# MAXPOLYNOMIAL DIVISION WITH APPLICATION TO NEURAL NETWORK SIMPLIFICATION

*Georgios Smyrnis*[1,2] *Petros Maragos*[1,2] *George Retsinas*[1]

[1]School of ECE, National Technical University of Athens, 15773 Athens, Greece
[2]Robot Perception and Interaction Unit, Athena Research Center, 15125 Maroussi, Greece
geosmirnis@gmail.com, maragos@cs.ntua.gr, gretsinas@central.ntua.gr

## ABSTRACT

In this work, we further the link between neural networks with piecewise linear activations and tropical algebra. To that end, we introduce the process of Maxpolynomial Division, a geometric method which simulates division of polynomials in the max-plus semiring, while highlighting its key properties and noting its connection to neural networks. Afterwards, we generalize this method and apply it in the context of neural network minimization, for two-layer networks used for binary classification problems, attempting to reduce the size of the hidden layer before the output. A tractable method to find an appropriate divisor and perform the division is introduced and evaluated in the IMDB Movie Review and MNIST datasets, with preliminary experiments demonstrating a capacity of this method to reduce the size of the network, without major loss of performance.

***Index Terms***— Tropical polynomials, tropical algebra, neural network minimization

## 1. INTRODUCTION

The fields of minimax algebra [1] and tropical geometry [2], which can be used to refer to the study of either the max-plus semiring $(\mathbb{R} \cup \{-\infty\}, \max, +)$ or its dual min-plus version, are fields of mathematics with applications in a variety of domains, such as the analysis of dynamic systems [3], [4], [5] and optimization [6], [7]. The importance of these max-plus models (which we shall refer to as tropical, for the rest of this work) has been further demonstrated by recent studies [8], [9], [10], [11], which link them with the building blocks of neural networks, in particular those with piecewise linear activations, demonstrating a profound connection between the two.

With the above context in mind, it is apparent that further study in the max-plus nature of these networks can help obtain insight in their workings. Such understanding may also help in furthering efforts in the simplification of large networks for a given task, the size of which is often required during their training, but might later be safely pruned for faster and more compact networks [12], [13], [14].

**Related Work.** Regarding the application of ideas from tropical geometry to the analysis of neural networks, [9] and [10] demonstrate that the output functions of a neural network with piecewise linear activations can be described via polynomials in the max-plus semiring [15], henceforth referred to as tropical polynomials or maxpolynomials. This description allows them to arrive at similar results, regarding the number of linear regions of a neural network map, as [16]. Moreover, [8] and [11] have demonstrated the use of morphological perceptrons, which rely on max-plus operations instead of normal inner products with the weights of the neuron, demonstrating particular ease when pruning such a network, while also linking these perceptrons with maxout networks [17]. Finally, [18] and [19] demonstrate the use of Log-Sum-Exp networks, which are linked with Geometric Programming [20], in approximating data as a difference of convex functions, similar to how maxpolynomials can be used to describe the output of a neural network.

Concerning the minimization of a trained neural network, the act of pruning an already trained, fully connected network using conventional methods has long been studied [21]. Recent work has advanced further, by proposing various methods by which a network can be pruned, and by expanding it in the context of convolutional neural networks. In particular, in [13] pruning is done by selecting filters with the least important output, while in [14] by stochastically removing connections between neurons, and subsequently neurons themselves, with both of these works showing remarkable results.

**Contributions.** This work advances the aforementioned study of the connection between neural networks and tropical geometry, while also providing theoretical contributions by geometrically defining an approximate method for Maxpolynomial Division, providing an algorithm for its computation and demonstrating its link with neural networks. Note that, while the problem of factoring tropical polynomials has already been studied in the one-dimensional case [15], [22], this work presents a novel method to extend it into multiple dimensions, in order to create an approximate representation similar to that of the division of regular polynomials. So far only the exact case, provided it is feasible, has been studied [23]. Afterwards, this method is applied to the problem of minimizing the hidden layer of a two-layer, fully connected network with one output neuron, trained for a binary classification problem, in order to demonstrate one of its possible applications, while also performing some preliminary experiments to prove its validity. This contribution strengthens the link between tropical algebra and neural networks, to better understand the inner workings of the latter.

