Rectified Linear Neural Networks with Tied-Scalar Regularization for LVCSR

Shiliang Zhang\textsuperscript{1}, Hui Jiang\textsuperscript{2}, Si Wei\textsuperscript{1}, Li-Rong Dai\textsuperscript{1}

\textsuperscript{1}National Engineering Laboratory for Speech and Language Information Processing University of Science and Technology of China, Hefei, Anhui, P. R. China
\textsuperscript{2}Department of Electrical Engineering and Computer Science York University, 4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada

\texttt{zsl2008@mail.ustc.edu.cn, hj@cse.yorku.ca, siwei@iflytek.com, lrdai@ustc.edu.cn}

\section*{Abstract}
It is known that rectified linear deep neural networks (RL-DNNs) can consistently outperform the conventional pre-trained sigmoid DNNs even with a random initialization. In this paper, we present another interesting and useful property of RL-DNNs that we can learn RL-DNNs with a very large batch size in stochastic gradient descent (SGD). Therefore, the SGD learning can be easily parallelized among multiple computing units for much better training efficiency. Moreover, we also propose a tied-scalar regularization technique to make the large-batch SGD learning of RL-DNNs more stable. Experimental results on the 309-hour Switchboard (SWB) task have shown that we can train RL-DNNs using batch sizes about 100 times larger than those used in the previous work, thus the learning of RL-DNNs can be accelerated by over 10 times when 8 GPUs are used. More importantly, we have achieved a word error rate of 13.8\% with a 6-hidden-layer RL-DNN trained by the frame-level cross-entropy criterion with the tied-scalar regularization. To our knowledge, this is the best reported performance on this task under the same experimental settings.

\textbf{Index Terms}: rectified linear units, RL-DNN, LVCSR, tied-scalar regularization

\section{1. Introduction}
Recently, neural networks have revived as a popular model in machine learning under the name of deep learning. Deep learning aims at learning neural networks with very deep architecture, such as deep neural networks (DNNs), which significantly outperform other machine learning methods and yield the state of the art performance in many real-world applications, such as speech recognition, computer vision and many others.

As the basic building block of DNNs, each neuron (hidden node) imposes a nonlinear activation function from its input to output. Traditionally, the standard logistic sigmoid and hyperbolic tangent functions are widely used in the neural networks literature. More recently, in [1, 2, 3], it has proposed to use a simpler nonlinear neuron, namely rectified linear unit (ReLU), which takes a rectified linear function as its activation function from input to output. In many large scale real-world applications [4, 5, 6, 7, 8], ReLU based DNNs (RL-DNN) have demonstrated several major advantages over the traditional DNNs using logistic sigmoid or hyperbolic functions [5]. Firstly, RL-DNNs normally yield better recognition performance than the regular sigmoid DNNs. Secondly, the training speed of RL-DNNs is much faster than that of the regular sigmoid DNNs because the learning process of RL-DNNs seems to converge faster. Next, rectified linear units generate exact zeros when inputs are not aligned with the internal weights in the model, as opposed to the sigmoid units that produce small noisy values. As a result, a learned RL-DNN is much sparser than its counterpart using logistic sigmoid units. The increased sparsity is believed to improve model generalization [3]. Moreover, based on the experimental observations in [4, 5], it has been widely conjectured that RL-DNNs are much easier to learn since the piece-wise linear property arising from ReLUs may be beneficial from the optimization perspectives.

As we know, the learning objective functions of DNNs are always highly non-convex, where many distinct local minima co-exist in the model parameter space. Therefore, the stochastic gradient descent (SGD) algorithm [9, 10, 11] is usually used to optimize the non-convex objective function to learn DNNs, which is believed to help the learning to escape from poor local optima. Unfortunately, using small mini-batches in SGD becomes a major computation bottleneck to parallelize the DNN learning among multiple GPUs or to distribute the learning to large CPU clusters.

In this paper, we have observed an interesting and useful property of RL-DNNs that we can learn RL-DNNs reliably using a very large mini-batch size, even up to 100k (102400), which is about 100 times larger than those used in the previous work. Therefore, it can largely overcome the major hurdle in the parallel training of DNNs since the learning can be easily parallelized by splitting each large “mini-batch” into multiple computing units for much faster training turnaround. As shown in our experiments on the Switchboard task, we can dramatically speedup the RL-DNN training by more than 10 times when using 8 GPUs in the parallel training. In our large mini-batch SGD training, we still need to carefully tune the learning rate as usual for the best possible performance. In this paper, we first give a heuristic but useful strategy to adjust the learning rates according to the used mini-batch size. More importantly, in order to make the large mini-batch training of RL-DNNs reliable and effective, we have proposed a new tied-scalar regularization technique, which can yield a significant performance improvement over the conventional methods. For example, we have achieved a word error rate of 13.8\% on the 309-hour Switchboard (SWB) task with a 6-hidden layer RL-DNN that takes filter bank features (FBK) as input and is trained with the frame-level cross-entropy criterion, which is about 11.5\% relative performance improvement over the conventional pre-trained sigmoid DNNs. To the best of our knowledge, this is the best reported perfor-

This work was partially supported by the National Nature Science Foundation of China (Grant No. 61273264) and the electronic information industry development fund of China (Grant No. 2013-472).
performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X$, the performance on this task under comparable experimental settings.

