to-sequence speech recognizers can achieve better recognition performance after using some heuristics such as regularization techniques Dropout and Label Smoothing, as well as Curriculum training technique such as Scheduled Sampling. The original motivation for Dropout was to solve the problem of model overfitting by preventing units from co-adapting too much, while Label Smoothing was proposed to solve the model's overconfidence in its own prediction results. Scheduled Sampling is proposed to partially address the mismatch of the decoder's input during the training and inference phases. Many researchers have also found that these heuristics can make the training of sequence-to-sequence models easier. However, few studies have shown that these heuristics can solve some known problems of sequence-to-sequence models, such as exposure bias, calibration, etc.

In order to study why these heuristics improve model performance and what exactly these heuristics compensate for the shortcomings of speech recognizers. In this report, I will implement a joint CTC attention model as baseline, and designing a pytorch framework for model training and experimentation to evaluate the defects of the model itself and study how Dropout, Label Smoothing and Scheduled Sampling compensate for these defects as well as study how they affect calibration, exposure bias, and ability to stop of the model to explain why these heuristics can improve model performance. My experimental results found that the heuristic has some impact on the above properties of the model, and the joint CTC attention model itself has some defects. These heuristics more or less make up for the defects of the model and improve the model performance.

1.1 Thesis Main Contribution

This project required both engineering and analytical work. The following shows all the work done by this thesis:

- Implement a pytorch framework for model training and experimentation.
- Implement the joint CTC attention model from the Suyoun Kim *et al.* [11].
- Apply Label Smoothing, Dropout and Scheduled Sampling to the model.
- Compare the CER/WER and learning curves of the model using heuristics.
- Analyze the calibration of the models using different heuristics through reliability diagram, expected calibration error and adaptive calibration error.
- Analyze whether models using different heuristics can reduce exposure bias by measuring CER/WER after applying random decoding noise during inference.
- Analyze the stopping ability of the decoder of models using different heuristics through the gap between actual and predicted sentence length.

1.2 Thesis Outline

The chapter 2 will introduce the basic concepts of all the key modules present in the thesis. The chapter 3 will introduce the definition and formulation of heuristics and their research status, as well as the connection with the work of the thesis. Chapter 4 will introduce experimental methods and evaluation metrics for the impact of heuristics on model performance. Chapter 5 will describe the experimental setup and experimental results. Chapter 6 will summarize the whole thesis.

Chapter 2

Background

In this chapter, I will introduce the pipeline of the speech recognizer one by one from the input data of the speech recognizer to the models generated using each individual component. I will first introduce the principles of input features that appear in speech recognizers and how these features are processed, and then I will introduce the basic modules of sequence-to-sequence speech recognition models, such as Long Short-Term Memory, attention mechanisms and Connectionist Temporal Classification. Finally, I will introduce the principles of the Listen, Attend and Spell model and the joint CTC attention model used in thesis and the comparison between them.

2.1 Acoustic Features

The acoustic features are processed data that are used as input features for speech recognizers. They are all extracted from the most primitive speech data - waveform. A waveform is a digital signal obtained by sampling real speech through a tool such as a microphone. It is shown in the upper part of Figure 2.1. We can find that the waveform takes the time as the x-axis and the amplitude as the y-axis, and the information in the figure represents the change of the amplitude over time. Since the amplitude and duration of the waveform are determined by the characteristics of the speaker's voice, for the same sentence, different people will get very different waveforms. This will make the waveform less suitable as input features for speech recognizers, because speech recognizers can not do good pattern recognition for too variable features. Therefore, the features based on the frequency domain have become the main features used in speech recognizers because of its stability.

There are two main types of acoustic features which are based on frequency domain and used in sequence-to-sequence speech recognition models. They are spectrogram and MFCC respectively. The spectrogram is the frequency domain representation of the waveform. It is shown in the middle part of Figure 2.1. We can see that the spectrogram takes time as the x-axis and frequency as the y-axis, and the colors in the figure indicate the energy levels of different frequencies at each time step. We can clearly see the frequency patterns of different phonemes from the spectrogram. Using spectrogram will be less affected by different human voices than waveform,

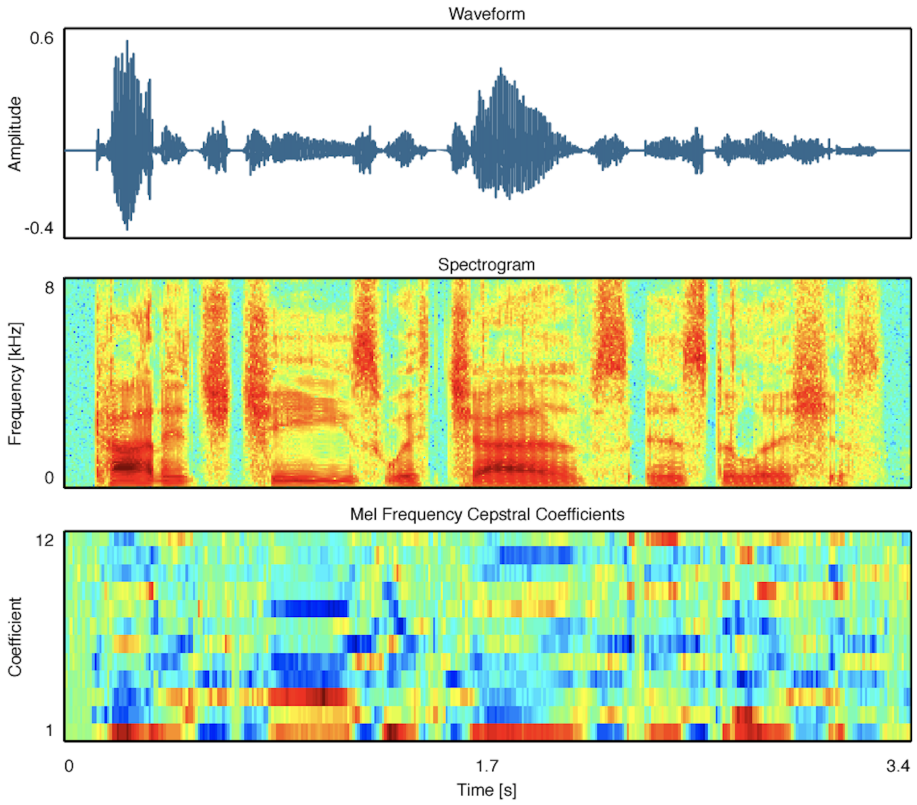


Figure 2.1: Schematic diagram of Acoustic Features

because the sound frequency of different people for the same sentence is relatively stable. In this way, the speech recognizer can better learn the frequency characteristics of different phonemes from the spectrogram, so as to have the ability to recognize speech.

Although a spectrogram is a very nice feature, generating a spectrogram is a very complex process. The first step is to pre-emphasize and frame the waveform. Pre-emphasis aims to make the higher frequency parts of the original waveform more pronounced, while framing allows subsequent processing to occur in a stationary waveform. The second step is Fourier transform, which can convert time domain data into frequency domain data, so each frame of data can be converted into a corresponding spectrum. The third step is mel filter sampling. In order to obtain logarithmic features similar to human hearing, each spectrum needs to be sampled using multiple mel filters to generate a mel spectrum. Finally, the spectrogram is generated by combining the mel-spectra generated from all frame data. A flowchart of feature processing is shown in Figure 2.2.

Another acoustic feature is the Mel Cepstral Coefficient (MFCC). It is shown in the lower part of Figure 2.1. It uses the coefficients as the y-axis instead of the frequency, because the vertical quantities at each time step represents the weights of the cosine waves at different frequencies. The weighted sum of these cosine waves of different frequencies can be approximately equal to the corresponding spectrum. Mel cepstral coefficients are obtained by doing discrete cosine transform after taking log of the

spectrogram. Compared with the spectrogram, MFCC can extract channel information, thereby filtering out the noise caused by the sound source. But because of this, MFCC uses features with fewer dimensions, so it is usually used in traditional machine learning algorithms. The neural network model can extract features directly from the spectrogram because of its powerful ability.

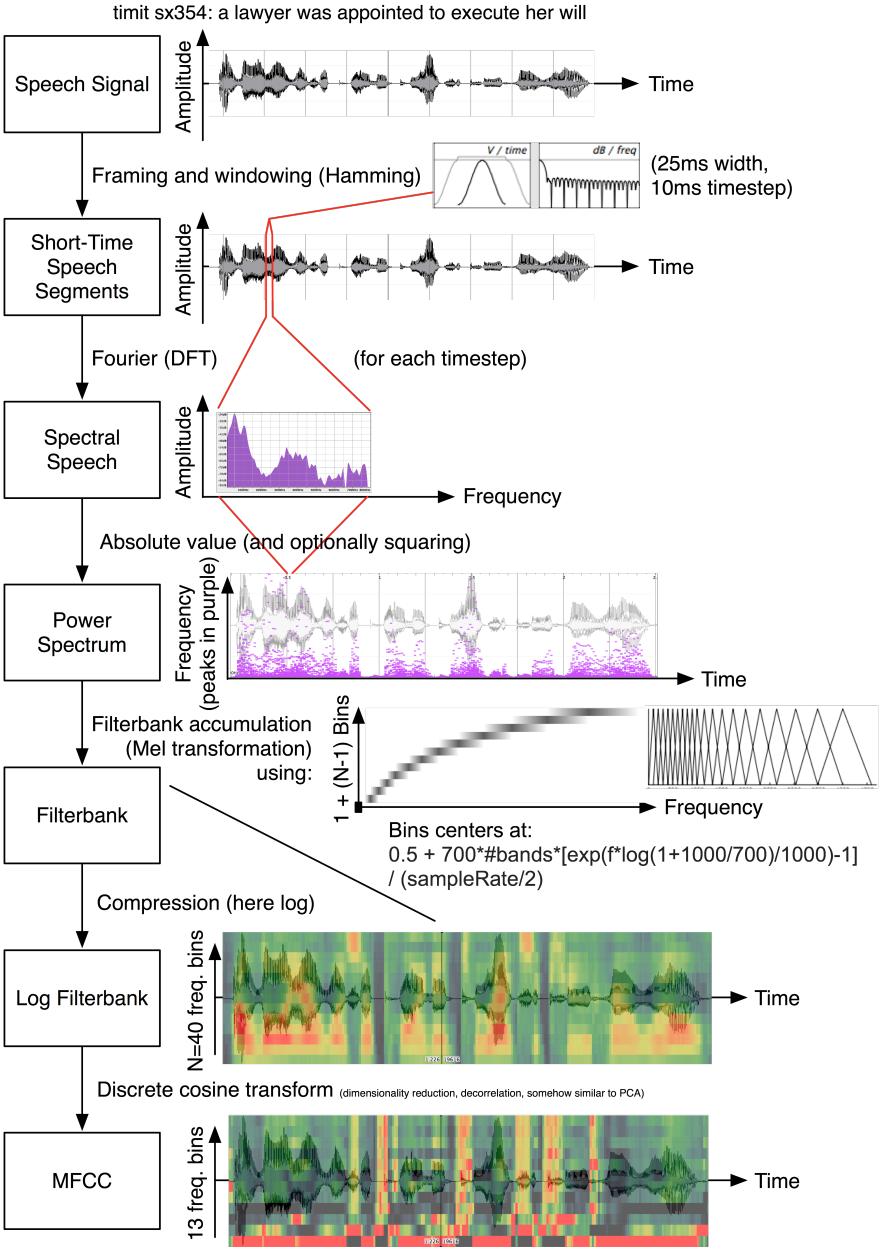


Figure 2.2: Schematic diagram of Feature Processing

2.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) is an artificial neural network that can model time series data. It was first proposed by S Hochreiter *et al.* [10] to solve the vanishing and

exploding gradient problem of RNN. They found that the gradient of the RNN using the sigmoid activation function will continue to grow or shrink through backpropagation, and eventually become close to zero or a large value. This will cause the optimizer to fail to update the RNN's weights and thus fail to learn long-term dependencies.

