

Achieving Compatible Numeral Handwriting Recognition Rate by a Simple Activation Function

Daniel Brüderle¹, Khamron Sunat^{2, *}, Sirapat Chiewchanwattana³,
Chidchanok Lursinsap⁴ and Suchada Siripant⁴

¹Kirchhoff Institute for Physics, Ruprecht-Karls University
INF 227, 69120 Heidelberg, Germany
bruederle@kip.uni-heidelberg.de

²Department of Computer Engineering, Mahanakorn University,
Cheum-Sampan Road, Nong-Chok, Bangkok, Thailand 10530
khamron_sunat@yahoo.com

³Department of Computer Science, KhonKaen University,
Khon Kaen, Thailand 40002
sunkra@kku.ac.th

⁴AVIC Research Center, Department of Mathematics, Chulalongkorn University
Phayathai Road, Patumwan, Bangkok 10330, Thailand
ichidcha@chula.ac.th, ssuchada@chula.ac.th

Abstract: Most of the supervised neural networks for numeral handwriting recognition employ the sigmoidal activation function to generate the outputs. Although this function performs rather well, its computational time as well as its hardware realization is costly and complicated. Here, we introduce a simple activation function in forms of a recursive piecewise polynomial function as an activation function. The accuracy of recognition can be adjusted according to the parameters of the function. In addition a new risk function measuring the discrepancy between the correct and estimated classification of the network is also presented to improve the performance. The proposed activation function and the risk function can achieve the same accuracy compatible with that from the sigmoidal function when tested with the benchmark data set.

Keywords: activation function, handwriting recognition, risk function, supervised neural network.

I. Introduction

Numeral handwriting recognition [2] is still one of the most studied problems in neural pattern recognition area. The recognition is achieved by training a supervised neural network with an appropriate activation function and

measuring the success by a risk function. The most popular activation function employed in this work is the sigmoidal or logistic function [3] since it can be adjusted to imitate a threshold function and it is also differentiable. However, this activation function faces the problem of costly computational time and space even though the recognition rate is high. In this paper, we introduce a new simple activation function in forms of a piecewise recursive polynomial function not only to reduce the computational time but also achieve the same compatible recognition rate. This activation function is adapted from the activation function previously suggested by Sunat [6]. The testing data set of size 6000 samples is obtained from CENPARMI database at Concordia University in Canada.

The rest of the paper is organized as follows. Section 2 explains how the features are extracted from a given image. Section 3 introduces our proposed activation function and risk function. Section 4 summarizes the comparison results. Section 5 concludes the paper.

II. Basic Setup

The features of a given numeral handwriting image are extracted by adopting the multi-resolution representation concept introduced by Liu et al. [2]. The concept is summarized as follows. In [2], Liu et al. proposed a single

* Corresponding Author

layer network for solving the CENPARMI classification task. Each numeral image is captured by a multi-resolution representation, i.e. any numeral image is presented to the network in three different resolutions, namely 8×8 , 4×4 and 2×2 pixels. This approach facilitates the processing of the data on both pixel and region levels for the network, which can preserve both local and global features on the same scale. It is also an important remedy for some shortcoming of the locally expanded high order inputs introduced further below. Before the original numeral image is processed into the three resolution sub-images, it is normalized to a 32×32 pixel array. Then, its stroke-width variations in the normalized image are eliminated by Hilditch's algorithm [2], [4].

Two different methods for extracting a low resolution image from the original one are suggested and tested. In this work, only the more successful one of both will be used, namely the *Gaussian pyramids representation*. This method achieves resolution reduction by low-pass filtering the given image. Starting from the original image (resolution $r \equiv 1$) with $2^n \times 2^n$ pixels (CENPARMI: $n = 32$ each lower resolution $r = 2^{-j}$ containing $2^{n-j} \times 2^{n-j}$ pixels is computed by convolution of the original $f_0(x, y)$ with an impulse response function

$$h(x, y) = \frac{1}{2\pi\sigma_x^2} \exp\left(-\frac{x^2 + y^2}{2\sigma_x^2}\right) \quad (1)$$

Hence, including a reasonable sampling position within the old image for every pixel in the new image, every single layer j of the Gaussian pyramid with the original image at the bottom ($j = 0$) and a single pixel at the top ($j = n$) is computed by

$$f^j(x, y) = \sum_{x'} \sum_{y'} f^0(x', y') \cdot h^j(2^j x + 2^{j-1} - x', 2^j y + 2^{j-1} - y') \quad (2)$$

Liu et al. empirically found the value of σ_x to be close to an optimum when it is set to $2/3$. Another method proposed by Liu et al. and adopted for this work is the usage of *locally expanded high order* inputs. This means that not only the pixel gray-scale values themselves are presented to the network input, but also the pair wise products of the values of all neighboring pixels in each single sub-image are considered. For keeping input values at about the same magnitude, the product of two pixel values is replaced by its square root

$$f_h(x_1, y_1, x_2, y_2) = \sqrt{f(x_1, y_1) \cdot f(x_2, y_2)}. \quad (3)$$

Figure 1 illustrates the pixel arrays of different resolutions, the connection bars between neighboring pixels representing the second order products.

