Self-Normalizing Neural Networks

Günter Klambauer **Thomas Unterthiner** **Andreas Mayr**

Sepp Hochreiter
LIT AI Lab & Institute of Bioinformatics,
Johannes Kepler University Linz
A-4040 Linz, Austria
`{klambauer,unterthiner,mayr,hochreit}@bioinf.jku.at`

Abstract

Deep Learning has revolutionized vision via convolutional neural networks (CNNs) and natural language processing via recurrent neural networks (RNNs). However, success stories of Deep Learning with standard feed-forward neural networks (FNNs) are rare. FNNs that perform well are typically shallow and, therefore cannot exploit many levels of abstract representations. We introduce self-normalizing neural networks (SNNs) to enable high-level abstract representations. While batch normalization requires explicit normalization, neuron activations of SNNs automatically converge towards zero mean and unit variance. The activation function of SNNs are "scaled exponential linear units" (SELUs), which induce self-normalizing properties. Using the Banach fixed-point theorem, we prove that activations close to zero mean and unit variance that are propagated through many network layers will converge towards zero mean and unit variance — even under the presence of noise and perturbations. This convergence property of SNNs allows to (1) train deep networks with many layers, (2) employ strong regularization schemes, and (3) to make learning highly robust. Furthermore, for activations not close to unit variance, we prove an upper and lower bound on the variance, thus, vanishing and exploding gradients are impossible. We compared SNNs on (a) 121 tasks from the UCI machine learning repository, on (b) drug discovery benchmarks, and on (c) astronomy tasks with standard FNNs, and other machine learning methods such as random forests and support vector machines. For FNNs we considered (i) ReLU networks without normalization, (ii) batch normalization, (iii) layer normalization, (iv) weight normalization, (v) highway networks, and (vi) residual networks. SNNs significantly outperformed all competing FNN methods at 121 UCI tasks, outperformed all competing methods at the Tox21 dataset, and set a new record at an astronomy data set. The winning SNN architectures are often very deep. Implementations are available at: github.com/bioinf-jku/SNNs.

Accepted for publication at NIPS 2017; please cite as:
`Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-Normalizing Neural Networks. In Advances in Neural Information Processing Systems (NIPS).`

Introduction

Deep Learning has set new records at different benchmarks and led to various commercial applications [25, 33]. Recurrent neural networks (RNNs) [18] achieved new levels at speech and natural language processing, for example at the TIMIT benchmark [12] or at language translation [36], and are already employed in mobile devices [31]. RNNs have won handwriting recognition challenges (Chinese and Arabic handwriting) [33, 13, 6] and Kaggle challenges, such as the "Grasp-and Lift EEG" competition. Their counterparts, convolutional neural networks (CNNs) [24] excel at vision and video tasks. CNNs are on par with human dermatologists at the visual detection of skin cancer [9]. The visual processing for self-driving cars is based on CNNs [19], as is the visual input to AlphaGo which has beaten one of the best human GO players [34]. At vision challenges, CNNs are constantly winning, for example at the large ImageNet competition [23, 16], but also almost all Kaggle vision challenges, such as the "Diabetic Retinopathy" and the "Right Whale" challenges [8, 14].

However, looking at Kaggle challenges that are not related to vision or sequential tasks, gradient boosting, random forests, or support vector machines (SVMs) are winning most of the competitions. Deep Learning is notably absent, and for the few cases where FNNs won, they are shallow. For example, the HIGGS challenge, the Merck Molecular Activity challenge, and the Tox21 Data challenge were all won by FNNs with at most four hidden layers. Surprisingly, it is hard to find success stories with FNNs that have many hidden layers, though they would allow for different levels of abstract representations of the input [3].

To robustly train very deep CNNs, batch normalization evolved into a standard to normalize neuron activations to zero mean and unit variance [20]. Layer normalization [2] also ensures zero mean and unit variance, while weight normalization [32] ensures zero mean and unit variance if in the previous layer the activations have zero mean and unit variance. However, training with normalization techniques is perturbed by stochastic gradient descent (SGD), stochastic regularization (like dropout), and the estimation of the normalization parameters. Both RNNs and CNNs can stabilize learning via weight sharing, therefore they are less prone to these perturbations. In contrast, FNNs trained with normalization techniques suffer from these perturbations and have high variance in the training error (see Figure 1). This high variance hinders learning and slows it down. Furthermore, strong regularization, such as dropout, is not possible as it would further increase the variance which in turn would lead to divergence of the learning process. We believe that this sensitivity to perturbations is the reason that FNNs are less successful than RNNs and CNNs.

Self-normalizing neural networks (SNNs) are robust to perturbations and do not have high variance in their training errors (see Figure 1). SNNs push neuron activations to zero mean and unit variance thereby leading to the same effect as batch normalization, which enables to robustly learn many layers. SNNs are based on scaled exponential linear units "SELUs" which induce self-normalizing properties like variance stabilization which in turn avoids exploding and vanishing gradients.

Self-normalizing Neural Networks (SNNs)

Normalization and SNNs. For a neural network with activation function f, we consider two consecutive layers that are connected by a weight matrix $\boldsymbol{W}$. Since the input to a neural network is a random variable, the activations $\boldsymbol{x}$ in the lower layer, the network inputs $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x}$, and the activations $\boldsymbol{y} = f(\boldsymbol{z})$ in the higher layer are random variables as well. We assume that all activations x_i of the lower layer have mean $\mu := \mathrm{E}(x_i)$ and variance $\nu := \mathrm{Var}(x_i)$. An activation y in the higher layer has mean $\tilde{\mu} := \mathrm{E}(y)$ and variance $\tilde{\nu} := \mathrm{Var}(y)$. Here $\mathrm{E}(.)$ denotes the expectation and $\mathrm{Var}(.)$ the variance of a random variable. A single activation $y = f(z)$ has net input $z = \boldsymbol{w}^T\boldsymbol{x}$. For n units with activation $x_i, 1 \leqslant i \leqslant n$ in the lower layer, we define n times the mean of the weight vector $\boldsymbol{w} \in \mathbb{R}^n$ as $\omega := \sum_{i=1}^n w_i$ and n times the second moment as $\tau := \sum_{i=1}^n w_i^2$.

We consider the mapping g that maps mean and variance of the activations from one layer to mean and variance of the activations in the next layer

$$\begin{pmatrix}\mu\\ \nu\end{pmatrix} \mapsto \begin{pmatrix}\tilde{\mu}\\ \tilde{\nu}\end{pmatrix} : \quad \begin{pmatrix}\tilde{\mu}\\ \tilde{\nu}\end{pmatrix} = g\begin{pmatrix}\mu\\ \nu\end{pmatrix} . \tag{1}$$

Normalization techniques like batch, layer, or weight normalization ensure a mapping g that keeps (μ, ν) and $(\tilde{\mu}, \tilde{\nu})$ close to predefined values, typically $(0, 1)$.

Definition 1 (Self-normalizing neural net). *A neural network is self-normalizing if it possesses a mapping $g : \Omega \mapsto \Omega$ for each activation y that maps mean and variance from one layer to the next*