Evaluation of Random Neural Layers in Deep Neural Networks
Corentin Hardy, Erwan Le Merrer, Gerardo Rubino, Bruno Sericola

To cite this version:

HAL Id: hal-01657644
https://hal.inria.fr/hal-01657644
Submitted on 7 Dec 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Evaluation of Random Neural Layers in Deep Neural Networks

Corentin Hardy, Erwan Le Merrer, Gerardo Rubino, Bruno Sericola

Queuing system as Neural Network

Random Neural Networks (RNN) are proposed in 1989 by Gelenbe. A RNN is a queuing system.

- Neurons are seen as queues receiving positive and negative customers:
  - when a positive customer (labeled +) enters in the queue at time t, it increases the queue size $N_t$ by one,
  - when a negative customer (labeled +) enters in the queue at time t, it deletes one positive customer in the queue, if any, decreasing the queue size $N_t$ by one,
- The first positive customer of the queue leaves the queue every $1/\mu$ in average.
- When a customer leaves a queue, it decreases the queue size by one.
- The activation value of a neuron is the probability that $N_t>0$ in steady state (ie., $t \to \infty$).

Gelenbe shows that it is possible to compute the activation values of neurons of a RNN. He introduces weight connections between queues which are tunable to train the RNN for a task with a Gradient Descent as in a classical Neural Network.

Random Neural Layer*

Inspired by Random Neural Networks, we introduce Random Neural Layer (RNL), which is compatible with other kinds of neural layers in Deep Neural Networks (DNN). The RNL takes as input a vector $X \in \mathbb{R}^m$. The RNL is composed of:
- A matrix of parameters $W^+ \in \mathbb{R}^{n \times m}$
- A matrix of parameters $W^- \in \mathbb{R}^{n \times m}$
- A vector of constants $M \in \mathbb{R}^n$

The activation values of the RNL is a vector $A \in \mathbb{R}^n$ computed such as:

$$A = N / D$$

where

$$N = W^+ X$$
$$D = W^- X + M$$

$N_i$ represents the mean throughput of positive customers of neuron $i$. $D_i$ is the mean throughput of negative customers plus the mean service rate $\mu$.

Recall on the classical fully-connected layers (FCL)

The activation value of a FCL with an input $X \in \mathbb{R}^m$ is:

$$A = F(WX + \beta)$$

where $\beta \in \mathbb{R}^n$ is the bias vector, $W \in \mathbb{R}^{n \times m}$ is the matrix of weights, and $F$ is the activation function.

Experiments

We experiment RNLs in classical architectures of DNNs (taken from Keras). We run different combinations by mixing classical FCL and RNL. In CNN models, convolutional layers are not changed.

<table>
<thead>
<tr>
<th>1st layer</th>
<th>2nd layer</th>
<th>3rd layer</th>
<th>Final accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNL</td>
<td>RNL</td>
<td>RNL</td>
<td>96.39%</td>
</tr>
<tr>
<td>RNL</td>
<td>RNL</td>
<td>FCL</td>
<td>97.16%</td>
</tr>
<tr>
<td>RNL</td>
<td>FCL</td>
<td>RNL</td>
<td>97.22%</td>
</tr>
<tr>
<td>RNL</td>
<td>FCL</td>
<td>RCL</td>
<td>97.31%</td>
</tr>
<tr>
<td>FCL</td>
<td>RNL</td>
<td>RNL</td>
<td>94.78%</td>
</tr>
<tr>
<td>FCL</td>
<td>FCL</td>
<td>RCL</td>
<td>98.06%</td>
</tr>
<tr>
<td>FCL</td>
<td>FCL</td>
<td>RCL</td>
<td>98.35%</td>
</tr>
<tr>
<td>FCL</td>
<td>FCL</td>
<td>FCL</td>
<td>98.31%</td>
</tr>
</tbody>
</table>

Final accuracy of a CNN on MNIST

<table>
<thead>
<tr>
<th>1st layer</th>
<th>2nd layer</th>
<th>3rd layer</th>
<th>4th layer</th>
<th>Final accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN</td>
<td>CNN</td>
<td>CNN</td>
<td>CNN</td>
<td>98.12%</td>
</tr>
<tr>
<td>CNN</td>
<td>CNN</td>
<td>RNL</td>
<td>FCL</td>
<td>98.68%</td>
</tr>
<tr>
<td>CNN</td>
<td>CNN</td>
<td>FCL</td>
<td>RNL</td>
<td>98.96%</td>
</tr>
<tr>
<td>CNN</td>
<td>CNN</td>
<td>FCL</td>
<td>FCL</td>
<td>99.03%</td>
</tr>
</tbody>
</table>

Take away message

DNNs containing a mix of classical neural layers and RNLs may obtain similar or better results, especially when the RNL is the last layer.

*code of RNL is available in https://github.com/Hardy-c/DNN-with-RNL