ProcessMiner Data Challenge: Activation Function in Deep Learning*

Chitta Ranjan[1], Yisha Xiang[2], Vahab Najari[1], and Adityan Rajendran[1]

{cranjan, vnajari, arajendran}@processminer.com
ProcessMiner Inc., Atlanta, GA 30308.
yisha.xiang@ttu.edu
Texas Tech University, Lubbock, TX 79409.

Abstract. Activation functions in deep learning are an active research area. They play a critical role in making deep learning networks nonlinear to solve complex problem. Despite some past research, activations development is still at its nascent stage. Activation functions are required to have certain properties such as a saturation region, non-linearity, and a non-decaying or exploding gradient. A few additional desired properties are normalization and regularization. Novel activations with these properties as well as with new unidentified special properties can be developed. A new developed activation need not be effective on every problem. Instead, its superiority on a specific type of problem is a major research contribution. This data challenge is aimed to motivate researchers to work in this direction to advance the field of deep learning.

Keywords: activation · deep learning

1 Story of Activation Functions

Activation functions are one of the primary drivers of neural networks. It introduces non-linear properties to a network. In some cases, a network without activation is equivalent to a simple regression model. It is the non-linearity of the activations that make a neural network capable of learning non-linear patterns in complex problems.

There are a variety of activations, e.g., `tanh`, `elu`, `relu`, etc. If appropriately chosen, an activation can significantly improve a model. An appropriate activation is the one that does not have **vanishing** and/or **exploding** gradient issues.

In fact, the vanishing and exploding gradient issues became a bottleneck in developing complex and large neural networks. They were first resolved to some extent with the rectified linear unit (`relu`) and `leaky-relu` in Maas et al. (2013).

`Relu` activation is defined as,

$$g(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

* By ProcessMiner, Inc.

The gradient of `relu` is 1 if $x > 0$ and 0 otherwise. Therefore, when a `relu` unit is "activated," i.e., the input in it is anything greater than zero, its derivative is 1. Due to this, the **gradient vanishment does not happen** for the positive-valued units in an arbitrarily deep network. These units are called *active* units.

Additionally, `relu` is nonlinear at $x = 0$ with a saturation region for $x < 0$. A saturation region is where the gradient is zero. The saturation region dampens the activation variance if it is too large in the lower layers. This helps in learning lower level features, e.g., more abstract distinguishing patterns in a classifier.

However, only one saturation region[1] is desirable. More than one saturation region, like in `tanh` (it has two saturation region shown in Figure 1b), make the variance too small causing gradients vanishment.

Theoretically, `relu` resolved the vanishing gradient issue. Still, researchers were skeptical about `relu` because without any negative outputs the `relu` activations' means (averages) are difficult to control. Their means can easily stray to large values.

Additionally, `relu`'s gradient is zero whenever the unit is inactive. This was believed to be restrictive because the gradient-based backpropagation does not adjust the weights of units that never activate initially and, eventually, causing cases where a unit never activates.

To alleviate this, the `relu` authors further developed `leaky-relu`. Unlike `relu`, it has a scaled-down output, $0.01x$ when $x < 0$. The activations are visualized for comparison in Figure 1a.

As seen in the figure, the `leaky-relu` has a small but non-zero output for $x < 0$. But this yields a non-zero gradient for every input. That is, it has no saturation region. This did not work in favor of `leaky-relu`.

To resolve the issues in `relu` and `leaky-relu`, `elu` was developed in Clevert et al. (2015). The `elu` activation is defined as,

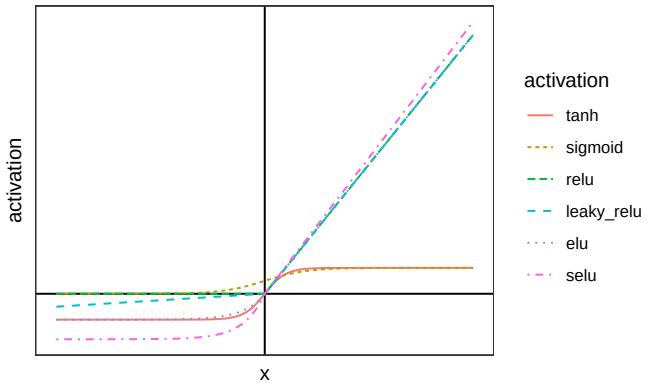
$$g(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(\exp x - 1), & \text{otherwise.} \end{cases} \tag{2}$$

`Elu`'s gradient is visualized in Figure 1b. In contrast to the `leaky-relu`, `elu` has a saturation in the negative region. As mentioned before, the saturation results in small derivatives which decrease the variance and, therefore, the information is well-propagated to the next layer.

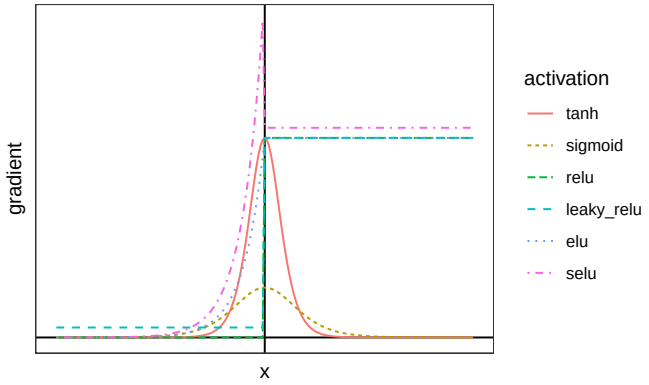
With this property and the non-zero negative outputs, `elu` could enable faster learning as they bring the gradient closer to the natural gradient (shown in Clevert et al. (2015)). However, despite the claims by `elu` or `leaky-relu`, they did not become as popular as `relu`. `Relu`'s robustness made it a default activation for most models.

