Tropical Backpropagation

Özgür Ceyhan
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
Luxembourg
ozgur.ceyhan@uni.lu

Federico Lucchetti
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
Luxembourg federico.lucchetti@uni.lu

Abstract

This work introduces tropicalization, a novel technique that delivers tropical neural networks as tropical limits of deep ReLU networks. Tropicalization transfers the initial weights from real numbers to those in the tropical semiring while maintaining the underlying graph of the network. After verifying that tropicalization will not affect the classification capacity of deep neural networks, this study introduces a tropical reformulation of backpropagation via tropical linear algebra. Tropical arithmetic replaces multiplication operations in the network with additions and addition operations with $\max$, and therefore, theoretically, reduces the algorithmic complexity during the training and inference phase. We demonstrate the latter by simulating the tensor multiplication underlying the feed-forward process of state-of-the-art trained neural network architectures and compare the standard forward pass of the models with the tropical ones. Our benchmark results show that tropicalization speeds up inference by 50 $\%$. Hence, we conclude that tropicalization bears the potential to reduce the training times of large neural networks drastically.

1 Introduction

Scientists and engineers often use logarithmic scales to distinguish between data subjected to different power laws. For instance, the change of coordinates,

$$(x, y) \mapsto (u, v) =: (\ln(x), \ln(y))$$

expands the first quadrant $\mathbb{R}^2_+$ to the whole plane $\mathbb{R}^2$ while transforming the curve defined by $y = ax^k$ into the straight line $v = ku + b$ with slope k and $a = e^b$, which easily differentiates the curves for different powers of k independently of the scale.

Tropicalization is a formalisation of this simple premise: we first replace the polynomial $p(x) = \sum_k a_k x^k$ (with $a_k \geq 0$, for now) with one-parameter family $p_\hbar(x) = \sum_k a_k^{1/\hbar} x^k$ and $\hbar > 0$. The graph of the logarithmic image $\hbar \ln(p_\hbar(x))$ forms a smooth family for $\hbar > 0$, and more importantly, the graphs tend towards the graph of piecewise linear functions

$$M_p(x) = \max_k (ku + b_k) \quad \text{where} \quad a_k = e^{b_k} \tag{1}$$

in the C^0 sense as $\hbar \to 0$. In other words, *taking the limit $\hbar \to 0$ transforms the polynomial geometry into piecewise linear geometry while preserving its topological properties*. This limit is called *the*

Preprint. Under review.

tropical limit. The effective use of tropicalization can be traced back to the 1980s and Oleg Viro's construction of real algebraic varieties with the prescribed topology. Over the last decade, tropical algebraic geometry has become a fully-fledged research field in itself (see (IMS) for an overview).

1.1 Tropical neural networks at first sight: Related studies

Even at first glance, the tropical limit (1) suggests an intimate connection with feedforward ReLU networks. This similarity triggered an interest in tropical versions of neural networks, particularly in their feedforward properties and transferring specific combinatorial/geometric tools. However, this also restricted the research to the diegesis of the tropical limit itself and overlooked the limiting process and, therefore, its implications.

Initial studies focused only on the equivalence between the tropical rational maps and the tropical neural networks. Among other things, a characterization of feedforward ReLU neural networks in terms of polytopes (ZNL18), and the number of regions which these networks can distinguish have been investigated (see (CM18) and also (MCT21)).

The interest in tropicalization in the context of neural networks has quickly widened into several areas, including localisation of spoofing attacks (TM18), the convex regression problem (MT19), the lottery ticket hypothesis, the generation of adversarial attacks (ABHGG), cardiac diagnostics (YDGZAGN). The training of feedforward neural networks in a tropical setup was studied in (Cey21).

1.2 Contributions

Building on the origins of tropicalization (Vi01), we develop a general framework for analyzing the correspondence between deep ReLU networks and their tropical limits. We set the generic classification problem in terms of geometric properties of a deep network and consider them in a one-parameter family. These intermediary models, depending on the free parameter $\hbar$, can realize the same classification problem without touching the underlying combinatorics of the network. More importantly, these models show that the tropical limit has the same classification capacity. This fundamental property follows from the continuity of the tropical limit.

Our description of tropical networks as a limit yields a significant improvement over previous results, as it enables the transfer of backpropagation into the tropical realm. Tropical backpropagation approximates a tropical network that is as capable. Moreover, tropical backpropagation has algorithmic advantages with almost no drawbacks. Tropicalization reduces the algorithmic complexity by eliminating multiplications in favor of addition and maximum.

