Kolmorogov-Arnold Networks: Why Piecewise Linear Activation Functions Work Best? Why Only Addition and Multiplication Work Well?

Martine Ceberio[1], Christoph Lauter[1 [0000−0001−7335−8220]], Vladik Kreinovich[1 [0000−0002−1244−1560]], Olga Kosheleva[2 [0000−0003−2587−4209]], Christian Servin[3 [0000−0001−9965−4386]], Jose David Diaz Roman[4], Boris Jesus Mederos Madrazo[4], and Jose Manuel Mejia Muñoz[4]

[1] Department of Computer Science, University of Texas at El Paso
500 W. University, El Paso, TX 79968, {mceberio,cqlauter,vladik}@utep.edu
[2] Department of Teacher Education, University of Texas at El Paso
500 W. University, El Paso, TX 79968, olgak@utep.edu
[3] Computer Science and Information Technology Systems Dept.
El Paso Community College, 919 Hunter, El Paso, TX 79915, USA,
cservin1@epcc.edu
[4] Institute of Engineering and Technology
Universidad Autónoma de Ciudad Juárez (UACJ), Ciudad Juárez 32310, México,
{david.roman,boris.mederos,jose.mejia}@uacj.mx

Abstract. In the traditional neural networks, for each neuron, the activation function is fixed, and learning means finding the appropriate weights. It has been recently shown that in some cases, we can enhance the effectiveness of a neural network if we also allow it to select appropriate activation function for each neuron. Such networks are known as Kolmorogov-Arnold networks (KAN) – after researchers who showed, in effect, that such networks can, in principle, exactly represent any continuous function on a bounded domain. Empirical results have shows that the best results are obtained when we use piecewise linear activation functions and when we use either the standard addition or multiplication to combine the inputs. In this paper, we provide theoretical explanation for both empirical observations.

Keywords: Kolmogorov-Arnold networks· Piecewise linear activation functions· Multiplicative and additive neural networks.

1 Formulation of the Problem

1.1 Kolmogorov-Arnold Networks (KAN): a Brief Reminder

The last decades has shown spectacular success of neural networks. A neural network consists of *neurons* – computational devices that transform inputs $z_1, \ldots, z_m$ into an output value $z = s(w_1 \cdot z_1 + \ldots + w_m \cdot z_m + w_0)$, where $s(x)$ is a function known as an *activation function*, and numbers w_i – called *weights* –

are determined during training. Some neurons directly process the measurement results (and/or other input values) $x_1, \ldots, x_n$, other neurons use, as inputs, the outputs of some other neurons. Training means funding the smallest value of the error function; this value is obtained by using gradient descent – via a method known as *backpropagation*. The output of one of the neurons is then returns to the user as the computation result $f_{NN}(x_1, \ldots, x_n)$; see, e.g., [3].

In the usual neural networks, the weights are changed during training, but the activation function remains the same. In particular, most neurons the current AI tools use the so-called Rectified Linear Units (ReLU), i.e., neurons with the activation function $s(z) = \max(0, z)$. It is known that in some applications, other activation functions lead to more effective results. To take this possibility into account, researchers have proposed to use, instead of a single activation function $s(z)$, a family of functions $s(z, C_1, \ldots, C_n)$, so that during training, we will determine not only the weights w_i, but also the parameters C_i. The values C_i are obtained by using the same backpropagation technique as finding the weights w_i.

It turned out that for many practical problems, this indeed leads to more effective and/or efficient computations; this was first shown in [9]; see also [4, 5]. (This effectiveness has been explained, e.g., in [10].)

1.2 Computational Comment

At first glance, introduction of additional parameters makes the problem more complicated, since now we have to find *both* the weights *and* the parameters describing the activation function. However, it is easy to show that this problem can be simplified: namely, that we can include weights into the list of parameters describing activation functions.