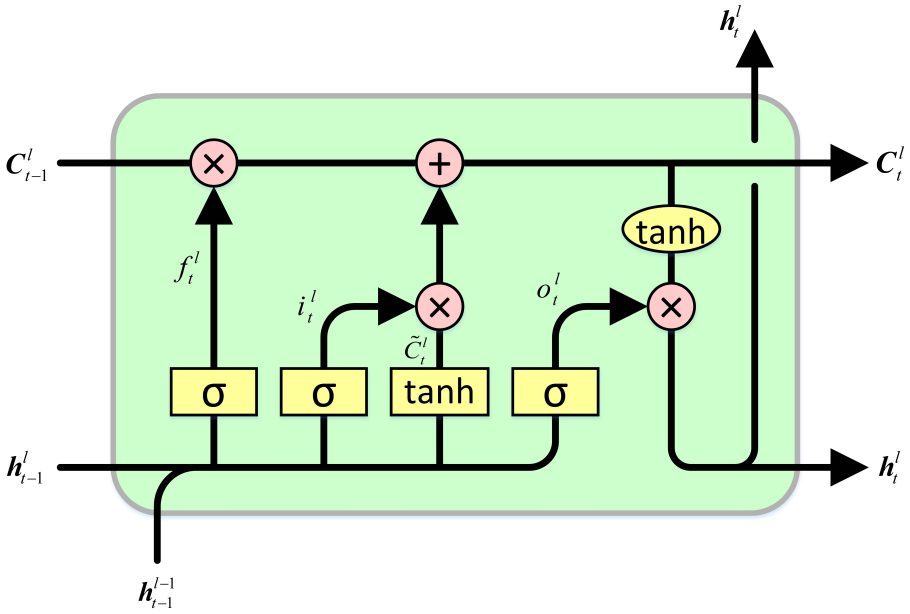


Figure 2.3: Schematic diagram of LSTM [16]

The LSTM solves this problem by making the internal state unaffected by nonlinear functions and keeping the value of gradient around one, which effectively avoids exponential vanishing and exploding gradients. The structure of LSTM is shown in Figure 2.3. LSTM consists of a hidden state, a memory cell and three gates, which are input gate, forget gate and output gate. The memory cell is mainly used to store and update historical information. The input gate mainly decides what input will be fed to the memory cell. The forget gate mainly decides what part of the previous memory cell will be kept. The output gate mainly decides what part of the memory cell will be given as output. The definition of LSTM can be expressed by the following formula:

$$i_t = \sigma(x_t U^i + h_{t-1} W^i) \tag{2.1}$$

$$f_t = \sigma(x_t U^f + h_{t-1} W^f) \tag{2.2}$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o) \tag{2.3}$$

$$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g) \tag{2.4}$$

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \tag{2.5}$$

$$h_t = \tanh(C_t) * o_t \tag{2.6}$$

where i_t, f_t, o_t, C_t, h_t represents the input gate, forget gate, output gate, memory cell and hidden state respectively. All three gates send the input and hidden state to the fully connected layer separately and add them together, and turn the output into a probability distribution through the sigmoid function. $\tilde{C}_t$ represents the current information processed in normal RNN. Memory cell will select the historical information that needs

to be retained through the forget gate, and then add some current information to the historical information through the input gate. Finally, the hidden state will be obtained by selecting part of the information from the memory cell through the output gate.

2.3 Attention Mechanism

The attention mechanism is a mechanism that enhances the model's attention to some parts of the input data, while reducing the attention to other parts. It is motivated by the idea that the network should focus more on small but important parts of the data. Figure 2.4 shows a schematic diagram of the attention mechanism in an RNN-based encoder-decoder network.

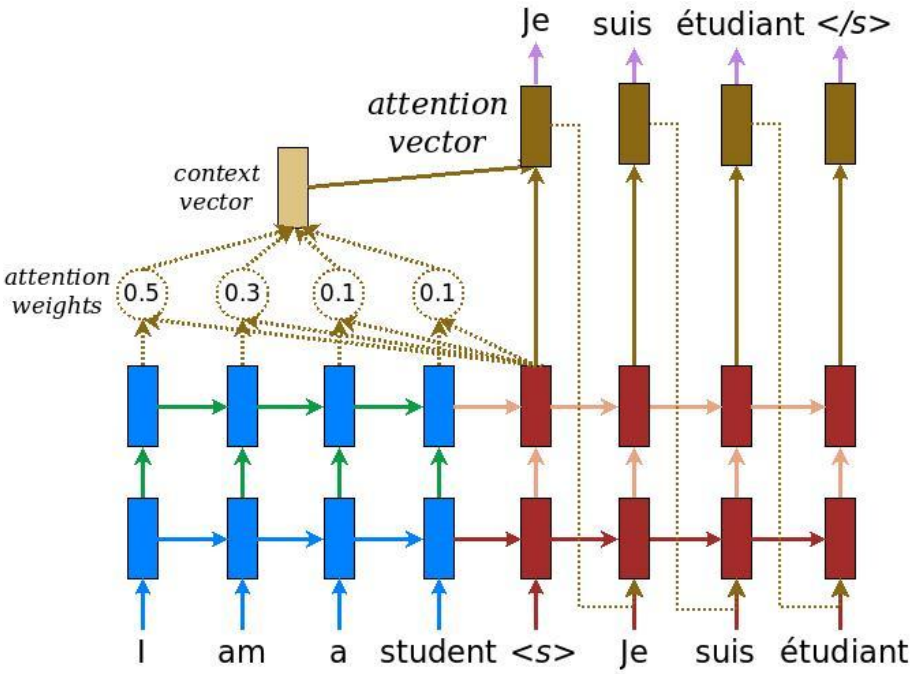


Figure 2.4: Schematic diagram of Attention Mechanism [13]

The most common attention mechanism is dot product attention which is an attention mechanism where an alignment score function is computed by dot product of the hidden state of the decoder and the output of the encoder, and use a softmax on the result. It was first proposed by Dzmitry Bahdanau *et al.* [1] to allow the model to automatically match the encoder and decoder information. The definition of dot product attention can be expressed by the following formula:

$$e_{ij} = \langle s_{s-1}, h_j \rangle \tag{2.7}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.8}$$

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.9}$$

where e_{ij} represents the score obtained by dot product of the hidden state of the decoder s_{s-1} and the output of the encoder h_j, α_{ij} represents alignment score after using softmax on e_{ij}. C_i represents the weighted sum of encoder outputs. Since α_{ij} can show which encoder outputs are important to the hidden state of the current decoder, C_i is equivalent to the weighted summation of all encoder outputs according to the attention degree of the attention mechanism, thereby obtaining a context vector indicating which encoder outputs that the decoder's hidden state pays attention to.

In addition, content based attention such as additive attention was also proposed by Bahdanau *et al.* [1]. Compared to dot product attention, it uses one-hidden layer feed-forward network to calculate the attention alignment score rather than directly compute similarity and get better performance. The definition of additive attention can be expressed by the following formula:

$$e_{ij} = \upsilon_a^T \tanh(W_1 h_t + W_2 \bar{h}_s) \tag{2.10}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.11}$$

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.12}$$

where e_{ij} represents the score obtained by output of feed-forward network using the hidden state of the decoder $\bar{h}_s$ and the output of the encoder h_t, α_{ij} represents alignment score after using softmax on e_{ij}. C_i represents the weighted sum of encoder outputs. Content based attention uses the same principle as dot product attention, except for the calculation of alignment score.

Although content based attention has superior performance, it does not perform well in long sentences. To address this problem, Jan K. Chorowski *et al.* [4] proposed a location-aware attention mechanism, which can still perform well in the case of decoding long sentences. They propose to add a convolution module on the basis of content based attention to make the model focus on a narrow part, so that it is not affected by the noise of other parts. The definition of location-aware attention can be expressed by the following formula:

$$e_{ij} = w^T \tanh(W s_{i-1} + V h_j + U f_{i,j} + b) \tag{2.13}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \tag{2.14}$$

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.15}$$

where e_{ij} represents the score obtained by output of feed-forward network using the hidden state of the decoder s_{i-1} and the output of the encoder h_j. e_{ij} is computed similar to content based attention but including an extra convolution module. α_{ij} represents alignment score after using softmax on e_{ij}. C_i represents the weighted sum of encoder outputs. Location-aware attention uses the same principle as content based attention, except for the calculation of alignment score.

2.4 Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) is an approach that avoids manual alignment of input data and labels that require segmentation. It is often used to train recurrent neural networks such as LSTMs because such models typically need to be trained using segmented data with aligned labels. CTC was first proposed by Graves *et al.* [6] to label unsegmented data. They combined CTC with an RNN network to predict the corresponding sentence directly from the input data without first segmenting all the data and aligning the labels of each segment. The Figure 2.5 shows a schematic diagram of Connectionist Temporal Classification.

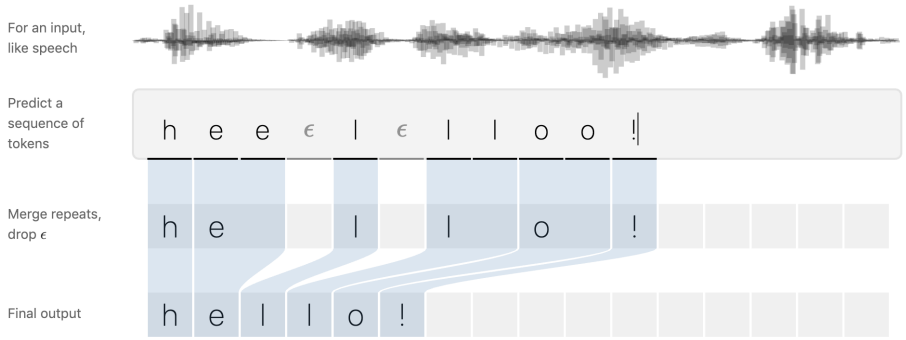


Figure 2.5: Schematic diagram of Connectionist Temporal Classification [8]

To be able to directly predict sentences without frame-aligned labels, CTC introduces a blank label to predict splits between different labels. In the decoding stage, CTC merges adjacent repeated labels in the RNN output sequence, and then removes blank labels to obtain the final output. In the encoding stage, CTC will take the total probability of all paths that can get the true sentence label after decoding as the optimization object of the loss function. This process can be realized by the hidden Markov model. The advantage of CTC is that it does not need to pre-label each frame of data to train the RNN model, and CTC guarantees the monotonicity of the prediction results. This greatly reduces the labeling cost of training data and ensures that the output results meet the characteristics of speech data.

2.5 Listen, Attend and Spell Model

The Listen, Attend and Spell is an RNN-based encoder-decoder model which was first proposed by William Chan *et al.* [3]. It is a sequence-to-sequence model that uses an attention mechanism to model the correspondence between acoustic features and characters. It uses bidirectional pyramidal LSTMs as the encoder, unidirectional LSTM as decoder, and content based attention as attention mechanism. It replaces the acoustic model, lexicon model and language model of traditional speech recognizers with these three modules, completely simplifying the end to end speech recognition system into an end to end speech recognition model. The Figure 2.6 shows a schematic diagram of LAS model.

Compared to CTC models such as DeepSpeech [9], the LAS model does not require a complex decoder for decoding, and is not limited by the conditional independence assumption. The architecture of the LAS model can be defined by the following formula:

$$P(y|x) = \prod_{u} P(y_u|x, y_{1:u-1}) \tag{2.16}$$

$$h = Encoder(x) \tag{2.17}$$

$$y_u \sim AttentionDecoder(h, y_{1:u-1}) \tag{2.18}$$

where $P(y|x)$ represents the probability that the input speech feature obtains the corresponding output sentence. It is obtained by multiplying the output probabilities for all time steps given input x and previous outputs $y_{1:u-1}$. h represents the output features of the encoder obtained from the input speech features x, and *Encoder* represents the pyramidal LSTMs. y_u represents the output of the current time step, and *AttentionDecoder* represents an RNN based decoder with content based attention mechanism.