In summary:

- We develop a general framework for analyzing the correspondence between the deep ReLU networks and their tropical limits.
- We set the generic classification problem in terms of geometric properties of a DNN and consider them in a one-parameter family. Here we show that, in the tropical limit, the classification capacity remains unchanged.
- We show how tropicalization reduces the algorithmic complexity by eliminating multiplications and put forward the tropical formulation of the backpropagation algorithm.
- We demonstrate the latter in a set of benchmark performance tests.

2 Classification problem, its one-parameter family, and backpropagation

In this section, we set the generic classification problem in terms of geometric properties of a smooth function, and then give its extension to a one-parameter family preserving all essential properties. This one-parameter family will be the foundation for the transition towards tropical neural networks and tropical backpropagation, which we present in §3.

2.1 Classification via the level sets of a smooth function

Consider a smooth map $f : \mathbb{R}^n_+ \to \mathbb{R}_+$, defined over the cone $\mathbb{R}^n_+ = (0, \infty)^n$, and the following *level sets*

$$Z_f(k) := \{\mathbf{x} \in \mathbb{R}^n_+ \mid e^k \leq f(\mathbf{x}) < e^{k+1}\}, \forall k \in \mathbb{Z}$$

of f. We define the map

$$F : \mathbb{R}^n_+ \to \mathbb{Z} : \mathbf{x} \mapsto k, \quad \forall \mathbf{x} \in Z_f(k).$$

Given an arbitrary finite data set Ω in $\mathbb{R}^n_+$, we can formulate a *classification problem*

$$N : \Omega \to \{1, \ldots, M\} \tag{2}$$

as the problem of finding a smooth map f which assigns the elements of Ω in the same class into the same level set $Z_f(k)$. That is, for $\mathbf{x}, \mathbf{y} \in \Omega$,

$$N(\mathbf{x}) = N(\mathbf{y}) \iff F(\mathbf{x}) = F(\mathbf{y}).$$

Clearly, this reformulation requires that the image $f(\mathbb{R}^n_+)$ contains at least $[0, M + 1)$ so that F can realise the classification problem (2).

2.2 Backpropagation in a nutshell

Multilayered feedforward neural networks using ReLU as activation function can produce arbitrarily close approximations for any continuous and piecewise smooth function on any compact domain in $\mathbb{R}^n$, see for instance (Sp64; Cy89; H91). Therefore, we can use the approximations of f discussed in §2.1 in order to implement a solution of the classification problem (2).

Consider a multi-layered ReLU network realising the function

$$f_{\mathbf{W}} : \mathbb{R}^n_+ \to \mathbb{R}. \tag{3}$$

Such a function, by its defitinion, is the iterative composition of finite linear sums

$$\sum \sigma(b^{(k)}_i + \sum_{j=1}^{n} w^{(k)}_{ij} x_j), \quad \sigma(x) = \max(0, x) \tag{4}$$

each realising the connections between *the layers* $k - 1$ and k with *the weight matrix* $\mathbf{W}^{(k)} := \left[w^{(k)}_{ij}\right]$, and *the bias vectors* $\mathbf{b}^{(k)} = \left[b^{(k)}_i\right]$. *The backpropagation algorithm* aims at minimising a predetermined *error function*, e.g.,

$$E : \mathbb{R}^{|\mathbf{W}|} \times \Omega_{tr} \to \mathbb{R} : (\mathbf{W}, \mathbf{x}_{tr}) \mapsto f_{\mathbf{W}}(\mathbf{x}_{tr}) - f(\mathbf{x}_{tr}) \tag{5}$$

which measures the difference between the value of the targeted function f and its approximation $f_{\mathbf{W}}$, over the space of weights $\mathbb{R}^{|\mathbf{W}|} := \{\mathbf{W}^{(k)} \mid k = 1, \ldots, L\}$ and the set of finite training data $\Omega_{tr} := \{\mathbf{x}_{tr,j}, j = 1, \ldots, T\}$. By following an iterative gradient descent procedure, one calculates the gradient matrix

$$(\nabla E)^{(k)} = \left[\frac{\partial E}{\partial w^{(k)}_{ij}}\right](\mathbf{x}_{tr}), \forall\, k \,\&\, \mathbf{x}_{tr}$$

for training data, and adjusts the weight matrix $\mathbf{W}^{(k)}$ by adding a correction term $\Delta\mathbf{W}^{(k)} = -\epsilon\nabla^{(k)}E$ in order to minimise the error function. The backpropagated error on the k-th layer is computed via an iterative matrix multiplication

