Primer on ctc loss - https://distill.pub/2017/ctc/
pseudo -code implementing ctc loss - https://gist.github.com/awni/56369a90d03953e370f3964c826ed4b0

A Novel Re-weighting Method for Connectionist Temporal Classification

Hongzhu Li[+] and Weiqiang Wang [*+]

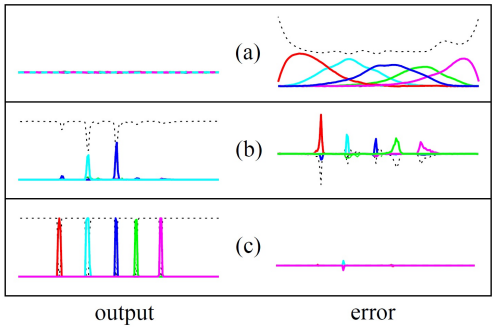
[+]University of Chinese Academy of Sciences, Beijing, China

Abstract

The connectionist temporal classification (CTC) enables end-to-end sequence learning by maximizing the probability of correctly recognizing sequences during training. With an extra *blank* class, the CTC implicitly converts recognizing a sequence into classifying each timestep within the sequence. But the CTC loss is not intuitive for such classification task, so the class imbalance within each sequence, caused by the overwhelming *blank* timesteps, is a knotty problem. In this paper, we define a piece-wise function as the pseudo ground-truth to reinterpret the CTC loss based on sequences as the cross entropy loss based on timesteps. The cross entropy form makes it easy to re-weight the CTC loss. Experiments on text recognition show that the weighted CTC loss solves the class imbalance problem as well as facilitates the convergence, generally leading to better results than the CTC loss. Beside this, the reinterpretation of CTC, as a brand new perspective, may be potentially useful in some other situations.

1. Introduction

The connectionist temporal classification (CTC) (Graves & Gomez, 2006) is a commonly used method in sequence recognition tasks, including speech recognition (Graves & Jaitly, 2014; Miao et al., 2016; Kim et al., 2017), text recognition (Alex et al., 2009; He et al., 2016b; Shi et al., 2017; Borisyuk et al., 2018) and so on. With an extra *blank* class, the output at each timestep in the sequence indicates either a specific label or no label. The outputs over all timesteps consist a sequence of labels and blanks, named as a *path*. A path is transformed into a label sequence by removing the repeated labels then the blanks in it, and different paths can correspond to the same label sequence. The CTC-based training is to maximize the probability of the correct labelling, which is calculated by summing up probabilities of all the corresponding paths.



initially everything is even distributed at start of thr training

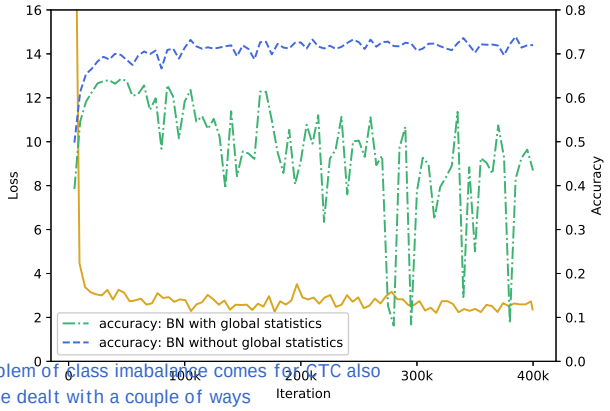
As training proceeds, we have outputs as peak with blank tokens between them

Figure 1. (Graves & Gomez, 2006) Evolution of the network output and CTC error signal during training. Lines with different colors denote different labels, and the dashed line is the *blank* class.

Some previous works try to improve the CTC with regularization or re-weighting/re-sampling heuristics. (Hu et al., 2018) propose a maximum entropy based regularization for CTC (EnCTC) to enhance exploration during training to get models with better generalization. They also propose an entropy-based pruning algorithm (EsCTC) to rule out unreasonable paths. For weakly-supervised action labelling in video, (Huang et al., 2016) introduces the Extended Connectionist Temporal Classification (ECTC) framework to enforce the consistency of all possible paths with frame-to-frame visual similarities. To solve the imbalance problem for sequence recognition, (Feng et al., 2019) modify the traditional CTC by fusing focal loss with it and thus make the model attend to the low-frequent samples at training stage. All these works treat a sequence or a path as an example, and calculate the loss or perform the re-sampling on the basis of sequences or paths.

Different from them, we propose to treat each timestep in a sequence as an individual example, and regard the sequence recognition task as a classification task for each timestep. The classification of each timestep is similar to the classification branch in objection detection (Liu et al., 2016), where the *blank* class corresponds to the background

[*]Corresponding author: wqwang@ucas.ac.cn



the classic problem of class imabalance comes for CTC also
Usually it can be dealt with a couple of ways
1. Some for of weighting (like class weights for classic Random Forests)
2.loss function tweaking or some sort of penalisation for excess class
2. Using sampling (like Random Deletion, etc)

Figure 2. The accuracy degradation phenomenon during CTC-based training. The network configuration is given in Table 1(b). The training set is Synth100 and test set is Train described in Section 3.1. The learning rate is 0.01 and batch size is 32, and other implementation details can be found in Section 3.2.

category and the labels correspond to the objects. In this case, the CTC may suffer from a classic problem in object detection, the imbalance between background and object samples. As shown in Figure 1, the outputs of a trained CTC network tend to form a series of spikes separated by strongly predicted blanks. It means only a few timesteps are label samples, and the rest are all blank samples, which are much more than the former. According to the error signals, the label samples are the hard examples during training, but become less harder as the network converges. By then, the network updating may be overwhelmed by the blanks.

In our experiments, we observe a phenomenon of accuracy degradation that supports this speculation. When a network is trained with the CTC loss and tested on the training set, the recognition accuracy starts to decrease after certain iterations and becomes very unstable. But if the batch normalization is performed within each mini-batch without using global statistics, the accuracy becomes reasonable, as illustrated in Figure 2. It shows that the network updating is unstable so the averaged means/var over the past iterations does not suit the current network weights, which is probably caused by the overwhelming blanks. Therefore, common heuristics for object detection to solve the class imbalance, such as online hard example mining (OHEM) (Shrivastava et al., 2016), focal loss (Lin et al., 2017) and GHM (Li et al., 2019), can be introduced to improve the CTC.

In this paper, we propose a novel method to re-weight the CTC, offering the theory basis and successful experimental experience. The main contributions are summarized as follows: (1) We reinterpret the CTC loss for sequence labelling as the cross entropy loss for classification problem, providing a new perspective to modify the CTC. (2) To deal with the class imbalance, we propose some weighted CTC losses, and demonstrate their effectiveness by comparison experiments on scene text recognition. The proposed weighted CTC has several advantages over the original CTC, including (1) preventing the accuracy degradation phenomenon, (2) alleviating the negative effects caused by imbalanced training data, and (3) facilitating the convergence for models, which means better performance and shorter training time.

2. Method

2.1. Connectionist Temporal Classification

The CTC (Graves & Gomez, 2006) is proposed for labeling sequence data within a single network architecture that doesn't need pre-segmentation and post-processing. The basic idea is to interpret the network outputs as a conditional probability distribution over all possible output label sequences. Given this distribution, an objective function can be derived that directly maximises the probabilities of the correct label sequences.

