ERA: Enhanced Rational Activations

Martin Trimmel*[1][0000−0001−5991−9845], Mihai Zanfir*[2][0000−0001−9347−0738], Richard Hartley[2,3][0000−0002−5005−0191], and Cristian Sminchisescu[1,2][0000−0001−5256−886X]

[1]Lund University [2]Google Research [3]Australian National University

martin.trimmel@math.lth.se

{mihaiz, richardhartley, sminchisescu}@google.com

Abstract. Activation functions play a central role in deep learning since they form an essential building stone of neural networks. In the last few years, the focus has been shifting towards investigating new types of activations that outperform the classical Rectified Linear Unit (ReLU) in modern neural architectures. Most recently, rational activation functions (RAFs) have awakened interest because they were shown to perform on par with state-of-the-art activations on image classification. Despite their apparent potential, prior formulations are either not safe, not smooth, or not "true" rational functions, and they only work with careful initialisation. Aiming to mitigate these issues, we propose a novel, enhanced rational function, *ERA*, and investigate how to better accommodate the specific needs of these activations, to both network components and training regime. In addition to being more stable, the proposed function outperforms other standard ones across a range of lightweight network architectures on two different tasks: image classification and 3d human pose and shape reconstruction.

Keywords: rational activation, activation function, deep learning, neural networks

1 Introduction

Neural networks keep ever-increasing in size and modern architectures can have billions of parameters. While larger models often perform better than smaller ones, their training entails heavy requirements on the computing infrastructure, resulting in increased costs and environmental concerns due to the electricity they consume [21,36]. The trend towards larger architectures has been reinforced in recent times by the advent of the Transformer [39], which has since its introduction moved into the field of computer vision [9], reducing the usage of more parameter-efficient CNN architectures like VGG [35] and ResNet [11]. The trend away from convolutions to dense linear layers continues with [37] who recently introduced a full-MLP architecture that performs on par with the state-of-the-art on large vision datasets.

* Denotes equal contribution. | Code available at: github.com/martrim/rational

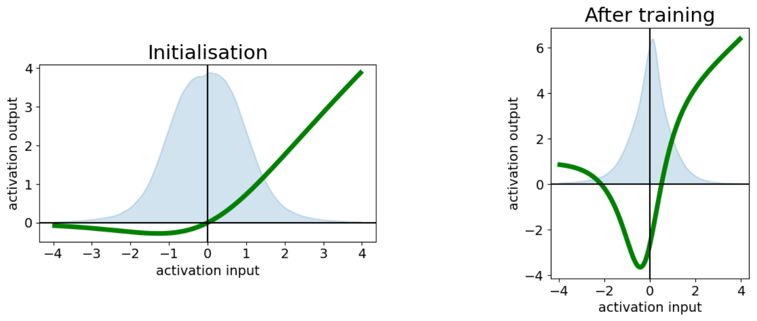

Fig. 1: Plots of the first *ERA* of a CNN before and after training on CIFAR10. The *ERA* is initialised as a Swish activation and has degree 5/4. The plots show that the learned activations differ drastically from the initialization. The shaded curve in the background shows the probability density distribution of the input to the activation (not true scale on the y-axis). The training of the network decreases the variance of the input. Plots of the derivatives are in the appendix.

Since the mathematical form of the learned network function depends heavily on the employed activation functions, choosing the right activation may be a key to increasing the network capacity while limiting the number of training parameters. Research has for a long time focused on non-parametric activation functions like ReLU [10,31], Leaky ReLU [27], GELU [12], Swish [34] and Mish [28]. On the other hand, parameterized activation functions may be auspicious because they have the potential to better adapt to the task at hand.

Recently, [29] showed that rational activation functions (previously employed by [5] in the context of graph convolutional networks) can perform on par with more traditional activation functions on image classification and proved that rational neural networks are universal function approximators. Moreover, rational functions can almost perfectly approximate all standard activation functions. Using rational functions as activations adds only around 10 parameters per network layer, which is negligible compared to the total number of network weights. [3] employ rational activations to train a GAN on the MNIST dataset and mathematically show that rational networks need exponentially smaller depth compared to ReLU networks to approximate smooth functions. [33] propose an interpretable network architecture based on continued fractions, which is a rational function of the input, despite not employing rational activations.

On the other hand, the applicability of rational activations is impeded by the fact that initializing them like the rest of the network via a normal or a uniform distribution typically results in the divergence of training. To mitigate this fact, [29] and [3] fit the activation's parameters to standard activation functions like Leaky ReLU and ReLU. Since each of [29], [3] and [7] use different rational functions, it is unclear whether there is a specific version which should be preferred in practice. Furthermore, rational activations have so far mostly been employed in large architectures to solve simple tasks. Their larger expressivity and adaptability thus begs the question of whether they have additional benefits when used in smaller networks or on more complex tasks.

