Annals of Mathematics and Artificial Intelligence (2022) 90:425–452
https://doi.org/10.1007/s10472-022-09786-2

REGULAR SUBMISSION

On the antiderivatives of $x^p/(1-x)$ with an application to optimize loss functions for classification with neural networks

Andreas Knoblauch[1]

Accepted: 10 January 2022 / Published online: 18 March 2022

Abstract
Supervised learning in neural nets means optimizing synaptic weights $\mathbf{W}$ such that outputs $\mathbf{y}(\mathbf{x};\mathbf{W})$ for inputs $\mathbf{x}$ match as closely as possible the corresponding targets $\mathbf{t}$ from the training data set. This optimization means minimizing a loss function $\mathcal{L}(\mathbf{W})$ that usually motivates from maximum-likelihood principles, silently making some prior assumptions on the distribution of output errors $\mathbf{y}-\mathbf{t}$. While classical crossentropy loss assumes triangular error distributions, it has recently been shown that generalized power error loss functions can be adapted to more realistic error distributions by fitting the exponent q of a power function used for initializing the backpropagation learning algorithm. This approach can significantly improve performance, but computing the loss function requires the antiderivative of the function $f(y) := y^{q-1}/(1-y)$ that has previously been determined only for natural $q \in \mathbb{N}$. In this work I extend this approach for rational $q = n/2^m$ where the denominator is a power of 2. I give closed-form expressions for the antiderivative $\int f(y)dy$ and the corresponding loss function. The benefits of such an approach are demonstrated by experiments showing that optimal exponents q are often non-natural, and that error exponents q best fitting output error distributions vary continuously during learning, typically decreasing from large $q > 1$ to small $q < 1$ during convergence of learning. These results suggest new adaptive learning methods where loss functions could be continuously adapted to output error distributions during learning.

Keywords Supervised learning · Classification · Crossentropy · Power error loss function · Deep learning · Incomplete beta function · Hypergeometric function

Mathematics Subject Classification (2010) MSC 26A42 · 33B20 · 68T05 · 68T10 · 92B20

This work was supported by Deutsches Bundesministerium für Verkehr und digitale Infrastruktur (Modernitätsfonds/mFUND) via project AI4Infra (FKZ 19F2112C), and by Ministerium für Wirtschaft, Arbeit und Wohnungsbau Baden-Württemberg (KI-Innovationswettbewerb BW für Verbundforschungsprojekte) via project KI4Audio (FKZ 36-3400.7/98). The author as well acknowledges support by the state of Baden-Württemberg through bwHPC.

✉ Andreas Knoblauch
knoblauch@hs-albsig.de

[1] KEIM Institute, Albstadt-Sigmaringen-University, Jakobstrasse 6, 72458, Albstadt, Germany

1 Introduction

Special functions like the beta, gamma or hypergeometric functions have many applications in various domains including probability theory, computational chemistry and statistical physics [2, 8, 39]. They are often used to express antiderivatives that are otherwise difficult to compute. Here I focus on the antiderivative

$$F(y) := \int_0^y \frac{y^{q-1}}{1-y} dy \quad \left(= \lim_{b \to 0} B(y; q, b) \right) \tag{1}$$

being a limit case of the incomplete beta function $B(y; a, b) := \int_0^y t^{a-1}(1-t)^{b-1} dt$ defined for $\mathrm{Re}(a), \mathrm{Re}(b) > 0$ and $y \in (0; 1)$. It is easy to see that $F(y)$ can also be written in terms of the hypergeometric function (see Proposition 1), but for many applications it is desirable to have expressions in closed form using only common functions that can be automatically derived and efficiently computed around the poles (here $y \to 1$). One such application is machine learning where the antiderivative $F(y)$ relates to loss functions like cross entropy (for $q = 1$) that are minimized to solve classification and related AI tasks [4]. This needs typically large amounts of annotated training data $\mathcal{D} := \{(\mathbf{x}_n, \mathbf{t}_n) | n = 1, ..., N\}$ for supervised learning of a prediction model with

$$\mathbf{y}_n := \mathbf{y}(\mathbf{x_n}; \mathbf{W}) \overset{!}{\approx} \mathbf{t}_n \qquad \in \mathbb{R}^K \tag{2}$$

for inputs $\mathbf{x}_n \in \mathbb{R}^D$ and targets or labels $\mathbf{t}_n$. Specifically, the learning task is to find "good" parameters $\mathbf{W}$ such that the model function $\mathbf{y}(\mathbf{x}; \mathbf{W})$ applied to the inputs $\mathbf{x}_n$ reproduces the annotation labels $\mathbf{t}_n$ as closely as possible. The model performance can be quantified by a loss function $\mathcal{L}(\{(\mathbf{y}_n, \mathbf{t}_n) | n = 1, \ldots, N\})$ evaluating the differences between model outputs $\mathbf{y}_n$ and targets $\mathbf{t}_n$. For example, for binary or multi-label classification tasks with binary labels $t_{nk} \in \{0, 1\}$ a good choice is binary cross entropy (BCE)

$$\mathcal{L}^{\mathrm{BCE}} := -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log(y_{nk}) + (1 - t_{nk}) \log(1 - y_{nk}) \tag{3}$$

whereas for multi-class classification or regression problems we may use categorical cross entropy $\mathcal{L}^{\mathrm{CCE}} := -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \log(y_{nk})$ with one-hot coding ($t_{nk} \in \{0, 1\}$, $\sum_k t_{nk} = 1$) or sum-of-squared-error $\mathcal{L}^{\mathrm{SSE}} := \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - \mathbf{t}_n||^2$ with $t_{nk} \in \mathbb{R}$ [4, 7, 11, 17, 22, 27, 35].

In the last decade, deep neural network models have become dominant for applications related to classification including object recognition and detection, image segmentation, speech understanding, autonomous driving, or robot control [12, 14, 21, 30, 34, 38]. This success can be attributed to an improved understanding of large-scale deep neural architectures and solving earlier problems like vanishing gradients blocking learning progress [3]. For example, to overcome such problems, improved activation function, weight initialization, regularization, and optimization methods have been developed [10, 13, 16, 18, 31, 37]. The current work complements these efforts by proposing a new family of improved loss functions based on the antiderivative (1) that enables continuous adaptation to training data or learning progress as explained in the following.

In deep neural networks, loss functions are typically minimized by gradient descent using the error backpropagation algorithm [4, 5, 28, 32, 33, 40]: After forward-propagating an input vector $\mathbf{x}_n$ through the network, the backpropagation algorithm initializes so-called

error signals

$$\delta_{nk} := \frac{\partial \mathcal{L}_n}{\partial a_{nk}} \tag{4}$$

for each output unit with firing rate $y_{nk} := \sigma_k(a_{nk})$ and dendritic activation potential $a_{nk} := \sum_j W_{kj} z_{nj}$ computed from a typically sigmoidal activation function σ_k. Similarly, for each hidden neuron j, the firing rate $z_{nj} := \sigma_j(a_j)$ with $a_j := \sum_j W_{kj} z_{ni}$ is computed recursively in the forward pass, where $z_{ni} := a_{ni} := x_{ni}$ for input units. After the initialization (4), error signals are backpropagated through the transposed synaptic weights $\mathbf{W}^{\mathrm{T}}$ towards the input layer using the recursion $\delta_{nj} = h'(a_{nj}) \sum_k W_{kj} \delta_k$. After this backward pass, each neuron j knows both its firing rate z_{nj} and error signal δ_{nj}. With this each synapse W_{ji} can compute its partial derivative $\frac{\partial \mathcal{L}_n}{\partial W_{ji}} = \frac{\partial \mathcal{L}_n}{\partial a_{nj}} \frac{\partial a_{nj}}{\partial W_{ji}} = \delta_{nj} z_{ni}$ as the product of postsynaptic error signal and presynaptic firing rate, and thus the corresponding weight change according to (stochastic) gradient descent is

$$\Delta W_{nji} := -\eta \frac{\partial \mathcal{L}_n}{\partial W_{ji}} = -\eta \delta_{nj} z_{ni} \tag{5}$$

where $\eta > 0$ is the learning rate. Thus, the initialization (4) determines synaptic learning (5) and should therefore be chosen as simple as possible for the sake of biological plausibility and computational efficiency. Indeed, for the three most commonly employed settings of I) regression with $\mathcal{L}^{\mathrm{SSE}}$ and linear outputs $y_{nk} = a_{nk} \in \mathbb{R}$, II) binary classification with $\mathcal{L}^{\mathrm{BCE}}$ and sigmoidal outputs $y_{nk} = \sigma(a_{nk}) \in (0; 1)$ with the logistic sigmoid $\sigma(a) := 1/(1+e^{-a})$, and III) categorical (multi-class) classification with $\mathcal{L}^{\mathrm{CCE}}$ and softmax outputs $y_{nk} = \mathcal{S}_k(\mathbf{a}_n) := \frac{\exp(-a_{nk})}{\sum_j \exp(-a_{nj})} \in (0; 1)$, the initialization (4) becomes simply the difference

$$\delta_{nk} = y_{nk} - t_{nk} = -\epsilon_{nk} \qquad \text{for} \quad \epsilon_{nk} := t_{nk} - y_{nk} \tag{6}$$

between model outputs and targets, that is (up to the sign), the output error ϵ_{nk}. However, as I have argued in previous work [19], such settings do not always maximize likelihood or other relevant performance measures like accuracy, as they rely on unrealistic prior assumptions like a triangular distribution of output errors, which is often not fulfilled. Therefore a novel more general initialization of error signals in the output layer has been suggested (see eq. 3.1 in [19]) using powers

$$\delta_{nk}(y_{nk}, t_{nk}) := (1 - 2t_{nk}) \cdot |y_{nk} - t_{nk}|^q \tag{7}$$

of the output errors with exponent $q > 0$. Interestingly, this new initialization method can significantly speed up learning and improve convergence of the backpropagation algorithm by adapting the exponent q to the true distribution of output errors [19]. However, many software platforms for machine learning like Keras, Tensorflow, and PyTorch [1, 6, 29]) do not directly initialize error signals like in (6), (7), but instead compute gradients via automatic differentiation [23] of the loss function. For this we require an explicit symbolic representation of the corresponding loss functions which, as we will see, involves integrating (4) for (7) or determining appropriate expressions for the antiderivative (1). While for the special case $q \in \mathbb{N}$ this problem is easy to solve, and corresponding loss functions have already been determined previously [19], it is more demanding to integrate (7) for general $q \in \mathbb{R}^+$, and the corresponding loss functions have been unknown so far. However, continually adapting q to training data and learning progress with arbitrary distributions of output errors requires also $q \notin \mathbb{N}$, including the case $0 < q < 1$.

In this paper I compute the loss functions that correspond to the power function initialization of error signals (7) for rational exponents of the form $q = n/2^m$, where the numerator

is a positive integer and the denominator is a power of 2 ($n \in \mathbb{N}, m \in \mathbb{N}_0$). With this it becomes possible to approximate the loss functions corresponding to (7) for any $q \in \mathbb{R}^+$ with arbitrary precision. To this end the paper is organized as follows: Section 2 shows that determining the loss functions corresponding to (7) for binary classification problems involves the antiderivative (1) and briefly recapitulates the solution for $q \in \mathbb{N}$. Section 3 determines (1) for the more general case of positive rational exponents $q = n/2^m$, where the most convenient final form is given by (30) in Theorem 3. Section 4 shows results from numerical learning experiments verifying correctness and demonstrating the benefits of the new loss functions. Finally, Section 5 gives a summary and discussion of the results.