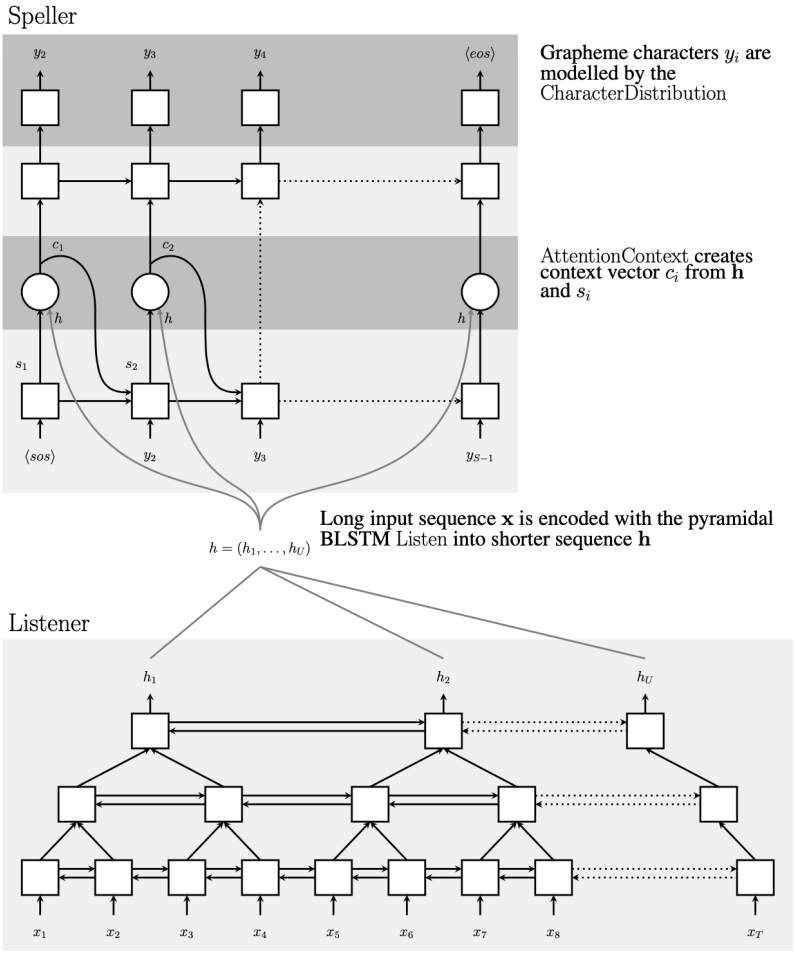


Figure 2.6: Schematic diagram of Listen, Attend and Spell [3]

2.6 Joint CTC Attention Model

The joint CTC attention model is an RNN-based encoder-decoder model inspired by LAS model [3] and connectionist temporal classification (CTC) [6]. It was first proposed by Suyoun Kim *et al.* [11] to make the predictions of the LAS model monotonic and speed up the convergence through CTC. The LAS model uses a neural network composed of pyramidal LSTMs as the encoder and a neural network composed of unidirectional LSTMs as the decoder. An attention mechanism is used between the encoder and decoder to represent the mapping between acoustic features and transcribed characters. The joint CTC attention model continues this architectural design, but uses CTC on the encoder to perform multi-task learning with the original objective. Such a model design is not only beneficial from the LAS model to directly transcribe speech features into characters, but also the use of CTC can speed up the convergence of model training. The Figure 2.7 shows a schematic diagram of joint CTC attention model.

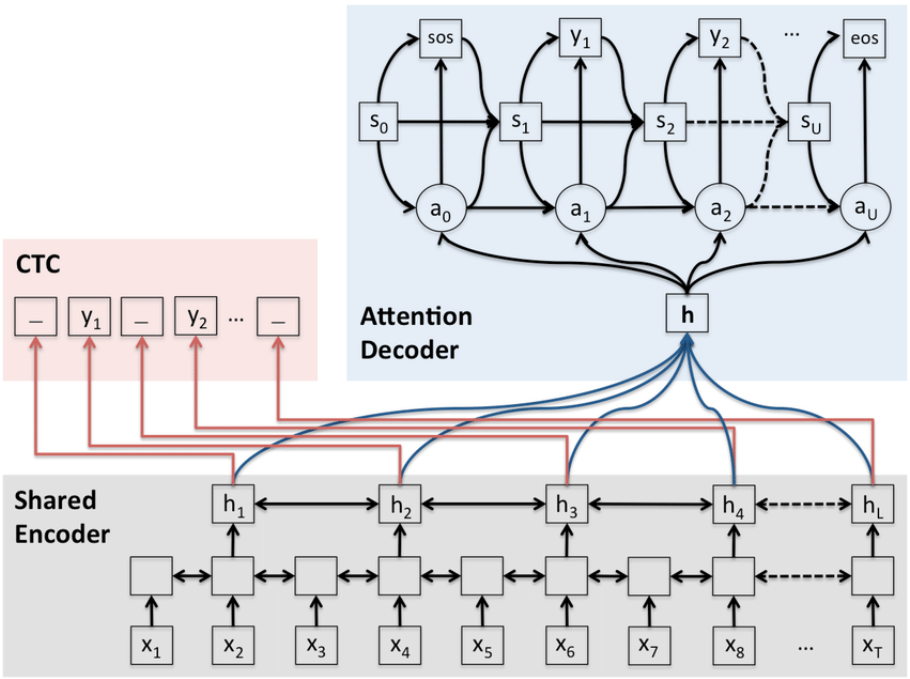


Figure 2.7: Schematic diagram of Joint CTC Attention Model [11]

The general architecture of the joint CTC attention model is the same as the LAS model except a CTC loss on the encoder which can be defined by the following formula:

$$P(y|x) = \prod_{u} P(y_u|x, y_{1:u-1}) \tag{2.19}$$

$$h = Encoder(x) \tag{2.20}$$

$$y_u \sim AttentionDecoder(h, y_{1:u-1}) \tag{2.21}$$

$$\mathcal{L}_{CTC} = CTC(h) \tag{2.22}$$

$$\mathcal{L}_{Attention} = Cross\_Ent(y) \tag{2.23}$$

where $P(y|x)$ represents the probability that the input speech feature obtains the corresponding output sentence. It is obtained by multiplying the output probabilities for all time steps given input x and previous outputs $y_{1:u-1}$. h represents the output features of the encoder obtained from the input speech features x, and $Encoder$ represents the pyramidal LSTMs. y_u represents the output of the current time step, and $AttentionDecoder$ represents decoder with attention mechanism. Finally, $\mathcal{L}_{CTC}$ and $\mathcal{L}_{Attention}$ represents CTC loss on the encoder and cross entropy loss on the decoder.

The difference between joint CTC attention model and LAS model is in the definition of the loss function. The loss function of the joint CTC attention model can be defined by the following formula:

$$\mathcal{L}_{MTL} = \lambda \mathcal{L}_{CTC} + (1-\lambda)\mathcal{L}_{Attention} \tag{2.24}$$

where $\mathcal{L}_{MTL}$ represents the total loss, λ represents the weight of each loss, $\mathcal{L}_{CTC}$ represents the CTC loss of the encoder and $\mathcal{L}_{Attention}$ represents the cross entropy loss of the decoder. λ is a tunable hyperparameter that affects the speed of model training convergence and it will also decide which part dominates the model's predictions.

The choice of this model benefits from the following points [21]. The training of the model does not require stepwise refinement. Many traditional speech recognizers such as ASR systems based on HMM-DNN need to build a bootstrap model, such as HMM-GMM model, to obtain a better initial alignment for model training. And they require the combination of multiple modules such as acoustic models, pronunciation dictionaries, and language models to perform speech recognition. Although WFST can combine these modules very well, designing WFST is still a complex task. Furthermore, traditional speech recognizers not only need to optimize each module individually, but also need to overcome incoherence in optimization. Linguistic information like pronunciation dictionaries also needs to be designed by professionals.

Using an attention based end-to-end speech recognizer can solve all the above problems. The model can directly encode the speech features through the encoder, and use the attention mechanism to match the encoded speech features with the characters generated by decoder, and finally get the transcribed sentences through the decoder. Compared with CTC-based end-to-end models, such as DeepSpeech [9], the biggest advantage of doing joint CTC training is to enforce monotonic alignments, which is more in line with the contextual dependency of speech itself.

Chapter 3

Training Heuristics

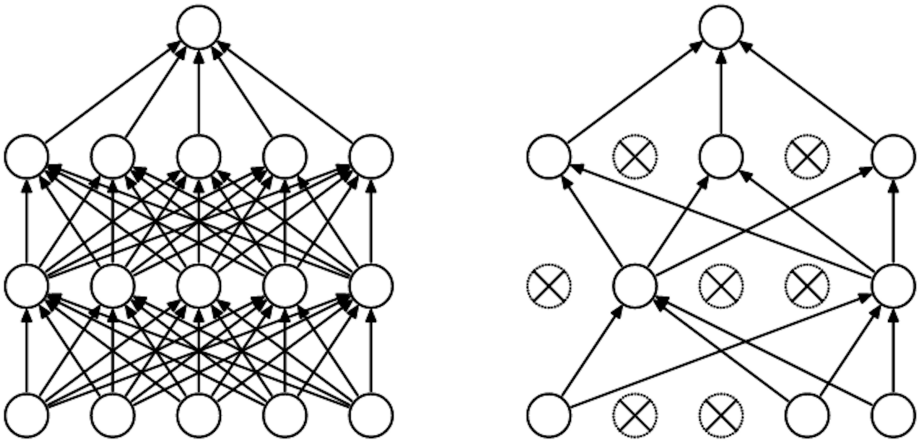
The heuristics refer to methods that can improve the performance of a model over the original architecture. In deep learning, regularization techniques such as Dropout and Label Smoothing, and Cirriculumn training techniques such as Scheduled Sampling are typical heuristics. These heuristics are proposed mainly because neural network models have certain disadvantages, such as being prone to overfitting. In recent years, speech recognition has been dominated by deep learning, and neural networks are widely used in the construction of speech recognizers. Therefore, the shortcomings of the neural network itself also have a bad effect on the performance of the speech recognizer. This necessitates the application of heuristics to speech recognizers to solve those issues.

Research on how and why Dropout, Label Smoothing and Scheduled Sampling improve model performance have emerged in the fields of computer vision and natural language processing. But in the speech domain, it remains to be studied whether there are similar reasons for these heuristics to improve the model performance. In addition, there are only very rare speech-related literature that mentions the reasons for the heuristics to improve model performance, most of them use the CTC model or the LAS model as the baseline, and there is basically no literature on verifying the heuristic effect on the joint CTC attention model. This gives me the opportunity to study the impact of heuristics on sequence-to-sequence models using CTC.

The rest of this chapter will introduce the three heuristics that will be studied in the thesis: Dropout, Scheduled Sampling and Label Smoothing.

3.1 Dropout Regularization

Dropout is currently the most famous and commonly used regularization technique in neural network models. It was first proposed by Geoffrey Hinton *et al.* [18] to solve the problem of model overfitting. Dropout is an ensemble method that samples many smaller neural networks during the training phase and uses a large network with smaller weights in the prediction phase to estimate the average prediction of those networks. The Figure 3.1 shows a schematic diagram before and after applying Dropout.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.1: Schematic diagram of Dropout [18]

The original motivation for Dropout is to let each hidden unit in the network learn how to co-adapt with randomly selected samples of other units. This makes each hidden unit more robust and pushes it to create useful functionality on its own without relying on other hidden units to correct their mistakes. Technically, this is done by using a dropout mask over hidden units, which randomly drops some neurons in the neural network layer with a certain probability during training. The sampled networks can effectively prevent overfitting problems, and using ensemble methods can make predictions more deterministic. The definition of Dropout can be expressed by the following formula:

- feed-forward operation without Dropout:

$$z_i^{(l+1)} = w_i^{(l+1)} y^{(l)} + b_i^{(l+1)} \tag{3.1}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{3.2}$$

- feed-forward operation with Dropout:

$$r_j^{(l)} \sim Bernoulli(p) \tag{3.3}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{3.4}$$

$$z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^{(l)} + b_i^{(l+1)} \tag{3.5}$$

$$y_i^{(l+1)} = f(z_i^{(l+1)}) \tag{3.6}$$

  where $r^{(l)}$ represents a Bernoulli mask layer that decides which neuron values of $y^{(l)}$ will be zero according to probability p.