At each timestep, the network outputs a probability distribution over the label set $L' = L \cup \{blank\}$, where L contains all the labels in the task and the extra $blank$ represents 'no label'. The activation y_k^t is interpreted as the probability of observing label k of L' at time t. Given the length T input sequence $\mathbf{x}$, we get the conditional probability $p(\pi|\mathbf{x})$ of observing a particular $path$ π through the lattice of label observations:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T, \tag{1}$$

where π_t is the label observed at time t along path π, and L'^T is the set of length T paths over L'.

Paths are mapped onto label sequences by an operation $\mathcal{B}$ that simply removes the repeated labels then the blanks in a sequence. For a given label sequence $\mathbf{l} \in L^{\leq T}$, more than one π corresponds to it, e.g. $\mathcal{B}(aa--ab-) = \mathcal{B}(-a-abb) = aab$, where '$-$' denotes the $blank$. We can evaluate the conditional probability of $\mathbf{l}$ as the sum of probabilities of all the corresponding paths:

here aa gets replaces as a if we have a blank between 2 repeated chars then both are kept 'a_ab' -> aab

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \tag{2}$$

The CTC loss function is defined as the negative log probability of correctly labelling the sequence:

$$\text{CTC}(\mathbf{l}, \mathbf{x}) = -\ln p(\mathbf{l}|\mathbf{x}). \tag{3}$$

During training, to backpropagate the gradient through the output layer, we need the derivatives of the loss function versus the outputs $\{a_k^t | t \in [1, T], k \in L'\}$ before the activation

Read more about focal loss https://amaarora.github.io/2020/06/29/FocalLoss.html

function is applied. For the softmax activation function

$$y_k^t = \frac{e^{a_k^t}}{\sum_{k'} e^{a_{k'}^t}}, \tag{4}$$

where k' ranges over L', the derivative with respect to a_k^t is

$$\frac{\partial \mathrm{CTC}(\mathbf{l}, \mathbf{x})}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{l}|\mathbf{x})} \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = k}} p(\pi|\mathbf{x}), \tag{5}$$

where $\sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \pi_t = k} p(\pi|\mathbf{x})$ is the sum of probabilities of all the paths corresponding to $\mathbf{l}$ that go through the label k at time t.

When the network is used for prediction, the predictions over all timesteps are converted into a label sequence. Since the computational complexity grows exponentially with the length of the path, it is not practical to find the most probable label sequence $\hat{\mathbf{l}}$. There are many approximate alternatives, and the **best path decoding** is one of the most commonly used methods. It assumes that the most probable output will correspond to $\hat{\mathbf{l}}$:

beam search ?

$$\begin{aligned} \hat{\mathbf{l}} &\approx \mathcal{B}(\pi^*) \\ \text{where } \pi^* &= \arg\max_{\pi} p(\pi|\mathbf{x}). \end{aligned} \tag{6}$$

It is not guaranteed to find the most probable label sequence, but the solution is good enough in most cases and the computation procedure is trivial.

2.2. Cross Entropy

The cross entropy (CE) is used to estimate the distance between two probability distributions. Given ground-truth $\mathbf{y}'$ and network outputs $\mathbf{y}$, the cross entropy loss is defined as

$$\mathrm{CE}(\mathbf{y}', \mathbf{y}) = -\sum_k y'_k \ln(y_k), \tag{7}$$

where k ranges over all the classes, y_k and y'_k are the model's estimated and ground-truth probabilities for class k respectively. Let $\{a_k\}$ be the model's outputs before the softmax activation function is applied, the loss function derivative with respect to a_k can be found by

$$\frac{\partial \mathrm{CE}(\mathbf{y}', \mathbf{y})}{\partial a_k} = y_k - y'_k. \tag{8}$$

2.3. Focal Loss

The focal loss (Lin et al., 2017) is designed to address the one-stage object detection scenario in which there is an extreme imbalance between foreground and background classes during training. Let $y \in \{\pm 1\}$ specify the ground-truth class for binary classification, and $p \in [0, 1]$ denote the model's estimated probability for the class with label $y = 1$. For notational convenience, define p_t as

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise.} \end{cases} \tag{9}$$

The main idea of focal loss is to reshape the loss function to down-weight easy examples and thus focus training on hard negatives. On the basis of cross entropy loss

$$\mathrm{CE}(p_t) = -\ln(p_t), \tag{10}$$

a modulating factor $(1-p_t)^\gamma$ is added, with tunable focusing parameter $\gamma \geq 0$. The focal loss is defined as

here -1(1-pt) is acting like a modulating/penalizing factor

$$\mathrm{FL}(p_t) = -(1 - p_t)^\gamma \ln(p_t). \tag{11}$$

To address class imbalance, a common method is to introduce a weighting factor $\alpha \in [0, 1]$ for class 1 and $1 - \alpha$ for class -1. For notational convenience, α_t is defined analogously as p_t. The α-balanced variant of the focal loss is defined as

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \ln(p_t). \tag{12}$$

2.4. Cross Entropy Form of CTC

Treating the sequence recognition as the classification for each timestep, we rewrite the CTC loss into the form of cross entropy loss. Given an input sequence $\mathbf{x}$ and its ground-truth label sequence $\mathbf{l}$, the network outputs probability distributions $\mathbf{Y} = \{y_k^t | t \in [1, T], k \in L'\}$ over the T timesteps of the sequence. We define $\mathbf{y}_t = \{y_k^t | k \in L'\}$ as the predicted probability distribution for the sample of timestep t, and assume there is a corresponding ground-truth probability distribution $\mathbf{y'}_t = \{y'^t_k | k \in L'\}$. The cross entropy loss of correctly labelling the sequence should be

$$\mathrm{CTC}(\mathbf{l}, \mathbf{x}) = \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t) = -\sum_t \sum_k y'^t_k \ln(y_k^t). \tag{13}$$

A feasible solution for $\mathbf{y'}_t$ can be found by following the conditions below:

$$\begin{cases} y'^t_k = \frac{1}{p(\mathbf{l}|\mathbf{x})} \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = k}} p(\pi|\mathbf{x}), \\ \frac{\partial y'^t_k}{\partial y^{t'}_{k'}} = 0, \forall t, t' \in [1, T], k, k' \in L'. \end{cases} \tag{14}$$

We can get the derivative of $\sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)$ versus a_k^t

$$\frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial a_k^t} = y_k^t - y'^t_k, \tag{15}$$

which equals to the CTC loss function derivative as in Equation (5).