Indeed, for most neurons, each inputs comes from some previous processing. This input has, thus, the form $z_i = s(t_i, C_1, \ldots, C_n)$ for some t_i. So the contribution $Z_i \stackrel{\text{def}}{=} w_i \cdot z_i$ of this input to the linear combination z takes the form

$$Z_i = w_i \cdot s(t_i, C_1, \ldots, C_n).$$

It is therefore reasonable to include w_i as one the parameters C_{n+1}, i.e., consider a family of activation functions $C_{n+1} \cdot s(z, C_1, \ldots, C_n)$. We can similarly include the weight w_0 into one the inputs, e.g., by considering $Z_1 = w_1 \cdot s(t_i, C_1, \ldots, C_n) + w_0$ and thus, the family $C_{n+1} \cdot s(z, C_1, \ldots, C_n) + C_{n+2}$ In this interpretation, each neuron simply applies its activation function to the sum $Z_1 + \ldots + Z_m$ of all the inputs Z_i.

1.3 Biological Comment

The original idea of artificial neurons came from the desire to simulate biological neurons and thus, borrow evolution-optimized nature's solution to the problem of data processing. From this viewpoint, it is worth mentioning that, according to the latest biological studies, some biological neurons can apply different

activation functions to the their inputs – exactly as Kolmogorov-Arnold neural networks; see, e.g., [11].

1.4 Terminological Comment: Why the Name

The name of these networks come from the theorem proved by A. N. Kolmogorov and V. Arnold. They proved that every continuous function of n variables can be represented as a composition of addition and functions of one variable. This is exactly what KAN computes: it either computes the sum, or it applies some function of one variable – namely, the activation function.

Specifically, Kolmogorov and Arnold proved an even stronger statement: that every continuous function $f(x_1, \ldots, x_n)$ on a bounded domain can be represented as

$$f(x_1, \ldots, x_n) = \sum_j f_j \left(\sum_{i=1}^{n} f_{i,j}(x_i) \right), \tag{1}$$

for some continuous functions $f_i(z)$ and $f_{i,j}(z)$.

1.5 First Remaining Challenge

The original paper [9] used splines – i.e., piecewise polynomial functions. In [2], it was recently shown that the effectiveness increases if we restrict ourselves to piecewise linear activation functions. How can we explain this empirical fact?

1.6 Second Remaining Challenge

Another interesting empirical observation is related to the fact that in the original Kolmogorov-Arnold Theorem, instead of addition $a + b$, we can use any commutative association binary operation of the type $a \oplus b \stackrel{\text{def}}{=} f^{-1}(f(a) + f(b))$, where $f^{-1}(x)$ denotes the inverse function. Indeed, once we implement this operation, we can compute addition as $a + b = f(f^{-1}(a) \oplus f^{-1}(b))$, i.e., as a superposition of this combination and functions of one variable – and with addition, we can represent any continuous function. In pacticular, for $f(x) = \ln(x)$, we have $f^{-1}(x) = \exp(x)$ and thus $a \oplus b = \exp(\ln(a) + \ln(b)) = a \cdot b$. If we use different functions $f(x)$, we get different operations. It turned out that out of all such binary operations, only multiplication led to effiicent results [8]. How can we explain this?

1.7 What We Do in This Paper

In this paper, we provide explanations for both empirical facts.

Remark 1. We understand that our explanations may not be fully convincing – few explanations are – so more convincing explanations are definitely welcome. However, we believe that some explanations are better than no explanations at all.

2 Why Piecewise Linear Activation Functions Work Best?

2.1 Machine Learning Is, in Effect, Interpolation

The whole idea of machine learning is as follows. We know that the value of a quantity y is determined by the values of some other quantities $x_1, \ldots, x_n$, i.e., in mathematical terms, that $y = f(x_1, \ldots, x_n)$ for some function $f(x_1, \ldots, x_n)$, but we do not know this function. All we know are the values $y^{(k)} = f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)$ of this function in finitely many cases $k = 1, \ldots, K$. Based on this information, we want to find the values of this function for all other tuples $(x_1, \ldots, x_n)$. In numerical mathematics, this is known as *interpolation*.

Similarly, based on finitely many examples, we cannot uniquely determine the family

$$s(z, C_1, \ldots, C_n)$$

of activation functions. So, we need to use interpolation.

2.2 Which Interpolation Should We Choose

The usual criterion for accuracy in statistics and in machine learning is minimizing the meas squared deviation. So, if we know the value of an activation function $s(z)$ at several points $z_1, \ldots, z_K$ and we want to interpolate it and come up with a function $s_0(z)$ that is defined for all z, it is reasonable to select a function for which the expected value of the mean square difference is the smallest possible:

$$\int_z \int_s (s_0(z) - s(z))^2 \, ds \, dz \to \min . \tag{2}$$

To make sense of this formula, we need to select some probability measure on the set of all continuous functions.

