*INTERSPEECH 2017*
August 20–24, 2017, Stockholm, Sweden

# Annealed f-smoothing as a Mechanism to Speed Up Neural Network Training

*Tara N. Sainath, Vijayaditya Peddinti, Olivier Siohan, Arun Narayanan*

Google, Inc., U.S.A

{tsainath, vpeddinti, siohan, arunnt}@google.com

## Abstract

In this paper, we describe a method to reduce the overall number of neural network training steps, during both cross-entropy and sequence training stages. This is achieved through the interpolation of frame-level CE and sequence level SMBR criteria, during the sequence training stage. This interpolation is known as f-smoothing and has previously been just used to prevent overfitting during sequence training. However, in this paper, we investigate its application to reduce the training time. We explore different interpolation strategies to reduce the overall training steps; and achieve a reduction of up to 25% with almost no degradation in word error rate (WER). Finally, we explore the generalization of f-smoothing to other tasks.

## 1. Introduction

Reducing neural network training time continues to be a popular research area, with speedups coming both from improved hardware [1], as well as algorithmic improvements [2]. Reducing training time is particularly important at Google, where we train our networks on thousands of hours of data. We recently looked at reducing training time by training acoustic models at a lower frame rate (LFR) [3]. We found that we could predict output targets at a 30ms frame rate, with no loss in accuracy. In this paper, we look to extend the LFR state-of-the-art system and further reduce overall number of training steps[1].

Most state-of-the art neural network acoustic models are trained with some sort of sequence objective function, such as state-level minimum Bayes risk (SMBR), which is well matched to the sequential nature of the speech recognition problem [4, 5, 6, 7]. Typically, the objective function requires generating sequence information in the form of word or phone-level lattices. Lattice quality and initialization of neural networks play a critical role in SMBR training [5]. Hence SMBR recipes rely on cross-entropy (CE) pretraining to both initialize the SMBR model and to generate the high quality lattices. While sequence training from scratch has been recently demonstrated in [8], we have not yet been able to match our best systems using such techniques.

The state-of-the-art systems at Google are trained on large amounts of data ($>$ 10,000 hrs) and use some variant of an LSTM model [9]. Given the large amount of data, both cross-entropy and sequence training are expensive stages. Furthermore, as we will show in Section 4, the longer we CE train our LSTM model, the better the WER after sequence training. Therefore, switching to sequence training earlier, as was done in [10] with feed-forward DNNs, is not recommended in our training recipe. Based on our observations we hypothesize that training for longer duration, in both CE and sMBR stages, helps LSTM-type models in large data situations, since they are good at memorizing the sequential nature of the data [11].

[1] A training step is one step of SGD.

In this paper, we explore if overall training time can be reduced by interpolating the CE objective function with the sequence training objective function, a technique known as frame-level smoothing (f-smoothing) [5]. The concept of f-smoothing has been previously used by many research labs to regularize the sequence objective to prevent the model from learning only silence and prevent overfitting [5, 8]. However, our motivation in adopting f-smoothing is to reduce the overall number of training steps, something which has not been explored up to this point to the best of our knowledge.

We explore different objective function interpolation schemes to reduce overall training time on a $12,500$ hr VoiceSearch task. We start with the typical static weight f-smoothing, which results in a reduction in WER but not a reduction in training steps. Next, we explore various strategies to dynamically change the interpolation weights during training and identify that an exponentially decaying weight on CE allows us to reduce overall training steps by 25%, with almost no loss in WER.

Finally, we identify events which help us determine an appropriate time to switch from the CE objective function to the interpolated objective function. Specifically, we track the changes in the CE loss and check if we can switch as soon as the changes in CE fall below a threshold. We also verify if this generalizes to other LVCSR tasks.

The rest of this paper is organized as follows. In Section 2 we describe describe the interpolation schemes we explored. The experimental setup is described in Section 3 and the results are presented in Section 4. Finally, Section 5 concludes the paper and discusses future work.