Our Contributions

- We propose an enhanced rational activation (i.e. *ERA*), incorporating a factorised denominator, which negates the requirement of using absolute values, as in prior work. This results in a safe, smooth rational function that exhibits better performance than previous versions.
- *ERA* allows us to derive a partial fractions representation, reducing memory requirements and further benefiting network performance.
- We investigate different ways of normalizing the input of the activation functions. We show that the right normalization has a great effect on the stability of training, allowing for the first time to investigate rational activations with random initialisation. In line with the recent paper [26], we find that layer normalization benefits all types of activations in CNNs, when compared to batch normalization. In the case of lightweight networks that we employ, we find that instance normalization achieves even better results.
- We study how *ERA* changes during training and conclude that neural networks are able to learn non-trivial activations that drastically differ from their initialisation.
- We ablate *ERA* across different types of neural network architectures, ranging from FCNs, CNNs, MLP-Mixers and Transformers on two different tasks: image classification and 3d human pose and shape reconstruction. For the latter task, we show that a very lightweight, real-time network, of just 3.9M parameters, can achieve results on par with state-of-the-art methods that employ networks in the order of tens or hundreds of millions of parameters.

2 Background

We start with a formal definition of rational functions:

Definition 1. *We say that $f : \mathbb{R} \to \mathbb{R}$ is a rational function if there are numbers $a_0, \dots a_m, b_0, \dots, b_n \in \mathbb{R}$ such that*

$$f(x) = \frac{P(x)}{Q(x)} = \frac{a_0 + a_1 x + \cdots + a_m x^m}{b_0 + b_1 x + \cdots + b_n x^n} \tag{1}$$

and $b_j \neq 0$ for at least one j. The numbers a_i, b_j are called the coefficients of f. If m and n are the largest indices such that $a_m \neq 0, b_n \neq 0$, we say that $deg(f) := m/n$ is the degree of f.

There are a number of points to consider when applying rational functions as activations. Firstly, the function f is undefined on poles (numbers x for which $Q(x) = 0$) and numerically unstable close to poles. In order to make the rational function *safe*, [7] and [29] propose the following modified denominators, respectively:

$$Q_A(x) = 1 + \sum_{i=1}^{n} \left| b_i x^i \right| \tag{2}$$

$$Q_B(x) = 1 + \left| \sum_{i=1}^{n} b_i x^i \right| \tag{3}$$

Since neither Q_A nor Q_B are polynomials, using them in the denominator does not produce a rational function. Apart from lacking the interpretability of a rational function, this makes gradient computation more costly. As an alternative, [3] do not consider the possibility of numerical instabilities and report that using a *non-safe* quadratic denominator is stable in their experiments. However, there persists the risk of numerical instabilities in more complex settings and deeper networks.

Another important point is the degree of the rational function. [29] uses a rational function of degree 5/4, whereas [3] uses a degree of 3/2, which may have helped them to avoid instabilities, due to the lower order of the denominator. They also note that superdiagonal types behave like a non-constant linear function at $\pm\infty$, which is intuitively desirable to avoid exploding, diminishing or vanishing values.

3 Enhanced Rational Function

We now investigate mathematically how to define a rational function without poles in $\mathbb{R}$, thus avoiding the problems mentioned in the previous section.

By a corollary to the Fundamental Theorem of Algebra, every univariate polynomial Q of degree n with coefficients in $\mathbb{C}$ has n complex roots. If the polynomial coefficients are real, the roots can moreover be shown to appear in complex conjugates, implying a factorization of the form:

$$Q(x) = \prod_i (x - x_i) \prod_j (x - z_j)(x - \overline{z}_j). \tag{4}$$

with $x_i \in \mathbb{R}; z_j \in \mathbb{C} \setminus \mathbb{R}$. Since we want Q to have no real roots (i.e. no poles in $\mathbb{R}$), we discard the first factor and obtain a factorisation of Q of the form:

$$Q_C(x) = \prod_{i=1}^{n/2} \left((x - c_i)^2 + d_i^2 \right) \tag{5}$$

for some $c_i \in \mathbb{R}, d_i \in \mathbb{R}_{>0}$. This gives us a rational function of the form:

$$\sigma(x) = \frac{P(x)}{Q_C(x)}, \tag{6}$$

which we call *ERA*. Since the numerator P has its degree one higher than the denominator, we may perform polynomial division. Moreover, by Theorem 2.4 in [4] we can make use of partial fractions to get an expression of the form

$$\sigma(x) = ax + b + \sum_{i=1}^{n/2} \frac{P_i(x)}{Q_i(x)} \tag{7}$$

with $\deg(P_i) = 1, \deg(Q_i) = 2$ for all i. The numbers $a, b \in \mathbb{R}$ denote learnable parameters. Our main motivation for using partial fractions is to make rational activations more computationally efficient: e.g. a rational function of degree 5/4 requires 9 additions/subtractions in equation (6) and equation (7), but the former requires 13 multiplications/divisions and the latter only 6 multiplications/divisions. Table 4 shows that the use of partial fractions also increases the accuracy of the network.

