Global Normalization for Streaming Speech Recognition in a Modular Framework

Ehsan Variani, Ke Wu, Michael Riley, David Rybach, Matt Shannon, Cyril Allauzen

Google Research
{variani,wuke,riley,rybach,mattshannon,allauzen}@google.com

Abstract

We introduce the Globally Normalized Autoregressive Transducer (GNAT) for addressing the label bias problem in streaming speech recognition. Our solution admits a tractable exact computation of the denominator for the sequence-level normalization. Through theoretical and empirical results, we demonstrate that by switching to a globally normalized model, the word error rate gap between streaming and non-streaming speech-recognition models can be greatly reduced (by more than 50% on the Librispeech dataset). This model is developed in a modular framework which encompasses all the common neural speech recognition models. The modularity of this framework enables controlled comparison of modelling choices and creation of new models. A JAX implementation of our models has been open sourced.¹

1 Introduction

Deep neural network models have been tremendously successful in the field of automatic speech recognition (ASR). Several different models have been proposed over the years: cross-entropy (CE) models with a deep feed-forward architecture [19], connectionist temporal classification (CTC) models [14] with recurrent architectures such as long short-term memory (LSTM) [20], and more recently sequence-to-sequence (Seq2Seq) models like listen, attend and spell (LAS) [8], recurrent neural network transducer (RNN-T) [13], and hybrid autoregressive transducer (HAT) [40]. When configured in non-streaming mode, these neural ASR models have reached state-of-the-art word error rate (WER) on many tasks. However, the WER significantly worsens when they are operating in streaming mode. In this paper, we argue that one main cause of this WER gap is that all the existing models are constrained to be locally normalized which makes them susceptible to the label bias problem [36, 2, 24, 4]. To address this problem, we introduce new category of globally normalized models called Globally Normalized Autoregressive Transducer (GNAT). Our contributions are:

1. Addressing the label bias problem in streaming ASR through global normalization that significantly closes more than 50% of the WER gap between streaming and non-streaming ASR.
2. Efficient, accelerator-friendly algorithms for the exact computation of the global normalization under the finite context assumption.
3. A modular framework for neural ASR which encompasses all the common models (CE, CTC, LAS, RNN-T, HAT), allowing creation of new ones, and extension to their globally normalized counterparts.

¹https://github.com/google-research/last
2 Streaming Speech Recognition

For an input feature sequence \( x = x_1 \ldots x_T \), usually represented as a sequence of real valued feature vectors (such as log mel), and a finite output alphabet \( \Sigma \), we wish to predict the corresponding output label sequence \( y = y_1 \ldots y_T \), \( y_i \in \Sigma \). We call each element \( x_i \) in \( x \) a frame, and each element \( y_i \) in \( y \) an output label. Common ASR models do not directly predict \( y \), but rather an alignment label sequence \( z = z_1 \ldots z_T \), \( z_i \in \Sigma \cup \Delta \). \( \Delta \) is a finite alphabet of control labels, such as the blank label in CTC or RNN-T, or the end-of-sequence label in LAS. There is a deterministic mapping \( B(z) : (\Sigma \cup \Delta)^* \to \Sigma^* \) for obtaining \( y \) from \( z \) (e.g. in RNN-T, we simply remove all the blank labels from \( z \)). An ASR model can then be broken down in two tasks: (1) assigning a score \( \omega(z_i|x, z_{<i}) \) to each alignment sequence \( z \), where \( \omega(z_i|x, z_{<i}) > 0 \) is the alignment score of predicting a single alignment label; (2) finding (usually approximately) arg max

A non-streaming ASR model’s alignment score \( \omega(z_i|x, z_{<i}) \) has access to the entire \( x \) for any \( i \). In contrast, a streaming model’s alignment score takes the form \( \omega(z_i|x, z_{<i}) \) easily interpretable as a conditional probability distribution \( P_o(z_i|x, z_{<i}) \), and thus \( \prod_i \omega(z_i|x, z_{<i}) \) easily interpretable as \( P_o(z|x) \). With samples from the true data distribution \( P(x,y) \), the modeling parameters are optimized by minimizing the negative log-conditional-likelihood loss \( E_P(x,y)[-\log P_o(y|x)] = E_P(x,y)[-\log \sum_{z:B(z)=y} P_o(z|x)] \).

