12-2019

Why Spiking Neural Networks Are Efficient: A Theorem

Michael Beer

Julio Urenda

Olga Kosheleva

Vladik Kreinovich

Comments:
Technical Report: UTEP-CS-19-111

Why Spiking Neural Networks Are Efficient: A Theorem

Michael Beer[1], Julio Urenda[2,3],
Olga Kosheleva[4], and Vladik Kreinovich[3]
[1]Institute for Risk and Reliability
Leibniz University Hannover
30167 Hannover, Germany
beer@irz.uni-hannover.de
[2]Department of Mathematical Sciences
[3]Department of Computer Science
[4]Department of Teacher Education
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
jcurenda@utep.edu, olgak@utep.edu, vladik@utep.edu

Abstract

Current artificial neural networks are very successful in many machine learning applications, but in some cases they still lag behind human abilities. To improve their performance, a natural idea is to simulate features of biological neurons which are not yet implemented in machine learning. One of such features is the fact that in biological neural networks, signals are represented by a train of spikes. Researchers have tried adding this spikiness to machine learning and indeed got very good results, especially when processing time series (and, more generally, spatio-temporal data). In this paper, we provide a theoretical explanation for this empirical success.

1 Formulation of the Problem

Why spiking neural networks: a historical reason. At this moment, artificial neural networks are the most successful – and the most promising – direction in Artificial Intelligence; see, e.g., [3].

Artificial neural networks are largely patterned after the way the actual biological neural networks work; see, e.g., [2, 3]. This patterning makes perfect sense: after all, our brains are the result of billions of years of improving evolution, so it is reasonable to conclude that many features of biological neural

networks are close to optimal – not very efficient features would have been filtered out in this long evolutionary process.

However, there is an important difference between the current artificial neural networks and the biological neural networks:

- when some processing of the artificial neural networks is implemented in hardware – by using electronic or optical transformation – each numerical value is represented by the intensity (amplitude) of the corresponding signal;
- in contrast, in the biological neural networks, each value – e.g., the intensity of the sound or of the light – is represented by a series of instantaneous spikes, so that the original value is proportional to the frequency of these spikes.

Since simulating many other features of biological neural networks has led to many successes, a natural idea is to also try to emulate the spiking character of the biological neural networks.

Spiking neural networks are indeed efficient. Interestingly, adding spiking to artificial neural networks has indeed led to many successful applications, especially in processing temporal (and even spatio-temporal) signals; see, e.g., [4] and references therein.

But why? A biological explanation of the success of spiking neural networks – based on the above evolution arguments – makes perfect sense, but it would be nice to supplement it with a clear mathematical explanation – especially since, in spite of all the billions years of evolution, we humans are not perfect as biological beings, we need medicines, surgeries, and other artificial techniques to survive, and our brains often make mistakes.

What we do in this paper. In this paper, we consider the question of signal representation from the mathematical viewpoint, and we show that the spiking representation is indeed optimal in some reasonable sense.

2 Analysis of the Problem and the First Result

Looking for basic functions. In general, to represent a signal $x(t)$ means to approximate it as a linear combination of some basic functions. For example, it is reasonable to represent a periodic signal as a linear combination of sines and cosines. In more general cases – e.g., when analyzing weather – it makes sense to represent the observed values as a linear combination of functions t, t^2, etc., representing the trend and sines and cosines that describe the periodic part of the signal. To get a more accurate presentation, we need to take into account that the amplitudes of the periodic components can also change with time, so we end up with terms of the type $t \cdot \sin(\omega \cdot t)$.

If we analyze how radioactivity of a sample changes with time, a reasonable idea is to describe the measured values $x(t)$ as a linear combination of

exponentially decaying functions $\exp(-k \cdot t)$ representing the decay of different isotopes, etc.