The rest of this work is structured as follows: In Section 2, we shall define Maxpolynomial Division, and highlight its key properties. In Section 3, we link this method with neural networks and demonstrate its application in minimizing a two-layer network, trained for a binary classification problem. Finally, in Section 4, we shall demonstrate the results of some initial experiments.

## 2. DIVISION OF TROPICAL POLYNOMIALS

In what follows, we shall provide a method to approximately divide a tropical polynomial $p(\boldsymbol{x}) = \max_{i=1}^{k} \left(\boldsymbol{a}_i^T \boldsymbol{x} + b_i\right)$, $\boldsymbol{x} \in \mathbb{R}^d$, by another tropical polynomial $d(\boldsymbol{x}) = \max_{i=1}^{k} \left(\tilde{\boldsymbol{a}}_i{}^T \boldsymbol{x} + \tilde{b}_i\right)$. Our algorithm outputs two tropical polynomials $q(\boldsymbol{x})$, $r(\boldsymbol{x})$, which are

Proceedings of the 45th International Conference on Acoustics, Speech and Signal Processing (ICASSP-2020), May 2020.

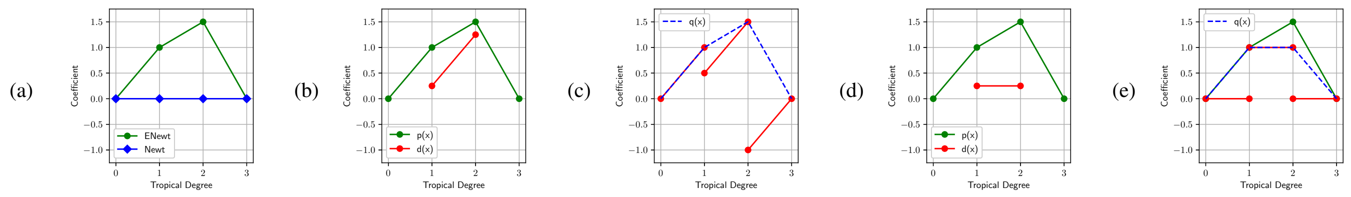

**Fig. 1**. (a): (E)Newton Polytope of max(3x,2x+1.5,x+1,0). (b),(c): Division by max(x+1,0), (d),(e): Division by max(x,0)

maximal (in a sense that will be later elaborated), for which:

$$p(\boldsymbol{x}) \geq \max\left(q(\boldsymbol{x}) + d(\boldsymbol{x}), r(\boldsymbol{x})\right), \ \forall \boldsymbol{x} \in \mathbb{R}^{\mathrm{d}} \tag{1}$$

**Definition 1** ([8]). *Let* $p(\boldsymbol{x}) = \max_{i=1}^{k}\left(\boldsymbol{a_i}^T\boldsymbol{x} + b_i\right)$, $\boldsymbol{x} \in \mathbb{R}^d$ *be a tropical polynomial. Its* ***Newton Polytope*** $\mathrm{Newt}(p)$ *is the convex hull of the set* $\mathrm{slopes}(p) = \{\boldsymbol{a}_i : i = 1, 2, \dots k\}$ *of slope vectors (tropical degrees of the terms), while its* ***Extended Newton Polytope*** $\mathrm{ENewt}(p)$ *is the convex hull of* $\{(\boldsymbol{a}_i, b_i) : i = 1, 2, \dots k\}$.

An example in 1 dimension can be seen in Fig. 1a. It is well-known [8], [9], [10] that a tropical polynomial $p$ is defined as a function only by the terms corresponding to vertices on the upper faces of $\mathrm{ENewt}(p)$. Hence, we assume that all points on the upper faces of $\mathrm{ENewt}(p)$ correspond to terms of the polynomial $p$, an assumption which does not change the polynomial as a function.

