Noname manuscript No.
(will be inserted by the editor)

An energy-based SOM model not requiring periodic boundary conditions

Alexander Gepperth

Received: date / Accepted: date

Abstract We present the ReST (**Re**silient **S**elf-organizing **T**issue) model, a self-organized neural model based on an infinitely often continuously differentiable (C^∞) energy function. ReST extends older work on energy-based self-organizing models [12] in several ways. First of all, it converts input-prototype distances into neural activities that are constrained to follow a log-normal distribution. This allows a problem-independent interpretation of neural activities which facilitates, e.g., outlier detection and visualization. And secondly, since all neural activities are constrained in particular to exhibit a predetermined temporal mean, the convolution that is contained in the energy function can be performed using the so-called zero-padding with correction (ZPC) instead of periodic boundary conditions. Since periodic boundary conditions impose much stronger constraints on prototypes, using ReST with ZPC leads to markedly lower quantization errors especially for small map sizes. Additional experiments are conducted showing the worth of a C^∞ energy function, namely for novelty detection and automatic control of SOM parameters.

Keywords energy functions · self-adaptation · neural networks · self-organizing maps

PACS PACS 170 · PACS 180 · PACS 200 · PACS 240

1 Introduction

This article is in the context of self-organized map (SOM) models that have a continuous energy function. The lack of such an energy function for the original SOM model [14] has been the subject of a multitude of articles [6,20], and several proposals were made to remedy this problem. The advantages of models whose learning rule is derived from the minimization of an energy function are numerous, while the only disadvantages are that the existence of an energy function imposes strong constraints on the used learning rules. In particular, it was shown that the original SOM learning rule cannot be derived from a continuous energy function[12]. In this article, we propose an energy-based model of a topologically self-organizing neural map which we term the ReST (**Re**silient **S**elf-organizing **T**issue) model. The term "resilient" has been added because the proposed model contains an additional mechanism that enforces log-normal neural outputs and is thus robust against sudden changes in input statistics. In general, one may cite the following advantages of energy-based SOM models (which we experimentally validate in this article for the energy-based ReST model):

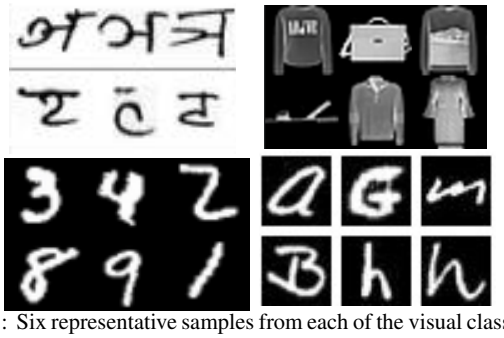

Fig. 1: Six representative samples from each of the visual classification datasets used in this study. From upper left to lower right: Devanagari, FashionMNIST, MNIST and EMNIST.

– **Estimation of learning success and parameter selection** A big issue for SOMs is to know whether the model has converged to a "desirable" state. For problems that do not allow a visual quality inspection of the learned prototypes (such as can be performed for the MNIST

A.Gepperth
University of Applied Sciences Fulda
Tel.: +49-(0)661-9640-3485
E-mail: alexander.gepperth@cs.hs-fulda.de

benchmark), there is no universal criterion to determine optimal values for the model parameters (final neighbourhood radius, final learning rate, topology etc.), whereas an energy function provides a scalar value that can be compared.
- **Proof of stability** If a continuous energy function exists and is bounded from below, this automatically guarantees the eventual convergence of SOM learning.
- **Use of advanced stochastic gradient descent methods** With a continuous energy function, many widely-used methods for performing stochastic gradient descent (SGD) in the domain of deep learning can be transferred to SOM learning. This is because methods such as RMSProp, Adam, AdaGrad, AdaDelta or "normal" SGD [9], which try to find "good" minima of an energy function, implicitly assume that such a quantity exists when adapting the meta-parameters of learning.
- **Outlier detection** When the energy (that is supposed to be minimized by the learning process) suddenly increases, this is a strong indication for a change in data statistics and can thus be used for outlier or concept drift detection. This latter property is especially relevant for our own ongoing work on incremental learning methods[8]. Conversely, if there is no energy function, it is not even very clear what quantity can be used for outlier detection, as there is no function that is minimized by learning.

Another novel point of ReST is the explicit enforcing of a particular temporal probability distribution (at least up to to the first and second moment) for all neural responses in a layer. This allows to interpret neural responses in a problem-independent way, which would not work for SOMs or other neural models since distances or neural activities in general are usually closely coupled to the problem that is being learned. Since the constrained optimization algorithm used for ReST learning achieves the desired statistical behavior by controlling two additional parameter per neuron, the visualization of those parameters can itself give interesting insights into the distribution of the training data, most prominently about sample density and variance in the Voronoi cell of a prototype.

1.1 Related work

Energy-based SOM models There has been a huge amount of primarily mathematical literature about SOMs. It was shown conclusively in [6] that the original Kohonen learning rule cannot be exactly derived from the minimization of *any* error function. In the same article, it is mentioned that the Kohonen learning rule follows instead from the individual minimization of per-neuron energy functions [20], but these functions are very complex, non-unique and do not lend themselves to a simple interpretation (e.g., minimization of a distortion measure or similar). Another approach was proposed by Kohonen[14] and taken further by Heskes[12]: instead of finding error functions whose minimization would lead to the Kohonen learning rule, these authors attempted to very slightly modify the Kohonen rule such that an energy function could be formulated. Obviously, the modification should in no way impair the self-organization capabilities of the model while allowing an intuitive interpretation through a (preferably simple) energy function. A modification satisfying these requirements was proposed in [12,11], offering a continuous energy function for discrete as well as continuous data distributions. While this was an important theoretical result, curiously enough there was no real follow-up in terms of applications in data visualization and/or clustering, which is surprising given the advantages an energy function can offer for SOM training, see above. It may be supposed that this lack of interest was due to the added computational complexity (an additional convolution needs to be calculated), as well as the problems that convolutions encounter at boundaries. Similar SOM variants having an energy function were proposed in [10] but they suffer from the same "convolution problem".

