1. Introduction

- Sequence-to-sequence ASR is constructed by an encoder + an (autoregressive) decoder
- The autoregressive decoder is trained with teacher-forcing method
- **Problem:**
  - Generation: training (groundtruth) != inference (own model prediction)
  - Objective: maximum likelihood estimation (MLE) != task-specific metric (WER, CER)
- Proposed an alternative optimization for seq2seq ASR via policy gradient

2. Seq2Seq ASR

- $x = [x_1, \ldots, x_T]$ (speech feature)
- $y = [y_{1}, \ldots, y_{T}]$ (text)
- $h^{E}_{t}$ = encoder states
- $h^{D}_{t}$ = decoder states
- $a_{t}$ = attention probability
- $c_{t}$ = weighted context vector
- Encoder: 1 FC + 3 Bi-LSTM 256x2
- Decoder: 1 LSTM 512
- Attention: MLP scorer

**Teacher Forcing:**

$$M_{ELE}(y_{1:n}, p(y_{1:n})) = M_{ELE}(y_{1:n}, p(y_{1:n}))$$

$$\sum_{t=1}^{T} y_{t} = c \Rightarrow \sum_{t=1}^{T} p(y_{t}) = c$$

3. Optimization with RL

- **Policy gradient:** RL algorithm for optimizing the expected rewards w.r.t a policy $\pi_{\theta}$
- Given a pair speech-text $(x(n), y(n))$, $R^{(n)}$ is the reward based on edit-distance (ED) between predicted $y$ and ground-truth $y^{(n)}$

\[
R^{(n)} = -ED(y, y^{(n)})
\]

\[
V_{\theta}E_{\gamma}[R^{n}|\pi_{\theta}] = E_{\gamma}\left[ V_{\theta} \log P(y|x^{(n)}; \theta) R^{(n)} \right]
\]

1. Final reward (f-reward):

$$V_{\theta}E_{\gamma}[R^{n}|\pi_{\theta}] = E_{\gamma}\left[ V_{\theta} \log P(y|x^{(n)}; \theta) R^{(n)} \right]$$

2. Time distributed reward (t-reward):

$$r_{t}^{n} = \begin{cases} 
- \left( ED(y_{1:t}, y^{(n)}) - ED(y_{1:t-1}, y^{(n)}) \right) & \text{if } t > 1 \\
- \left( ED(y_{1:t}, y^{(n)}) - |y^{(n)}| \right) & \text{if } t = 1 
\end{cases}$$

$$R^{(n)} = \sum_{t=1}^{T} y^{(n)} - r_{t}^{n}$$

$$V_{\theta}E_{\gamma}\left[ \sum_{t=1}^{T} r_{t}^{n} |\pi_{\theta} \right] = E_{\gamma}\left[ \sum_{t=1}^{T} R_{t}^{(n)} V_{\theta} \log P(y_{1:t}, x^{n}; \theta) \right]$$

4. Experimental Setup & Results

- Features: log Mel-spec (40-dim + $\Delta$ + $\Delta^{2}$)
- Text: 26 letters (A-Z)+’(‘)+’<noisy’+’<eos’)
- Dataset: Wall Street Journal
- Train: train_si84 (WSJ0) & train_si284 (WSJ1)
- Dev: dev93 & Test: eval92
- RL hyperparameters:
  - discount factor $\gamma = \{0, 0.5, 0.95\}$

<table>
<thead>
<tr>
<th>Models</th>
<th>Result (CER%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTC</td>
<td>8.97</td>
</tr>
<tr>
<td>Att Enc-Dec</td>
<td>7.69</td>
</tr>
<tr>
<td>MLE+RL</td>
<td>7.26</td>
</tr>
<tr>
<td>Att Enc-Dec + RL (f-reward $R$)</td>
<td>6.64</td>
</tr>
<tr>
<td>Att Enc-Dec + RL (t-reward $R_t$, $\gamma = 0$)</td>
<td>6.37</td>
</tr>
<tr>
<td>Att Enc-Dec + RL (t-reward $R_t$, $\gamma = 0.5$)</td>
<td>6.10</td>
</tr>
</tbody>
</table>

5. Conclusion

- By treating our decoder as a policy network, we can:
  1. Sample the whole transcription based on model’s prediction in the training process
  2. Directly optimize the model with negative Levenshtein distance as the reward
- Our experiment shows by combining RL + MLE, we significantly improve the performance
- **Best combination:** MLE+RL time-distributed reward