2 Generalized loss functions for exponents $q \in \mathbb{N}$

In order to determine the generalized loss function that is minimized by the backpropagation algorithm, we have to integrate (4) using the generalized error signal initialization (7). Here it is sufficient to consider the loss contribution $\mathcal{L}_{nk}$ of one output unit $y_{nk} = \sigma_k(a_{nk})$ after presenting the training vector tuple $(\mathbf{x}_n, \mathbf{t}_n)$. Thus, up to an additive constant, the loss function corresponding to the generalized initialization (7) with exponent $q > 0$ is

$$\begin{aligned}\tilde{\mathcal{L}}^{(q)}_{nk}(y_{nk}, t_{nk}) &:= \int_{-\infty}^{a_{nk}} \frac{\partial \mathcal{L}_n}{\partial a_{nk}} da_{nk} = \int_{-\infty}^{a_{nk}} (1-2t_{nk}) \cdot |\sigma_k(a_{nk}) - t_{nk}|^q da_{nk} \\ &= \int_0^{y_{nk}} \frac{(1-2t_{nk}) \cdot |y_{nk} - t_{nk}|^q}{\sigma_k'(a_{nk})} dy_{nk} \end{aligned} \tag{8}$$

where the last equation follows from the substitution $y_{nk} = \sigma_k(a_{nk})$ with the derivative $\frac{dy_{nk}}{da_{nk}} = \sigma_k'(a_{nk})$. For the most common case of logistic sigmoids in the output layer, $\sigma_k(a_{nk}) = \sigma(a_{nk}) \in (0,1)$ for $\sigma(a) := \frac{1}{1+e^{-a}}$ with $\sigma'(a) = \sigma(a)(1-\sigma(a))$, we have $\sigma_k'(a_{nk}) = \sigma(a_{nk})(1-\sigma(a_{nk})) = y_{nk}(1-y_{nk})$ and thus, skipping variable indices $y := y_{nk}$ and $t := t_{nk}$ for brevity,

$$\tilde{\mathcal{L}}^{(q)}_{nk}(y,t) = \int_0^y \frac{(1-2t)|y-t|^q}{y(1-y)} dy = \begin{cases} \int_0^y \frac{y^{q-1}}{1-y} dy = F(y) & , t = 0 \\ -\int_0^y \frac{(1-y)^{q-1}}{y} dy = F(1-y) - F(1) & , t = 1 \end{cases} \tag{9}$$

where the last equation (for $t = 1$) follows with the substitution $y \mapsto 1-y$ and corresponds to an improper integral with diverging $F(1) \to \infty$. This shows that in order to determine the loss function for generalized error signal initialization, we have to find the antiderivative (1), where we choose the additive constants in each case such that the resulting loss is zero for correct predictions:

Theorem 1 (Loss for power function error initialization) *For feed-forward neural networks using the logistic sigmoid function $y = \sigma(a) \in (0;1)$ in the output layer, the loss function corresponding to the power function initialization (7) of the error signals for backpropagation with exponent $q > 0$ is*

$$\mathcal{L}^{(q)}_{nk}(y,t) = (1-t)F(y) + tF(1-y) \tag{10}$$

where $F(y) = \int_0^y \frac{y^{q-1}}{1-y} dy$ is the antiderivative (1) being a limit case of the incomplete beta function. In particular, the resulting loss function has zero baseline and is symmetric,

$$\mathcal{L}^{(q)}_{nk}(0,0) = \mathcal{L}^{(q)}_{nk}(1,1) = 0 \quad \textit{and} \quad \mathcal{L}^{(q)}_{nk}(y,1) = \mathcal{L}^{(q)}_{nk}(1-y,0) \tag{11}$$

Proof : The theorem follows immediately from (9) by merging the two cases $t = 0$ and $t = 1$, and noting that $F(0) = 0$, such that we just have to skip the offset $F(1)$ for $t = 1$ in order to get zero loss in case the neural network makes a correct prediction $y = t \in \{0, 1\}$. □

Therefore, the remainder of the paper deals mainly with determining closed form expressions for the antiderivative $F(y)$. This is particularly easy for natural exponents $q \in \mathbb{N}$ [19]:

Theorem 2 (Power error loss for natural exponent $q \in \mathbb{N}$) *For $q \in \mathbb{N}$ the antiderivative (1) and corresponding loss function of Theorem 1 become*

$$F(y) = -\log(1-y) - \sum_{i=1}^{q-1} \frac{y^i}{i} \tag{12}$$

$$= -\log(1-y) + \sum_{r=0}^{q-2} \binom{q-1}{r} \frac{(-1)^{q-r}}{q-1-r} \left((1-y)^{q-1-r} - 1\right) \tag{13}$$

$$\mathcal{L}_{nk}^{(q)}(y,t) = \mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) - \sum_{i=1}^{q-1} \frac{(1-t)y^i + t(1-y)^i}{i} \tag{14}$$

$$= \mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) - \sum_{i=0}^{q-1} a_i^{(q)} y^i \qquad \textit{with coefficients} \tag{15}$$

$$a_0^{(q)} := t \sum_{j=1}^{q-1} \frac{1}{j} \qquad \textit{and}$$

$$a_i^{(q)} := \frac{\left((-1)^i \sum_{j=i}^{q-1} \binom{j-1}{i-1} - 1\right) t + 1}{i} \qquad \textit{for } i = 1, \ldots, q-1$$

the latter being the binary cross entropy loss $\mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) := -t\log(y) - (1-t)\log(1-y)$ as in (3) minus a polynomial $a^{(q)}(y) := \sum_{i=0}^{q-1} a_i^{(q)} y^i$ in y of degree $q-1$, where the coefficients $a_i^{(q)}$ are linear in t.

For the proof of this and the following Theorems and Propositions see Appendix A. It is convenient to rewrite (15) in terms of another set of coefficients $b_i^{(q)}$,

$$\mathcal{L}_{nk}^{(q)}(y,t) = \mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) - \sum_{i=0}^{q-1} a_i y^i = \mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) - b_0^{(q)} t - \sum_{i=1}^{q-1} \frac{(-1)^i b_i^{(q)} t + 1}{i} y^i$$

$$= \mathcal{L}_{nk}^{\mathrm{BCE}}(y,t) - b_0^{(q)} t + \frac{b_1^{(q)} t - 1}{1} y - \frac{b_2^{(q)} t + 1}{2} y^2 + \frac{b_3^{(q)} t - 1}{3} y^3 - \ldots \tag{16}$$

where the coefficients $b_0 := \sum_{j=1}^{q-1} \frac{1}{j}$ and $b_i := \sum_{j=i}^{q-1} \binom{j-1}{i-1} - (-1)^i$ for $i = 1, 2, \ldots, q-1$ can be precomputed as shown by Table 1 in Appendix A for $q = 1, 2, \ldots, 12$.

3 The antiderivative $\int \frac{y^{q-1}}{1-y} dy$ for real and rational exponents $q > 0$

We see from (10) that computing the generalized loss functions $\mathcal{L}_{nk}^{(q)}(y,t)$ requires the antiderivative (1) with

$$F(y) := \tilde{F}(y) - \tilde{F}(0) \quad \text{for} \quad \tilde{F}(y) := \int f(y)dy \quad \text{and} \quad f(y) := \frac{y^{q-1}}{1-y} \tag{17}$$

It is easy to verify that $F(y)$ can be expressed in terms of the hypergeometric function ${}_2F_1(a,b;c,z) := \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{z^n}{n!}$ with the $(x)_n := \pi_{i=0}^{n-1} x + i$ being rising Pochhammer symbols:

Proposition 1 ($F(y)$ for real q and outputs $|y| < 1$) *For $q \in \mathbb{R}\backslash\{0, -1, -2, \cdots\}$ and $|y| < 1$ we have*

$$F(y) = \frac{y^q}{q} {}_2F_1(q, 1; q+1, y) = y^q \sum_{n=0}^{\infty} \frac{y^n}{q+n} = y^q \left(\frac{1}{q} + \frac{y}{q+1} + \frac{y^2}{q+2} + \ldots\right) \tag{18}$$

Like with the incomplete beta function in (1), expressing $F(y)$ in terms of the limit of an infinite sum is not viable as current software libraries employing automatic differentiation (like Tensorflow or PyTorch) cannot efficiently handle such expressions. For example, using (18) to approximate $F(y) \to \infty$ for $y \to 1$ would need to sum a very large number of terms (as each term except the first one is < 1). Instead, we have to find finite expressions for $F(y)$ in terms of common functions that have simple derivatives. With computer algebra systems (CAS) like Mathematica or Maxima [24, 26] it is possible to further explore (17). Trying some particular values shows that for rational exponents $q = \frac{n}{N}$ with $n, N \in \mathbb{N}$ appropriate antiderivatives exist in closed form. However, for larger values of the denominator N the results of the CAS are inconvenient and involve complicated sums over complex roots. Still, the results are relatively simple if $N = 2^m$ is a power of 2. Therefore we focus on the case $q = \frac{n}{2^m} \in \mathbb{Q}^+$ for $n \in \mathbb{N}$, $m \in \mathbb{N}_0$, which is sufficient to approximate $F(y)$ for any $q \in \mathbb{R}_0^+$ with arbitrary precision. We start with the case $0 < q < 1$ and $0 \leq y < 1$ and then generalize to $y \geq 0$ and $q > 0$. Trying the CAS for some special cases $q = n/2^m \in (0; 1)$ with $n, m \in \mathbb{N}$ and $0 < n < 2^m =: N$ gives the correct hypothesis for the antiderivative $\tilde{F}(y)$:

Proposition 2 (Complex $F(y)$ for exponents $0 < q < 1$, outputs $0 \leq y < 1$)

$$\tilde{F}(y) = -\sum_{Z:Z^N=1} Z^n \log\left(y^{1/N} - Z\right) \tag{19}$$

$$= -\sum_{k=0}^{N-1} \log\left(y^{1/N} - e^{j2\pi\frac{k}{N}}\right) \cdot e^{j2\pi\frac{kn}{N}} \quad = \text{DFT}\{s[k]\}[N-n] \tag{20}$$

$$\tilde{F}(0) = \frac{-\pi \sin(2\pi\frac{n}{N})}{1 - \cos(2\pi\frac{n}{N})} - j\pi \tag{21}$$

where j is the imaginary unit and the sum is over the $N := 2^m$th complex unit roots $Z = e^{j2\pi\frac{k}{N}}$ for $k = 0, 1, \ldots, N-1$, and thus $\tilde{F}(y)$ corresponds to the $N-$nth value of the Discrete Fourier Transform (DFT) of the discrete N-periodic signal $s[k] := \log\left(y^{1/N} - e^{j2\pi\frac{k}{N}}\right)$.

Via (17), Proposition 2 provides closed-form expressions for $F(y)$ for $0 < q, y < 1$ that involve complex numbers. However, by definition (1), there should exist an equivalent real-valued representation for $F(y) \in \mathbb{R}$. In the following we simplify computation of $F(y)$ by reducing the number of terms and eliminating complex-valued expressions:

Proposition 3 (Real-valued representation of $F(y)$ for $0 < q < 1$ and $0 \leq y < 1$) *Let $q = n/2^m \in (0;1)$ with $n, m \in \mathbb{N}$ and $0 \leq y < 1$. Then (17) with (20) and (21) simplifies for $N := 2^m \geq 4$ to the real-valued expression*

$$
\begin{aligned}
F(y) = \tilde{F}(y) - \tilde{F}(0) &= \frac{\pi \sin(2\pi \frac{n}{N})}{1 - \cos(2\pi \frac{n}{N})} + \log\left(\frac{(1 + y^{1/N})^{(-1)^{n+1}}}{1 - y^{1/N}}\right) \\
&- \cos(\pi \frac{n}{2}) \cdot \log(y^{2/N} + 1) - 2\sin(\pi \frac{n}{2}) \cdot \arctan(y^{-1/N}) \\
&- 2 \sum_{k=1}^{N/4-1} \cos(2\pi \frac{kn}{N}) \cdot \log\left(r_k \cdot r_{N/2-k}^{(-1)^n}\right) - \sin(2\pi \frac{kn}{N}) \cdot \left(\varphi_k - (-1)^n \varphi_{N/2-k}\right)
\end{aligned}
\tag{22}
$$

where r_k and φ_k must be computed from (37) or (40). For the remaining case $N = 2$ corresponding to $q = 1/2$ we have

$$
F(y) = \int_0^y \frac{1}{(1-y)y^{1/2}} dy = \log\left(\frac{1+\sqrt{y}}{1-\sqrt{y}}\right) \tag{23}
$$

If $q = n/N = n/2^m \in (0;1)$ is in reduced form, we gain a considerable simplification because then n is odd for any $m \geq 1$, we have $(-1)^n = -1$, and the case disctinctions involved in computing r_k and φ_k get aligned:

Proposition 4 (Simplification for irreducible $0 < q = n/2^m < 1, 0 \leq y < 1$) *For $q = n/2^m \in (0;1)$ with $n, m \in \mathbb{N}$, $N := 2^m \geq 4$, odd $n = 1, 3, 5, \ldots, N-1$ (that is, n/N is irreducible), and $0 \leq y < 1$, it is*

$$
2 \sum_{k=1}^{N/4-1} \sin(2\pi \frac{kn}{N}) = \frac{\sin(2\pi \frac{n}{N})}{1 - \cos(2\pi \frac{n}{N})} - (-1)^{(n-1)/2} \tag{24}
$$

$$
\begin{aligned}
F(y) = \log\left(\frac{1 + y^{1/N}}{1 - y^{1/N}}\right) &+ (-1)^{(n-1)/2} \cdot 2\arctan(y^{1/N}) \\
&+ \sum_{k=1}^{N/4-1} \cos(2\pi \frac{kn}{N}) \cdot \log\left(\frac{1 + 2y^{1/N}\cos\left(2\pi \frac{k}{N}\right) + y^{2/N}}{1 - 2y^{1/N}\cos\left(2\pi \frac{k}{N}\right) + y^{2/N}}\right) \\
&+ \sum_{k=1}^{N/4-1} 2\sin(2\pi \frac{kn}{N}) \cdot \arctan\left(\frac{2\sin\left(2\pi \frac{k}{N}\right) y^{1/N}}{1 - y^{2/N}}\right)
\end{aligned}
\tag{25}
$$

So far we have determined the antiderivative $F(y)$ for $0 \leq y < 1$, which is sufficient for computing loss functions for binary classification, where y corresponds to a class probability. For other applications, it may be useful to include also the case $y > 1$:

Proposition 5 (Including the case $y > 1$ for $0 < q < 1$) *The antiderivative of $f(y) = \frac{1}{(1-y)y^{1-q}}$ for $y \in \mathbb{R}_0^+ \backslash \{1\}$ assuming irreducible $q = n/N \in (0;1)$ with $N := 2^m \geq 4$ and*

odd $n = 1, 3, 5, \ldots N-1$ *is*

$$\begin{aligned}\int \frac{1}{(1-y)y^{1-q}}dy \;=\; F(y) &= \log\left(\frac{1+y^{1/N}}{|1-y^{1/N}|}\right) + (-1)^{(n-1)/2}\cdot 2\arctan(y^{1/N}) \\ &+ \sum_{k=1}^{N/4-1} \cos(2\pi\frac{kn}{N})\cdot\log\left(\frac{1+2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}{1-2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}\right) \\ &+ \sum_{k=1}^{N/4-1} 2\cdot\sin(2\pi\frac{kn}{N})\cdot\arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{1-y^{2/N}}\right) \qquad (26)\end{aligned}$$

whereas for the remaining case $N = 2$ *corresponding to* $q = 1/2$ *we have*

$$\int \frac{1}{(1-y)y^{1/2}}dy = \log\left(\frac{1+\sqrt{y}}{|1-\sqrt{y}|}\right) \qquad (27)$$