In recent years, Yarin Gal *et al.* [5] proposed that Dropout can be used for Bayesian approximation. Later, Apoorv Vyas *et al.* [20] took advantage of this property and used Dropout on CTC-based speech recognizers to estimate the error rate of the speech recognizer and locate specific error locations. All these results show that Dropout can effectively reduce the uncertainty of the model.

3.2 Scheduled Sampling

Scheduled Sampling was first proposed to solve the problem of exposure bias which refers to the train-test discrepancy that arises when an auto-regressive generative model uses only the ground truth context during training and the generated context during testing [17]. It was proposed by Samy Bengio *et al.* [2] to solve the discrepancy between the training process and the decoding of the RNN-based sequence-to-sequence model. The Figure 3.2 shows a schematic diagram of Scheduled Sampling.

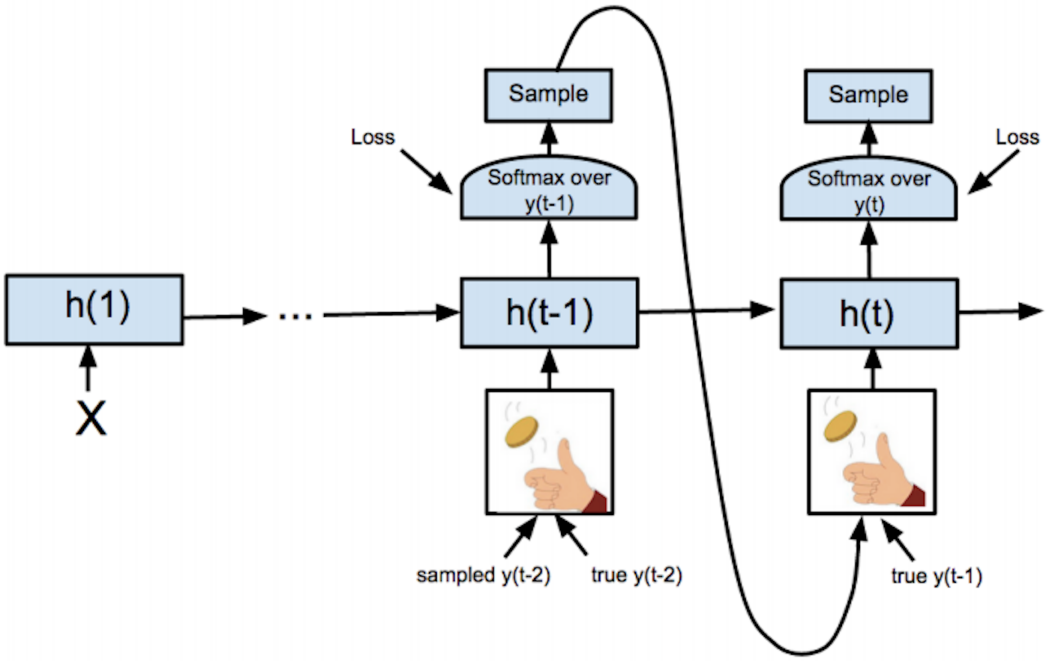


Figure 3.2: Schematic diagram of Scheduled Sampling [2]

During the training phase, the model uses the ground truth label from previous time step as input to predict a new token, which is also called teacher force training. But in the inference stage, the model uses the output of the previous time step as input to predict a new token instead of using the ground truth label. This means that every inference in the training phase is based on the correct sequence, while the decoding phase will most likely infer based on the wrong sequence, resulting in the inference error being gradually amplified. The definition of Scheduled Sampling can be expressed by the following formula:

$$\log P(Y|X;\theta) = \sum_{t=1}^{T} \log P(y_t|h_t;\theta) \tag{3.7}$$

where $\log P(Y|X;\theta)$ represents the log probability of the predicted sequence and h_t is computed by a recurrent neural network as follows:

$$h_t = \begin{cases} f(h_{t-1}, x_t;\theta) & \text{if } p > S \\ f(h_{t-1}, y_{t-1};\theta) & \text{otherwise} \end{cases} \tag{3.8}$$

where θ represents parameters of model, S represents sampling rate and p represents probability. x_t represents ground truth label and y_{t-1} represents previous output.

This problem also occurs in sequence-to-sequence speech recognition models, because such speech recognizers also use an auto-regressive language model as a decoder. In my work, I use random replacement of the generated labels to influence the decoding performance of the speech recognizer, determine whether the model is affected by exposure bias by evaluating the model's resilience to erroneous sequences, and test whether Scheduled Sampling can effectively reduce the effect of exposure bias on the model.

3.3 Label Smoothing

Another well-known heuristic is Label Smoothing which was first proposed to addresses the problem of models being overconfident in their own predictions. It is a regularization technique that introduces noise for the labels. The definition of Label Smoothing can be expressed by the following formula:

$$y_{smooth} = y_{1hot} - \varepsilon(y_{1hot} - \frac{1}{k}y_{ones}) \tag{3.9}$$

where y_{ones} represents an all-one vector, y_{1hot} is a one-hot vector representing ground truth, k represents number of classes and ε represents Label Smoothing rate.

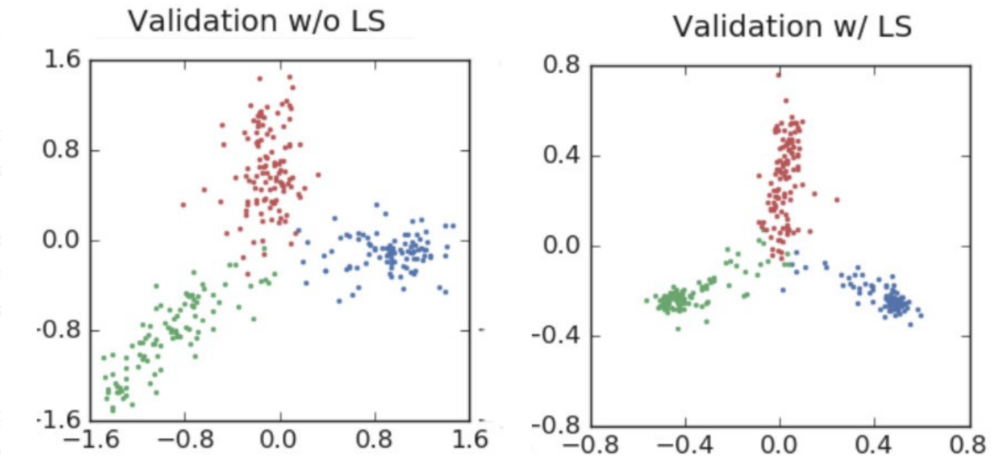


Figure 3.3: The effect of Label Smoothing on the similarity of representations [14]

Label Smoothing was first proposed by Christian Szegedy *et al.* [19]. They proposed a mechanism that a certain proportion of the prediction probability corresponding to the correct label is evenly distributed to each label, thereby reducing the model's confidence in its own prediction results. This mechanism not only regularizes the model but also makes it more adaptable.

Later, Geoffrey Hinton *et al.* [14] delved into how Label Smoothing affects representation learning and calibration of models. They found that Label Smoothing can improve

the similarity of representations of the same class and the dissimilarity of representations of different classes. The Figure 3.3 shows the effect of Label Smoothing on the similarity of representations. Besides that, Label Smoothing can make the model more calibrated, resulting in better recognition performance in the decoding stage.

Recently, Mun-Hak Lee *et al.* [12] published a study on the calibration of sequence-to-sequence models for speech recognition. By evaluating the calibration of the model, they found that the LAS and CTC models have problems of overconfidence in their predictions, and used techniques such as Label Smoothing to calibrate the models can obtain better recognition performance.

All the above previous works are based on classification models for computer vision or machine translation models for natural language processing, and recent research is also based on CTC and LAS models. There is not much research based on the joint CTC attention model yet. Therefore, in this report, I will focus on the impact of Label Smoothing on the joint CTC attention model by evaluating model calibration.

Chapter 4

Measuring Impacts of Heuristics

There are many ways to evaluate the impact of various heuristics on sequence-to-sequence models. The most common metrics are WER and CER that measure the accuracy of model predictions. In addition, there are some more advanced evaluation metrics, such as model calibration, which is used to measure the confidence of the model in the prediction, as well as measuring the exposure bias of the decoder, etc. In this chapter, I will introduce the definitions and formulations of these mentioned evaluation methods, and how I will use these methods in experiments to evaluate the impact of heuristics on model performance.

4.1 CER and WER

Character Error Rate (CER) and Word Error Rate (WER) are common evaluation criteria for evaluating the performance of speech recognizers. Both evaluation criteria are based on Levenshtein distance. Levenshtein distance is the minimum number of edits (insertions, deletions, or substitutions) of a word or character required to change one sequence to another. CER calculates the error by calculating the Levenshtein distance between the predicted sentence and the reference sentence at the character level, and dividing by the length of the reference sentence. WER uses the same calculation method as CER, but calculates the error at the word level. The definitions of CER and WER can be represented by the following formulas:

$$WER/CER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C} \tag{4.1}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, C is the number of correct words/characters, N is the number of words/characters in the reference.

The reason for using WER and CER to evaluate the performance of the speech recognizer is that usually the predicted sentence length of the speech recognizer is inconsistent with the reference sentence length, this is because the decoder cannot know the true length of the predicted sentence and can only stop when the EOS tag is output.

This will lead to some other evaluation criteria, such as accuracy, which need a one-to-one correspondence, cannot be used for the evaluation of speech recognizers. Using Levenshtein distance can avoid this problem and be more in line with the evaluation criteria for sequences.

4.2 Calibration

Calibration is the difference between the model's output probability and the likelihood of the actual correct answer. It is often used to assess the confidence of the model in the predicted results. Calibration has been proposed in the last century and has only been used in the evaluation of deep learning models until the last few years. In 2017, Chuan Guo *et al.* [7] proposed to use reliability diagram to visualize model calibration and use expected calibration error to evaluate model calibration. The reliability diagram is a histogram that evaluates the average accuracy within each probability range based on the probabilities of all output samples of the model. It can clearly see the gap between the average probability and the average accuracy of the samples in each probability range. This gap indicates that the model is somehow overconfident or underconfident in its predictions. The Figure 4.1 shows a schematic diagram of reliability diagram.

