Which Activation Function Works Best for Training Artificial Pancreas: Empirical Fact and Its Theoretical Explanation

1st Lehel Dénes-Fazakas
Physiological Controls Research Center
Appl. Informatics & Appl. Math. Doctoral School
Óbuda University
Budapest, Hungary
denes-fazakas.lehel@uni-obuda.hu

2nd László Szilágyi
Computational Intelligence Res. Group
Sapientia University
Tg. Mures, Romania, and
Physiological Controls Res. Center
Óbuda University
Budapest, Hungary
lalo@ms.sapientia.ro

3rd György Eigner
Physiological Controls Res. Center
Biomatics and Applied AI Institute
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
eigner.gyorgy@uni-obuda.hu

4th Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
El Paso, Texas, USA
olgak@utep.edu

5th Martine Ceberio
Department of Computer Science
University of Texas at El Paso
El Paso, Texas, USA
mceberio@utep.edu

6th Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, Texas, USA
vladik@utep.edu

***Abstract*—One of the most effective ways to help patients at the dangerous levels of diabetes is an artificial pancreas, a device that constantly monitors the patient's blood sugar level and injects insulin based on this level. Patient's reaction to insulin is highly individualized, so the artificial pancreas needs to be trained on each patient. It turns out that the best training results are attained when instead of the usual ReLU neurons, we use their minor modification known as Exponential Linear Units (ELU). In this paper, we provide a theoretical explanation for the empirically observed effectiveness of ELUs.**

***Index Terms*—diabetes, artificial pancreas, neural network, Rectified Linear Unit (ReLU), Leaky ReLU, Exponential Linear Unit (ELU)**

I. Formulation of the Problem: Diabetes, Artificial Pancreas, and Empirical Data About Its Training

Diabetes is a serious problem. Any dynamical system – be it an engine or a living being – needs energy to function. To many living creatures, energy comes with food. Food is digested. Energy contained in food is transformed into glucose – a form of sugar. Glucose is then circulated by blood into different organs that transform it into other molecules that fuel the cells. Glucose transformation is controlled by a special hormone called *insulin*. It is produced by an organ known as *pancreas*.

For most people, this process works well. However, in 8% of the people – almost half a billion – either not enough insulin in generated or the cells of the body do not react properly to the insulin. This condition is known as *diabetes*.

Diabetes is usually diagnosed by an increase blood sugar level: when the glucose is not absorbed, the blood sugar level increases.

Diabetes is a serious problem: it prevents fuel from coming to the cells that form the body, and this can have drastic consequences – including death. Every year, more than a million people die of diabetes [1].

How to help diabetic patients: enter artificial pancreas. One of the main reasons for diabetes is that a body does not produce enough insulin. So, a natural treatment is to inject missing insulin into the body.

Usually, diabetic patients periodically check their blood sugar levels and, if needed, inject insulin. This is not a perfect arrangement: it requires a significant amount of efforts on behalf of the patient, and it does not prevent unhealthy fluctuations of the blood sugar level. To help patients, researchers have been developing automatic systems that continuously measure the blood sugar level and inject insulin if needed. Such systems act as substitutes for a malfunctioning pancreas

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395, and by the AT&T Fellowship in Information Technology. It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI). The work of L. Szilágyi was additionally supported by the Consolidator Researcher Program of Óbuda University. Project no. 2019-1.3.1-KK-2019-00007. has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2019-1.3.1-KK funding scheme. This project has been supported by the National Research, Development, and Innovation Fund of Hungary, financed under the TKP 2021-NKTA-36 funding scheme.

and are thus known as *artificial pancreas* [2].

These systems need to be trained. Different patients have different reactions to insulin. Because of this, artificial pancreas systems need to be individualized. To best way to adjust the system to a patient is to train this system on the history of this patient. This way, the system will be able to predict how this particular patient will react to different insulin doses – and, based on these predictions, select the does that is optimal for this patient.

What is the best way to train these systems: empirical results. At present, the most effective training technique is using artificial neural networks; see, e.g., [3]. So naturally, such networks have been used to train the artificial pancreas.

An artificial neural network consists of *neurons* that transform input signals $x_1, \ldots, x_n$ into the output signal

$$y = s(w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0),$$

where w_i are coefficients (known as *weights*) that need to be determined during the training, and $s(z)$ is a function known as *activation function*. The name "neurons" comes from the fact that these computational units simulate – in a very simplified way – biological neurons, cells that form our brain. In the brain, the corresponding function $s(z)$ is (non-strictly) increasing: if $z \le z'$, then $s(z) \le s(z')$. Because of this, in most applications, researchers use increasing activation functions. The most widely used activation function is

$$s(z) = \max(0, z).$$

This function is known as the *Rectified Linear Unit* (ReLU, for short). However, other activation functions are also used.