Data visualization using self-organized algorithms One of the primary application of self-organizing maps has always been the visualization of high-dimensional data, see, e.g., [7] which is particularly intuitive due to the topology-preservation property of SOMs. Various measures can then be superimposed on a visualization of prototypes, such as average quantization error, hit count or U-matrix (distances between adjacent prototypes), see [22]. A major interest in computing such SOM-based data representations has always been to understand the local properties of a data distribution as represented by SOM prototypes[19]. This is something that emerges automatically from the constrained-optimization learning rule of the ReST model, which intrinsically requires the computation of the first two moments of input-prototype distances for each neuron, allowing an easy visualization that offers a qualitatively new insight into the distribution of the data. Other modern data visualization schemes include t-SNE (Stochastic Neighbour Embedding, [18,21]) or SONE (self-organized neighbour embedding ,[3]), which place great emphasis on an as-correct as possible representation of the data but do not (easily) give access to local statistics.

2 Methods and data

In all experiments, we use the ReST model as described in Sec. 2.1, together with the visual classification datasets described in Sec. 3.1.

2.1 The ReST model

We assume a dataset (or a mini-batch) of input vectors $\mathbf{x}_n \in \mathbb{R}^k$ and a two-dimensional set of $K \times K$ neurons with non-negative activities $a_i \geq 0, i = 1 \ldots, K^2$. It is convenient to express activities computed for an input $\mathbf{x}_n$ as a one-dimensional vector $\mathbf{a}_n \in \mathbb{R}^{K^2}$. A neuron with (linear) index i and coordinates x_i, y_i has an associated prototype $\mathbf{p}_i \in \mathbb{R}^k, i = 1, \ldots, K^2$, as well as an $K \times K$ neighbourhood matrix that we write as a one-dimensional vector $\mathbf{g}_i \in \mathbb{R}^{K^2}$ in analogy to the vector of activities. Differing from the SOM model, each neuron furthermore possesses two internal variables o_i and s_i that play a role in enforcing log-normal statistics for the activities $\mathbf{a}_n$ that are computed as follows:

$$d_{ni} = \sqrt{(\mathbf{p}_i - \mathbf{x}_n)^2} \tag{1}$$

$$\tilde{a}_{ni} = o_i - s_i d_{ni} \tag{2}$$

$$a_{ni} = \exp(\tilde{a}_{ni}) \tag{3}$$

These internal variables, also denoted per-neuron parameters, have the function of raising or lowering the geometric mean (o_i) and geometric standard deviation (s_i) of each neuron's activity such as to obtain certain target values for these quantities. The log-normal distribution was chosen because it can readily enforce sparsity of neural responses. Since it is characterized by geometric mean and geometric standard deviation (which is multiplicative w.r.t. the geometric mean), it is fundamentally asymmetric in nature. Demanding a low geometric mean and a high geometric standard deviation will thus produce responses that are generally positive and low but can strongly deviate upwards (but not much downwards) on occasion, which is very useful if, e.g., classification is performed on ReST activities.

The adaptation of the prototypes $\mathbf{p}_i$ is now achieved by minimizing the energy function

$$c_{ni} = \langle \mathbf{g}_i, \log \mathbf{a}_n \rangle = \langle \mathbf{g}_i, \tilde{\mathbf{a}}_n \rangle \tag{4}$$

$$\mathcal{E} = \frac{1}{N} \sum_n \langle \mathbf{c}_n, \mathbf{S}(\mathbf{c}_n) \rangle . \tag{5}$$

The first equation essentially represents a convolution operation as the per-neuron vectors $\mathbf{g}_i$ are (for self-organized models) represented by Gaussians centered on neuron i. Generally, one assumes such Gaussians to be periodic where they exceed the map boundaries (for neurons that are close to these boundaries). In this article, we investigate the possibility to simply cut off the Gaussians at map boundaries but to re-weigh them according to the part that is "lost".

The logarithm and the vector-valued softmax function $\mathbf{S}(\mathbf{v})$ in eqn.(4) are applied in a component-wise fashion as

$$e_i = \exp(\beta v_i) \tag{6}$$

$$S(\mathbf{v})_i = \frac{e_i}{\sum_j e_j} \equiv S_i, \tag{7}$$

β being a parameter that controls the selectivity of the softmax: for higher β values, the output $S(\mathbf{v})$ will tend to be more strongly peaked, the maximal value closer to 1.0 and the rest to 0.0. For lower β values, this relationship is inversed. The minimization of the energy function is performed as a constrained optimization problem, the constraint being that the temporal distribution of activities $\mathbf{a}_n$ is log-normal with parameters μ and σ. This implies that $\log \mathbf{a}_n$ (with logarithm applied component-wise!) is normally distributed, with the empirical mean and standard deviation $\hat{\mu}$, $\hat{\sigma}$ coinciding with μ, σ:

$$\hat{\mu} \equiv \frac{1}{N} \sum_n \log a_{ni} = \frac{1}{N} \sum_n \tilde{a}_{ni} \stackrel{!}{=} \mu \tag{8}$$

$$\hat{\sigma} \equiv \sqrt{\frac{1}{N} \sum_n (\log a_i - \hat{\mu})^2} = \sqrt{\frac{1}{N} \sum_n (\tilde{a}_i - \hat{\mu})^2} \stackrel{!}{=} \sigma \tag{9}$$

From these requirements, the per-neuron parameters o_i and s_i can be determined unambiguously from the first two moments of the input-prototype distances

$$s_i = \sqrt{\frac{\sigma^2}{\overline{d_i^2} - \overline{d_i}^2}} \tag{10}$$

$$o_i = \mu + s_i \overline{d_i}, \tag{11}$$

which can be computed empirically over a dataset of N samples:

$$\overline{d_i} = \frac{1}{N} \sum_n d_{ni} \tag{12}$$

$$\overline{d_i^2} = \frac{1}{N} \sum_n d_{ni}^2 \tag{13}$$

In a mini-batch setting, we instead take averages over the current mini-batch of N samples (the extreme case being fully online learning where $N = 1$). If we wish to compute the averages $\overline{d_i}$ and $\overline{d_i^2}$ over periods longer than the mini-batch size N, we replace eqns.(12) by a composite method that makes use of a moving average of mini-batches averages:

$$\overline{d_i}(\nu) = (1 - \alpha_d N) \overline{d_i}(\nu - 1) + \alpha_d \sum_n d_{ni} \tag{14}$$

$$\overline{d_i^2}(\nu) = (1 - \alpha_d N) \overline{d_i^2}(\nu - 1) + \alpha_d \sum_n d_{ni}^2 \tag{15}$$

where variable ν expresses the number of the current mini-batch. We scale the adaptation rate $\alpha_d < 1$ with the mini-batch size N since a larger N implies that more samples are used per step in eqn.(14), and thus adaptation can proceed more quickly. Please note that a choice of $\alpha_d = 0$ would turn off the moving average mechanism. In this case only the current mini-batch would be considered, as it is the case in eqn.(12).