Let us now re-address the antiderivative $\int \frac{y^{q-1}}{1-y}dy$ for $q = n/N > 1$ and $y \geq 0$:

Proposition 6 (Rational exponents $q > 1$ with $y \geq 0$) *Let* $q = n/N > 1$ *be irreducible with* $N := 2^m$, $m \in \mathbb{N}$, *and odd* $n = 1, 3, 5, \ldots N-1$. *Further, let* $\eta := n - N$, $\tilde{n} := n \bmod N$, *and* $M := (\eta - 1) \operatorname{div} N$. *Then the antiderivative of* $f(y) = \frac{y^{q-1}}{(1-y)}$ *for* $y \in \mathbb{R}_0^+ \backslash \{1\}$ *is for* $N \geq 4$

$$\begin{aligned}\int \frac{y^{q-1}}{1-y}dy \;=\; &-Ny^{\tilde{n}/N}\sum_{i=0}^{M}\frac{y^{M-i}}{n-(i+1)\cdot N} \\ &+\log\left(\frac{1+y^{1/N}}{|1-y^{1/N}|}\right) + (-1)^{(\tilde{n}-1)/2}\cdot 2\arctan(y^{1/N}) \\ &+ \sum_{k=1}^{N/4-1} \cos(2\pi\frac{k\tilde{n}}{N})\cdot\log\left(\frac{1+2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}{1-2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}\right) \\ &+ \sum_{k=1}^{N/4-1} 2\cdot\sin(2\pi\frac{k\tilde{n}}{N})\cdot\arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{1-y^{2/N}}\right) \;=\; F(y) \qquad (28)\end{aligned}$$

whereas for $N = 2$ *we have*

$$\int \frac{y^{q-1}}{1-y}dy = F(y) = -Ny^{\tilde{n}/N}\sum_{i=0}^{M}\frac{y^{M-i}}{n-(i+1)\cdot N} + \log\left(\frac{1+y^{1/N}}{|1-y^{1/N}|}\right) \qquad (29)$$

To account for the different ranges of the variable y and the exponent q, we can finally merge the results of Theorem 2 and Propositions 5,6 to a unifying theorem using the Heaviside function $H(y)$ and the discrete Dirac impulse $\delta[n]$ defined in the proofs of Propositions 4 and 3 (that is, $H(y) = 1$ for $y \geq 0$ and $H(y) = 0$ otherwise, and $\delta[n] = 1$ for $n = 0$ and $\delta[n] = 0$ otherwise):

Theorem 3 (Final antiderivative $F(y)$ for $q > 0$ and $y \geq 0$) *For irreducible* $q = n/N \in \mathbb{Q}^+$ *with* $n \in \mathbb{N}$, $N = 2^m \in \mathbb{N}$, $m \in \mathbb{N}_0$, $\tilde{n} := n \bmod N$, *and* $\tilde{q} := n \operatorname{div} N$, *an antiderivative*

of $f(y) := \frac{y^{q-1}}{1-y}$ *is given by*

$$\begin{aligned}\int \frac{y^{q-1}}{1-y}dy = F(y) := &-Ny^{\tilde{n}/N}\cdot\left(\frac{H(N-2)}{n-\tilde{q}\cdot N}+\sum_{i=1}^{\tilde{q}-1}\frac{y^i}{n-(\tilde{q}-i)\cdot N}\right)\cdot H(q-1)\\ &+\log\left(\frac{1+y^{1/N}\cdot H(N-2)}{|1-y^{1/N}|}\right)\\ &+(-1)^{(\tilde{n}-1)/2}\cdot 2\arctan(y^{1/N})\cdot H(N-4)\\ &+\sum_{k=1}^{N/4-1}\cos(2\pi\frac{k\tilde{n}}{N})\cdot\log\left(\frac{1+2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}{1-2y^{1/N}\cos\left(2\pi\frac{k}{N}\right)+y^{2/N}}\right)\\ &+\sum_{k=1}^{N/4-1}2\cdot\sin(2\pi\frac{k\tilde{n}}{N})\cdot\arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{1-y^{2/N}}\right)\end{aligned}\tag{30}$$

where the last two sums are relevant only for $N \geq 2^3 = 8$.
$F(y)$ *is strictly increasing for* $0 \leq y < 1$ *and strictly decreasing for* $y > 1$.
Furthermore, $F(0) = 0$, $\lim_{y\to 1} F(y) = \infty$, *and for large* $y \to \infty$

$$\begin{aligned}F(y) \approx &-Ny^{\tilde{n}/N}\cdot\left(\frac{H(N-2)}{n-\tilde{q}\cdot N}+\sum_{i=1}^{\tilde{q}-1}\frac{y^i}{n-(\tilde{q}-i)\cdot N}\right)\cdot H(q-1)\\ &-\log(y-1)\cdot\delta[N-1]+(-1)^{(\tilde{n}-1)/2}\cdot\pi\cdot H(N-4)\\ &\begin{cases}\to 0 & ,q=\frac{1}{2}\\ \to(-1)^{(\tilde{n}-1)/2}\cdot\pi & ,0<q<1, N\geq 4\\ =-\log(y-1)\to-\infty & ,q=1\\ \sim-\frac{1}{q-1}y^{q-1}\to-\infty & ,q>1\end{cases}\end{aligned}\tag{31}$$

Note that, by adding a constant $C := -(-1)^{(\tilde{n}-1)/2}\cdot\pi = (-1)^{(\tilde{n}+1)/2}\cdot\pi$ for $N \geq 4$ and $y > 1$ we obtain an equivalent antiderivative

$$\hat{F}(y) := F(y) + (-1)^{(\tilde{n}+1)/2}\cdot\pi\cdot H(N-4)\cdot H(y-1)\tag{32}$$

for $f(y)$ with the same properties as $F(y)$ in Theorem 3, except that $\hat{F}(y)$ has a unique limit $\lim_{y\to\infty}\hat{F}(y) = 0$ for $0 < q < 1$. Note also that C corresponds to the constant (50) that we have skipped previously to get $F(0) = 0$ for all $q > 0$.

Figure 1 illustrates $f(y)$ and $F(y)$ from (30) for different values of the exponent q. Note that the dependency of $F(y)$ on q is monotonic for $0 \leq y < 1$, but non-monotonic and even discontinuous for finite $1 < y \ll \infty$, where monotonicity and continuity are restored in the limit of very large $y \to \infty$ due to (31).

I have also verified $F(y)$ from (30) by numerical differentiation using Matlab [25] with variable precision arithmetics (function vpa with a precision of 500 decimal digits) to compute relative errors between the numerical derivative $F'_{\text{num}}(y) := \frac{F(y+\Delta y)-F(y)}{\Delta y}$ and $f(y) := \frac{y^{q-1}}{1-y}$ sampling from $y \in [y_0; 1-y_0]\cup[1+y_0; 1000]$. The relative errors were largest around the poles at $y \approx 0$ and $y \approx 1$, whereas apart from the poles, they generally decreased for larger y and increased for larger q. For minimal pole distance $y_0 = 10^{-6}$ and difference $\Delta y = 10^{-20}$, the relative error for $q \leq 50$ was always below 10^{-12}, thus confirming Theorem 3.

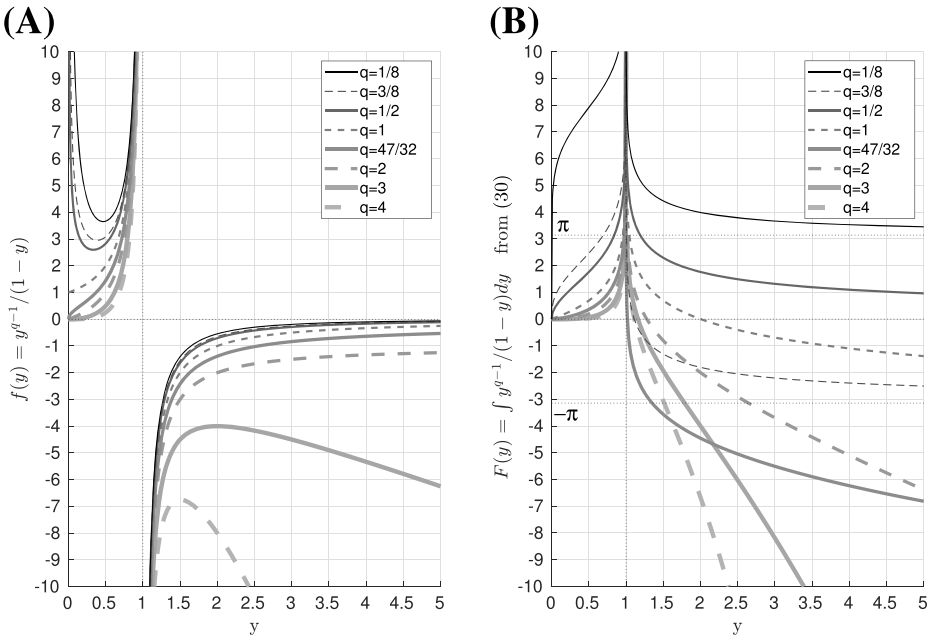

Fig. 1 Illustration of $f(y) = \frac{y^{q-1}}{1-y}$ and the antiderivative $F(y) = \int f(y)dy$ as computed from (30) for exponents $q \in \{\frac{1}{8}, \frac{3}{8}, \frac{1}{2}, 1, \frac{47}{32}, 2, 3, 4\}$

4 Applying the generalized loss function to neural networks

The purpose of the following experiments is, first, to verify Theorem 1 with (30) as used for implementing $\mathcal{L}_{nk}^{(q)}$ in Appendix B, and, second, to demonstrate the usefulness of the loss function $\mathcal{L}_{nk}^{(q)}$ for general $q > 0$. For the verification, I have implemented a simple recurrent neural network (Simple RNN; see Fig. 2A) for sequence classification and trained it on the Embedded Reber Grammar data set (see Appendix B for details; cf. [19]). Figure 2 shows results from single trial learning experiments using fixed standard parameters (initial learning rate $\eta_0 = 0.001$ and minibatch size MBS=4) without any further hyperparameter optimization: Fig. 2B shows test error as function of learning epoch for different error exponents $q \in \{\frac{1}{4}, \frac{1}{2}, 1, \frac{711}{256} \approx 2.777, \frac{3157}{512} \approx 6.166\}$. The exponents q have been chosen to test all (conditional) terms of (30) in sufficient detail. It can be seen that the two implementations (Keras vs. custom) yield very similar, but not identical results. At least in the initial phase of learning, test errors are virtually identical for both implementations, suggesting the correctness of (10), (30), and the implementation in Appendix B. To further confirm correctness, Fig. 2C shows for the custom implementation the maximum relative error of gradients estimated from backpropagation compared to computing gradients from the partial derivatives of the loss function with respect to all synaptic weights (where maximum is over all partial gradient vectors for synaptic connections A, B, U, and the two bias vectors of u and y). It can be seen that, at least initially, all relative gradient errors are below 10^{-6}, which finally confirms the correctness of (10), (30), and the implementation in Appendix B. During learning, relative errors typically increase, but are always below 10^{-4}. The increase is most pronounced for large $q \gg 1$ or large denominators of q. This increase

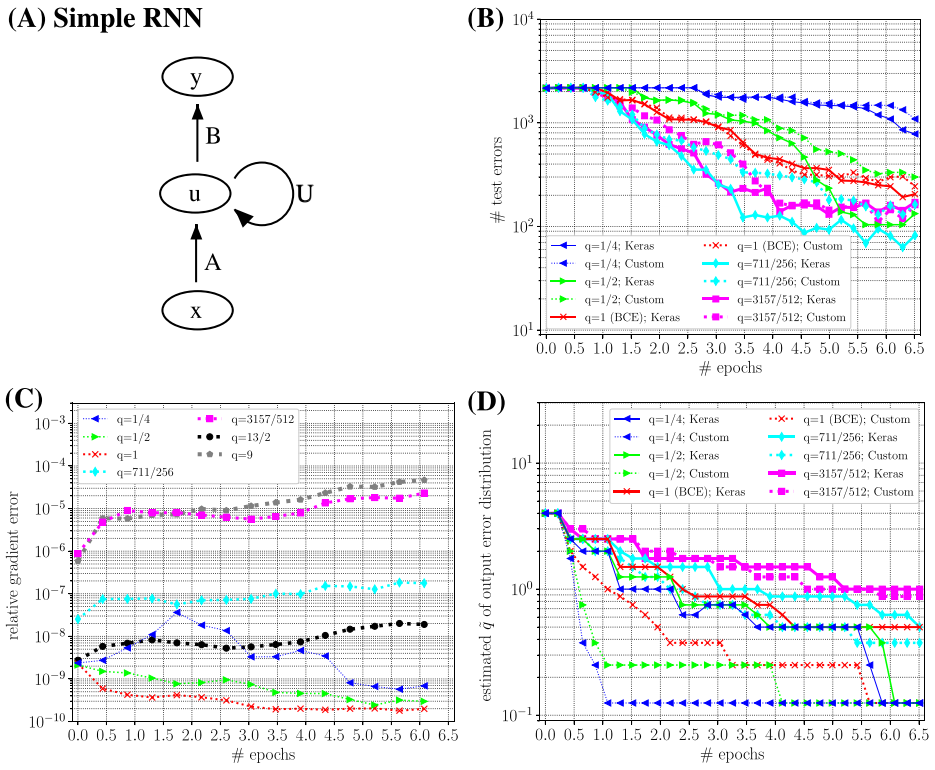

Fig. 2 Single trial learning experiments to verify the formulas for the generalized power error loss functions (10) and (30). **A**: Architecture of the Simple Recurrent Neural Network model. **B**: Test errors for the Embedded Reber Grammar data set as obtained from neural network implementations using either Keras (solid; automatic differentiation of (10) with (30)) or a custom neural network library (dotted; backpropagation algorithm using (5),(7)). **C**: Maximal relative error between gradients computed with backpropagation (as in (B)) and a naive estimation of partial derivatives from differential quotients (adding $\delta = 10^{-8}$ to each synaptic weight). **D**: Estimated power error exponent $\tilde{q}$ obtained from the (absolute) output error distributions of the experiments in (B) represented as histograms with 10 equally spaced bins. For each histogram, $\tilde{q}$ is estimated by selecting the best fitting theoretical histogram (minimal Euclidean distance) obtained from (33) for $q \in \{\frac{1}{8}, \ldots, \frac{7}{8}, 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 4, \ldots, 20\}$ (cf., Fig. 4). All experiments employ identical non-optimized standard hyperparameters (ADAM optimizer, $\eta_0 = 0.001$, minibatch size 8), identical initial synaptic weights (Glorot/Xavier uniform), and identical presentation order of training data

may be explained by steeper loss surfaces for $q \gg 1$ and increasing numerical errors due to increasing numbers of mutually canceling terms in (30).