So, in precise terms, selecting a representation means selecting an appropriate family of basic functions. In general, we may have several parameters $c_1, \ldots, c_n$ characterizing functions from each family. Sometimes, there is only one parameter, as in sines and cosines. In other cases, we can have several parameters – e.g., in control applications, it makes sense to consider decaying periodic signals of the type $\exp(-k \cdot t) \cdot \sin(\omega \cdot t)$, with two parameters k and ω. In general, elements $b(t)$ of each such family can be described by a formula $b(t) = B(c_1, \ldots, c_n, t)$ corresponding to different tuples $c = (c_1, \ldots, c_n)$.

Dependence on parameters must be continuous in some reasonable sense. We want the dependence $B(c_1, \ldots, c_n, t)$ to be computable, and it is known that all computable functions are, in some reasonable sense, continuous; see, e.g., [7].

Indeed, in real life, we can only determine the values of all physical quantities c_i with some accuracy: measurements are always not 100% accurate, and computations always involve some rounding. For any given accuracy, we can provide the value with this accuracy – but it will practically never be the exact value. Thus, the approximate values of c_i are the only thing that our computing algorithm can use when computing the value $B(c_1, \ldots, c_n, t)$. This algorithm can ask for more and more accurate values of c_i, but at some point it must produce the result. At this point, we only known approximate values of c_i, i.e., we only know the interval of possible values of c_i. And for all the values of c_i from this interval, the result of the algorithm provides, with the given accuracy, the approximation to the desired value $B(c_1, \ldots, c_n, t)$. This is exactly what continuity is about!

One has to be careful here, since the real-life processes may actually be, for all practical purposes, discontinuous. Sudden collapses, explosions, fractures do happen.

For example, we want to make sure that a step-function which is equal to 0 for $t < 0$ and to 1 for $t \geq 0$ is close to an "almost" step function which is equal to 0 for $t < 0$, to 1 for $t \geq \varepsilon$ (for some small ε) and to t/ε for $t \in (0, \varepsilon)$.

In such situations, we cannot exactly describe the value at moment t – since the moment t is also measured approximately, but what we can describe is its values at a moment close to t. In other words, we can say that the two functions $a_1(t)$ and $a_2(t)$ are ε-close if:

- for every moment t_1, there exists moments t_{21} and t_{22} which are ε-close to t_1 (i.e., for which $|t_{2i} - t_1| \leq \varepsilon$) and for which $a_1(t_1)$ is ε-close to a convex combination of values $a_2(t_{2i})$, and
- for every moment t_2, there exists moments t_{11} and t_{12} which are ε-close to t_2 and for which $a_2(t_2)$ is ε-close to a convex combination of values $a_1(t_{1i})$.

Additional requirement. Since we consider linear combinations of basic functions, it does not make sense to have two basic functions that differ only by a

constant: if $b_2(t) = C \cdot b_1(t)$, then there is no need to consider the function $b_2(t)$ at all; in each linear combination we can replace $b_2(t)$ with $C \cdot b_1(t)$.

We would like to have the simplest possible family of basic functions. How many parameters c_i do we need? The fewer parameters, the easier it is to adjust the values of these parameters, and the smaller the probability of *overfitting* – a known problem of machine learning in particular and of data analysis in general, when we fit the formula to the observed data and its random fluctuations too well and this make it much less useful in other cases where random fluctuations will be different.

We cannot have a family with no parameters at all – that would mean, in effect, that we have only one basic function $b(t)$ and we approximate every signal by an expression $C \cdot b(t)$ obtained by its scaling. This will be a very lousy approximation to real-life processes – since these processes are all different, they do not resemble each other at all.

So, we need at least one parameter. Since we are looking for the simplest possible family, we should therefore consider families depending on a single parameter c_1, i.e., families consisting of functions $b(t) = B(c_1, t)$ corresponding to different values of the parameter c_1.

Most observed processes are limited in time. From our viewpoint, we may view astronomical processes are going on forever – although, in reality, even they are limited by billions of years. However, in general, the vast majority of processes that we observe and that we want to predict are limited in time: a thunderstorm stops, a hurricane end, after-shocks of an earthquake stop, etc.