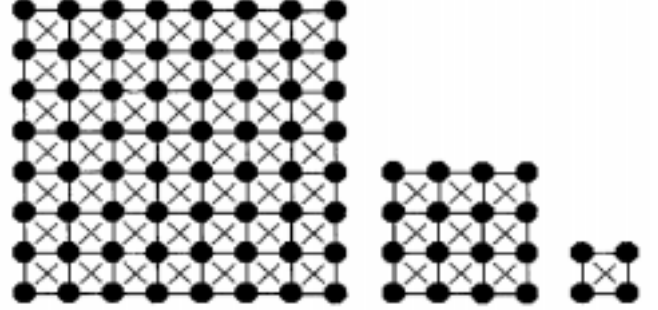


Figure 1. 2D pixel patterns with resolutions 8×8 , 4×4 and 2×2 . The bars represent the local second order expansions described in the text. Every dot and every bar denote an input value to the network. The picture is taken from [2].

Every dot and every bar in the figure denotes an input to the neural network. The total number of inputs becomes $8 \times 8 + 4 \times 4 + 2 \times 2$ (pixel arrays of differently resolved image) $+ 7 \times 7 \times 4 + 2 \times 7$ (second order inputs for 8×8 image) $+ 3 \times 3 \times 4 + 2 \times 3$ (second order inputs for 4×4 image) $+ 6$ (second order inputs for 2×2 image) $= 342$.

Let α_j be the activation of a neuron j , i.e. the sum of all input values x_i running into this neuron and weighted with an individual synaptic strength w_{ij} ,

$$\alpha_j = \sum_i w_{ij} \cdot x_i. \quad (4)$$

Since the CENPARMI handwriting recognition task is a 10-class problem, the network presented in [2] consists of $N_{\text{class}} = 10$ output neurons with a so-called *logistic* activation function (AF), i.e. the neuron output y_j is determined by its activation as follows:

$$y_j^{\text{lgf}}(\alpha_j) = \frac{1}{1 + e^{-\alpha_j}}, j = 1 \dots N_{\text{class}}. \quad (5)$$

The network serves as a classifier, i.e. for every input pattern presented to the input, there should always be exactly one output neuron active, namely the one associated with the correct class, while all other output neurons should exhibit their minimum value. This desired ideal classification function can be written as

$$\text{class} : X \rightarrow S \quad (6)$$

with X being the set of input patterns (for this study the pre-processed CENPARMI database) and

$$S = \left\{ \mathbf{s}^{(j)} \mid \mathbf{s}^{(j)} \left(\underbrace{0, 0, \dots, 0}_{j-1}, 1, \underbrace{0, \dots, 0}_{m-j} \right) \right\} \subset \{0, 1\}^{N_{\text{class}}}. \quad (7)$$

Let \mathbf{w} be the vector of all synaptic weights within the network, i.e. the vector of all free parameters \mathbf{w}_{ij} in the proposed setup. Furthermore, let $\mathbf{s}(\hat{\mathbf{x}}) \in S$ be the desired network output vector for an applied input pattern $\hat{\mathbf{x}} \in X$. Then \mathbf{w} can be trained in a supervised way, namely, by reducing the mean square error function (MSE) [3]

$$E^{\text{MSE}}(\mathbf{w}) = \frac{1}{2} \sum_{j=1}^{N_{\text{class}}} (y_j - s_j)^2 \quad (8)$$

with the widely used error back-propagation (BP) algorithm. Minimizing an arbitrary error function with BP can easily be derived from the following equation

$$\begin{aligned} -\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}_{ij}(t)} &= -\frac{\partial E(\mathbf{w})}{\partial \alpha_j} \frac{\partial \alpha_j}{\partial \mathbf{w}_{ij}} \\ &= -\frac{\partial E(\mathbf{w})}{\partial y_j} \frac{\partial y_j}{\partial \alpha_j} \\ &= \delta_j x_i. \end{aligned} \quad (9)$$

The so-called generalized delta term δ_j encapsulates

$$\delta_j = -\frac{\partial E(\mathbf{w})}{\partial y(j)} \frac{\partial y_j}{\partial \alpha_j}. \quad (10)$$

Equation (9) leads to the well known gradient descent rule for weight updating

$$\Delta \mathbf{w}_{ij} = -\eta \delta_j x_i, \quad (11)$$

including some small, dimensionless learning rate η . This weight change normally is applied iteratively, i.e. for every available input pattern a single weight update is performed, usually multiple times in a round-robin manner, until the network output satisfies some predefined criterion (typically an error value to be under-run).

The classification decision for a non-ideal classification network is based on a simple maximum search over all N_{class} output neurons. The one with the largest output value is associated with the guessed class.