According to Equation (1) and (2), $p(\pi|\mathbf{x})$ and $p(\mathbf{l}|\mathbf{x})$ are calculated on the basis of $\mathbf{Y}$ and $\mathbf{l}$. It seems unreasonable

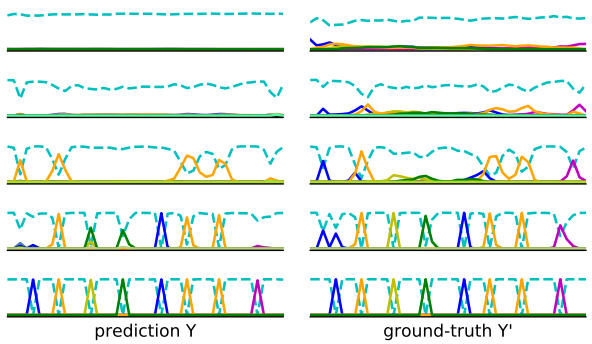


Figure 3. Examples of predicted probability distributions $\mathbf{Y}$ and the corresponding ground-truths $\mathbf{Y}'$. This is the same sequence during different iterations. Lines with different colors denote different labels, and the dashed line is the blank class.

that the derivative of $y_k'^t$ versus $y_{k'}^{t'}$ equals to zero, when $y_k'^t$ depends on $\mathbf{Y}$. We argue that the definition of $\mathbf{y'}_t$ is valid, and elaborate on it as follows.

At first, we define an intermediate variable $\mathbf{M} = \{m_k^t | t \in [1, T], k \in L'\}$, where

$$m_k^t = \frac{1}{p(\mathbf{l}|\mathbf{x})} \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = k}} p(\pi|\mathbf{x}), \tag{16}$$

also denoted by $\mathbf{M} = f(\mathbf{Y}, \mathbf{l})$ for the expression convenience. In this function, the $(\mathbf{Y}, \mathbf{l})$ is a $(T|L'| + |\mathbf{l}|)$-dimensional independent variable, whose value is different for each sequence in each mini-batch. Over the entire training phase, $(\mathbf{Y}, \mathbf{l})$ has finite discrete values. Indexing the values of $(\mathbf{Y}, \mathbf{l})$ by i, we define the ground-truth $\mathbf{Y}'$ as a piecewise function

$$\mathbf{Y}' = g(\mathbf{Y}, \mathbf{l}) = \begin{cases} f(\mathbf{Y}_i, \mathbf{l}_i) & \text{if } \exists i, (\mathbf{Y}, \mathbf{l}) \in U(\mathbf{Y}_i, \mathbf{l}_i) \\ whatever & \text{otherwise,} \end{cases} \tag{17}$$

where $U(\cdot)$ stands for neighborhood. It's easy to know

$$\begin{cases} \mathbf{Y'}_i = g(\mathbf{Y}_i, \mathbf{l}_i) = f(\mathbf{Y}_i, \mathbf{l}_i), \\ \partial \mathbf{Y'}_i / \partial \mathbf{Y}_i = g'(\mathbf{Y}_i, \mathbf{l}_i) = 0. \end{cases} \tag{18}$$

Omitting i and substituting $\mathbf{Y} = \{y_k^t\}$ and $\mathbf{Y}' = \{y_k'^t\}$ into the above equation, we can get Equation (25).

Some examples of $\mathbf{Y}$ and the corresponding $\mathbf{Y}'$ are illustrated in Figure 3 for an intuitive perception.

2.5. Weighted CTC Loss

Given the cross entropy loss function in Equation (26), we can apply weighting methods for classification tasks to improve CTC. We should notice that, compared with the ground-truth in a general classification task, $\mathbf{y'}_t$ does not follow the probability distribution where one of the classes has a probability of 1 and the other classes have a probability of 0, so there is no ground-truth $class$. We adopt the weighting method in two different ways to accommodate this situation. One way is to assign weights for different classes, and it is called *class weighting*. The other way is *sample weighting*, where each sample weight is calculated based on $\mathbf{y}'_t$.

At first, we introduce weighting factors to balance the label and $blank$ samples. The **class-weighted CTC** loss function is defined as

$$\mathrm{CTC}^{\mathrm{cs}}(\mathbf{l}, \mathbf{x}) = -\sum_t \sum_k \alpha_k y_k'^t \ln(y_k^t), \tag{19}$$

where

$$\alpha_k = \begin{cases} 1 - \alpha & \text{if } k = blank \\ \alpha & \text{otherwise} \end{cases} \tag{20}$$

is the weighting factor for class k, and $\alpha \in [0, 1]$ is a tunable parameter. The **sample-weighted CTC** loss function is

$$\mathrm{CTC}^{\mathrm{sp}}(\mathbf{l}, \mathbf{x}) = -\sum_t \alpha_t \sum_k y_k'^t \ln(y_k^t), \tag{21}$$

where α_t is the weighting factor for sample t defined as

$$\alpha_t = \alpha(1 - y_{blank}'^t) + (1 - \alpha) y_{blank}'^t. \tag{22}$$

It is easy to know that when α is taken as 0.5, the above two weighted CTC losses are equivalent to the CTC loss.

We introduce focal loss to CTC, naming it as *connectionist temporal focal loss (CTFL)*, to focus the training process on hard samples. Extending focal loss from binary classification to multi-class case is straightforward. Defining p_t as the estimated probability for the ground-truth class, $(1 - p_t)^\gamma$ is used to down-weight easy samples, where $\gamma \geq 0$ is a tunable focusing parameter. But as analyzed before, there is no ground-truth class for $\mathbf{y'}_t$. In this case, we extend the sample weights of focal loss to class weights form $|y_k^t - y_k'^t|^\gamma$, where $|y_k^t - y_k'^t|$ denotes the distance between the estimated and ground-truth probabilities for class k. For class weighting, we use the distances as the class weights of each sample, and define the **classes-weighted CTFL** as

$$\mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l}, \mathbf{x}) = -\sum_t \sum_k |y_k^t - y_k'^t|^\gamma y_k'^t \ln(y_k^t). \tag{23}$$

As with sample weighting, each sample weight is calculated by summing the distances over all classes. The **smaple-weighted CTFL** is given as

$$\mathrm{CTFL}^{\mathrm{sp}}(\mathbf{l}, \mathbf{x}) = -\sum_t (\sum_k |y_k^t - y_k'^t|^\gamma)(\sum_k y_k'^t \ln(y_k^t)). \tag{24}$$

It is easy to notice that when the value of γ is 0, CTFL degenerates into the CTC loss.

See the appendix for the loss derivatives and formula derivation processes.

3. Experiments

To evaluate the effects of the weighted CTC losses, we compare them with the CTC loss according to the convergence process and recognition performance of the models. For all the experiments, the accuracy refers to sequence accuracy, i.e. the percentage of testing images correctly recognized. Although different losses are adopted for training, the output is always the CTC loss, for it is an indicator of the probability of correctly recognizing a sequence according to Equation (3).

3.1. Datasets

For all the following experiments, we use the synthetic dataset released by (Jaderberg et al., 2014) as the training data. The dataset consists of 8 million word images and their corresponding ground-truth words. All the images are generated by a synthetic data engine using a 90k word dictionary, and are of different sizes. For training efficiency, we construct a training set **Synth** consisting of 32×256 images: At first, all word images are scaled to have height 32 without changing their aspect-ratios. If the scaled width is larger than 256, we continue to scale the image to 32×256. If there's enough room for the next scaled image, we append it to this image after 20 columns of zeros. Otherwise, we pad the scaled image to width 256 with zeros. Besides, we construct the other training set **Synth100**, which is more balanced between different classes. It is the same as Synth but containing 100 times extra copies of the images containing digits. The character number of each class in Synth and Synth100 is displayed in Figure 4.

