Report Of The Work Done On Discipline of Deep Learning By Marcelo Pedro

***Abstract*— In this work, we test and evaluate the end-to-end automatic speech recognition (ASR) using a Criticizing Language Model (CLM) proposed by [1], using the Librispeech dataset [2] and a private dataset provided by Datanuvem Soluções Eireli.**

Keywords— automatic speech recognition, Criticizing Language Mode, Librispeech, private dataset

1. Introduction

In terms of technological development, we may still be at least a couple of decades away from having truly autonomous, intelligent artificial intelligence systems communicating with us in a genuinely "human-like" way.

But, in many ways, we're progressing steadily towards this future scenario at a surprisingly fast pace thanks to the continuing development of what is known as automated speech recognition technology. And at least so far, it's looking to promise some truly useful innovations in user experience for all sorts of applications.

ASR is the technology that allows human beings to use their voices to speak with a computer interface in a way that, in its most sophisticated variations, resembles normal human conversation.

2. Related Works

A few other ASR are Fairseq [3], NeMo [4], and ESPnet [5]. Fairseq is developed by Facebook to support sequence-to-sequence processing. It includes models such as ConvS2S [6], transformers [7], and wav2vec [8]. NeMo is a toolkit for conversational AI developed by NVIDIA, which provides useful neural modules for many speeches processing tasks, including speech recognition, speaker diarization, voice-activity detection and text-to-speech. ESPnet started as an end-to-end speech recognition library and progressively grew to support different tasks.

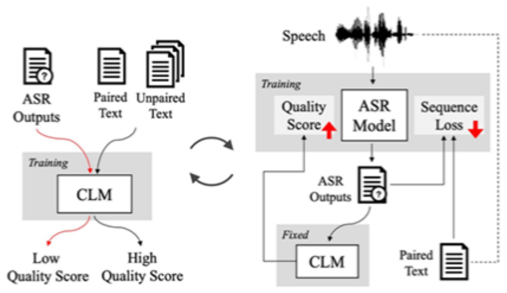

Figure 1 Overview of the Adversarial Training (AT) approach for end-to-end speech recognition. The two steps here are conducted iteratively: (a) a Criticizing Language Model (CLM) is trained to evaluate the quality score given a text sequence, and (b) and ASR model is trained to minimize the sequence loss calculated with ground truth while maximizing the scores given by CLM.

3. Methodology

In the Adversarial Training approach to end-to-end speech recognition proposed by [1], the ASR model is considered as a generator conditioned on the input speech signal whose output is the corresponding transcription. A Criticizing Language Model (CLM) is used as a discriminator to distinguish real text from ASR transcriptions. The ASR model and CLM are trained iteratively, so they learn from each other step by step. Figure 1 gives an overview of the proposed approach.

In Figure 1 (a) for CLM training step, the CLM learns to assign higher scores to real text and lower scores to ASR transcriptions. The real text here does not have to be paired with audio, which is how the unpaired text can be involved in the training processes. This CLM is to evaluate the quality of each given text sequence by offering a score for adversarial purposes, with details given in Sec. 2.2.

In Figure 1(b) for ASR model learning step, the parameters of CLM are fixed and we train the ASR model by minimizing the sequence loss (e.g. seq2seq loss and/or CTC loss) evaluated with the ground truth just as typical end-to-end training. At the same time, with CLM acting as a discriminator evaluating the quality score for the output of ASR model, the ASR model also has to learn to generate transcriptions obtaining higher quality scores from CLM. The details of the ASR model are in Sec. 2.3.

Note that the ASR model and the CLM are learned iteratively both from scratch. No pre-training is needed. Each of them improves itself based on the challenges offered by the other in each iteration. Once the training ends, the ASR model is expected to implicitly leverage the linguistic knowledge learned from CLM, and the latter is no longer used during testing. This approach can be used with any existing end-to-end speech recognition frameworks and any language modeling framework. Below we take one example set of the proposed approach to explain the details.

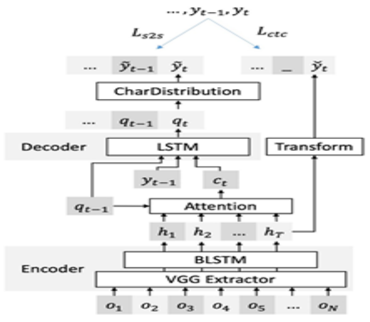

Figure 2 Network architecture of the ASR model.