**Maxpolynomial Division Algorithm.** Let $p(\boldsymbol{x})$, $d(\boldsymbol{x})$ be two tropical polynomials. The algorithm to divide $p(\boldsymbol{x})$ by $d(\boldsymbol{x})$ is:

1. Let $C \subseteq \mathbb{Z}^d$ be the set of slopes $\boldsymbol{c}$ with which we can shift $\mathrm{Newt}(d)$, so that $\mathrm{Newt}\left(\boldsymbol{c}^T\boldsymbol{x} + d(\boldsymbol{x})\right) \subseteq \mathrm{Newt}(p)$.
2. For every element $\boldsymbol{c} \in C$, define $q_{\boldsymbol{c}} \in \mathbb{R}$ as the largest value $q$ for which $\mathrm{ENewt}(p)$ is higher (with respect to the last dimension) than $\mathrm{ENewt}\left(q + \boldsymbol{c}^T\boldsymbol{x} + d(\boldsymbol{x})\right)$, or equivalently $p(\boldsymbol{x}) \geq q + \boldsymbol{c}^T\boldsymbol{x} + d(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathbb{R}^d$.
3. Output the maxpolynomials $q(\boldsymbol{x}) = \max_{\boldsymbol{c} \in C}\left(q_{\boldsymbol{c}} + \boldsymbol{c}^T\boldsymbol{x}\right)$ and $r(\boldsymbol{x}) = \max_{t(\boldsymbol{x}) \in T}(t(\boldsymbol{x}))$, where $T$ is the set of terms $\boldsymbol{a}_j^T\boldsymbol{x} + b_j$ of $p(\boldsymbol{x})$ for which there is no value $\boldsymbol{c}$ such that $\boldsymbol{a}_j \in \mathrm{Newt}\left(\boldsymbol{c}^T\boldsymbol{x} + d(\boldsymbol{x})\right)$. As in classic polynomial division, we shall refer to $q(\boldsymbol{x})$ as the *quotient*, and $r(\boldsymbol{x})$ as the *remainder* of the division of $p(\boldsymbol{x})$ by $d(\boldsymbol{x})$.

*Remark* 1. The above algorithm is equivalent to the morphological opening of the upper hull of $\mathrm{ENewt}(p)$, which is equivalent to a function $n_p : \mathbb{Z}^d \to \mathbb{R} \cup \{-\infty\}$ mapping tropical degrees to their coefficients, by the upper hull of $\mathrm{ENewt}(d)$ (similarly $n_d : \mathbb{Z}^d \to \mathbb{R} \cup \{-\infty\}$). This operation [24], [25] is the composition of an erosion, and a dilation using the same element. This process can be seen in Fig. 1b-1e, where the divisor is shifted and heightened as much as possible, to match the dividend, which is exactly how our algorithm operates. Due to this equivalence, we can process the elements of $C$ in parallel, since there is no cancellation of terms caused by the order of calculations.

**Theorem 1.** *For any tropical polynomials* $p(\boldsymbol{x}), d(\boldsymbol{x})$*, the outputs* $q(\boldsymbol{x}), r(\boldsymbol{x})$ *of our algorithm satisfy (1).*

Indeed, the upper hull of $\mathrm{ENewt}\left(q(\boldsymbol{x}) + d(\boldsymbol{x})\right)$ lies strictly below that of $\mathrm{ENewt}(p)$, and since the maximum of two tropical polynomials has an Extended Newton Polytope that is the convex hull of the union of the respective polytopes [9], its upper faces will be exactly those of $\mathrm{ENewt}(p)$, thus $\max\left(q(\boldsymbol{x}) + d(\boldsymbol{x}), p(\boldsymbol{x})\right) = p(\boldsymbol{x})$. A similar result can be obtained for the remainder.

**Theorem 2.** *The tropical polynomials* $q(\boldsymbol{x}), r(\boldsymbol{x})$ *are maximal, in the sense that for any other two tropical polynomials* $\tilde{q}(\boldsymbol{x})$, $\tilde{r}(\boldsymbol{x})$*, for which (1) holds, containing terms of the same degree as* $q(\boldsymbol{x})$ *and* $r(\boldsymbol{x})$*, respectively, the value* $\max\left(q(\boldsymbol{x}) + d(\boldsymbol{x}), r(\boldsymbol{x})\right)$ *is maximum among* $\max\left(\tilde{q}(\boldsymbol{x}) + d(\boldsymbol{x}), \tilde{r}(\boldsymbol{x})\right)$.

This is derived by the fact that $q_{\boldsymbol{c}} \geq \tilde{q}_{\boldsymbol{c}}$, for any given degree of the polynomials, along with the fact that $r(\boldsymbol{x})$ contains only terms of $p(\boldsymbol{x})$, so there can be no larger remainder satisfying (1).