There are four popular benchmarks for scene text recognition used for model performance evaluation, namely IIIT5k-word (IIIT5k), Street View Text (SVT), ICDAR 2003 (IC03) and ICDAR 2013 (IC13). **IIIT5k** (Mishra et al., 2012) contains 3,000 cropped word images collected from the Internet. **SVT** (Kai et al., 2012) contains 647 word images cropped from 249 street-view images that are collected from Google Street View. **IC03** (Lucas et al., 2003) contains 251 scene images, we discard words that either contain non-alphanumeric characters or have less than three characters, and get 860 cropped word images. **IC13** (Karatzas et al., 2013) contains 1,095 word images in total, we discard words that contain non-alphanumeric characters, and get 1,015 word images with ground-truths. In addition, we construct a subset **Train** with the first 64,000 images taken from the training set to evaluate the model performance on the training data.

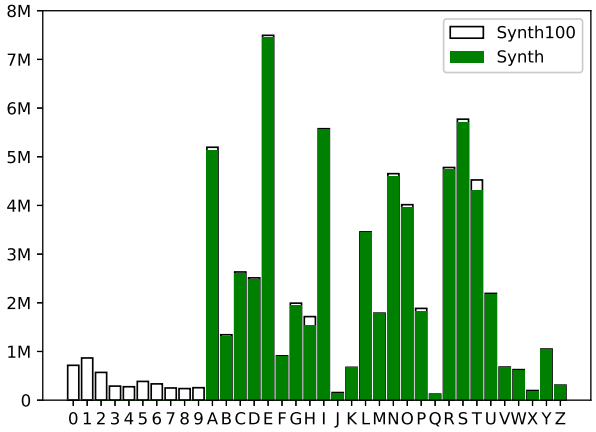


Figure 4. The number of characters for each class in Synth and Synth100. The word images containing digits in Synth100 are 100 times more than those in Synth.

3.2. Implementation Details

There are two networks used in our experiments, one is CRNN (Shi et al., 2017), the other is a CNN replacing the BLSTM layers in CRNN with residual blocks (He et al., 2016a). The network configurations are summarized in Table 1.

Unless otherwise stated, we use the CNN as the default network and Synth100 as the default training set in our experiments. We implement the network architecture within the Caffe (Jia et al., 2014) framework, with custom implementation for the loss layer. Networks are trained with stochastic gradient descent (SGD). The decay rate of weights is 0.0005, and the momentum is 0.9. The initial learning rate is 0.01, and it is decreased by a factor of 0.1 after every fix number of iterations denoted as learning rate step. Three different training strategies are used in our experiments: **bs32-400k**: batch size = 32, learning rate step = 100,000, max iterations = 400,000; **bs32-800k**: batch size = 32, learning rate step = 200,000, max iterations = 800,000; **bs256-40k**: batch size = 256, learning rate step = 20,000, max iterations = 40,000.

For all the experiments, we get the recognition results by the lexicon-free best path decoding (Graves & Gomez, 2006).

3.3. Complexity Analysis

We propose four weighted forms of the CTC loss, and compare the algorithm complexities of them to CTC. According to their loss function derivatives, the gradient updating procedures are performed based on y_k^t and $y_k'^t$, which

Table 1. Network configurations. 'Conv' is short for convolutional layer, and 'MP' is short for max pooling layer. 'c' stands for channels, which denotes the number of feature maps for convolutional layer, the number of hidden units for BLSTM layer, and the bottleneck channels for residual unit. 'k', 's', 'p' stand for kernel, stride and padding sizes respectively. 'bn' stands for batch normalization, 'softmax' stands for softmax activation function. The residual unit used here is the full pre-activation version proposed in (He et al., 2016a).

(a) CRNN

TYPE	CONFIGURATION
INPUT	$1 \times W \times 32$
CONV	C64,K3X3,P1X1
MP	K2X2,S2X2
CONV	C128,K3X3,P1X1
MP	K2X2,S2X2
CONV	C256,K3X3,P1X1,BN
CONV	C256,K3X3,P1X1
MP	K2X2,S1X2,P1X0
CONV	C512,K3X3,P1X1,BN
CONV	C512,K3X3,P1X1
MP	K2X2,S1X2,P1X0
CONV	C512,K2X2,BN
BLSTM	C256
BLSTM	C256
OUTPUT	C37,SOFTMAX
LOSS	-

(b) CNN

TYPE	CONFIGURATION
INPUT	$1 \times W \times 32$
CONV	C64,K3X3,P1X1
MP	K2X2,S2X2
CONV	C128,K3X3,P1X1
MP	K2X2,S2X2
CONV	C256,K3X3,P1X1,BN
CONV	C256,K3X3,P1X1
MP	K2X2,S1X2,P1X0
CONV	C512,K3X3,P1X1,BN
CONV	C512,K3X3,P1X1
MP	K2X2,S1X2,P1X0
CONV	C512,K2X2
RESUNIT	C128,K5X1,P2X0
RESUNIT	C128,K5X1,P2X0,BN
OUTPUT	C37,SOFTMAX
LOSS	-

are also calculated for the CTC loss. First, a softmax activation is applied to get the normalized network outputs $\{y_k^t\}$, whose time complexity is $O(T|L'|)$ for the cpu implementation and $O(|L'|)$ for the parallel gpu implementation. Then a dynamic-programming algorithm similar to the forward-backward algorithm for HMMs (Rabiner, 1993) is performed to calculate $\{y_k'^t\}$, whose time complexity is $O(T|\mathbf{l}|)$ for both cpu and gpu implementations. For the CTC loss, there is the final subtraction, whose time complexity is $O(T|L'|)$ for cpu and $O(1)$ for gpu. Meanwhile, the weighted losses need additional calculations of the weights. We obtain the time complexity of CTC by summing up the above terms, and list the time complexity of the additional calculation for each weighted loss in Table 2. It's obvious that the additional calculation doesn't change the time complexity, so the amount of additional calculation in the weighted loss is acceptable. This is also validated by experiments, where the changes of training time are negligible.

The space complexity of CTC is $O(T|\mathbf{l}|)$. Since the algorithm makes the most of the original space and keep down the additional space complexity of a weighted CTC to $O(T)$, which doesn't change the original space complexity.

In one word, the proposed weighted CTC losses have the same time and space complexity as the CTC loss.

Table 2. The additional time complexity and training time for each weighted loss, compared with the CTC loss. Trn. Time denotes training time spent for 100,000 iterations.

METHOD	COMPLEXITY-CPU	COMPLEXITY-GPU	TRN. TIME
CTC	$O(T\|\mathbf{l}\|) + O(T\|L'\|)$	$O(T\|\mathbf{l}\|) + O(\|L'\|)$	197MIN
CTC^{cs}	$O(T\|L'\|)$	$O(\|L'\|)$	199MIN
CTC^{sp}	$O(T)$	$O(1)$	195MIN
CTFL^{cs}	$O(T\|L'\|)$	$O(\|L'\|)$	200MIN
CTFL^{sp}	$O(T\|L'\|)$	$O(\|L'\|)$	194MIN

3.4. Overall Comparison

We train a group of models with each weighted CTC loss under variable hyper-parameters. The training strategy is bs32-400k. The convergence processes of models are shown in Figure 5, and the recognition performances are illustrated in Figure 6. Note that when α is taken as 0.5 and γ is taken as 0, a weighted CTC loss becomes the CTC loss.