2.1 Label Bias in Streaming ASR

For a non-streaming, locally normalized model, the negative log-conditional-likelihood loss is minimized in the limit (assuming enough model capacity) by setting \( \omega(z_i|x, z_{<i}) \) to the true conditional probability \( P(z_i|x, z_{<i}) \), leading to \( \prod_i \omega(z_i|x, z_{<i}) \) being equal to the true posterior \( P(z|x) \).

For a streaming model, \( x \) is replaced by \( x_{<t(i)} \) in the alignment score (e.g. by using a unidirectional encoder). Here the negative log-conditional-likelihood loss is minimized in the limit by setting \( \omega(z_i|x_{<t(i)}, z_{<i}) \) to \( P(z_i|x_{<t(i)}, z_{<i}) \). As a result, the product \( \prod_i \omega(z_i|x_{<t(i)}, z_{<i}) = \prod_i P(z_i|x_{<t(i)}, z_{<i}) \) in general not equal to \( P(z|x) \) anymore. In other words, using a streaming locally normalized model means that the estimated alignment sequence posterior is the product of some locally normalized alignment scores which depend only on partial input \( x_{<t(i)} \), and as a result can no longer accurately represent the true conditional distribution. This will bias the model towards predictions with low-entropy estimated posterior probabilities at each decoding step. This degrades the model ability to revise previous decisions, a phenomenon called label bias [24, 2].

2.2 Global Normalization

Traditionally, globally normalized models such as Conditional Random Fields (CRF) [24] are used to address the label bias problem. This paper seeks to apply global normalization to modern neural architectures that are more similar to CTC, RNN-T, or LAS, rather than traditional linear models, with the purpose of addressing label bias problem for streaming ASR models.

A globally normalized model does not constrain the alignment score \( \omega(z_i|x, z_{<i}) \) to be locally normalized; it only requires it to be any non-negative score, as long as the denominator \( Z(x) = \sum_x \prod_i \omega(z_i|x, z_{<i}) \) is finite. The finite denominator allows us to interpret \( \prod_i \omega(z_i|x, z_{<i}) / Z(x) \) as a conditional probability distribution \( P_o(z|x) \). Minimum negative log-conditional-likelihood training can be more expensive for globally normalized models due to the need to compute \( Z(x) \) and the corresponding gradients. However with our proposed modular framework, globally normalized model training can be made practical with careful modelling choices on modern hardware.

Any locally normalized model is by definition also globally normalized, whose \( Z(x) = 1 \). While the reverse is not true in general, Appendix E shows that under non-streaming settings with a mild condition, locally and globally normalized models actually express the same class of conditional
distributions. Based on this observation, we argue that under non-streaming settings, with adequately powerful neural architectures, maximum log-conditional-likelihood training should yield behaviorally similar locally or globally normalized models, and thus similar WERs in testing. Results from [37, 17] and our own experiments in Section 6 validate this.

3 A Modular Framework for Neural ASR

In this section, we introduce a modular framework for neural ASR, using the weighted finite state automaton (WFSA) formalism to calculate the conditional probabilities via alignment scores. The modular framework clearly expresses the modelling choices and enables practical globally normalized model training and inference. We use the WFSA formalism as the language for describing our framework because of its succinctness and precision. Our algorithms are implemented from scratch to run on TPUs, without using existing toolkits such as OpenFst [1], Kaldi [33] or K2 [21].

3.1 Preliminaries

We begin with an introduction to the relevant concepts and notations.

A semiring \((\mathbb{K}, \oplus, \otimes, 0, 1)\) consists of a set \(\mathbb{K}\) together with an associative and commutative operation \(\oplus\) and an associative operation \(\otimes\), with respective identities \(0\) and \(1\), such that \(\otimes\) distributes over \(\oplus\), and \(0 \otimes x = x \otimes 0 = 0\). The real semiring \((\mathbb{R}, +, \times, 0, 1)\) is used when the weights represent probabilities. The log semiring \((\mathbb{R} \cup \{-\infty\}, \oplus_{\log}, +, -\infty, 0)\), isomorphic to the real semiring via the log mapping, is often used in practice for numerical stability.\(^2\) The tropical semiring \((\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)\) is often used in shortest-path applications.

