Gram-CTC: Automatic Unit Selection and Target Decomposition for Sequence Labelling

Hairong Liu* [1] **Zhenyao Zhu*** [1] **Xiangang Li** [1] **Sanjeev Satheesh** [1]

Abstract

Most existing sequence labelling models rely on a fixed decomposition of a target sequence into a sequence of basic units. These methods suffer from two major drawbacks: 1) the set of basic units is fixed, such as the set of words, characters or phonemes in speech recognition, and 2) the decomposition of target sequences is fixed. These drawbacks usually result in sub-optimal performance of modeling sequences. In this paper, we extend the popular CTC loss criterion to alleviate these limitations, and propose a new loss function called *Gram-CTC*. While preserving the advantages of CTC, Gram-CTC automatically learns the best set of basic units (grams), as well as the most suitable decomposition of target sequences. Unlike CTC, Gram-CTC allows the model to output variable number of characters at each time step, which enables the model to capture longer term dependency and improves the computational efficiency. We demonstrate that the proposed Gram-CTC improves CTC in terms of both performance and efficiency on the large vocabulary speech recognition task at multiple scales of data, and that with Gram-CTC we can outperform the state-of-the-art on a standard speech benchmark.

1. Introduction

In recent years, there has been an explosion of interest in sequence labelling tasks. Connectionist Temporal Classification (CTC) loss (Graves et al., 2006) and Sequence-to-sequence (seq2seq) models (Cho et al., 2014; Sutskever et al., 2014) present powerful approaches to multiple applications, such as Automatic Speech Recognition (ASR) (Chan et al., 2016a; Hannun et al., 2014; Bahdanau et al., 2016), machine translation (Sébastien et al., 2015), and parsing (Vinyals et al., 2015). These methods are based on 1) a fixed and carefully chosen set of basic units, such as words (Sutskever et al., 2014), phonemes (Chorowski et al., 2015) or characters (Chan et al., 2016a), and 2) a fixed and pre-determined decomposition of target sequences into these basic units. While these two preconditions greatly simplify the problems, especially the training processes, they are also strict and unnecessary constraints, which usually lead to suboptimal solutions. CTC models are especially harmed by fixed basic units in target space, because they build on the independence assumption between successive outputs in that space - an assumption which is often violated in practice.

The problem with fixed set of basic units is obvious: it is really hard, if not impossible, to determine the optimal set of basic units beforehand. For example, in English ASR, if we use words as basic units, we will need to deal with the large vocabulary-sized softmax, as well as rare words and data sparsity problem. On the other hand, if we use characters as basic units, the model is forced to learn the complex rules of English spelling and pronunciation. For example, the "oh" sound can be spelled in any of following ways, depending on the word it occurs in - { o, oa, oe, ow, ough, eau, oo, ew }. While CTC can easily model commonly co-occuring grams together, it is impossible to give roughly equal probability to many possible spellings when transcribing unseen words. Most speech recognition systems model phonemes, sub-phoneme units and senones *e.g,* (Xiong et al., 2016a) to get around these problems. Similarly, state-of-the-art neural machine translation systems use pre-segmented word pieces *e.g,* (Wu et al., 2016a) aiming to find the best of both worlds.

In reality, groups of characters are typically cohesive units for many tasks. For the ASR task, words can be decomposed into groups of characters that can be associated with sound (such as 'tion' and 'eaux'). For the machine translation task, there may be values in decomposing words as root words and extensions (so that meaning may be shared explicitly between 'paternal' and 'paternity'). Since this information is already available in the training data, it is perhaps, better to let the model figure it out by itself. At the same time, it raises another import question: how to de-

*Equal contribution [1]Baidu Silicon Valley AI Lab, 1195 Bordeaux Dr, Sunnyvale, CA 94089, USA. Correspondence to: Hairong Liu <liuhairong@baidu.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

compose a target sequence into basic units? This is coupled with the problem of automatic selection of basic units, thus also better to let the model determine. Recently, there are some interesting attempts in these directions in the seq2seq framework. For example, Chan et al (Chan et al., 2016b) proposed the Latent Sequence Decomposition to decompose target sequences with variable length units as a function of both input sequence and the output sequence.

In this work, we propose **Gram-CTC** - a strictly more general version of CTC - to automatically seek the best set of basic units from the training data, called *grams*, and automatically decompose target sequences into sequences of grams. Just as sequence prediction with cross entropy training can be seen as special case of the CTC loss with a fixed alignment, CTC can be seen as a special case of Gram-CTC with a fixed decomposition of target sequences. Since it is a loss function, it can be applied to many seq2seq tasks to enable automatic selection of grams and decomposition of target sequences without modifying the underlying networks. Extensive experiments on multiple scales of data validate that Gram-CTC can improve CTC in terms of both performance and efficiency, and that using Gram-CTC the models outperform state-of-the-arts on standard speech benchmarks.

2. Related Work

The basic text units that previous works utilized for text prediction tasks (*e.g,*, automatic speech recognition, handwriting recognition, machine translation, and image captioning) can be generally divided into two categories: hand-crafted ones and learning-based ones.