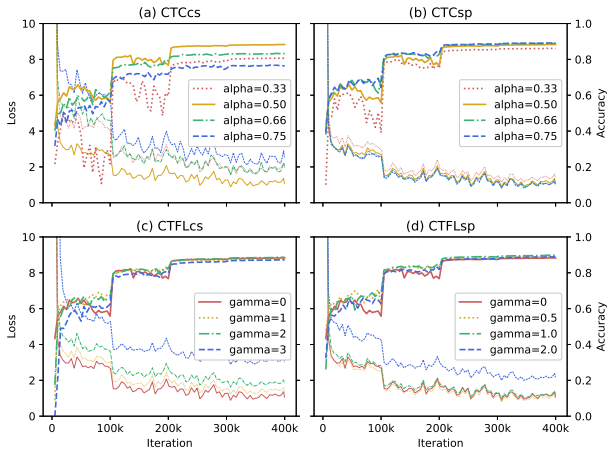


Figure 5. The convergence processes of models trained with the weighted CTC losses under variable hyper-parameters. The thinner lines indicate the CTC loss, and the wider lines denote the recognition accuracies on the training set.

The parameter α in CTC^{cs} and CTC^{sp} is used to adjust the ratio of influences coming from label and $blank$ samples. Figure 5(a) shows the effect of CTC^{cs} is not ideal. Figure 5(b) suggests that the accuracy degradation is caused by excess $blank$ samples, which is consistent with our class imbalance speculation, for the degradation is more obvious under a lower α. CTC^{sp} can prevent the accuracy degradation by focusing more attention on label samples, but it brings no obvious improvement for the recognition performance, as shown in Figure 6(b). Based on our experience, CTC^{sp} benefits the model performance in some situations, but the improvements are minor compared with CTFL. So

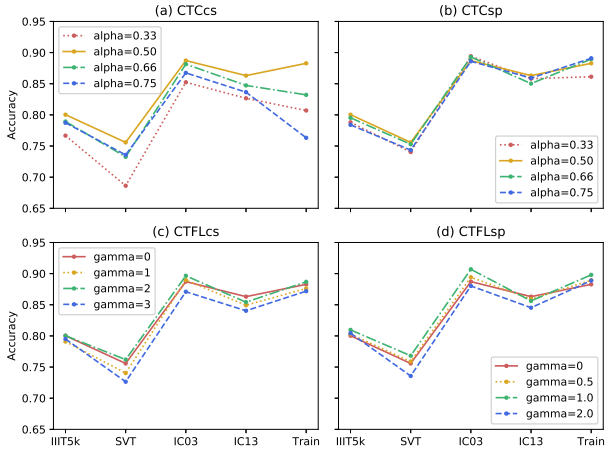


Figure 6. Recognition accuracies on four test sets the training set. Each subfigure corresponds the performances of models trained with a weighted CTC losses under variable hyper-parameters.

we ignore the α-weighting for the rest of this paper, and focus on discussing the effects of CTFL.

According to Figure 5(c) and 5(d), CTFL can prevent the accuracy degradation, as well as improve the recognition performance on the training set. CTFL^{sp} performs slightly better than CTFL^{cs} on recognition accuracy. Moreover, compared with CTFL^{cs}, CTFL^{sp} has no negative impact on the CTC loss, i.e. the probability of correctly recognizing a sequence. It means that CTFL^{sp} will not damage the recognition performance when the decoding method is based on the probability. Figure 6(c) and 6(d) suggest $\gamma = 2$ for CTFL^{cs} and $\gamma = 1$ for CTFL^{sp}. These values are adopted for the rest of our experiments. However it is possible that the optimal values of the parameters are different for different tasks.

3.5. Deal with Class Imbalance

To investigate the effectiveness of CTFL for imbalanced classes, we train a set of models with the CTC loss and CTFL on Synth and Synth100 respectively. The training strategy is bs32-400k. The recognition performances on four test sets are presented in Table 3. The results are also illustrated in Figure 7 to provide a direct perception.

Comparing between the models trained with the CTC loss, there are large margins between the accuracies on IIIT5k, IC03 and IC13 of models trained on Synth and Synth100. Each accuracy margin is consistent with the digits ratio of the corresponding test set according to Table 3. Therefore, we speculate that the model trained with CTC on Synth cannot correctly recognize digits, and it is cased by the severe class imbalance in Synth as shown in Figure 4.

Table 3. Recognition accuracies (%) on four English scene text datasets. Digits Ratio (%) indicates the percentage of word examples containing digits in the dataset. Synth and Synth100 in the first column denote the training sets for the network, CTC and CTFL in the second column denote the loss functions used during training.

TRN. SET	METHOD	IIIT5K	SVT	IC03	IC13
		9.2	0	3.8	8.4
SYNTH	CTC	72.9	**75.6**	85.6	78.7
	CTFL^{cs}	78.6	74.5	89.1	**84.7**
	CTFL^{sp}	**80.4**	**75.6**	**90.3**	**84.7**
SYNTH100	CTC	80.0	75.6	88.7	**86.3**
	CTFL^{cs}	80.0	76.2	89.7	85.4
	CTFL^{sp}	**81.0**	**76.8**	**90.7**	85.6

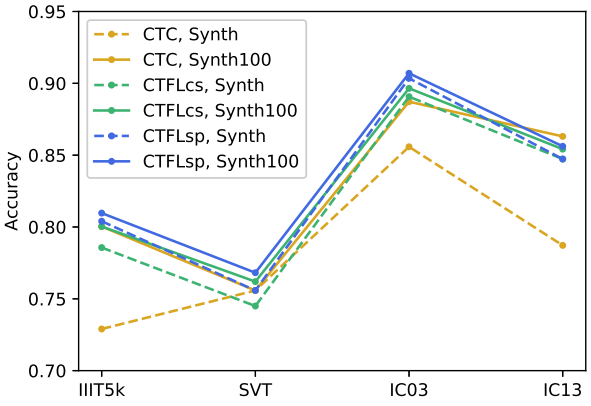


Figure 7. The illustration of recognition performances in Table 3.

The CTFL focus the network on hard samples during training, thus improves the network's recognition ability for disadvantaged classes, i.e. digits in this case. Comparing between the models trained on Synth, the models trained with CTFL achieve much better performances than CTC, and their performances are nearly as good as models trained on the balanced training set Synth100.

3.6. Facilitate Convergence

The convergence process of a model is influenced by the training settings, including batch size, learning rate, network architecture, etc. To evaluate the effect of CTFL on convergence for different training settings, in addition to the experiments above, we conduct another two groups of comparison experiments. The training settings and model performances for the three groups of experiments are shown in Table 4.

Comparing models trained with the CTC loss over different

Table 4. The performances of models trained with different loss functions under three groups of training settings. Trn. St. is short for training settings, which include the network configuration and the training strategy. Note that CTC(800k) denotes the model is trained with the CTC loss under strategy bs32-800k.