A weighted finite-state automaton (WFSA) \(A = (\Sigma, Q, i, F, E)\) over a semiring \(\mathbb{K}\) is specified by a finite alphabet \(\Sigma\), a finite set of states \(Q\), an initial state \(i \in Q\), a set of final states \(F \subseteq Q\), a finite set of transitions \(E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \mathbb{K} \times Q\) (\(e\) denotes the empty label sequence). Given a transition \(e \in E\), \(p[e]\) denotes its origin or previous state, \(n[e]\) its destination or next state, \(o[e]\) its label, and \(\omega[e]\) its weight. A path \(\pi = e_1 \ldots e_k\) is a sequence of consecutive transitions \(e_i \in E\); \(n[e_{i-1}] = p[e_i], i = 2 \ldots k\). The functions \(n, p, \omega\) on transitions can be extended to paths by setting: \(n[\pi] = n[e_k]\) and \(p[\pi] = p[e_1]\) and by defining the weight of a path as the \(\otimes\)-product of the weights of its constituent transitions: \(\omega[\pi] = \omega[e_1] \otimes \ldots \otimes \omega[e_k]\). An unweighted finite-state automaton (FSA) \(A = (\Sigma, Q, i, F, E)\) is simply a WFSA whose transitions and final states are all weighted by \(1\).

\(\Pi(Q_1, Q_2)\) is the set of all paths from a subset \(Q_1 \subseteq Q\) to a subset \(Q_2 \subseteq Q\). \(\Pi(Q_1, y, Q_2)\) is the subset of all paths of \(\Pi(Q_1, Q_2)\) with label sequence \(y = y_1 \ldots y_{\ell}\), \(y_i \in \Sigma\). A path in \(\Pi(I, F)\) is said to be accepting or successful. The weight associated by WFSA \(A\) to any label sequence \(y\) is given by \(A(y) = \bigoplus_{\pi \in \Pi(I, F)} \omega[\pi] \otimes p[n[\pi]]\). The weight of \(A\) is the \(\oplus\)-sum of weights of all accepting paths \(W(A) = \bigoplus_{\pi \in \Pi(I, F)} \omega[\pi] \otimes p[n[\pi]]\). For a semiring \(\mathbb{K}\) where \(\otimes\) is also commutative, the intersection (or Hadamard product) of two WFSA \(A_1\) and \(A_2\) is defined as: \((A_1 \cap A_2)(y) = A_1(y) \otimes A_2(y)\). [28] gives an algorithm to compute the intersection. We can view \(y\) as a WFSA that accepts only \(y\) with weight \(1\), then \(A(y) = W(A \cap y)\).

3.2 Probabilistic Modeling and Inference on Acyclic Recognition Lattices

For any feature sequence \(x = x_1 \ldots x_T\), we use a model with trainable parameters \(\theta\) to induce a recognition lattice WFSA \(A_{\theta, x}\). For a label sequence \(y = y_1 \ldots y_{\ell}\), the weight associated with it, \(A_{\theta, x}(y)\), is interpreted under the log semiring as the unnormalized negative log conditional probability \(P_{\theta}(y \mid x) = \frac{\exp(A_{\theta, x}(y))}{\exp(W(A_{\theta, x}))} = \frac{\exp(W(A_{\theta, x}^\theta(y))}{\exp(W(A_{\theta, x}))}\). The recognition lattice \(A_{\theta, x}\) is designed to be acyclic, and therefore the weight of the automata in both the numerator and denominator above can be efficiently computed by visiting the states of the corresponding WFSA in topological order [27]. See Appendix C for our accelerator friendly version of this algorithm. We can thus train an ASR model by minimizing the negative log-conditional-likelihood on the training corpus \(\mathcal{D}\), and choosing \(\theta^* = \arg \min_\theta E_{(x,y) \in \mathcal{D}}[\log(P_{\theta}(y \mid x)\mid - \log(W(A_{\theta, x} - W(A_{\theta, x} \cap y))) = \arg \min_\theta E_{(x,y) \in \mathcal{D}}[W(A_{\theta, x}) - W(A_{\theta, x} \cap y)]\).
In general, there can be more than one path in \( A_{\theta, x} \) that accepts the same \( y \). During inference, finding the optimal \( \hat{y} = \arg \max_{y} P_{\theta}(y|x) \) requires running the potentially expensive WFSA disambiguation algorithm [30] on \( A_{\theta, x} \). As a cheaper approximation, we instead look for the shortest path \( \hat{x} \) in \( A_{\theta, x} \) under the tropical semiring, and use the corresponding label sequence as the prediction, again using the standard shortest path algorithm for an acyclic WFSA [27].