Denominator	**Add**	**Subtract**	**Multiply**	**Divide**
Non-factorised [29]	9	0	13	1
***ERA* no partial frac**	7	2	12	1
ERA	7	2	4	2

Table 1: Number of arithmetic operations to compute different types of rational functions. Using partial fractions decreases the number of costly multiplication/division operations from 14 to 6.

Using a polynomial Q as in equation (5) as the denominator may result in numerical instabilities when the d_i are small. Hence, we propose a numerically safe version by adding a small number $\epsilon > 0$:

$$Q_C(x) = \epsilon + \prod_{i=1}^{n} \left((x - c_i)^2 + d_i^2\right) \tag{8}$$

In practice, we choose $\epsilon = 10^{-6}$.

The proposed activation functions *ERA* in equations (6) and (7) are not polynomials. Hence, it follows from the following theorem that networks using *ERA* as activation function are universal function approximators. (We note that [29] use the same argument to show that the use of their proposed rational activation function yields networks that are universal function approximators.)

Theorem 1 (Theorem 3.1, [32]). *For any i, let $C(\mathbb{R}^i)$ denote the set of continuous functions $\mathbb{R}^i \to \mathbb{R}$. Let $\sigma \in C(\mathbb{R})$. Then*

$$M(\sigma) = \mathit{span}\{\sigma(\mathbf{w} \cdot \mathbf{x} - \theta) : \theta \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^n\} \tag{9}$$

is dense in $C(\mathbb{R}^n)$ in the topology of uniform converge on compacta, if and only if σ is not a polynomial.

4 Experiments

4.1 Image Classification

Setup We train neural networks to perform image classification on the MNIST [22], CIFAR10 [19] and ImageNet [8] datasets, using the Tensorflow [1] framework. Since we are aiming to improve lightweight networks, we focus on two small architectures. We consider fully-connected (FCN) and convolutional neural (CNN) networks, defined recursively by

$$\mathbf{x}_{i+1} = (\sigma_i \circ \ell_i)(\mathbf{x}_i), \quad 1 \leq i \leq 4; \tag{10}$$

where $\mathbf{x}_1$ is the input, ℓ_i is the i^{th} linear layer of the network, followed by an activation function σ_i. In the fully-connected case, ℓ_i is a dense layer and in the convolutional case, ℓ_i is a convolutional layer. We train the networks for 100 epochs using SGD, a batch size of 64 and an initial learning rate of 0.01 which decays by a factor of 10 after having completed 25, 50 and 75 training epochs. All network layers use l2 weight regularization with parameter 0.001. We apply data augmentation with horizontal and vertical shifts of up to 6 pixels and enabling horizontal flipping of images.

Input Normalization There is a range of existing normalization methods that are commonly being employed in neural networks [14,2,38,40]. In general, the input $\mathbf{x}$ to a normalisation layer undergoes two steps: A *standardization step*

$$\mathbf{y} = \frac{\mathbf{x} - \boldsymbol{\mu}}{\boldsymbol{\sigma}} \tag{11}$$

that makes $\mathbf{x}$ have zero mean and unit standard deviation along one or multiple *standardization axes*, and a *learnable step*

$$\mathbf{z} = \boldsymbol{\beta} + \boldsymbol{\gamma} \cdot \mathbf{y} \tag{12}$$

where the distribution of the output $\mathbf{z}$ the learned mean $\boldsymbol{\beta}$ and the learned standard deviation $\boldsymbol{\gamma}$ along some *learnable axes*. By exploring multiple options, we find that instance normalization [38] works best for CNNs and layer normalization [2] works best for fully-connected networks. Our ablations on the standardization and learnable axes are available in the supplementary material.

Initialization and Stability Whereas previous works [29,3] have each focused on a single activation (ReLU and Leaky ReLU) for initializing the rational function, we perform a broader ablation study, encompassing also the newer GELU and Swish activations. Initialising rational activation functions as any standard activation always leads to convergence in our image classification experiments (except when the activation from [29] was initialised as Swish). This leads us to ask whether such an initialisation is necessary or if standard random Glorot initialization can also be used. Unfortunately, random initialization always leads to divergence of the network training in the standard setting. We notice that this is due to 2 reasons:

1. The distribution of the input to the activation shifts during training, which hampers training because the activation parameters need to be constantly adjusted to the input.
2. Random initialisation can lead to unstable derivatives, which can lead to very large gradient values.