Trn. St.	Method	IIIT5k	SVT	IC03	IC13	Train
CNN bs32-400k	CTC	80.0	75.6	88.7	**86.3**	88.3
	CTFL^{cs}	80.0	76.2	89.7	85.4	88.7
	CTFL^{sp}	**81.0**	**76.8**	**90.7**	85.6	**89.8**
CNN bs256-40k	CTC	77.2	71.4	86.9	84.1	85.5
	CTFL^{cs}	78.7	72.6	87.7	84.5	86.6
	CTFL^{sp}	**78.8**	**75.1**	**88.6**	**85.2**	**87.9**
CRNN bs32-400k	CTC	76.4	69.1	87.2	82.8	84.0
	CTC(800k)	77.1	72.3	87.3	82.8	86.3
	CTFL^{cs}	76.5	**74.8**	89.7	**85.1**	88.3
	CTFL^{sp}	**78.3**	73.7	**90.9**	84.8	**89.5**

training settings, the CNN and strategy bs32-400k lead to the best model performance. The convergence process is illustrated in Figure 5 with $\gamma = 0$. Accuracy degradation appears early during training, but as the learning rate decreases, the degradation gradually disappears and does not affect the final model performance. In this situation, the CTFL prevents the accuracy degradation and stabilizes the convergence process. But its effect of facilitating the convergence and improving the performance is not so obvious.

Compared with training strategy bs32-400k, the strategy bs256-40k leads to under-fitting models. According to Table 4, the accuracies of models trained by the CTC loss drop by about 2-4% due to the different training strategy. The convergence processes of this strategy are illustrated in Figure 8. Compared with the CTC-based training, the CTFL greatly facilitates the convergence and leads to a much better model performance. We think that the advantages of CTFL over CTC is more obvious when the CTC-based training suffers from under-fitting. This opinion is also supported by the third group of experiments, where the under-fitting is caused by the network structure, for recurrent structures are usually difficult to train. As shown in Figure 9, the models trained with CTFL easily outperform the model trained with CTC in convergence, even for the model trained for double the time.

On one hand, it usually takes a lot of effort to find the proper training settings. The CTFL facilitates the convergence, thus ensures a relatively reasonable model performance for various training settings. On the other hand, under the same training settings, models trained with CTFL always achieve better or similar performances within less training iterations.

3.7. Comparison With CRNN

Convolutional recurrent neural network (CRNN) (Shi et al., 2017) is one of the state-of-the-art approaches in CTC-based

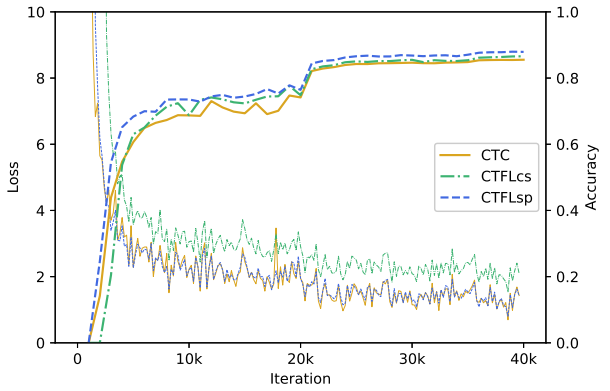


Figure 8. The convergence processes of models of CNN trained under strategy bs256-40k.

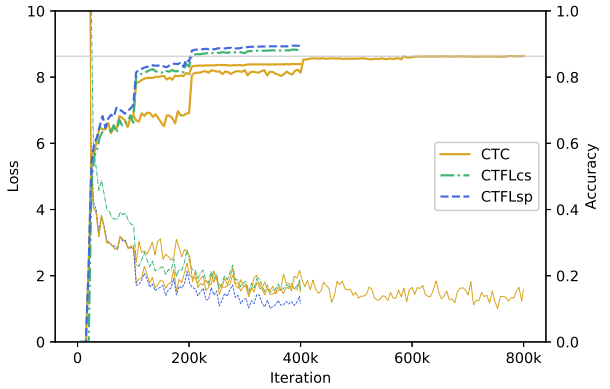


Figure 9. The convergence processes of models of CRNN trained under strategy bs32-400k or bs32-800k.

text recognition, and it is adopted in our experiments. The results of CRNN in Table 4 fall behind the reported results in (Shi et al., 2017), because the training sets and test sets are different. Although the training and test sets are built from the same datasets, the details of the building process can be different, which affects the experimental results. As shown in Table 5, we test the trained model released by (Shi et al., 2017) (downloaded from their code webpage, and supposed to have similar performance as the reported results) on our test sets, and get somehow inferior results.

To ensure a fair comparison, we compare results obtained on our test sets. Besides, we adopt the same training set and rescaling strategy, despite slightly different training settings, that are described in (Shi et al., 2017). According to Table 5, the CTC model gets similar results as the released CRNN model, and the model trained with CTFL achieves better results.

Table 5. The results of models with CRNN architecture. Our models are trained with SGD, batch-size 64, for 600k iterations on rescaled 100 × 32 images then 200k iterations on variable-size images.

	METHOD	IIIT5K	SVT	IC03	IC13
REPORTED	(SHI ET AL., 2017)	81.2	82.7	91.9	89.6
RESULTS ON OUR TEST SETS	(SHI ET AL., 2017)	80.3	81.6	90.0	86.2
	CTC	80.2	79.9	90.9	86.6
	CTFL^{sp}	**82.0**	**81.8**	**91.5**	**88.6**

4. Conclusion

In this paper, we provide a new perspective to modify the CTC method. The basic idea is to treat, instead of a sequence, each timestep in the sequence as a sample. Accordingly, we reinterpret the CTC loss for sequences into the cross entropy loss for timesteps through a pseudo ground-truth. The cross entropy form makes it possible for the CTC loss to cooperate with re-weighting strategy.

We introduce label/blank weighting and focal loss to CTC and get four weighted CTC losses. The experiments show that the smaple-weighted CTFL generally performs best among them. The proposed losses are proven to have the same complexity as CTC and the following benefits: eliminating the accuracy degradation, better performance when trained on imbalanced data, and contributing to faster and better convergence in some cases.

Apart from the re-weighting method in this paper, the reinterpretation of CTC may be potentially useful in other contexts that is worthy of exploration in the future.

References

Alex, G., Marcus, L., Santiago, F., Roman, B., Horst, B., and Jrgen, S. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 31(5):855–868, 2009.

Borisyuk, F., Gordo, A., and Sivakumar, V. Rosetta: Large scale system for text detection and recognition in images. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 71–79, 07 2018.

Feng, X., Yao, H., and Zhang, S. Focal CTC Loss for Chinese Optical Character Recognition on Unbalanced Datasets. *Complexity*, 2019:11, 2019. doi: 10.1155/2019/9345861.

Graves, A. and Gomez, F. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning*, pp. 369–376, 2006.

Graves, A. and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1764–1772, Bejing, China, 22–24 Jun 2014. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645, 2016a.

He, P., Huang, W., Qiao, Y., Chen, C. L., and Tang, X. Reading scene text in deep convolutional sequences. In *Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3501–3508, 2016b.

Hu, L., Sheng, J., and Changshui, Z. Connectionist temporal classification with maximum entropy regularization. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Huang, D.-A., Fei-Fei, L., and Niebles, J. C. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision (ECCV)*, pp. 137–153. Springer, 2016.