For the MSE error function defined in (8) and the logistic

AF defined in (5), the learning rule (11) concretizes to

$$\Delta \mathbf{w}_{ij}^{\text{MSE}} = \eta (s_j - y_j) y_j' x_i. \quad (12)$$

Inserting the logistic AF (see (5)) and its derivative is $\frac{\partial y^{\text{lg}}}{\partial \alpha} = y^{\text{lg}} (1 - y^{\text{lg}})$, the gradient descent rule used in [2], [3] emerges:

$$\Delta \mathbf{w}_{ij}^{\text{MSE,lg}} = \eta (s_j - y_j) y_j (1 - y_j) x_i. \quad (13)$$

So far, all setup suggestions by Liu et al. have been followed for this study. A control experiment *exactly* following their methods led to the same results as theirs, see Sec.4 for details.

III. New Approach

Computing an ANN output mainly means computing the output of each involved neuron as a function of its activation. Inspired by, among others, [5] and [7], the author of [6] proposed a specific method to model sigmoid-like AF with a piecewise defined polynomial.

The basic polynomial formulation of a function close to the hyperbolic tangent is

$$g(\alpha, p) = \begin{cases} -1, & \alpha \leq -2^{p+1}, \\ -1 + \left(1 + 2^{-(p+1)} \cdot \alpha\right)^{3 \cdot 2^p - 1}, & -2^{p+1} < \alpha < 0, \\ +1 - \left(1 - 2^{-(p+1)} \cdot \alpha\right)^{3 \cdot 2^p - 1}, & 0 \leq \alpha < 2^{p+1}, \\ +1, & \alpha \geq 2^{-(p+1)}. \end{cases} \quad (14)$$

As shown in [6], this piecewise polynomial with p being zero or a positive integer can be implemented in a fast recursive way for software simulations, leading to the term *p-recursive piecewise polynomial* (p-RPP) for the new AF. It is also proven that $g(\alpha, p)$ is differentiable in α , and a fast pseudo code algorithm providing both g and g' is given. Figure 2 shows the graph of the second-order 0-RPP (solid line) compared with the hyperbolic tangent (dash-dotted line). Even for this lowest order both curves fit quite well, as the difference plot (dash-dashed line) exhibits.

Hence, the main advantage of replacing a real sigmoid AF by the p-RPP is the computation efficiency. In [6] the speed-up is documented with some systematic experiments. The results show a computation time decrease for 3-RPP of 30% to 64% compared to a sigmoid AF, (5), and a decrease of 40% up to more than 80% compared with hyperbolic tangent function, all results being strongly compiler-and precision-dependent. Of course the speed-up is even larger for p-RPP functions of lower orders, e.g. 65% less computation time

being the worst value for 0-RPPs compared with hyperbolic tangent.

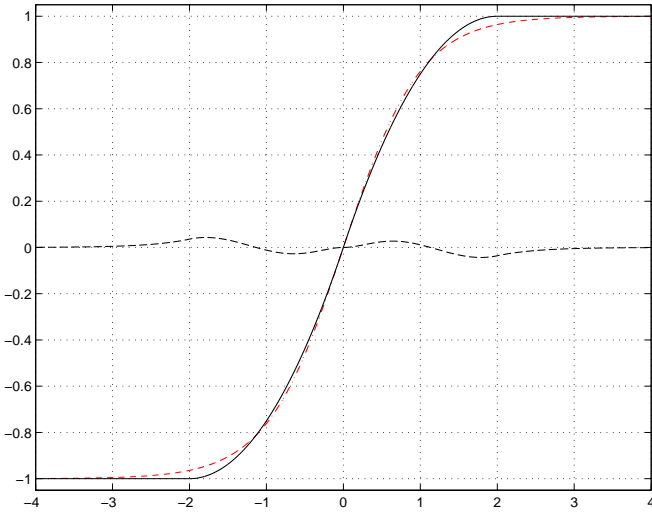


Figure 2. The graph of 0-RPP function (solid) shown as hyperbolic tangent (dash-dot), and the difference between both (dash-dash). The picture is taken from [6].

A major problem for the training of an ANN with a gradient descent method like BP algorithms is AF exhibiting ranges of constant values, since *no gradient* means *no weight change*. If input patterns generate an activation value in these regions, the synaptic weights will not change anymore, i.e. training can run into saturation without reaching a global minimum. Typical AF like the logistic function (see (5)) or the hyperbolic tangent does not have regions with a gradient being totally zero. Therefore, they may run into phases of very slow learning, but never will completely get stuck.

Another problem of constant output for larger ranges of activation arises especially for classification tasks. The classification guess of a network normally is decided by just associating the neuron having the *largest* output with the predicted class. We will call this the *classify-by-max* (CBM) criterion. But for the p-RPP function, which outputs its maximum (minimum) value for all activation values $\alpha \geq 2^{p+1}$ ($\alpha \leq -2^{p+1}$), the probability of two neurons exhibiting *exactly* the same output value for a specific input pattern becomes significantly large. If, for example, two 0-RPP neurons are activated to the values $\alpha_1 = 2.01$ and $\alpha_2 = 13.35$, both will output $y_1 = y_2 = 1$. Hence, the zero-gradient viz. stereotypic-output ranges are a problematic feature of p-RPP neurons. This has to be countervailed against with extra efforts.