$$\Delta\mathbf{W}^{(k)} = \mathbf{D}^{(k)}\mathbf{W}^{(k+1)} \cdots \mathbf{D}^{(l)}\mathbf{W}^{(l+1)}\mathbf{e} \tag{6}$$

where for each layer k, the matrix $\mathbf{D}^{(k)}$ is the diagonal matrix composed of the derivatives of the activation function with respect to its arguments, and the vector $\mathbf{e}$ contains the derivatives of the output errors with respect to the arguments of the last layer. Depending on the size and the depth of the ReLU network, this algorithm produces a function $f_{\mathbf{W}}$ that remains as close as possible to the targeted function f on the training set Ω_{tr}. For details, see for instance (Ro96, §7.3.3) or (DBBNG, §6.5). For the deep ReLU networks, these approximations work effectively (see (PV17; Ya16)).

2.3 Classification problem in a one-parameter family, its realisation and backpropagation

The function $f_{\mathbf{W}}$ realised by a multi-layered ReLU network (3) is a composition of linear functions and $\max$, therefore it is a sum of linear monomials

$$a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)} x_i \tag{7}$$

in their definition domains in $\mathbb{R}^n_+$. The coefficients $a_{jk}^{(m)}$ are either the weights $w_{jk}^{(m)}$ or the biases $b_i^{(m)}$ in (4).

Any real polynomial $f_{\mathbf{W}}$ is a difference $f^+_{\mathbf{W}} - f^-_{\mathbf{W}}$ of polynomials with positive coefficients. Consider now a one-parameter family of ReLU network

$$f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar} : \mathbb{R}^n_+ \to \mathbb{R}, \tag{8}$$

realised by the deep ReLU with the same combinatorial network structure, but with new weights and biases;

$$(w_{ij}^{(m)})^{1/\hbar}, \quad \& \quad (b_i^{(m)})^{1/\hbar}, \quad \hbar \in [0,1]. \tag{9}$$

Thus, the monomials in (7) can be replaced as follows;

$$a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)} x_i \mapsto \begin{cases} (a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)})^{1/h} x_i & \text{if } a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)} > 0, \text{ i.e., in } f^+_{\mathbf{W}}, \\ -(a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)})^{1/h} x_i & \text{if } a_{ef}^{(L)} \cdots a_{jk}^{(2)} a_{ij}^{(1)} < 0, \text{ i.e., in } f^-_{\mathbf{W}}. \end{cases}$$

Remark 2.3.1 *Note that the separation of positive and negative coefficients, that is, $f^{\pm}_{\mathbf{W}}$ and $f^{\pm}_{\mathbf{W},\hbar}$, is essential to keep these monomials well defined without choosing a non-trivial branch of the logarithm.*

Lemma 2.3.2 *For $\hbar \in (0,1]$, the level sets $Z_{f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}}(k/\hbar)$ form a one-parameter smooth family and, it specializes to the level set $Z_{f_{\mathbf{W}}}(k)$ when $\hbar = 1$.*

The above statement follows directly from the definition of the one parameter family $f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}$, that is, when $\hbar = 1$, one can directly calculate the limit $f^+_{\mathbf{W},\mathbf{1}} - f^-_{\mathbf{W},\mathbf{1}} = f_{\mathbf{W}}$ as $\lim_{\hbar\to 1}(w_{ij}^{(m)})^{1/\hbar} = w_{ij}^{(m)}$ and $\lim_{\hbar\to 1}(b_i^{(m)})^{1/\hbar} = b_i^{(m)}$.

Moreover, it allows us define the error function

$$E_\hbar := f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar} - f,$$

and provide a one-parameter family of solutions via the gradient matrix of $E_\hbar$, i.e.,

$$\Delta \mathbf{W}_\hbar^{(k)} = \mathbf{D}_\hbar^{(k)} \mathbf{W}_\hbar^{(k+1)} \cdots \mathbf{D}_\hbar^{(l)} \mathbf{W}_\hbar^{(l+1)} \mathbf{e}_\hbar. \tag{10}$$

While the introduction of the parameter $\hbar$ in (8) and the backpropagation (10) extends our target (the classification in (2)) to a family of solutions, its limit $\hbar \to 0$ does not exists in this form as the exponents $1/\hbar$ in (9) blow up in that limit. However, this limit can be reinstated via rescaling and taking logarithmic image of the family $f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}$, which takes us to the realm of tropical geometry.