Problem 1 is solved via the normalization techniques that we investigated previously. In order to mitigate problem 2, we apply weight clipping. Both techniques together greatly stabilise the training of randomly initialised activations and lead to network convergence in all of our image classification experiments.

Network	**Activation**	**Test Accuracy**
FCN	Leaky ReLU	51.8 ± 0.1
FCN	ReLU	52.6 ± 0.1
FCN	Swish	52.3 ± 0.2
FCN	Rational, [29]	55.0 ± 0.2
FCN	***ERA***	$\mathbf{58.8} \pm 0.2$
CNN	Leaky ReLU	76.0 ± 0.2
CNN	ReLU	77.3 ± 0.1
CNN	Swish	76.8 ± 0.2
CNN	Rational, [29]	82.1 ± 0.2
CNN	***ERA***	$\mathbf{84.4} \pm 0.2$

Table 2: Test accuracy on CIFAR10 in percent, conditioned on the network type and the activation function. Our version outperforms all the baselines for both of the network types.

Results

Outperforming the baselines. As shown in Table 2, our best results outperform non-rational activation functions considerably by a margin of more than 6% in both fully-connected and convolutional neural networks on CIFAR10. In addition, the test accuracy of the modified rational function we proposed is significantly higher than the one achieved by the function proposed by [29], achieving a margin of more than 2% for the CNN and almost 4% for the fully-connected neural network. The fact that both our rational function and the baseline one clearly outperform standard activations, shows that rational activation functions are indeed qualified for increasing the network capacity.

Ablation studies. We perform multiple ablations, in particular on the initialisation of the network, the normalisation of the input of the activation functions and the use of partial fractions. Table 3 shows consistent benefits of initialising *ERA*s as Swish activations. We note that although randomly initialised *ERA*s

perform somewhat worse than the other initialisations, they still outperform all the standard activations that we investigated (cf. Table 2). As shown in Table 4 also note that normalising the input of the activations gives consistent benefits across different settings and multiple network types. Table 4 also indicates a small benefit in performance when using partial fractions instead of standard fractions.
We refer the reader to the supplementary material for a more comprehensive overview of our ablation studies.

Initialization	**FCN**	**CNN**
GELU	58.7 ± 0.3	82.8 ± 0.5
Leaky ReLU	57.8 ± 0.2	82.7 ± 0.2
Random	57.5 ± 0.4	82.5 ± 0.5
ReLU	57.9 ± 0.1	82.9 ± 0.2
Swish	$\mathbf{58.8} \pm 0.2$	$\mathbf{84.4} \pm 0.2$

Table 3: Test accuracy on CIFAR10 in percent, conditioned on the network type and the initialisation of the rational activation. Swish consistently outperforms the other initialisations, whereas random initialisation performs slightly worse. Layer Normalisation was applied to the input of the activation.

Configuration	**FCN**	**CNN**
Ours (baseline)	57.0 ± 0.4	81.8 ± 0.3
+ layer normalization	57.9 ± 0.2	84.1 ± 0.2
+ partial fraction	$\mathbf{58.8} \pm 0.2$	$\mathbf{84.4} \pm 0.2$

Table 4: Test accuracy on CIFAR10 in percent, conditioned on the network type and if partial fractions are used. Both layer normalization and partial fractions give consistent benefits. All networks were initialised with the Swish activation.

Analysis

Plotting of *ERA*s In Figure 1 we compare one of our the best *ERA*s before and after training. The activation function was initialised as a Swish activation and after training vaguely resembles a quadratic function where most of the probability mass of the data lies. We observe similar results for many other *ERA*s that were initialised as Swish activations. On the other hand, the shapes of randomly initialised *ERA*s vary a lot, depending on initial parameter configuration and the convergence of training. Figure 2 displays a successfully trained (final test accuracy higher than 80%) randomly initialised *ERA* of degree 5/4 and its

first derivative before and after training. In this particular example, we observe an activation function that is close to linear on the data distribution which is something that we observe regularly in the case of random initialisation. We also note that the relatively small values of the first derivative do not seem to hurt network performance.
Due to space limitations, we present a more in-depth analysis of plots of rational activations and their derivatives in the appendix.