ASR

Any network architecture for end-to end speech recognition can be used here, while Figure 2 gives the one used in this work, following the previous work [9] of integrating attentioned Seq2seq with CTC. The model takes a sequence of speech features O = $\mathbf{o}_1$, $\mathbf{o}_2$, ..., $\mathbf{o}_N$ with length N as the input. O is encoded into sequence of hidden state H = $\mathbf{h}_1$, $\mathbf{h}_2$, ..., $\mathbf{h}_T$ by the encoder (consists of a VGG extractor performing input downsampling followed by several BLSTM layers) with T being the output sequence length. The decoder is a single layer LSTM maintaining its own hidden state q. For each time index t, location-aware attention mechanism [10] Attention is used to integrate H with the previous decoder state $\mathbf{q}_{t-1}$ to generate the context vector ct. The decoder then decodes $\mathbf{c}_t$ together with the ground truth one-hot vector of the previous time step $\mathbf{y}_{t-1}$ to $\mathbf{q}_t$. Finally, a fully connected layer with softmax activation CharDistribution takes $\mathbf{q}_t$ and predicts the distribution vector $\bar{y}_t$. $\mathbf{h}_t$ is also projected to $\bar{y}_t$ with linear layer Transform as output to help in learning of the encoder as shown in previous work [12]. The ASR model outputs two-character sequences, ȳ = $\bar{y}_1$, $\bar{y}_2$, ..., $\bar{y}_T$ and ŷ = $\hat{y}_1$, $\hat{y}_2$, ..., $\hat{y}_T$, respectively supervised by Seq2seq loss and CTC loss. CLM only takes ȳ as input. During testing, ȳ and ŷ are integrated into a single output sequence just as done in the previous work [9].

Seq2seq Loss. The Seq2seq ASR model is to estimate the posterior probability,

$$P_{s2s}(\tilde{y}|O) = \prod_{l=1}^{T} P_{s2s}(\tilde{y}_l|\tilde{y}_{1:l-1}, O).$$

The loss function of the Seq2seq model can be then computed as below,

$$L_{s2s} \equiv -\log P_{s2s}(y|O) = -\sum_{t=1}^{T} \log P_{s2s}(y_t|y_{1:t-1}, O),$$

except here $\mathbf{y}$ = $\mathbf{y}_1$, $\mathbf{y}_2$, ..., $\mathbf{y}_T$ is the ground truth of O with length T.

Language Model

CLM takes either real text or ASR transcriptions as input and outputs a scalar s as the quality score. The real text is represented as a sequence of one-hot vectors y = y1, y2, ..., yL, while for ASR transcriptions this is a sequence of vectors for distributions ÿ = ÿ1, ÿ2, ÿ3, ... Figure 3 illustrates an example architecture of CLM used in this work. The input vector sequence y (or y˜) is first projected to a lower dimensional space through a single layer neural net. Next, two layers of one-dimensional convolution neural network extracts features for each time index. Finally, average pooling over the time axis is applied to get a single representative feature, which is then transformed to a scalar s (the quality score) with linear projection.

The reason a convolution-based network instead of a recurrent network is used in Fig 2 is twofold. Convolution with small window size captures local relation features, which can then be averaged over time. Also, CNN based network is relatively more computationally efficient, which is important in adversarial training. But other network architectures such as RNN-LM [9] can also be used here

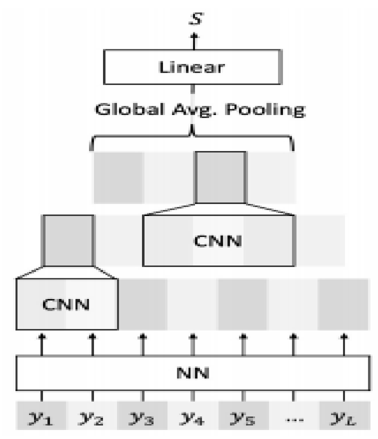

Figure 3 Network architecture of the CLM.

