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

1. Adversarial Training
2. Adversarial Training of ASR
3. Experiment
4. References
Adversarial Training
Generative Adversarial Networks

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 objective

- 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 to 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

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

$$WGAN : W(D, G) = \frac{1}{K} \sup_{\|f\|_1 \leq K} \mathbb{E}_{x \sim p_r(x)}[D(x)] - \mathbb{E}_{z \sim p_z(z)}[D(G(z))]$$

**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 \( w \in W = [-0.01, 0.01] \)
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: $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size. $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.

Require: $w_0$, initial critic parameters. $\theta_0$, initial generator’s parameters.

1: while $\theta$ has not converged do
2:   for $t = 0, \ldots, n_{\text{critic}}$ do
3:     Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
4:     Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}(z)$ a batch of prior samples.
5:     $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(\theta(z^{(i)})) \right]$
6:     $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
7:     $w \leftarrow \text{clip}(w, -c, c)$
8:   end for
9:   Sample $\{z^{(i)}\}_{i=1}^m \sim \mathbb{P}(z)$ a batch of prior samples.
10:  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(\theta(z^{(i)}))$
11:  $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while

"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 $\lambda$, the number of critic iterations per generator iteration $n_{\text{critic}}$, the batch size $m$, Adam hyperparameters $\alpha, \beta_1, \beta_2$.

**Require:** initial critic parameters $w_0$, initial generator parameters $\theta_0$.

1: while $\theta$ has not converged do
2:   for $t = 1, \ldots, n_{\text{critic}}$ do
3:     for $i = 1, \ldots, m$ do
4:         Sample real data $x \sim \mathbb{P}_r$, latent variable $z \sim p(z)$, a random number $\epsilon \sim U[0, 1]$.
5:         $\tilde{x} \leftarrow G_{\theta}(z)$
6:         $\tilde{x} \leftarrow \epsilon x + (1 - \epsilon) \tilde{x}$
7:         $L^{(i)} \leftarrow D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{\tilde{x}} D_w(\tilde{x})\|_2 - 1)^2$
8:     end for
9:     $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
10:    end for
11:   Sample a batch of latent variables \( \{z^{(i)}\}_{i=1}^{m} \sim p(z) \).
12:   $\theta \leftarrow \text{Adam}(\nabla_{\theta} \frac{1}{m} \sum_{i=1}^{m} -D_w(G_{\theta}(z)), \theta, \alpha, \beta_1, \beta_2)$
13: end while

"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
• 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: \( L_{ASR} = \lambda_{s2s} L_{s2s} + (1 - \lambda_{s2s}) L_{ctc} - \lambda_{CLM} \text{CLM}(\tilde{y}) \)
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:
  \[ L_D = \mathbb{E}_{\tilde{y} \sim P_d}[CLM(\tilde{y})] - \mathbb{E}_y \sim P_d[CLM(y)] \]
  gradient penalty:
  \[ gp = \mathbb{E}_{\tilde{y} \sim P_{\tilde{y}}}[((\| \nabla_{\tilde{y}} CLM(\tilde{y}) \| - 1)^2] \]
CLM Architecture

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

![Diagram showing the flow of data through time steps, with arrows indicating the transitions between time steps.]

- 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

- \( y = y_1, y_2, \ldots y_T \) 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: \( P(y|O) = -\sum_{\pi \in y'} P(\pi|O) \)
- through approximating: \( P(\pi|O) \approx \prod_{t=1}^{T} P_{ctc}(\tilde{y}_t|O) \)
- \( \tilde{y}_t \) corresponds to the output of the RNN at time step \( t \)
- CTC Loss: \( L_{ctc} \equiv -\log P(y|O) \)
ASR learning

• 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

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)

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

Advances in Joint CTC-Attention based End-to-End Speech Recognition with a 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


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

Questions?