Hand-crafted Basic Units. Fixed sets of characters (graphemes) (Graves et al., 2006; Amodei et al., 2015), word-pieces (Wu et al., 2016b; Collobert et al., 2016; Zweig et al., 2016a), words (Soltau et al., 2016; Sébastien et al., 2015), and phonemes (Lee and Hon, 1988; Sercu and Goel, 2016; Xiong et al., 2016b) have been widely used as basic units for text prediction, but all of them have drawbacks. Using these fixed deterministic decompositions of text sequences defines a prior, which is not necessarily optimal for end-to-end learning.

- Word-segmented models remove the component of learning to spell and thus enable direct optimization towards reducing Word Error Rate (WER). However, these models suffer from having to handle a large vocabulary (1.7 million in (Soltau et al., 2016)), out-of-vocabulary words (Soltau et al., 2016; Sébastien et al., 2015) and data sparsity problems (Soltau et al., 2016).
- Using characters results in much smaller vocabularies (*e.g,* 26 for English and thousands for Chinese), but it requires much longer contexts compared to using words or word-pieces and poses the challenge of composing characters to words (Graves et al., 2006; Chan et al., 2015), which is very noisy for languages like English.
- Word-pieces lie at the middle-ground of words and characters, providing a good trade-off between vocabulary size and context size, while the performance of using word pieces is sensitive to the choice of the word-piece set and its decomposition.
- For the ASR task, the use of phonemes was popular in the past few decades as it eases acoustic modeling (Lee and Hon, 1988) and good results were reported with phonemic models (Xiong et al., 2016b; Sercu and Goel, 2016). However, it introduces the uncertainties of mapping phonemes to words during decoding (Doss et al., 2003), which becomes less robust especially for accented speech data.

Learning-based Basic Units. More recently, attempts have been made to learn basic unit sets automatically. (Luong and Manning, 2016) proposed a hybrid Word-Character model which translates mostly at the word level and consults the character components for rare words. Chan et al (Chan et al., 2016b) proposed the Latent Sequence Decompositions framework to decomposes target sequences with variable length-ed basic units as a function of both input sequence and the output sequence.

There exist some earlier works on the "unit discovery" task (Cartwright and Brent, 1994; Goldwater et al., 2006). A standard problem with MLE solutions to this task is that there are degenerate solutions, *i.e.*, predicting the full corpus with probability 1 at the start. Often Bayesian priors or "minimum description length" constraints are used to remedy this.

3. Gram-CTC

3.1. CTC

CTC (Graves et al., 2006) is a very popular method in seq2seq learning since it does not require the alignment information between inputs and outputs, which is usually expensive, if not impossible, to obtain.

Since there is no alignment information, CTC marginalizes over all possible alignments. That is, it tries to maximize $p(l|x) = \sum_\pi p(\pi|x)$, where x is input, and π represent a valid alignment. For example, if the size of input is 3, and the output is 'hi', whose length is 2, there are three possible alignments, '-hi', 'h-i' and 'hi-', where '-' represents *blank*. For the details, please refer to the original paper (Graves et al., 2006).

3.2. From CTC to Gram-CTC

In CTC, the basic units are fixed, which is not desirable in some applications. Here we generalize CTC by considering a sequence of basic units, called *gram*, as a whole, which is usually more reasonable in many applications.

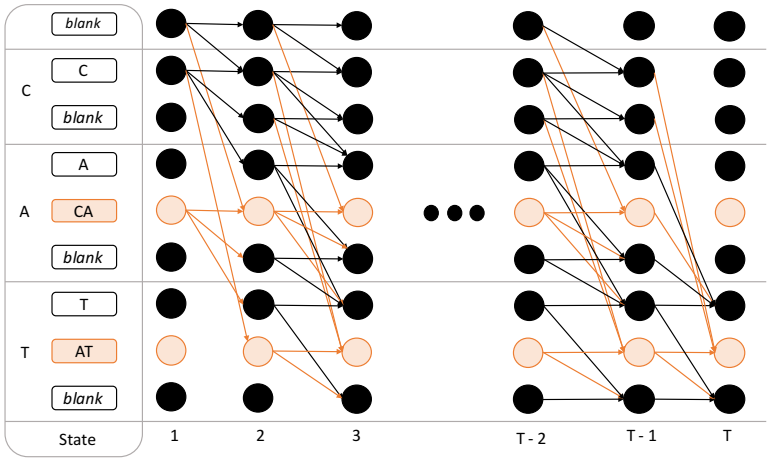


Figure 1. Illustration of the states and the forward-backward transitions for the label 'CAT'. Here we let G be the set of all uni-grams and bi-grams of the English alphabet. The set of all valid states S for the label l = 'CAT' are listed to the left. The set of states and transitions that are common to both vanilla and Gram-CTC are in black, and those that are unique to Gram-CTC are in orange. In general, any extension that collapses back to l is a valid transition - For example, we can transition into ('CAT', 1) from ('CAT', 1), ('CA', 2), ('CA', 1) and ('CA', 0) but not from ('CAT', 0) or ('CAT', 2)