A promising idea is to find a mechanism that keeps the activation of p-RPP neurons within the bounds for non-linearity $[-2^{p+1}, 2^{p+1}]$, while network training is performed. Furthermore, a situation that surely should be avoided is a pre-training synapse weight initialization that already pushes

neurons into saturation for many input patterns due to large activation values. This will probably happen with large initial weights.

In the following section the results of the handwriting recognition experiment as proposed in [2] are reproduced. The same experiment is repeated using p-RPP neurons without further improvements. A strong dependency of training success from the weight initialization can be shown. The loss of training performance for high initial weights is obvious and can be assigned primarily to the zero-gradient ranges and the problems resulting from it as described above.

A method supporting and developing a network-wide moderate neuron activation will be introduced now. If learning means reducing the MSE error function as defined in (8), the intention - and most times the result - is a network exhibiting one large output value for signaling a class recognition and small output values for all other neurons. The tendency during training with MSE is to manipulate the synaptic weights such, that for all input patterns this large-small gap is optimized. Thus, BP training with MSE often pushes some synapses to extreme weights in order to reduce error contribution from a subset of input patterns and their according targets. This may lead to extreme activation values for some other patterns. As explained above, such a development is problematic for the AF with stereotypic output ranges like p-RPP.

In [1] an error functional is proposed that can be transformed into an error function for the handwriting recognition problem, involving all output neurons at the same time for a single weight update. The main idea is to add a *defuzzification layer* right behind the output of the neural network. This output layer has as many units as the network has output neurons or classes, and shall perform a mapping from any vector \mathbf{y} of all possible network outputs Y to the desired target vector $\mathbf{s} \in S$, with S according to the definition in (7),

$$c : Y \rightarrow S \quad (15)$$

Vector \mathbf{y} is computed by the activation function having \mathbf{x} and \mathbf{w} as its variables. Hence, an ideal defuzzification layer satisfies $c(\mathbf{y}(\mathbf{x}, \mathbf{w})) = \text{class}(\mathbf{x})$ for all input patterns $\mathbf{x} \in X$.

Obviously, an error function based on such *crisp* outputs, associating input patterns with elements of S , is not differentiable and therefore not suited to be used within a BP-like network training. The authors of [1] approximated the *fuzzy-to-crisp* mapping c by the differentiable function

$$c^*(\mathbf{y}(\mathbf{x}, \mathbf{w})) = \left(\frac{\mathbf{y}(\mathbf{x}, \mathbf{w})}{\|\mathbf{y}(\mathbf{x}, \mathbf{w})\|_q} \right)^u, \quad (16)$$

with u and q being values larger than 1 and with the convention that $(\mathbf{v})^u = (v_1, v_2, \dots, v_m)^u := (v_1^u, v_2^u, \dots, v_m^u)$.

An easy-to-follow derivation in their publication shows that

$$\lim_{u,q \rightarrow \infty} c^*(\mathbf{y}(\mathbf{x}, \mathbf{w})) = c(\mathbf{y}(\mathbf{x}, \mathbf{w})) \quad (17)$$

Using this function c^* , we can define a new error function

$$J(\mathbf{w}) := \left\| c^*(\mathbf{y}(\mathbf{x}, \mathbf{w})) - \text{class}(\mathbf{x}) \right\|^2. \quad (18)$$

This error function can be used on a subset of all possible input data, which may be assumed to exhibit the same distribution like the total data. Then, the sum over all elements of this training set is an empirical estimation of the general misclassification rate of the network. Therefore, the authors of [1] call the accumulation of the new error function over a training set *approximate differentiable empirical risk functional* (ADERF). The term *functional* is based on the definition of a risk functional (RF): Within the statistical learning theory it denotes the expected value of an arbitrary loss function L measuring the discrepancy between the correct and the estimated classification of a system. We will keep to that acronym for convenience, $E^{\text{ADERF}} \equiv J$, but still will call E^{ADERF} a *function*, since we apply weight changes iteratively during training (and do not calculate a loss function estimate based upon the whole training set).

The great advantage of the ADERF error function for the p-RPP AF arises from the following fact: Assuming a nearly ideal defuzzification layer, the ADERF error can be pushed close to zero without the need of pulling the network output *to rails*, i.e. close to maximum viz. minimum values. After training with this error function is stopped and the defuzzification layer is removed, the network output will classify well in terms of the CBM criterion, but the winner in most cases will not output close to its maximum value. Roughly spoken, we abandon the goal of a network output close to the vectors defined by S , which is unreachable in a perfect way for most problems, anyway, but it is satisfied with a correct CBM classification at all. With this relinquishment, weight configurations emerge from training that result in activation values more probably located in the non-linear ranges of p-RPP and, therefore, tend to avoid stereotypic network outputs.