Some neurons process the measurement results, they for the first layer. Other neurons process the results of the neurons from the first layer, etc., until finally, some neurons generate the desired output.

To design the training scheme for a neural network, we need to select the value of a large number of scheme's parameters; these parameters are known as *hyperparameters*. It is known that an appropriate selection of hyperparameters can drastically increase the training effectiveness. Because of this, researchers have tested different combinations of hyperparameters (and of different activation functions) to analyze which combinations work the best [4]. In addition to known activation functions, they also tried modified version of these functions.

Empirically, one of the most successful modifications of ReLU are *leaky* activation functions. The idea behind them is that when the linear combination

$$w_1 \cdot x_1 + \ldots + w_n \cdot x_n + w_0$$

is negative, ReLU function returns 0 – and thus, does not react to changes in the inputs. So as not to lose possible helpful information, it is desirable to modify the ReLU function so that it produces non-zero values $s(z)$ for negative z. The most well-known leaky version of ReLU uses

$$s(z) = -\alpha \cdot z \tag{1}$$

for $z < 0$, where α is a small positive number. Interestingly, as shown in [4], for training artificial pancreas, the most effective leaky activation function turned out to be a function for which

$$s(z) = \alpha \cdot (\exp(\beta \cdot z) - 1) \tag{2}$$

for $z < 0$, where $\beta > 0$. This function was first proposed in [5]; see also [6]. Since this function is exponential for $z < 0$ and linear for $z > 0$, it is called Exponential Linear Unit (ELU, for short).

Natural question. There are infinitely many possible activation functions, and it is therefore not possible to try all of them. So, a natural question is:

- whether the ELU function is simply the best of the ones that we tried – and there may be better activations that we did not trye yet,
- or there is some theoretical reason why this particular activation function worked the best.

What we do in this paper. In this paper, we prove that leaky ReLU and ELU are indeed, in some reasonable sense, optimal among all leaky version of ReLU. This result makes us more confident that ELU is indeed the activation function that should be used for the training of an artificial pancreas.

II. Let Us Formulate the Problem in Precise Terms

What are we looking for. We are interested in a continuous non-strictly increasing function $s(z)$ for which $s(z) = z$ for all $z > 0$. Since we know the values of this function for all non-negative z, it is sufficient to describe its values for all negative z.

In these terms, what we are looking for are non-strictly increasing functions $s(z)$ which are defined for all $z \le 0$ and for which $s(0) = 0$.

We need to have a family of such functions. An experience teaches us that in different situations, different activation functions may be more effective – if this was not the case, we would not need different activation functions. So, what we want to select is not a single function $s(z)$, but rather a whole *family* of such functions. In other words, we are looking not for a single function $s(z)$, bur rather for an expression $s(z, C_1, \ldots, C_k)$ that, for different values of the parameters $C_1, \ldots, C_k$, would provide us with different activation functions.

Possibility of linearization. We know that ReLU functions are very effective in many applications. Because of this, we want our selected activation functions to be close to ReLU. In other words, we want the desired function $s(z)$ to be close to 0 for $z < 0$.

In general, any sufficiently smooth dependence can be expanded in Taylor series. In particular, for the dependence $s(z, C_1, \ldots, C_k)$ on the parameters C_i, we have

$$s(z, C_1, \ldots, C_k) =$$
$$s_0(z) + \sum_{i=1}^{k} C_i \cdot s_i(z) + \sum_{i=1}^{k} \sum_{j=1}^{k} C_i \cdot C_j \cdot s_{ij}(z) + \ldots$$

Here, quadratic terms are much smaller than linear terms, cubic terms are even smaller, etc. In our case, the whole function is small, which means its linear part is small. Thus, quadratic terms are negligible: e.g., if linear terms are 10%, their square is 1%, which is much smaller. Thus, it is safe to ignore quadratic and higher order terms and assume that the dependence on the coefficients C_i is linear:

$$s(z, C_1, \ldots, C_k) = s_0(z) + \sum_{i=1}^{k} C_i \cdot s_i(z).$$

Again, we know that ReLU is effective, so we want ReLU to be a particular case of our family – corresponding, e.g., to $C_1 = \ldots = C_k = 0$. For ReLU, $s(z) = 0$ for all $z < 0$, so we should take $s_0(z) = 0$ and the family takes the form

$$\left\{ s(z) = \sum_{i=1}^{k} C_i \cdot s_i(z) \right\}_{C_1, \ldots, C_k}. \tag{3}$$