3 Tropical neural networks and tropical backpropagation

In this section, we introduce the basic notions in tropical arithmetics and tropical geometry, which we then use to introduce the tropical neural networks as universal approximators as well as a tropical version of the backpropagation algorithm.

3.1 Tropical semiring: Arithmetic without multiplication

The *tropical semiring* is the limit $\hbar \to 0$ of the continuous family of semirings $S_\hbar := (\mathbb{R}_+, \oplus_\hbar, \otimes_\hbar)$ defined by the following arithmetic operations; for $a, b \in \mathbb{R}_+$,

$$\begin{aligned} a \oplus_\hbar b &:= \begin{cases} \hbar \ln(e^{a/\hbar} + e^{b/\hbar}) & \text{when } \hbar > 0, \\ \max\{a, b\} & \text{when } \hbar \to 0, \end{cases} \\ a \otimes_\hbar b &:= \hbar \ln(e^{a/\hbar} \cdot e^{b/\hbar}) = a + b. \end{aligned} \tag{11}$$

For $\hbar > 0$, the logarithmic map $D_\hbar$

$$D_\hbar : \mathbb{R}_+ \to \mathbb{R} : x \mapsto \hbar \ln(x) \tag{12}$$

is a semiring isomorphism, i.e., $D_\hbar(a+b) = D_\hbar(a) \oplus_\hbar D_\hbar(b)$ and $D_\hbar(a \cdot b) = D_\hbar(a) \otimes_\hbar D_\hbar(b)$. Therefore $S_\hbar$ for $\hbar > 0$ can be considered as a copy of $(\mathbb{R}_+, +, \times)$ with the usual operations of addition and multiplication[1].

The family $S_\hbar$ relates the ordinary addition and multiplication operations on the set of real numbers with the tropical arithmetic in the limit. This limiting process is also called the *Maslov dequantization*, see, for instance (IMS; Vi01). The tropical limit S_0 admits a tropical division, however, the subtraction is impossible due to the idempotency of $\oplus_\hbar$, i.e. $x \oplus_\infty x = \max\{x, x\} = x$.

3.2 Classification problem in the tropical limit

For $\hbar > 0$, we define the level sets $Z_{f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}}(k/\hbar)$ in terms of the solutions of the equations

$$f^+_{\mathbf{W},\hbar}(\mathbf{x}) = f^-_{\mathbf{W},\hbar}(\mathbf{x}) + e^{k/\hbar}.$$

In order to make sense of its limit $\hbar \to 0$, we simply consider its image under the map $D_\hbar$.

Proposition 3.2.1 *The level sets defined by $D_\hbar(f^+_{\mathbf{W},\hbar}(\mathbf{x})) = D_\hbar(f^-_{\mathbf{W},\hbar}(\mathbf{x}) + e^{k/\hbar})$ are all diffeomorphic for small enough $\hbar$.*

We prove this proposition by showing that all those level sets for $\hbar \in [0, \epsilon)$ for small ϵ are homeomorphic to the level sets at the limit $\hbar \to 0$. Moreover, the correspondence is smooth for $\hbar > 0$.

This statement essentially requires, for a function $f_\hbar : \mathbb{R}^n_+ \to \mathbb{R}_+ : \mathbf{x} \mapsto \sum (a_i)^{1/\hbar} x_i + a_0^{1/\hbar}$ with positive real coefficients $a_i = e^{b_i}$, to compare the log graph f and the maximum $m_{f_\hbar}(\mathbf{x}) = \max((a_1)^{1/\hbar} x_1, \ldots, (a_n)^{1/\hbar} x_n, (a_0)^{1/\hbar})$. Set $x_i = e^{u_i/\hbar}$ and consider the limit

$$\begin{aligned}
\lim_{\hbar \to 0} D_\hbar(f_\hbar) &= \lim_{\hbar \to 0} \hbar \ln(e^{(u_1+b_1)/\hbar} + \cdots + e^{(u_n+b_n)/\hbar} + e^{b_0/\hbar}) \\
&= \lim_{\hbar \to 0} \hbar \ln(\max(e^{(u_1+b_1)/\hbar}, \ldots, e^{(u_n+b_n)/\hbar}, e^{b_0/\hbar})) \\
&= \lim_{\hbar \to 0} \max(\hbar \ln e^{(u_1+b_1)/\hbar}, \ldots, \hbar \ln e^{(u_n+b_n)/\hbar}, \hbar \ln e^{b_0/\hbar}) \\
&= \max(a_1 x_1, \ldots, a_n x_n, a_0).
\end{aligned}$$

The passage from the first line to the second follows from the fact that, in the limit $\hbar \to 0$, the function $e^{a/\hbar} + e^{b/\hbar}$ is dominated by $e^{a/\hbar}$ if $a > b$. We use the monotonicity of $\ln$ to pass to the third line, and the final result follows from the definition of the variables.

