Adversarial Training of End-to-end Speech Recognition Using a Criticizing Language Model

Alexander H. Li et al., 2018

Siting Liang
Institute for Computational Linguistics of University Heidelberg
Seminar: Advances in Seq2Seq

Overview

Adversarial Training

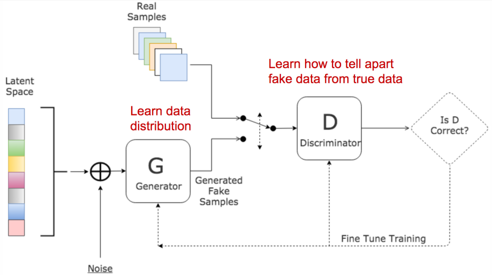


Figure 1: Generic Architecture of GAN (Goodfellow et al., 2014)

- Generator (G) predicts the associated features given a hidden representation.
- Discriminator (D) estimates the probability of the generated feature representation coming from the real dataset.

GAN's training objective

$$\min_G \max_D L(D, G) = \mathbb{E}_{x\sim P_r(x)}[\log D(x)] + \mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))]$$
$$= \mathbb{E}_{x\sim P_r(x)}[\log D(x)] + \mathbb{E}_{x\sim P_g(x)}[\log(1 - D(x)]$$

- ”min-max optimization” updates each model independently,
- G is trained to make discriminator to produce a high probability for a fake sample generated from G
- minimize $\mathbb{E}_{z\sim P_z(z)}[\log(1 - D(G(z)))$
- maximize $\mathbb{E}_{x\sim P_r(x)}[\log D(x)]$ through learning the real data distribution and has no impact on G

GAN's training obejctive

- Problems in training:
  - once trained, the gradient of the loss functions will be close to zero and $D(x)$ gives no effective critic for updating G,
  - but if the discriminator is unable of distinguish fake from true, it couldn't pass accurate feedback to generator.

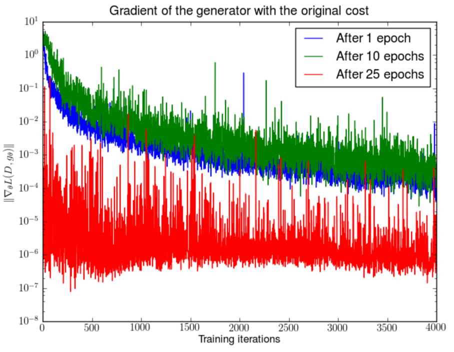


Figure 2: Discriminator gets better after 4000 iterations, the gradient norms vanishes fast (Lil'Log, 2017-08-20)

Improve GAN using Wasserstein Distance as Loss Function

$$GAN: \min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

$$WGAN: W(D,G) = \frac{1}{K}\sup_{\|f\|_L\leq K} \mathbb{E}_{x\sim p_r(x)}[D(x)] - \mathbb{E}_{z\sim p_z(z)}[D(G(z))]$$

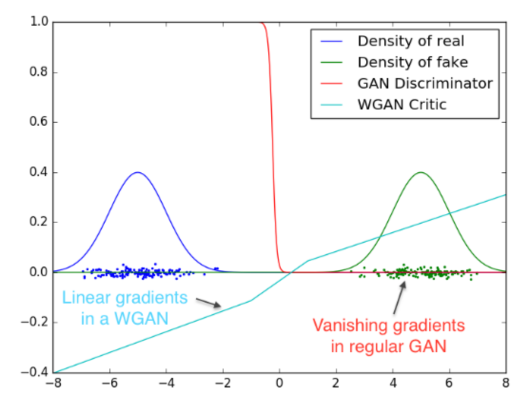


Figure 3: Using a linear loss function is giving clean gradient everywhere (Arjovsky, Chintala, Bottou, 2017)

WGAN changes

- $W(p_r, p_g) = \frac{1}{K} \sup_{||f||_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$
- end up with $\max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$
- supremum is attained for $w \in W$,
  i.e. f_w depends on a compact space W, not individual weights anymore
- practical trick to enforce Lipschitz constraint: clamp w after every update to a range, such as
  set $w \in W = [-0.01, 0.01]$

WGAN algorithm

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.
Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

```
while θ has not converged do
    for t = 0, ..., n_critic do
        Sample {x^(i)}_{i=1}^m ~ P_r a batch from the real data.
        Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
        g_w ← ∇_w [1/m Σ_{i=1}^m f_w(x^(i)) − 1/m Σ_{i=1}^m f_w(g_θ(z^(i)))]
        w ← w + α · RMSProp(w, g_w)
        w ← clip(w, −c, c)
    end for
    Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
    g_θ ← −∇_θ 1/m Σ_{i=1}^m f_w(g_θ(z^(i)))
    θ ← θ − α · RMSProp(θ, g_θ)
end while
```

$$g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$$

$$g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$$

"Weight clipping is a clearly terrible way to enforce a Lipschitz constraint."