Loss Function. A major problem here is that soft distribution vectors produced by the ASR model is very different from one-hot vectors for real text data, making the task of CLM trivial, and the ASR model almost always fail to compete against it. Thanks to Wasserstein GAN (WGAN) [11] which addressed the above problem to some good extent. Based on the concept of WGAN, CLM is designed to estimate the Earth-Mover (Wasserstein-1) [12] distance between sequences from real data and ASR output. The loss function of CLM is the weighted sum of a loss L_D and a gradient penalty gp as follows,

$$L_{CLM} = \lambda_{CLM} L_D + \lambda_{gp}\, gp,$$

in which λ_{CLM}, λ_{gp} are wieghts and L_D and gp are respectively in Eq (2) and Eq (3) below.

$$L_D = \mathop{\mathbb{E}}_{\tilde{y}\sim\mathbb{P}_a}\left[CLM(\tilde{y})\right] - \mathop{\mathbb{E}}_{y\sim\mathbb{P}_d}\left[CLM(y)\right],$$

where CLM(y) is the quality score for y given by CLM, $Þ_a$ Pa the distribution of ASR output ÿ and $Þ_d$ the distribution of real text y. The 1-Lipschtiz restriction is imposed for CLM by applying the gradient penalty [13] as below,

$$gp = \mathop{\mathbb{E}}_{\hat{y}\sim\mathbb{P}_{\hat{y}}}\left[(\|\nabla_{\hat{y}} CLM(\hat{y})\| - 1)^2\right],$$

where ŷ are samples generated by randomly interpolating between ÿ and y, and $Þ_{\hat{y}}$ is the distribution of ŷ.

4. Experimental Setup

The ASR model and CLM were trained using two-fold strategy. In the first phase we used LibriSpeech [2], which is an English dataset. The experiments done on the LibriSpeech dataset were done as follows. 460 hours of clean speech data and their transcriptions are used as the paired data. The clean development set and clean test set were used for evaluation. In the second phase we used a private Portuguese dataset provided by Datanuvem Soluções Eireli. This dataset contains 3 hours of Portuguese speech. This dataset contains noisy audio speech made by people and answering machine audio speech both recorded by telephone. On the both phases we used the end-to-end speech processing toolkit ESPnet [5] for data preprocessing and customized it for our adversarial training processes. We followed the previous work [9, 14] to use 80-dimensional log Mel-filter bank and 3-dimensional pitch features as the acoustic features. Text data are represented by sequences of 5000 subword units one-hot vectors. For the CLM model, the dimension of the output of all layers were set to 128 except the last. The first convolution had a window size of 2 and stride of 1, and the second had window size 3 and stride 1. Batch normalization is applied between layers. For the ASR model, the encoder included a 6-layer VGG extractor with down sampling used in the previous work [9] and a 5-layer BLSTM with 320 units per direction. 300-dimensional location-aware attention [10] was used in the attention layer. The decoder was a single layer LSTM with 320 units. $\boldsymbol{\lambda_{gp}}$ was set to 10 and $\boldsymbol{\lambda_{s2s}}$ is set to 0.5. $\boldsymbol{\lambda_{CLM}}$ is set to $\mathbf{10^{-4}}$ since CLM output value was usually much higher than other loss values. Also, the update frequency of CLM is set to 5 times less than the ASR model to stabilize AT process.

5. Results

The CLM was trained in the first phase for 1538 epochs and achieved 4,59 for the total loss in the end of the training, as we can see on the Figure 4, on the Librispeech dataset.

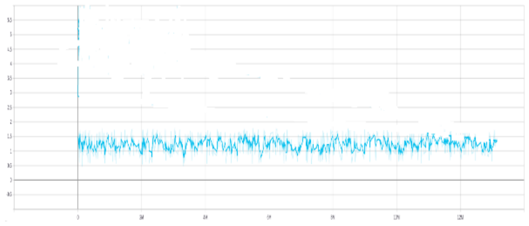

Figure 4 Charge of the loss of the CLM during training of the first phase

The ASR model was trained in the first phase for 69 epochs and achieved 0.4 for the total loss in the end of the training, as we can see on the Figure 5, on the Librispeech dataset.

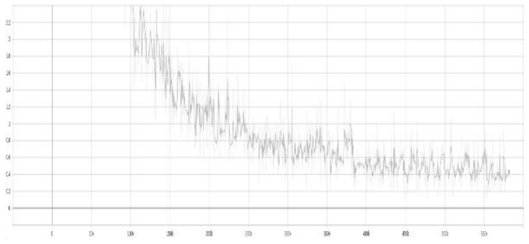

Figure 5 Charge of the loss of the ASR during training of the first phase

In the end of the first phase, we took the ASR and CLM model trained in this phase and evaluated on the test set of the Librispeech dataset, obtaining the following result in the CSV file named decode_example_test_output.csv available on https://github.com/LCAD-UFES/Deep-Learning-2020-1/tree/main/End-to-end-ASR-Pytorch/result.