For the ADERF error function, the general weight updating rule (11) can be given the specific form

$$\Delta w_{ij}^{\text{ADERF}} = -\eta \delta_j^{\text{ADERF}} y'_j x_i \quad (19)$$

with

$$\delta_j^{\text{ADERF}} = \sum_{k=1}^{N_{\text{class}}} \frac{\partial E^{\text{ADERF}}}{\partial c_k^*} \frac{\partial c_k^*}{\partial y_j} \quad (20)$$

$$= \sum_{k=1}^{N_{\text{class}}} (c_k - s_k) u \frac{y_k^{u-1}}{\|y\|_q^u} \left(\delta_{ij}^{\text{Kr}} - \frac{y_k y_j^{q-1}}{\|y\|_q^q} \right)$$

Note that $\delta_{ij}^{\text{Kr}} = \begin{cases} 0, & i \neq j \\ 1, & i = j \end{cases}$ is the Kronecker delta and has

nothing to do with δ_j^{ADERF} . With this new weight updating rule, we hope to alleviate the problems emerging from the appliance of the p-RPP AF. The parameters u and q still have to be selected sensibly. See the following section for an experimental parameter adjustment.

IV. Experimental Results

All experiments presented in this section are approaches to solve the classification task based on the CENPARMI handwriting recognition database, consisting of 6000 images of handwritten numerals. In order to gain comparable results, this set usually is divided into a training set T_{tr} (first 4000 samples) and a test set T_{te} (remaining 2000 samples). This division has been kept throughout the work.

For all experiments, the network architecture was a single layer neural network consisting of ten neurons, with multi-resolution locally expanded second order input, altogether 342 float input signals. Furthermore, a bias input providing a value of 1 was fed into each neuron, hence the total number of trainable synaptic weights became $N_{syn} = 3430$. Every experiment was performed N_{exp} times with different random synapse weight initializations in order to get statistical information about reliability, mean and standard deviation of training success. The weight values were selected randomly from an interval $[-W_{max}, W_{max}]$ with a uniform distribution. If not mentioned explicitly, all results are the recognition rates of the network after 1000 epochs. The recognition rate G of correctly classified patterns from T_{te} after training with only T_{tr} , i.e. the generalization capability of the trained network, is considered the measure for training success. For each epoch, every pattern of the training data set was applied to the network input and a weight change was performed according to (11). The learning rate η was set to a fixed value of 0.05. All data presented here have been gathered with a full-custom C++ implementation of the setup.

In a first step, the experiment proposed in [2], i.e. BP training based on the MSE error function and neurons computing the logistic AF, was reproduced for control reasons. The weights were initialized with $W_{max} = 1$, in accordance with the published data, the number of runs for statistics was $N_{exp} = 20$. The achieved recognition rate of $G = (97.10 \pm 0.15)\%$ fits the published result of 97.05% (unfortunately given without error estimation) very well.

In a second step the new p-RPP AF was introduced to all neurons, and the modified experiment was repeated, again with $N_{\text{exp}} = 20$. Due to the existence of zero-gradient ranges we expected a worse result compared to the logistic AF, possibly depending on the initialization of synaptic weights. As will be seen below, some runs still gained recognition rates similar to the training with the logistic AF, but others indeed obviously got stuck and converged at much lower rates. For the logistic AF, the maximum deviation from the best result G^* over a series of identical (except random initialization) experiments was below 1% of G^* . Thus, we propose the percentage R of training runs which achieved at least 99% of the maximum value over all runs to be a coarse measure for training success reliability,

$$R = \frac{N_{99}}{N_{\text{exp}}}, \text{ with } N_{99} = \# \text{ of run } i: G_i \geq 0.99G^*. \quad (21)$$

Table 1 summarizes the results of the first p-RPP experiment, namely the training of p-RPP and logistic AF network (standard setup, $N_{\text{exp}} = 20$) with different values for the weight initialization parameter, ranging from $W_{\text{max}} = 0.25$ to $W_{\text{max}} = 1.5$. For the logistic AF, the reliability of training success is found to be independent of W_{max} , at least for values $W_{\text{max}} \leq 1.25$. In contrast, for the p-RPP AF the data exhibits a strong dependency on W_{max} , just as expected due to the zero-gradient regions of the piecewise polynomial.

AF W_{max}	Logistic	0-RPP	1-RPP
0.25	$G = 97.09 \pm 0.05$ $R = 100$	95.4 ± 3.9 80	97.21 ± 0.08 100
0.50	97.10 ± 0.09 100	90.3 ± 5.3 35	97.22 ± 0.09 100
0.75	97.09 ± 0.1 100	88.0 ± 8.9 40	94.4 ± 4.4 70
1.00	97.10 ± 0.15 100	84.0 ± 8.2 15	89.4 ± 8.2 40
1.25	97.09 ± 0.10 100	72.2 ± 12.1 25	88.2 ± 7.1 25
1.50	96.62 ± 2.12 95	72.0 ± 14.8 10	84.8 ± 9.9 15

AF W_{max}	2-RPP	3-RPP
0.25	$G = 97.23 \pm 0.06$ $R = 100$	97.22 ± 0.09 100
0.50	97.24 ± 0.12 100	97.22 ± 0.08 100
0.75	95.8 ± 3.4 85	97.23 ± 0.11 100
1.00	94.8 ± 5.2 80	96.7 ± 2.1 95
1.25	92.0 ± 4.9 45	94.4 ± 4.4 70
1.50	86.6 ± 9.3 30	92.9 ± 4.7 55

Table 1. Recognition rates G (upper values, in %) on the test set after training with different values of the weight initialization parameter W_{max} , given with the measured success reliability R (lower values, in %). Each column lists the results for another AF. The errors for G are standard deviations. The errors for R are difficult to estimate, but a very coarse statistical estimation following $\Delta N = \pm\sqrt{N}$ leads to $\Delta R \approx \pm 20\%$.