2.1.1 ReST learning rule

For performing gradient descent for the energy function of eqn.(4), we take its derivative w.r.t. to the k-th element of prototype i:

$$\frac{\partial E}{\partial p_{ik}} = \frac{\partial}{\partial p_{ik}} \frac{1}{N} \sum_{nj} c_{nj} S(\mathbf{c})_{nj} = \tag{16}$$

$$= \frac{1}{N} \sum_{nj} \left(S(\mathbf{c}_n)_j \frac{\partial c_{nj}}{\partial p_{ik}} + \frac{\partial S(\mathbf{c})_{nj}}{\partial p_{ik}} c_{nj} \right) = \tag{17}$$

$$= \frac{1}{N} \sum_{nj} \left(S(\mathbf{c}_n)_j \frac{\partial c_{nj}}{\partial p_{ik}} + \beta S(\mathbf{c}_n)_i \left(\delta_{ij} - S(\mathbf{c}_n)_j\right) \right) \tag{18}$$

where we have used the expression $\partial_j S_i = \beta S_i(\delta_{ij} - S_j)$ for the derivative of the softmax function. If we assume that the softmax function is parameterized such that it puts 1.0 at the position of the maximal value (whose index is expressed by $*$), and 0 everywhere else, we obtain:

$$\frac{\partial E}{\partial p_{ik}} \approx \frac{1}{N} \sum_{n} \frac{\partial c_{n*}}{\partial p_{ik}} \tag{19}$$

and arrive at the update rule (with learning rate α

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{\alpha s_i g_{*i}}{2N} \sum_n \frac{\mathbf{p}_i - \mathbf{x}_n}{||\mathbf{p}_i - \mathbf{x}_n||} \tag{20}$$

where we have one more time designed the index of the best-matching unit (BMU) by a star:

$$* = \arg\max_i c_i \tag{21}$$

If we had omitted the square root in the definition of input-prototype distances in eqn. (1), we would have arrived at the equivalent rule

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{\alpha s_i g_{*i}}{N} \sum_n \left(\mathbf{p}_i - \mathbf{x}_n\right) \tag{22}$$

which differs (for the online case of $N = 1$) from the energy-based SOM model proposed in [12] only by a factor of s_i for each neuron. Regarding the original SOM model [15], the additional difference is that the best-matching unit is not determined from input-prototype distances but from the convolution $\mathbf{c}$ of activities with the neighbourhood matrix as given in eqn. (4). We can therefore see that the learning rules (20,22) scale each neuron's prototype adaptation by a factor that is, by eqn. (10), inversely proportional to the variance of activities of that neuron. Thus, neurons whose prototypes are either too unspecific or too generic (resulting in uniformly low or high activations with low variance) receive a competitive advantage. We also note that this mechanism is self-limiting: increased prototype adaptation usually increases the variance of a neuron's activities, thus eventually annulling the competitive advantage and leading to stable competitive learning dynamics.

2.1.2 Implementation of constrained optimization

Minimizing the energy function (4) is performed by performing repeated gradient descent steps using learning rule (20) on the whole available training data set or mini-batch, each step followed by an explicit enforcement of the constraints by applying eqn. (10), this again being followed by an update of the averages using eqn.(12).

For speeding up convergence, the neighbourhood matrix $\mathbf{g}_i$ of neuron i is modelled as a Gaussian whose standard deviation $S(\nu)$ is decayed exponentially over time, as it is usual with SOMs:

$$g_{ij} = \exp\left(-\frac{(x_j - x_i)^2 + (y_j - y_i)^2}{2S(\nu)^2} \right) \tag{23}$$

In contrast to normal SOM learning we do not decay the ReST learning rate α over time, since this complicates advanced gradient descent strategies and introduces unnecessary parameters. Additionally, we impose an initial period without prototype adaptation where only the neural statistics are adapted. This allows to perform "adiabatic" prototype updates causing only small corrections to the already converged o_i and s_i, which avoids potentially problematic feedback loops between the two adaptation processes. The detailed training procedure, as well as the relevant parameters, are detailed in Alg. 1.

2.1.3 Choice of ReST parameters

The self-adaptation process is governed by the parameters μ and σ of the log-normal distribution that the activities a_i are required to obey, which raises the question of what their intrinsic significance could be, especially within the context of self-organizing maps and incremental learning. First of all, from the properties of log-normal distributions we know that the quantity e^{μ} represents both the geometric mean and at the same time the median of a log-normally distributed variable, so essentially we could just fix a median value M and compute $\mu = \log M$ from it. The median for this distribution is smaller but usually close to the arithmetic mean as well so we can also see M as a rough indicator for the arithmetic time average of a neuron's activity. The quantity e^{σ} is sometimes termed the geometric standard deviation and can be expressed as

$$e^{\sigma} = \exp\left(\sqrt{\frac{1}{N} \sum_n \left(\log \frac{a_{ni}}{e^{\hat{\mu}}} \right)^2} \right) =$$

$$= \sqrt[N]{\Pi_n \exp\left(\left(\log \frac{a_{ni}}{e^{\hat{\mu}}} \right)^2 \right)} = E_n^g \sqrt{\exp\left(\left(\log \frac{a_{ni}}{e^{\hat{\mu}}} \right)^2 \right)} \tag{24}$$

and is thus related to the geometric mean of the expression $\sqrt{\exp\left(\left(\log \frac{a_{ni}}{e^{\hat{\mu}}} \right)^2 \right)}$. This expresses the multiplicative

Algorithm: Constrained ReST optimization

Parameters:

- nr of iterations T
- mini-batch size N
- initial and final neigh. radius S_0, S_∞
- learning rate α
- self-adaptation rate α_d
- time parameters t_A, t_0 and t_∞
- target values σ, $M = e^\mu$ for self-adaptation
- softmax parameters: $\beta = 300$, applied 3 times

Result: trained prototypes $\mathbf{p}_i$

begin

Initialize all prototypes $\mathbf{p}_i$ to small random values ;

Initialize moving averages $\overline{d_i}(0) = 0$ and $\overline{d_i^2} = 0$;

Initialize per-neuron parameters $s_i = 0.5$, $o_i = 0$;

Compute decay time constant $\lambda = -\frac{\log(-S_\infty/S_0)}{t_\infty - t_0}$;

for *mini-batch* $\nu < T$ **do**

compute nb.radius $S(\nu)$ and learning rate $\alpha(\nu)$: **begin**

if $\nu < t_A$ **then** $\alpha(\nu) = 0$, $S(\nu) = S_0$;

else if $\nu < t_0$ **then** $\alpha(\nu) = \alpha$, $S(\nu) = S_0$;

else if $\nu < t_\infty$ **then** $\alpha(\nu) = \alpha$, $S(\nu) = S_0 e^{-\lambda\nu}$;

else $\alpha(\nu) = \alpha$, $S(\nu) = S_\infty$;

end

recompute nb. matr. $\mathbf{g}_i$ based on $S(\nu)$;

select a random mini-batch $\mathbf{x}_n$, $0 < n < N$;

update prototypes $\mathbf{p}_i$ according to eqn. (20) ;

enforce constraint using eqn. (10) ;

adapt averages $\overline{d_i}(\nu)$ and $\overline{d_i^2}(\nu)$ using eqn. (14) ;

end

return $\mathbf{p}_i$

end

Algorithm 1: Mini-batch based learning with the ReST model.

spread of values around their empirical geometric mean $e^{\hat{\mu}}$, regardless of the direction. Higher values of e^σ will push the activities further away from their geometric mean, forcing them to be more specific, either close to 0 or far away from it. We can thus think of σ as a parameter controlling the sparsity of neural responses, which previous studies on transfer functions for self-organized maps [17] found to be an important factor for performing classification based on SOM activities.

The actual values for μ and σ ultimately depend on the chosen application: when, e.g., classification is performed on the activities a_i, then a low median and high sparsity may be beneficial, so as to have generally low values except those that exceed the median strongly. For outlier detection, the precise values are less important since only the deviation from the imposed probability distribution need to be detected.

In order to guarantee identical functioning of the WTM mechanism for variable map sizes, the softmax function needs to be parameterized correctly, and more specifically as a function of the number of neurons in the SOM. We therefore need to set the parameter β such that qualitatively identical behavior ensues for any map size. We measure identical behavior by demanding that the the maximal response of the softmax function be ξ when given a vector $\mathbf{x} \in \mathbb{R}^k$ that consists of $n-1$ times value B and 1 time value λB. Solving this for β gives us the expression

$$\beta = \frac{\ln(\xi^{-1} - 1) - \ln(n-1)}{B(1-\lambda)} \tag{25}$$

The softmax function is a very useful tool for obtaining a "hard" yet differentiable winner selection, in addition to allowing a steady transition between "hard" and "soft" winner selection. In some cases, problems can occur: first of all, sensible choices for B and λ may be hard to obtain because they depend on the learning dynamics. Furthermore, when $\beta > 700$, numerical issues arise due to the exponentials involved. Fortunately there is a simple rule-of-thumb solution for both problems that consists of applying a softmax function with "best guess" parameters *several times* in eqn. (4). This complicates the gradient, but as long as the final softmax function gives a sufficiently hard winner assignment, the learning rule (20) remains valid. Software frameworks like TensorFlow can compute the gradient symbolically, so even the exact gradient can be used regardless of how often softmax was applied. We found that a three-fold application was always sufficient to guarantee a unique winner selection.

The parameter S_0 is usually made to depend on the map size. A rule of thumb that always worked well is to choose it proportional to the diagonal of the quadratic $K \times K$ map, i.e., $S_0 = \frac{K}{4}$. In contrast, classification experiments always give best results the smaller S_∞ is, so this is always fixed at small values like $S_\infty = 0.01$. The values of t_0, t_A and t_∞ can be determined empirically be requiring that i) self-adaptation has occurred before t_A ii) the energy function has converged to a stable value before t_0 and iii) that the energy function is as low as possible while still satisfying all constraints at t_∞. Here, we see the value of an energy function as it can be used to determine convergence, so these parameters which for SOMs have to be obtained by visual inspection, can be determined by cross-validation. By a similar reasoning, a good value for the learning rate can be obtained, where smaller values are always acceptable but lead to increased training time. The mini-batch size is generally

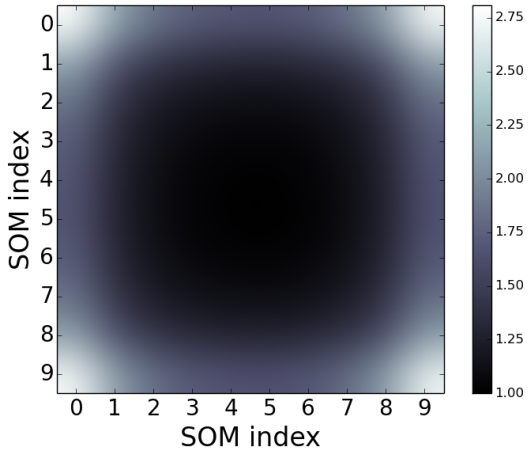

Fig. 2: Visualization of the inverted re-weighting map χ_i^{-1} for a 10x10 map and a neighbourhood radius of $S = 2$. It can be ob served that at the center of the map the values are equal to 1.0 corresponding to the fact that no correction for boundary effects is necessary there. In the corners, where the convolution filter has the lowest overlap with the distance map, correction factors grow strongly to compensate.

assumed to be $N = 100$ in this article. The self-adaptation rate, $\alpha_d N$, should be chosen such that the constraints are approximately upheld during prototype adaptation, meaning it will depend on the choice of α and is thus not a free parameter but can be indirectly obtained by cross-validation.