2.3 Which Probability Measure Should We Select

The whole idea of KAN comes from the Kolmogorov-Arnold theorem, according to which any any continuous function – in particular, any function that is not differentiable – can be represented as a superposition of addition and functions of over variable. These functions cannot be all smooth (= differentiable) since superposition of differential functions is also differentiable (its derivative is described by the chain rule), and thus, we will not be able to represent all continuous functions. So, we need to consider functions $s(z)$ that are not necessarily differentiable.

What does it mean in statistical terms? Differentiable means, in effect, that for small Δz, the differences $s(z + \Delta z) - s(z)$ and $s(z) - s(z - \Delta z)$ are strongly correlated: both of these differences are close to $s'(z) \cdot \Delta z$, where, as usual s' means the derivative of the function. Since we do not want to limit ourselves

to differentiable functions, it is reasonable to require that the differences $s(z+\Delta z)-s(z)$ and $s(z)-s(z-\Delta z)$ be statistically independent.

Under this assumption, for any $a<b$ and for any large number N, we can consider points $z_i=a+i\cdot(b-a)/N$ between a and b, so that $z_0=a$ and $z_N=b$, and conclude that the difference

$$s(b)-s(a)=(s(z_1)-s(z))+(s(z_1)-s(z_1))+\ldots+(s(b)-s(z_{N-1})) \qquad (3)$$

can be represented as a sum of a large number of small independent random variables. In this case, according to the Central Limit Theorem (see, e.g., [12]), its distribution is Gaussian (normal).

It is known that, in general, the distribution of a multi-D Gaussian random vector $\eta=(\eta_1,\ldots,\eta_n)$ is uniquely determined by its means $m_i=E[\eta_i]$ and covariances $E[(\eta_i-m_i)\cdot(\eta_j-m_j)]$. In our case, we thus need to know the means values $E[s(a)]$ for all a and the corresponding covariances. Let us describe reasonable mean and covariance values.

We are discussing the distribution that would be applicable to all possible practical situations. In many practical situations – e.g., when measuring time or temperature – the choice of the starting point is just a question of convenience. We can select a different starting point which is z_0 values smaller, then all the numerical values are increased by z_0: $a\mapsto a+z_0$ and $b\mapsto b+z_0$. This change of the starting point does not change the physical situation. So, it makes sense to require that statistical characteristics also should not change under such a transformation, i.e., that we should have $E[s(a)]=E[s(a+z_0)]$ for all a and z_0. In particular, for any a and b, if we take $z_0=b-a$, we conclude that $E[s(a)]=E[s(b)]$. Thus, for all a and b, we have $E[s(b)-s(a)]=E(s(b)]-E[s(a)]=0$.

Since the mean value of the difference $s(b)-s(a)$ is 0, the variance – that we will denote by $V(a,b)$ – of this difference is simply equal to the expected value of its square $V(a,b)=E\left[(s(b)-s(a))^2\right]$. For the variance, the requirement that nothing should depend on the choice of the starting point means that for all a, b, and z_0, we should have $V(a,b)=V(a+z_0,b+z_0)$. In particular, for $z_0=-a$, we conclude that $V(a,b)=V(0,b-a)$, i.e., that $C(a,b)=v(b-a)$, where we denoted $v(z)\stackrel{\text{def}}{=}V(0,z)$.

As we have mentioned earlier, for any $a>0$ and $b>0$, the differences $s(a)-s(0)$ and $s(a+b)-s(a)$ are independent. It is known that the variance of the sum of two independent random variables is equal to the sum of their variances. So, for the sum $s(a+b)-s(0)$ of the two differences, we get

$$V(a+b,0)=V(a,0)+V(a+b,b),$$

i.e., in terms of the function $v(z)$, that

$$v(a+b)=v(a)+v(b). \qquad (4)$$

Variances are non-negative, and it is known (see, e.g., [1]) that all non-negative solutions to the equation (4) have the form $v(a)=C\cdot a$ for some $C\ge 0$. So, $V(a,b)=C\cdot(b-a)$.