Let G be a set of n-grams of the set of basic units C of the target sequence, and τ be the length of the longest gram in G. A Gram-CTC network has a softmax output layer with $|G|+1$ units, that is, the probability over all grams in G and one additional symbol, *blank*. To simplify the problem, we also assume $C \subseteq G$. [1]

For an input sequence x of length T, let $y = N_w(x)$ be the sequence of network outputs, and denote by y_k^t as the probability of the k-th gram at time t, where k is the index of grams in $G' = G \cup \{blank\}$, then we have

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

Just as in the case of CTC, here we refer to the elements of ${G'}^T$ as paths, and denote them by π, which represents a possible alignment between input and output. The difference is that for each word in the target sequence, it may be decomposed into different sequences of grams. For example, the word 'hello' can only be decomposed into the sequence ['h', 'e', 'l', 'l', 'o'] for CTC (assume uni-gram CTC here), but it also can be decomposed into the sequence ['he', 'll', 'o'] if 'he' and 'll' are in G.

For each π, we map it into a target sequence in the same way as CTC using the collapsing function that 1) removes all repeated labels from the path and then 2) removes all blanks. Note that essentially it is these rules which determine the transitions between the states of adjacent time steps in Figure 1. This is a many-to-one mapping and we denote it by B. Note that other rules can be adopted here and the general idea presented in this paper does not depend on these specific rules. For a target sequence l, $B^{-1}(l)$ represents all paths mapped to l. Then, we have

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

This equation allows for training sequence labeling models without any alignment information using CTC loss, because it marginalizes over all possible alignments during training. Gram-CTC uses the same effect to enable the model to marginalize over not only alignments, but also decompositions of the target sequence.

Note that for each target sequence l, the set $B^{-1}(l)$ has $O(\tau^2)$ more paths than it does in CTC. This is because there are $O(\tau)$ times more valid states per time step, and each state may have a valid transition from $O(\tau)$ states in the previous time step. The original CTC method is thus, a special case of Gram-CTC when $G = C$ and $\tau = 1$. While the quadratic increase in the complexity of the algorithm is non trivial, we assert that it is a trivial increase in the overall training time of typical neural networks, where the computation time is dominated by the neural networks themselves. Additionally, the algorithm extends generally to any arbitrary G and need not have all possible n-grams up to length τ.

3.3. The Forward-Backward Algorithm

To efficiently compute $p(l|x)$, we also adopt the dynamic programming algorithm. The essence here is identifying

[1] This is because there may be no valid decompositions for some target sequences if $C \not\subseteq G$. Since Gram-CTC will figure out the ideal decomposition of target sequences into grams during training, this condition guarantees that there is at least one valid decomposition for every target sequence.

the states of the problem, so that we may solve future states by reusing solutions to earlier states. In our case, the state must contain all the information required to identify all valid extensions of an incomplete path π such that the collapsing function will eventually collapse the complete π back to l. For Gram-CTC, this can be done by collapsing all but the last element of the path π. Therefore, the state is a tuple $(l_{1:i}, j)$, where the first item is a collapsed path, representing a prefix of the target label sequence, and $j \in \{0, \dots, \tau\}$ is the length of the last gram $(l_{i-j+1:i})$ used for making the prefix. $j = 0$ is valid and means that *blank* was used. We denote the gram $(l_{i-j+1:i})$ by $g_i^j(l)$, and the state $(l_{1:i}, j)$ as $s_i^j(l)$. For readability, we will further shorten $s_i^j(l)$ to s_i^j and $g_i^j(l)$ to g_i^j. For a state s, its corresponding gram is denoted by s_g, and the positions of the first character and last character of s_g are denoted by $b(s)$ and $e(s)$, respectively. During dynamic programming, we are dealing with sequence of states, for a state sequence ζ, its corresponding gram sequences is unique, denoted by ζ_g.

Figure 1 illustrates partially the dynamic programming process for the target sequence 'CAT'. Here we suppose G contains all possible uni-grams and bi-grams. Thus, for each character in 'CAT', there are three possible states associated with it: 1) the current character, 2) the bi-gram ending in current character, and 3) the *blank* after current character. There is also one *blank* at beginning. In total we have 10 states.

Supposing the maximum length of grams in G is τ, we first scan l to get the set S of all possible states, such that for all $s_i^j \in S$, its corresponding $g_i^j \in G'$. $i \in \{0, \dots, |l|\}$ and $j \in \{0, \dots, \tau\}$. For a target sequence l, define the forward variable $\alpha_t(s)$ for any $s \in S$ to the total probability of all valid paths prefixes that end at state s at time t.

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\zeta | B(\zeta_g) = l_{1:e(s)}, \zeta_t = s} \prod_{t'=1}^{t} y^{t'}_{\zeta_{t'g}} \tag{3}$$

Following this definition, we have the following rules for initialization

$$\alpha_1(s) = \begin{cases} y_b^1 & s = s_0^0 \\ y^1_{g_i^i} & s = s_i^i \quad \forall i \in \{1, \dots, \tau\} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and recursion

$$\alpha_t(s) = \begin{cases} \hat{\alpha}_{t-1}^i * y_b^t & \\ \qquad \text{when } s = s_i^0, & \\ [\hat{\alpha}_{t-1}^{i-j} + \alpha_{t-1}(s)] * y^t_{g_i^j} & \\ \qquad \text{when } s = s_i^j \text{ and } g_i^j \neq g_{i-j}^j, & \\ [\hat{\alpha}_{t-1}^{i-j} + \alpha_{t-1}(s) - \alpha_{t-1}(s_{i-j}^j)] * y^t_{g_i^j} & \\ \qquad \text{when } s = s_i^j \text{ and } g_i^j = g_{i-j}^j & \end{cases} \tag{5}$$

where $\hat{\alpha}_t^i = \sum_{j=0}^{\tau} \alpha_t(s_i^j)$ and y_b^t is the probability of *blank* at time t.