The above algorithm can also be extended, in the case of multiple divisor polynomials, by simply performing the division on each of these separately, again due to there being no cancellation of terms. Theoretical details on all of the above subjects can be seen in [26].

**Examples.** Let $p(x) = \max(3x, 2x + 1.5, x + 1, 0)$ and $d(x) = \max(x + 1, 0)$. The valid set of degrees for the quotient is $C = \{0, 1, 2\}$. For $c = 1$, we see that for $q_1 + x + \max(x + 1, 0)$, the optimum can be found for $q_1 = 0.5$, when the right vertex of the divisor coincides with the third vertex of the dividend. Similarly we find $q_0 = 0, q_2 = -1$. Thus we get $q(x) = \max(2x - 1, x + 0.5, 0)$. We can verify that $p(x) = q(x) + d(x)$, as in Fig. 1c. However, if $d(x) = \max(x, 0)$, then for $c = 1$ we have $q_1 + x + \max(x, 0)$. Thus $q_1 = 1$, leading to one of the vertices of the dividend being higher than the result (Fig. 1e).

**Formulation via GGP.** If the coefficients of $p(\boldsymbol{x})$ are positive (which can always be done by adding a large enough value to the dividend, and subtracting it from the result), maxpolynomial division can also be formulated as a Generalized Geometric Programming (GGP) problem [20]. Indeed, the quotient of maxpolynomial division can also be found by solving the following GGP, over the coefficients $q_{\boldsymbol{c}}$ of $q(\boldsymbol{x})$, and the auxiliary variables $l_{\boldsymbol{c}} > 0$, $\forall \boldsymbol{c} \in C$:

$$\min_{q_{\boldsymbol{c}}, l_{\boldsymbol{c}}} \ \sum\nolimits_{\boldsymbol{c} \in C}\left\{l_{\boldsymbol{c}}^{-1}\right\}, \ \text{s.t.} \quad \begin{aligned} l_{\boldsymbol{c}} q_{\boldsymbol{c}}^{-1} &\leq 1, \ \forall \boldsymbol{c} \in C \\ (q \oplus d)_{\boldsymbol{j}} p_{\boldsymbol{j}}^{-1} &\leq 1, \ \forall \boldsymbol{j} \in \mathrm{slopes}(p) \end{aligned} \tag{2}$$

where $\mathrm{slopes}(p)$ is the set of slope vectors $\boldsymbol{j}$ of $p(\boldsymbol{x})$, and $p_{\boldsymbol{j}}$ its coefficients. We also define $(q \oplus d)_{\boldsymbol{j}} = \max_{\boldsymbol{c} \in C}\left(q_{\boldsymbol{c}} + d_{\boldsymbol{j}-\boldsymbol{c}}\right)$ as the max-plus convolution of the coefficients $d_{\boldsymbol{i}}$ of $d(\boldsymbol{x})$ and $q_{\boldsymbol{c}}$ of $q(\boldsymbol{x})$, similar to how the coefficients of regular polynomial products are computed. The above problem is a GGP, as the goal and both types of inequality constraints are generalized posynomials [20] of the variables $l_{\boldsymbol{c}}$ and $q_{\boldsymbol{c}}$. It can be proven [26] that the solution to (2) is the same as that of our algorithm, since the lower bounds $l_{\boldsymbol{c}}$ push the shifted versions of $d(\boldsymbol{x})$ as close as possible to $p(\boldsymbol{x})$.

## 3. APPLICATION IN NEURAL NETWORKS

Let us now consider networks containing one hidden layer with ReLU activations, and an output layer with one neuron, with sigmoid activation, used to solve a binary classification problem. As seen in [10], the network as a function, without the sigmoid activation, is equivalent to the difference of two tropical polynomials, corresponding to the positive and negative weights of the output neuron, each with a zonotope as its Extended Newton Polytope (also seen in [9]). The bias of the output neuron can be ignored during the division algorithm, simply by being added to the result.

**Divisibility of Neural Network Polynomials.** For the networks studied, the following holds:

**Theorem 3.** *Each of the two tropical polynomials which construct the function of our neural network can be divided exactly by the tropical polynomial corresponding to any of the line segments from which it is constructed.*

This follows immediately from the formulation of the neural network, as the sum of small tropical polynomials, and the maximal nature of the algorithm's results. This indicates the link of the operation we introduced, with the building blocks of a neural network.