3.3 Inducing the WFSA

Our framework decomposes the sequence prediction task in ASR into three components, each playing a specific role in inducing the recognition lattice \( A_{\theta, x} \).

- The context dependency FSA \( C = (\Sigma, Q_C, i_C, F_C, E_C) \) is an \( \epsilon \)-free, unweighted FSA, whose states encode the history of the label sequence produced so far. \( C \) is fixed for a given model, independent of input \( x \).
- The alignment lattice FSA \( L_T = (\Sigma, Q_T, i_T, F_T, E_T) \) is an acyclic, unweighted FSA, whose states encode the alignment between input frames \( x \) and output labels \( y \). \( L_T \) depends on only the length \( T \) of input \( x \).
- The weight function \( \omega_{\theta, x} : Q_T \times Q_C \times (\Sigma \cup \epsilon) \rightarrow \mathbb{R} \). \( \omega_{\theta, x} \) is the only component that contains trainable parameters and requires full access to \( x \). This function defines the transition weights in the recognition lattice \( A_{\theta, x} \).

We will discuss how one can define these components in detail in the next section. With \( (C, L_T, \omega_{\theta, x}) \) given, the recognition lattice \( A_{\theta, x} \) is defined as follows:

\[
\begin{align*}
Q_{\theta, x} &= Q_T \times Q_C \\
i_{\theta, x} &= (i_T, i_C) \\
F_{\theta, x} &= F_T \times F_C \\
E_{A_{\theta, x}} &= \left\{ ((q_a, q_c), y, \omega_{\theta, x}(q_a, q_c, y), (q'_a, q'_c)) \mid \\
& \quad y \in \Sigma, (q_a, y, q'_a) \in E_T, (q_c, y, q'_c) \in E_C \right\} \\
& \quad \cup \left\{ ((q_a, q_c), \epsilon, \omega_{\theta, x}(q_a, q_c, \epsilon), (q'_a, q'_c)) \mid \\
& \quad (q_a, \epsilon, q'_a) \in E_T, q_c \in Q_C \right\} \\
\rho_{A_{\theta, x}}(q) &= 1, \forall q \in F_{\theta, x}
\end{align*}
\]

In other words, the topology (states and unweighted transitions) of the recognition lattice \( A_{\theta, x} \) is the same as the FSA intersection \( L_T \cap C \) (see Figure 1c for a concrete example); and the transition weights are defined using \( \omega_{\theta, x} \). The \( \epsilon \)-freeness of \( C \) and the acyclicity of \( L_T \) implies that the recognition lattice \( A_{\theta, x} \) is also acyclic.

4 Components of a GNAT Model

In this section, we define globally normalized autoregressive transducer (GNAT) through the framework above, by specifying each model component.