The total probability of the target sequence l is then expressed in the following way:

$$p(l|x) = \sum_{j=0}^{\tau} \alpha_T(s^j_{|l|}) \tag{6}$$

similarly, we can define the backward variable $\beta_t(s)$ as:

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\zeta | B(\zeta_g) = l_{b(s):l}, \zeta_t = s} \prod_{t'=t}^{T} y^{t'}_{\zeta_{t'g}} \tag{7}$$

For the initialization and recursion of $\beta_t(s)$, we have

$$\beta_T(s) = \begin{cases} y_b^T & s = s_T^0 \\ y^T_{g_T^i} & s = s_T^i \quad \forall i \in \{1, \dots, \tau\} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and

$$\beta_t(s) = \begin{cases} \hat{\beta}_{t+1}^i * y_b^t & \\ \qquad \text{when } s = s_i^0, & \\ [\hat{\beta}_{t+1}^{i+j} + \beta_{t+1}(s)] * y^t_{g_i^j} & \\ \qquad \text{when } s = s_i^j \text{ and } g_i^j \neq g_{i+j}^j, & \\ [\hat{\beta}_{t+1}^{i+j} + \beta_{t+1}(s) - \beta_{t+1}(s_{i+j}^j)] * y^t_{g_i^j} & \\ \qquad \text{when } s = s_i^j \text{ and } g_i^j = g_{i+j}^j & \end{cases} \tag{9}$$

where $\hat{\beta}_t^i = \sum_{j=0}^{\tau} \beta_t(s_{i+j}^j)$

3.4. BackPropagation

Similar to CTC, we have the following expression:

$$p(l|x) = \sum_{s \in S} \frac{\alpha_t(s)\beta_t(s)}{y^t_{s_g}} \quad \forall t \in \{1, \dots, T\} \tag{10}$$

The derivative with regards to y_k^t is:

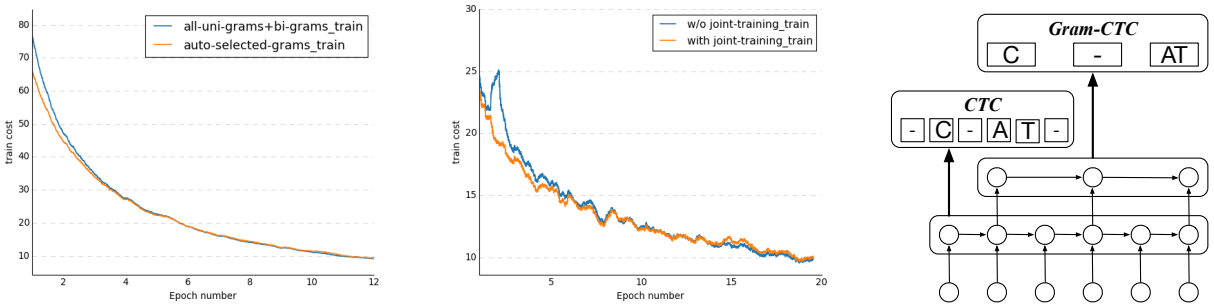
$$\frac{\partial p(l|x)}{\partial y_k^t} = \frac{1}{{y_k^t}^2} \sum_{s \in lab(l,k)} \alpha_t(s)\beta_t(s) \tag{11}$$

where $lab(l, k)$ is the set of states in S whose corresponding gram is k. This is because there may be multiple states corresponding to the same gram.

For the backpropagation, the most important formula is the partial derivative of loss with regard to the unnormalized output u_k^t.

$$\frac{\partial \ln p(l|x)}{\partial u_k^t} = y_k^t - \frac{1}{y_k^t Z_t} \sum_{s \in lab(l,k)} \alpha_t(s)\beta_t(s) \tag{12}$$

where $Z_t \stackrel{\text{def}}{=} \sum_{s \in S} \frac{\alpha_t(s)\beta_t(s)}{y^t_{s_g}}$.



(a) Training curves before (*blue*) and after (*orange*) auto-refinement of grams.

(b) Training curves without (*blue*) and with (*orange*) joint-training

(c) Joint-training Architecture

Figure 2. (Figure 2a) compares the training curves before (blue) and after (orange) auto-refinement of grams. They look very similar, although the number of grams is greatly reduced after refinement, which makes training faster and potentially more robust due to less gram sparsity. Figure (2b) Training curve of model with and without joint-training. The model corresponding to the orange training curve is jointly trained together with vanilla CTC, such models are often more stable during training. Figure (2c) Typical joint-training model architecture - vanilla CTC loss is best applied a few levels lower than the Gram-CTC loss.