2.2 Implementing zero-padding with correction (ZPC) boundary conditions with ReST

The convolution in eqn.(4) can exceed the boundaries of the neural map, meaning that the Gaussian in the scalar product can have nonzero values for indices larger than n or smaller than 0. This problem has been long known in the computer vision literature when dealing with image convolutions [13], and is usually addressed by imposing particular boundary conditions. Traditional possibilities are *zero-padding* where all elements that do not fall into the image are treated as zero, or *periodic boundary conditions* where the image is considered a torus in both dimensions, and thus elements outside the image are taken from the opposite side of the image. Whatever boundary conditions are imposed, although they make convolution formally possible, they corrupt the integrity of the convolved image at its borders because they infer values for the original image that do not exist. Either, for zero-padding boundary conditions, convolution results are weak because of many adjacent zero values produced by the boundary conditions, or convolution results are uncorrelated with their neighbourhood as the opposite of the map influences the results. In this article, we propose a simple solution to this problem in the context of zero-padding conditions: we statically **re-weigh convolution results near the map borders** according to the part of the neighbourhood filter that falls outside the map. Without the re-weighting, activities would fall off towards the map borders because larger and larger parts of the neighbourhood filter are missing (or rather: applied to zeroes outside the image, thus not giving a contribution). As a consequence, units at the map border would effectively have no chance to ever become BMUs and thus the representational capabilities of the SOM would be impaired. The re-weighting is a multiplicative unit-wise operation:

$$\chi_i = \sum_j g_{ij}$$
$$g_{ij} \rightarrow \frac{g_{ij}}{\chi_i} \tag{26}$$

The re-weighting vector $\boldsymbol{\chi}$ measures, at each point (x_i, y_i), which fraction of the mass of the applied neighbourhood filter lies within the map, see Fig. 2. Inserted into the definition of the activity map $\mathbf{c}$ in eqn.(4), one perceives that the introduction of the re-weighting map does not depend on the model parameters (i.e., the prototypes $\mathbf{p}_i$) and thus does not affect the learning rule other than by a position-dependent constant factor. We will refer to this strategy of treating boundary conditions as "zero-padding with correction" (ZPC).

An alternative solution is to impose periodic boundary conditions for the convolution in eqn.(4) by appropriately calculating the per-neuron neighbourhood maps $\mathbf{g}_i$ for every neuron i. This solution is probably slightly faster to implement with existing optimized methods. It is the purpose of this article to show that the ReST model can indeed work for both types of boundary conditions, and that the proposed ZPC boundary conditions achieve significantly lower quantization errors (leaving all other parameters unchanged) because no constraints are imposed on prototypes close to the map boundaries. This should be especially true for small map sizes, in which we are particularly interested.

3 Experiments

The ReST model used in all experiments is implemented in TensorFlow 1.5 [1] using the Python interface. The (quite complex) gradient computation in analogy to eqns.(20,16) are computed automatically by the software. For implementing gradient descent for the energy function (4), we use again TensorFlow to create a plain stochastic gradient descent optimizer from the computed gradient. The reason why we do not use more advanced optimizers is that, very simply, it is not necessary, although all of the optimizers mentioned in the introduction can minimize the ReST energy function equally well.

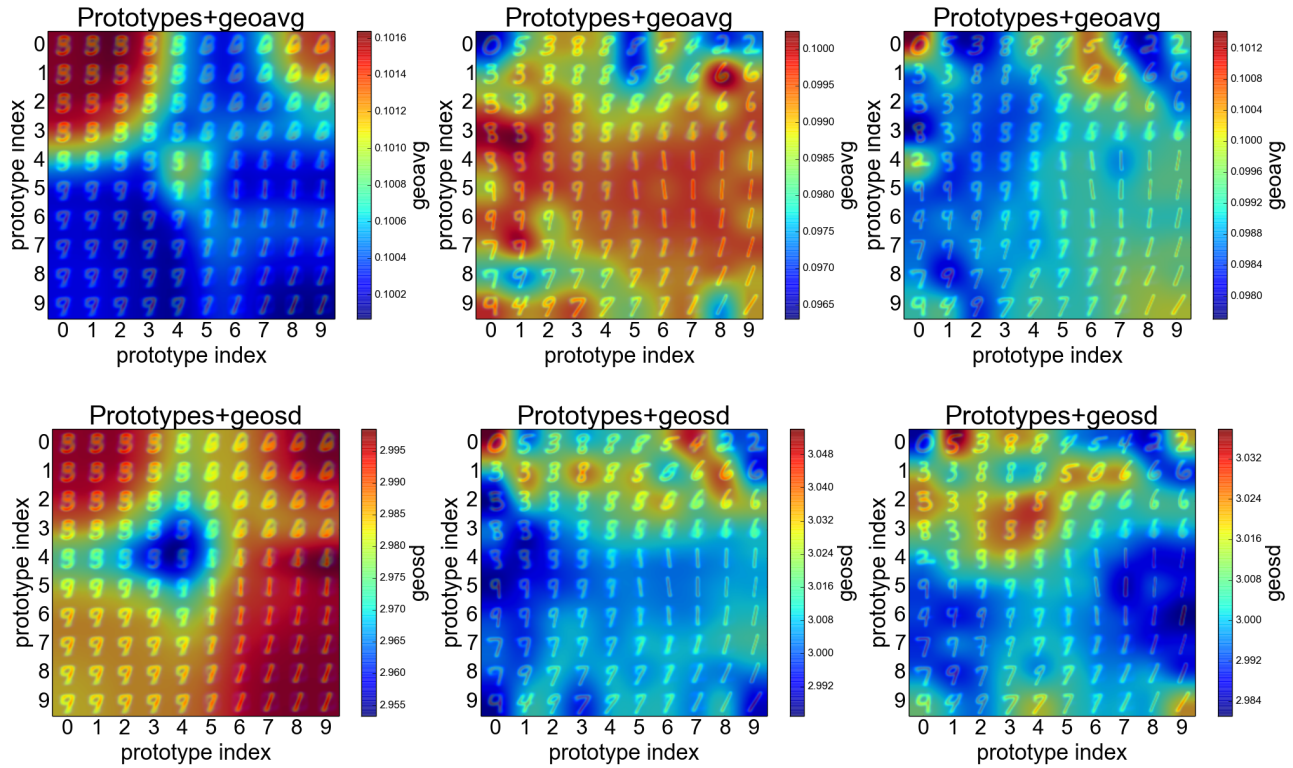

Fig. 4: **(best viewed in color)** Different stages of ReST training on the MNIST dataset. Upper row, from left to right: ReST prototypes with long-term geometric activity averages superimposed on them for times $\nu = 7000, 12000, 24000$. Lower row, from left to right: ReST prototypes with long-term geometric standard deviation averages superimposed on them for times $\nu = 7000, 12000, 24000$. We can observe that activity averages and deviations are strictly adhered to by inspecting the color bars at the side of each figure: the displayed colors are in a very close range around the target values of 0.1 (e^{μ}) and 3(e^{σ}). Equally observable: a typical SOM-like topological organization of prototypes.