4.1 Context Dependency

GNAT uses an \( n \)-gram context-dependency defined by \( C_n = (\Sigma, Q_n, i_n, F_n, E_n) \), where \( Q_n = \Sigma^{\leq n} \) corresponds to a label history of length up to \( n \). The initial state \( i_n = \epsilon \) is the empty label sequence. The transitions \( E_n \subseteq Q_n \times \Sigma \times Q_n \) correspond to truncated concatenation: \( E_n = \{(q, y, q') \mid q \in Q, y \in \Sigma \} \) where \( q' \) is the suffix of \( qy \) with length at most \( n \). For example, when \( n = 2 \), the transition \( (ab, c, bc) \) goes from state \( ab \) to state \( bc \) with label \( c \). All states are final: \( F_n = Q_n \). See Figure 1a for the FSA \( C_1 \) when \( \Sigma = \{a, b\} \). Appendix C demonstrates how \( C_n \) can be efficiently intersected during the shortest distance computation thanks to the highly regular structure of \( C_n \).

Although not studied in this paper, our modular framework makes it easy to switch to a more sophisticated context dependency, such as clustered histories often used for context-dependent phone models, or a variable context length as used in \( n \)-gram language models [29].
4.2 Alignment Lattices

Given the input feature sequence length $T$, the alignment lattice FSA $L_T$ defines all the possible alignments between the feature sequence and allowed label sequences. Since the feature sequence length $T$ usually differs from the label sequence length $U$, many different alignments between the feature sequence and a label sequence can be defined. The states in an alignment lattice FSA encode how the next label or $\epsilon$-transition corresponds to some position in the feature sequence.

We can choose different structures for $L_T$ by encoding one or both of the positions in the feature sequence and the label sequence. A simple example is the frame dependent alignment similar to [14], where each frame is aligned to at most one label:

$$Q_T = \{0, \ldots, T\}$$
$$i_T = 0$$
$$F_T = \{T\}$$
$$E_T = \{(t - 1, y, t) \mid y \in \Sigma \cup \{\epsilon\}, 1 \leq t \leq T\}$$

Here any state $t < T$ represents a position in the feature sequence. We start by aligning to the initial frame of the feature sequence, and repeatedly shift to the next frame for every subsequent label or $\epsilon$-transition until all frames have been visited. Figure 1b depicts a frame dependent alignment lattice.
To allow a label sequence longer than the feature sequence, we can use the \( k \)-constrained label and frame dependent alignment similar to [13]:

- \( Q_T = \{(t, n) \mid 0 \leq t \leq T - 1, 0 \leq n \leq k\} \cup \{(T, 0)\} \)
- \( i_T = (0, 0) \)
- \( F_T = \{(T, 0)\} \)
- \( E_T = \{(t, n - 1), (t, n) \mid y \in \Sigma, 0 \leq t \leq T - 1, 1 \leq n \leq k\}\)
  \[ \cup \{(t - 1, k), (t, 0) \mid 1 \leq t \leq T\} \]

Here, up to \( k \) consecutive label transitions can align to any single frame. An \( \epsilon \)-transition is then taken to explicitly shift the alignment to the next frame. The number of labels aligned to one frame is constrained by a constant \( k \) solely in order to impose acyclicity. Figure 1d depicts a \( k \)-constrained label and frame dependent alignment lattice.

Some models may only depend on the position in the label sequence, similar to [8]. In this case we can bound the length of the label sequence by some function \( l(T) \), and use the following label dependent alignment:

- \( Q_T = \{0, \ldots, l(T)\} \)
- \( i_T = 0 \)
- \( F_T = \{l(T)\} \)
- \( E_T = \{(u - 1, y, u) \mid y \in \Sigma, 1 \leq u \leq l(T)\}\)
  \[ \cup \{(u, \epsilon, l(T)) \mid 0 \leq u \leq l(T) - 1\} \]

Here, each label can be seen as aligning to the entire feature sequence, and the \( \epsilon \)-transition serves as an explicit termination of the label sequence. Figure 1e depicts a label dependent alignment lattice.

### 4.3 Weight functions

The weight function \( \omega_{\theta, x} \) translates trainable parameters into transition weights for states in the recognition lattice \( A_{\theta, x} \). The choice of the weight function depends on the choice of the context and alignment lattice FSA, especially the alignment lattice where the meaning of a state directly affects how the weight function can access \( x \).