4. Methodology

Here we describe additional techniques we found useful in practice to enable the Gram-CTC to work efficiently as well as effectively.

4.1. Iterative Gram Selection

Although Gram-CTC can automatically select useful grams, it is challenging to train with a large G. The total number of possible grams is usually huge. For example, in English, we have 26 characters, then the total number of bi-grams is $26^2 = 676$, the total number of tri-grams are $26^3 = 17576, \ldots$, which grows exponentially and quickly becomes intractable. However, it is unnecessary to consider many grams, such as 'aaaa', which are obviously useless.

In our experiments, we first eliminate most of useless grams from the statistics of a huge corpus, that is, we count the frequency of each gram in the corpus and drop these grams with rare frequencies. Then, we train a model with Gram-CTC on all the remaining grams. By applying (decoding) the trained model on a large speech dataset, we get the real statistics of gram's usage. Ultimately, we choose high frequency grams together with all uni-grams as our final gram set G. Table 1 shows the impact of iterative gram selection on WSJ (without LM). Figure 2a shows its corresponding training curve. For details, please refer to Section 5.2.

4.2. Joint Training with Vanilla CTC

Gram-CTC needs to solve both decomposition and alignment tasks, which is a harder task for a model to learn than CTC. This is often manifested in unstable training curves, forcing us to lower the learning rate which in turn results in models converging to a worse optima. To overcome this difficulty, we found it beneficial to train a model with both the Gram-CTC, as well as the vanilla CTC loss (similar to joint-training CTC together with CE loss as mentioned in (Sak et al., 2015)). Joint training of multiple objectives for sequence labelling has also been explored in previous works (Kim et al., 2016; Kim and Rush, 2016).

A typical joint-training model looks like Figure 2c, and the corresponding training curve is shown in Figure 2b. The effect of joint-training are shown in Table 4 and Table 5 in the experiments.

5. Experiments

We test the Gram-CTC loss on the ASR task, while both CTC and the introduced Gram-CTC are generic techniques for other sequence labelling tasks. For all of the experiments, the model specification and training procedure are the same as in (Amodei et al., 2015) - The model is a recurrent neural network (RNN) with 2 two-dimensional convolutional input layers, followed by K forward (Fwd) or bidirectional (Bidi) Gated Recurrent layers, N cells each, and one fully connected layer before a softmax layer. In short hand, such a model is written as '2x2D Conv - KxN GRU'. The network is trained end-to-end with the CTC, Gram-CTC or a weighted combination of both. This combination is described in the earlier section.

In all experiments, audio data is is sampled at 16kHz. Linear FFT features are extracted with a hop size of 10ms and window size of 20ms, and are normalized so that each input feature has zero mean and unit variance. The network inputs are thus spectral magnitude maps ranging from 0-8kHz with 161 features per 10ms frame. At each epoch, 40% of the utterances are randomly selected to add

Loss	WER
CTC, uni-gram	16.91
CTC, bi-gram	21.63
Gram-CTC, handpick	17.01
Gram-CTC, all uni-grams + bi-grams	16.89
Gram-CTC, auto-refinement	**16.66**

Table 1. Results of different gram selection methods on WSJ dataset.

Loss (stride)	WER 2	WER 4	Epoch Time (mins) 2	Epoch Time (mins) 4
CTC, uni-gram	16.91	23.76	29	16
CTC, bi-gram	20.57	21.63	23	12
Gram-CTC	16.66	18.87	35	18

Table 2. Performances with different model strides on WSJ dataset.

background noise to. The optimization method we use is stochastic gradient descent with Nesterov momentum. Learning hyperparameters (batch-size, learning-rate, momentum, and etc.) vary across different datasets and are tuned for each model by optimizing a hold-out set. Typical values are a learning rate of 10^{-3} and momentum of 0.99.

5.1. Data and Setup

Wall Street Journal (WSJ). This corpora consists primarily of read speech with texts drawn from a machine-readable corpus of Wall Street Journal news text, and contains about *80 hours* speech data. We used the standard configuration of train si284 dataset for training, dev93 for validation and eval92 for testing. This is a relatively 'clean' task and often used for model prototyping (Miao et al., 2015; Bahdanau et al., 2016; Zhang et al., 2016; Chan et al., 2016b).

Fisher-Switchboard. This is a commonly used English conversational telephone speech (CTS) corpora, which contains *2300 hours* CTS data. Following the previous works (Zweig et al., 2016b; Povey et al., 2016; Xiong et al., 2016b; Sercu and Goel, 2016), evaluation is carried out on the NIST 2000 CTS test set, which comprises both Switchboard (SWB) and CallHome (CH) subsets.

10K Speech Dataset. We conduct large scale ASR experiments on a noisy internal dataset of *10,000 hours*. This dataset contains speech collected from various scenarios, such as different background noises, far-field, different accents, and so on. Due to its inherent complexities, it is a very challenging task, and can thus validate the effectiveness of the proposed method for real-world application.