## 2. The Interpolated Objective Function

This section describes different interpolation mechanisms we explored. Please note that we use a convex combination of the frame level CE and sequence level SMBR objective functions.

### 2.1. f-smoothing

The idea of frame-level smoothing (f-smoothing) is to interpolate the sequence and frame-level objective functions [5] as shown in Equation 1. Here, $\lambda$ is the interpolation weight which is often tuned empirically. As detailed in [12], when sequence training is initiated, the sequence training criteria improves while the frame accuracy degrades. They hypothesize that this overfitting problem is caused by the fact that the sequence is in a higher dimensional space than frames. This overfitting is addressed by smoothing the sequence-level criteria with a frame-discriminative criteria known as f-smoothing. It has been shown in several papers ([5, 6, 8]) that f-smoothing improves WER.

$$F_{TOT}(\theta;\lambda) = \lambda F_{CE}(\theta) + (1-\lambda)F_{SEQ}(\theta) \quad (1)$$

Our objective in this work is different than the past work



http://dx.doi.org/10.21437/Interspeech.2017-231

on f-smoothing, in that we attempt to reduce the training time using the interpolated objective function. In this section, we detail different approaches explored for determining $\lambda$.

### 2.2. Mechanisms of varying $\lambda$

#### *2.2.1. A decaying $\lambda$*

In our case, we intend to switch to the SMBR training stage as early as possible to reduce the overall training time. However experimental evidence suggests that prolonged CE training can help improve the overall performance; so we consider the option of switching to the interpolated objective function early, but with a higher weight on CE, and decaying this weight as training progresses. It is to be noted that in the existing f-smoothing recipes the value of $\lambda$ is kept constant during training.

The interpolated objective function at each step $s$ of training is shown in Equation 2, with $\lambda(s)$ indicating a changing interpolation weight as a function of the training step.

$$F_{TOT}(\theta, s) = \lambda(s)F_{CE}(\theta) + (1 - \lambda(s))F_{SEQ}(\theta) \quad (2)$$

We explored the use of an exponentially decaying $\lambda(s)$:

$$\lambda(s) = \alpha * d^{(s/s_d)}, \quad (3)$$

where $s$ is the number of global steps, $s_d$ is the decay time period, $d$ is the exponential constant and $\alpha$ is the initial decay. In the results section, we will detail the differences in WER for different choices of $d$, $\alpha$ and $s_d$.

#### *2.2.2. $\lambda$ Perturbation*

Instead of just decaying $\lambda$ which is the weight on the CE objective function, we look to increase the weight on CE, for a few steps, occasionally during training. Motivated by work on adverserial training [13], we are also curious to see if changing the objective function by injecting more weight on CE can help get us out of a local minima and improve overall training. The objective function is still the same as Equation 2, but the schedule of $\lambda$ is not strictly decaying anymore. In Section 4.3.2 we will give examples of different $\lambda$ perturbations considered.

#### *2.2.3. Predicting $\lambda$ using an LSTM*

Instead of empirically choosing a constant $\lambda$ or a decay schedule, we investigate if we can automatically learn $\lambda$. Motivated from the Neural Adaptive Beamforming (NAB) model [14] which uses an LSTM to predict weights during beamforming, we look to have an LSTM learn the $\lambda$. Specifically, given the input feature vector $\mathbf{x}$, we train an LSTM to predict $\lambda_{l,t}$ at each frame $t$. We average the $\lambda_{l,t}$ across all frames to get a $\lambda_l$ per utterance, and use this to determine the convex combination of objective functions. Further, in order to ensure that the optimal $\lambda$ learned is non-trivial (i.e., not 0 or 1), we also use $l_2$ regularization to ensure that the optimal weight learned is in a reasonable range. Equation 4 represents this modified cost function, where the $\beta$ term is the $l_2$ reguarlization penalty and $\bar{\lambda}$ is chosen to ensure that the predicted $\lambda_l$ stays in a reasonable range.