The experiments discussed in this paper use frame dependent and \( k \)-constrained label and frame dependent alignment lattices. In these two types of alignment lattices, a non-final state \( q_a \) in \( Q_T \) contains a position \( \tau(q_a) \) in the feature sequence (the state itself in the case of frame dependent alignment lattices; the first value in the state in the case of label and frame dependent alignment lattices). Weight functions can thus be defined in three steps,

1. Feed \( x \) into an encoder, such as unidirectional or bidirectional RNN, or a self-attention encoder to obtain the sequence of hidden units \( h \) of dimension \( D \). In the experiments we compared streaming vs non-streaming encoders.
2. Map a single frame of hidden units \( h[t] \) and context state \( q_c \) to a \((|\Sigma| + 1)\)-dimensional vector, corresponding to the unnormalized transition weights for \( y \in \Sigma \cup \{\epsilon\} \).
3. Optionally locally normalize transition weights across \( y \in \Sigma \cup \{\epsilon\} \) given \( (q_a, q_c) \).

For step 2, we experiment with the following concrete modelling choices with varying degree of parameter sharing,

**Per-state linear projection (unshared)** For every context state \( q_c \), we obtain a \( D \times (|\Sigma| + 1) \) projection matrix \( W_{q_c} \) and a \((|\Sigma| + 1)\)-dim bias vector \( b_{q_c} \) from \( \theta \), and define for \( y \in \Sigma \cup \{\epsilon\} \):

\[
\omega_{\theta, x}(q_a, q_c, y) = (W_{q_c} \cdot h[\tau(q_a)]) + b_{q_c})[y]
\]

**Shared linear projection with per-state embedding (shared-emb)** We obtain from \( \theta \) (a) for every context state \( q_c \), a \( D \)-dimensional state embedding \( E_{q_c} \), (b) independent of context states a \( D \times (|\Sigma| + 1) \) projection matrix \( W \) and a \((|\Sigma| + 1)\)-dim bias vector \( b \), and define for \( y \in \Sigma \cup \{\epsilon\} \):

\[
\omega_{\theta, x}(q_a, q_c, y) = (W \cdot \tanh(h[\tau(q_a)] + E_{q_c}) + b)[y]
\]
WFST composition cascades result in weaker modeling power compared to our weight function. All the existing locally normalized models can be explained within a Modular Framework. Among these, MMI [3, 7] is the most relevant to our work. In MMI, a sequence level score is factorized into an acoustic model (AM) score and a language model (LM) score. The denominator is usually approximated over a pruned lattice. Recently [34] introduced lattice-free MMI, which replaces the word level LM with a 4-gram phone LM during training. Both the GNAT and MMI are globally normalized. The GNAT model differs from MMI in several ways: (1) Unlike MMI, GNAT does not require an external LM, nor does it impose any constraint on how the sequence level scores are defined. (2) In MMI training, the LM is kept frozen while the AM parameters are optimized. In GNAT all the model parameters are trained together. (3) GNAT’s recognition lattice encodes the same set of weighted alignment paths during training and inference, therefore the denominator computation in GNAT is exactly matched between training and inference. This does not hold for the standard MMI, because by using different LM topologies between training and inference, the denominator computation in training time only serves as an approximation of the denominator during inference.

Implementation-wise, GNAT trains from scratch without any need for initialization or special regularization techniques as used in the lattice-free version of MMI [34]. In addition we were able to train GNAT models with accelerators without any techniques discussed in [34].

The concept of global normalization has also been visited with deep neural networks [17, 10, 42, 44]. These models can be seen as special cases of MMI based on aspects such as how the sequence level scores are factorized, their use of fixed LM for training, and the mismatching training and inference LMs. Thus the comparison between MMI and GNAT applies here as well. Their use of WFST composition cascades results in weaker modeling power compared to our weight function formulation, and their use of non-streaming encoders combined with their model formulation makes globally normalized models equally expressive to the locally normalized counterparts (Appendix E).