From this viewpoint, to get a reasonable description of such processes, it is desirable to have basic functions which are also limited in time, i.e., which are equal to 0 outside some finite time interval. This need for finite duration is one of the main reasons in many practical problems, a decomposition into wavelets performs much better that a more traditional Fourier expansion into linear combinations of sines and cosines; see, e.g., [1] and references therein.

Shift- and scale-invariance. Processes can start at any moment of time. Suppose that we have a process starting at moment 0 which is described by a function $x(t)$. What if we start the same process t_0 moments earlier? At each moment t, the new process has been happening for the time period $t+t_0$. Thus, at the moment t, the new process is at the same stage as the original process will be at the future moment $t + t_0$. So, the value $x'(t)$ of a quantity characterizing the new process is equal to the value $x(t + t_0)$ of the original process at the future moment of time $t + t_0$.

There is no special starting point, so it is reasonable to require that the class of basic function not change if we simply change the starting point. In other words, we require that for every t_0, the shifted family $\{B(c_1, t+t_0)\}_{c_1}$ coincides with the original family $\{B(c_1, t)\}_{c_1}$.

Similarly, processes can have different speed. Some processes are slow, some are faster. If a process starting at 0 is described by a function $x(t)$, then a λ times faster process is characterized by the function $x'(t) = x(\lambda \cdot t)$. There is no

special speed, so it is reasonable to require that the class of basic function not change if we simply change the process's speed. In other words, we require that for every $\lambda > 0$, the "scaled" family $\{B(c_1, \lambda \cdot t\}_{c_1}$ coincides with the original family $\{B(c_1, t)\}_{c_1}$.

Now, we are ready for the formal definitions.

Definition 1. *We say that a function $b(t)$ is* limited in time *if it equal to 0 outside some interval.*

Definition 2. *We say that a function $b(t)$ is a* spike *if it is different from 0 only for a single value t. This non-zero value is called the height of the spike.*

Definition 3. *Let $\varepsilon > 0$ be a real number. We say that the numbers a_1 and a_2 are* ε-close *if $|a_1 - a_2| \le \varepsilon$.*

Definition 4. *We say that the functions $a_1(t)$ and $a_2(t)$ are* ε-close *if:*

- *for every moment t_1, there exists moments t_{21} and t_{22} which are ε-close to t_1 (i.e., for which $|t_{2i} - t_1| \le \varepsilon$) and for which $a_1(t_1)$ is ε-close to a convex combination of values $a_2(t_{2i})$, and*
- *for every moment t_2, there exists moments t_{11} and t_{12} which are ε-close to t_2 and for which $a_2(t_2)$ is ε-close to a convex combination of values $a_1(t_{1i})$.*

Comment. One can check that this definition is equivalent to the inequality $d_H(A_1, A_2) \le \varepsilon$ bounding the Hausdorff distance $d_H(A_1, A_2)$ between the two sets A_i each of which is obtained from the closure C_i of the graphs of the corresponding function $a_i(t)$ by adding the whole vertical interval $t \times [a, b]$ for every two points (t, a) and (t, b) with the same first coordinate from the closure C_i.

Definition 5. *We say that a mapping $B(c_1, t)$ that assigns, to each real number c_1, a function $b(t) = B(c_1, t)$ is* continuous *if, for every value c_1 and for every $\varepsilon > 0$, there exists a real number $\delta > 0$ such that, if c_1' is δ-close to c_1, then the function $b(t) = B(c_1, t)$ is ε-close to the function $b'(t) = B(c_1', t)$.*

Definition 6. *By a* family of basic functions*, we mean a continuous mapping for which:*

- *for each c_1, the function $b(t) = B(c_1, t)$ is limited in time, and*
- *if c_1 and c_1' are two different numbers, then the functions $b(t) = B(c_1, t)$ and $b'(t) = B(c_1', t)$ cannot be obtained from each other by multiplication by a constant.*