$$F_{TOT}(\theta) = \lambda_l(\mathbf{x})F_{CE}(\theta) + (1 - \lambda_l(\mathbf{x}))F_{SEQ}(\theta) + \beta(\lambda_l(\mathbf{x}) - \bar{\lambda})^2 \quad (4)$$

### 2.3. When to switch to f-smoothing

As we will show in Section 4, we are not able to train a competitive model from random initialization using the interpolated objective function. Hence CE pretraining is necessary; and determining the point of switch from the CE objective function to the interpolated objective function is a problem of interest. Intuitively, we should stop training with CE only when the change in the objective function starts diminishing. To test this, we calculate the change in the objective function every 50 million steps. When the change falls below a certain threshold, we will stop CE and continue with f-smoothing. Please note that this is similar to validation based early-stopping [15], however we switch as soon as the change is below a specified threshold and do not wait till the objective function deteriorates on the cross-validation data.

## 3. Experimental Details

Our main experiments are conducted on a 12,500 hour noisy training set consisting of 15 million English utterances. This data set is created by artificially corrupting clean utterances using a room simulator, adding varying degrees of noise and reverberation such that the overall SNR is between 0dB and 30dB, with an average SNR of 12dB. The noise sources are from YouTube and daily life noisy environmental recordings. All training sets are anonymized and hand-transcribed, and are representative of Google's voice search traffic.

Models trained on the noisy test set are evaluated on 5 test sets, each containing 13k utterances of about 11 hours each. All 5 sets use exactly the same utterances, but have varying levels of noise and reverberation. The eval sets include the "clean" test set that adds no noise, the simulated mild-noise test set that adds noise to clean set at SNRs between 5 to 30dB, and the simulated moderate-noise set that adds both reverberation and additive noise at SNRs ranging from 0 to 20 dB. The final two eval sets are "rerecorded sets"; the clean rerecorded set was obtained by rerecording the clean evaluation set in a room with mild reverberation ( 200 msec). The noisy rerecorded set was obtained by adding rerecorded noise to this set at SNRs ranging from 0 to 20 dB. In order to focus on high-level trends in this paper, we report results averaging across the five test sets.

All experiments in this paper use 128-dimensional log-mel features, computed with a 32-ms window and shifted every 10ms. Low frame rate (LFR) [3] models are used, where at the current frame $t$, these features are stacked with 3 frames to the left and downsampled to a 30ms frame rate. The neural network architecture consists of 5 LSTM layers with 600 cells per layer. During training, the network is fully unrolled the length of the utterance. In addition, the output state label is delayed by 5 frames, as we have observed that information about future frames improves the prediction of the current frame [3]. All neural networks are trained with the cross-entropy criterion, using asynchronous stochastic gradient descent (ASGD) optimization [2].

## 4. Results

### 4.1. Baseline : No f-smoothing

Our first set of experiments examine the change in WER when we switch from the CE to SMBR objective function after different number of CE training steps. Here a step is considered 1 step of SGD, which is computed using 1 utterance in our setup.

From Table 1 it can be seen that prolonged CE pretraining

before the switch to SMBR training is beneficial. It is important to note that all SMBR results are reported at ∼400M training steps. Further we also found that training for more CE steps does not further improve performance.

Table 1: *WER after SMBR training when switching from different points during CE training*

| CE Steps (M) | SMBR WER |
|---|---|
| 120 | 16.8 |
| 240 | 16.3 |
| 360 | 16.2 |
| 480 | 16.0 |
| 600 | 15.9 |
| 720 | 15.9 |

### 4.2. f-smoothing with a constant $\lambda$

Next, we compare WERs with and without f-smoothing, and at different switching points from CE, in Table 2. The optimal $\lambda$, determined from a tuning exercise, is also indicated in the table. It can be seen that f-smoothing improves WER, independent of the switching point. Once again, prolonged CE pretraining is shown to help even with f-smoothing.