Neural networks are trained by gradient descent. Since we want our neural network to be close to the ReLU-based ones, we thus need to make sure that not only the activation function is close to ReLU, but that its derivative exists and is close to the derivatives of ReLU. In other words, for negative z, the selected function must be differentiable and its derivative $s'(z)$ must be close to 0 – i.e., we must have a small bound b on the value of this derivative. So, we arrive at the following definition.

Definition 1. *By a* modified ReLU function*, we mean a differentiable non-strictly increasing function $s(z)$ which:*

- *is defined for all $z \le 0$,*
- *is not identically 0, is non-strictly increasing,*
- *has the property $s(0) = 0$, and*
- *has all the values of its derivative bounded by some constant b.*

Definition 2. *Let k be a positive integer. By a* k-dimensional family*, we mean a family of type* (3) *with differentiable functions $s_i(z)$.*

Need for fast computations. Neural networks have numerous applications. Some applications – e.g., Large Language Models like ChatGPT – use literally trillions of weights and require high-performance computers to train. In contrast, in our application, we are talking about a small device that will be either imbedded in a patient or carried by the patient. With this device, we cannot perform too many computations.

In particular, we need to select activation functions that are as simple to compute as possible. This means, for example, that we should select the smallest possible number of terms k in the general description of the family (3). In other words, we need to select a family (3) with the smallest possible k.

What we mean by optimal. Usually, "optimal" means that the value of some objective function is the largest (or, if appropriate, the smallest). However, this is not the most general way to describe optimality. For example, if we are selecting the most stable control, i.e., the control with the smallest possible value of instability, we may have several controls which are, in this sense, the best. In such situations, it is reasonable to use this non-uniqueness to optimize something else – e.g., minimize the control's non-smoothness.

If this still leaves us with several optimal alternatives, we can use this non-uniqueness to optimize something else – until we reach the *final* optimality criterion, for which there is exactly one optimal alternative.

To take these complexities into account, we will not limit ourselves to numerical optimality criteria, we will take into account that in general, an optimality criterion is a method that allows us to decide, at least for some pairs of alternatives a and b, where b is better (or of the same quality) that a. We will denote this relation by $a \le b$.

Of course, if b is better than a and c is better than b, then c should be better than a. In mathematics, this property is known as *transitivity*. Relations with this property are known as *pre-orders*. Thus, we arrive at the following definition.

Definition 3. *Let a set A be given; its elements will be called* alternatives*.*

- *By an* optimality criterion *on the set A, we mean a transitive relation $\le$ (pre-order) on this set.*
- *We say that an alternative $a_{\rm opt}$ is* optimal *if $a \le a_{\rm opt}$ for all $a \in A$.*
- *We say that an optimality criterion is* final *if for this criterion, there is exactly one optimal alternative.*

Need for shift-invariance. We can use different starting points for measuring the inputs z. For example, for the artificial pancreas, a reasonable idea is to use the difference between the current blood sugar level and the ideal blood sugar level – this corresponds to using the ideal blood sugar level as the starting point.

This sounds like a reasonable idea, but, according to the medical professionals, there are many different ways for selecting this ideal level. For example, we can select the general ideal level of 100 units, but this may be somewhat misleading:

- If a child's blood sugar level is getting close to 90, this is a reason to be seriously alarmed – since for children, this level is usually much smaller.
- On the other hand, for an older patient, even levels larger than 100 are OK and probably do not require insulin injections – provided that the patient otherwise feels well.

It is therefore more reasonable to use different starting points for different patients, depending on their age, gender, etc.

As medical doctors learn more about diseases, what is considered an ideal level changes. For example, in recent decades, there have been changes in thresholds separating healthy and unhealthy blood pressure, healthy and unhealthy cholesterol level, etc. When the starting point changes, the differences z processed by our neural network change to $z + z_0$ for some value z_0 – which is the difference between the new and the original starting points. The selection of the starting point is just the question of convenience. So, it makes sense

to require that the relative quality of different k-dimensional families should not change if we simply change the starting point.

Let us describe this requirement in precise terms.