Although hyperparameters have not yet been optimized, Fig. 2A shows the existence of an optimal error exponent somewhere between $q = 1$ and $q = 7$. In particular, learning for $q = 711/256 \approx 2.777$ and $q = 3157/512 \approx 6.166$ reaches an error count < 300 by factor 1.5-2 faster than for binary cross entropy loss ($q = 1$). This is consistent with previous results evaluating more complex network models involving LSTM layers and integer $q \in \mathbb{N}$ (see Fig.8, Fig. 9A in [19]).

Figure 3 shows corresponding results after optimizing the hyperparameters initial learning rate (η_0) and minibatch size (MBS), and averaging over 16 learning trials (by taking medians, similar as in previous works [19]). For all tested $q \in \{0.25, 0.5, \ldots, 2.75, 3, 3.5, \ldots, 9, 12, 15\}$ it was possible to reach zero average test error

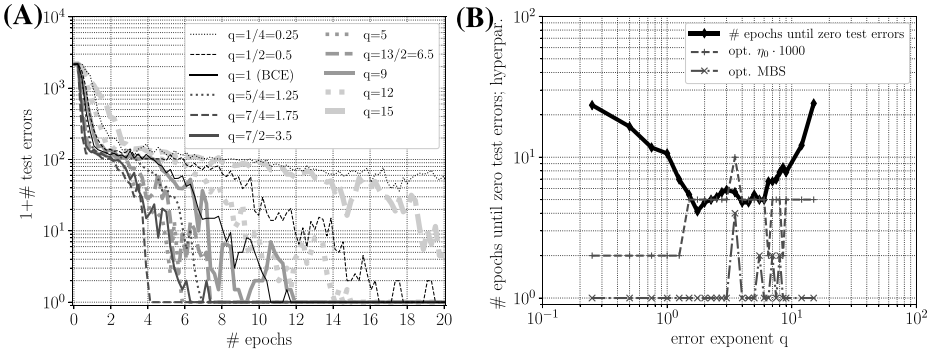

Fig. 3 Learning experiments of the Simple RNN (Keras implementation) applied to the Embedded Reber Grammar data set when optimizing hyperparameters η_0 and MBS and averaging over 16 trials (taking median error for each epoch; otherwise similar setting as in Fig. 2). **A**: Test errors for the Embedded Reber Grammar data set when learning for 20 epochs for different error exponents q as indicated by the legends. **B**: Minimal epoch number required to reach zero test error as function of q. The plot also shows optimal hyperparameters η_0 (initial learning rate) and MBS (minibatch size) from grid search with $\eta_0 \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.03, 0.05, 0.1\}$ and MBS $\in \{1, 2, 4, 8\}$. Curves in (A) correspond to (B)

(see also remarks in Appendix B). Therefore the first epoch number reaching zero average errors was used as a criterion for optimizing hyperparameters. Best $q = 1.75$ reached zero errors after 4.12 learning epochs, whereas $q = 1$ (BCE) required 10.63 epochs. Thus, optimizing the exponent q of the generalized power error loss function (10) yields factor > 2.5 improvement in learning time. This demonstrates that optimal exponents q may in general be non-integer. Still, there is a broad range of q between 1.25 and 9 where learning performance improves significantly compared to classical BCE. Note also that optimal hyperparameters are quite independent of q, mostly being $\eta_0 = 0.005$ and MBS$= 1$. This suggests that optimizing q may cause only little additional costs during hyperparameter optimization.

To understand the potential usefulness of the general case $q > 0$, let us reconsider a relationship found in [19] between the loss functions $\mathcal{L}_{nk}^{(q)}(y, t)$ from (10) and the corresponding distributions $r(\epsilon)$ of output errors $\epsilon := t - y \in (-1; 1)$ defined by (6): Specifically, $\mathcal{L}_{nk}^{(q)}(y, t)$ turns out to be optimal in maximizing the likelihood of the classification model if output errors are distributed with density function (see [19], eq. 5.7)

$$r(\epsilon) := p_0 r_0(\epsilon) + p_1 r_1(-\epsilon) = C_0 e^{-\mathcal{L}_{nk}^{(q)}(-\epsilon,0)} + C_1 e^{-\mathcal{L}_{nk}^{(q)}(\epsilon,0)} \tag{33}$$

where p_t are the prior class probabilities that an input belongs to class t, $r_t(\epsilon)$ are the conditional output error densities given t, and $C_t := p_t / \int_0^1 e^{-\mathcal{L}_{nk}^{(q)}(y,0)} dy$ are corresponding normalization constants. While [19] has computed $r(\epsilon)$ only for $q \in \mathbb{N}$, we can now use (30) to approximate $r(\epsilon)$ for any $q \in \mathbb{R}^+$ with arbitrary precision. Figure 4 shows the output error distributions $r(\epsilon)$ for some values of q. It can be seen that $q \gg 1$ corresponds to a uniform (rectangular) distribution, $q = 1$ to a linear (triangular) distribution, and $q < 1$ to distributions where most output errors are close to zero. This suggests two hypotheses about the relation between learning progress, error distributions, and an optimal choice for the exponent q of $\mathcal{L}_{nk}^{(q)}$. First, for any reasonable loss function, the exponent parameter $\tilde{q}$ best fitting the current error distribution should decrease with learning progress from large values $\tilde{q} > 1$ towards small values $\tilde{q} < 1$. This is confirmed by Fig. 2D: For all investigated

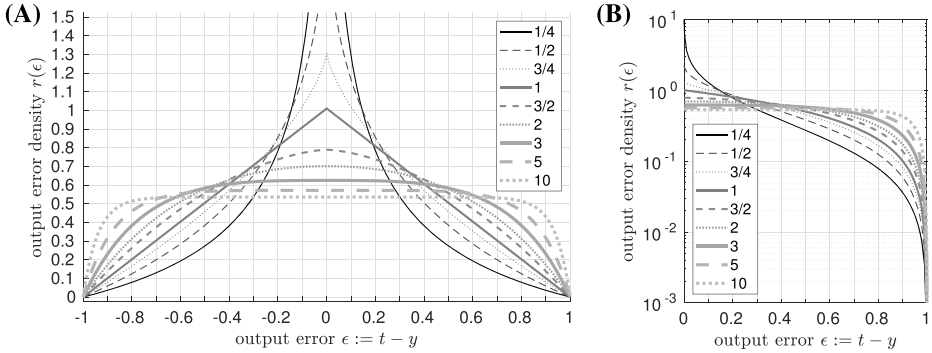

Fig. 4 **A**: Illustration of output error density $r(\epsilon)$ as computed from (33) for balanced class priors $p_0 = p_1 = 0.5$. Each curve corresponds to a specific power error exponent $q \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}, 2, 3, 5, 10\}$ as indicated by the legends. **B**: Same as (A), but logarithmic scale where, for symmetry, data is shown only for positive output errors $\epsilon > 0$

loss functions, the best fit $\tilde{q}$ decreases with training epochs. While $\tilde{q} = 4$ for initial synaptic weights, most error distributions have $\tilde{q} < 1$ after 5 learning epochs (see Appendix B for further details). Second, adapting the error exponent q of the loss function $\mathcal{L}_{nk}^{(q)}$ during learning to the distribution of output errors should improve learning performance.

Although a thorough investigation of the latter hypothesis is out of the scope of the current work, Fig. 5 shows results for a simplified setting employing a Convolutional Neural Network of moderate depth classifying the CIFAR-10 dataset after 15 training epochs (see Appendix B for details). In previous works, employing the power error loss function with fixed q in similar networks improved learning only marginally [19]. In the current experiments, the exponent parameter q of $\mathcal{L}_{nk}^{(q)}$ can be adapted once after 5 training epochs. For the control experiments with fixed q the results are in line with the previous findings: The case $q > 1$ improves learning to some degree, whereas $q < 1$ typically impairs learning performance. However, loss functions with adaptation, employing $q > 1$ in the early learning phase (epoch 1-5) and $q < 1$ in a later phase (epoch 6-15), can significantly improve accuracy (e.g., from 0.855 for fixed BCE or $q = 1$ to 0.863 for early $q = 60/8 = 7.5$ and late $q = 0.5$). By contrast, employing the reverse order (early $q < 1$ and late $q > 1$) impairs learning. This confirms the second hypothesis and shows that the case $q < 1$ can be useful if employed in a later training phase.

5 Summary and discussion

Motivated from classification applications with neural networks, this work gives closed-form expressions for the antiderivative $F(y)$ of the function $f(y) = \frac{y^{q-1}}{1-y}$ defined in (1), where the exponent $q = n/N$ should be rational with $n \in \mathbb{N}$ and N a power of 2. The most general and convenient form for $F(y)$ is given by (30) in Theorem 3. The special case for $q \in \mathbb{N}$ simplifies to (12) or (13) in Theorem 2, and has already been discussed in prior work [19]. Other intermediate representations involving complex roots and further special cases are given by Propositions 2-6. In principle, it would be possible to extend the range of exponents $q = n/N$ to more general forms with $N \in \mathbb{N}$ being an arbitrary integer, but

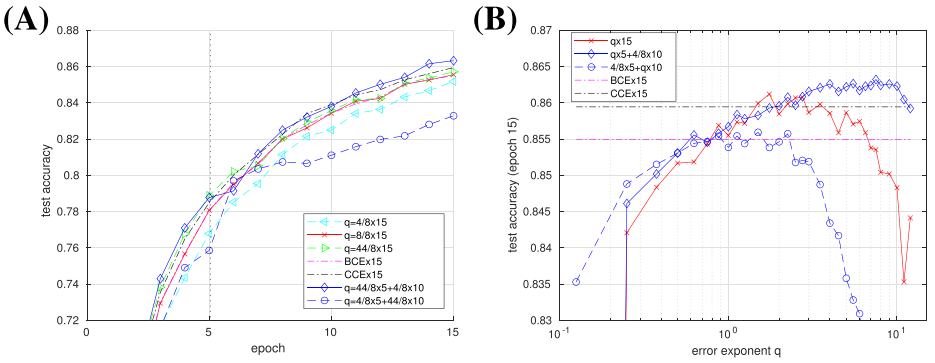

Fig. 5 Effect of adapting loss functions: **A**: Test accuracy as function of training epochs for fixed q ($q = 4/8$, $q = 8/8 = 1$, $q = 44/8$) and changing q after 5 epochs ($q = 44/8 \to 4/8 = 0.5$ and $q = 4/8 \to 44/8$) as indicated by the legends. Results for BCE and CCE are given for reference. **B**: Test accuracy after 15 training epochs as function of q for the cases fixed q (qx15), changing q at epoch 5 for the remaining epochs to 4/8 (qx5+4/8x10), and changing q in the reverse order (4/8x5+qx10). Note that starting with large $q > 1$ and then changing to small $q < 1$ (but not the reverse order) can significantly improve performance

this seems to lead to much more inconvenient formulas. As $q = n/N$ for $N = 2^m$ can approximate any rational or real-valued exponent with arbitrary precision, the current results seem sufficient for most applications.

Here I have considered a neural network application involving binary classification with logistic sigmoidal output units. For this network type, maximum-likelihood optimization is equivalent to minimizing the power error loss function (10) of Theorem 1 with the antiderivative $F(y)$ from Theorem 3. For that the exponent q can be related to the distribution of output errors (Fig. 4) and the initialization of error signals (7) for backpropagation learning [19]. Although knowing the correct loss function is actually not necessary for a custom gradient descent implementation based on error backpropagation with the power error initialization (7), modern neural network libraries like Keras, Tensorflow, and PyTorch [1, 6, 29] employ automatic differentiation [23] of the loss function to determine gradients for learning synaptic weights. Therefore Theorems 1 and 3 with the Python-based implementation of the power error loss function in Appendix B enable using such libraries for neural network learning with power error initialization.