We conclude the homeomorphic nature of limiting and nearby level sets simply by computing the derivative,

$$\lim_{\hbar \to 0} \frac{d}{d\hbar} D_\hbar(f_\hbar) = 0$$

which essentially dictates that the level set $Z_{f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}}(k/\hbar)$ for small enough $\hbar$ is topologically the same as the limit set at the limit $\hbar \to 0$.

Proposition 3.2.2 *The correspondence in Proposition 3.2.1 transforms the realisation of the classification problem* (2)

3.3 Tropicalisation of backpropagation

Proposition 3.2.2 states that we can apply the map $D_\hbar$ (12) to each step of the backpropagation algorithm (10) in one-parameter family and take the limit $\hbar \to 0$ to define *the tropical version of the backpropagation algorithm*. This provides us a different realization of the original classification problem (2) via different level sets as stated above (via preserving the topology of the level sets).

[1]The parameter $\hbar$ is not just the reminiscent of the Planck constant. The tropical limit $\hbar \to 0$ is essentially the quasi-classical (i.e., zero temperature) limit of a certain model in quantum mechanics.

	ReLU network	Log image of ReLU network in family
	$f_{\mathbf{W}} : \mathbb{R}_+^n \to \mathbb{R}_+$	$D_\hbar(f^+_{\mathbf{W}}), D_\hbar(f^-_{\mathbf{W}} + e^{k/\hbar}) : \mathbb{R}_+^n \to \mathbb{R}$
Monomials	$a^{(L)}_{ef} \cdots a^{(2)}_{jk} a^{(1)}_{ij} x_i$	$c^{(L)}_{ef} \cdots c^{(2)}_{jk} c^{(1)}_{ij} x_i$ where $a^{(*)}_{**} = e^{c^{(*)}_{**}/\hbar}$
Level sets	$Z_{f_{\mathbf{W}}}(k)$	Homeomorphic to $Z_{f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}}(k/\hbar)\ \forall\ \hbar << 1$
Classification	$\mathbf{x} \mapsto k,\ \forall\ \mathbf{x} \in Z_f(k)$	$\mathbf{x} \mapsto k,\ \forall \mathbf{x} \in Z_{f^+_{\mathbf{W},\hbar} - f^-_{\mathbf{W},\hbar}}(k/\hbar)$

While the classification problem in one parameter family preserves its nature, the computational properties of backpropagation change in the limit $\hbar \to 0$.

The transition to the limit $\hbar \to 0$ is accomplished by taking the tropical image of each entry of the matrices, and then replacing the matrix addition and multiplication by their respective tropical arithmetic operations $\oplus_\infty$ and $\otimes_\infty$ in (11):

Let $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [B_{ij}]$ be $n \times m$ matrices. The *tropical matrix sum*, $\mathbf{A} \oplus_\infty \mathbf{B}$, is then obtained by evaluating the tropical sum $\oplus_\infty$ in (11) of the corresponding matrix entries as follows;

$$(\mathbf{A} \oplus_\infty \mathbf{B})_{ij} := a_{ij} \oplus_\infty b_{ij} = \max\{a_{ij}, b_{ij}\}.$$

The *tropical multiplication* $\mathbf{A} \otimes_\infty \mathbf{B}$ *of two matrices* $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{m \times n}$ and $\mathbf{B} = [b_{ij}] \in \mathbb{R}^{n \times p}$ is given by the matrix $\mathbf{C} = [c_{ij}] \in \mathbb{R}^{m \times p}$ with entries

$$\mathbf{C}_{ij} := \oplus_\infty (a_{ik} \otimes_\infty b_{kj}) = \max_k \{a_{ik} + b_{kj}\}. \tag{13}$$

It is calculated using tropical arithmetic operations $\oplus_\infty$ and $\otimes_\infty$ in (11).