For the second phase we take the pre-trained CLM in the first phase, and trained for 2582 epochs and achieved 0,11 for the total loss in the end of the training, as we can see on the Figure 6, on the private dataset provided by Datanuvem Soluções Eireli.

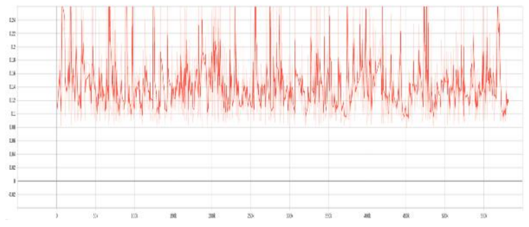

Figure 6 Charge of the loss of the CLM during training of the second phase

For the second phase we take the pre-trained ASR in the first phase, and trained for 1332 epochs and achieved 1,2e^-4 for the total loss in the end of the training, as we can see on the Figure 7, on the private dataset provided by Datanuvem Soluções Eireli.

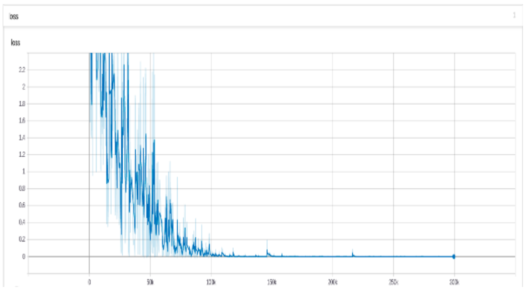

Figure 7 Charge of the loss of the ASR during training of the second phase

In the end of the second phase, we took the ASR and CLM model trained in this phase and evaluated on the test set of the Librispeech dataset, obtaining the following result in the CSV file named decode_example_portuguese_test_output.csv available on https://github.com/LCAD-UFES/Deep-Learning-2020-1/tree/main/End-to-end-ASR-Pytorch/result.

Conclusion

We achieve good results in the end of the first phase as can see in the CSV file named decode_example_test_output.csv available on https://github.com/LCAD-UFES/Deep-Learning-2020-1/tree/main/End-to-end-ASR-Pytorch/result. And we achieve not good results in the end of the second phase as we can see the CSV file named decode_example_portuguese_test_output.csv available on https://github.com/LCAD-UFES/Deep-Learning-2020-1/tree/main/End-to-end-ASR-Pytorch/result. One theory for the bad results on the second phase is index error during the training on the dataset provided by Datanuvem Soluções Eireli. We will fix it on the next days.

6. References

[1] Liu, Alexander H., et al. "Sequence-to-Sequence Automatic Speech Recognition with Word Embedding Regularization and Fused Decoding." ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2020.

[2] Panayotov, Vassil, et al. "Librispeech: an asr corpus based on public domain audio books." 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE, 2015.

[3] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli. Fairseq: A fast, extensible toolkit for sequence modeling, 2019. arXiv:1904.01038.

[4] O. Kuchaiev, J. Li, H. Nguyen, O. Hrinchuk, R. Leary, B. Ginsburg, S. Kriman, S. Beliaev, V. Lavrukhin, J. Cook, P. Castonguay, M. Popova, J. Huang, and J. M. Cohen. NeMo: a toolkit for building AI applications using Neural Modules, 2019. arXiv:1909.09577.

[5] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai. ESPnet: End-to-end speech processing toolkit. In Proc. of Interspeech, 2018.

[6] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin. Convolutional sequence to sequence learning. In Proc. of ICML. PMLR, 2017.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In Proc. of NeurIPS, 2017.

[8] A.Baevski, Y. Zhou, A. Mohamed, andM.Auli. wav2vec2.0: A framework for self-supervised learning of speech representations. In Proc. of NeurIPS, 2020.

[9] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, "Advancesinjoint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm," in Interspeech, 2017.

[10] 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, 2015, pp. 577–585.

[11] Martin Arjovsky, Soumith Chintala, and L´eon Bottou, "Wasserstein generative adversarial networks," in International Conference on Machine Learning, 2017, pp. 214–223.

[12] Cédric Villani, Optimal transport: old and new, vol. 338, Springer Science & Business Media, 2008.

[13] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, "Improved training of wasserstein gans," in Advances in Neural Information Processing Systems, 2017, pp. 5767–5777.

[14] Tomoki Hayashi, Shinji Watanabe, Yu Zhang, Tomoki Toda, Takaaki Hori, Ramon Astudillo, and Kazuya Takeda, "Backtranslation-style data augmentation for end-to-end asr," arXiv preprint arXiv:1807.10893, 2018.