For the special case of natural exponents $q \in \mathbb{N}$, this power error loss function has been derived and evaluated already in previous work [19]. There it has also been shown that optimizing q can significantly improve learning performance and convergence over various classical loss functions (like BCE, CCE, SSE), in particular for binary classification tasks in deep or recurrent networks. The current work extends these previous results for rational error exponents $q = n/2^m > 0$. Numerical and learning experiments have verified the correctness of Theorems 1 and 3 and the implementation of (10) and (30) in Python for Keras given in Appendix B. The experiments confirm that the usual outcome of optimizing error exponents is at least a moderate improvement of learning performance and convergence compared to cross entropy ($q = 1$), where optimal q is typically larger than one and not integer. Moreover, they show that adaptive loss functions decreasing q to values below 1 during learning may provide significant further improvements. A more thorough investigation of a continuous adaptation of q to the current distribution of output errors should be done in future work.

Appendix A: Proofs and supplements of Sections 2 and 3

Proof of Theorem 2: By iterated polynomial division it is easy to verify that for $n \in \mathbb{N}$

$$\frac{y^n}{y-1} = y^{n-1} + y^{n-2} + \ldots + y + 1 + \frac{1}{y-1} \tag{34}$$

and for $q \in \mathbb{N}$ and $y \in (0;1)$ therefore $F(y) := \int_0^y \frac{y^{q-1}}{1-y} dy = -\int_0^y \sum_{i=0}^{q-2} y^i + \frac{1}{1-y} dy = -\sum_{i=0}^{q-2} \frac{y^{i+1}}{i+1} - \log(1-y)$ showing (12). The second form (13) has been used previously [19] and is given here for completeness. The two forms are equivalent as (13) satisfies $F(0) = 0$ and, with the binomial sum, has the correct derivative $F'(y) = \frac{1}{1-y} + \sum_{r=0}^{q-2} \binom{q-1}{r}(-1)^{q-r}(-(1-y)^{q-2-r}) = \frac{1}{1-y} - \frac{(-1)^q}{1-y}(\sum_{r=0}^{q-1} \binom{q-1}{r}(-1)^r(1-y)^{q-1-r} - (-1)^{q-1}) = \frac{1-(-1)^q((-y)^{q-1}+(-1)^q)}{1-y} = \frac{y^{q-1}}{1-y}$. Then (14) follows from inserting (12) into (10),

$$\begin{aligned}\mathcal{L}_{nk}^{(q)}(y,t) &= (1-t)(-\log(1-y) - \sum_{i=1}^{q-1} \frac{y^i}{i}) + t(-\log(y) - \sum_{i=1}^{q-1} \frac{(1-y)^i}{i}) \\ &= -t\log(y) - (1-t)\log(1-y) - \sum_{i=1}^{q-1} \frac{(1-t)y^i + t(1-y)^i}{i}\,.\end{aligned}$$

With the binomial sum $(1-y)^i = \sum_{j=0}^{i} \binom{i}{j}(-y)^j$, the sum in (14) writes as the polynomial

$$\begin{aligned}a^{(q)}(y) &= \sum_{i=1}^{q-1} \frac{(1-t)y^i + t\sum_{j=0}^{i}\binom{i}{j}(-y)^j}{i} = \sum_{i=1}^{q-1} \frac{(1-t)y^i + t}{i} + \sum_{i=1}^{q-1}\sum_{j=1}^{i} \frac{t\binom{i}{j}(-y)^j}{i} \\ &= \sum_{i=1}^{q-1} \frac{t}{i} + \sum_{i=1}^{q-1} \frac{1-t}{i} y^i + \sum_{j=1}^{q-1}\sum_{i=j}^{q-1} \frac{t}{i}\binom{i}{j}(-y)^j \overset{!}{=} \sum_i a_i^{(q)} y^i\end{aligned}$$

from which we can read the polynomial coefficients $a_0^{(q)} = \sum_{i=1}^{q-1} \frac{t}{i}$ and

$$a_i^{(q)} = \frac{1-t}{i} + (-1)^i \sum_{j=i}^{q-1} \frac{t\binom{j}{i}}{j} = \frac{1-t}{i} + (-1)^i \sum_{j=i}^{q-1} \frac{t\binom{j-1}{i-1}}{i} = \frac{\left((-1)^i \sum_{j=i}^{q-1}\binom{j-1}{i-1} - 1\right)t + 1}{i}$$

for $i = 1, \ldots, q-1$. Table 1 gives examples for the alternative coefficients $b_i^{(q)}$ of (16). □

Proof of Proposition 1: It is well known that $\frac{d}{dz} {}_2F_1(a,b;a+1,z) = a \cdot \frac{(1-z)^{-b} - {}_2F_1(a,b;a+1,z)}{z}$. Defining $G(y) :=_2 F_1(q,1;q+1,y)$ for brevity we get from this $(y^q G(y)/q)' = y^{q-1}G(y) + y^q(q\frac{(1-y)^{-1}-G(y)}{y})/q = y^{q-1}/(1-y)$. □

Table 1 Coefficients $b_i^{(q)}$ to compute generalized loss functions $\mathcal{L}_{nk}^{(q)}(y,t)$ for $q \in \mathbb{N}$ from (16)

q	$b_0^{(q)}$	$b_1^{(q)}$	$b_2^{(q)}$	$b_3^{(q)}$	$b_4^{(q)}$	$b_5^{(q)}$	$b_6^{(q)}$	$b_7^{(q)}$	$b_8^{(q)}$	$b_9^{(q)}$	$b_{10}^{(q)}$	$b_{11}^{(q)}$
1	0											
2	1	2										
3	3/2	3	0									
4	11/6	4	2	2								
5	25/12	5	5	5	0							
6	137/60	6	9	11	4	2						
7	49/20	7	14	21	14	7	0					
8	363/140	8	20	36	34	22	6	2				
9	761/280	9	27	57	69	57	27	9	0			
10	7129/2520	10	35	85	125	127	83	37	8	2		
11	7381/2520	11	44	121	209	253	209	121	44	11	0	
12	83711/27720	12	54	166	329	463	461	331	164	56	10	2

For example, for $q = 1, \ldots, 5$ we have $\mathcal{L}_{nk}^{(1)}(y,t) = \mathcal{L}_{nk}^{\text{BCE}}(y,t) := -t \log y - (1-t)\log(1-y)$, $\mathcal{L}_{nk}^{(2)}(y,t) = \mathcal{L}_{nk}^{\text{BCE}}(y,t) - t + (2t-1)y$, $\mathcal{L}_{nk}^{(3)}(y,t) = \mathcal{L}_{nk}^{\text{BCE}}(y,t) - \frac{3}{2}t + (3t-1)y - \frac{1}{2}y^2$, $\mathcal{L}_{nk}^{(4)}(y,t) = \mathcal{L}_{nk}^{\text{BCE}}(y,t) - \frac{11}{6}t + (4t-1)y - \frac{2t+1}{2}y^2 + \frac{2t-1}{3}y^3$, $\mathcal{L}_{nk}^{(5)}(y,t) = \mathcal{L}_{nk}^{\text{BCE}}(y,t) - \frac{25}{12}t + (5t-1)y - \frac{5t+1}{2}y^2 + \frac{5t-1}{3}y^3 - \frac{1}{4}y^4$

Proof of Proposition 2: We have to show that $\tilde{F}'(y) = f(y)$. In fact, it is

$$\begin{aligned}
\tilde{F}'(y) &= -\sum_{Z:Z^N=1} Z^n \frac{d}{dy}\log(y^{1/N} - Z) \\
&= -\sum_{Z:Z^N=1} Z^n \frac{\frac{1}{N}y^{1/N-1}}{y^{1/N} - Z} = \frac{-1}{Ny^{(N-1)/N}} \sum_{Z:Z^N=1} \frac{1}{y^{1/N} - Z} \cdot Z^n \\
&= \frac{-1}{Ny^{(N-1)/N}} \sum_{k=0}^{N-1} \frac{1}{y^{1/N} - e^{j2\pi\frac{k}{N}}} \cdot e^{j2\pi\frac{kn}{N}} = \frac{-1}{Ny^{(N-1)/N}} \text{DFT}\{u[k]\}[N-n] \quad (35)
\end{aligned}$$

where the last equation involves the DFT $U[n] \bullet\!\!-\!\!\circ\, u[k]$ of the discrete N-periodic signal

$$\begin{aligned}
u[k] &:= \frac{1}{y^{1/N} - e^{j2\pi\frac{k}{N}}} = \frac{1}{y^{1/N}} \cdot \frac{1}{1 - y^{-1/N}e^{j2\pi\frac{k}{N}}} \\
&= c \cdot \frac{1 - e^{\alpha N}}{1 - e^{\alpha} \cdot e^{j2\pi\frac{k}{N}}} \qquad \text{with} \quad c := \frac{1}{y^{1/N}(1 - e^{\alpha N})} \quad \text{and} \quad \alpha := -\log(y)/N\,.
\end{aligned}$$

Using some well known facts of the DFT [36] for signals $v[k] \circ\!\!-\!\!\bullet\, V[n]$ and $w[k] \circ\!\!-\!\!\bullet\, W[n]$

$$\begin{aligned}
v[k] := e^{\alpha k} \circ\!\!-\!\!\bullet\, V[n] &:= \frac{1 - e^{\alpha N}}{1 - e^{\alpha} \cdot e^{-j2\pi\frac{n}{N}}} && \text{(DFT of exponential)} \\
cw[k] \circ\!\!-\!\!\bullet\, cW[n] & && \text{(linearity)} \\
w[-k] \circ\!\!-\!\!\bullet\, w[-n] & && \text{(reflected signal)} \\
W[k] \circ\!\!-\!\!\bullet\, N \cdot w[-n] & && \text{(duality of DFT)}
\end{aligned}$$

it follows $u[k] := c \cdot V[-k] \circ\!\!-\!\!\bullet\, c \cdot N \cdot v[n] = cN \cdot e^{\alpha n} =: U[n]$. Thus, inserting $U[N-n] = cNe^{\alpha(N-n)} = cNe^{-(N-n)\log(y)/N} = cNy^{-(N-n)/N}$ in (35) with $e^{\alpha N} = e^{-\log(y)} = 1/y$

yields

$$\begin{aligned}\tilde{F}'(y) &= \frac{-1}{Ny^{(N-1)/N}}U[N-n] = \frac{-cNy^{-(N-n)/N}}{Ny^{(N-1)/N}} = \frac{-y^{-(N-n)/N}}{y^{(N-1)/N}y^{1/N}(1-e^{\alpha N})} \\ &= \frac{-y^{-(1-n/N)}}{y(1-\frac{1}{y})} = \frac{y^{-(1-q)}}{1-y} = \frac{1}{(1-y)\cdot y^{1-q}} = f(y)\end{aligned}$$

proving (19), (20). We still have to prove (21) because determining the generalized loss function (10) with (17) involves subtracting $\tilde{F}(0)$: With the geometric-type sum

$$\sum_{k=0}^{N-1} kz^k = \frac{(N-1)z^{N+1} - Nz^N + z}{(z-1)^2}$$

(which can easily be proved by induction), the complex logarithm $\log(jr) \in (-\pi;\pi]$ in the primary sheet

$$\begin{aligned}\log(-e^{j2\pi\frac{k}{N}}) &= \log(e^{j\frac{2\pi}{N}k - j\pi}) = \log(e^{j\frac{2\pi}{N}(k-\frac{N}{2})}) = j\frac{2\pi}{N}(k-\frac{N}{2}) + j2\pi K \quad \text{for } K \in \mathbb{Z} \\ &= \begin{cases} j\frac{2\pi}{N}(k-\frac{N}{2}) & ,k = 1,2,\ldots,N-1 \\ j\frac{2\pi}{N}(k-\frac{N}{2}) + j2\pi & ,k = 0 \end{cases},\end{aligned}$$

using $z := e^{j2\pi n/N}$ and $\mathrm{DFT}\{1\}[n] = \delta[n] = 0$ for $n = 1,\ldots,N-1$, where $\delta[n]$ is the discrete Dirac impulse (that is, $\delta[n] = 0$ for $n = 0$, and $\delta[n] = 0$ for $n \neq 0$), we get from (20)

$$\begin{aligned}\tilde{F}(0) &= -\sum_{k=0}^{N-1} \log\left(-e^{j2\pi\frac{k}{N}}\right)\cdot e^{j2\pi\frac{kn}{N}} = -\sum_{k=0}^{N-1}(-j\pi + j2\pi\frac{k}{N})e^{j2\pi\frac{kn}{N}} - j2\pi \\ &= j\pi\,\mathrm{DFT}\{1\}[N-n] - j\frac{2\pi}{N}\sum_{k=0}^{N-1} kz^k - j2\pi = -j\frac{2\pi}{N}\cdot\frac{(N-1)z^{N+1} - Nz^N + z}{(z-1)^2} - j2\pi \\ &= -j\frac{2\pi}{N}\cdot\frac{(N-1)e^{j2\pi n(N+1)/N} - Ne^{j2\pi n} + e^{j2\pi n/N}}{(z-1)^2} - j2\pi \\ &= -j\frac{2\pi}{N}\cdot\frac{(N-1)e^{j2\pi n/N} - N + e^{j2\pi n/N}}{(z-1)^2} - j2\pi \\ &= -j\frac{2\pi}{N}\cdot\frac{Ne^{j2\pi n/N} - N}{(z-1)^2} - j2\pi = -j\frac{2\pi}{N}\frac{N(z-1)}{(z-1)^2} - j2\pi = \frac{2\pi j}{1-e^{j2\pi n/N}} \\ &= \frac{2\pi j}{1-\cos(2\pi\frac{n}{N}) - j\sin(2\pi\frac{n}{N})} - j2\pi = \frac{2\pi j(1-\cos(2\pi\frac{n}{N}) + j\sin(2\pi\frac{n}{N}))}{(1-\cos(2\pi\frac{n}{N}))^2 + \sin^2(2\pi\frac{n}{N})} - j2\pi \\ &= \frac{2\pi j(1-\cos(2\pi\frac{n}{N})) - 2\pi\sin(2\pi\frac{n}{N})}{1-2\cos(2\pi\frac{n}{N})+1} - j2\pi = \frac{-\pi\sin(2\pi\frac{n}{N})}{1-\cos(2\pi\frac{n}{N})} - j\pi\end{aligned}$$