Definition 7. *We say that a family of basic functions $B(c_1, t)$ is* shift-invariant *if for each t_0, the following two classes of functions of one variable coincide:*

$$\{B(c_1, t)\}_{c_1} = \{B(c_1, t + t_0)\}_{c_1}.$$

Definition 8. *We say that a family of basic functions $B(c_1, t)$ is* scale-invariant *if for each $\lambda > 0$, the following two classes of functions of one variable coincide:*

$$\{B(c_1, t)\}_{c_1} = \{B(c_1, \lambda \cdot t)\}_{c_1}.$$

Proposition 1. *If a family of basic functions $B(c_1, t)$ is shift- and scale-invariant, then for every c_1, the corresponding function $b(t) = B(c_1, t)$ is a spike, and all these spikes have the same height.*

Discussion. This result explains the efficiency of spikes: namely, a family of spikes is the only one which satisfies the reasonable conditions of shift- and scale-invariance, i.e., the only family that does not change if we change the starting point of the process and/or change the process's speed.

Proof. Let us assume that the family of basic functions $B(c_1, t)$ is shift- and scale-invariant. Let us prove that all the functions $b(t) = B(c_1, t)$ are spikes.

1°. First, we prove that none of the functions $B(c_1, t)$ is identically 0.

Indeed, the zero function can be contained from any other function by multiplying that other function by 0 – and this would violate the second part of Definition 6 (of a family of basic functions).

2°. Let us prove that each function from the given family is a spike.

Indeed, each of the functions $b(t) = B(c_1, t)$ is not identically zero, i.e., it attains non-zero values for some t. By the Definition 6 of a family of basic functions, each of these functions is limited in time, i.e., the values t for which the function $b(t)$ is non-zero are bounded by some interval. Thus, the values $t_- \stackrel{\text{def}}{=} \inf\{t : b(t) \neq 0\}$ and $t_+ \stackrel{\text{def}}{=} \sup\{t : b(t) \neq 0\}$ are finite, with $t_- \leq t_+$.

Let us prove that we cannot have $t_- < t_+$. Indeed, in this case, the interval $[t_-, t_+]$ is non-degenerate. Thus, by an appropriate combination of shift and scaling, we will be able to get this interval from any other non-degenerate interval $[a, b]$, with $a < b$: indeed, it is sufficient to take the transformation $t \to \lambda \cdot t + t_0$, where $\lambda = \dfrac{t_+ - t_-}{b - a}$ and $t_0 = \lambda \cdot a - t_-$. For each of these transformations, due to shift- and scale-invariance of the family, the correspondingly re-scaled function $b'(t) = b(\lambda \cdot t + t_0)$ also belongs to the family $B(c_1, t)$, and for this function, the corresponding values t'_- and t'_+ will coincide with a and b. All these functions are different – so, we will have a 2-dimensional family of functions (i.e., a family depending on 2 parameters), which contradicts to our assumption that the family $B(c_1, t)$ is one-dimensional.

The fact that we cannot have $t_- < t_+$ means that we should have $t_- = t_+$, i.e., that every function $b(t)$ from our family is indeed a spike.

3°. To complete the proof, we need to prove that all the spikes that form the family $B(c_1, t)$ have the same height.

Let us describe this property in precise terms. Let $b_1(t)$ and $b_2(t)$ be any two functions from the family. According to Part 2 of this proof, both functions are spikes, so:

- the value $b_1(t)$ is only different from 0 for some value t_1; let us denote the corresponding height $b_1(t_1)$ by h_1;
- similarly, the value $b_2(t)$ is only different from 0 for some value t_2; let us denote the corresponding height $b_2(t_2)$ by h_2.

We want to prove that $h_1 = h_2$.

Indeed, since the function $b_1(t)$ belongs to the family, and the family is shift-invariant, then for $t_0 \stackrel{\text{def}}{=} t_1 - t_2$, the shifted function $b_1'(t) \stackrel{\text{def}}{=} b_1(t + t_0)$ also belongs to this family. The shifted function is non-zero when $t + t_0 = t_1$, i.e., when $t = t_1 - t_0 = t_2$, and it has the same height h_1.