Table 2: *WER with static f-smoothing for different CE Steps*

| CE Steps (M) | f-smoothing | WER |
|---|---|---|
| 360 | 0 | 16.2 |
| | 0.001 | 16.0 |
| 480 | 0 | 16.0 |
| | 0.01 | 15.9 |
| 600 | 0 | 15.9 |
| | 0.001 | 15.7 |

### 4.3. f-smoothing with a varying $\lambda$

Given that f-smoothing with a constant $\lambda$ does not help to reduce the overall number of training steps, we explore if varying $\lambda$ during training has any effect. For all of these experiments, we examined the behavior when sequence training is started after running 360M steps of CE pretraining.

#### *4.3.1. Exponentially Decaying $\lambda$*

In these experiments, we examine the WER behavior with a decaying $\lambda$, as illustrated in Figure 1. The yellow line in the figure plots the the WER during a training session with static f-smoothing ($\lambda = 0.001$), where we switched from CE pretraining after 600M steps. This is the best system identified in Section 4.2. The blue and red curves correspond to training sessions where we switch from CE pretraining after 360M steps and decay the value of $\lambda$ from 0.1. The blue curve corresponds to a session where $\lambda$ is decayed to 0 and the red curve corresponds to a session where a lower bound is placed on $\lambda$. This lower bound $\lambda = 0.001$ is the same value as the one used for f-smoothing with constant $\lambda$. It can be seen that lower bounding $\lambda$ is better than decaying it to 0, and that decaying it to 0 is actually very detrimental.

The main observation here is that we are able to match the performance of our best system by using a large initial $\lambda$ (0.1) and decaying it, even though we switch to SMBR training at an earlier CE step. We compare the WERs achieved when switching from CE pretraining at different steps in Section 4.4.

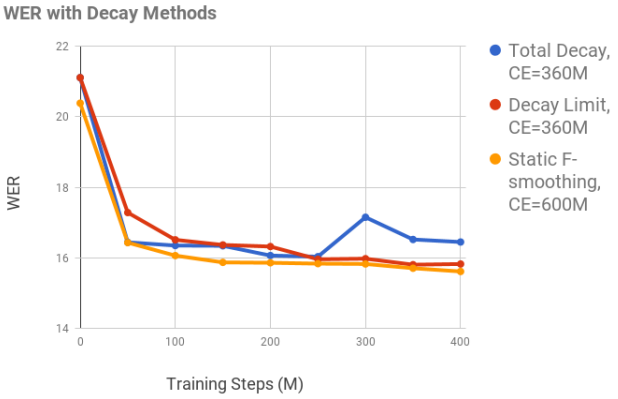


Figure 1: *WER measured during SMBR training with constant and decaying $\lambda$.*

Next, we explore the difference in WER for different choices of initial decay ($\alpha$) and decay time period ($s_d$) (see Equation 3). For these experiments, the exponential constant ($d$) is chosen to be 0.1. It appears that the best option is to decay from $\alpha = 0.1$ with a decay time period of $s_d = 100M$ steps. With these settings, we converge to the final f-smoothing value of 0.001 at around 200M steps, which is approximately halfway through sequence training.

Table 3: *WER with different decay rates*

| $\alpha$ | $s_d$ | WER |
|---|---|---|
| 0.1 | 50 | 16.1 |
| | 100 | 15.8 |
| | 150 | 16.1 |
| 0.2 | 50 | 16.0 |
| | 100 | 16.1 |

#### *4.3.2. $\lambda$ Perturbation*

Next, we explore if periodically increasing the weight on CE from a static $\lambda = 0.001$ is beneficial. Figure 2 illustrates two different perturbation schemes explored in this paper. Every 10-15M steps, we increase the weight on CE for roughly 1M steps, and decay that increase as we proceed through training. In addition, table 4 shows the WER results for these different perturbation schemes. The table shows that perturbation training hurts WER, and is even worse than no f-smoothing, and hence we did not tune this further.