□

Proof of Proposition 3: We can rewrite (20) as

$$\tilde{F}(y) = -\sum_{k=0}^{N-1}\tilde{F}[k] \quad \text{for} \quad \tilde{F}[k] := \log\left(y^{1/N} - e^{j2\pi\frac{k}{N}}\right)\cdot e^{j2\pi\frac{kn}{N}} \tag{36}$$

where $\tilde{F}[k]$ is a discrete N-periodic signal. In polar form,

$$y^{1/N} - e^{j2\pi\frac{k}{N}} = y^{1/N} - \cos\left(2\pi\frac{k}{N}\right) - j\sin\left(2\pi\frac{k}{N}\right) = r_k e^{j\varphi_k} \qquad \text{with}$$

$$r_k := \sqrt{\left(y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)\right)^2 + \sin^2\left(2\pi\frac{k}{N}\right)} = \sqrt{y^{2/N} - 2y^{1/N}\cos\left(2\pi\frac{k}{N}\right) + 1}$$

$$\varphi_k := \begin{cases} -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N}-\cos\left(2\pi\frac{k}{N}\right)}\right) & , \text{if } y^{1/N} \geq \cos\left(2\pi\frac{k}{N}\right) \\ -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N}-\cos\left(2\pi\frac{k}{N}\right)}\right) - \pi & , \text{if } y^{1/N} < \cos\left(2\pi\frac{k}{N}\right) \text{ and } \sin\left(2\pi\frac{k}{N}\right) \geq 0 \\ -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N}-\cos\left(2\pi\frac{k}{N}\right)}\right) + \pi & , \text{if } y^{1/N} < \cos\left(2\pi\frac{k}{N}\right) \text{ and } \sin\left(2\pi\frac{k}{N}\right) < 0 \end{cases} \quad (37)$$

where the last two cases are necessary to distinguish between different sheets of the complex logarithm, and thus

$$\begin{aligned} \tilde{F}[k] &:= \log\left(r_k e^{j\varphi_k}\right) \cdot e^{j2\pi\frac{kn}{N}} = (\log(r_k) + j\varphi_k) \cdot (\cos(2\pi\frac{kn}{N}) + j\sin(2\pi\frac{kn}{N})) \qquad (38) \\ &= \cos(2\pi\frac{kn}{N}) \cdot \log(r_k) - \sin(2\pi\frac{kn}{N}) \cdot \varphi_k + j\left(\cos(2\pi\frac{kn}{N}) \cdot \varphi_k + \sin(2\pi\frac{kn}{N}) \cdot \log(r_k)\right). \end{aligned}$$

As (36) obviously implies conjugate complex symmetry $\tilde{F}[-k] = \tilde{F}[N-k] = \tilde{F}^*[k]$, we get for $0 \leq y < 1$

$$\begin{aligned} \tilde{F}(y) &= -\sum_{k=0}^{N-1} \tilde{F}[k] = -\tilde{F}[0] - \tilde{F}[N/2] - \sum_{k=1}^{N/2-1} (\tilde{F}[k] + \tilde{F}[-k]) \\ &= -\log(y^{1/N} - 1) - (-1)^n \log(y^{1/N} + 1) - 2\sum_{k=1}^{N/2-1} \mathrm{Re}\{\tilde{F}[k]\} \\ &= -j\pi - \log(1 - y^{1/N}) - (-1)^n \log(1 + y^{1/N}) - 2\sum_{k=1}^{N/2-1} \cos(2\pi\frac{kn}{N}) \log(r_k) - \sin(2\pi\frac{kn}{N})\varphi_k \\ &= -j\pi + \log\left(\frac{(1 + y^{1/N})^{(-1)^{n+1}}}{1 - y^{1/N}}\right) - 2\tilde{G}(y) \qquad (39) \end{aligned}$$

where, in the sum $\tilde{G}(y) := \sum_{k=1}^{N/2-1} \cos(2\pi \frac{kn}{N}) \cdot \log(r_k) - \sin(2\pi \frac{kn}{N}) \cdot \varphi_k$, the terms with index $k = 1, ..., \frac{N}{4} - 1$ are similar to those with index $\frac{N}{2} - k$. Specifically,

$$
\begin{aligned}
&\cos(2\pi \frac{(\frac{N}{2}-k)n}{N}) = \cos(-(2\pi \frac{kn}{N} - \pi n)) = (-1)^n \cos(2\pi \frac{kn}{N}),\\
&-\sin(2\pi \frac{(\frac{N}{2}-k)n}{N}) = -\sin(-(2\pi \frac{kn}{N} - \pi n)) = (-1)^n \sin(2\pi \frac{kn}{N}),\\
&\cos(2\pi \frac{(\frac{N}{2}-k)}{N}) = -\cos(2\pi \frac{kn}{N}), \qquad \sin(2\pi \frac{(\frac{N}{2}-k)n}{N}) = \sin(2\pi \frac{kn}{N}),\\
&r_k = \sqrt{y^{2/N} - 2y^{1/N}\cos\left(2\pi \frac{k}{N}\right) + 1}, \qquad r_{N/2-k} = \sqrt{y^{2/N} + 2y^{1/N}\cos\left(2\pi \frac{k}{N}\right) + 1}\\
&\varphi_k = \begin{cases} -\arctan\left(\frac{\sin\left(2\pi \frac{k}{N}\right)}{y^{1/N} - \cos\left(2\pi \frac{k}{N}\right)}\right) & \text{, if } y^{1/N} \geq \cos\left(2\pi \frac{k}{N}\right)\\ -\arctan\left(\frac{\sin\left(2\pi \frac{k}{N}\right)}{y^{1/N} - \cos\left(2\pi \frac{k}{N}\right)}\right) - \pi & \text{, otherwise} \end{cases}\\
&\varphi_{N/2-k} = -\arctan\left(\frac{\sin\left(2\pi \frac{k}{N}\right)}{y^{1/N} + \cos\left(2\pi \frac{k}{N}\right)}\right), \qquad (40)
\end{aligned}
$$

where the last equation for $\varphi_{N/2-k}$ follows from $y \geq 0$ and $\cos\left(2\pi \frac{N/2-k}{N}\right) < 0$ for $k = 1, \ldots, \frac{N}{4} - 1$. Thus for $N \geq 4$

$$
\begin{aligned}
\tilde{G}(y) \;=\; &\cos(\pi \frac{n}{2}) \cdot \log(r_{N/4}) - \sin(\pi \frac{n}{2}) \cdot \varphi_{N/4}\\
&+ \sum_{k=1}^{N/4-1} \cos(2\pi \frac{kn}{N}) \cdot \log(r_k) - \sin(2\pi \frac{kn}{N}) \cdot \varphi_k\\
&+(-1)^n \cos(2\pi \frac{kn}{N}) \log(r_{N/2-k}) + (-1)^n \sin(2\pi \frac{kn}{N}) \varphi_{N/2-k}\\
=\; &\frac{\cos(\pi \frac{n}{2}) \cdot \log(y^{2/N}+1)}{2} + \sin(\pi \frac{n}{2}) \cdot \arctan(y^{-1/N})\\
&+ \sum_{k=1}^{N/4-1} \cos(2\pi \frac{kn}{N}) \cdot \log\left(r_k \cdot r_{N/2-k}^{(-1)^n}\right) - \sin(2\pi \frac{kn}{N}) \cdot \left(\varphi_k - (-1)^n \varphi_{N/2-k}\right) \quad (41)
\end{aligned}
$$

such that inserting (41) in (39) and using (21) we obtain (22). The remaining special case (23) is easily shown by the derivative $\left(\log \frac{1+\sqrt{y}}{1-\sqrt{y}}\right)' = \frac{1-\sqrt{y}}{1+\sqrt{y}} \cdot \frac{\frac{1}{2\sqrt{y}}(1-\sqrt{y}) + (1+\sqrt{y})\frac{1}{2\sqrt{y}}}{(1-\sqrt{y})^2} = \frac{1}{(1-y)\sqrt{y}}$. □

***Proof of Proposition 4**:* For irreducible $q = n/N = n/2^m \in (0;1)$, we have odd n for $m \geq 1$ and $(-1)^n = -1$. From (40) we then obtain for $k = 1, \ldots, \frac{N}{4} - 1$

$$\log\left(r_k \cdot r_{N/2-k}^{(-1)^n}\right) = \log\left(\frac{r_k}{r_{N/2-k}}\right) = \frac{1}{2}\log\left(\frac{y^{2/N} - 2y^{1/N}\cos\left(2\pi\frac{k}{N}\right) + 1}{y^{2/N} + 2y^{1/N}\cos\left(2\pi\frac{k}{N}\right) + 1}\right) \quad (42)$$

$$\varphi_k - (-1)^n\varphi_{N/2-k} = \varphi_k + \varphi_{N/2-k} = -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)}\right)$$
$$-\pi \cdot \left(1 - H\left(y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)\right)\right) - \arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} + \cos\left(2\pi\frac{k}{N}\right)}\right) \quad (43)$$

with the Heaviside function $H(y) = 1$ if $y \geq 0$ and $H(y) = 0$ otherwise. We can further simplify the last equation using the addition theorem of the arctan function

$$\arctan(a) + \arctan(b) = \begin{cases} \arctan\left(\frac{a+b}{1-ab}\right) & , ab < 1 \\ \pi + \arctan\left(\frac{a+b}{1-ab}\right) & , ab > 1 \text{ and } a+b \geq 0 \\ -\pi + \arctan\left(\frac{a+b}{1-ab}\right) & , ab > 1 \text{ and } a+b < 0 \end{cases}.$$

Specifically, for $a := \frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)}$ and $b := \frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} + \cos\left(2\pi\frac{k}{N}\right)}$ we get

$$ab = \frac{\sin^2\left(2\pi\frac{k}{N}\right)}{y^{2/N} - \cos^2\left(2\pi\frac{k}{N}\right)}, \quad \text{thus } ab < 1 \Leftrightarrow \begin{cases} \sin^2(.) + \cos^2(.) = 1 < y^{2/N} & , y^{2/N} > \cos^2(.) \\ \sin^2(.) + \cos^2(.) = 1 > y^{2/N} & , y^{2/N} < \cos^2(.) \end{cases},$$

$$a + b = \frac{\sin\left(\frac{2\pi k}{N}\right)\left(y^{\frac{1}{N}} + \cos\left(\frac{2\pi k}{N}\right)\right) + \sin\left(\frac{2\pi k}{N}\right)\left(y^{\frac{1}{N}} - \cos\left(\frac{2\pi k}{N}\right)\right)}{y^{\frac{2}{N}} - \cos^2\left(\frac{2\pi k}{N}\right)} = \frac{2\sin\left(\frac{2\pi k}{N}\right)y^{\frac{1}{N}}}{y^{\frac{2}{N}} - \cos^2\left(\frac{2\pi k}{N}\right)},$$

$$\arctan\left(\frac{a+b}{1-ab}\right) = \arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{y^{2/N} - 1}\right).$$

Note that for $k \in \{1, \ldots, \frac{N}{4} - 1\}$ both $\sin\left(2\pi\frac{k}{N}\right) > 0$ and $\cos\left(2\pi\frac{k}{N}\right) > 0$. Thus, for $0 < y < 1$ the conditions $ab < 1$ and $a + b < 0$ are both equivalent to $y^{1/N} < \cos\left(2\pi\frac{k}{N}\right)$. Therefore the addition theorem implies

$$\arctan(a) + \arctan(b) = \arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{y^{2/N} - 1}\right) + \pi \cdot H\left(y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)\right)$$

and (43) gets $\varphi_k - (-1)^n\varphi_{\frac{N}{2}-k} = \varphi_k + \varphi_{\frac{N}{2}-k} = -\pi - \arctan\left(\frac{2\sin\left(\frac{2\pi k}{N}\right)y^{\frac{1}{N}}}{y^{\frac{2}{N}} - 1}\right)$ (44)

such that with (42), the identity $\arctan(\frac{1}{x}) = \frac{\pi}{2} - \arctan(x)$, and $(-1)^n = -1$ the antiderivative (22) becomes for odd n with $\cos(\pi\frac{n}{2}) = 0$ and $\sin(\pi\frac{n}{2}) = (-1)^{(n-1)/2}$

$$F(y) = \frac{\pi \sin(\frac{2\pi n}{N})}{1-\cos(\frac{2\pi n}{N})} + \log\left(\frac{1+y^{\frac{1}{N}}}{1-y^{\frac{1}{N}}}\right) - 2\cdot(-1)^{\frac{n-1}{2}}\cdot(\frac{\pi}{2} - \arctan(y^{\frac{1}{N}})) - \sum_{k=1}^{\frac{N}{4}-1}\cos(\frac{2\pi kn}{N})$$
$$\cdot\log\left(\frac{y^{\frac{2}{N}} - 2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+1}{y^{\frac{2}{N}} + 2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+1}\right) - \sum_{k=1}^{\frac{N}{4}-1} 2\sin(\frac{2\pi kn}{N})\cdot\left(\pi + \arctan\left(\frac{2\sin\left(\frac{2\pi k}{N}\right)y^{\frac{1}{N}}}{y^{\frac{2}{N}}-1}\right)\right). \quad (45)$$

