A Novel Learnable Dictionary Encoding Layer for End-to-End Language Identification

Weicheng Cai\(^1\), Zexin Cai\(^1\), Xiang Zhang\(^3\), Xiaoxi Wang\(^4\) and Ming Li\(^1,2\)

1. School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China
2. Data Science Research Center, Duke Kunshan University, Kunshan, China
3. Tencent Inc., Beijing, China
4. Jiangsu Jinling Science and Technology Group Limited, Nanjing, China

m3442@duke.edu

Introduction

In recent decades, in order to get the utterance level vector representation, dictionary learning procedure is widely used.

A dictionary, which contains several temporal orderless center components (or units, words, clusters), can encode the variable-length input sequence into a single utterance level vector representation.

### Dictionary Learning

- **QV codebook (K-means)**
- **UBM (GMM)**
- **Phoneme decoder (DNN)**
- **Phonetic tokenizer (GMM / DNN)**

### Vector Encoding

- **Average Quantization Distortion (GMM Skelldow)**
- **GMM Superc VECTOR (GMM I-vector)**
- **Bay-of-words, H-gram taken statistics**

### LDE Intuition

- **Motivated by GMM Supervector encoding procedure**, we design a learnable dictionary encoding (LDE) layer on top of front-end CNN. The LDE layer simultaneously learns the encoding parameters along with an inherent dictionary in a fully supervised manner.

- The inherent dictionary is learned from the distribution of the descriptors by passing the gradient through assignment weights. During the training process, the updating of extracted convolutional features can also benefit from the encoding representations.

### Experimental Results and Discussion

- **The task of interest is the closed-set language detection. There are 34 target languages in testing corpus, which included 7510 utterances split among three nominal durations: 30, 10 and 1 seconds.**

- **In order to get higher abstract representation better for utterances with long duration, we design a deep CNN based on the well known ResNet-18 architecture, as is described in Table 2. The total parameters of the CNN-LDE system are 6.5M.**

- **For CNN-TAP system, a simple average pooling layer followed by FC layer is built on top of the front-end CNN. For CNN-LDE system, the average pooling layer is replaced with a LDE layer.**

- **Because we have not separated validation set, even, we only use the converged model after the last step of optimization. For each training step, an integer \( i \) (within [20, 100]) is randomly generated, and each data in the mini-batch is cropped or extended to 2 frames.**

- **In testing stage, all the 3s, 10s, and 30s duration data is tested on testing speech utterance to the trained neural network one by one.**

- **The CNN-LDE system outperforms the CNN-TAP system with all different number of dictionary components.**

- **When the numbers of dictionary component increased from 16 to 64, the performance improved significantly especially with regard to EER.**

- **The CNN-LDE system achieves significant performance improvement especially with regard to EER.**

### LDE Implementation

- The LDE layer is a directed acyclic graph and all the components are different with the input \( \mathbf{x} = \{x_1, \ldots, x_n\} \) and the learnable parameters. Given a set of frames feature sequence and a learned dictionary center \( \mathbf{c} = \{c_1, \ldots, c_n\} \) each frame of feature \( x_i \) can be assigned with a weight to each component \( w_i \), and the corresponding residual vector is denoted by

\[
\mathbf{r}_i = x_i - \mathbf{c}_i \cdot w_i.
\]

- The non-negative assigning weight is given by a softmax function:

\[
\mathbf{w}_i = \frac{e^{\mathbf{r}_i' \mathbf{w}}}{\sum_{i=1}^{n} e^{\mathbf{r}_i' \mathbf{w}}}
\]

- The LDE layer concatenates the aggregated residual vectors with assigned weights. The resulting encoder outputs a fixed dimensional representation

\[
\mathbf{E} = \{e_1, \ldots, e_n\}
\]

### Table 1

<table>
<thead>
<tr>
<th>System ID</th>
<th>System Description</th>
<th>Feature</th>
<th>Encoding Method</th>
<th>EER(%)</th>
<th>HTER(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>CNN-TAP</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>6.94</td>
<td>2.36</td>
</tr>
<tr>
<td>3</td>
<td>CNN-LDE(^{1+16})</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>6.91</td>
<td>2.37</td>
</tr>
<tr>
<td>4</td>
<td>CNN-LDE(^{32})</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>6.70</td>
<td>2.45</td>
</tr>
<tr>
<td>5</td>
<td>CNN-LDE(^{64})</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>6.25</td>
<td>2.54</td>
</tr>
<tr>
<td>6</td>
<td>CNN-LDE(^{128})</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>5.66</td>
<td>2.49</td>
</tr>
<tr>
<td>7</td>
<td>CNN-LDE(^{256})</td>
<td>CNN FeatureMaps</td>
<td>LDE</td>
<td>4.77</td>
<td>2.54</td>
</tr>
<tr>
<td>8</td>
<td>Forum 102 / 185</td>
<td>Forum FeatureMaps</td>
<td>LDE</td>
<td>4.40</td>
<td>2.47</td>
</tr>
</tbody>
</table>

**The CNN-LDE system outperforms the CNN-TAP system with all different number of dictionary components.**