Remark 2. A Gaussian random process for which the differences $s(b) - s(a)$ for all $a < b$ have mean value 0 and the variance is proportional to $b - a$ is known as a *Wiener process* or a *random walk*.

2.4 So What Is the Optimal Interpolation?

Now that we have selected a reasonable probability distribution on the set of all continuous functions, we can start answering the original question: once we know the values $t_1 = s(z_1)$, ..., $t_k = s(z_K)$, what is the optimal interpolation $s(z)$? The solution to this question is known; see, e.g., [7].

Indeed, the minimized mean square value is the sum of the values corresponding to different z; . Thus, to minimize this mean square value, it is sufficient to minimize, for each z, the expected value $E[(s_0(z) - s(z))^2]$, i.e., to find the value $s_0(z)$ for which this conditional mean square value is the smallest possible. Differentiating this expression with respect to the unknown $s_0(z)$ and equating the derivative to 0, we conclude that $2 \cdot (s_0(z) - E[s(z)]) = 0$, i.e., that $s_0(z)$ should be equal to the expected value of $s(z)$ under the condition that we have $s(z_k) = t_k$ for all k.

Specifically, let us find out what is the optimal interpolated value $s(z)$ when z is between z_k and z_{k+1}. Knowing the values t_k is equivalent to knowing the differences $d_k = s(z_{k+1}) - s(z_k)$. Based on these differences, we want to reconstruct the difference $s(z) - s(z_k)$.

Of course, the variables that are independent on the desired quantity cannot effect its value. Since differences corresponding to non-intersecting intervals are independent, the desired difference is independent of all the differences $s(z_{i+1}) - s(z_i)$ except for one of them: $s(z_{k+1}) - s(z_k)$. Thus, only this difference affects the optimal interpolation result. So, we want to find the conditional expected value $E[X|Y = c]$, where $X = s(z) - s(z_k)$, $Y = s(z_{k+1}) - s(z_k)$, and $c = d_k$.

We can solve this problem by taking into account that the conditional value of X under the condition $Y = c$ is the same as the conditional value of the linear combination $X + \beta \cdot (Y - c)$, for all possible values β. So, to use the independence argument, let us find the value β for which this combination is independent of Y. For Gaussian variables, once the covariance between them is 0, the variables are independent. So, let us find the value β for which the covariance $E[((X + \beta \cdot Y) \cdot Y]$ is equal to 0. This covariance is equal to $E[X \cdot Y] + \beta \cdot E[Y^2]$. Thus, this expression is equal to 0 when $\beta = -E[XY]/E[Y^2]$.

For this β, the linear combination is independent of Y, and thus, its conditional expected value – under the condition that Y has value c – is simply equal to the unconditional expected value of this linear combination. Since $E[X] = E[Y] = 0$, this unconditional expected value of the linear combination is equal to

$$E[X + \beta \cdot (Y - c)] = -\beta \cdot c = \frac{E[XY]}{E[Y^2]} \cdot c. \tag{5}$$

In our case, as we have mentioned earlier,

$$E[Y^2] = E\left[(s(z_{k+1}) - s(z_k))^2\right] = C \cdot (z_{k+1} - z_k), \tag{6}$$

and

$$E[X \cdot Y] = E[(s(z) - s(z_k)) \cdot (s(z_{k+1}) - s(z_k))]. \tag{7}$$

Here,

$$s(z_{k+1}) - s(z_k) = (s(z_{k+1}) - s(z)) + (s(z) - s(z_k)), \tag{8}$$

so

$$E[X \cdot Y] = E[(s(z) - s(z_k)) \cdot (s(z_{k+1}) - s(z))] + E\left[(s(z) - s(z_k))^2\right]. \tag{9}$$

The first term in the right-hand side is 0, since the corresponding intervals do not intersect. The second term is equal to $C \cdot (z - z_k)$. Thus, the formula (5) leads to the following optimal interpolation value

$$s(z) - s(z_k) = \frac{C \cdot (z - z_k)}{C \cdot (z_{k+1} - z_k)} \cdot (s(z_{k+1}) - s(z_k)), \tag{10}$$

thus

$$s_0(z) = s(z_k) + \frac{z - z_k}{z_{k+1} - z_k} \cdot (s(z_{k+1}) - s(z_k)). \tag{11}$$

This is a linear function of z.

