HAL
open science

Evaluation of Random Neural Layers in Deep Neural Networks

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

To cite this version:

Corentin Hardy, Erwan Le Merrer, Gerardo Rubino, Bruno Sericola. Evaluation of Random Neural Layers in Deep Neural Networks. NIPS 2017 - workshop Deep Learning: Bridging Theory and Practice, Dec 2017, Long Beach, United States. pp.1, 2017. hal-01657644

HAL Id: hal-01657644

https://inria.hal.science/hal-01657644v1

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**.

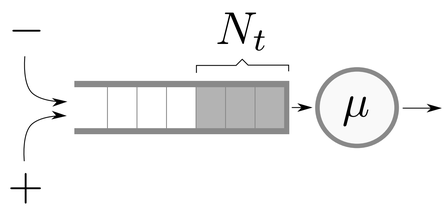

Fig 1 : A queue as Neuron

- 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 $X \in R^m$. The RNL is composed of :

- A matrix of parameters $W^+ \in R_+^{n \times m}$
- A matrix of parameters $W^- \in R_+^{n \times m}$
- A vector of constants $M \in R_+^n$

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

$$A = N/D \qquad \text{(element-wise division)}$$

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 μ.

Recall on the classical fully-connected layers (FCL)

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

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

where $\beta \in R^n$ is the bias vector, $W \in R^{n \times m}$ 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, convolutionnal layers are not changed.

1st layer	2nd layer	3rd layer	Final accuracy
RNL	RNL	RNL	96.39%
RNL	RNL	FCL	97.16%
RNL	FCL	RNL	97.22%
RNL	FCL	FCL	97.31%
FCL	RNL	RNL	94.78%
FCL	RNL	FCL	98.06%
FCL	FCL	RNL	**98.35%**
FCL	FCL	FCL	98.31%

Final accuracy of a MLP model on MNIST

1st layer	2nd layer	3rd layer	4th layer	Final accuracy
CNN	CNN	RNL	RNL	98.12%
CNN	CNN	RNL	FCL	98.68%
CNN	CNN	FCL	RNL	98.96%
CNN	CNN	FCL	FCL	**99.03%**

Final accuracy of a CNN on MNIST

1^{rt} layer	2nd layer	3rd layer	4th layer	5th layer	6th layer	Final accuracy
CNN	CNN	CNN	CNN	RNL	RNL	75.57%
CNN	CNN	CNN	CNN	RNL	FCL	50.83%
CNN	CNN	CNN	CNN	FCL	RNL	**83.92%**
CNN	CNN	CNN	CNN	FCL	FCL	81.17%

Final accuracy of a CNN on CIFAR10

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