**Approximation with Loss Minimization.** A network as described above, assuming that it was properly trained, will have weights which correspond to a local optimum of the loss function $L(u)$, where $u$ is the output of the network, without the final activation. A change in these weights will lead to a different $u'$ for this particular input, which is expected to strictly increase the loss function, when computed over the whole dataset. Thus, this increase is minimized, when $u' \approx u$. In other works [12], [13], this minimization of the difference of feature maps is among a list of criteria by which irrelevant neurons are pruned. Given that these terms correspond to vertices of the related polytopes, it is also possible to regard this criterion, in the context of tropical polynomial division.

In particular, we seek to adapt the division algorithm, so that it minimizes the aforementioned difference. This can be formulated as the below optimization problem, where $p(\boldsymbol{x})$ is the polynomial of the positive or the negative part of the network, and $D$ a dataset:

$$\min_{q_{\boldsymbol{c}}, \boldsymbol{c} \in C} \sum_{\boldsymbol{x} \in D} |q(\boldsymbol{x}) + d(\boldsymbol{x}) - p(\boldsymbol{x})|^2 \tag{3}$$

This optimization problem is a max-linear fitting problem [27], where the function to approximate is $p(\boldsymbol{x}) - d(\boldsymbol{x})$. Thus, given a divisor $d(\boldsymbol{x})$, the algorithm described in that work can be used (where the data points are alternatively assigned to terms of $q(\boldsymbol{x})$, and then each term is fit via least-squares), to calculate its solution. Note that here we attempt to approximate the output of the network, using a given structure provided by the polynomial $d(\boldsymbol{x})$, instead of direct optimization of a given feature map as in [12].

An alternative optimization goal, more in line with the process of tropical division described, would be the difference between the coefficients of the same degrees of the original polynomial and its approximation, that is, to minimize $\sum_{\boldsymbol{i} \in \text{slopes}(p)} |(q \oplus d)_{\boldsymbol{i}} - p_{\boldsymbol{i}}|^2$. This is examined in [26], with similar results.

**Practical Application.** To avoid solving the above, possibly difficult, optimization problems (with output variables scaling exponentially with $d$), we propose the following tractable method, to find a suboptimal solution to the optimization problem (3), to approximate a given network of the above form. We disregard whether the weights are integers or real numbers, since, given a finite representation of the latter, the difference is minimal [9], [10].

**Neural Network Approximation.** Let $p_+$, $p_-$ be the positive and negative parts of the network we want to approximate with a smaller one consisting of $f\%$ neurons of the original in the hidden layer, where $f$ a desired percentage. The algorithm used consists of two phases.

**In the first phase:**

1. Randomly sample a subset $X$ of the training set.
2. For each $\boldsymbol{x}_i \in X$ calculate the corresponding vertex $\boldsymbol{u}_+^i$, $\boldsymbol{u}_-^i$ in $\text{ENewt}(p_+)$ and $\text{ENewt}(p_-)$, activated by this input, that is, $(\boldsymbol{u}_+^i)^T \boldsymbol{x}_i = p_+(\boldsymbol{x}_i)$, similarly for $\boldsymbol{u}_-^i$.
3. Sort each set of vertices in decreasing number of appearances.
4. For the set of $\boldsymbol{u}_+^i$, set the first neuron weight $\boldsymbol{w}_1$ equal to the first vertex in the sorted list.
5. For the jth vertex $\boldsymbol{u}_+^j$ in the list, up to $f\%$ of the neurons in the positive part, randomly pick one of the previous weights $\boldsymbol{w}_k$, and set $\boldsymbol{w}_j \leftarrow \boldsymbol{u}_+^j - \boldsymbol{w}_k$.
6. Repeat steps 4. and 5. for the negative part of the network.
7. Keep the weights created ($f\%$ of the original) as the first layer of the network, and assign weights $\pm 1$ in the output layer, for the positive and negative parts, respectively.

**In the second phase:**

8. For each sample in $X$, calculate the activation of the output neuron in the original network, minus the same activation in the approximation network.
9. Set a bias in the output neuron, equal to the mean of these values, plus the original bias of the output neuron.