Definition 4. *Let z_0 be a real number.*

- *For each family a of type (3) corresponding to the functions $s_i(z)$, by its z_0-*shift *$S_{z_0}(a)$, we mean the family corresponding to the functions $s_i(z + z_0)$.*
- *We say that the optimality criterion is* shift-invariant *if for each value z_0 and for every two families a and b for which $a \le b$, we have $S_{z_0}(a) \le S_{z_0}(b)$.*

Now, we are ready to formulate our main result.

III. Main Result

Proposition.

- *The smallest k for which there exists a shift-invariant final optimality criterion on the set of all k-dimensional families that contains at least one modified ReLU function is $k = 2$.*
- *For $k = 2$, for each shift-invariant final optimality criterion, if the optimal family contains a modified ReLU function, then this function is either a leaky ReLU (1) or has the ELU form (2).*

Discussion. This result explains that out of the simplest families – for which k is the smallest possible – leaky ReLU and ELU are indeed the optimal modified ReLU functions. Since in our case, ELU works better than Leaky ReLU, this means that ELU is optimal.

Proof.

1°. Let us first prove that for each shift-invariant final optimality criterion, the optimal family $a_{\rm opt}$ is itself shift-invariant, i.e., $S_{z_0}(a_{\rm opt}) = a_{\rm opt}$.

Indeed, by definition of optimality, we have $a \le a_{\rm opt}$ for all $a \in A$. In particular, for each a and for each z_0, we have $S_{-z_0}(a) \le a_{\rm opt}$. Since the optimality criterion is shift-invariant, this implies that $S_{z_0}(S_{-z_0}(a)) \le S_{z_0}(a_{\rm opt})$. One can easily check that $S_{z_0}(S_{-z_0}(a)) = a$, so we get $a \le S_{z_0}(a_{\rm opt})$ for all $a \in A$. By definition of optimality, this means that the alternative $S_{z_0}(a_{\rm opt})$ is optimal. But our optimality criterion is final, which means that it has only optimal alternative. Thus, we must indeed have $S_{z_0}(a_{\rm opt}) = a_{\rm opt}$.

2°. Let $s_1(z), \ldots, s_k(z)$ be the functions that form the optimal family. According to Part 1 of the proof, the optimal family is shift-invariant. This means that if we shift each function from this family, we still get a function from the same family. In particular, we can shift the functions $s_i(z)$ themselves – each of them belongs to the family when we take $C_i = 0$ and all other C_j equal to 0. Thus, for each i, we have

$$s_i(z + z_0) = \sum_{j=1}^{n} C_{ij}(z_0) \cdot s_j(z), \qquad (4)$$

for some coefficients C_{ij} depending on z_0.

We can plug in k different values $z_1, \ldots, z_k$ into the formula (4) and get the following system of k linear equations with k unknowns $C_{i1}(z_0), \ldots, C_n(z_0)$:

$$s_i(z_m + z_0) = \sum_{j=1}^{n} C_{ij}(z_0) \cdot s_j(z_m), \quad m = 1, \ldots, k. \qquad (5)$$

The solution $c_{ij}(z_0)$ of a linear system is a linear function of its right-hand sides. The right-hand sides $s_i(z_m + z_0)$ are smooth functions of z_0, so we conclude that the functions $C_{ij}(z_0)$ are also differentiable. Thus, all the functions in the equality (4) are differentiable.

So, we can differentiate both sides of the equality (4) with respect to z_0, resulting in:

$$s_i'(z + z_0) = \sum_{j=1}^{n} C_{ij}'(z_0) \cdot s_j(z).$$

In particular, for $z_0 = 0$, we get

$$s_i'(z) = \sum_{j=1}^{n} c_{ij} \cdot s_j(z), \qquad (6)$$

where we denoted $c_{ij} \stackrel{\rm def}{=} C_{ij}'(0)$.

The system (6) is a system of linear differential equations with constant coefficients. It is known that each solution to this system of equations is a linear combination of the terms $z^v \cdot \exp(\lambda \cdot z)$, where:

- λ is an eigenvalue of the matrix c_{ij} and
- v is a non-negative integer that is smaller than the multiplicity of this eigenvalue (and is, thus, equal to 0 when this multiplicity is 1).

3°. Let us use this result to show that the case $k = 1$ is not possible.

Indeed, in this case, the multiplicity is 1, so all possible solutions of the system (6) have the form

$$s(z) = C_1 \cdot \exp(\lambda \cdot z).$$

The only way for this function to satisfy the condition $s(0) = 0$ is to have $C_1 = 0$, in which case the function $s(z)$ is identically 0. Thus, for $k = 1$, the optimal family cannot contain a modified ReLU function.