Simplifying this result yields Proposition 4: As $\arctan(0) = 0$, $\log(1) = 0$, and $F(y) = 0$ by definition (see (22)), the trigonometric identity (24) follows from (45) for $y = 0$. Inserting (24) back into (45) yields (25). □

Proof of Proposition 5: Recapitulating our results so far, we find that Proposition 2 (including all equations in the proof) and (36)–(38) hold also for $y > 1$. With this we can easily verify that for $y > 1$ we have real-valued $\log(y^{1/N} - 1) \in \mathbb{R}$ and the antiderivative (39) becomes

$$\tilde{F}(y) = \log\left(\frac{(1+y^{1/N})^{(-1)^{n+1}}}{y^{1/N}-1}\right) - 2\tilde{G}(y) \quad (46)$$

Then, because of $y > 1 \geq \cos(2\pi\frac{k}{n})$, we see that (40) simplifies to

$$\varphi_k = -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} - \cos\left(2\pi\frac{k}{N}\right)}\right) \quad \text{and} \quad \varphi_{N/2-k} = -\arctan\left(\frac{\sin\left(2\pi\frac{k}{N}\right)}{y^{1/N} + \cos\left(2\pi\frac{k}{N}\right)}\right),$$

whereas all equations of (40) and (41) still hold true. Thus, in correspondence to (22) we get for $y > 1$ and any $n = 1, 2, 3, 4, \ldots, N-1$

$$\tilde{F}(y) = \log\left(\frac{(1+y^{1/N})^{(-1)^{n+1}}}{y^{1/N}-1}\right) - \cos(\pi\frac{n}{2})\log(y^{2/N}+1) - 2\sin(\pi\frac{n}{2})\arctan(y^{-1/N})$$
$$-2\sum_{k=1}^{N/4-1}\cos(2\pi\frac{kn}{N})\log\left(r_k\cdot r_{N/2-k}^{(-1)^n}\right) - \sin(2\pi\frac{kn}{N})\left(\varphi_k - (-1)^n\varphi_{N/2-k}\right) \quad (47)$$

employing real-valued computations only (whereas (22) is still a valid antiderivative for $y > 1$, but includes imaginary numbers). For *irreducible* $q = n/2^m$ with odd $n = 1, 3, 5, \ldots, N-1$, it is easy to verify that (42) and (43) remain valid, whereas $y^{1/N} - \cos\left(2\pi\frac{k}{N}\right) > 0$ for $y > 1$ implies always $ab < 1$ and therefore (44) simplifies to

$$\varphi_k - (-1)^n\varphi_{N/2-k} = \varphi_k + \varphi_{N/2-k} = -\arctan\left(\frac{2\sin\left(2\pi\frac{k}{N}\right)y^{1/N}}{y^{2/N}-1}\right). \quad (48)$$

Thus, (47) becomes for odd n with (42), (44), $\arctan(\frac{1}{x}) = \frac{\pi}{2} - \arctan(x)$, $(-1)^n = -1$, $\cos(\pi\frac{n}{2}) = 0$, and $\sin(\pi\frac{n}{2}) = (-1)^{(n-1)/2}$

$$\tilde{F}(y) = \log\left(\frac{(1+y^{\frac{1}{N}})^{(-1)^{n+1}}}{y^{\frac{1}{N}}-1}\right) - 2\cdot(-1)^{\frac{n-1}{2}}\cdot(\frac{\pi}{2} - \arctan(y^{\frac{1}{N}})) - \sum_{k=1}^{\frac{N}{4}-1}\cos(\frac{2\pi kn}{N})$$
$$\cdot\log\left(\frac{y^{\frac{2}{N}} - 2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+1}{y^{\frac{2}{N}} + 2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+1}\right) + \sum_{k=1}^{\frac{N}{4}-1} 2\cdot\sin(\frac{2\pi kn}{N})\cdot\left(-\arctan\left(\frac{2\sin\left(\frac{2\pi k}{N}\right)y^{\frac{1}{N}}}{y^{\frac{2}{N}}-1}\right)\right)$$
$$= \log\left(\frac{1+y^{\frac{1}{N}}}{y^{\frac{1}{N}}-1}\right) + (-1)^{\frac{n-1}{2}}\cdot 2\arctan(y^{\frac{1}{N}}) + \sum_{k=1}^{\frac{N}{4}-1}\cos(\frac{2\pi kn}{N})$$
$$\cdot\log\left(\frac{1+2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+y^{\frac{2}{N}}}{1-2y^{\frac{1}{N}}\cos\left(\frac{2\pi k}{N}\right)+y^{\frac{2}{N}}}\right) - \sum_{k=1}^{\frac{N}{4}-1} 2\cdot\sin(\frac{2\pi kn}{N})\cdot\arctan\left(\frac{2\sin\left(\frac{2\pi k}{N}\right)y^{\frac{1}{N}}}{y^{\frac{2}{N}}-1}\right) + C \quad (49)$$

where with the identity (24) the constant is $C = \pi\cdot(-1)^{(n+1)/2}$. (50)

Merging this with Proposition 4 gives almost immediately Proposition 5: Skipping the constant C in (49) and then comparing to (25) reveals that, after taking absolute values $|1-y^{1/N}|$, both (25) for $0 \le y < 1$ and (49) for $y > 1$ represent the same function that is unified by (26). The case $N = 2$ corresponding to (27) can be shown as in (23).

□

***Proof of Proposition 6**:* We assume *irreducible* $q = n/N > 1$ with $N := 2^m > 1$ and odd $n > N$. Using odd $\eta := n - N > 0$ and substituting $u := y^{1/N}$ with $du/dy = \frac{y^{1/N-1}}{N} = \frac{1}{Ny^{(N-1)/N}}$, $y = u^N$, $dy = Ny^{(N-1)/N}du = Nu^{N-1}du$ we obtain

$$\int\frac{y^{q-1}}{1-y}dy = \int\frac{y^{(n-N)/N}}{1-y}dy = \int\frac{y^{\eta/N}}{1-y}dy = \int\frac{u^\eta}{1-u^N}Nu^{N-1}du = N\int\frac{u^{N+\eta-1}}{1-u^N}du \quad (51)$$

By polynomial division we get for $M := (\eta-1)\text{ div }N \ge 0$ and even $R := (\eta-1)\bmod N = \eta - 1 - M\cdot N \in \{0,\ldots,N-2\}$

$$\frac{u^{N+\eta-1}}{u^N-1} = u^{\eta-1} + \frac{u^{\eta-1}}{u^N-1} = \ldots = \sum_{i=0}^{M}u^{\eta-1-i\cdot N} + \frac{u^R}{u^N-1} \quad (52)$$

and therefore, with re-substituting $u := y^{1/N}$, $\eta := n - N$, and $du = \frac{dy}{Ny^{(N-1)/N}}$, (51) becomes

$$\int\frac{y^{q-1}}{1-y}dy = -N\int\frac{u^{N+\eta-1}}{u^N-1}du = -N\sum_{i=0}^{M}\int u^{\eta-1-i\cdot N}du - N\int\frac{u^R}{u^N-1}du$$
$$= -N\sum_{i=0}^{M}\frac{u^{\eta-i\cdot N}}{\eta-i\cdot N} + N\int\frac{u^R}{1-u^N}du = -N\sum_{i=0}^{M}\frac{y^{\eta/N-i}}{\eta-i\cdot N} + N\int\frac{y^{R/N}}{1-y}\cdot\frac{dy}{Ny^{(N-1)/N}}$$
$$= -N\sum_{i=0}^{M}\frac{y^{(\eta\text{ div }N)+(\eta\bmod N)-i}}{n-N-i\cdot N} + \int\frac{1}{(1-y)\cdot y^{(N-1-R)/N}}dy$$
$$= -N\sum_{i=0}^{M}\frac{y^{M-i+(R+1)/N}}{n-(i+1)\cdot N} + \int\frac{1}{(1-y)\cdot y^{1-(R+1)/N}}dy \quad (53)$$

where (53) uses $\eta \operatorname{div} N = (\eta - 1) \operatorname{div} N$ and $\eta \bmod N = ((\eta - 1) \bmod N) + 1 = R + 1$ as $\eta := n - N$ is odd while $N := 2^m$ is even, and so $\eta \bmod N \in \{1, 3, 5, \ldots, N-1\}$ is odd. Using (53) completes the proof: As $\tilde{n} := R + 1 = \eta \bmod N = n \bmod N \in \{1, 3, 5, \ldots, N-1\}$ is odd, $\tilde{q} := (R+1)/N = \tilde{n}/N$ is irreducible and $0 < \tilde{q} < 1$, thus the remaining integral in (53) follows from (26) for $N \geq 4$ or from (27) for $N = 2$, replacing q and n by $\tilde{q}$ and $\tilde{n}$, respectively. Thus we get (28) for $N \geq 4$ and (29) for $N = 2$. Both antiderivatives have already the correct offset 0 for $y = 0$, so they correspond to continuations of $F(y)$ as defined earlier. □

***Proof of Theorem 3**:* First, let us show that (30) contains Proposition 6 (as special case for irreducible $q = n/N > 1$ with $N \geq 2$ and $y \in \mathbb{R}_0^+ \backslash \{1\}$): Indeed, for $M := (n - N - 1) \operatorname{div} N = (n \operatorname{div} N) - 1 = \tilde{q} - 1$, the first sums in (6) and (29) become after substituting i by $M - i$

$$-Ny^{\tilde{n}/N} \cdot \sum_{i=0}^{M} \frac{y^{M-i}}{n - (i+1) \cdot N} = -Ny^{\tilde{n}/N} \cdot \sum_{i=0}^{\tilde{q}-1} \frac{y^i}{n-(M-i+1)\cdot N} ,$$

which, for the case $N \geq 2$ with $H(N-2) = H(q-1) = 1$, equals the first line of (30) writing the first summand for $i = 0$ as a separate term for later case distinctions. It is easy to verify that also the remaining terms in (6) and (29) are equivalent to those in (30) as here $H(N-2)$ and $H(N-4)$ select the necessary terms, respectively.

Second, we show that (30) contains Theorem 2 (as special case for $q \in \mathbb{N}$, $0 \leq y < 1$) and extends it for $y > 1$: As here $q = n/N = n \in \mathbb{N}$ with $N = 1$, and thus $\tilde{n} = 0$, $\tilde{q} = n = q$, $M = q - 1$, and $H(N-2) = H(N-4) = 0$, $H(q-1) = 1$, the antiderivative (30) becomes

$$\begin{aligned} F(y) &= -Ny^{\tilde{n}/N} \cdot \left(\frac{H(N-2)}{n - \tilde{q} \cdot N} + \sum_{i=1}^{\tilde{q}-1} \frac{y^i}{n - (\tilde{q} - i) \cdot N} \right) + \log \left(\frac{1 + y^{1/N} \cdot H(N-2)}{|1 - y^{1/N}|} \right) \\ &= -\sum_{i=1}^{q-1} \frac{y^i}{i} - \log(|1-y|) , \end{aligned}$$

which equals (12) for $0 \leq y < 1$ and, also for the case $y > 1$, is still a proper antiderivative of $f(y)$, as can easily be seen by recapitulating the proof of (12) using (34).

Third, it is easy to verify that (30) contains Proposition 5 (as special case for irreducible $0 < q = n/N < 1$ with $N \geq 2$ and $y \in \mathbb{R}_0^+ \backslash \{1\}$) using $H(q-1) = 0$, $H(N-2) = 1$, and either $H(N-4) = 1$ (for (26)) or $H(N-4) = 0$ (for (27)).