Figure 4: WGAN algorithm ([1] Arjovsky et al, 2017)

Improved WGAN with gradient penalty

Penalize the network if its gradient norm moves away from 1 (the gradient norm has a constant upper bound of 1)

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

```
while θ has not converged do
    for t = 1, ..., n_critic do
        for i = 1, ..., m do
            Sample real data x ~ P_r, latent variable z ~ p(z), a random number ε ~ U[0, 1].
            x̃ ← G_θ(z)
            x̂ ← εx + (1 − ε)x̃
            L^(i) ← D_w(x̃) − D_w(x) + λ(‖∇_x̂ D_w(x̂)‖_2 − 1)^2
        end for
        w ← Adam(∇_w (1/m) Σ_{i=1}^m L^(i), w, α, β_1, β_2)
    end for
    Sample a batch of latent variables {z^(i)}_{i=1}^m ~ p(z).
    θ ← Adam(∇_θ (1/m) Σ_{i=1}^m −D_w(G_θ(z)), θ, α, β_1, β_2)
end while
```

Line 4: Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0,1]$.

$$\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$$

$$\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$$

$$L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$$

$$w \leftarrow \text{Adam}(\nabla_w \tfrac{1}{m}\textstyle\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$$

Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^m \sim p(\boldsymbol{z})$.

$$\theta \leftarrow \text{Adam}(\nabla_\theta \tfrac{1}{m}\textstyle\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$$

"1-Lipschitz is enforced through the penalty on the gradient norm. "

Figure 5: WGAN with gradient penalty([2] Gulrajani et al. 2017.)

Adversarial Training of ASR

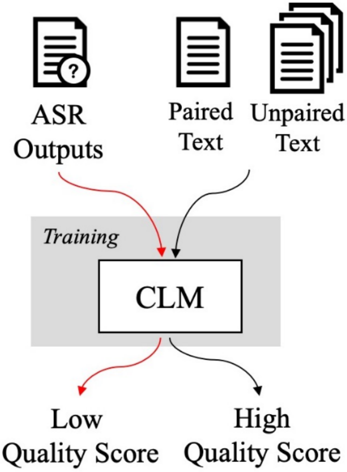
Apply WGAN to ASR

- Motivation:
  - To utilize huge unpaired text data,
  - Language model as discriminator doesn't need to be pre-trained, no extra computation during testing,
- System:
  - (Seq2Seq) Encoder: VGG + BLSTM layers, Decoder: a single LSTM-RNN
  - Seq2Seq with CTC: Connectionist temporal classification (Graves et al., 2006)
  - Criticizing Language Model (CLM)
- Total loss: $L_{ASR} = \lambda_{s2s} L_{s2s} + (1 - \lambda_{s2s}) L_{ctc} - \lambda_{CLM} CLM(\tilde{y})$

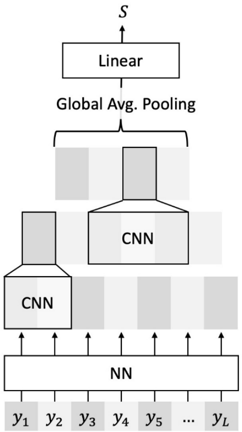
Criticizing Language Model

- Advantage: Real text doesn't have to be paired with audio
- Input: either real text (one hot vectors) or ASR transcriptions (soft distribution vectors)
- WGAN: estimates Wasserstein distance between real data sequence and ASR output
- Loss with gradient penalty:
  $L_{CLM} = \lambda_{CLM} L_D + \lambda_{gp} gp$
  where:
  $L_D = \mathbb{E}_{\tilde{y} \sim P_a}[CLM(\tilde{y})] - \mathbb{E}_{y \sim P_d}[CLM(y)]$
  gradient penatly:
  $gp = \mathbb{E}_{\tilde{y} \sim P_{\tilde{y}}}[(\|\nabla_{\tilde{y}} CLM(\tilde{y})\| - 1)^2]$