If $h_1 \neq h_2$, this would contradict to the second part of Definition 6 (of the family of basic functions) – because then we would have two functions $b_1'(t)$ and $b_2(t)$ in this family, which can be obtained from each other by multiplying by a constant. Thus, the heights must be the same.

The proposition is proven.

3 Main Result: Spikes Are, In Some Reasonable Sense, Optimal

It is desirable to check whether spiked neurons are optimal. In the previous section, we showed that spikes naturally appear if we require reasonable properties like shift- and scale-invariance. This provides some justification for the spiked neural networks.

However, the ultimate goal of neural networks is to solve practical problems. From this viewpoint, we need to take into account that a practitioner is not interested in invariance or other mathematical properties, a practitioner wants to optimize some objective function. So, from the practitioner's viewpoint, the main question is: are spiked neurons optimal?

Different practitioners have different optimality criteria. The problem is that, in general, different practitioners may have different optimality criteria. In principle, we can pick one such criterion (or two or three) and analyze which families of basic functions are optimal with respect to these particular criterion – but this will not be very convincing to a practitioner who has a different optimality criterion.

An ideal explanation should work for *all* reasonable optimality criteria. This is what we aim at in this section. To achieve this goal, let us analyze what we mean by an optimality criterion, and which optimality criteria can be considered reasonable. In this analysis, we will follow a general analysis of computing-related optimization problems performed in [5].

What is an optimality criterion: analysis. At first glance, the answer to this question may sound straightforward: we have an objective function $J(a)$ that assigns, to each alternative a, a numerical value $J(a)$, and we want to select

an alternative for which the value of this function is the largest possible (or, if we are interested in minimizing losses, the smallest possible).

This formulation indeed describes many optimality criteria, but not all of them. Indeed, assume, for example, we are looking for the best method a for approximating functions from a given class. A natural criterion may be to minimize the mean squared approximation error $J(a)$ of the method a. If there is only one method with the smallest possible mean squared error, then this method is selected. But what if there are several different methods with the same mean squared error – and this is often the case. In this case, we can use this non-uniqueness to optimize something else: e.g., select, out of several methods with the same mean squared error, the method for which the average computation time $T(a)$ is the smallest.

In this situation, the optimality criterion cannot be described by single objective function, it takes a more complex form. Namely, we say that a method a' is better than a method a if:

- either $J(a) < J(a')$,
- or $J(a) = J(a')$ and $T(a) < T(a')$.

This additional criterion may still leave us with several equally good methods. We can use this non-uniqueness to optimize yet another criterion: e.., worst-case computation time, etc.

The only thing which is needed to describe an optimality criterion is that this criterion must enable us to compare the quality of different alternatives. In mathematical terms, this criterion must enable us to decide which alternatives are better (or of the same quality); let us denote this by $a \le a'$. Clearly, if a' is better than a (i.e., $a \le a'$) and a'' is better than a' ($a' \le a''$), then a'' is better than a ($a \le a''$), so the relation $\le$ must be transitive. Such relations are known as *pre-orders*.

Comment. Not all such relations are orders: that would require an additional property that if $a \le b$ and $b \le a$, then $a = b$, and, as we have mentioned earlier, this is not necessarily true.

An optimality criterion must be final. In terms of the relation $\le$, optimal means better than (or of the same quality as) all other alternatives: $a \le a_{\rm opt}$ for all a.

As we have mentioned earlier, if we have several different optimal alternatives, then we can use this non-uniqueness to optimize something else – i.e., in effect, to modify the corresponding optimality criterion. Thus, when the optimality criterion allows several different optimal alternatives, this means that this criterion is *not* final, it has to be modified. For a *final* criterion, we should have only one optimal alternative.

An optimality criterion must be invariant. In real life, we deal with real-life processes $x(t)$, in which values of different quantities change with time t. The corresponding numerical values of time t depend on the starting point that

we use for measuring time and on the measuring unit: e.g., 1 hour is equivalent to 60 seconds; numerical values are different, but from the physical viewpoint, this is the same time interval.