By using the image of the error term (5) under $\lim_{\hbar \to 0} D_\hbar$ and the tropical linear algebra defined above, we formulate *the tropical gradient descent* iteration as follows

$$\begin{aligned} \mathbf{W}^{(k)}_{\text{new}} &= \mathbf{W}^{(k)} \oplus_\infty \Delta \mathbf{W}^{(k)} \\ &= \mathbf{W}^{(k)} \oplus_\infty -\epsilon \left(\mathbf{D}^{(k)} \otimes_\infty \mathbf{W}^{(k+1)} \otimes_\infty \cdots \otimes_\infty \mathbf{D}^{(l)} \otimes_\infty \mathbf{W}^{(l+1)} \otimes_\infty \mathbf{e} \right). \end{aligned} \tag{14}$$

Propositions 3.2.1 and 3.2.2 allow us to conclude that

Corollary 3.3.1 *The tropical backpropagation algorithm in* (14) *approximates to a deep ReLU network which is as capable as the function* $f_{\mathbf{W}}$*, produced by the regular backpropagation* (6) *in solving the classification problem stated in* (2)*.*

4 Evaluation

In this section, we show the performance gains from tropicalization.

Custom Implementation. To demonstrate the performance gain from tropical tensor multiplication, we implemented a custom matrix multiplication using a CUDA kernel in Python and compiled it with Numba (NUMBA)[2]. Moreover, we wrote a blocked algorithm and leveraged the availability of CUDA's fast shared memory in order to parallelize blocks and cooperatively distribute the computational task to multiple threads. This task served as skeleton for both the standard and the tropic matrix multiplication. We fixed the number of threads per block to 32 for both approaches.

Simulations. We limited our evaluation to the inference stage in which, for simplicity, we simulated the feed-forward process of state-of-the-art DNN architectures. The latter are typically composed of stacked feed-forward single layer perceptron (SLP) and convolutional neural network (CNN) layers, both of which heavily rely on multidimensional tensor multiplications. Input data shapes range from $(10, 10)$ to $(299, 299, 3)$. Each CNN layer is composed of between 16 to 512 filters, and performs matrix multiplication using a kernel shape of 3x3 or 5x5. The number of neurons inside a SLP range from 16 to 512 and are mainly stacked in the classifier part of neural-network models, usually

[2]All source code and benchmark scripts are provided in the additional materials.

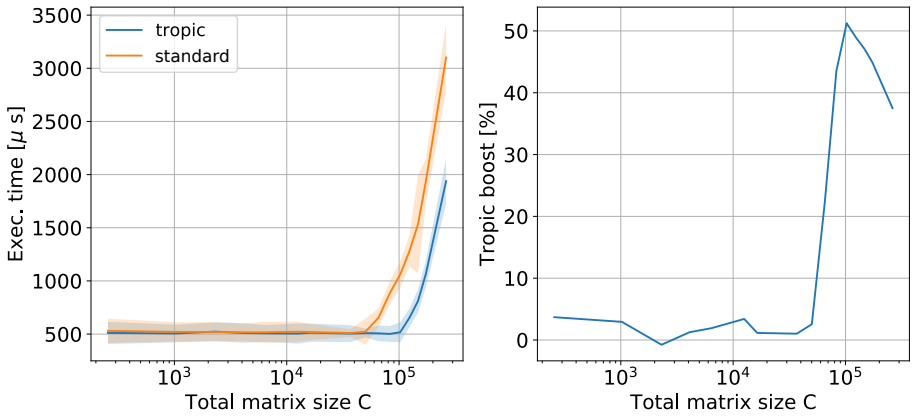

Figure 1: Performance results comparing standard and tropic matrix multiplication for different matrix sizes. Top figure shows absolute execution times where the shaded filled areas depict one standard deviation. Bottom figure shows boost due to the tropicalization.

accepting flattened 1D arrays of sizes ranging from 10 to 10000. Hence, the feed-forward process of SLPs can involve a significant amount of matrix multiplications up to (1,50000)x(50000,512), resulting in an algorithmic complexity of $O(10^6)$. We denote by $A \cdot B$ the matrix multiplication and define the total multiplication size (proportional to the complexity) $C = N^3$ where (N, N), (N, N) are the shapes of A and B respectively. For a fixed value of C, we executed 100000 single-precision floating-point operations. Values for C ranged from 32 to 512 with step sizes equal to the number of threads per block (32).

Hardware. We evaluated our benchmarks on a a 8-core AMD Ryzen 7 3700X with 64GB RAM memory and a NVIDIA RTX3090 graphic processing unit, running Ubuntu 20.04.