Table 4: *WER with adverserial $\lambda$ selection*

| $\lambda$ | WER |
|---|---|
| static | 16.0 |
| decayed | 15.8 |
| perturb-1 | 16.1 |
| perturb-2 | 16.2 |

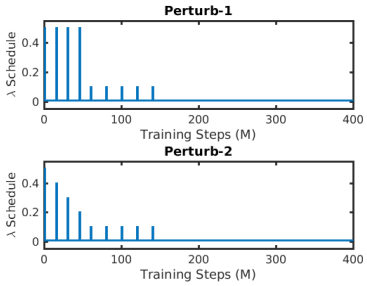


Figure 2: *WER for different λ perturbation schemes*

#### 4.3.3. *Predicting λ using an LSTM*

Finally, we explore the behavior when we use an LSTM to predict $\lambda$ adaptively. For these experiments $\bar{\lambda}$ in Equation 4 is set to be 0.001, the optimal f-smoothing value identified for this task. Table 5 shows the WER for different regularization penalties ($\beta$). The table shows that it is not beneficial to vary $\lambda$ per utterance, and in fact after tuning $\beta$ we can at most match the performance of the system with static $\lambda$ from Section 4.2

This is likely because the regularizer always tries to ensure that the predicted $\lambda$ is in the range of $\bar{\lambda}$. We could start decaying $\bar{\lambda}$, but given that the static version did not yield improvements, we did not pursue this further.

Table 5: *WER with adaptive λ*

| $\lambda$ | $\beta$ | WER |
|---|---|---|
| static | - | 16.0 |
| decayed | - | 15.8 |
| LSTM-predicted | 100 | 16.3 |
| | 1,000 | 16.1 |
| | 10,000 | 16.0 |

### 4.4. When to switch to f-smoothing

Based on the results in Sections 4.2 and 4.3, we chose to decay the $\lambda$ exponentially. Now, we measure the impact of switching from CE to the interpolated objective function at various CE steps, on the WER. These results are shown in Table 6.

Table 6: *WER with decaying λ for different CE steps (M)*

| f-smoothing | CE Steps | SMBR WER | Total Steps |
|---|---|---|---|
| static $\lambda = 0.001$ | 600 | 15.7 | 950 |
| decayed $\lambda$ with $\alpha = 0.1$ | 120 | 16.4 | 520 |
| $d = 0.1, s_d = 100M$, | 240 | 16.0 | 640 |
| $lowerbound = 0.001$ | 360 | 15.8 | 710 |
| | 480 | 15.8 | 930 |

As it can be seen that determining an optimal point of switch can impact the WER significantly, we try to determine it automatically. The change in CE objective function was expected to be an informative signal. Plotting this change every 50M training steps (see Figure 3) we identified that it stabilizes (i.e., changes by less than 0.05) after 300 M steps, which is roughly on par with Table 6. More importantly, by using f-smoothing we can reduce the overall number of training steps by 25% with almost no loss in accuracy. In addition, notice that pulling CE early does not result in a much longer number of SMBR or overall training steps.

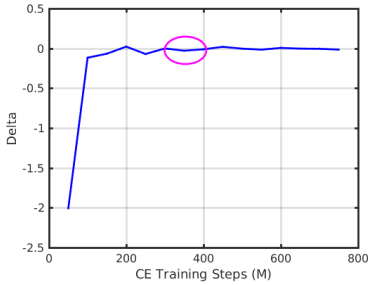


Figure 3: *Change in CE loss every 50m steps*

#### 4.4.1. *Generalization*

Finally we verify if the change in CE loss is a good signal to determine the CE switching point in other LVCSR tasks. We repeated the experiment in Section 4.4 on a different LVCSR task where an acoustic model with a different neural network architecture was used. In this task, we trained a grid LSTM [16] model on a 22M utterance (roughly 18,000 hrs) Voice Search task. Table 7 shows the results after sequence training, and Figure 4 plots the change in CE every 50M steps of training. Using the same thresholding criteria as above, the change in CE loss stabilizes around 300M/350M steps, and the table indicates that this is a reasonable place to start training with f-smoothing. The table also shows that we can achieve a 30% overall reduction in training steps with no loss in accuracy.