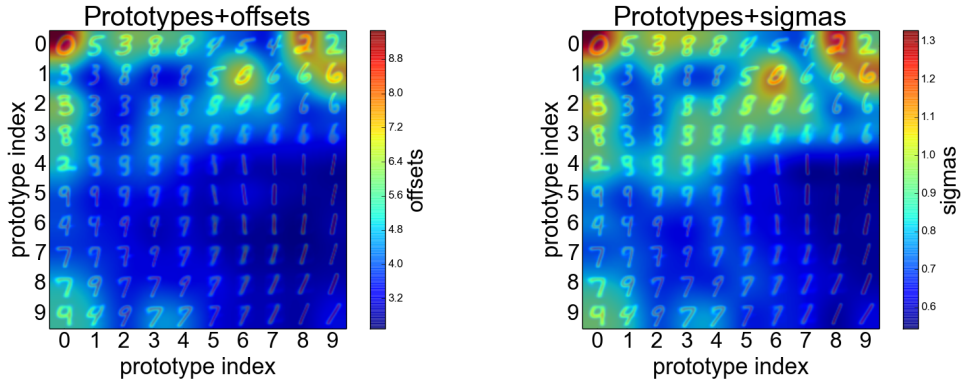

Fig. 5: **(best viewed in color)** Distribution of per-neuron parameters o_i and s_i for MNIST, after convergence of the ReST layer at iteration 24000. To be compared to rightmost column in Fig. 4 which corresponds to the same iteration. Comparison reveals that, although the values of geometric mean and standard deviation are very uniform across the ReST layer (Fig. 4), the per-neuron parameters that achieve this uniformity are not, because each neuron would have very different statistics without adaptation. ReST layer regions with low offsets indicate that these neurons would have a high average activity without adaptation; a similar reasoning holds for the variances. Please compare as well to Fig. 3 for an interpretation: "blue regions" in the right-hand plot reveal that data points are more dense in this region, whereas bright regions indicate that data points are far apart.

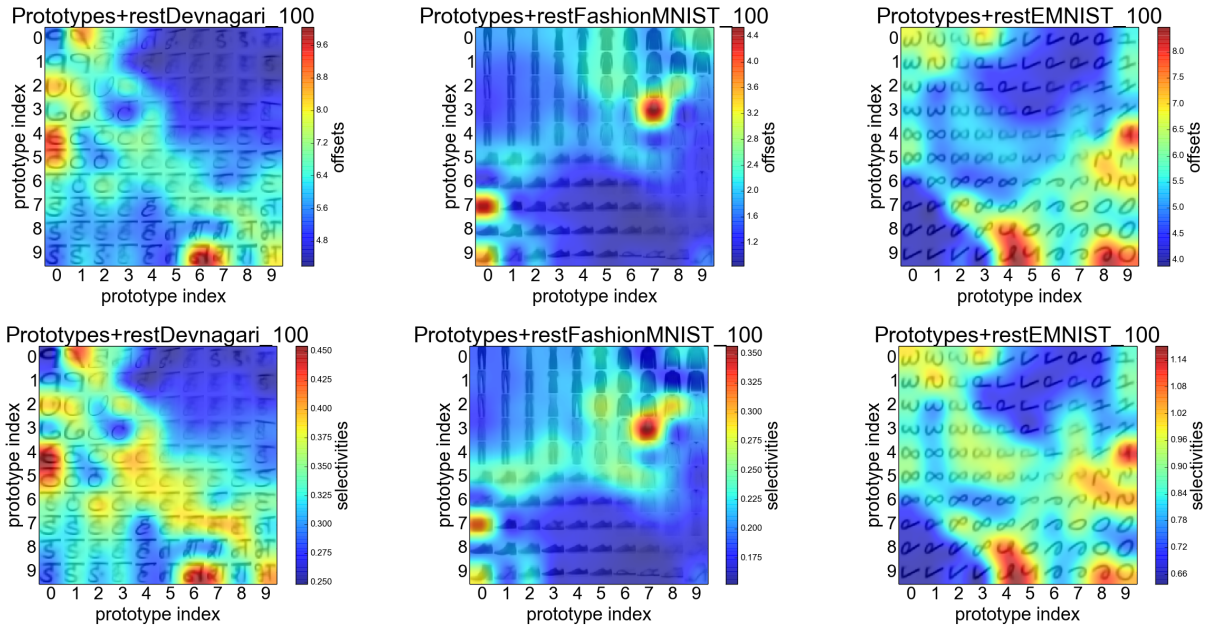

Fig. 6: **(best viewed in color)** Distribution of per-neuron parameters o_i (offsets, upper row) and s_i (selectivities, lower row) for Devanagari (left column), FashionMNIST(middle column) and EMNIST (right column) after convergence of the ReST layer at iteration 24000. Training was performed on classes 1-9 of all datasets.

Table 1: Overview of each dataset's detailed properties. Image dimensions are given as width × height × channels. Concerning data imbalance, the largest percentual difference in sample count between any two classes is given for training and test data, a value of 0 indicating a perfectly balanced dataset.

Properties / **Dataset**	image size	number of elements		class balance (%)	
		train	test	train	test
Devanagari	32 × 32 ×1	18.000	2.000	0.3	2.7
EMNIST	28 × 28 ×1	345.035	57.918	2.0	2.0
FashionMNIST	28 × 28 ×1	60.000	10.000	0	0
MNIST	28 × 28 ×1	55.000	10.000	2.2	2.4

3.1 Data

We select the following datasets (see Tab. 1). In order to construct SLTs uniformly across datasets, we choose the 10 best-represented classes (or random classes if balanced) if more are present.