Challenges The main challenge with the GNAT model is its scalability to a larger number of label contexts. At each training step, the model requires \( O(|Q_C| \times |\Sigma|) \) semiring multiplications and additions. For \( n \)-gram context dependency, \( |Q_C| = (|\Sigma|^{n+1} - 1)/(|\Sigma| - 1) \), thus the computation scale exponentially by value of \( n \). However as shown in Appendix D, due to the particular structure of this space the practical computation and memory cost benchmarks do not scale exponentially with \( 0 \leq n \leq 2 \). We also note that large value of \( n \) might also not be necessary: The HAT model [40] reports that a Seq2Seq model with a label history of just the two previous phonemes performs on par with a similar model with a full history trained on very large voice-search corpus. Similar observations are reported in studies with grapheme and wordpiece units [43, 11]. Due to the data sparsity there might not be enough training data to fully represent all \( n \)-gram context states, so increasing the value of \( n \) might not necessarily lead to a WER reduction. One way of dealing with...
larger numbers of states is to use standard pruning techniques to keep only some of the most common states in the training data.

6 Experiments

Data We use the full 960-hour Librispeech corpus [32] for experiments. The input features are 80-dim. log Mel extracted from a 25 ms window of the speech signal with a 10 ms shift. The SpecAugment library with baseline recipe parameters were used [15]. The ground truth transcription is used without any processing and tokenized by the 28 graphemes that appear in the training data.

Architecture Attention-based architectures allows using the same parameterization for streaming and non-streaming models, thus for all the experiments we used 12-layer Conformer encoders [15] with model dimension 512, followed by a linear layer with output dimension 640. The Conformer parameters are set such that the only difference between streaming and non-streaming models is the right context: at each time frame t, the streaming models only access the left context (feature frames from 1 to t), while the non-streaming models can see the entire acoustic feature sequence. To enforce the consistency of the encoder architecture between streaming and non-streaming modes, we removed all the sub-architecture which behaved differently between these two modes. Specifically, we removed the convolution sub-sampling layer, and also forced the stacking layers to only stack within the left context. The baseline experiments use a shared-rnn weight function defined in section 4.3. A single layer LSTM is used with 640 cells. The experiments with the unshared weight function use a linear layer of size $|Q_c| \times 640$ to project the encoder activation at each time frame into the transition weights of the recognition lattice. In our experiments, $|\Sigma| = 32$. For the n-gram context dependency, $|\bar{Q}_c| = (|\Sigma|^{n+1} - 1)/(|\Sigma| - 1)$. The experiments with the shared-emb weight function use an embedding table of size $|Q_c| \times 128$.

Training All models are trained on $8 \times 8$ TPUs with a batch size 2048. The training examples with more than 1961 feature frames or more than 384 labels are filtered out. We used Adam optimizer [22] ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 10^{-9}$) with the Transformer learning rate schedule [41] (10k warm-up steps and peak learning rate 0.05$\sqrt{512}$). We applied the same regularization techniques and the training hyperparameters used in the baseline recipe of [15].

Evaluation We report WER results on standard Librispeech test sets: test_clean and test_other. The WER is either computed with sum-path algorithm or max-path algorithm. The sum-path algorithm approximately merges the alignment hypothesis corresponding to the same label sequence prefix after removal of epsilons. In ideal decoding, sum-path should result in the most likely output label sequence. The max-path algorithm computes the highest scoring alignment path using algorithms in Appendix C.

Baselines The RNN-T baselines are presented in the rows corresponding to the “RNN-T” context dependency in Table 1a. Unlike the other rows in Table 1a, the decoder RNN in RNN-T in principle encodes an unlimited output history as context. The standard RNN-T model effectively uses the label frame dependent alignment lattice with an infinite k. The rows corresponding to the frame dependent alignment lattice represent a slightly modified RNN-T model, where each frame is aligned to exactly one output (either a blank, or a lexical label). The WER differences between non-streaming RNN-T baselines in Table 1a and [15] are mainly due to our modifications to the Conformer encoder for a controlled comparison against streaming models.

Choice of the context dependency Table 1a compares effect of n-gram context dependency for $n = 1, 2, 3$ and baseline RNN-T models. The general observation is that increasing $n$ leads to better WER independent of the other choices of the modeling parameters. However, the model with 3-gram context dependency already performs on par with RNN-T baseline. The 2-gram context dependency perform almost on par as baseline on clean test set while still lagging on the other test set. This is consistent with the earlier observations in [40, 43, 11].