Results. Figure 1 shows the results of our benchmarks. Absolute execution times for both approaches stay low for relatively small matrix sizes but then quickly increase. Comparatively, performance boost due to the tropicalization stays below 6 $\%$ for small matrix sizes below $4 \cdot 10^4$, but quickly increases up to 50 $\%$ after this limit.

5 Discussions and Conclusions

5.1 Algorithmic complexity of backpropagation: Standard vs Tropical

The algorithmic complexity of the standard backpropagation computations in (6) has two components: (1) The complexity of calculating the matrix product. (2) The complexity of the arithmetic operations involved in the matrix multiplications.

The standard multiplication of matrices of size $n \times m$ and $m \times p$ has the algorithmic complexity of $O(nmp)$. Any simplification in the structure of the network will decrease the algorithmic complexity as the sizes of the resulting matrices will decrease.[3,4] Therefore, simplifying the network structure would be one of the strategies to decrease the algorithmic complexity as the number of nonzero matrix entries decreases. However, it would require changes in the code.

The tropicalization of DNNs does not touch the combinatorial structure of the network itself, it only changes the underlying arithmetic with the following effect on computational complexity.

[3] One can design smaller and more robust neural networks via advanced pruning techniques, see for example (HMD15; LKDSG; TF97; YCS16).

[4] Even though there are matrix multiplication algorithms with better asymptotic complexities (see, for instance, (Ga12)), they come with a large constant factors in their running times.

5.2 Complexities of real and tropical arithmetics

As part of accelerating matrix multiplication, both the vector operation $c_{i,j} = \sum_k a_{i,k} b_{k_j}$ and the embedded multiply-accumulate operation received several optimization steps at hardware level. For example, while addition amounts to $O(n)$ complexity for n-bits, solutions such as various forms of carry look-ahead adders reduced this complexity to $O(log(n))$. Similar optimizations are applicable for taking the maximum (with constant complexity on average and $O(log(n))$ in general) and for multiplication (e.g., from $O(n^2)$ to $O(n^{\log_2(3)})$ for Karatsuba algorithm (Ka95)). However, the general trend remains: $O(n^\lambda)$ with $\lambda > 1$ for multiplication versus $O(log(n))$ for a respective radix implementation of addition and maximum.

Lower algorithmic complexities are achieved only asymptotically, and they are usually effective only for integer arithmetic. While neural networks are often used in combination with 8-bit integer arithmetic, this entails a loss of precision that may not be acceptable in all application scenarios. We conclude that

Corollary 5.2.1 *The tropical backpropagation algorithm in* (14) *has a lower algorithmic complexity relative to the backpropagation algorithm* (6) *on the ring of real numbers.*

5.3 Execution time and energy consumption

The actual execution time of a specific code depends on numerous factors, such as the processor speed, the instruction set, disk speed, and the compiler used. An old rule of thumb in designing numerical experiments that dictates avoiding multiplications and divisions in simulations in favor of additions and subtractions in order to improve the actual execution time may heuristically seem redundant on modern processors as they closed the time-cost-gap between addition and multiplication drastically. However, one can still observe the advantages in carefully designed setups, see Figure 1.

The energy consumption does not favor multiplications over summations either: the required energy for multiplication is always considerably higher than the summation (Hor14, pg. 32). Both execution time and energy consumption measures suggest that the reduction of algorithmic complexity has considerable benefits.

It should further be noted that similar instruction combinations are possible with maximum-addition as we have seen for multiply-accumulate and that all traversal and cache optimizations that apply to the original matrix, still hold for its tropical variant.

5.4 Final remarks

This work laid out the mathematical framework of tropical neural networks and their training via backpropagation. After demonstrating theoretically that the tropicalization decreases the computational complexity while preserving the training objectives, we then proceeded by simulating the forward pass of a DNN using standard and tropical matrix multiplications in various sizes. We confirmed that the tropicalization does indeed benefit matrix multiplications in terms of drastic improvement in processing speed hence we can extrapolate with confidence the same conclusion to the forward pass and consequentially to the training of DNNs.

References

[ABHGG] M. Alfarra, A. Bibi, H. Hammoud, M. Gaafar, B. Ghanem *On the Decision Boundaries of Neural Networks: A Tropical Geometry Perspective.* `https://arxiv.org/abs/2002.08838`.

[Cey21] Ö. Ceyhan *Algorithmic Complexities in Backpropagation and Tropical Neural Networks.* `https://arxiv.org/abs/2101.00717`.