MNIST [16] is the common benchmark for computer vision systems and classification problems. It consist of gray scale images of handwritten digits (0-9).
EMNIST [4] is an extended version of MNIST with additional classes of hand-written letters. There are different variations of this dataset: we extract the ten best-represented classes from the *By_Class* variation containing 62 classes.
Devanagari [2] contains gray scale images of Devanagari handwritten letters. From the 46 character classes (1.700 images per class) we extract 10 random classes.
FashionMNIST [23] consists of images of clothes in 10 classes and is structured like the MNIST dataset. We use this dataset for our investigations because it is a "more challenging classification task than the simple MNIST digits data [23]".

3.2 Experimental parameters and their justification

Unless stated otherwise, ReST parameters are chosen as follows (in the terms of Sec. 2.1): $K = 10$, $N = 100$, $T = 30000$, $t_A = 5000$, $t_0 = 10000$, $t_\infty = 20000$, $S_0 = K/4$, $S_\infty = 0.1$, $\alpha_d = 0.01$, $\alpha = 0.05$, $e^\sigma = 3$ and $M = e^\mu = 0.1$. The initial neighbourhood radius is set smaller ($K/4$) than usual for SOMs since we found that this has no impact on results but allows for shorter convergence times. The choices for μ and σ are mainly motivated by visualization (relatively high geometric standard deviation and low geometric mean so deviations are well visible). Softmax is applied 3 times with a parameter of $\beta = 500$ which was empirically found to be an excellent choice. The choices for t_A, t_0 and t_∞ might seem too low for a dataset of 60000 samples such as MNIST, but bear in mind that the mini-batch size has been set to $N = 100$, so 10000 iterations correspond

to 1000000 processed samples, which should give the algorithm ample time to converge. This holds for all the other datasets described in Sec. 3.1 as well.

3.3 Preliminary experiment: intuitive interpretation of the self-adaptation parameters

To better understand what the self-adaptation mechanism in ReST actually does, we create a set of 10.000 two-dimensional data points $\mathbf{x}_i \in \mathbb{R}^2$ which are drawn from a normal distribution with mean $\boldsymbol{\mu} = (0.5, 0.5)^T$ and standard deviation $\Sigma = 0.15$. We subsequently train a ReST layer of size $K = 10$ using the default parameters of Sec. 3.2. The final prototype positions and values of the per-neuron parameters s_i and o_i are shown in Fig. 3 and show the following things:

- where data points are more dense(sparse), offsets o_i are lower(higher). This is intuitive since prototypes that react to less frequently occurring samples need to have a higher offset to maintain a constant average activity.
- where data points are more dense(sparse), parameters s_i are higher(lower), meaning that a neuron will react less(more) strongly to nearby samples. This is intuitive as well, since a higher number of nearby samples would mean a near-constant activity, with low variance, if neurons could not become more selective in their reactions.

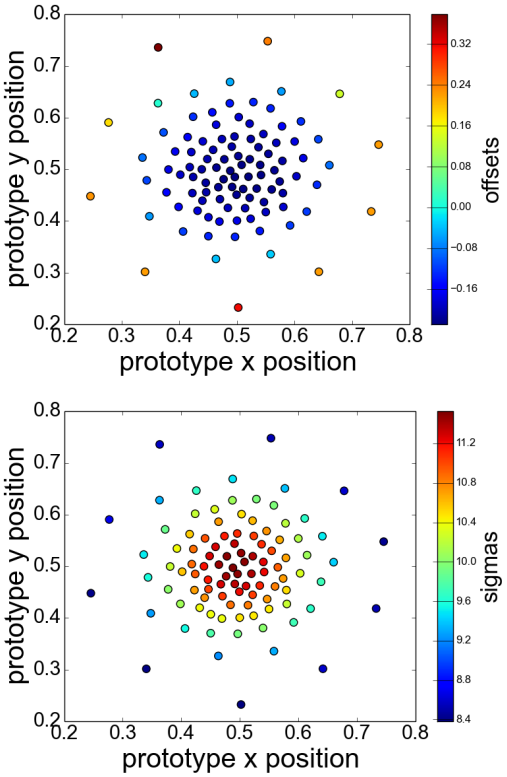

Fig. 3: **(best viewed in color)** Prototype positions overlaid in color with per-neuron parameters o_i (left) and s_i (right) when training a ReST layer on a 2D normal distribution.

We therefore link the per-neuron parameters s_i to the selectivity of the corresponding neuron.

These results show that ReST neurons can adapt to the sample density in their Voronoi cell, a behavior that closely mimicks self-adaptation mechanisms in biological neurons.

3.4 Self-organization and self-adaptation in the ReST model

In this section we will demonstrate that the ReST model, while differing from both the original SOM model [15] and the energy-based "Heskes model" [12], achieves the same basic type of prototype self-organization. At the same time, we will demonstrate the effectiveness of ReST's self-adaptation process as described in Sec. 2.1 and comment on its beneficial effects. To this end, we will conduct simulations with all datasets described in Sec. 3.1. ReST parameters are chosen as described in Sec. 3.2. After convergence of the neighbourhood radius to its asymptotic value at t_∞, we wait for another 6000 iterations to allow the energy function to stabilize, then (if needed) statistics is collected for 1000 iterations and subsequently evaluated. Histograms of all neural activities during these 1000 iterations are computed and compared to the theoretical log-normal distribution determined by μ and σ.

Self-organization From Fig. 4, it can be observed that self-organization proceeds exactly in the same manner as in a SOM, starting with a coarse "global ordering" of prototypes followed by refinement as $S(\nu)$ is decreased, showing that ReST performs essentially the same function as a SOM, only with convergence in 2D guaranteed and a self-adaptation process that gives a probabilistic interpretation to the computed activities.

Self-adaptation As can furthermore be seen in Fig. 7, the fit between theoretical and measured distribution is generally acceptable for all datasets, although of course a perfect fit is not to be expected. This is because we only fit the first two moments of the log activities to defined values. For a better fit, at least the third moment of the log activities should be controlled, which would however result in a more complex constrained optimization scheme. Fig. 5 shows this homogeneity is achieved by quite heterogeneous settings of the per-neuron parameters o_i and s_i, see eqns. (1,10,11).