Table 7: *WER with decaying λ for different CE steps (M)*

| f-smoothing | CE Steps | SMBR WER | Total Steps |
|---|---|---|---|
| static, $\lambda = 0.01$ | 600 | 14.3 | 800 |
| decayed $\lambda$ with $\alpha = 0.1$ | 250 | 14.7 | 450 |
| $d = 0.1, s_d = 100M$, | 300 | 14.4 | 500 |
| $lowerbound = 0.01$ | 350 | 14.3 | 550 |
| | 400 | 14.3 | 600 |
| | 450 | 14.3 | 650 |

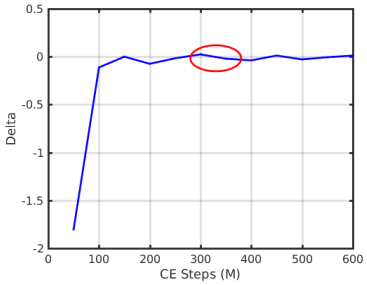


Figure 4: *Change in CE loss every 50ms steps, grid lstm model*

## 5. Conclusions

In this paper, we described a method to reduce the overall number of neural network training steps through f-smoothing. We found that by using a decaying interpolation strategy, we could reduce overall number of steps by 25% with almost no degradation in WER. In addition, we demonstrated that this technique is generalizable across tasks.

# 6. References


[1] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. ICML*, 2009.

[2] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large Scale Distributed Deep Networks," in *Proc. NIPS*, 2012.

[3] G. Pundak and T. N. Sainath, "Lower Frame Rate Neural Network Acoustic Models," in *Proc. Interspeech*, 2016.

[4] B. Kingsbury, "Lattice-Based Optimization of Sequence Classification Criteria for Neural-Network Acoustic Modeling," in *Proc. ICASSP*, 2009.

[5] H. Su, G. Li, D. Yu, and F. Seide, "Error Back Propagation for Sequence Training of Context-Dependent Deep Networks for Conversational Speech Transcription," in *Proc. ICASSP*, 2013.

[6] K. Vesely, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative Training of Deep Neural Networks," in *Proc. Interspeech*, 2013.

[7] G. Heigold, E. McDermott, V. Vanhoucke, A. Senior, and M. Bacchiani, "Asynchronous Stochastic Optimization for Sequence Training of Deep Neural Networks," in *Proc. ICASSP*, 2014.

[8] 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 *Proc. Interspeech*, 2016.

[9] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," in *Proc. Interspeech*, 2014.

[10] T. N. Sainath, B. Kingsbury, A. Mohamed, G. Dahl, G. Saon, H. Soltau, T. Beran, A. Aravkin, and B. Ramabhadran, "Improvements to Deep Convolutional Neural Networks for LVCSR," in *Proc. ASRU*, 2013.

[11] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Senior, and O. Vinyals, "Learning the Speech Front-end with Raw Waveform CLDNNs," in *Proc. Interspeech*, 2015.

[12] D. Yu and L. Deng, "Deep neural network sequence-discriminative training," in *Automatic Speech Recognition: A Deep Learning Approach*, pp. 137–153.

[13] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial Training of Neural Networks," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, January 2016.

[14] B. Li, T. N. Sainath, R. J. Weiss, K. W. Wilson, and M. Bacchiani, "Neural Network Adaptive Beamforming for Robust Multichannel Speech Recognition," in *in Proc. Interspeech*, 2016.

[15] L. Prechelt, "Automatic early stopping using cross validation: quantifying the criteria," *Neural Networks*, vol. 11, no. 4, pp. 761–767, 1998.

[16] T. N. Sainath and B. Li, "Modeling time-frequency patterns with lstm vs. convolutional architectures for lvcsr tasks," in *Proc. Interspeech*, 2016.