2. Preliminaries: RL-DNNs

The structure of DNNs is a conventional multi-layer perceptron with many hidden layers. Given an observation vector $X
where \( e^{\ell} \) denotes for the error signal computed for the \( j \)-th hidden node in the layer \( \ell \), and \( N_{\ell} \) denotes the number of the hidden units in \( \ell \)-th layer. As for the initialization, we use the maximum \( L_2 \) norm of all (randomly) initial weight vectors in each layer to initialize all \( a^\ell \), and all weight vectors in each layer are normal-
ized by \( \alpha \) to satisfy the norm constraints: \( \| w_k^\ell \| \leq 1 \ (\forall k, \ell) \) (9).

During the SGD learning, we need to constrain the \( L_2 \) norm of each weight vector if its norm exceeds 1 after each update:

\[
\| w_k^\ell \| \leftarrow \frac{w_k^\ell}{\| w_k^\ell \|} \quad \text{if} \quad \| w_k^\ell \| > 1
\]

The proposed tied-scalar regularization can prevent all DNN weights from growing too large, which makes it suitable for learning RL-DNNs with a very large initial learning rate for those larger mini-batch sizes.

### 4. Experiments

In this paper, we have examine how to training RL-DNNs with large mini-batch sizes and further investigate the proposed tied-
scalar regularization on the Switchboard (SWB) database. The
SWB training data consists of 309-hour Switchboard-I training
database and 20-hour Call Home English data. We divide the whole training data into two sets: training set and cross-
validation set. The training set contains 99.5% training data, and the cross-validation set contains the remaining 0.5%. Evalu-
ation is performed in terms of word error rate (WER) on the
NIST 2000 Hub5 evaluation set (containing 1831 utterances),
denoted as Hub5e00.

#### 4.1. Baseline systems (GMM/HMM and sigmoid DNNs)

The GMM/HMM baseline is a standard tied-state cross-word
tri-phone system, using the 39-dimension PLP features (static,
first and second derivatives) as feature and Gaussian mixture
models (GMM) as acoustic model, which is estimated with
the maximum likelihood estimation (MLE) and then discrimina-
tively trained based on the minimum phone error (MPE) cri-
terion [16]. Before the model training, all feature vectors are
pre-processed with the cepstral mean and variance normalization (CMVN) per conversation side. The final hidden Markov
model (HMM) consists of 8,991 tied states and 40 Gaussian
components per state. In the decoding, we use a trigram lan-
guage model (LM) that is trained by using 3 million words
from the training transcripts and another 11 million words of
the Fisher English Part 1 transcripts. The performance of the
baseline GMM/HMMs systems is listed in Table 1.

For the baseline DNN systems, we use the logistic sigmoid
function as the activation function for all hidden nodes. We
follow the same training procedure as described in [17, 18, 19, 20]
to train the conventional context dependent DNN/HMM with the
tied-triphone state alignment obtained from the above MLE
tained GMM/HMMs baseline system. The input vectors are
either 39-dimension PLP or 123-dimension mel-warped filter-
bank (FBK) features concatenated from all consecutive frames
within a long context window of 5(4+5). The sigmoid DNN
is first pre-trained using the RBM-based layer-wise pre-training
and then fine-tuned with 10 epochs of frame-level cross-entropy
(CE) training. In the fine-tuning, we used SGD with mini-
batches of 1024 (1k) frames. The initial learning rate is set to
0.2. We have trained sigmoid DNNs with 5 and 6 hidden layers
and 2,048 nodes per layer. The performance of these baseline
DNN/HMMs systems is also listed in Table 1 for comparison.

#### 4.2. RL-DNNs

We train RL-DNNs (with 5 or 6 hidden layers of 2,048 ReLU
nodes per layer) using the conventional back-propagation (BP)
with varying batch sizes, being 1k (1024), 10k (10240), 20k
(20480), 50k (51200) and 100k (102400). The learning pa-
rameters of RL-DNNs are similar to those of sigmoid DNNs.
The learning schedule of RL-DNNs is described in section 2.1.
Experimental results in Table 2 show that we can still achieve
very competitive recognition performance when we use much
larger mini-batch sizes to train RL-DNNs. For instance, the
recognition performance of the 6-hidden-layer RL-DNN only
changes from 15.0% and 15.5% when the mini-batch size is
increased from 1k to 100k. Moreover, the learning curves in
Figure 1 also indicate that the learning of RL-DNNs converges
very well for a wide range of mini-batch sizes. Particularly, for
a 6-hidden-layer RL-DNN trained using a batch size of 10k, we
have achieved 15.0% in WER, a 1.2% absolute error reduction
over the baseline pre-trained sigmoid DNNs in Table 1.