Phase 1 creates a network which attempts to match the most important vertices of the polytope, and does so perfectly if they remain on its upper faces. It also seeks to identify correlated neurons, which tend to fire together, intuitively leading to some level of regularization of the network. As for the idea behind phase 2, we show that:

**Theorem 4.** *If $d(\boldsymbol{x})$ matches the degrees of the terms of $p(\boldsymbol{x})$, then $q(\boldsymbol{x}) = q_{\boldsymbol{0}}$, and the goal in (3) can be minimized if we set:*

$$q_{\boldsymbol{0}} = \frac{1}{|X|} \sum_{\boldsymbol{x} \in X} (p(\boldsymbol{x}) - d(\boldsymbol{x})) \tag{4}$$

Indeed, if the quotient polynomial becomes a constant, then the goal in (3) becomes a sum of squared distances from given points, which is minimized by setting their mean as that constant. Thus, phase 2 adds a bias to account for the rest of the vertices.

Regarding the time complexity of the algorithm, the number of steps for phase 1 scales as $|X| \log |X| + N$ (where $N$ the number of remaining neurons), since there will be $|X|$ vertices to sort at most. The number of steps required for phase 2 scales linearly in $|X|$. Assuming a reasonable amount of time necessary for the calculation of the network outputs for these samples, the method is tractable. It might also be faster than training a new network, which requires updates scaling as $|D|$ per epoch, for each of the $N$ neurons.

This process may be applied to DNNs, possibly via iterating over the layers. This, however, is beyond the scope of this work.

## 4. EXPERIMENTS

We will now test our method, first in the IMDB Movie Review dataset [28], with the preprocessing proposed, while also truncating or padding each sequence to a length of 200, and then in the MNIST dataset [29], where only the pairs 3-5 and 4-9 are considered (even in a highly accurate model like [30], these can still be confused). We made use of the Python library Keras [31], to create and test our models. The accuracies for our method are averages for five runs.

**IMDB Movie Review Dataset.** In this experiment, we examined 3 different models, trained using an 80%-20% training-validation split, with early stopping based on the validation loss with a patience of 2 epochs. The learning rate was set to $5 \cdot 10^{-4}$. All three models learned a word embedding in a 50-dimensional space. After this, the first model contained a simple fully connected network, with one hidden layer containing 1024 neurons. The second and the third model employed a bidirectional LSTM with 32

| Percentage of Neurons Kept | FC (1024) | | LSTM + FC (256) | | 1D CNN + FC (256) | |
|---|---|---|---|---|---|---|
| 100% (Original) | 84.056±0.750 | | 85.006±0.353 | | 83.927±0.378 | |
| | Reduced | Baseline | Reduced | Baseline | Reduced | Baseline |
| 75% | 84.263±0.348 | 83.852 | 85.108±0.220 | 85.033 | 84.059±0.374 | 84.354 |
| 50% | 84.314±0.322 | 84.232 | 85.029±0.374 | 85.166 | 84.006±0.378 | 83.882 |
| 25% | 84.366±0.344 | 83.970 | 85.092±0.295 | 85.214 | 83.930±0.323 | 84.400 |
| 10% | 84.333±0.396 | 85.021 | 85.126±0.204 | 85.042 | 84.042±0.412 | 83.964 |
| 2% | 84.314±0.413 | 85.254 | 84.878±0.447 | 83.254 | 83.922±0.363 | 82.428 |
| 1% | 84.206±0.418 | 85.346 | 85.017±0.350 | 82.947 | 83.915±0.359 | 78.758 |

**Table 1**. Average accuracy on test set (IMDB). Baseline is discarding original model and training new one from scratch.

units and a one-dimensional CNN (two layers of 8 and 4 units, ReLU activations, kernels of 3 pixels and max-pooling of 5 pixels), respectively, creating a smaller representation as input for the fully connected part, with 256 neurons in the hidden layer.

Our algorithm was applied using 2000 points from the training set. This number was chosen to demonstrate that, in this simple case, only a few samples are required for good results. Various percentages of the neurons of the hidden fully connected layer were kept, and the relevant results are shown in Table 1, which also contains results for freshly trained networks, for each number of neurons (this is a relatively strict baseline, since the network is allowed to adapt itself freely). The drop in performance is less than 0.5%, a negligible cost for reducing the size of the fully connected part of the network by a factor of up to 100. Thus, minimization via tropical polynomial division methods appears capable of producing a neural network with similar performance, but significantly fewer terms.