So, indeed, the optimal interpolation leads to a function thatn is linear on each segment $[z_k, z_{k+1}]$ – i.e., to a piecewise linear function. This explains why piecewise linear activation functions are so efficient.

3 Why Only Addition and Multiplication Work Well?

Let us now explain why out of all possible combination operations, only addition and multiplication work well. To come up with such an explanation, let us consider reasonable requirements on combination operations.

3.1 First Requirement: We Want the Expression $a \oplus b$ to Be Differentiable

One of the reasons why neural networks are so successful is that backpropagation is a very fast optimization procedure. We want to be able to use this procedure, and for this purpose, we need to make sure that all our operations are differentiable.

The operation $a \oplus b$ is based on some function $f(x)$. So, to make sure that this operation is differentiable, it is reasonable to require that function $f(x)$ should be differentiable.

3.2 Scale-Invariance: Another Natural Requirement

In most computations, we process the values of the physical quantities. For example, when we predict weather, we input temperature, wind speed, humidity at different locations – and we predict the future values of these quantities at

desired locations. In a computer, all these quantities are described by numbers. It is important to take into account that the numerical value of each quantity depends on the choice of a measuring unit. For example, for the same wind speed, we get different numerical results if we measure it in kilometers per hour or in miles per hour.

In general, if we replace the original measuring unit by a new one which is λ times smaller, then all numerical values get multiplied by λ: $x \mapsto \lambda \cdot x$. For example, if we replace meter with centimeters, then 1.7 m becomes $100 \cdot 1.7 = 170$ cm. It is reasonable to require that the computations remain the same whatever measuring units we use. It makes no sense to use one algorithm for meters and a completely different one for centimeters. In particular, this means that if we re-scale a and b into $\lambda \cdot a$ and $\lambda \cdot b$, rhen then the value $(\lambda \cdot a) \oplus (\lambda \cdot b)$ should represent the same quantity as the original combination $a \oplus b$, but maybe in different units, i.e., we should have $(\lambda \cdot a) \oplus (\lambda \cdot b) = \mu(\lambda) \cdot (a \oplus b)$ for some μ depending on λ. In other words, we want the operation $a \oplus b$ to be, in this sense, scale-invariant.

For addition $a \oplus b = a + b$, we have $\lambda \cdot a + \lambda \cdot b = \lambda \cdot (a + b)$, so the scale-invariance property is true for $\mu(\lambda) = \lambda$. For multiplication $a \oplus b = a \cdot b$, we have $(\lambda \cdot a) \cdot (\lambda \cdot b) = \lambda^2 \cdot (a \cdot b)$, so the scale-invariance property holds for $\mu(\lambda) = \lambda^2$.

3.3 Definitions and the Main Result of This Section

Now we are ready to formulate (and prove) the main result of this section.

Definition 1. *Let $f(x)$ an invertible differentiable functions. Then, we denote*

$$a \oplus_f b \stackrel{\text{def}}{=} f^{-1}(f(a) + f(b)). \tag{12}$$

Definition 2. *We say that the operation $a \oplus_f b$ is* scale-invariant *if for every $\lambda > 0$ there exists a $\mu(\lambda) > 0$ for which, for all a and b, we have*

$$(\lambda \cdot a) \oplus_f (\lambda \cdot b) = \mu(\lambda) \cdot (a \oplus_f b). \tag{13}$$

Proposition 1. *The only scale-invariant differentiable operations $\oplus_f$ are:*

- *addition $a \oplus_f b = a + b$ and*
- *multiplication $a \oplus_f b \stackrel{\text{def}}{=} C \cdot a \cdot b$ for some constant C.*

Remark 3. This explains why only addition and multiplication lead to effective results.

Proof. 1°. Let us first use scale-invariance to derive an equality in terms of the derivative of the desired function $f(a)$.