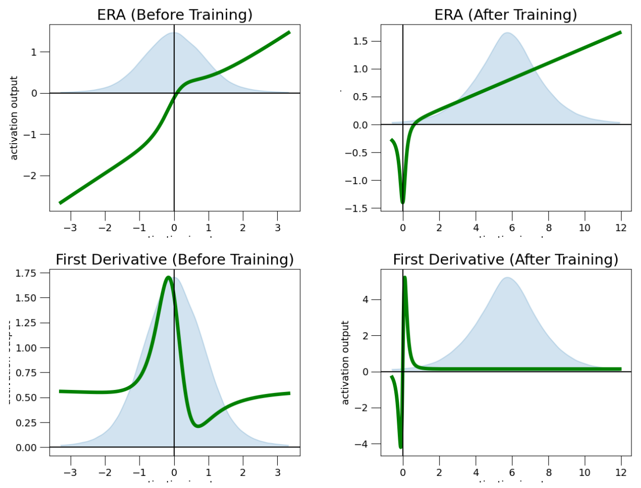

Fig. 2: Plots of the first *ERA* of a CNN before and after training on CIFAR10. The *ERA* is randomly initialised and has degree 5/4. Both the activation and its derivatives differ drastically from the initialization. The shaded curve in the background shows the probability density distribution of the input to the activation (not true to scale on the y-axis). In contrast to Swish initialised *ERA*s, random initialisation can result in very small or large derivative values which can make the training more difficult.

Loss landscapes Inspired by the analysis in [28], we use the methodology of [23], based on filter normalisation, to create 3D visualisations of the loss functions. Figure 3 shows the plots of two Convolutional Neural Networks trained on CIFAR10. Here the xy-plane corresponds to configurations of network parameters and the z-axis corresponds to the corresponding value of the loss function evaluated on the training data. The *ERA* used in the top row is randomly initialised, whereas the one used in the bottom row was initialised with the Swish activation. In both cases, the rational activation had degree 5/4. Both networks achieve a high test accuracy of more than 80%. We see that despite the different initalisations, the loss landscapes are similar and reasonably well-behaved

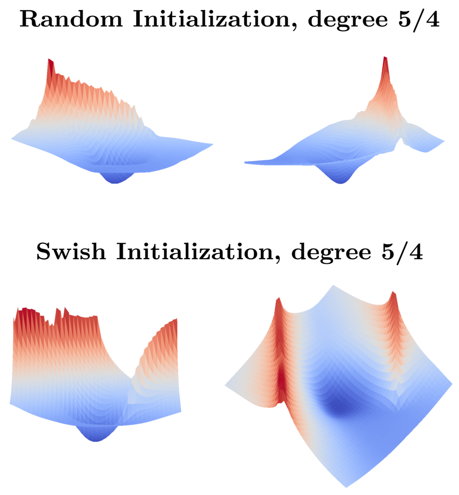

Fig. 3: Plots of the loss surfaces of a CNN trained on CIFAR10, using *ERA*. **Top**: randomly initialized *ERA* of degree 5/4. **Bottom**: Swish-initialized *ERA* of degree 5/4. In each row, the two images show the same loss surface from different viewing angles. *The plots are best viewed digitally in 3d. The 3d versions are included in the supplementary material.*

(i.e. there is a single local minimum in the ”valleys”). We perform a more detailed comparison in the supplementary material, where we also show the loss landscapes of randomly initialised networks that did not converge or achieved a mediocre performance.

4.2 3D Human Pose and Shape Reconstruction

Backbone	**Head**	**Hidden dim**	**#Parameters**
MobileNet	MLP-Mixer	64	6.6M
MobileNet	MLP-Mixer	256	10.0M
MobileNet	Transformer	64	3.9M
MobileNet	Transformer	256	6.7M
ResNet50	Transformer	256	*25.0M* [41]

Table 5: Number of parameters for our lightweight versions of the architecture introduced in THUNDR [41], for the task of 3d pose and shape reconstruction.

In this section, we experiment with and ablate our activation function *ERA* for the task of 3d pose and shape reconstruction of humans from a monocular image. We choose an implementation of *ERA* initialised from a Swish activation and using a partial fraction formulation as suggested in our previous experiments on image classification. We adopt and adapt the architecture previously introduced in THUNDR [41]. At its core, THUNDR is a hybrid convolutional-transformer architecture [9]. In its original implementation, the backbone consists of a ResNet50 [11] and the regression head is a Transformer [39] architecture. We replace the backbone with a lightweight MobileNet [13] architecture and further ablate the choice of the head architecture, by also considering an MLP-Mixer [37]. Our focus here is to assess the impact of *ERA* on lightweight networks and, more specifically, in more recent architectures, i.e. Transformers and MLP-Mixers. We do not replace the standard activations used in the backbone convolutional architecture, which is pre-trained on ImageNet. This could have an impact on the performance even further, but this is not in the scope of this paper and we leave it aside for future work.

Setup As in the original THUNDR paper, we experiment with two different training regimes, both weakly supervised and a mixed regime, weakly and fully supervised. For weak supervision, we use images from both the COCO2017 [25] and OpenImages [20] datasets. We use a mix of ground-truth 2d keypoint annotations, where available, and 2d detections from pre-trained models. For fully supervised training we consider the standard Human3.6M [15] dataset, where ground-truth 3d is available. For all experiments we report results on protocol 2 (P2) of the Human3.6M test set.