CLM Architecture



(a) CLM learning step

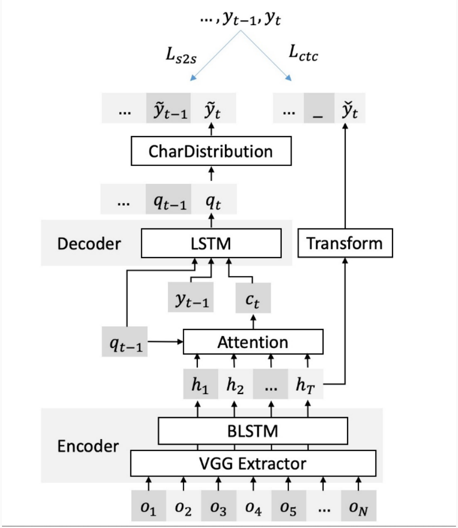


CLM architecture: first CNN with window size 2 and stride 1, second CNN has window size 3 and stride 1

Automatic Speech Recognition System

- Input: sequence of acoustic features
- Downsampling: 6-layer VGG extractor
- Sequence encoder: a 5-layer BLSTM with 320 units per direction, T output sequence length.
- Attention module: 300-dimension allocation-aware attention
- Sequence decoder: a single layer LSTM with 320 units.

ASR Architecture



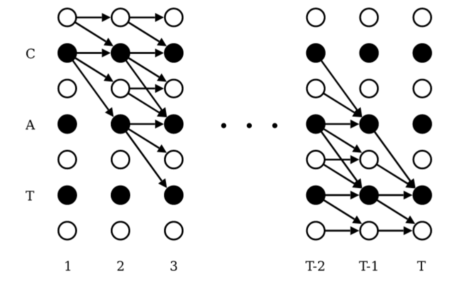
- ASR outputs two character sequences
- Linear Transform:

  "CTC network on top of the encoder and is jointly trained with the attention-based decoder. During the beam search process, we combine the CTC predictions."

  (Hori et al., 2017)

Connectionist Temporal Classification

- Motivation: In case of lacking one-to-one correspondence, to train RNNs to label unsegmented sequences directly (2006)
- Introducing blanks to the original sequences

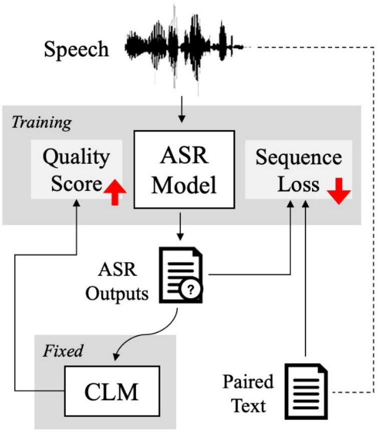


- Independence assumption,
- Left: calculates all possible paths from time step 1 to T,
- Right: unrolls and removes all blanks and duplicates,
- Summarize the probabilities by RNN on the remained paths.

CTC Loss

- y = $y_1, y_2, ...y_T$ is ground truth of O with length T
- inserting blank symbols into y
- obtaining set of all possible sequences $y^{'}$ and $\pi \in y^{'}$ after removing blanks and duplicates
- computing the posterior probability: $P(y|O) = -\Sigma_{\pi \in y^{'}} P(\pi|O)$
- through approximating: $P(\pi|O) \approx \Pi_{t=1}^{T} P_{ctc} (\tilde{y}_t|O)$
- $\tilde{y}_t$ corresponds to the output of the RNN at time step t
- CTC Loss: $L_{ctc} \equiv -logP(y|O)$

ASR learning



(b) ASR model learning step

- Total Loss: $L_{ASR} = \lambda_{s2s} L_{s2s} + (1 - \lambda_{s2s}) L_{ctc} - \lambda_{CLM} CLM(\tilde{y})$
- both ASR and CLM are learned from scratch, no pre-training for CLM
- but during ASR model learning: fix CLM parameters,
- and L_{s2s} and/or L_{ctc} are evaluated with ground truth,
- during testing, drop CLM, and two outputs of ASR are integrated into one sequence.