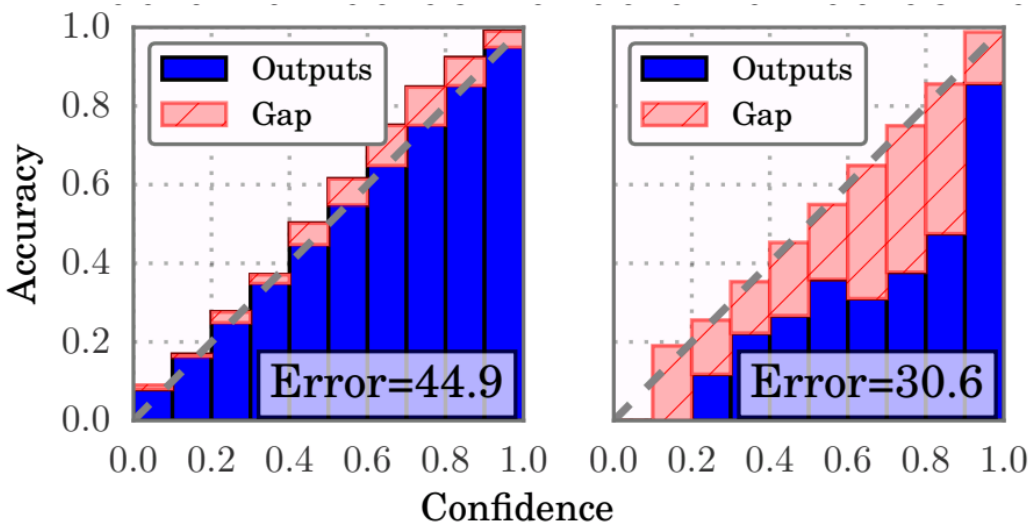


Figure 4.1: Schematic diagram of Reliability Diagram [7]

The expected calibration error (ECE) is approximated by dividing the predicted probabilities of all samples into M equal probability ranges and taking the weighted average of the differences in the accuracy and confidence of the samples in each probability range. The definitions of ECE can be represented by the following formulas:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{4.2}$$

where n represents number of samples, m represents number of classes, $acc(B_m)$ and $conf(B_m)$ represents bin-wise accuracy and confidence.

Although ECE and the reliability diagram are good for evaluating model calibration, these methods still suffer from some drawbacks. First, ECE only considers the difference between all predicted probabilities of the model and the actual accuracy, and does not consider model calibration in different classes. Second, the use of uniform binning will cause the problem that the data samples are not balanced in different bins. This is because the vast majority of predicted probabilities will be concentrated in a larger probability range, while only a few predicted probabilities will be distributed in a smaller probability range. This results in only a few probability ranges that make a major contribution to ECE.

To address these issues, Jeremy Nixon *et al.* [15] proposed Adaptive Calibration Error. Adaptive Calibration Error uses bins with the same number of samples to calculate the difference between average probability and average accuracy, rather than using different probability ranges. In addition, it computes the calibration distribution for each class separately and takes the average instead of using all the predicted values directly. The definition of Adaptive Calibration Error can be represented by the following formula:

$$ACE = \frac{1}{KR}\sum_{k=1}^{K}\sum_{r=1}^{R}|acc(r,k) - conf(r,k)| \tag{4.3}$$

where $acc(r,k)$ and $conf(r,k)$ are the accuracy and confidence of adaptive calibration range r for class label k, respectively.

In this thesis, I will use both ECE and ACE along with the reliability diagram to evaluate the calibration of the model. And investigate whether Label Smoothing and other heuristics have an impact on the calibration of the model.

4.3 Uncertainty

Uncertainty means that when the model performs Monte Carlo sampling on a test sentence in the inference stage, it usually has different errors in different sampling results, and these errors can well estimate the uncertainty of the model. In recent years, Yarin Gal *et al.* [5] proposed that Dropout can be used to model the uncertainty of the model by means of Bayesian approximation. They found that the model after Monte Carlo sampling by Dropout can well reflect the uncertainty of the model for certain predictions, and the average result of multiple sampling can make prediction of the model more stable, so as to accurately estimate the error of the model prediction. The Figure 4.2 shows an example using Dropout uncertainty to locate errors.

Later, Apoorv Vyas *et al.* [20] took advantage of this property and used Dropout on CTC-based speech recognizers to estimate the error rate of the speech recognizer and locate specific error locations. More specifically, they use Dropout to sample multiple different models during inference and use these models to infer the same test utterance. Since the neurons randomly selected by Dropout are different each time, the inference results of the model will also be different, but only some parts will differ between inference results. These parts are usually caused by the model being uncertain about their predictions. Therefore, they can use these parts to estimate the uncertainty of the model, and use them to locate where the model predictions are wrong.

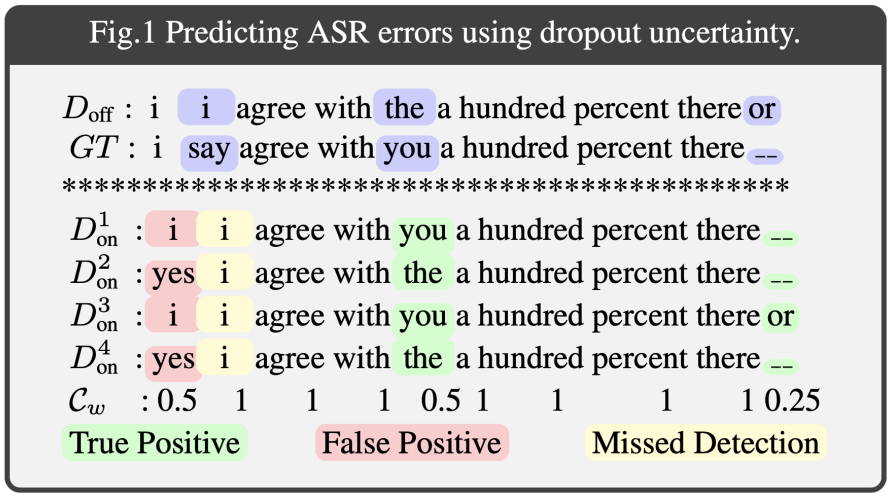


Figure 4.2: Using Dropout Uncertainty to Locate Errors [20]

To estimate model uncertainty, the word error rate estimation is used to quantify model uncertainty. The word error rate estimation first chooses an utternace as input and generates a certain number of inference results through Dropout, then calculates the edit distance of them in pairs, and keeps the pairs with the largest edit distance to calculate their average edit distance and average sentence length. Finally, the ratio of the average edit distance to the average sentence length is used as the word error rate estimation of that utternace. The definition of word error rate estimation can be expressed by the following formula:

$$ {}^{i}E_{\mu} = \frac{\sum_{j=1}^{K} {}^{i}_{k}E_{j}}{K} \tag{4.4} $$

$$ {}^{i}L_{\mu} = \frac{\sum_{j=1}^{K} {}^{i}_{k}L_{j}}{K} \tag{4.5} $$

$$ {}^{i}WER = \frac{{}^{i}E_{\mu}}{{}^{i}L_{\mu}} \tag{4.6} $$

$$ WER_{data} = \frac{\sum_{i=1}^{|D|} {}^{i}E_{\mu}}{\sum_{i=1}^{|D|} {}^{i}L_{\mu}} \tag{4.7} $$

where ${}^{i}E_{\mu}$ represents average edit distance and ${}^{i}L_{\mu}$ represents average sentence length. In addition, the word error rate estimation for the entire dataset can be expressed using the ratio of the average edit distance to the average sentence length for all predicted utterances.

To locate model errors, the word-level confidence is used to quantify model errors. The word-level confidence calculates what proportion of the inference results of each word after Dropout sampling is consistent with the inference results without Dropout. The definition of word-level confidence can be expressed by the following formula:

$$ C_w = \frac{\sum_{k=1}^{N} I(D^k_{on}[w] = D_{off}[w])}{N} \tag{4.8} $$

where $D^k_{on}[w]$ represents the word predicted by the inference result of the kth sampled model and $D_{off}[w]$ represents the word predicted by the inference result of the model without Dropout. It's worth noting that this method requires word-to-word alignment, so it can only be computed by using teacher forcing during decoding.

4.4 Exposure Bias

Exposure bias is caused by train-test discrepancy that arises when an auto-regressive generative model uses only the ground truth context during training and the generated context during testing [17]. The sequence-to-sequence speech recognition model also uses auto-regressive generative model as decoder, so it will also be affected by exposure bias. The most typical solution to exposure bias is Scheduled Sampling proposed by Samy Bengio *et al.* [2]. The Scheduled Sampling replaces the ground truth input with the previous prediction output with a certain probability in the training phase, so as to alleviate the mismatch problem in the training and inference phase. The Figure 4.3 shows a schematic diagram of exposure bias.

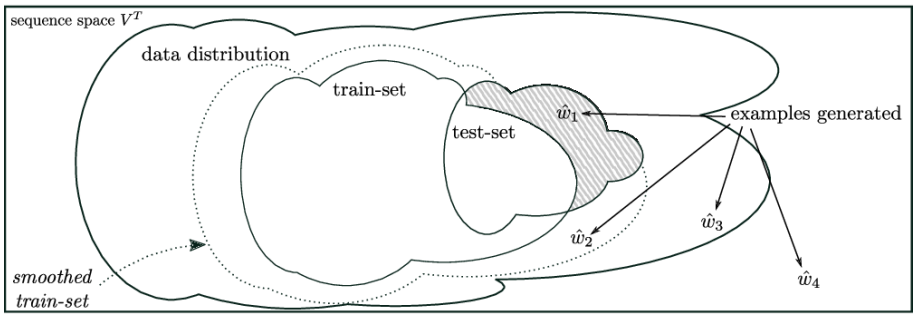


Figure 4.3: Schematic diagram of Exposure Bias [17]

Although many speech recognition models use Scheduled Sampling to improve the mode performance, few studies have evaluated the extent to which Scheduled Sampling mitigates exposure bias. Therefore, I decided to use random replacement of the generated labels to influence the decoding performance of the speech recognizer, determine whether the model is affected by exposure bias by evaluating the model's resilience to erroneous sequences, and test whether Scheduled Sampling can effectively reduce the effect of exposure bias on the model.

In this thesis, I will use CER/WER to evaluate the resilience of a model with and without Scheduled Sampling to random errors, to assess how well Scheduled Sampling can alleviate the problem of exposure bias.

4.5 Ability to Stop

Ability to Stop is a common metric used to measure the performance of decoders based on the auto-regressive generative model. This type of decoder cannot know the exact stop position in advance, the stop position can only be determined by the EOS tag.

This results in many times the decoder producing erroneous and verbose results due to its inability to output EOS tags. The inability or difficulty of the decoder to stop is the main reason for the poor performance of speech recognizers when evaluated on datasets. Therefore, it is important to evaluate the stopping ability of the decoder.

In the past few years, some studies have shown that the attention mechanism of sequence-to-sequence models affects the stopping ability of the decoder. Jan K. Chorowski *et al.* [4] proposed a Location-aware attention mechanism, which can still perform well in the case of decoding long sentences. Before his research, the commonly used dot product attention mechanisms and content based attention mechanisms could not achieve ideal results in the encoding of long sentences, mainly because the attention mechanism was too flexible and it is difficult to match the current decoding result with the correct encoder output, resulting in limited stopping ability of the decoder.

Besides the optimization of the attention mechanism, many other factors may also affect the stopping ability of the decoder. In theory, any heuristic that improves the performance of the model should improve the stopping ability of the decoder more or less, because the performance improvement is often related to whether the decoder is decoding in the right place.

In this thesis, I will use the difference between the actual sentence length and the decoded sentence length to evaluate the stopping ability of the decoder in the joint CTC attention model. Furthermore, I will explore whether different heuristics also have an effect on the ability of the decoder to stop.

Chapter 5

Experiments

In this chapter, I will first introduce the data used for the experiments - the Wall Street Journal, and then I will introduce the specific implementation details of the model architecture and hyperparameters used to train the model. Finally, I will show the experimental results of the impact of each heuristic on the model performance. My baseline model is the joint CTC attention model and the heuristics I will study are Dropout, Scheduled Sampling, and Label Smoothing. I mainly study the impact of these heuristics on CER/WER, learning curve, model calibration, exposure bias, and ability to stop. My experimental results found that the heuristic has some impact on the above properties of the model, and the joint CTC attention model itself has some defects. These heuristics more or less make up for the defects of the model and improve the model performance.