Another advantage of using large batch size is that the training
can be easily parallelized among multiple computing units to
significantly improve training efficiency. In our experiments,
we have implemented a parallel scheme to train a 6-hidden-
layer RL-DNN with a mini-batch size of 8k using a system of 4
GPUs (NVIDIA 750 Tesla K20). It has shown that the training
speed can be expedited by about 3 times by simply distrib-

<table>
<thead>
<tr>
<th>model</th>
<th>method</th>
<th>Hub5e00</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM-HMM</td>
<td>MLE</td>
<td>28.7</td>
</tr>
<tr>
<td>DNN-HMM</td>
<td>PLP</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>FBK</td>
<td>16.1</td>
</tr>
</tbody>
</table>

Table 1: Word error rates (WER in %) of various baseline systems in Switchboard.
Table 2: Word error rates (WER in %) in the Switchboard of RL-DNNs learned using PLP features with various batch sizes and training speedup factors when 8 GPUs are used in parallel.

<table>
<thead>
<tr>
<th>batch size</th>
<th>5*2048</th>
<th>6*2048</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WER(%) speedup</td>
<td>WER(%) speedup</td>
</tr>
<tr>
<td>1k</td>
<td>15.9 -</td>
<td>15.4 -</td>
</tr>
<tr>
<td>10k</td>
<td>15.6 x5.4</td>
<td>15.0 x5.4</td>
</tr>
<tr>
<td>20k</td>
<td>15.5 x8.2</td>
<td>15.3 x8.4</td>
</tr>
<tr>
<td>50k</td>
<td>15.7 x11.2</td>
<td>15.3 x11.7</td>
</tr>
<tr>
<td>100k</td>
<td>16.0 x12.2</td>
<td>15.5 x12.7</td>
</tr>
</tbody>
</table>

Figure 1: Comparison of various learning curves of sigmoid DNNs and RL-DNNs (6 hidden layers of 2048 neurons) on the Switchboard task using different batch sizes (1k, 10k and 100k); The left figure is for training data and the right figure for the 0.5% held-out development set.

4.3. RL-DNNs with tied-scalar regularization

We further investigate the performance of RL-DNNs with the tied-scalar regularization. Here we have trained RL-DNNs using either PLP or FBK features. The training scheme of RL-DNNs with tied-scalars is similar to that of RL-DNNs except that we use a different learning rate for all tied-scalars, which is set to 0.002 in our experiments. The learning rate of the tied-scalars is kept fixed during the training. The advantage of using tied-scalars is that we can train RL-DNNs with a large batch size more reliably. We just need to set an initial learning rate as eq. (6). The learning curves of the tied-scalars of all hidden layers are plotted in Figure 2, which indicates that these tied-scalars gradually increase during the learning and can eventually converge very well at the end. In this experiment, we can see that the tied-scalars in all hidden layers converge to 1.4 while those in the input and output layers converge to 1.2. Compared to the \( L_2 \) norm regularization in [15], the proposed tied-scalar regularization can automatically learn the optimal upper bound for the \( L_2 \) norm. Experimental results in Table 3 show that the proposed tied-scalar regularization can further improve the performance of RL-DNNs significantly. For example, we have achieved a word error rate of 15.1% by using PLP feature for a 5-hidden-layer RL-DNN with tied-scalar while the baseline system in Table 1 is 16.5%. Moreover, for a 6-hidden-layer RL-DNN with tied-scalars (use FBK as input) trained using a batch size of 10k, we have achieved 13.8% in WER. To our knowledge, this is the best reported performance on this task for the speaker-independent training (without speaker-specific adaptation and normalization as in [23, 24]) using the frame-level cross-entropy error criterion.

5. Conclusions

In this work, we have presented an empirical study to show that RL-DNNs can be effectively learned with a very large mini-batch size and have further proposed a tied-scalar regularization method to make the learning of RL-DNNs with large batch sizes more reliable. In this way, the training of RL-DNNs can be easily parallelized using multiple GPUs for better efficiency. Our experiments on the Switchboard task have shown that we can speedup the learning by more than 10 times using 8 GPUs, meanwhile it can still yield very competitive performance. For example, we have achieved a WER of 13.8% with a 6-hidden-layer RL-DNN with tied-scalars trained using a mini-batch size of 10k, which is the best reported performance in this case.
6. References