In any case, `relu` and other activations developed thus far could not address the gradient explosion issue. Batch normalization, a computation outside of ac-

[1] A *saturation* region where the gradient is zero.



(a) Activation output.



(b) Activation gradient.

Fig. 1: Activations functions. The top chart compares the shape of activations $g(x)$ and the bottom compares their gradients $\frac{\partial g(x)}{\partial x}$.
Source: *Understanding Deep Learning* by Chitta Ranjan Ranjan (2020).

tivation done in a `BatchNormalization` layer, was typically used to address it. Until Scaled Exponential Linear Unit (`selu`) was developed in Klambauer et al. (2017).

`Selu` can construct a self-normalizing neural network. This addresses both the vanishing and exploding gradient issues at the same time.

A `selu` activation, shown in Equation 3 below, appears to be a minor change in `elu` in Equation 2 with a λ factor.

$$g(x) = \begin{cases} \lambda x, & \text{if } x > 0 \\ \lambda\alpha(\exp x - 1), & \text{otherwise} \end{cases} \tag{3}$$

where, $\lambda > 1$.

But Klambauer et al. (2017) proved that the simple change brought an important property of *self-normalization* that none of the predecessors had.

The development of `selu` in Klambauer et al. (2017) outlines the desired properties of an activation. Among them, the presence of negative and positive values, and a (one) saturation region are the priorities. Figure 1a and 1b display the presence/absence of these properties among the popular activations. Only `elu` and `selu` have both the properties. However, `selu` went beyond `elu` with two additional attributes,

- **Larger gradient**. A gradient larger than one. This increases the variance if it is too small in the lower layers. This would make learning low-level features in deeper networks possible.
  Moreover, the gradient is larger around $x = 0$ compared to `elu` (see Figure 1b). This reduces the noise from weaker nodes and guides them to their optimal values faster.
- **Balanced variance**. A fixed point where the variance damping (due to the gradient saturation) is equalized by variance inflation (due to greater than 1 gradient). This controls the activations from vanishing or exploding.

2 Research Problem

2.1 Background

The story of activations is told above to inspire the development of novel activations. The essential properties learned from the past activation research are,

1. **Non-linearity**. It makes a network nonlinear to solve complex problems.
2. **Gradient**. A region where the activation gradient is ≥ 1 and $< 1+\delta$, where δ is small, to avoid gradient vanishment and explosion, respectively.
3. **Saturation region**. A region where the gradient becomes 0 to reduce variance.

Additional desired properties are,

1. **Normalization**, and
2. **Regularization**.

2.2 Problem Statement

Develop a novel activation function which has a better efficacy on at least one type of problem, such as object detection, time series, speech translation, etc.

2.3 Example

An example for a novel activation development and implementation is taken from Section 4.8 in Ranjan (2020).

Here a new activation, *Thresholded Exponential Linear Unit* (`telu`), is defined as,

$$g(x) = \begin{cases} \lambda x, & \text{if } x > \tau \\ 0, & \text{if } -\tau \leq x \leq \tau \\ \lambda\alpha(\exp x - 1), & \text{if } x < -\tau \end{cases}. \tag{4}$$

In this activation, weak nodes smaller than τ will be deactivated. The idea behind thresholding small activations is applying regularization directly through the `telu` activation function. Regularization is another desired property mentioned in § 2.1.

Section 4.8 in the UDL book Ranjan (2020) shows an implementation using TensorFlow in Python.

3 Manuscript Outline for Submission

A submission should be a manuscript submitted or ready to submit to IISE proceedings, or a relevant conference/journal. It is also recommended to post the manuscript on Arxiv. The manuscript should follow the author guidelines of the intended proceedings/journal.

The manuscript should contain the following sections. Authors can rename the below mentioned sections but ensure that they contain the recommended topics.

- Abstract
- Introduction
  - High level description of the proposed activation.
  - Application area, e.g., speech analysis.
  - Research contribution.
- Proposed Activation
  - Activation function
  - Theoretical properties
- Experimental Validation
  - Validate its performance on synthesized data that corresponds to the type of problem covered in the research.

- Benchmark performance against other activations.
- Sensitivity analysis.

- Real Data Analysis
  - Analysis results on real data sets. It is recommended to perform analysis on a minimum of three real data sets. Analysis on five or more data sets is preferred. The data sets can be public, proprietary, or both (e.g., if you are performing real data analysis on five data sets, all could be public, proprietary, or some could be public and the rest proprietary). For proprietary data sets, provide the source of the data, and describe it in sufficient detail. For public data sets, a brief description and a reference for the details is sufficient.
  - Compare performance against other activations.
- Discussion
  - Discuss the performance and sensitivity analysis in experimental validation.
  - Discuss the real data analysis
- Conclusion
- Bibliography
  - Cite Ranjan (2020) as it provides the motivation and background for this research.

4 Deadline

The submission deadline is: April 29, 2022.
Submission can be made by sharing the arxiv link or manuscript pdf to:

- Dr. Yisha Xiang, yisha.xiang@ttu.edu, and
- Dr. Chitta Ranjan, cranjan@processminer.com.

Participants are encouraged to work in teams. The finalist teams will require to present their work at the IISE annual conference, 2022.
Winners' Prize:

- **First Prize: $2,000**
- **Second Prize: $1,000**

Bibliography

D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017.

A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

C. Ranjan. *Understanding Deep Learning: Application in Rare Event Prediction*. Connaissance Publishing, Dec 2020. https://doi.org/10.13140/RG.2.2.34297.49765. URL: `www.understandingdeeplearning.com`.