5.1 Benchmark Dataset

In the experiments, I used the Wall Street Journal dataset as the benchmark dataset to evaluate the experimental results. The Wall Street Journal is a speech corpus designed to facilitate the development and evaluation of large-vocabulary, speaker-independent, continuous speech recognition systems. It primarily consists of reading speech with text extracted from a machine-readable corpus of Wall Street Journal news text. I used a clean corpus with character labels - WSJ-si284 as the dataset and split it into 90% and 10% as training and validation sets. In the experiment, I used the same data as the validation set and test set, because the hyperparameters of the model completely followed the paper [11], so the hyperparameters of the model were not adjusted using the validation set. For preprocessing of input data, I used 40-dimensional log Mel spectrograms, with their first and second order temporal derivatives to obtain a total of 120-dimensional feature values per frame as input. For labels, I followed Suyoun Kim *et al.* [11] to use 33 distinct labels: 26 characters, apostrophe, period, dash, space, noise, sos/eos tokens and an extra blank label for CTC loss.

5.2 Model Architecture

For the joint CTC attention model, the encoder consists of a 4-layer BiLSTM with 320 cells in each layer and direction, as well as the projection layer on the top of the encoder to map the output of BiLSTM to 320-dimensional features. The second and third BiLSTM layers subsamples the output by half, reducing the frame length of the encoder output by a factor of 4. The CTC loss funcion is applied to the top of the encoder after projection layer. The decoder consists of a layer of unidirectional LSTM with 320 cells and there is a softmax layer on the top of the decoder. A location-aware attention mechanism is used between the decoder and the encoder. 10 centered convolution filters of width 100 were used to extract the convolutional features. I used the sharpening factor $\gamma = 2.0$ to enhance the alignment score. For the decoding of the model, I used greedy search decoder which takes the sequence with the most likely output. I did not use any dictionaries or language models in the experiments.

5.3 Training Settings

During the training phase, the Adam optimizer with gradient clipping was used for optimization. The learning rate of the optimizer is 0.001, and the max_norm of gradient clipping is 1.0. I used a single P100 GPU to train the model and the total training iterations are 15 epochs with batch size 16. I used exponential learning rate scheduler which multiplies the learning rate by 0.9 after each epoch to make the model converge better. The entire training process uses a self-designed training framework, and all models used for the experiment are based on the model parameters of the 15th epoch.

5.4 Experimental Results

In this section, I will present my experimental results for different heuristics. The heuristics I studied are Dropout, Scheduled Sampling and Label Smoothing. I will present my experimental results with the impact of these heuristics on various properties of the joint CTC attention model with $\lambda = 0.2$. These properties are model calibration, exposure bias, and ability to stop. In addition, I will also show how different heuristics improve model performance through CER/WER and learning curve. Based on these experimental results, I will analyze whether the joint CTC attention model itself has some defects and how different heuristics affect the model's property. The hyperparameters of the model used for all experiments are shown in Table 5.1.

The Table 5.1 shows 7 models using different heuristics. Each heuristic has two models, using two different hyperparameters. I will compare the model using each heuristic with the baseline, so as to analyze the influence of each heuristic on the properties of the model. In the next few sections, I will show the performance impact of models using different heuristics in terms of CER/WER, learning curve, model calibration, exposure bias and ability to stop.

Model ID	Dropout Rate	Scheduled Sampling Rate	Label Smoothing Rate
1	0.00	0.00	0.00
2	0.20	0.00	0.00
3	0.40	0.00	0.00
4	0.00	0.10	0.00
5	0.00	0.20	0.00
6	0.00	0.00	0.05
7	0.00	0.00	0.10

Table 5.1: Model hyperparameters used for all experiments

5.5 Dropout Regularization

5.5.1 CER/WER

In this section, I will analyze the performance of the model by character error rate and word error rate. The following Table 5.2 shows a comparison of the character error rate and word error rate for the baseline model and models using different Dropout rates.

Model ID	Dropout Rate	Character Error Rate	Word Error Rate
1	0.00	0.1077	0.2483
2	**0.20**	**0.0988**	**0.2219**
3	0.40	0.1055	0.2289

Table 5.2: CER and WER for models using Dropout and baseline

From the table, we can find that the best character error rate and word error rate are obtained when the Dropout rate is 0.20. In addition, we can also find that the two metrics can also be improved to a certain extent from baseline when using a Dropout rate of 0.40. This shows that Dropout can indeed improve the performance of the joint CTC attention model. Based on such experimental results, I will delve into the reasons for the improvement in terms of the learning curve of the model.

5.5.2 Learning Curve

In this section, I will analyze the reason why Dropout improves model performance by comparing the learning curve of the baseline and the model with a Dropout rate of 0.20. The learning curves are shown in Figure 5.1. We can find that the overall trend of the learning curve of the Dropout model and the baseline model is consistent, the only difference is that the gap between each training loss and validation loss is reduced compared to the baseline model. This means that Dropout reduces the variance while maintaining the bias of the original model, thereby reducing overfitting. This experimental result verifies that Dropout improves the performance of the model by reducing the overfitting of the model.

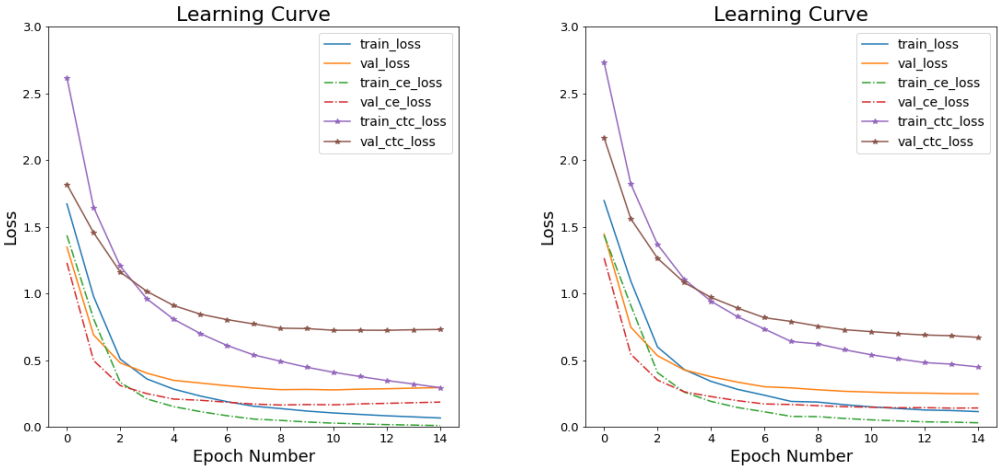


Figure 5.1: The left figure shows the learning curve of baseline model and the right figure shows the learning curve of the model using Dropout rate 0.20.

5.5.3 Calibration

In this section, I will analyze the reason why Dropout improves model performance by comparing the reliability diagram and ECE/ACE scores of the baseline and the model with a Dropout rate of 0.20. The Figure 5.2 below shows the reliability diagram based on ECE score.

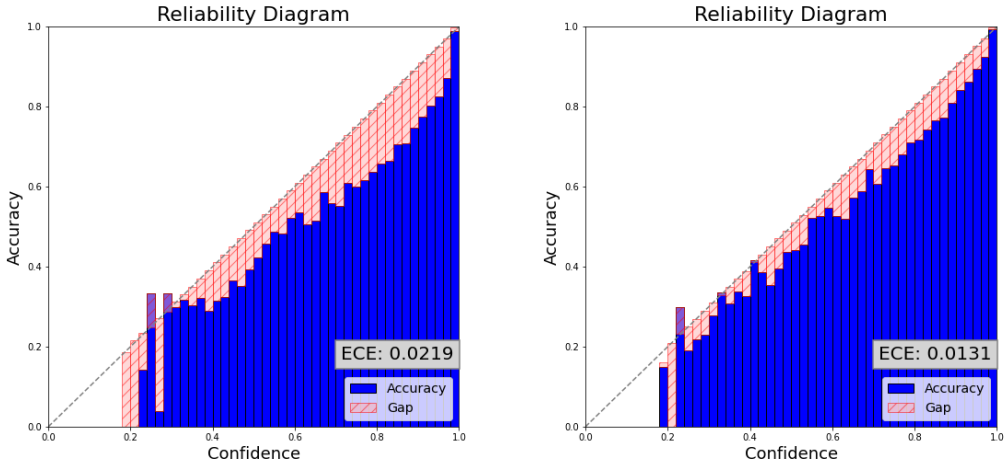


Figure 5.2: The left figure shows the reliability diagram of baseline and the right figure shows the model using Dropout rate 0.20. (ECE)

From the figure, we can find that the gap between accuracy and confidence score is reduced in model using Dropout, compared to the baseline model. And the ECE score dropped from 0.0219 to 0.0131 which means Dropout can be helpful for model calibration. We can also find similar results on reliability diagram based on ACE score in Figure 5.3. We can find that the ACE score drops from 0.0593 to 0.0563 and the gap

of some bins with smaller confidence scores is reduced. This is also reflected in the ECE reliability diagram.

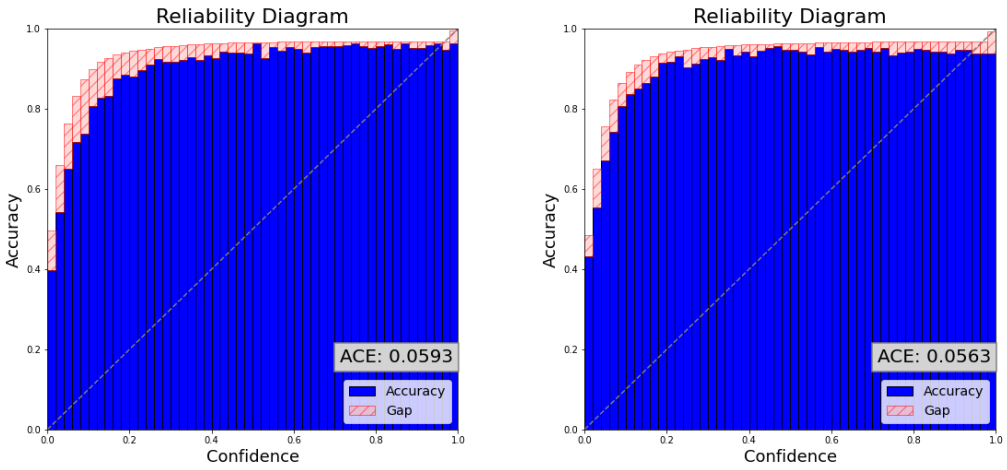


Figure 5.3: The left figure shows the reliability diagram of baseline and the right figure shows the model using Dropout rate 0.20. (ACE)

From the above calibration results, we can find that the main reason for the reduction of the gap of bins with lower confidence score is the improvement of accuracy. This means that Dropout improves the accuracy of some predictions with low confidence scores. This is in line with Dropout's ability to reduce uncertainty, because the lower the confidence score, the more uncertain the prediction results. Therefore, Dropout enhances model calibration by reducing the uncertainty of model predictions.

5.5.4 Exposure Bias

In this section, I will use CER/WER to evaluate the prediction accuracy of the model with and without Dropout when the wrong input is randomly generated during decoding, in order to assess how well Dropout can alleviate the problem of exposure bias. The following Table 5.3 shows a comparison of the character error rate and word error rate for the baseline model and models using different Dropout rates. Note that the noise ratio means the probability to generate the incorrect input at each time step during decoding.

Noise Ratio	Dropout Rate	Character Error Rate	Word Error Rate
0.05	0.00	**0.2181**	0.4957
0.05	0.20	0.2220	**0.4862**
0.05	0.40	0.2361	0.4970
0.10	0.00	**0.3097**	0.6595
0.10	0.20	0.3236	**0.6556**
0.10	0.40	0.3463	0.6741

Table 5.3: CER and WER for models using Dropout and baseline

We can clearly find that the baseline model achieves the best character error rate when the noise ratio is 0.05 and 0.10. The best word error rate at both noise ratio is obtained by a model with a Dropout rate of 0.20. In general, compared to the baseline model, the model using Dropout is more susceptible to exposure bias. This shows that the exposure bias cannot be reduced by using Dropout.