Ablations. MLP-Mixers and Transformers both have a hidden dimension hyperparameter that has a big impact on the number of network parameters. We ablate two settings, one with low capacity (64) and one with higher capacity, as proposed in [41] (256). Different from [41], we do not share the parameters across the encoder layers, to better assess the stability of training when multiple instances of *ERA* are learned. We ablate *ERA* with two denominator settings, one with lower expressivity ($\deg(Q) = 2$) and one with higher ($\deg(Q) = 4$). For a comparison with other activation functions, we use ReLU (as in [41]), GELU and Swish.

We follow the training schedule and hyper-parameters described in [41]. We do not investigate further the impact of these choices on the performance of the different activation functions, and leave this for future work. Note that the parameters are tailored in the original paper for the specific choice of ReLU as the activation function. For each different choice of head-model architecture, hidden size and activation function we run 3 different trials and report averaged metrics over the trials. The metrics we use are the standard ones for 3d reconstruction. For 3d joint errors we report mean per joint position error (MPJPE) and MJPE-PA, which is MPJPE after rigid alignment via Procrustes Analysis. For evaluating 3d shape we use the mean per vertex position error (MPVPE) between the vertices of the predicted and ground-truth meshes, respectively.

Results

Number of parameters and running time. In Table 5 we show the number of parameters for our lightweight networks as a function of the backbone and head architectures, and the hidden dimension. Our networks have significantly fewer parameters than the one used in [41], with the smallest one having around 3.9M parameters. Using *ERA* as activation only adds an insignificant number of parameters (depending on the degree of the denominator), between 10 and 20 more for each *ERA* instance. On a desktop computer with an NVIDIA GeForce RTX 2080 Ti graphics card, all of our networks run in real-time at inference, with a frame-per-second rate of around $100 - 110$.

Numerical results. We present results in tables 6, 7 for different training regimes, weakly supervised and mixed, respectively. In the former, our proposed activation *ERA* achieves the best results across the board, with the most noticeable performance boost in the case of lowest capacity (i.e. hidden dimension of 64) for both head-networks type. In the mixed training regime, this observation still holds. But, as the network capacity increases, the gap between *ERA* and other standard activations shrinks, and even reverses, as in the case of MLP-Mixers with higher capacity. Nevertheless, we see in Table 7 that the best performing setting is *ERA* with a high capacity Transformer. Overall, the biggest improvement in performance is achieved in the setting with the lowest capacity, a Transformer with a hidden dimension of 64, where *ERA* has a significantly lower error than all other activations. This holds true for both training regimes, either weakly or with mixed supervision. We compare this setting to the state of the art, and show that this lightweight network is on par or better than other methods with considerably bigger architecture sizes. For this comparison we selected the network that achieved the best training error out of 3 different trials.

The effect of the degree of the denominator Q for *ERA*, seems to be quite minimal, with a slight edge for the setting in which the denominator is set to 2. In general, Transformers seem to perform better than MLP-Mixers for this task, but we did not perform any hyperparameter search for either of the architectures. Additional details, experiments and ablations are included in the supplementary.

5 Conclusions

We have presented *ERA*, a learnable rational activation function that can replace standard activations in a wide range of neural network architectures. Under a theoretical framework, we analysed and showed that *ERA* is a smooth, safe, rational function and can be refactored using a partial fraction representation, which empirically exhibits improved performance. We also showed that the right normalization of inputs has a great effect on the stability and the performance of *ERA*, and, as a byproduct, we can mitigate some of the effects of initialization, even from a random distribution, something not possible before in prior work on rational layers. Across different types of lightweight neural network architectures and for two different tasks, image classification, and 3d human pose and shape reconstruction, *ERA* increases network performance and closes the gap between different sized architectures by increasing the expressivity of the network. Specifically, on the task of 3d human pose and shape reconstruction, we show that a real-time, lightweight network, with only 3.9M parameters, can achieve results that are competitive with state-of-the-art methods, where network parameters are in the order of tens or hundreds of millions.

Acknowledgements. This work was supported in part by the European Research Council Consolidator grant SEED, CNCSUEFISCDI PN-III-PCCF-2016-0180, Swedish Foundation for Strategic Research (SSF) Smart Systems Program, as well as the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	72.2 ±1.3	109.4 ±2.1	117.1 ±2.0
ReLU	-	64	69.2 ±1.8	102.8 ±2.9	109.8 ±3.2
Swish	-	64	72.8 ±0.8	109.5 ±0.5	117.2 ±0.3
ERA	2	64	**67.7** ±0.5	**98.0** ±1.0	**104.3** ±1.5
ERA	4	64	69.0 ±1.2	100.3 ±1.6	107.0 ±2.1
GELU	-	256	66.4 ±0.5	95.0 ±1.0	101.1 ±1.1
ReLU	-	256	66.8 ±1.2	97.2 ±1.7	103.5 ±2.0
Swish	-	256	67.0 ±0.2	97.0 ±0.6	103.6 ±0.6
ERA	2	256	**65.2** ±0.2	93.2 ±0.9	99.4 ±1.4
ERA	4	256	**65.2** ±0.4	**92.7** ±0.5	**98.7** ±0.8