Scale-invariance implies that if

$$a \oplus_f b = (a + \Delta a) \oplus_f (b + \Delta b), \tag{24}$$

then

$$(\lambda \cdot a) \oplus_f (\lambda \cdot b) = ((\lambda \cdot (a + \Delta a)) \oplus_f (\lambda \cdot (b + \Delta b)). \tag{15}$$

By definition of the operation $\oplus_f$, this means that if

$$f(a) + f(b) = f(a + \Delta a) + f(b + \Delta b), \tag{16}$$

then

$$f(\lambda \cdot a) + f(\lambda \cdot b) = f(\lambda \cdot (a + \Delta a)) + f(\lambda \cdot (b + \Delta b)). \tag{17}$$

By moving all the a-terms to one side of (16) and all the b-terms to the other side, we conclude that

$$-(f(a + \Delta a) - f(a)) = f(b + \Delta b) - f(b). \tag{18}$$

For small Δa and Δb, we get $f(a + \Delta a) = f(a) + f'(a) \cdot \Delta a + o(\Delta a)$, so (18) takes the form

$$-f'(a) \cdot \Delta a = f'(b) \cdot \Delta b + o(\Delta a), \tag{19}$$

hence, to get the desired equality (14), we need to select Δb and Δa for which the following equality holds:

$$\frac{\Delta b}{\Delta a} = -\frac{f'(b)}{f'(a)} + o(1). \tag{20}$$

For these same Δa and Δb, the equality (15) similarly leads to

$$-(f(\lambda \cdot a + \lambda \cdot \Delta a) - f(\lambda \cdot a)) = f(\lambda \cdot b + \lambda \cdot \Delta b) - f(\lambda \cdot b), \tag{21}$$

and thus, to

$$\frac{\lambda \Delta b}{\lambda \Delta a} = \frac{\Delta b}{\Delta a} = -\frac{f'(\lambda \cdot b)}{f'(\lambda \cdot a)} + o(1). \tag{22}$$

When $\Delta a \to 0$, then from (20) and (22), we can conclude that

$$\frac{f'(b)}{f'(a)} = \frac{f'(\lambda \cdot b)}{f'(\lambda \cdot a)}. \tag{23}$$

for all a, b, and λ.

2°. Let us now derive a functional equation for $f'(a)$.

For this purpose, let us separate the variables in the formula (23). We can do it if we multiply both sides by $f'(\lambda \cdot a)$ and divide both sides by $f'(b)$. Then, we get the following equality:

$$\frac{f'(\lambda \cdot a)}{f'(a)} = \frac{f'(\lambda \cdot b)}{f'(b)}. \tag{24}$$

The fact that this equality holds for all a and b means that the ratio on each side of the equality does not depend on a, it depends only on λ; let us denote it by

$$r(\lambda) = \frac{f'(\lambda \cdot a)}{f'(a)}. \tag{25}$$

Multiplying both sides by $f'(a)$, we conclude that for all a and $\lambda > 0$, we have:

$$f'(\lambda \cdot a) = r(\lambda) \cdot f'(a). \tag{26}$$

3°. Let us now solve the resulting functional equation for $f'(a)$ and find possible functions $f(a)$.

The function $f(a)$ is differentiable, its derivative $f'(a)$ is measurable. Thus the ratio $r(\lambda)$ is also measurable. It is known – see, e.g., [1] – that every measurable solution to the functional equation (26) has the form $f'(a) = c_1 \cdot x^d$ for some real numbers c_1 and d. Here, $c_n \neq 0$ – since otherwise the function $f(a)$ will be constant, but we assumed that this function is invertible.

Integrating the derivative $f'(a)$, we conclude that:

- for $d \neq -1$, we have
$$f(a) = c_2 \cdot a^\alpha + c_0, \tag{27}$$
where $c_2 = c_1/(d+1)$, $\alpha = d+1$, and c_0 is an integration constant;
- for $d = -1$, we have
$$f(a) = c_1 \cdot \ln(a) + c_0, \tag{28}$$
where c_0 is an integration constant.

After some preliminary analysis, let us consider these two cases one by one.

4°. Let us first perform some preliminary analysis of the scale-invariance condition.

By definition (12), the condition $a \oplus_f b = c$ means that $f^{-1}(f(a)+f(b)) = c$. If we apply the function $f(z)$ to both sides of this equality, we conclude that

$$f(a) + f(b) = f(c). \tag{29}$$

Similarly, the condition (13) means that

$$f(\lambda \cdot a) + f(\lambda \cdot b) = f(\lambda \cdot c). \tag{30}$$

Thus, scale-invariance means that for all a, b, and $\lambda > 0$, (29) implies (30).