5.5.5 Ability to Stop

Table 5.4 below shows the number of sentences for which the decoder did not stop on the validation set for the baseline and the model using Dropout. I set 3 different length gap to judge whether the sentence stops or not. They are the number of sentences for which the value of the gap between the predicted data length and the ground truth data length is greater than 10, 20, and 100, respectively.

Length Gap	Dropout Rate	Non-Stop Number
10	0.00	56
10	**0.20**	**36**
10	0.40	57
20	0.00	6
20	**0.20**	**2**
20	0.40	12
100	0.00	5
100	**0.20**	**1**
100	0.40	6

Table 5.4: Number of non-stop sentences for models using Dropout and baseline

From the table, we can find that a model with a Dropout of 0.20 generally outperforms the baseline in stopping ability, but a model with a Dropout of 0.40 will get a worse stopping ability than the baseline. At the same time, we can also find that the performance of the previous CER/WER part when the Dropout is 0.40 is worse than Dropout 0.20. This is most likely because using an excessively large Dropout value will oversimplify the complexity of the model, resulting in performance degradation.

5.6 Scheduled Sampling

5.6.1 CER/WER

Table 5.5 below shows a comparison of the character error rate and word error rate of the baseline model and the model using different Scheduled Sampling rates.

From the table, we can find that the best character error rate and word error rate are from different models. The best character error rate is obtained by the model using Scheduled Sampling rate of 0.20, and the best word error rate is obtained by the baseline model. In addition, we can also find that the character error rate can also be improved to a certain extent from baseline when using a Scheduled Sampling rate of 0.10. However, all the models using Scheduled Sampling achieved a worse word error

Model ID	Scheduled Sampling Rate	Character Error Rate	Word Error Rate
1	0.00	0.1077	**0.2483**
2	0.10	0.1057	0.2563
3	0.20	**0.1043**	0.2642

Table 5.5: CER and WER for models using Scheduled Sampling and baseline

rate than the baseline model. This shows that Scheduled Sampling can indeed improve the character error rate of the joint CTC attention model but worse the word error rate of the joint CTC attention model. In addition, we can find that the higher the Scheduled Sampling rate, the lower the character error rate, but the higher the word error rate. This means that Scheduled Sampling may be able to avoid exposure bias on the model at the cost of damaging word integrity. Based on such experimental results, I will delve into the reasons for the improvement of character error rate and deterioration of word error rate in terms of the learning curve of the model.

5.6.2 Learning Curve

In this section, I will analyze the reason why Scheduled Sampling improves the model performance by comparing the learning curve of the baseline and the model with a Scheduled Sampling rate of 0.20. The learning curves are shown in Figure 5.4. We can find that the overall trend of the CTC loss curves of the Scheduled Sampling model and the baseline model is consistent. However, the cross entropy curves are elevated compared to the baseline. This is because the ground truth output is not completely used in the training phase, but the previous output is used as the input of the model with a certain probability. This will make the training of the early epoch more difficult, because it is easy to introduce more wrong inputs, thereby increasing the cross entropy.

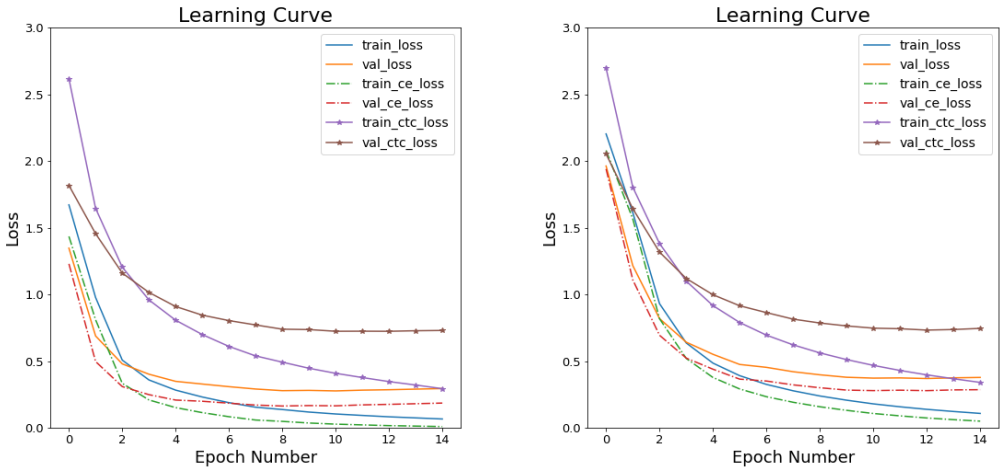


Figure 5.4: The left figure shows the learning curve of baseline model and the right figure shows the learning curve of the model using Scheduled Sampling rate 0.20.

Although the learning curve using Scheduled Sampling delays the premature conver-

gence of the entire training process compared to the baseline model, no substantial difference can be seen. Therefore, I will delve into the impact of Scheduled Sampling on the joint CTC attention model through an analysis of model calibration.

5.6.3 Calibration

In this section, I will analyze the reason why Scheduled Sampling improves the character error rate of the model by comparing the reliability diagram and ECE/ACE scores of the baseline and the model with a Scheduled Sampling rate of 0.20. The Figure 5.5 shows the reliability diagram based on ECE score.

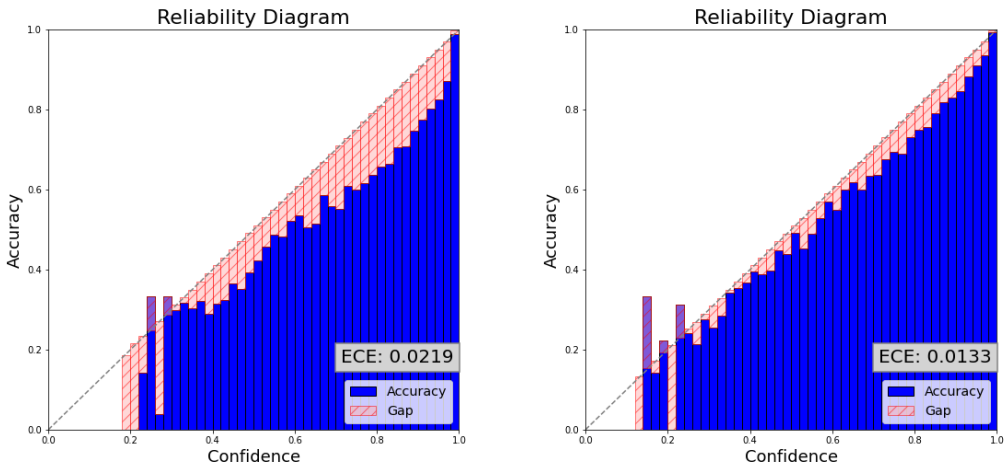


Figure 5.5: The left figure shows the reliability diagram of baseline and the right figure shows the model using Scheduled Sampling rate 0.20. (ECE)

From the figure, we can find that the gap between accuracy and confidence score is reduced in model using Scheduled Sampling, compared to the baseline model. And the ECE score dropped from 0.0219 to 0.0133 which means Scheduled Sampling can be helpful for model calibration. We can also find similar results on reliability diagram based on ACE score. The Figure 5.6 shows the reliability diagram based on ACE score. We can find that the ACE score drops from 0.0593 to 0.0530 and the gap of some bins with smaller confidence scores is reduced. This is also reflected in the ECE reliability diagram.

From the above calibration results, we can find that the main reason for the decrease in the gap of bins with lower confidence scores is the decrease in the confidence score of each bins when there is no change in accuracy. This is in line with Scheduled Sampling mechanism, because the ground truth label is not fully used for training, instead, some of the input is replaced by the previous output. This can lead to some wrong inputs that affect the prediction accuracy during training, so that the confidence score of the prediction results obtained through teacher force training is reduced. Therefore, Scheduled Sampling enhances model calibration by introducing input noise to the prediction results.

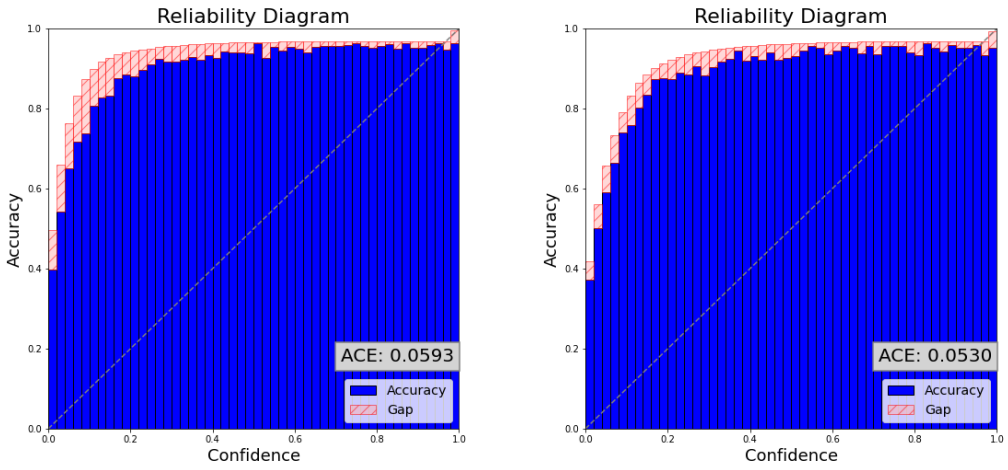


Figure 5.6: The left figure shows the reliability diagram of baseline and the right figure shows the model using Scheduled Sampling rate 0.20. (ACE)

5.6.4 Exposure Bias

In this section, I will use CER/WER to evaluate the prediction accuracy of the model with and without Scheduled Sampling when the wrong input is randomly generated during decoding, in order to assess how well Scheduled Sampling can alleviate the problem of exposure bias. The following Table 5.6 shows a comparison of the character error rate and word error rate for the baseline model and models using different Scheduled Sampling rates. Note that the noise ratio means the probability to generate the incorrect input at each time step during decoding.

Noise Ratio	Scheduled Sampling Rate	Character Error Rate	Word Error Rate
0.05	0.00	0.2181	0.4957
0.05	0.10	0.1997	0.4936
0.05	**0.20**	**0.1816**	**0.4907**
0.10	0.00	0.3097	0.6595
0.10	0.10	0.2901	0.6625
0.10	**0.20**	**0.2613**	**0.6557**

Table 5.6: CER and WER for models using Scheduled Sampling and baseline

We can clearly find that the model with a Scheduled Sampling rate of 0.20 achieves the best character error rate and word error rate. Similarly, we can also find that using a model with a Scheduled Sampling rate of 0.10 can also reduce the impact of exposure bias. This experimental results verifies that the model using Scheduled Sampling can well reduce the impact of exposure bias.

5.6.5 Ability to Stop

Table 5.7 below shows the number of sentences for which the decoder did not stop on the validation set for the baseline and the model using Scheduled Sampling. I set 3

different length gap to judge whether the sentence stops or not. They are the number of sentences for which the value of the gap between the predicted data length and the ground truth data length is greater than 10, 20, and 100, respectively.

Length Gap	Scheduled Sampling Rate	Non-Stop Number
10	0.00	56
10	0.10	38
10	**0.20**	**32**
20	0.00	6
20	0.10	10
20	**0.20**	**6**
100	0.00	5
100	0.10	4
100	**0.20**	**2**

Table 5.7: Number of non-stop sentences for using Scheduled Sampling and baseline

From the table, we can find that a model with a Scheduled Sampling rate of 0.20 generally outperforms the baseline in stopping ability. At the same time, we can also find that the stopping ability of the model using Scheduled Sampling rate of 0.10 mostly outperforms the baseline except when length gap is 20. This is most likely because the model using Scheduled Sampling is more immune to exposure bias, so it is not easy to fall into an infinite loop.

5.7 Label Smoothing

5.7.1 CER/WER

Table 5.8 below shows a comparison of the character error rate and word error rate of the baseline model and the model using different Label Smoothing rates.