5.2. Gram Selection

We employ the WSJ dataset for demonstrating different strategies of selecting grams for Gram-CTC, since it is a widely used dataset and also small enough for rapid idea verification. However, because it is small, we cannot use large grams here due to data sparsity problem. Thus, the auto-refined gram set on WSJ is not optimal for other larger datasets, where larger grams could be effectively used, but the procedure of refinement is the same for them.

We first train a model using all uni-grams and bi-grams (29 uni-grams and $26^2 = 676$ bi-grams, in total 705 grams), and then do decoding with the obtained model on another speech dataset to get the statistics of the usage of grams. Top 100 bi-grams together with all 29 uni-grams (auto-refined grams) are used for the second round of training. For comparison, we also present the result of the best handpicked grams, as well as the results on uni-grams. All the results are shown in Table 1.

Some interesting observations can be found in Table 1. First, the performance of auto-refined grams is only slightly better than the combination of all uni-grams and all bi-grams. This is probably because WSJ is so small that gram learning suffers from the data sparsity problem here (similar to word-segmented models). The auto-refined gram set contains only a small subset of bi-grams, thus more robust. This is also why we only try bi-grams, not including higher-order grams. Second, the performance of best handpicked grams is worse than auto-refined grams. This is desirable. It is time-consuming to handpick grams, especially when you consider high-order grams. The method of iterative gram selection is not only fast, but usually better. Third, the performance of Gram-CTC on auto-refined grams is only slightly better than CTC on uni-grams. This is because Gram-CTC is inherently difficult to train, since it needs to learn both decomposition and alignment. WSJ is too small to provide enough data to train Gram-CTC.

5.3. Sequence Labelling in Large Stride

Using a large time stride for sequence labelling with RNNs can greatly boost the overall computation efficiency, since it effectively reduces the time steps for recurrent computation, thus speeds up the process of both forward inference and backward propagation. However, the largest stride that can be used is limited by the gram set we use. The (uni-gram) CTC has to work in a high time resolution (small stride) in order to have enough number of frames to output every character. This is very inefficient as we know the same acoustic feature could correspond to several grams of different lengths (*e.g.*, {'i', 'igh', 'eye'}) . The larger the grams are, the larger stride we are potentially able to use.

DS2 (Amodei et al., 2015) employed non-overlapping bi-gram outputs to allow for a larger stride. This imposes an artificial constraint forcing the model to learn, not only the spelling of each word, but also how to split words into bi-grams. For example, *part* is split as [*pa*, *rt*] but the word

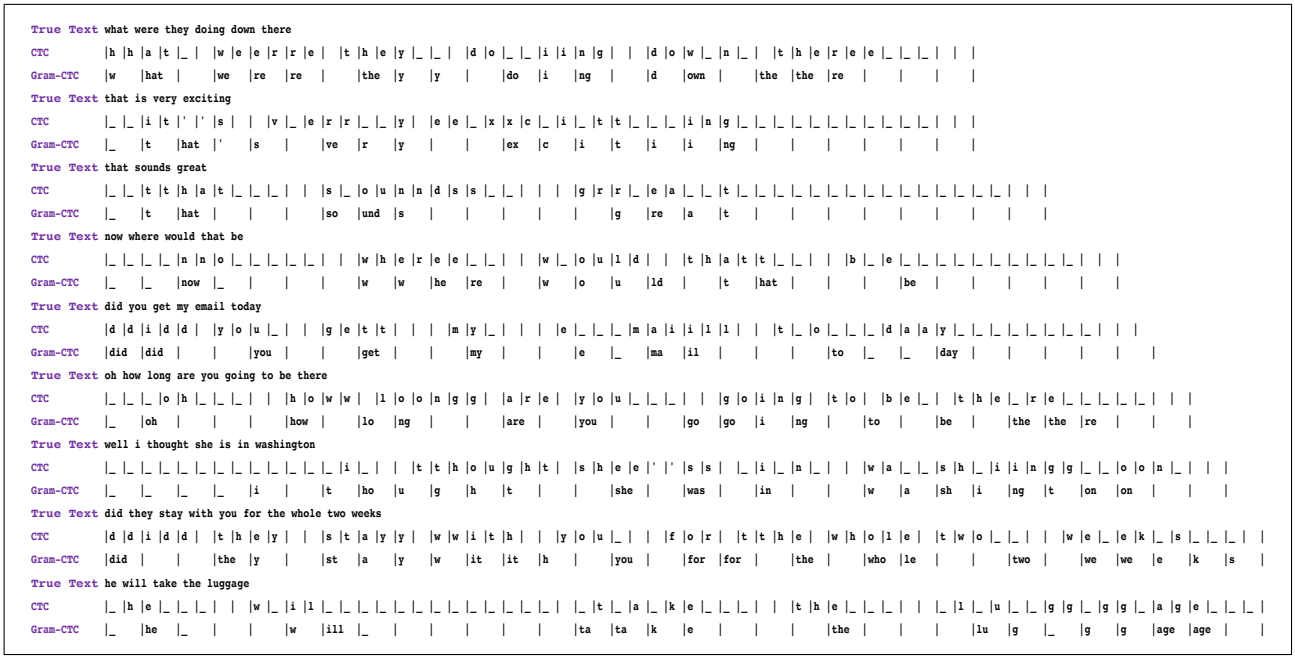


Figure 3. Max-decoding results (without collapsing) of CTC and Gram-CTC on utterances from Switchboard dataset. The predicted characters (by CTC) or grams (by Gram-CTC) at each timestep are separated by "|". As the Gram-CTC model is trained with doubled stride as that of CTC model, we place the grams at a doubled width as we do with characters for better viewing. The "_" represents *blank*.

apart is forced to be decomposed as [*ap*, *ar*, *t*]. Gram-CTC removes this constraint by allowing the model to decompose words into larger units into the most convenient or sensible decomposition. Comparison results show this change enables Gram-CTC to work much better than bi-gram CTC, as in Table 2.