4°. Let us prove that the case $k = 2$ is possible.

Indeed, we can have the following optimality criterion:

- the family

$$\{C_1 \cdot \exp(\beta \cdot z) + C_2\}_{C_1, C_2}$$

  is better than all other families, and
- all other families are of the same quality.

One can easily check that this criterion is final and shift-invariant, and that the optimal family contains the modified ReLU function (2).

5°. To complete the proof, let us prove that for $k = 2$, for each shift-invariant final optimality criterion, if the optimal family

contains a modified ReLU function, then this function is either a leaky ReLU (1) or it has the ELU form (2).

Indeed, in this case, we have two options:

- either we have two different eigenvalues $\lambda_1 \neq \lambda_2$,
- or we have a double eigenvalue λ.

5.1°. In the first case, when we have two different eigenvalues, a generic function $s(z)$ from the optimal family has the form

$$s(z) = C_1 \cdot \exp(\lambda_1 \cdot z) + C_2 \cdot \exp(\lambda_2 \cdot z). \qquad (7)$$

Without losing generality, we can assume that $\lambda_1 < \lambda_2$. In this case, for $z \to -\infty$, the term corresponding to λ_1 asymptitically dominates both the values of the function and the values of its derivative. Since the derivative has to be bounded, we thus have $\lambda_1 \geq 0$ – otherwise, the derivative would grow exponentially when $z \to -\infty$.

If we have $\lambda_1 > 0$, then, since $\lambda_1 < \lambda_2$, we will also have $\lambda_2 > 0$. In this case, for $z \to -\infty$, both terms in the expression (7) tend to 0. So, such a family cannot contain a modified ReLU, since a modified ReLU should be different from 0, non-strictly increasing and have $s(0) = 0$, while here we have

$$0 = \lim_{z \to -\infty} \leq s(z) \leq s(0) = 0,$$

so $s(z) \equiv 0$.

So, the only remaining case is when $\lambda_1 = 0 < \lambda_2$. In this case, a generic function $s(z)$ from this family has the form $C_1 + C_2 \cdot \exp(\beta \cdot z)$, where we denoted $\beta \stackrel{\text{def}}{=} \lambda_2$. Monotonicity implies that $C_2 > 0$, and the requirement that $s(0) = 0$ implies that $C_1 = -C_2$. Thus, we indeed get the ELU expression.

5.2°. When we have a double eigenvalue λ, the general solution has the form $C_1 \cdot \exp(\lambda \cdot z) + C_2 \cdot z \cdot \exp(\lambda \cdot z)$. Similar to Part 5.1, we conclude that:

- the value λ cannot be negative – then the derivative $s'(z)$ will not be bounded, and
- the value λ cannot be positive – since then we would not get monotonicity.

So, the only remaining option is when $\lambda = 0$, in which case $s(z) = C_1 + C_2 \cdot z$. The requirement that $s(0) = 0$ implies that $C_1 = 0$, so we get the leaky ReLU.

The proposition is proven.

REFERENCES

[1] K. Ogurtsova, L. Guariguata, N. Barengo, R. Paz Lopez-Doriga J. Sacre, W. Julian, S. Karuranga, H. Sun, E.J. Boyko, and D.J. Magliano, "IDF diabetes Atlas: Global estimates of undiagnosed diabetes in adults for 2021", *Diabetes research and clinical practice*, Elsevier, Vol. 183, Pgs. 109118, 2022.

[2] J. Tasic, M. Takács and L. Kovacs, "Control Engineering Methods for Blood Glucose Levels Regulation", *Acta Polytechnica Hungarica*, Vol. 19, No. 7, Pgs. 127–152, 2022.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, Cambridge, Massachusetts, 2016.

[4] L. Dénes-Fazakas, M. Siket, L. Szilágyi, G. Eigner, and L. Kovacs, "Effect of hyperparameters of reinforcement learning in blood glucose control", *Proceedings of the 2023 IEEE Conference on Systems, Man, and Cybernetics*, Maui, Hawaii, October 1–4, 2023, to appear.

[5] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (ELUs)*, 2015, arXiv:1511.07289. [Online]. Available: http://arxiv.org/abs/1511.07289

[6] A. A. Alkhouly, A. Mohammed, and H. A. Hefny, "Improving the performance of deep neural networks using two proposed activation functions", *IEEE Access*, 2021, Vol. 9, pp. 82249–82271.