5°. Let us now consider the case when $d \neq -1$ and $f(a)$ is described by the formula (27).

In this case, (29) means that

$$c_2 \cdot a^\alpha + c_0 + c_2 \cdot b^\alpha + c_0 = c_2 \cdot c^\alpha + c_0, \tag{31}$$

i.e., equivalently,

$$c_2 \cdot a^\alpha + c_2 \cdot b^\alpha - c_2 \cdot c^\alpha = -c_0, \tag{32}$$

and (30) takes the form

$$c_2 \cdot (\lambda \cdot a)^\alpha + c_0 + c_2 \cdot (\lambda \cdot b)^\alpha + c_0 = c_2 \cdot (\lambda \cdot c)^\alpha + c_0, \tag{33}$$

i.e., equivalently,

$$c_2 \cdot \lambda^\alpha \cdot a^\alpha + c_2 \cdot \lambda^\alpha \cdot b^\alpha - c_2 \cdot \lambda^\alpha \cdot c^\alpha = -c_0. \tag{34}$$

Multiplying both sides of (32) by λ^α, we conclude that

$$c_2 \cdot \lambda^\alpha \cdot a^\alpha + c_2 \cdot \lambda^\alpha \cdot b^\alpha - c_2 \cdot \lambda^\alpha \cdot c^\alpha = -\lambda^\alpha \cdot c_0. \tag{35}$$

The left-hand sides of (34) and (35) are the same, so the right-hand sides must be equal, and we must have $-c_0 = -\lambda^\alpha \cdot c_0$. For $\lambda \neq 1$, this implies that $c_0 = 0$. So, the formula (31) takes the form

$$c_2 \cdot a^\alpha + c_2 \cdot b^\alpha = c_2 \cdot c^\alpha. \tag{36}$$

If we divide both sides by $c_2 \neq 0$, then we get $c^\alpha = a^\alpha + b^\alpha$, i.e.,

$$a \oplus_f b = (a^\alpha + b^\alpha)^{1/\alpha}. \tag{37}$$

Let us now take into account that the function $a \oplus_f b$ must be differentiable. If we use the chain rule to differentiate the expression (37) with respect to a, we conclude that

$$\frac{\partial}{\partial a}\left[(a^\alpha + b^\alpha)^{1/\alpha}\right] = \frac{1}{\alpha} \cdot (a^\alpha + b^\alpha)^{1/\alpha - 1} \cdot \alpha \cdot a^{\alpha - 1}. \tag{38}$$

Let us simplify this formula. We can represent $a^\alpha + b^\alpha$ as

$$a^\alpha + b^\alpha = a^\alpha \cdot \left(1 + \left(\frac{b}{a}\right)^\alpha\right). \tag{39}$$

Thus,

$$(a^\alpha + b^\alpha)^{1/\alpha - 1} = (a^\alpha)^{1/\alpha - 1} \cdot \left(1 + \left(\frac{b}{a}\right)^\alpha\right)^{1/\alpha - 1}. \tag{40}$$

Here,

$$(a^\alpha)^{1/\alpha - 1} = a^{\alpha \cdot (1/\alpha - 1)} = a^{1-\alpha}, \tag{41}$$

so

$$(a^\alpha + b^\alpha)^{1/\alpha - 1} = a^{1-\alpha} \cdot \left(1 + \left(\frac{b}{a}\right)^\alpha\right)^{1/\alpha - 1}. \tag{42}$$

Substituting (42) into the formula (38) and taking into account that the factors $1/\alpha$ and α cancel each other, we conclude that

$$\frac{\partial}{\partial a}\left[(a^\alpha + b^\alpha)^{1/\alpha}\right] = a^{1-\alpha} \cdot \left(1 + \left(\frac{b}{a}\right)^\alpha\right)^{1/\alpha - 1} \cdot a^{\alpha - 1}. \tag{43}$$

Here, $a^{1-\alpha} \cdot a^{\alpha - 1} = 1$, so

$$\frac{\partial}{\partial a}\left[(a^\alpha + b^\alpha)^{1/\alpha}\right] = \left(1 + \left(\frac{b}{a}\right)^\alpha\right)^{1/\alpha - 1}. \tag{44}$$

When a and b are close to 0, we should get close to the derivative of the function $a \oplus_f b$ for $a = b = 0$. However, when $\alpha \neq 1$, we have $1/\alpha - 1 \neq 0$ and thus, we will gave different expressions for different ratios b/a. So, the only case when the operation $a \oplus_f b$ is differentiable everywhere – in particular, for $a = b = 0$ – is when $\alpha = 1$. In this case, the formula (37) turns into $c = a + b$.

