CTC Mixtures and Processes

Benjamin Striner
Carnegie Mellon University
bstriner@cs.cmu.edu

Abstract

We propose using a mixture or process of connectionist temporal classification (CTC) distributions to model sequence generation. Unlike typical recurrent sequence generation, the model is not autoregressive on the output sequence, allowing a broader set of architectures and more efficient training and inference. Unlike naive CTC models, which are not autoregressive but also not universal, a process or sufficiently-large mixture of CTC models is universal and capable of expressing the complexity of language. We propose a novel architecture for training a combination of CTC model and generative model. By using a generative model to produce a latent distribution and conditioning all frames on the latent distribution, the frames can be independently predicted while being dependent on each other. We present results on ASR tasks (WSJ and LibriSpeech) using a VAE as the generative component. Our model performs comparably to equivalent autoregressive models with greater performance and architectural flexibility while outperforming naive CTC and qualitatively demonstrating independent frame predictions impossible under naive CTC.

1 Introduction and Related Work

Automatic speech recognition (ASR) is the process of generating text from speech. One of the many complications of ASR is that the output is a variable-length discrete sequence generated from inputs of different lengths. There are many approaches to ASR that differ in method of representing the alignment between input audio and output text.

- HMM-based models are the traditional approach, modeling the audio as the output of a sequence of transitions between states, each state corresponding to a symbol in the sequence. Hybrid HMM neural network models are an extension that provide more expressive power [1].
- Autoregressive neural models such as Listen, Attend, Spell (LAS) model each output symbol in the sequence as conditioned on the input audio and all previous output symbols. These models may also include a special end-of-sequence symbol. An attention system is typically used for the generation of each output symbol to attend to a different region of the input [2].
- CTC models condition only on the audio and include a special skip-this-frame symbol. CTC models typically require a language model to clean up dirty outputs due to independence of each frame [3]. Due to being purely feed-forward, CTC network architectures can be more diverse and more parallel for training and inference.

Autoregressive models are limiting because each frame is dependent on all previous frames so inference networks must run frame-wise instead of layer-wise. CTC models are limiting because each frame is independent of other frames so the modeled distribution is not universal. We propose to utilize a mixture or process of CTC models such that we can model a universal distribution of sequences and alignments without autoregressing on outputs.

Preprint. Work in progress.

We propose to accomplish this by introducing a latent variable Z that is concatenated to the audio inputs X such that each output symbol Y_i is independent of every other symbol given Z, X, but each output symbol is not conditionally independent given just X. This latent variable can be trained by building an encoder $Z \sim f(X)$ and using variational or adversarial methods to minimize divergence between the marginal distribution of Z over examples and $N(0, 1)$. We demonstrate a VAE that encodes and decodes text, conditioned on audio, that uses CTC loss for the output distribution.

1.1 CTC Loss

CTC loss addresses the problem of labeling unsegmented sequence data [4]. An additional output is introduced to sequential network outputs that indicates whether to skip a frame or not. This enables training and inference where the length of the target sequence may be any length up to the length of the sequence output by the network. Each frame outputs predictions independent of other frames given the network inputs.

CTC loss implicitly calculates alignment between an input and output sequence. CTC loss performs a marginalization over all possible alignments using a dynamic programming lattice. No explicit alignment is produced but activations tend to be "peaky" and sparse. The frame at which the model outputs a given symbol tends to be near the frame where a symbol is spoken even though the model is never provided any alignment data.

Common implementations of CTC loss can be used to provide a loss between input and output sequences as long as the outputs are discrete and the output sequence is the same length or shorter than the input sequence. Similar lattice-based implementations would be possible for non-softmax output distributions.

- Typically, CTC loss is used to output phonemes [5]. With the help of a pronunciation dictionary and a language model this can produce a robust system.
- CTC loss can be used to produce character sequences directly but requires a language model to produce valid spellings [6].
- CTC loss can also be used to produce word sequences but suffers from out-of-vocabulary (OOV) problems [7].

We will be focusing on character-based CTC in this paper because character-based models are potentially the most powerful. Character-based CTC can potentially make an end-to-end system without the need for a language model and without OOV problems. Character-based CTC also suffers the most from conditional independence that causes poor spelling. However, our model could be applied to phonemes, characters, words, or other sequences.

CTC loss can be integrated with a language model, creating the RNN transducer [8].

CTC loss can be used as an auxiliary loss function in a larger ASR system [9].

Typically, LSTM networks are used for CTC models but recent work has included residual convolutional neural networks (ResNets) [10]. We will be using bi-directional LSTM (BLSTM) networks for our model.