Jaderberg, M., Simonyan, K., Vedaldi, A., and Zisserman, A. Synthetic data and artificial neural networks for natural scene text recognition. *Eprint Arxiv*, 2014.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., and Long, J. Caffe: Convolutional architecture for fast feature embedding. *Eprint Arxiv*, pp. 675–678, 2014.

Kai, W., Babenko, B., and Belongie, S. End-to-end scene text recognition. In *IEEE International Conference on Computer Vision*, 2012.

Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Bigorda, L. G. I., Mestre, S. R., Mas, J., Mota, D. F., Almazan, J. A., and Heras, L. P. D. L. Icdar 2013 robust reading competition. In *International Conference on Document Analysis & Recognition*, 2013.

Kim, S., Hori, T., and Watanabe, S. 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)*, pp. 4835–4839, March 2017.

Li, B., Liu, Y., and Wang, X. Gradient Harmonized Single-stage Detector. *AAAI*, 2019.

Lin, T. Y., Goyal, P., Girshick, R., He, K., and Dollar, P. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, PP (99):2999–3007, 2017.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pp. 21–37, 2016.

Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., and Young, R. Icdar 2003 robust reading competitions. *Proc of the Icdar*, 7(2-3):105–122, 2003.

Miao, Y., Gowayyed, M., and Metze, F. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *Automatic Speech Recognition & Understanding*, 2016.

Mishra, A., Alahari, K., and V. Jawahar, C. Scene text recognition using higher order language priors. 09 2012.

Rabiner, L. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of The IEEE - PIEEE*, 77, 01 1993.

Shi, B., Bai, X., and Yao, C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(11):2298, 2017.

Shrivastava, A., Gupta, A., and Girshick, R. Training region-based object detectors with online hard example mining. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 761–769, 2016.

A. Cross Entropy Form of CTC

In the paper, we define a pseudo ground-truth $\mathbf{y'}_t = \{y'^t_k | k \in L'\}$, where

$$\begin{cases} y'^t_k = \frac{1}{p(\mathbf{l}|\mathbf{x})} \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \atop \pi_t = k} p(\pi|\mathbf{x}), \\ \frac{\partial y'^t_k}{\partial y^{t'}_{k'}} = 0, \forall t, t' \in [1, T], k, k' \in L', \end{cases} \tag{25}$$

to reinterpret the CTC loss as the sum of cross entropy losses. To this end, we need to prove that $\mathbf{y'}_t$ is a feasible solution of

$$\mathrm{CTC}(\mathbf{l}, \mathbf{x}) = \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t) = -\sum_t \sum_k y'^t_k \ln(y^t_k). \tag{26}$$

It means given the definition of $\mathbf{y'}_t$, Equ. (26) holds.

For an ideal situation, the CTC and the cross-entropy loss are both 0, equality holds. Therefore, we only need to prove that the derivatives of the them are equivalent.

Having $\{a^t_k | t \in [1, T], k \in L'\}$ denote the unnormalized network outputs, we normalize them with the softmax activation,

$$y^t_k = softmax(a^t_k) = \frac{e^{a^t_k}}{\sum_{k'} e^{a^t_{k'}}}. \tag{27}$$

It's easy to know

$$\frac{\partial y^{t'}_{k'}}{\partial a^t_k} = \begin{cases} 0 & \text{if } t' \neq t \\ y^t_k(1 - y^t_k) & \text{if } t' = t, k' = k \\ -y^t_k y^t_{k'} & \text{if } t' = t, k' \neq k. \end{cases} \tag{28}$$

The derivation of cross entropy formatted CTC versus y^t_k can be calculated as

$$\begin{aligned} &\frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial y^t_k} \\ =& -\frac{\partial \sum_{t',k'} y'^{t'}_{k'} \ln(y^{t'}_{k'})}{\partial y^t_k} \\ =& -\frac{\partial y'^t_k \ln(y^t_k)}{\partial y^t_k} \\ =& -y'^t_k \frac{\partial \ln(y^t_k)}{\partial y^t_k} + \ln(y^t_k) \frac{\partial y'^t_k}{\partial y^t_k} \\ =& -\frac{y'^t_k}{y^t_k}, \end{aligned} \tag{29}$$

and its derivation with respect to a^t_k can be calculated as

$$\begin{aligned} &\frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial a^t_k} \\ =& \sum_{t',k'} \frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial y^{t'}_{k'}} \frac{\partial y^{t'}_{k'}}{\partial a^t_k} \\ =& \sum_{k'} \frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial y^t_{k'}} \frac{\partial y^t_{k'}}{\partial a^t_k} \\ =& \frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial y^t_k} \frac{\partial y^t_k}{\partial a^t_k} + \sum_{k' \neq k} \frac{\partial \sum_t \mathrm{CE}(\mathbf{y'}_t, \mathbf{y}_t)}{\partial y^t_{k'}} \frac{\partial y^t_{k'}}{\partial a^t_k} \\ =& (-\frac{y'^t_k}{y^t_k}) y^t_k (1 - y^t_k) + \sum_{k' \neq k} (-\frac{y'^t_{k'}}{y^t_{k'}})(-y^t_k y^t_{k'}) \\ =& y'^t_k y^t_k - y'^t_k + \sum_{k' \neq k} y'^t_{k'} y^t_k \\ =& y^t_k \sum_{k'} y'^t_{k'} - y'^t_k \\ =& y^t_k \frac{1}{p(\mathbf{l}|\mathbf{x})} \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) - y'^t_k \\ =& y^t_k - y'^t_k. \end{aligned} \tag{30}$$

It is equal to the derivative of CTC given in the paper, so Equ.(26) holds.

B. Derivation Process of $\mathrm{CTC}^{\mathrm{cs}}$

The **class-weighted CTC** loss function is

$$\mathrm{CTC}^{\mathrm{cs}}(\mathbf{l}, \mathbf{x}) = -\sum_t \sum_k \alpha_k y'^t_k \ln(y^t_k), \tag{31}$$

where

$$\alpha_k = \begin{cases} 1 - \alpha & \text{if } k = blank \\ \alpha & \text{otherwise.} \end{cases} \tag{32}$$

The derivation of CTC^{cs} versus y^t_k can be calculated as

$$\begin{aligned} &\frac{\partial \sum_t \mathrm{CTC}^{\mathrm{cs}}(\mathbf{l}, \mathbf{x})}{\partial y^t_k} \\ =& -\frac{\partial \sum_{t',k'} \alpha_{k'} y'^{t'}_{k'} \ln(y^{t'}_{k'})}{\partial y^t_k} \\ =& -\alpha_k \frac{y'^t_k}{y^t_k}, \end{aligned} \tag{33}$$

and its derivation with respect to a_k^t can be calculated as

$$\begin{aligned}
&\frac{\partial \mathrm{CTC}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial a_k^t} \\
=&\frac{\partial \sum_t \mathrm{CTC}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial y_k^t}\frac{\partial y_k^t}{\partial a_k^t}+\sum_{k'\neq k}\frac{\partial \sum_t \mathrm{CTC}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial y_{k'}^t}\frac{\partial y_{k'}^t}{\partial a_k^t} \\
=&(-\alpha_k\frac{y_k'^t}{y_k^t})y_k^t(1-y_k^t)+\sum_{k'\neq k}(-\alpha_{k'}\frac{y_{k'}'^t}{y_{k'}^t})(-y_k^t y_{k'}^t) \\
=&\alpha_k y_k'^t y_k^t-\alpha_k y_k'^t+\sum_{k'\neq k}\alpha_{k'}y_{k'}'^t y_k^t \\
=&y_k^t\sum_{k'}\alpha_{k'}y_{k'}'^t-\alpha_k y_k'^t.
\end{aligned} \tag{34}$$