Choice of the alignment lattice The comparison of different alignment lattices in Table 1a suggests that this choice does not significantly contribute to the model performance. While there is a performance gap for 1-gram context dependency, we do not think there is a principal argument in favor of frame dependent alignment lattice. We speculate that this is more due to the choice of optimization parameters. However, the choice of lattice type can have some side effects. For example as $k$ in $k$-constrained label frame dependent alignment lattice increases, the model has more
Table 1: Experiment results

ability to delay its prediction to the end of the signal. This implicit lookahead can translate into performance gains particularly for unidirectional models. By limiting this quantity to 1, we observed that performance on clean and other sets degrades by 34.9% and 36.7%, respectively.

**Choice of the weight function normalization** Table 1b examines the effect of weight normalization on non-streaming and streaming models. Here we present models with 1-gram and 2-gram context dependency with a frame dependent alignment lattice. Note that for 0-gram context dependency with a frame dependent alignment lattice it is easy to show that locally normalized and globally normalized models are equivalent. For non-streaming models, the normalization seems to not have an impact on the performance quality neither for 1-gram nor for 2-gram context dependency experiments. This is validates our argument in the end of Section 2.2 that locally and globally normalized models express the same class of conditional distributions under this particular setting. On the other hand, streaming models significantly benefit from global normalization: For the clean test set, the globally normalized model outperforms the locally normalized model by about 21% relative WER for 1-gram context dependency and by about 20% relative gain for 2-gram context dependency.

The globally normalized model with 2-gram context dependency also beat the baseline streaming RNN-T model in Table 1a and performs significantly closer to the non-streaming RNN-T baseline. The equivalent streaming RNN-T model performs 5.1% on test clean and the non-streaming model performs 2.5% on same test set. The globally normalized model decoded with max-path algorithm performs 3.8% on same test set. The globally normalized model effectively closed almost 50% of the performance gap between streaming and non-streaming models.

The reported performance for the globally normalized models is from max-path decoding, while the baselines benefit from sum-path decoding. Comparing the locally normalized models’ WER from max-path decoding in Table 1b and their counterparts in Table 1a, it is clear that sum-path decoding leads to an extra WER gain. This gain is more significant on test_other. So we expect the globally normalized model performs even better when decoded with sum-path.

The standard sum-path algorithms use several heuristics particularly for path merging and pruning. While similar merging techniques can be applied to the globally normalized models, the pruning heuristics require several adjustments. This is particularly due to the nature of the globally normalized models where the transition weights are not constrained and can take any value, unlike locally normalized models where the transition weights are constrained to be positive number between 0 and 1 and sum to 1 for all the weights leaving the same state in recognition lattice. We plan to explore approaches to improve globally normalized model decoding in the future.
Choice of the weight function architecture  Finally Table 1c compares different choices of the architectures for a streaming model with 2-gram context dependency and frame dependent alignment lattice. While the unshared and shared-rnn architectures are very different in terms of parameter sharing among states, both perform well, though the shared-rnn architecture performs slightly better. The shared-emb architecture performs significantly worst than shared-rnn architecture. Note that the shared-rnn model is able to learn common structures across states in the context dependency while shared-emb does not have such capability.

7 Conclusion

The GNAT model was proposed and evaluated with the focus on the label bias problem and its impact on the performance gap between streaming and non-streaming locally normalized ASR. The finite context property of this model allows exact computation of the sequence level normalization which makes this model differ from existing globally normalized models. Furthermore, the same property allows accelerator friendly training and inference. We showed that the streaming models with globally normalized criteria can significantly close the gap between streaming and non-streaming models by more than 50%. Finally, the modular framework introduced in this paper to explain the GNAT model encompasses all the common neural speech recognition models. This enables fair and accurate comparison of different models via controlled modelling choices and creation of new ASR models.
References


[21] k2-fsa authors. k2-fsa.