6°. Finally, let us consider the case when $d = -1$, and the function $f(a)$ is described by the formula (28).

In this case, (29) means that for $c = a \oplus_f b$, we have

$$c_1 \cdot \ln(a) + c_0 + c_1 \cdot \ln(b) = c_1 \cdot \ln(c) + c_0. \tag{45}$$

If we divide both sides by $c_1 \neq 0$ and move the term c_0/c_1 into the left-hand side, we conclude that

$$c = \ln(a) + \ln(b) + c_0/c_2. \tag{46}$$

Applying $\exp(z)$ to both sides, we conclude that $c = C \cdot a \cdot b$, where we denoted $C \stackrel{\text{def}}{=} \exp(c_0/c_1)$.

The proposition is proven.

Acknowledgments. This work was supported by the AT&T Fellowship in Information Technology, by the Institute for Risk and Reliability, Leibniz Universitaet Hannover, Germany, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Focus Program SPP 100+ 2388, Grant Nr. 501624329, by the European Union under the project ROBOPROX (No. CZ.02.01.01/00/22 008/0004590), by the Center of Excellence in Econometrics, Faculty of Economics, Chiang Mai University, Thailand, by the Ho Chi Minh City University of Banking, Vietnam, and by Thang Long University, Hanoi, Vietnam.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aczél, J., Dhombres, J.: Functional Equations in Several Variables. Cambridge University Press, Cambridge, Massachusetts (2008)
2. Bouarah, R., Barbe, V., Volkova, A. de Dinechin, F.: KANHard: Training Hardware-Friendly Kolmogorov–Arnold Networks, Technical Report, https://hal.science/hal-05388006v1 (2025)
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, Massachusetts (2016)
4. Hadj Kilani, B.: Convolutional Kolmogorov–Arnold Networks: A Survey, Technical Report, https://hal.science/hal-05177765 (2025)
5. Hou, Y., Ji, T., Zhang, D., Stefanidis, A.: Kolmogorov-Arnold Networks: A Critical Assessment of Claims, Performance, and Practical Viability. arXiv: 2407.11075, https://arxiv.org/abs/2407.11075 (2025)
6. Kolmogorov, A. N.: On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Dokl. Akad. Nauk SSSR **114**, 369–373 (1957)

7. Kreinovich, V.: Applying Wiener measure to calculations. In: Proceedings of the 3rd Leningrad Conference of Young Mathematicians, pp. 6–9, Leningrad, USSR, in Russian (1974)
8. Liu, Z., Ma1, P., Wang, Y., Matusik1, W., Tegmark, M.: KAN 2.0: Kolmogorov-Arnold Networks Meet Science, arXiv: 2408.10205, https://arxiv.org/abs/2408.10205 (2024)
9. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., Tegmark, M.: KAN: Kolmogorov-Arnold Networks, arXiv: 2404.19756, https://arxiv.org/abs/2404.19756(2024, updated version 2025)
10. Nguyen, H. T., Kreinovich, V., Kosheleva, O.: Why Kolmolgorov-Arnold Networks (KAN) work so well: a qualitative explanation. In: Phuong, N. H., Chau, N. T. H., Kreinovich V. (eds.), Explainable Artificial Intelligence and Other Soft Computing Techniques: Biomedical and Related Applications, Springer, Cham, Switzerland (to appear)
11. Paganos, P.: Single-nucleus profiling highlights the all-brain echinoderm nervous system, Science Advances **11**:45, Paper adx7753, DOI: 10.1126/sciadv.adx7753 (2025)
12. Sheskin, D. J.: Handbook of Parametric and Nonparametric Statistical Procedures. Chapman and Hall/CRC, Boca Raton, Florida (2011)