Experiment

Setup

- Paired data set: LibriSpeech 100 hours of speech and transcriptions, clean
- Unpaired data set: texts from 360 hours clean speech, but 500 hours of noisy speech
- Framework: customize ESPnet toolkit[1] with adversarial training
- Acoustic features: 80-dimensional log Mel-filer bank and 3 dimensional pitch features (Kaldi feature extraction)
- Vocabulary: 5000 subwords
- Hyperparameters: $\lambda_{gp} = 10$, $\lambda_{s2s} = 0.5$, $\lambda_{CLM} = 10^{-4}$

[1] `https://espnet.github.io/espnet/index.html`

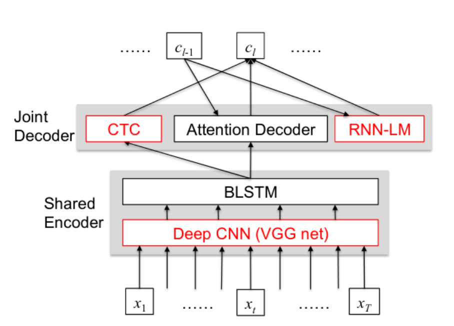
Benchmarks results

Data	Method	CER/WER (%) Dev	CER/WER (%) Test	WER $\Delta^{\dagger}$ Test
(A) w/o unpair text	(a) Baseline	10.5 / 21.6	10.5 / 21.7	-
	(b) +LM	10.9 / 20.0	11.1 / 20.3	6.5%
	(c) +AT	**9.5 / 19.9**	**9.6 / 20.1**	**7.4%**
	(d) +Both	9.4 / 17.9	9.7 / 18.3	15.7%
(B) w/ 360hrs text	(e) +LM	10.5 / 19.6	10.6 / 19.6	9.7%
	(f) +AT	**9.1 / 19.1**	**9.5 / 19.2**	**11.5%**
	(g) +Both	9.0 / 17.1	9.1 / 17.3	20.3%
	(h) BT‡	10.3 / 23.5	10.3 / 23.6	6.3%
	(i) BT+LM‡	9.8 / 21.6	10.0 / 22.0	12.7%
(C) w/ 860hrs text	(j) +LM	9.9 / 18.6	10.2 / 18.8	13.4%
	(k) +AT	**8.6 / 18.5**	**8.8 / 18.7**	**13.8%**
	(l) +Both	7.9 / 15.3	8.2 / 15.8	27.2%

Figure 6: Speech recognition performance. Baseline: plain end-to-end ASR framework, ”+LM” refers to shallow fusion decoding jointly with RNN-LM(Hori et al., 2017), ”+AT” refers to the adversarial training, ”+Both” indicates training with AT and joint decoding with RNN-LM, ”BT” is the prior work of back-translation (Hayashi et al., 2018)

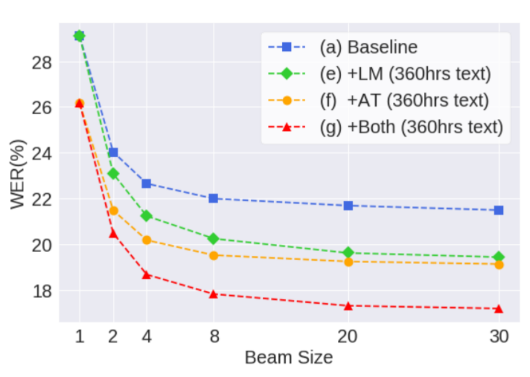
Separately trained RNN-LM

Advances in Joint CTC-Attention based End-to-End Speech Recognition witha Deep CNN Encoder and RNN-LM ((Hori et al., 2017)



- The RNN-LM information is combined at the logits level or pre-softmax.
- Pre-trained RNN-LM or jointly trained with other networks

Varying beam size



- AT consistently improved the performance
- in terms of utilizing extra text data: AT outperformed RNN-LM

References

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou.
"Wasserstein GAN." arXiv preprint arXiv:1701.07875 (2017).

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville.
"Improved training of wasserstein gans." arXiv preprint arXiv:1704.00028 (2017).

Takaaki Hori, Shinji Watanabe, Yu Zhang, and WilliamChan.
"Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm". inInter-speech, 2017.

Alex Graves, Santiago Fern´andez, Faustino Gomez, and J¨urgen Schmidhube
"Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376.

Questions?