Furthermore, the baseline seems better for the first model, for a small number of neurons, possibly due to the large input in the fully connected part making the problem easy, no matter the number of neurons. However, for the other two models, the accuracy of a new network is lower when only a few neurons are kept (note that for the last two rows of these models, to help them converge, early stopping patience and learning rate were set to $5$ and $10^{-4}$, respectively). This may be due to the small input of the fully connected part, making the classes entangled. Thus, our method seems useful if there is a small number of features, benefiting from a higher dimensional projection, before minimization. Moreover, Table 2 shows that our method is faster than training all different models, thus making it preferable over training and fine-tuning them, to attain similar performance.

| Model | Runtime of our method (sec) | Training time for all baseline nets. (sec) |
|---|---|---|
| FC | 82.6±4.6 | 97.2 |
| LSTM+FC | 113.4±0.1 | 630.2 |
| CNN+FC | 17.8±0.8 | 142.3 |

**Table 2**. Runtime comparison.

Finally, we create an iterative version of our algorithm. In each step, the amount of neurons in the hidden layer is halved, using our method, but the two fully connected layers are retrained for 10 epochs (with early stopping with patience 2) after each step, to compensate for any drops in accuracy. Results can be seen in Table 3, for the first and the third model (the second one seems to have little margin for improvement). This is, in general, a slight improvement over Table 1, since retraining the network alleviates mistakes made in its approximation, with a more detailed retraining phase possibly leading to better results. Furthermore, the same observations as before

| Neurons Kept | FC (1024) | CNN+FC (256) |
|---|---|---|
| 100% | 84.056±0.750 | 83.927±0.378 |
| 50% - Iter. 1 | 84.334±0.376 | 84.080±0.459 |
| 25% - Iter. 2 | 84.379±0.273 | 84.094±0.484 |
| 3.1% - Iter. 5 | 84.235±0.311 | 84.049±0.558 |
| 1.6% - Iter. 6 | 84.299±0.262 | 84.054±0.579 |

**Table 3**. Average accuracy, iterative method with retraining.

can be made, regarding the comparison with the baseline networks.

**MNIST Dataset - Pairs 3-5, 4-9.** Here, our model consists of the two convolutional layers (this time, with max-pooling of 3) and two fully connected layers, with a hidden layer of 1000 neurons. This way, we can again learn a small representation for our data. The results are in Table 4, along with those of training a fresh network. It can be seen that the performance drop caused by our method

| Perc. Kept | 3-5, Reduced | 3-5, Basel. | 4-9, Reduced | 4-9, Basel. |
|---|---|---|---|---|
| 100% | 99.180±0.269 | - | 99.046±0.264 | - |
| 50% | 99.106±0.438 | 99.127 | 99.046±0.282 | 99.056 |
| 25% | 99.117±0.370 | 99.138 | 98.985±0.337 | 98.905 |
| 10% | 99.106±0.375 | 99.190 | 99.005±0.309 | 99.166 |
| 1% | 99.180±0.325 | 99.138 | 98.805±0.371 | 98.945 |

**Table 4**. Average accuracy on test set (MNIST).

is again negligible, in experiments for both pairs. Even if the same results can be obtained by training a smaller network, our method has the same benefits as before, given that it allows us to begin with a bigger network, which is more likely to converge, and obtain adequate results, without the cost of training several networks.

## 5. CONCLUSIONS AND FUTURE WORK

In this work, we made theoretical contributions in the link between tropical geometry and neural networks, by constructing a framework for maxpolynomial division, and making use of the intuition it provides in order to simplify two-layer fully connected networks trained for binary classification problems, with preliminary results being favorable for its potential application. The application of this method in problems with multiple classes is an ongoing work, along with using the theoretical aspects of this work to define similar methods for more general networks, such as deep or convolutional models. Moreover, further study of the division of polynomials in the max-plus semiring, using the concept introduced in this work, must be done, in order to identify more of its properties.

## 6. REFERENCES

[1] R. Cuninghame-Green, *Minimax Algebra*, Springer-Verlag, 1979.

[2] D. Maclagan and B. Sturmfels, *Introduction to Tropical Geometry*, Amer. Math. Soc., 2015.