MLP-Mixer

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	80.6 ±0.3	125.5 ±0.6	132.8 ±0.9
ReLU	-	64	78.8 ±0.9	122.9 ±2.7	130.4 ±2.8
Swish	-	64	81.5 ±0.7	127.4 ±2.8	134.9 ±2.3
ERA	2	64	73.8 ±2.0	112.3 ±3.5	**119.5** ±2.7
ERA	4	64	**73.6** ±1.5	**112.1** ±3.5	**119.5** ±3.9
GELU	-	256	63.0 ±0.2	89.1 ±1.1	94.5 ±0.7
ReLU	-	256	69.8 ±0.7	101.9 ±1.9	109.5 ±1.8
Swish	-	256	64.0 ±0.5	91.1 ±1.6	96.7 ±1.3
ERA	2	256	**62.5** ±0.8	**88.5** ±1.8	**93.6** ±2.0
ERA	4	256	63.5 ±0.5	89.6 ±1.9	95.0 ±2.1

Transformer

Table 6: Ablation for networks with **MLP-Mixer (Top)** and **Transformer (Bottom)** architectures, trained in a **weakly supervised** regime. Results are in mm, averaged over 3 trials, reported on the Human3.6M test dataset, protocol P2. We mark the **best** and second best results.

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	53.5 ±1.2	72.1 ± 1.5	76.1 ± 1.3
ReLU	-	64	51.6 ±0.7	71.8 ±1.5	75.8 ±1.1
Swish	-	64	54.2 ±0.8	73.9 ±2.5	78.2 ±3.0
ERA	2	64	51.4 ±0.2	70.9 ±1.6	74.4 ±1.7
ERA	4	64	**51.2** ±1.1	**69.6** ±1.5	**73.4** ±1.5
GELU	-	256	**49.6** ±0.8	68.9 ±1.2	72.9 ±1.6
ReLU	-	256	49.9 ±0.9	68.3 ±1.5	72.2 ±1.6
Swish	-	256	**49.6** ±0.4	**68.2** ±0.8	**72.1** ±1.6
ERA	2	256	50.1 ±0.8	70.6 ±1.8	74.3 ±2.4
ERA	4	256	50.8 ±2.1	70.7 ±4.8	74.8 ±4.6

MLP-Mixer

Activation	**deg Q**	**Hidden dim**	**MPJPE-PA**	**MPJPE**	**MPVPE**
GELU	-	64	50.9 ±1.6	71.6 ±2.7	76.2 ±3.0
ReLU	-	64	51.5 ±0.5	72.8 ±1.2	77.2 ±0.6
Swish	-	64	51.8 ±0.7	71.9 ±0.6	76.4 ±0.5
**ERA*	2	64	**47.2** ±1.5	**66.7** ±1.9	**71.3** ±2.6
ERA	4	64	47.5 ±0.6	67.4 ±1.4	72.1 ±1.4
GELU	-	256	44.8 ±0.6	**63.1** ±1.5	**67.3** ±2.0
ReLU	-	256	45.4 ±0.3	64.2 ±0.5	67.8 ±0.1
Swish	-	256	44.9 ±0.2	63.7 ±1.3	67.7 ±1.4
ERA	2	256	**43.7** ±0.4	63.4 ±0.3	67.4 ±0.5
ERA	4	256	44.8 ±0.9	63.5 ±1.4	67.6 ±1.6

Transformer

Method	**MPJPE-PA**	**MPJPE**
HMR [16]	56.8	88.0
GraphCMR [18]	50.1	-
Pose2Mesh [6]	47.0	64.9
I2L-MeshNet [30]	41.7	55.7
SPIN [17]	41.1	-
METRO [24]	36.7	54.0
THUNDR [41]	34.9	48.0
[41] + ERA* (lightest network)	45.9	64.0

State of the art

Table 7: Ablation for networks with **MLP-Mixer (Top)** and **Transformer (Middle)** architectures, pre-trained in a weakly supervised regime and fine-tuned in a **fully-supervised** regime. Results are in mm, averaged over 3 trials, reported on the Human3.6M test dataset, protocol P2. We mark the **best** and second best results. **(Bottom)** We report our *best performing lightest network with *ERA* out of the 3 trials (selected based on training error), relative to the state-of-the-art methods. Our network is the lightest of the architectures, by a large margin, with around 3.9M parameters (e.g. THUNDR has 25M parameters).