In Table 2, we compare the performance of trained model and training efficiency on two strides, 2 and 4. For Gram-CTC, we use the auto-refined gram set from previous section. As expected, using stride 4 almost cuts the training time per epoch into half, compared to stride 2. From stride 2 to stride 4, the performance of uni-gram CTC drops quickly. This is because small grams inherently need higher time resolutions. As for Gram-CTC, from stride 2 to stride 4, its performance decreases a little bit, while in experiments on the other datasets, Gram-CTC constantly works better in stride 4. One possible explanation is that WSJ is too small for Gram-CTC to learn large grams well. In contrast, the performance of bi-gram CTC is not as good as that of Gram-CTC in either stride.

5.4. Decoding Examples

Figure 3 illustrates the max-decoding results of both CTC and Gram-CTC on nine utterances. Here the label set for CTC is the set of all characters, and the label set for Gram-CTC is an auto-refined gram set containing all uni-grams and some high-frequency high-order grams. Here Gram-CTC uses stride 4 while CTC uses stride 2.

From Figure 3, we can find that: 1) Gram-CTC does automatically find many intuitive and meaningful grams, such as 'the', 'ng', and 'are'. 2) It also decomposes the sentences into segments which are meaningful in term of pronunciation. This decomposition resembles the phonetic decomposition, but in larger granularity and arguably more natural. 3) Since Gram-CTC predicts a chunk of characters (a gram) each time, each prediction utilizes larger context and these characters in the same predicted chunk are dependent, thus potentially more robust. One example is the word 'will' in the last sentence in Figure 3. 4) Since the output of network is the probability over all grams, the decoding process is almost the same as CTC, still end-to-end. This makes such decomposition superior to phonetic decomposition. In summary, Gram-CTC combines the advantages of both CTC on characters and CTC on phonemes.

5.5. Comparison with Other Methods

5.5.1. WSJ DATASET

The model used here is [2x2D conv, 3x1280 Bidi GRU] with a CTC or Gram-CTC loss. The results are shown in Table 3. For all models we trained, language model can greatly improve their performances, in term of WER. Though this dataset contains very limited amount of text data for learning gram selection and decomposition, Gram-

Architecture	WER
Phoneme CTC + trigram LM (Miao et al., 2015)	7.3
Grapheme CTC + trigram LM (Miao et al., 2015)	9.0
Attention + trigram LM (Bahdanau et al., 2016)	9.3
DeepConv LAS + no LM (Zhang et al., 2016)	10.5
DeepConv LAS + LSD + no LM (Chan et al., 2016b)	**9.6**
Temporal LS + Cov + LM (Chorowski and Navdeep, 2016)	**6.7**
Vanilla CTC + no LM (ours)	16.91
Vanilla CTC + LM (ours)	7.11
Gram-CTC + no LM (ours)	16.66
Gram-CTC + LM (ours)	**6.75**

Table 3. Comparison with previous published results with end-to-end training on WSJ speech dataset. The numbers in bold are the best results with and without a language model

CTC can still improve the vanilla CTC notably.

5.5.2. FISHER-SWITCHBOARD

The acoustic model trained here is composed of two 2D convolutions and six bi-directional GRU layer in 2048 dimension. The corresponding labels are used for training N-gram language models.

- Switchboard English speech 97S62
- Fisher English speech Part 1 - 2004S13, 2004T19
- Fisher English speech Part 2 - 2005S13, 2005T19

We use a sample of the Switchboard-1 portion of the NIST 2002 dataset (2004S11 RT-02) for tuning language model hyper-parameters. The evaluation is done on the NIST 2000 set. This configuration forms a standard benchmark for evaluating ASR models. Results are in Table 4.

We compare our model against best published results on *in-domain* data. These results can often be improved using *out-of-domain* data for training the language model, and sometimes the acoustic model as well. Together these techniques allow (Xiong et al., 2016b) to reach a WER of 5.9 on the SWBD set.

5.5.3. 10K SPEECH DATASET

Finally, we experiment on a large noisy dataset collected by ourself for building large-vocabulary Continuous Speech Recognition (LVCSR) systems. This dataset contains about 10000 hours speech in a diversity of scenarios, such as far-field, background noises, accents. In all cases, the model is [2x2D Conv, 3x2560 Fwd GRU, LA Conv] with only a change in the loss function. 'LA Conv' refers to a look ahead convolution layer as seen in (Amodei et al., 2015) which works together with forward-only RNNs for deployment purpose.