[CM18] V. Charisopoulos, P. Maragos. *A Tropical Approach to Neural Networks with Piecewise Linear Activations.* (2018), `https://arxiv.org/abs/1805.08749`.

[Cy89] G. Cybenko, *Approximations by superpositions of sigmoidal functions.* Mathematics of Control, Signals, and Systems (1989), 2 (4),303–314.

[DBBNG] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, R. Garcia, *Incorporating Second-Order Functional Knowledge for Better Option Pricing.* Advances in Neural Information Processing Systems 13, MIT Press, 2001, 472–478.

[Ga12] F. Le Gall, *Faster algorithms for rectangular matrix multiplication*, Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012), pp. 514–523.

[HMD15] S. Han, H. Mao, W.J. Dally, *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.* `https://arxiv.org/abs/1510.00149`

[H91] K. Hornik, *Approximation Capabilities of Multilayer Feedforward Networks.* Neural Networks, 4(2) (1991), 251–257.

[Hor14] M. Horowitz, *1.1 computing's energy problem (and what we can do about it).* Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014.

[IMS] I. Itenberg, G. Mikhalkin, E.I. Shustin, *Tropical Algebraic Geometry.* Oberwolfach Series Birkhäuser; 2 edition (2009).

[Ka95] A.A. Karatsuba, *The Complexity of Computations*. Proceedings of the Steklov Institute of Mathematics. 211 (1995): 16–183.

[LKDSG] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, *Pruning Filters for Efficient ConvNets.* `https://arxiv.org/abs/1608.08710`

[MCT21] P. Maragos, V. Charisopoulos, E. Theodosis, *Tropical Geometry and Machine Learning,* Proceedings of the IEEE, pp. 728-755, vol. 109, no. 5, 2021, doi: 10.1109/JPROC.2021.3065238.

[NUMBA] S.K. Lam, A. Pitrou, & S. Seibert, (2015). *Numba: A llvm-based python jit compiler.* In Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC (pp. 1–6).

[MT19] P. Maragos, E. Theodosis, *Tropical Geometry and Piecewise-Linear Approximation of Curves and Surfaces on Weighted Lattices*, `https://arxiv.org/abs/1912.03891`.

[PV17] P. Petersen, F. Voigtlaender, *Optimal approximation of piecewise smooth functions using deep ReLU neural networks.* `https://arxiv.org/abs/1709.05289v3`

[Ro96] R. Rojas, *Neural Networks: A Systematic Introduction.* Springer, 1996.

[Ru16] S. Ruder, *An overview of gradient descent optimization algorithms.* `https://arxiv.org/pdf/1609.04747.pdf`.

[ST19] G. Smyrnis, P. Maragos *Tropical Polynomial Division and Neural Networks*, `https://arxiv.org/abs/1911.12922`.

[Sp64] D. Sprecher, *On the Structure of Continuous Functions of Several Variables*, Transactions of the American Mathematical Society, Vol. 115 (1964), pp. 340–355.

[TM18] E. Theodosis, P. Maragos, *An Adaptive Pruning Algorithm for Spoofing Localisation Based on Tropical Geometry*, `https://arxiv.org/abs/1811.01017`.

[TF97] G. Thimm, E. Fissler, *Pruning of neural networks.* `http://publications.idiap.ch/downloads/reports/1997/rr97-03.pdf`

[Vi01] O. Viro, *Dequantization of Real Algebraic Geometry on a Logarithmic Paper.* Proceedings of the 3rd European Congress of Mathematicians, Birkhäuser, Progress in Math, 201, (2001), 135–146.

[YDGZAGN] H. Yao, H. Derksen, J.R. Golbus, J. Zhang, K.D. Aaronson, J. Gryak, K. Najarian, *A Novel Tropical Geometry-based Interpretable Machine Learning Method: Application in Prognosis of Advanced Heart Failure.* `https://arxiv.org/abs/2112.05071`

[Ya16] D. Yarotsky, *Error bounds for approximations with deep ReLU networks.* `https://arxiv.org/abs/1610.01145`

[YCS16] T.J. Yang, Y.H. Chen, V. Sze, *Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning.* https://arxiv.org/abs/1611.05128

[ZNL18] L. Zhang, G. Naitzat, L.H. Lim, *Tropical Geometry of Deep Neural Networks.* Proceedings of the 35th International Conference on Machine Learning, PMLR 80:5824-5832, 2018. https://arxiv.org/abs/1805.07091