1.2 Variational Autoencoders

Variational Autoencoders (VAEs) are a class of generative models [11]. The outputs are modeled as a deterministic transformation from some known latent distribution like a Gaussian $x \sim f(z)$. In order to train this model, we introduce a variational distribution $Z \sim q(x)$, the encoder. The reparameterization trick is used to backpropagate through samples drawn from the distribution defined by the encoder function.

$$
\begin{aligned}
\log p(x) &= \log \mathbb{E}_z p(x|z) \\
&= \log \mathbb{E}_q \frac{p(z)}{qz} p(x|z) \\
&\geq \mathbb{E}_q \log \frac{p(z)}{qz} p(x|z) \\
&= -KL(q(z|x)|p(z)) + \mathbb{E}_q \log p(x|z)
\end{aligned}
$$

These models can be thought of as autoencoders with an additional regularization term that encourages the latent distribution to be similar to a prior. The loss is a bound on the true log-likelihood. During inference, random samples are drawn from the prior distribution and decoded, reproducing the target distribution.

2 CTC VAE

We propose a Connectionist Temporal Classification Variational Autoencoder (CTC VAE) as a method to solve the lack of conditional dependence. A CTC model always produces frame-wise activations that are independent of other frames given the input sequence. The "input sequence" is typically taken to be the acoustic features such as MFCCs or whatever else, and therefore each frame is conditionally independent of each other frame given the input features.

We take the input sequence to be a concatenation of the acoustic features and a latent embedding produced by the VAE model. The predictions at each frame are conditionally independent of other frames given the input features and an embedding of the output. Therefore, the model can learn distributions where the predictions at each frame are conditionally dependent on other frames given the input features.

In the following subsections, we will detail an idealized toy example and then describe our proposed architecture.

2.1 Idealized Toy Example

Imagine input data in which a given 3-frame utterance is labeled "ran" with probability $.5$ and "rum" with probability $.5$. A typical CTC model will produce the following frame-wise predictions.

- Frame 1 is labeled "r" with probability 1
- Frame 2 is labeled "a" with probability $.5$ and "u" with probability $.5$
- Frame 3 is labeled "n" with probability $.5$ and "m" with probability $.5$

Decoding these conditionally-independent CTC predictions results in the following distribution.

- "ran" with probability $1 \times .5 \times .5 = .25$
- "rum" with probability $.25$
- "ram" with probability $.25$
- "run" with probability $.25$

The negative log likelihood (NLL) of the given labels under the CTC model is therefore $\log 4$ but the entropy of the labels is $\log 2$. A model that learns a distribution with a large divergence from the true distribution is not a good model and clearly not a universal distribution. We would like a model that can represent a distribution with an entropy of $\log 2$ in this example.

Our CTC VAE can achieve a loss of $\log 2$ in this example by the following means.

- Half of the latent space ($.5$) produces the activations for "ran" with probability 1.
- Half of the latent space ($.5$) produces the activations for "rum" with probability 1.

Our model is capable of representing the correct distribution. This is because rather than a single CTC-induced distribution, we are able to represent multiple or infinite CTC distributions. During training, the “ran” instance should be encoded to the part of the latent space that decodes to “ran” and the same for “rum”. During evaluation, random samples from the latent space should have equal probability of decoding to “ran” or “rum”, and no probability space is mapped to “ram“ or “run“.

The optimal word error rate of our model would be %50 and the optimal word error rate of a naive CTC model would be %25. However, the character error rate of both models would be %50. Our model can represent dependencies that result in higher accuracy at larger scales of resolution.

2.2 Architecture

A typical CTC architecture is shown in Figure 1. The input features are handled by a deterministic network, typically composed of BLSTM or CNN layers, and the CTC loss is applied on top. A separately-trained language model is typically used to clean the outputs.

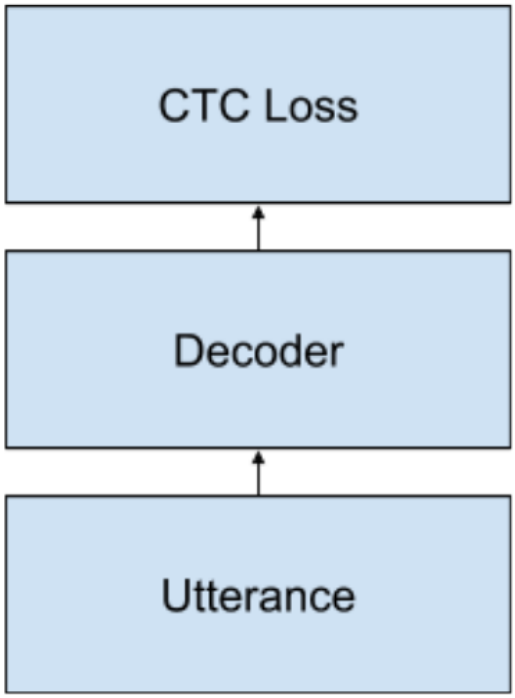

Figure 1: Typical CTC architecture. Language models and other additional architecture may be layered after the CTC decoding.