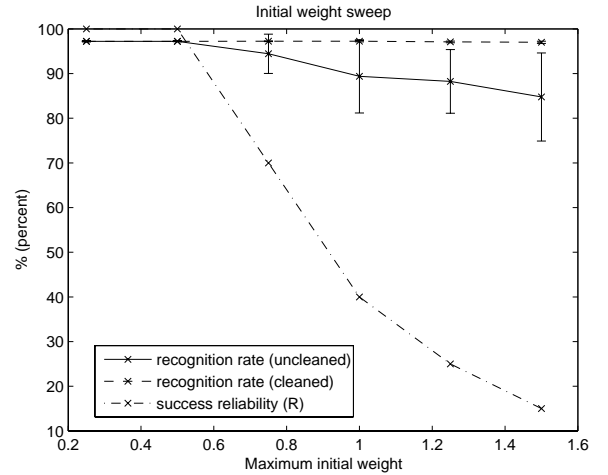


Figure 3. Solid line: Recognition rates G of a 1-RPP network after training with different weight initializations ($W_{\text{max}} \equiv \text{Max initial weight}$). Dash-dotted line: Measured success reliability R for the same experiments, i.e. the ratio of experiment runs that do not differ more than 1% from the best run. Dashed line: Mean and standard deviation over only those runs lying within reliability range.

Figure 3 illustrates the break of training success reliability for weight initializations selected too large. It is a plot of the

W_{\max} -sweep found in Table 1 for the 1-RPP AF. The solid line represents the mean recognition rate G , the error bars denoting the standard deviation over all runs. Since for higher values of W_{\max} training success obviously decreases, the experimentally found reliability measure R , i.e. the ratio of experiment runs, that do not differ more than 1% from the best run, is also included (dash-dotted line). Obviously, for large values of W_{\max} the solid line contains *dirty* data, i.e. runs that got stuck due to the insufficiency of the p-RPP AF. In that case the standard deviation is not a good error measure for the *uncleaned* data. The dashed line gives the mean and standard deviation over only those runs lying within reliability range and is therefore called *cleaned*. Together with the reliability plot it provides much better information about characteristics of a training setup.

Another insight from these results is that selecting initial weights small enough already makes the p-RPP AF a fast and well performing alternative to continuously defined AF. This can be seen, for example, from the recognition rate for 1-RPP with $W_{\max} = 0.25$. The number even exceeds the best value for the logistic AF. Generally, all results for experiment series with $R = 100\%$ provided similar or even better recognition rates compared to the logistic AF runs.

But how to select the initial weight range for an arbitrary classification problem has not yet been answered. It is not practical to perform each experiment multiple times with different weight initializations and hope to find a setup providing reliably successful training. Reducing or eliminating the break of performance for large starting weights is fundamental for making p-RPP AF a useful alternative to continuously defined sigmoidal functions. The ADERF error function with its inherent feature to optimize classification without pushing weights too far might be the appropriate tool.

Hence, the MSE error was replaced by the ADERF error function in a third step, while both the p-RPP and the logistic AF were used in different experiment series.

A question not yet answered is how to select the ADERF parameters u and q (see (16)). A priori, we consider them to be integers larger than 0, because the necessary exponent computing can be performed in a fast recursive way. The larger both values are set, the more exactly the defuzzification layer will provide a crisp classification. But computation time will grow, either, since more multiplications have to be performed. Computational costs during training are not considered the topic of optimization for this work, we focus on speeding up classification *after* training, i.e. when the network is exposed to arbitrary patterns (including not trained ones). Anyway, choosing the step-size of u and q smaller than 1 was decided to be not necessary. Furthermore, the differentiability of ADERF as one of its important properties arises from the approximated character of ADERF. Hence, training might be supported better, if u and q are selected low. This leads to two opposing demands, namely large vs. small values for the ADERF parameters.

Thus, a systematic parameter sweep was performed, repeating the network training with p-RPP AF and the ADERF error function and applying different integer pairs (u, q) . For every parameter pair, $N_{\text{exp}} = 40$ experiments with 300 epochs each were recorded. Each of the 40 runs had an individual random weight initialization determined by $W_{\max} = 1$, since this value was problematic for all MSE approaches with p-RPP AF shown so far. Table 2 summarizes the experimentally found reliability values R gained from these runs.