C. Derivation Process of $\mathrm{CTC}^{\mathrm{sp}}$

The **sample-weighted CTC** loss function is

$$\mathrm{CTC}^{\mathrm{sp}}(\mathbf{l},\mathbf{x})=-\sum_t\alpha_t\sum_k y_k'^t\ln(y_k^t), \tag{35}$$

where

$$\alpha_t=\alpha(1-y_{blank}'^t)+(1-\alpha)y_{blank}'^t. \tag{36}$$

Since the derivation of α_t versus y_k^t can be calculated as

$$\frac{\partial\alpha_{t'}}{\partial y_k^t}=\frac{\partial\alpha(1-y_{blank}'^{t'})+(1-\alpha)y_{blank}'^{t'}}{\partial y_k^t}=0, \tag{37}$$

and its derivation versus a_k^t is

$$\frac{\partial\alpha_{t'}}{\partial a_k^t}=0. \tag{38}$$

The derivation of CTC^{sp} with respect to a_k^t is

$$\begin{aligned}
&\frac{\partial \mathrm{CTC}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial a_k^t} \\
=&-\frac{\partial\sum_{t'}\alpha_{t'}\sum_{k'}y_{k'}'^{t'}\ln(y_{k'}^{t'})}{\partial a_k^t} \\
=&-\frac{\partial y_k'^t\ln(y_k^t)}{\partial a_k^t}\alpha_t \\
=&(y_k^t-y_k'^t)\alpha_t.
\end{aligned} \tag{39}$$

D. Derivation Process of $\mathrm{CTFL}^{\mathrm{cs}}$

The **classes-weighted CTFL** is defined as

$$\mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})=-\sum_t\sum_k|y_k^t-y_k'^t|^\gamma y_k'^t\ln(y_k^t), \tag{40}$$

and its derivative versus y_k^t is

$$\begin{aligned}
&\frac{\partial \mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial y_k^t} \\
=&-\frac{\partial|y_k^t-y_k'^t|^\gamma y_k'^t\ln(y_k^t)}{\partial y_k^t} \\
=&-y_k'^t|y_k^t-y_k'^t|^{\gamma-1}(\mathrm{sign}(y_k^t-y_k'^t)\gamma\ln(y_k^t)+\frac{|y_k^t-y_k'^t|}{y_k^t}),
\end{aligned} \tag{41}$$

and its derivative versus a_k^t is

$$\begin{aligned}
&\frac{\partial \mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial a_k^t} \\
=&\frac{\partial \sum_t \mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial y_k^t}\frac{\partial y_k^t}{\partial a_k^t}+\sum_{k'\neq k}\frac{\partial \sum_t \mathrm{CTFL}^{\mathrm{cs}}(\mathbf{l},\mathbf{x})}{\partial y_{k'}^t}\frac{\partial y_{k'}^t}{\partial a_k^t} \\
=&(-y_k'^t|y_k^t-y_k'^t|^{\gamma-1}(\mathrm{sign}(y_k^t-y_k'^t)\gamma\ln(y_k^t)+\frac{|y_k^t-y_k'^t|}{y_k^t}))y_k^t(1-y_k^t) \\
+&\sum_{k'\neq k}(-y_{k'}'^t|y_{k'}^t-y_{k'}'^t|^{\gamma-1}(\mathrm{sign}(y_{k'}^t-y_{k'}'^t)\gamma\ln(y_{k'}^t)+\frac{|y_{k'}^t-y_{k'}'^t|}{y_{k'}^t}))(-y_k^t y_{k'}^t) \\
=&y_k^t\sum_{k'}(y_{k'}'^t|y_{k'}^t-y_{k'}'^t|^{\gamma-1}(\mathrm{sign}(y_{k'}^t-y_{k'}'^t)\gamma y_{k'}^t\ln(y_{k'}^t)+|y_{k'}^t-y_{k'}'^t|) \\
&-y_k'^t|y_k^t-y_k'^t|^{\gamma-1}(\mathrm{sign}(y_k^t-y_k'^t)\gamma y_k^t\ln(y_k^t)+|y_k^t-y_k'^t|) \\
=&y_k^t\sum_{k'}z_{k'}^t-z_k^t,
\end{aligned} \tag{42}$$

where

$$z_k^t=y_k'^t|y_k^t-y_k'^t|^{\gamma-1}(\mathrm{sign}(y_k^t-y_k'^t)\gamma y_k^t\ln(y_k^t)+|y_k^t-y_k'^t|). \tag{43}$$

E. Derivation Process of $\mathrm{CTFL}^{\mathrm{sp}}$

The **smaple-weighted CTFL** is given as

$$\mathrm{CTFL}^{\mathrm{sp}}(\mathbf{l},\mathbf{x})=-\sum_t(\sum_k|y_k^t-y_k'^t|^\gamma)(\sum_k y_k'^t\ln(y_k^t)), \tag{44}$$

and we find its derivative with respect to a_k^t as

$$\begin{aligned}
\frac{\partial \mathrm{CTFL}^{\mathrm{sp}}(\mathbf{l},\mathbf{x})}{\partial a_k^t}=&(y_k^t-y_k'^t)\sum_{k'}|y_{k'}^t-y_{k'}'^t|^\gamma+ \\
&(y_k^t\sum_{k'}z_{k'}^t-z_k^t)\sum_{k'}y_{k'}'^t\ln y_{k'}^t,
\end{aligned} \tag{45}$$

$$z_k^t=\mathrm{sign}(y_k^t-y_k'^t)\gamma y_k^t|y_k^t-y_k'^t|^{\gamma-1}. \tag{46}$$

To simplify the calculation process, we assume that the partial derivative of the sample weight $\sum_k|y_k^t-y_k'^t|^\gamma$ versus y_k^t is negligible. In this case, there is

$$\frac{\partial\sum_k|y_k^t-y_k'^t|^\gamma}{\partial a_k^t}\approx 0, \tag{47}$$

and the simplified derivative of $\mathrm{CTFL}^{\mathrm{sp}}$ can be calculated

as

$$
\begin{aligned}
&\frac{\partial \mathrm{CTFL}^{\mathrm{sp}}(\mathbf{l}, \mathbf{x})}{\partial a_k^t} \\
=&-\frac{\partial \sum_t (\sum_k |y_k^t - y_k'^t|^\gamma)(\sum_k y_k'^t \ln(y_k^t))}{\partial a_k^t} \\
\approx&-\frac{\partial y_k'^t \ln(y_k^t)}{\partial a_k^t} \sum_k |y_k^t - y_k'^t|^\gamma \\
=&(y_k^t - y_k'^t) \sum_{k'} |y_{k'}^t - y_{k'}'^t|^\gamma .
\end{aligned}
\tag{48}
$$