Our proposed architecture is shown in Figure 2.

- An encoder
  - A neural network forms a representation of the acoustic features
  - A neural network forms a representation of the transcript
  - The two representations are aligned using an attention mechanism
  - The transcript features are concatenated to the acoustic features using that alignment
  - A neural network transforms the concatenated features into a latent μ and σ
  - Draw samples from $N(\mu, \sigma^2)$ using the reparameterization trick.
- A decoder that looks like a typical CTC architecture except with an additional latent input.

The attention mechanism applies a fully-connected layer to the transcript features and a different fully-connected layer to the acoustic features. The dot product between these two outputs is computed to create an energy between each step of the transcript and each step of the utterance. Then, a softmax

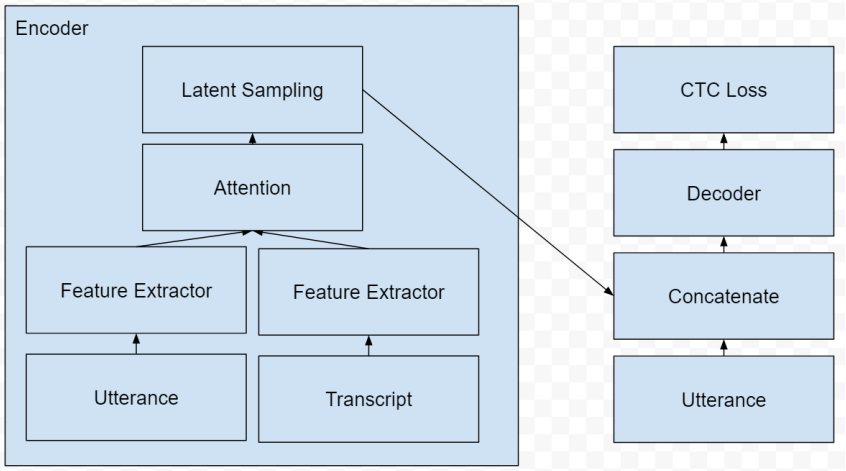

Figure 2: Proposed CTC VAE architecture.

activation is applied to normalize the outputs over the transcript dimension. Typical learned attention is shown in Figure 3. The model learns to "look" at specific portions of the text when it is producing the latent representation for each frame of audio.

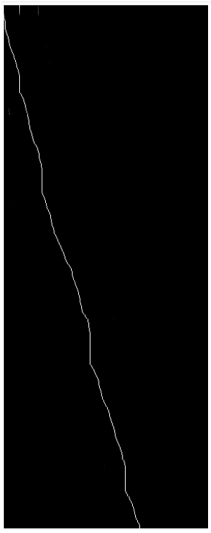

Figure 3: Typical learned attention

We also experimented with a modified model, shown in Figure 4. In this model, a single network creates representations of acoustic features. Both encoder and decoder are conditioned on a shared representation. This reduces the number of computations and parameters.

3 Experiments and Results

We compared results using LAS (baseline autoregressive model), CTC (baseline non-autoregressive model) and our CTC VAE on the datasets, evaluating word and character error rates on the LibriSpeech and WSJ datasets. We used Kaldi for input processing to produce high-resolution mel features and pitch features. We subsampled every 3 frames during training and averaged every 3 frames during evaluation to reduce the input sequence length. We did not perform additional data augmentation or other regularization of the models for a fair comparison. Models were trained with the Adam optimizer with a learning rate of $3e-4$.

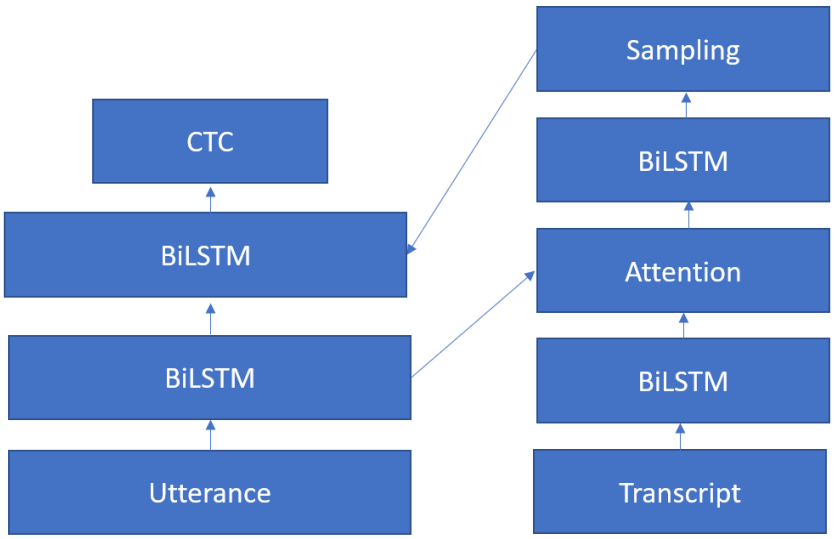

Figure 4: Proposed CTC VAE shared-parameter architecture.