Model ID	Label Smoothing Rate	Character Error Rate	Word Error Rate
1	0.00	0.1077	0.2483
2	**0.05**	**0.0969**	**0.2197**
3	0.10	0.0970	0.2260

Table 5.8: CER and WER for models using Label Smoothing and baseline

From the table, we can find that the best character error rate and word error rate are obtained when the Label Smoothing rate is 0.05. In addition, we can also find that the two metrics can also be improved to a certain extent from baseline when using a Label Smoothing rate of 0.10. This shows that Label Smoothing can indeed improve the performance of the joint CTC attention model. Based on this results, I will delve into the reasons for the improvement in terms of the learning curve of the model.

5.7.2 Learning Curve

In this section, I will analyze the reason why Label Smoothing improves the model performance by comparing the learning curve of the baseline and the model with a Label Smoothing rate of 0.05. The learning curves are shown in Figure 5.7. We can find that the overall trend of the CTC loss curves of the Label Smoothing model and the baseline model is consistent. However, the entire cross-entropy curves are extremely elevated compared to the baseline model. This verifies that the Label Smoothing penalizes the predicted probability so that the cross-entropy at training time increases overall. Although the learning curve of using Label Smoothing is very different from the learning curve of the baseline, it is not clear how the performance of the model can be improved by Label Smoothing. Therefore, I will delve into the impact of Label Smoothing on the joint CTC attention model through an analysis of model calibration.

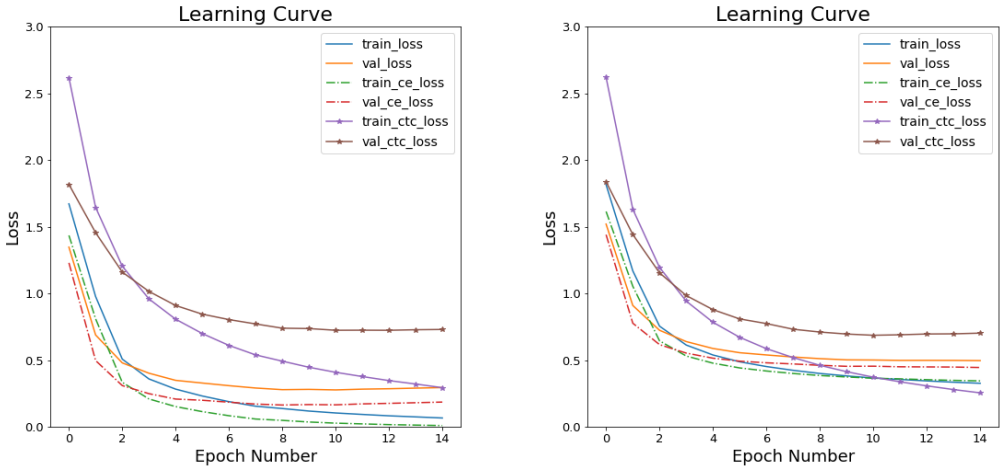


Figure 5.7: the left figure shows learning curve of baseline and the right figure shows learning curve of model using Label Smoothing rate 0.05.

5.7.3 Calibration

In this section, I will analyze the reason why Label Smoothing improves model performance by comparing the reliability diagram and ECE/ACE scores of the baseline and the model with a Label Smoothing rate of 0.05. The Figure 5.8 below shows the reliability diagram based on ECE score.

From the figure, we can find that the gap between accuracy and confidence score is increased in model using Label Smoothing, compared to the baseline model. Unlike Dropout and Scheduled Sampling, the overall accuracy score is higher than the confidence score in each bins, and the ECE score increases from 0.0219 to 0.0393 which means Label Smoothing can not be helpful for model calibration in joint CTC attention model. We can also find similar results on reliability diagram based on ACE score in Figure 5.9. We can find that the ACE score increases from 0.0593 to 0.0785 and the gap of some bins with smaller confidence scores is increased. This is also reflected in the ECE reliability diagram.

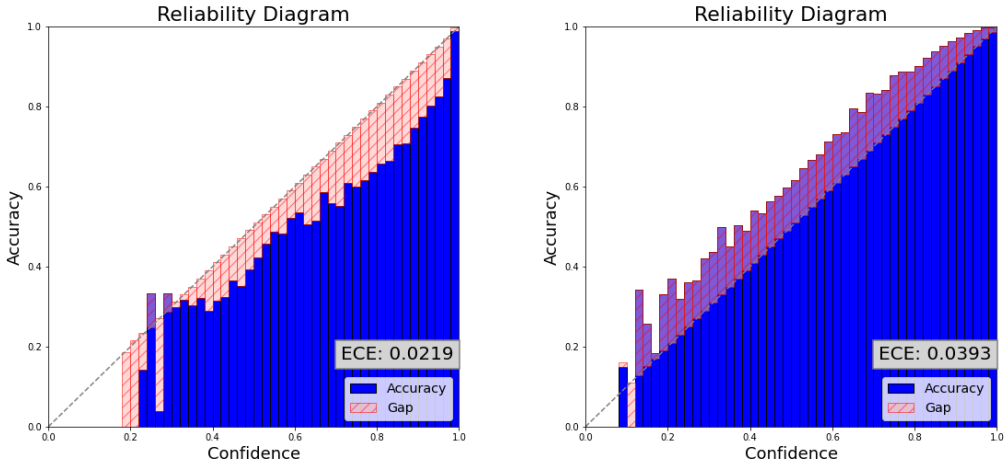


Figure 5.8: the left figure shows the reliability diagram of baseline and right figure shows the reliability diagram of Label Smoothing rate 0.05. (ECE)

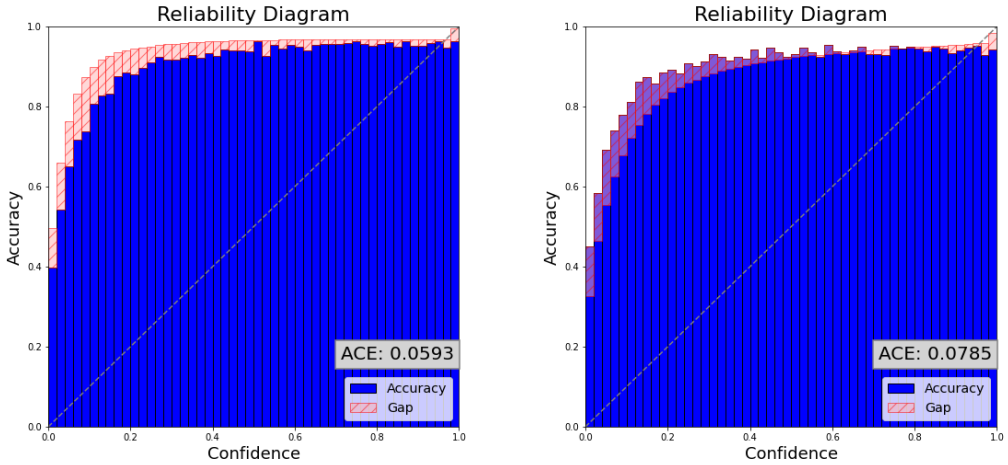


Figure 5.9: the left figure shows the reliability diagram of baseline and right figure shows the reliability diagram of Label Smoothing rate 0.05. (ACE)

Although the calibration of the model is found to be worse than that of baseline model from the figure, this is indeed a good result for the joint CTC attention model. From the baseline model, we can see that the confidence score of the model is basically higher than the accuracy, indicating that the model is too confident in its own prediction results, while the confidence score of the model after using Label Smoothing is basically lower than the accuracy, indicating that the model become less confident in its own predictions. This shows that the label smoothing technique can effectively solve the overconfidence problem of the model.

5.7.4 Exposure Bias

In this section, I will use CER/WER to evaluate the resilience of a model with and without Label Smoothing to random errors, to assess how well Label Smoothing can alleviate the problem of exposure bias. The following Table 5.9 shows a comparison of the character error rate and word error rate for the baseline model and models using different Label Smoothing rates.

Noise Ratio	Label Smoothing Rate	Character Error Rate	Word Error Rate
0.05	0.00	0.2181	0.4957
0.05	0.10	0.2195	0.4764
0.05	**0.20**	**0.2122**	**0.4741**
0.10	0.00	**0.3097**	0.6595
0.10	0.10	0.3311	0.6568
0.10	0.20	0.3184	**0.6544**

Table 5.9: CER and WER for models using Label Smoothing and baseline

We can clearly find that the baseline model achieves the best character error rate when the noise ratio is 0.10, but the best word error rate is obtained by a model with a Label Smoothing rate of 0.10. The model using Label Smoothing of 0.05 achieves the best character error rate and word error rate at the same time when the noise ratio is 0.05. In general, compared to the baseline model, the model using Label Smoothing slightly reduce the exposure bias. This shows that the exposure bias can be reduced by using Label Smoothing.

5.7.5 Ability to Stop

Table 5.10 below shows the number of sentences for which the decoder did not stop on the validation set for the baseline and the model using Label Smoothing. I set 3 different length gap to judge whether the sentence stops or not. They are the number of sentences for which the value of the gap between the predicted data length and the ground truth data length is greater than 10, 20, and 100, respectively.

Length Gap	Label Smoothing Rate	Non-Stop Number
10	0.00	56
10	0.05	47
10	**0.10**	**43**
20	0.00	6
20	**0.05**	**6**
20	0.10	7
100	0.00	5
100	**0.05**	**3**
100	0.10	4

Table 5.10: Number of non-stop sentences for using Label Smoothing and baseline

From the table, we can find that a model with both Label Smoothing rate of 0.05 and 0.10 generally outperforms the baseline in stopping ability. This is most likely because the model using Label Smoothing greatly improves the accuracy of predictions with high uncertainty, making it easier for the decoder to get correct sentence without looping it.

Chapter 6

Conclusions

In this report, I implement a joint CTC attention model as a baseline and design experiments to evaluate the defects of the model itself and study how Dropout, Label Smoothing and Scheduled Sampling compensate for these defects as well as study how they affect calibration, exposure bias, ability to stop of the model to explain why these heuristics can improve model performance.

Below is a summary of the results.

- Dropout, Scheduled Sampling and Label Smoothing can improve generalization.
- Dropout does reduce model overfitting and can enhance model calibration by reducing uncertainty in model predictions. Dropout can also improve the stopping ability of the decoder, but it will worsen the exposure bias problem of the model.
- Scheduled Sampling enhance model calibration and effectively reduces exposure bias problem, as well as improving the stopping ability of the decoder.
- Label Smoothing can be used to control the confidence of the models, but does not necessarily improves calibration. Label Smoothing can slightly reduce exposure bias problem and improve the stopping ability of the decoder.

In future work, I will study the performance impact of heuristics on more advanced models, such as speech transformers and conformers. I can also compensate for those defects by designing new losses or components based on the defects in the model.

Bibliography

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[3] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[4] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28, 2015.

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[8] Awni Hannun. Sequence modeling with ctc. *Distill*, 2(11):e8, 2017.

[9] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839, 2017.

[12] Mun-Hak Lee and Joon-Hyuk Chang. Deep Neural Network Calibration for E2E Speech Recognition System. In *Proc. Interspeech 2021*, pages 4064–4068, 2021.

[13] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[14] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[15] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshops*, volume 2, 2019.

[16] Christopher Olah. Understanding lstm networks. 2015.

[17] Florian Schmidt. Generalization in generation: A closer look at exposure bias. *arXiv preprint arXiv:1910.00292*, 2019.

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[19] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[20] Apoorv Vyas, Pranay Dighe, Sibo Tong, and Hervé Bourlard. Analyzing uncertainties in speech recognition using dropout. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6730–6734. IEEE, 2019.

[21] Shinji Watanabe, Takaaki Hori, Suyoun Kim, John R Hershey, and Tomoki Hayashi. Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253, 2017.