[3] P. Butkovič, *Max-linear Systems: Theory and Algorithms*, Springer, 2010.

[4] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*, J. Wiley & Sons, 1992, web ed. 2001.

[5] P. Maragos, "Dynamical systems on weighted lattices: General theory," *Math. Control Signals Syst.*, vol. 29, no. 21, 2017.

[6] G. Cohen, S. Gaubert, and J.P. Quadrat, "Duality and separation theorems in idempotent semimodules," *Linear Alegbra and its Applications*, vol. 379, pp. 395–422, 2004.

[7] M. Akian, S. Gaubert, and A. Guterman, "Tropical Polyhedra Are Equivalent To Mean Payoff Games," *Int'l J. Algebra and Computation*, vol. 22, no. 1, 2012.

[8] V. Charisopoulos and P. Maragos, "Morphological Perceptrons: Geometry and Training Algorithms," in *Proc. Int'l Symp. Mathematical Morphology (ISMM)*. 2017, vol. 10225 of *LNCS*, pp. 3–15, Springer, Cham.

[9] V. Charisopoulos and P. Maragos, "A tropical approach to neural networks with piecewise linear activations," *arXiv preprint arXiv:1805.08749*, 2018.

[10] L. Zhang, G. Naitzat, and L.-H. Lim, "Tropical geometry of deep neural networks," in *Proc. Int'l Conf. on Machine Learning*. 2018, vol. 80, pp. 5824–5832, PMLR.

[11] Y. Zhang, S. Blusseau, S. Velasco-Forero, I. Bloch, and J. Angulo, "Max-Plus Operators Applied to Filter Selection and Model Pruning in Neural Networks," in *Proc. Int'l Symp. Mathematical Morphology (ISMM)*. 2019, vol. 11564 of *LNCS*, pp. 310–322, Springer Nature.

[12] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. ICCV '17*, 2017, pp. 1389–1397.

[13] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proc. ICCV '17*, Oct 2017.

[14] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Proc. NIPS '15*, pp. 1135–1143. 2015.

[15] D. Speyer and B. Sturmfels, "Tropical mathematics," *Math. Mag.*, vol. 82, 09 2004.

[16] G.F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. NIPS '14*, pp. 2924–2932. 2014.

[17] I. Goodfellow, D. Warde-Farley, M. Mirza, Aaron. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML '13*. 2013, vol. 28, pp. 1319–1327, PMLR.

[18] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE Trans. NNLS*, 2018.

[19] G. C. Calafiore, S. Gaubert, and C. Possieri, "A universal approximation result for difference of log-sum-exp neural networks," *CoRR*, vol. abs/1905.08503, 2019.

[20] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, pp. 67–127, 05 2007.

[21] G. Castellano, A. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feedforward neural networks," *IEEE Trans. NN*, vol. 8, pp. 519–31, 02 1997.

[22] R. A. Cuninghame-Green and P. F. J. Meijer, "An algebra for piecewise-linear minimax problems," *Discrete Applied Mathematics*, vol. 2, no. 4, pp. 267 – 294, 1980.

[23] R. A. Crowell, "The tropical division problem and the Minkowski factorization of generalized permutahedra," *arXiv preprint arXiv:1908.00241*, 2019.

[24] C. Ronse and H.J.A.M. Heijmans, "The Algebraic Basis of Mathematical Morphology. Part II: Openings and Closings," *Computer Vision, Graphics, and Image Processing: Image Understanding*, vol. 54, pp. 74–97, 1991.

[25] P. Maragos, "Morphological signal and image processing," in *Digital Signal Processing Handbook*. CRC Press, 1998.

[26] G. Smyrnis and P. Maragos, "Tropical polynomial division and neural networks," *arXiv preprint arXiv:1911.12922*, 2019.

[27] A. Magnani and S. P. Boyd, "Convex piecewise-linear fitting," *Optim. Eng.*, vol. 10, pp. 1–17, 2009.

[28] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proc. ACL '11*. June 2011, pp. 142–150, ACL.

[29] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[30] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten, "Improving the capacity of very deep networks with maxout units," in *Proc. ICASSP '18*, April 2018, pp. 2971–2975.

[31] F. Chollet et al., "Keras," `https://keras.io`, 2015.