For our CTC-based experiments, we used a CTC model with a 5-layer 320 dimensional BLSTM. The encoder utilized a 2-layer 256 dimensional BLSTM for each of: audio feature extraction, transcript feature extraction, and projection from the output of the attention mechanism to the latent space. During evaluation the encoder portion is not used. As such, our experiments are evaluated comparing the same CTC architectures during evaluation, with the addition of a VAE.

On LibriSpeech:

- LAS model: 18% WER
- CTC model: 25% WER, 9% CER
- CTC VAE model: 19% WER, 6% CER

On WSJ:

- LAS model: 20% WER
- CTC model: 28% WER, 10% CER
- CTC VAE model: 21% WER, 7% CER

Qualitatively, we see results that are not possible with a vanilla CTC. For a single utterance, we can sample from the latent distribution multiple times and get multiple decodings. The frames in each of these decodings is dependent on the latent encoding. For example, for a given utterance, we sampled multiple greedy CTC decodings and found an output distribution split between the following two decodings:

- “IF I CAN GET PATIENCE”
- “IF I CAN GET PATIENTS”

These decodings are two similar-sounding words produced at the character level by a CTC model. Note the similarity to our “rum” “ran” example. Our decodings did NOT include the following, which would necessarily be included by a CTC model:

- “IF I CAN GET PATIENTE”
- “IF I CAN GET PATIENCS”

4 Discussion

We presented a modification to CTC models called CTC VAE that allows CTC models to learn distributions with conditionally-dependent frames. This model allows CTC to learn correct spelling and makes CTC appropriate for a true end-to-end model without additional language models or other support. We showed how generally a variational or adversarial approach could be used to train CTC processes, such that an IWAE or AAE could improve on the latent encodings.

The ability to perform sequence prediction without autoregression changes the flow of information and the way in which inference is performed. Even a large bi-directional recurrent network is orders faster than a smaller uni-directional recurrent network if it can run layer-wise instead of step-wise.

We presented character-based end-to-end CTC VAE models for ASR. Future work would include using combinations of CTC and other, more powerful, generative models. Future work would also include using similar models unconditionally or conditioned on different inputs for alternative tasks and modifying the CTC lattice to support alternative output distributions.

5 Challenges and Remaining Work

We implemented CUDA wrappers around warp-ctc for a more stable CTC implementation than was available.

Our baselines and hopefully our model as well can benefit greatly from data augmentation. Current state-of-the-art results are only possible with advanced audio augmentation techniques and we hope to include augmentation. Seeing almost %10 absolute word error reduction from SpecAugment as preprocessing.

We plan to architect our model with residual CNNs. LSTM networks traditionally work best for CTC models. We plan to investigate whether the same holds true for CTC VAE models.

We also plan to perform a deeper qualitative and quantitative analysis on the out-of-vocabulary (OOV) problem, analyzing how models handle inference of words not in the training vocabulary and creation of new words.

References

[1] José Novoa, Josué Fredes, and Néstor Becerra Yoma. Dnn-based uncertainty estimation for weighted DNN-HMM ASR. *CoRR*, abs/1705.10368, 2017.

[2] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. Listen, attend and spell. *CoRR*, abs/1508.01211, 2015.

[3] Yosuke Higuchi, Shinji Watanabe, Nanxin Chen, Tetsuji Ogawa, and Tetsunori Kobayashi. Mask ctc: Non-autoregressive end-to-end asr with ctc and mask predict, 2020.

[4] 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*, ICML '06, pages 369–376, New York, NY, USA, 2006. ACM.

[5] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. Phoneme recognition in TIMIT with BLSTM-CTC. *CoRR*, abs/0804.3269, 2008.

[6] Kyuyeon Hwang and Wonyong Sung. Character-level incremental speech recognition with recurrent neural networks. *CoRR*, abs/1601.06581, 2016.

[7] Jinyu Li, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong. Advancing acoustic-to-word CTC model. *CoRR*, abs/1803.05566, 2018.

[8] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012.

[9] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney. Improved training of end-to-end attention models for speech recognition. *CoRR*, abs/1805.03294, 2018.

[10] Yisen Wang, Xuejiao Deng, Songbai Pu, and Zhiheng Huang. Residual convolutional CTC networks for automatic speech recognition. *CoRR*, abs/1702.07793, 2017.

[11] Carl Doersch. Tutorial on Variational Autoencoders. *arXiv e-prints*, page arXiv:1606.05908, June 2016.