We are interested in a universal technique for processing data. It is therefore reasonable to require that the relative quality of different techniques should not change if we simply change the starting point for measuring time or a measuring unit.

Let us describe all this in precise terms.

Definition 9. *Let a set A be given; its elements will be called* alternatives.

- *By an* optimality criterion $\le$ *on the set A, we mean a transitive relation (i.e., a* pre-order*) on this set.*
- *An element $a_{\rm opt}$ is called* optimal *with respect to the criterion $\le$ is for all $a \in A$, we have $a \le a_{\rm opt}$.*
- *An optimality criterion is called* final *if, with respect to this criterion, there exists exactly one optimal alternative.*

Definition 10. *For each family of basic functions $B(c_1, t)$ and for each value t_0, by its* shift $T_{t_0}(B)$*, we mean a family that assigns, to each number c_1, a function $B(c_1, t + t_0)$.*

Definition 11. *We say that an optimality criterion on the class of all families of basic functions is* shift-invariant *if for every two families B and B' and for each t_0, $B \le B'$ implies that $T_{t_0}(B) \le T_{t_0}(B')$.*

Definition 12. *For each family of basic functions $B(c_1, t)$ and for each value $\lambda > 0$, by its* scaling $S_\lambda(B)$*, we mean a family that assigns, to each number c_1, a function $B(c_1, \lambda \cdot t)$.*

Definition 13. *We say that an optimality criterion on the class of all families of basic functions is* scale-invariant *if for every two families B and B' and for each $\lambda > 0$, $B \le B'$ implies that $S_\lambda(B) \le S_\lambda(B')$.*

Now, we are ready to formulate our main result.

Proposition 2. *For every final shift- and scale-invariant optimality criterion on the class of all families of basic functions, all elements of the optimal family are spikes of the same height.*

Proof. Let us prove that the optimal family $B_{\rm opt}$ is itself shift- and scale-invariant; then this result will follow from Proposition 1.

Indeed, let us consider any transformation T – be it shift or scaling. By definition of optimality, $B_{\rm opt}$ is better than (or is of the same quality) as any other family B: $B \le B_{\rm opt}$. In particular, for every B, this is true for the family $T^{-1}(B)$, i.e., $T^{-1}(B) \le B_{\rm opt}$, where, as usual, T^{-1} denotes the inverse transformation.

Due to invariance of the optimality criterion, $T^{-1}(B) \le B_{\rm opt}$ implies that $T(T^{-1}(B)) \le T(B_{\rm opt})$, i.e., that $B \le T(B_{\rm opt})$. This is true for each family B,

thus the family $T(B_{\rm opt})$ is optimal. However, we assumed that our optimality criterion is final, which means that there is only one optimal family. Thus, we have $T(B_{\rm opt}) = B_{\rm opt}$, i.e., the optimal family $B_{\rm opt}$ is indeed invariant with respect to any of the shifts and scalings. Now, by applying Proposition 1, we conclude the proof of this proposition.

Acknowledgments

This work was supported in part by the US National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science) and HRD-1242122 (Cyber-ShARE Center of Excellence).

The authors are greatly thankful to Nikola Kasabov and to all the participants of the 2019 IEEE Series of Symposia on Computational Intelligence (Xiamen, China, December 4–6, 2019) for valuable discussions.

References

[1] P. S. Addison, *The Illustrated Wavelet Transform Handbook: Introductory Theory and Applications in Science, Engineering, Medicine and Finance*, CRC Press, Boca Raton, Florida, 2016.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

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

[4] N. K. Kasabov (ed.). *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence*, Springer Verlag, Cham, Switzerland, 2019.

[5] H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.

[6] S. K. Reed, *Cognition: Theories and Application*, Wadsworth Cengage Learning, Belmont, California, 2010.

[7] K. Weihrauch, *Computable Analysis: An Introduction*, Springer-Verlag, Berlin, Heidelberg, New York, 2000.