AF (u, q)	0-RPP	1-RPP	2-RPP	3-RPP
(1,5)	5	5	18	18
(2,5)	25	78	95	100
(3,5)	55	68	90	88
(4,5)	28	25	65	47
(5,5)	15	25	43	43
(2,1)	28	28	22	25
(2,2)	75	80	78	90
(2,3)	55	88	95	95
(2,4)	43	75	97	100
(2,10)	40	65	90	95

Table 2. Experimentally measured reliability R of training success (in %) for p-RPP networks with the ADERF error function, depending on different choices for the ADERF parameters u and q . Again, error estimation can only be

$$\text{guessed by } \Delta R \approx \pm \frac{\sqrt{N_{\text{exp}}}}{N_{\text{exp}}} = \pm 12\% .$$

The experiments show that for a high weight initialization, Some (u, q) configurations can not remedy the phenomenon of recognition rate convergence far below the values normally reached with the logistic AF and MSE, but that for others the success reliability can be increased significantly. Exemplarily for the 2-RPP AF setup, Figure 4 plots the uncleaned version of G (solid line) and the reliability R (dash-dotted) plus the cleaned G (dashed line) of the sweeps of the ADERF parameters u and q .

Considering all results, selecting $u = 2$ and $q \approx p + 2$ is proven to be a reasonable choice when training networks with the ADERF error function and p-RPP neurons with $p \leq 3$. The dependency $q_{\text{optimal}(p)}$ probably arises from the insufficiency of the low-order sigmoid approximation and the possibility to compensate it by a softer gradient of c^* .

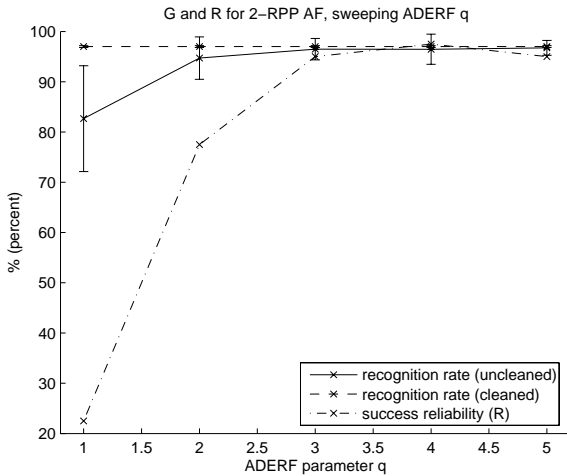
For all p-RPP AFs, the new ADERF error function with well chosen parameters increases the robustness of training against the weight initialization problem and, inherently, avoids pushing the activation of p-RPP neurons out of their

non-linear regions. Hence, one can fully benefit from the advantage of the p-RPP AF, namely its fast computability, without loss of classification performance.

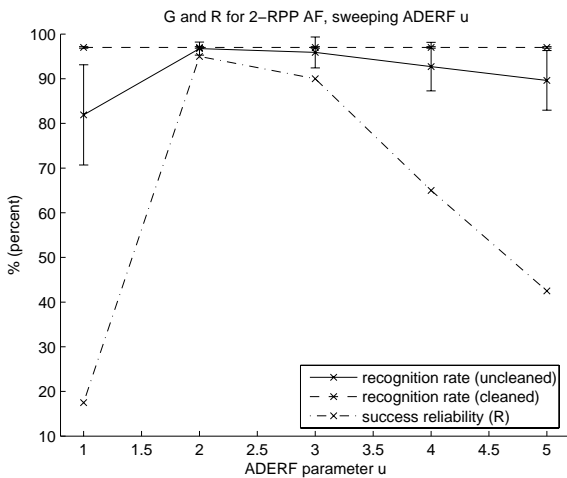
Furthermore, when summarizing the recognition results for MSE and ADERF ($u = 2, q = 6$), applied to two AFs considered so far plus a new one (we call it *absolute sigmoid*),

$$y^{\text{abssig}}(\alpha) = \frac{\alpha}{1 + |\alpha|}, \quad (22)$$

we find that ADERF training on networks with the new, more sophisticated function (22) even beats the result achieved with MSE (see Table 3). For the other two functions the decrease of recognition rate arising from the use of ADERF is marginal. The absolute sigmoid AF trained with ADERF and the 3-RPP AF trained with both MSE and ADERF provide results which are better than the data presented in [2] and thus are also better than all other published results reviewed in that paper.



(a) ADERF q swept



(b) ADERF u swept

Figure 4. Recognition results G (solid line), success reliability R (dash-dotted) and the cleaned version of G (dashed) for training a 2-RPP network with the ADERF error function, varying ADERF parameters u and q . Upper: Parameter q is swept from 1 to 5 for a fixed value $u = 2$. Lower: Parameter u is swept from 1 to 5 for a fixed value $q = 5$.

V. CONCLUSION

The task of pattern recognition with feed-forward networks of formal neurons was introduced to be the general topic of this work.