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. ArXiv **abs/1607.06450** (2016)
3. Boulle, N., Nakatsukasa, Y., Townsend, A.: Rational neural networks (2020), `https://proceedings.neurips.cc/paper/2020/file/a3f390d88e4c41f2747bfa2f1b5f87db-Paper.pdf`
4. Bradley, W.T., Cook, W.J.: Two proofs of the existence and uniqueness of the partial fraction decomposition. In: International Mathematical Forum. pp. 1517–1535 (2012)
5. Chen, Z., Chen, F., Lai, R., Zhang, X., Lu, C.: Rational neural networks for approximating jump discontinuities of graph convolution operator. CoRR **abs/1808.10073** (2018), `http://arxiv.org/abs/1808.10073`
6. Choi, H., Moon, G., Lee, K.M.: Pose2mesh: Graph convolutional network for 3d human pose and mesh recovery from a 2d human pose. In: ECCV. pp. 769–787. Springer (2020)
7. Delfosse, Q., Schramowski, P., Molina, A., Beck, N., Hsu, T.Y., Kashef, Y., Rüling-Cachay, S., Zimmermann, J.: Rational activation functions (2020)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database (2009). https://doi.org/10.1109/CVPR.2009.5206848
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021), `https://openreview.net/forum?id=YicbFdNTTy`
10. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics **36**, 193–202 (1980)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2016). https://doi.org/10.1109/CVPR.2016.90
12. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
13. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR **abs/1704.04861** (2017), `http://arxiv.org/abs/1704.04861`
14. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR **abs/1502.03167** (2015), `http://arxiv.org/abs/1502.03167`
15. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(7), 1325–1339 (jul 2014)

16. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: CVPR (2018)
17. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2252–2261 (2019)
18. Kolotouros, N., Pavlakos, G., Daniilidis, K.: Convolutional mesh regression for single-image human shape reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4501–4510 (2019)
19. Krizhevsky, A.: Learning multiple layers of features from tiny images. Master's thesis, University of Toronto (2009)
20. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al.: The open images dataset v4. International Journal of Computer Vision **128**(7), 1956–1981 (2020)
21. Lacoste, A., Luccioni, A., Schmidt, V., Dandres, T.: Quantifying the carbon emissions of machine learning (2019), `http://arxiv.org/abs/1910.09700`
22. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database. ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist (2010)
23. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), `https://proceedings.neurips.cc/paper/2018/file/a41b3bb3e6b050b6c9067c67f663b915-Paper.pdf`
24. Lin, K., Wang, L., Liu, Z.: End-to-end human pose and mesh reconstruction with transformers. In: CVPR. pp. 1954–1963 (2021)
25. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
26. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s (2022)
27. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models (2013)
28. Misra, D.: Mish: A self regularized non-monotonic neural activation function. arXiv preprint arXiv:1908.08681 (2019)
29. Molina, A., Schramowski, P., Kersting, K.: Padé activation units: End-to-end learning of flexible activation functions in deep networks (2019)
30. Moon, G., Lee, K.M.: I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In: European Conference on Computer Vision (ECCV) (2020)
31. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines (2010)
32. Pinkus, A.: Approximation theory of the mlp model in neural networks. Acta Numerica **8**, 143–195 (1999). https://doi.org/10.1017/S0962492900002919
33. Puri, I., Dhurandhar, A., Pedapati, T., Shanmugam, K., Wei, D., Varshney, K.R.: Cofrnets: Interpretable neural architecture inspired by continued fractions (2021), `https://openreview.net/forum?id=kGXlIEQgvC`
34. Ramachandran, P., Zoph, B., Le, Q.V.: Searching for activation functions (2018), `https://openreview.net/forum?id=SkBYYyZRZ`
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

36. Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for modern deep learning research (Apr 2020). https://doi.org/10.1609/aaai.v34i09.7123, `https://ojs.aaai.org/index.php/AAAI/article/view/7123`
37. Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., Dosovitskiy, A.: Mlp-mixer: An all-mlp architecture for vision. CoRR **abs/2105.01601** (2021), `https://arxiv.org/abs/2105.01601`
38. Ulyanov, D., Vedaldi, A., Lempitsky, V.S.: Instance normalization: The missing ingredient for fast stylization. CoRR **abs/1607.08022** (2016), `http://arxiv.org/abs/1607.08022`
39. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need (2017), `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`
40. Wu, Y., He, K.: Group normalization. CoRR **abs/1803.08494** (2018), `http://arxiv.org/abs/1803.08494`
41. Zanfir, M., Zanfir, A., Bazavan, E.G., Freeman, W.T., Sukthankar, R., Sminchisescu, C.: Thundr: Transformer-based 3d human reconstruction with markers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 12971–12980 (October 2021)