Finally, the monotonicity of $F(y)$ follows from $f(y) > 0$ for $0 < y < 1$ and $f(y) < 0$ for $y > 1$, and the limits and asymptotic expressions (31) are easily verified by inspecting each case. In particular, $F(0) = 0$ follows from $0^i = 0$, $\log(1) = 0$, and $\arctan(0) = 0$. $F(1) = \infty$ follows from $\lim_{y \to 1} \log(\frac{1+y^{\frac{1}{N}} \cdot H(N-2)}{|1-y^{1/N}|}) = \infty$ and all other terms remaining finite for $y \to 1$. For $y \to \infty$, approximation (31) follows because $\arctan(\frac{2\sin(\frac{2\pi k}{N}) y^{\frac{1}{N}}}{1 - y^{\frac{2}{N}}}) \to 0$, $\log(\frac{1+2y^{\frac{1}{N}} \cos(\frac{2\pi k}{N}) + y^{\frac{2}{N}}}{1-2y^{\frac{1}{N}} \cos(\frac{2\pi k}{N}) + y^{\frac{2}{N}}}) \to 0$, $\arctan(y^{\frac{1}{N}}) \to \frac{\pi}{2}$, and $\log(\frac{1+y^{\frac{1}{N}} \cdot H(N-2)}{|1-y^{\frac{1}{N}}|}) \to 0$ for $N \geq 2$. From this the asymptotics can easily be seen, in particular for $q > 1$ the dominating

term in (31) is $-\frac{N}{n-N} y^{\tilde{n}/N+\tilde{q}-1} = -\frac{1}{(n-N)/N} y^{(n \bmod N)/N+n \operatorname{div} N-1} = -\frac{1}{q-1} y^{(n-N)/N} = -\frac{1}{q-1} y^{q-1}$. □

Appendix B: Implementation details

Figures 2–3 used the **Simple RNN** of Fig. 2A with D inputs, $M = 10$ hidden units, and K outputs. The layers are linked by dense connections A, U, B and include also bias weights for layers u, y. Activation functions are tanh for u and the logistic sigmoid σ for y. Synaptic connections are initialized by uniform Xavier [10]. Training used ADAM optimizer [18, 31] with standard parameters $\beta_1 = 0.9, \beta_2 = 0.999$. Experiments used either **Keras** 2.2.5 with a Tensorflow 1.14.0 backend [1, 6] or a **custom neural network library** for backpropagation. While Keras computes gradients based on automatic differentiation [23] of the loss function (see Python code below for the power error loss (10) with (30)), the custom implementation uses (5) with the power error initialization (7).

The **Embedded Reber Grammar Problem** is to predict the next output symbol of a finite automaton with non-deterministic state transitions [15]. Inputs are symbol sequences $\mathbf{x}_n(1), \mathbf{x}_n(2), \ldots, \mathbf{x}_T$ generated by the automaton, representing each of the $D = 7$ symbols by a one-hot input vector $\mathbf{x}_n(\tau)$. The output to be predicted at time τ is the next symbol $\mathbf{x}_n(\tau + 1)$ generated by the automaton in the next time step ($K = 7$). Due to the non-determinism, target vectors $\mathbf{t}_n(\tau)$ can have multiple one-entries, one for each possible output symbol. Network decision $\hat{\mathbf{y}}_n(\tau)$ is evaluated as correct if $\hat{\mathbf{y}}_n(\tau) = \mathbf{t}_n(\tau)$ at decision threshold 0.5. Learning used $N = 2048$ sequences (90% training, 10% validating/testing). Average sequence length is $T = 12$ (maximum 40).

Remarks to Fig. 3: Reaching zero *median* errors suggests that *all* 16 learning trials of an experiment converged to zero error, while previous works reported problems of simple RNN solving this data set [9, 15]. However, as zero median means that at least half of trials reached zero errors, I have analyzed also *mean* test errors (instead of medians), which also reached zero within the total learning duration of 65 epochs for all exponents $0.5 \leq q \leq 15$ (data not shown). The profile of minimal epoch numbers until zero test errors was similar to Fig. 3B, although absolute values were about factor 2-2.5 larger (best was 9 epochs for $q = 1.5$ vs. 26 epochs for $q = 1$). The discrepancy may be that earlier works used a suboptimal initialization of synaptic weights, causing either vanishing or exploding gradients [3, 15]. Additional experiments (data not shown) revealed that at least $q > 1$ can still reach zero error, even if initial weights deviate substantially (by factors between 0.125 and 3) from Xavier initialization. This is consistent with the idea that, for $q > 1$, the power error loss provides a better gradient-to-loss ratio and, thereby, avoids flat loss landscapes and vanishing gradients [19].

Remarks to Fig. 2D: Panel (D) shows estimated power exponents $\tilde{q}$ of absolute output error distributions $|\epsilon| = |t - y|$ corresponding to panels (B) and (C). Estimated $\tilde{q}$ are obtained from selecting the minimum Euclidean distance of histogram vectors (10 equally spaced bins of interval $|\epsilon| \in [0; 1]$) between experimental and theoretical distributions (33) evaluated for $q \in \{\frac{1}{8}, \ldots, \frac{7}{8}, 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 4, \ldots, 20\}$. As expected, $\tilde{q}$ decreases gradually with epoch number, and may therefore be used to characterize learning progress as discussed in Section 4. Again, there are some deviations between Keras and the custom implementation, in particular, in later learning phases (but note the logarithmic scale).

Figure 5 used a sequential 2D **Convolutional Neural Network** (CNN) architecture implemented with PyTorch 1.7.0+cu101 [29], including 6 CNN-Layers (kernel size 3, stride

1, "same" padding), 2 Max-Pooling layers (MaxPool; kernel size 2; stride 2), 2 Batch-Normalization (BN) layers, 2 Fully Connected (FC) layers, and 2 Dropout-Layers: Input $\rightarrow$ CNN(32) $\rightarrow$ BN(32) $\rightarrow$ ReLU $\rightarrow$ CNN(64) $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ CNN(128) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ CNN(128) $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ Dropout($p = 0.05$) $\rightarrow$ CNN(256) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ CNN(256) $\rightarrow$ ReLU $\rightarrow$ MaxPool $\rightarrow$ Dropout($p = 0.1$) $\rightarrow$ FC(1024) $\rightarrow$ ReLU $\rightarrow$ FC(512) $\rightarrow$ Dropout($p = 0.1$) $\rightarrow$ FC(512) $\rightarrow \sigma$. Numbers in brackets correspond to channels (CNN/BN) and size (FC). Activation functions were rectified linear units (ReLU) except σ for outputs (or softmax for CCE loss). The **CIFAR10** dataset consists of 50000 training and 10000 test images (RGB) of size 32×32 from 10 classes [20]. Accuracies were computed from maximum decisions averaged (mean value) over 16 learning trials. Standard parameters were used without any further optimizations (ADAM optimizer as above, but initial learning rate 0.0001, minibatch 64).

Backpropagation involving automatic differentiation (as with Keras and PyTorch) requires **implementing the power error loss** function (10) with (30). All experiments involving Keras used the following Python code (for PyTorch replace `K` and `tf` by `torch` and `clip_by_value` by `clamp`):

```
import numpy as np, tensorflow as tf
from keras import backend as K

def powererrorloss_wrapper(n=3,N=1,_eps=1e-7):
    """
    computes power error loss function eq.10 for q=n/N using eq.30
    it is assumed that n and N are positive integers with N=2^m being a power of two
    _eps: clip y and powers of y to interval [_eps;1-_eps] to avoid NaN etc.
    """

    def F(y):   # eq.30: assumes that n is odd, N=2^m is a power of two for m>=0, and 0<y<1
        y=tf.clip_by_value(y,_eps,1.0-_eps)
        n_tilde = n%N
        q_tilde = n//N
        y_1divN = tf.clip_by_value(K.pow(y,1.0/N),_eps,1.0-_eps)
        y_2divN = tf.clip_by_value(K.pow(y,2.0/N),_eps,1.0-_eps)
        y_ntildedivN = tf.clip_by_value(K.pow(y,n_tilde/N),_eps,1.0-_eps)
        # (i) first line of F(y) in eq.30
        L = 0
        if n>=N:     # if q>=1
           if N>=2: L=L+1.0/(n-q_tilde*N)
           for i in range(1,q_tilde): L=L+K.pow(y,i)*(1.0/(n-(q_tilde-i)*N))
           L=L*(-N*y_ntildedivN)
        # (ii) second line of eq.30
        tmp=1.0/tf.clip_by_value(1.0-y_1divN,_eps,1.0-_eps) # abs(.) not necessary (clipping)
        if N>=2: tmp=tmp*(1.0+y_1divN);
        L=L+K.log(tmp);
        # (iii) third line of eq.30
        if N>=4:
           tmp=(n_tilde-1)//2             # exponent of (-1)^((n_tilde-1)/2)
           if ((tmp//2)*2) == tmp: sg=1   # sign of (-1)^((n_tilde-1)/2)
           else: sg=-1
           L=L+sg*2.0*tf.atan(y_1divN)
        # (iv) fourth line of eq.30
        if N>=4:
           for k in range(1,N//4):
              tmp=tf.clip_by_value(1.0+2.0*y_1divN*np.cos(2.0*np.pi*k/N)+y_2divN,_eps,4.)
              tmp=tmp/tf.clip_by_value(1.0-2.0*y_1divN*np.cos(2.0*np.pi*k/N)+y_2divN,_eps,4.)
              L=L+np.cos(2.0*np.pi*k*n_tilde/N)*K.log(tmp)
        # (v) fifth line of eq.30
           for k in range(1,N//4):
              tmp=2.0*np.sin(2.0*np.pi*k/N)*y_1divN
              tmp=tf.atan(tmp/tf.clip_by_value(1.0-y_2divN,_eps,1.0-_eps))
              L=L+2.0*np.sin(2.0*np.pi*k*n_tilde/N)*tmp
        return L
```

```
def powererrorloss(t,y):
    loss=(1.0-t)*F(y)+t*F(1.0-y) # eq.10
    return K.mean(loss)          # averaging to be consistent with other Keras losses

# main program of powererrorloss_wrapper
assert n>0 and isinstance(n,int),'n must be positive integer!'
assert N>0 and isinstance(N,int),'N must be positive integer!'
m,N_=0,N
while N_>1:                      # check if N is power of 2
    N_old=N_
    N_=N_//2
    assert N_*2==N_old,'N must be power of two!'
    m=m+1
while ((n//2)*2==n) and (N>1):   # reduce n/N
    n=n//2
    N=N//2
return powererrorloss
```

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of Interests The author declares that he has no conflict of interest.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al: Tensorflow: A system for large-scale machine learning. In: 12Th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283 (2016)
2. AlAhmad, R., Almefleh, H.: Antiderivatives and integrals involving incomplete beta functions with applications. Aust. J. Math. Anal. Appl. **17**(2), 11 (2020)
3. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
4. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
5. Bryson, A.E., Ho, Y.C.: Applied Optimal Control: Optimization, Estimation, and Control. Blaisdell, New York (1969)
6. Chollet, F.: Keras. https://github.com/fchollet/keras (2015)
7. Cover, T., Thomas, J.: Elements of Information Theory. Wiley, New York (1991)
8. Ferreira, C., Lopez, J., Perez Sinusia, E.: Uniform representations of the incomplete beta function in terms of elementary functions. Electron. Trans. Numer. Anal. **48**, 450–461 (2018)
9. Gers, F., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. Neural Comput. **12**(10), 2451–2471 (2000)
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y., Titterington, M. (eds.) Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol. 9, pp. 249–256. JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy (2010)
11. Good, I.: Some terminology and notation in information theory. Proceedings of the IEE - Part C: Monographs **103**(3), 200–204 (1956)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press. http://www.deeplearningbook.org (2016)

13. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 770–778. IEEE Computer Society (2016). https://doi.org/10.1109/CVPR.2016.90
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach, F., Blei, D. (eds.) Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 37, pp. 448–456. PMLR, Lille, France (2015)
17. Janocha, K., Czarnecki, W.: On loss functions for deep neural networks in classification. arXiv:1702.05659 (2017)
18. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) 3rd Proceedings of the International Conference on Learning Representations (ICLR), arXiv:1412.6980v9 (2015)
19. Knoblauch, A.: Power function error initialization can improve convergence of backpropagation learning in neural networks for classification. Neural Comput. **33**(8), 2193–2225 (2021)
20. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., Department of Computer Science University of Toronto (2009)
21. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc. (2012). https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
22. Krotov, D., Hopfield, J.: Dense associative memory for pattern recognition. arXiv:1606.01164 (2016)
23. Linnainmaa, S.: Taylor expansion of the accumulated rounding error. BIT Numer. Math. **16**(2), 146–160 (1976)
24. Mathematika: Version 12.3.1. Wolfram Research, Inc., Champaign, IL (2021). https://www.wolfram.com/mathematica
25. MATLAB: version 9.7.0.1247435 (R2019b). The MathWorks Inc., Natick, Massachusetts (2019)
26. Maxima: Maxima, a computer algebra system. version 5.43.2 (2020). http://maxima.sourceforge.net/. See also www.integral-calculator.com
27. Palm, G.: Novelty, Information and Surprise. Springer, Berlin (2012)
28. Parker, D.: Learning-logic: casting the cortex of the human brain in silicon. Tech. Rep. Tr-47, Center for Computational Research in Economics and Management Science. MIT Cambridge MA (1985)
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'AlchéBuc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
30. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention (MICCAI), LNCS, vol. 9351, pp. 234–241. Springer, Berlin (2015)
31. Ruder, S.: An overview of gradient descent optimization algorithms. arXiv:1609.04747 (2016)
32. Rumelhart, D., Hinton, G., Williams, R.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
33. Rumelhart, D., McClelland, J., Group, P.R. (eds.): Parallel Distributed Processing, Explorations in the Microstructure of Cognition, vol. 1. Foundations. MIT Press, Cambridge (1986)
34. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
35. Shannon, C., Weaver, W.: The Mathematical Theory of Communication. University of Illinois Press, Urbana/Chicago (1949)
36. Smith, S.: The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, San Diego (1997)
37. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(56), 1929–1958 (2014)
38. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (2019)

39. Tenne, N.: Uniform asymptotic expansions of the incomplete gamma functions and the incomplete beta function. Math. Comput. **29**(132), 1109–1114 (1975)
40. Werbos, P.J.: Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis, Harvard University (1974)