A specific classification benchmark problem, namely the CENPARMI handwritten numeral database, served as a standard test for all methods developed within this work. The already published multi-resolution locally expanded high-order neural network architecture was explained and successfully developed further by changing the activation function of the neurons from continuously defined sigmoidal curves to piecewise defined, sigmoid-like polynomials. In this context, success is meant in terms of speed and correct classification rates in generalization mode, i.e. during normal operation after training, when the network is exposed to patterns not trained. The insufficiency of the piecewise defined polynomial activation function was proven to be compensable by using a new error function for training. This error function gains its advantage for being used in combination with the polynomial activation function from its inherent mechanism keeping neuron activations of the trained network low, thus avoiding the neurons to run into activation regions of constant output. For the CENPARMI dataset, the combination of piecewise polynomials and this new error function was shown to provide faster and better classification results than the work referred to as a starting point.

Err Funct		AF→	logistic	3-RPP	Abs sigmoid
MSE	G		97.09	97.22	96.69
	ΔG		± 0.05	± 0.09	± 0.06
	R		100	100	100
ADERF (2,6)	G		96.82	97.18	97.20
	ΔG		± 0.09	± 0.07	± 0.06
	R		100	100	100

Table 3. Recognition results G and measured training success reliability R for logistic, 3-RPP and absolute sigmoid AFs (in %). Each of them was trained with MSE and ADERF. For both error functions, a weight initialization of $W_{\max} = 0.25$ was chosen in order to maximize training

reliability for p-RPP. The parameters for the ADERF runs have been set to $u = 2$ and $q = 2$.

References

- [1] G. Castellano, A. M. Fanelli, C. Mencar. "An Empirical Risk Functional to Improve Learning in a Neuro-Fuzzy Classifier", *IEEE transactions on Systems, Man, and Cybernetics*, 34, pp. 725–731, February 2004.
 - [2] C.-L. Liu, J. H. Kim, R.-W. Dai. "Multiresolution Locally Expanded HONN for Handwritten Numeral Recognition", *Pattern Recognition Letters*, 18(10), pp.1019–1025, 1997.
 - [3] S. Haykin. *Neural Networks: A Comprehensive Foundation* 2nd Ed., Prentice Hall Inc. New Jersey, 1999.
 - [4] C.J. Hilditch. "Linear Skeletons from Square Cupboards", *Machine Intelligence*, 4, pp. 403–420, 1969.
 - [5] HK. Kwan. "Simple Sigmoid-Like Activation Function Suitable for Digital Hardware Implementation", *IEE Electronics Letters*, 28(15), pp. 1379–1380, 1992.
 - [6] K. Sunat. "Principles of Convergent Rate and Generalization Enhancement for Feedforward Sigmoid-Like Network". *PhD thesis, Chulalongkorn University Bangkok*, 2003.
 - [7] M. Zhang, S. Vassiliadis, and JG. Delgado- Frias. "Sigmoid Generators for Neural Computing Using Piecewise Approximation", *IEEE Trans Computers*, 45(2), pp. 1045–1049, 1996.
- Daniel Brüderle** was born in Offenburg, Germany, in 1978. He studied physics with focus on computer science in Heidelberg, Germany. He gained his diploma in physics at the Kirchhoff Institute in Heidelberg in 2004. He now works as a Ph.D. student in the "Electronic Vision(s)" group at the Kirchhoff Institute. His major field of interest is the application of hardware neural networks and liquid computing.
- Khamron Sunat** was born in Trad, Thailand, in 1965. He graduated in chemical engineering in Chulalongkorn university, Thailand, in 1989. He received his M.Sc. in computational science in 1998 and Ph.D. in computer science from Chulalongkorn university in 2004. He now works as a lecturer in the department of computer engineering at Mahanakorn university of technology, Thailand and joined in research group in Advanced Virtual and Intelligent Computing (AVIC) Research Center, Chulalongkorn University. His research interests are in neural networks, soft computing, fuzzy system and pattern recognition.
- Sirapat Chiewchanwattana** graduated in statistics from Khon Kaen university, Thailand. She received her M.Sc. in computer science from the National Institute of Development Administration, Thailand. She is now a Ph.D. student at Chulalongkorn university, Thailand. She works as a lecturer at Khon Kaen university and joined in research group in Advanced Virtual and Intelligent Computing (AVIC) Research Center, Chulalongkorn University. Her research interests are in neural networks, soft computing, and pattern recognition.
- Chidchanok Lursinsap** was born in Bangkok, Thailand in 1956. He graduated in computer engineering from Chulalongkorn University, Thailand. He received MS and PhD in computer science from University of Illinois at Urbana-Champaign, USA. He is a professor in computer science at Department of Mathematics and head of AVIC Center at Chulalongkorn University. His research interests are in neural computing, bioinformatics, and plant simulation.
- Suchada Siripant** graduated in mathematics from Elon College, USA in 1972. She received MA in mathematics from University of North Carolina, USA in 1974. She is an associate professor in computer science at Department of Mathematics. Her research interests are in computer graphic, visualization and plant modeling.

Author Biographies