As with the Fisher-Switchboard dataset, the optimal stride is 4 for Gram-CTC and 2 for vanilla CTC on this dataset. Thus, in both experiments, both Gram-CTC and vanilla

Architecture	SWBD WER	CH WER
Iterated-CTC (Zweig et al., 2016b)	11.3	18.7
BLSTM + LF MMI (Povey et al., 2016)	8.5	15.3
LACE + LF MMI [2] (Xiong et al., 2016b)	8.3	14.8
Dilated convolutions (Sercu and Goel, 2016)	7.7	**14.5**
Vanilla CTC (ours)	9.0	17.7
Gram-CTC (ours)	7.9	15.8
Vanilla CTC + Gram-CTC (ours)	**7.3**	14.7

Table 4. Comparison with previous published results on Fisher-Switchboard benchmark ("SWBD" and "CH" represent Switchboard and Callhome portions, respectively) using *in-domain* data. We only list results using single models here.

Architecture	WER(No LM)	WER(With LM)
Vanilla CTC	29.1	19.77
Gram-CTC	27.56	19.53
Vanilla CTC + Gram-CTC	25.59	**18.52**

Table 5. LVCSR results on 10K speech dataset.

CTC + Gram-CTC are trained mush faster than vanilla CTC itself. The result is shown in Table 5. Gram-CTC performs better than CTC. After joint-training with vanilla CTC and alignment information through a CE loss, its performance is further boosted, which verifies joint-training helps training. In fact, with only a small additional cost of time, it effectively reduces the WER from 27.56% to 25.59% (without language model).

6. Conclusions and Future Work

In this paper, we have proposed the *Gram-CTC* loss to enable automatic decomposition of target sequences into learned grams. We also present techniques to train the Gram-CTC in a clean and stable way. Our extensive experiments demonstrate the proposed Gram-CTC enables the models to run more efficiently than the vanilla CTC, by using larger stride, while obtaining better performance of sequence labelling. Comparison experiments on multiple-scale datasets show the proposed Gram-CTC obtains state-of-the-art results on various ASR tasks.

An interesting observation is that the learning of Gram-CTC implicitly avoids the "degenerated solution" that occurring in the traditional "unit discovery" task, without involving any Bayesian priors or the "minimum description length" constraint. Using a small gram set that contains only short (up to 5 in our experiments) as well as high-frequency grams may explain the success here.

We will continue investigating techniques of improving the optimization of Gram-CTC loss, as well as the applications of Gram-CTC for other sequence labelling tasks.

References

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. ACM, 2006.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

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, 2016a.

Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Yoshua Bengio, et al. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

Jean Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. 2015.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pages 2773–2781, 2015.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems*, pages 577–585, 2015.

W Xiong, J Droppo, X Huang, F Seide, M Seltzer, A Stolcke, D Yu, and G Zweig. The microsoft 2016 conversational speech recognition system. *arXiv preprint arXiv:1609.03528*, 2016a.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016a.

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. Latent sequence decompositions. In *Arxiv*, 2016b.

Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016b.

Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

Geoffrey Zweig, Chengzhu Yu, Jasha Droppo, and Andreas Stolcke. Advances in all-neural speech recognition. *arXiv preprint arXiv:1609.05935*, 2016a.

Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*, 2016.

K-F Lee and H-W Hon. Large-vocabulary speaker-independent continuous speech recognition using hmm. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 123–126. IEEE, 1988.

Tom Sercu and Vaibhava Goel. Dense prediction on sequences with time-dilated convolutions for speech recognition. *arXiv preprint arXiv:1611.09288*, 2016.

Wayne Xiong, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. Achieving human parity in conversational speech recognition. *arXiv preprint arXiv:1610.05256*, 2016b.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

Mathew Magimai Doss, Todd A Stephenson, Hervé Bourlard, and Samy Bengio. Phoneme-grapheme based speech recognition system. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 94–98. IEEE, 2003.

Minh-Thang Luong and Christopher D Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788*, 2016.

Timothy Andrew Cartwright and Michael R Brent. Segmenting speech without a lexicon: The roles of phonotactics and speech source. *arXiv preprint cmp-lg/9412005*, 1994.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics, 2006.

Hasim Sak, Andrew W. Senior, Kanishka Rao, and FranÃğoise Beaufays. Fast and accurate recurrent neural network acoustic models for speech recognition. *CoRR*, abs/1507.06947, 2015.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. Joint ctc-attention based end-to-end speech recognition using multi-task learning. *arXiv preprint arXiv:1609.06773*, 2016.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pages 167–174. IEEE, 2015.

Yu Zhang, William Chan, and Navdeep Jaitly. Very deep convolutional networks for end-to-end speech recognition. *arXiv preprint arXiv:1610.03022*, 2016.

Geoffery Zweig, Ghengzhu Yu, Jasha Droppo, and Andreas Stolcke. Advances in all-neural speech recognition. *arXiv preprint arXiv:1609.05935*, 2016b.

Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur. Purely sequence-trained neural networks for asr based on lattice-free mmi. *Submitted to Interspeech*, 2016.

Jan Chorowski and Jaitly Navdeep. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016.