3.5 Boundary conditions and quantization error

This experiment investigates the effects of different boundary conditions when training a ReST model described in Sec. 2.1. In particular, we compare the "zero-padding with correction" (ZPC) strategy of Sec. 2.1 with periodic boundary conditions. Periodic boundary conditions impose more

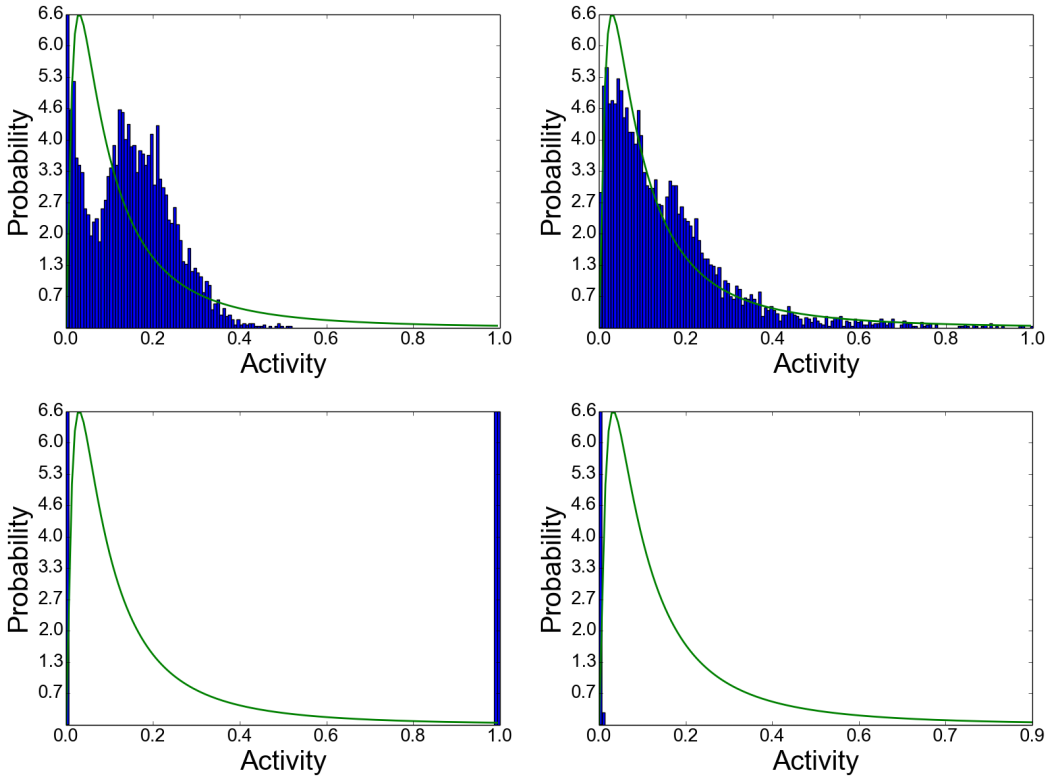

Fig. 7: Activity histograms for neuron $(4, 4)$ in a ReST layer trained on the Devanagari (left) and MNIST (right) problems, both for the case of enabled (upper row) and disabled (lower row) self-adaptation. The theoretical log-normal density is superimposed onto the histograms as a solid green line, showing a generally good match for both tasks.

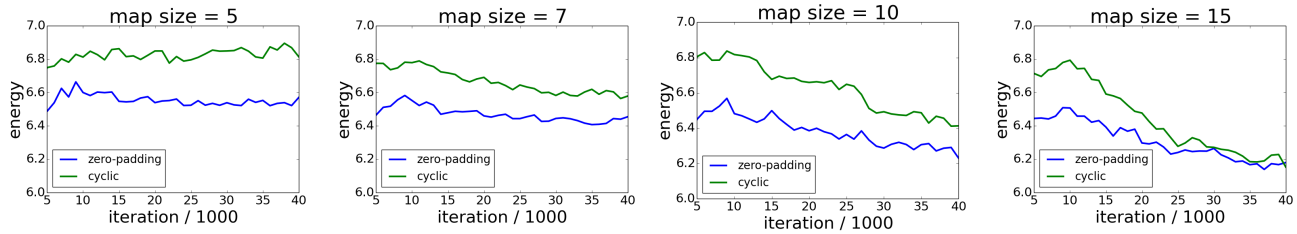

Fig. 8: Quantization errors for an energy-based ReST layer, comparing periodic and ZPC boundary conditions, showing a clear difference between the two conditions especially for small map sizes. A smaller map size manifests itself by higher final quantization errors, and it is evident that larger maps suffer less from periodic boundary conditions than small ones.

constraints on prototypes as they are applied at every map boundary. In contrast, ZPC boundary conditions impose no constraints on prototypes at to the borders, leaving more freedom to model the data faithfully. The basic quantity we consider here is the so-called quantization error measure, which for a sample $\mathbf{x}_n$ and a set of prototypes $\{\mathbf{p}_i\}$ is defined as $q(\mathbf{x}_n, \{\mathbf{p}_i\}) = \min_i ||\mathbf{x}_n - \mathbf{p}_i||$, and which measures the capacity of prototypes to approximate input to a ReST layer. The basic question we ask is: does the manner of treating boundary conditions show up in the quantization error of a ReST layer (see 2.1). To investigate this, we train such a layer, with both kinds of boundary conditions on the training set of the MNIST problem described in Sec. 3.1 and measure its quantization error on the test set.

Map size is varied in the interval $K = \{5, 7, 10, 15\}$ in order to assess the influence of the map size. The other parameters are the same as in Sec. 3.2 except for $T = 40000$, $S_\infty = 1.5$ for $K > 5$ and $S_0 = 2.5$, $S_\infty = 1.0$ for $K = 5$. This setting of S_∞ differs from the preceding section where S_∞ is small. This would make neighbouring prototypes independent, which, in turn, would make the re-weighting map a delta spike and thus eliminate the boundary correction effect. Boundary treatment intrinsically makes sense only for not too small values of S_∞.

We compare quantization errors on the MNIST test set over time for both boundary condition types, results are given in Fig. 8. As can be seen, the proposed ZPC boundary conditions work and generate prototypes with consistently lower

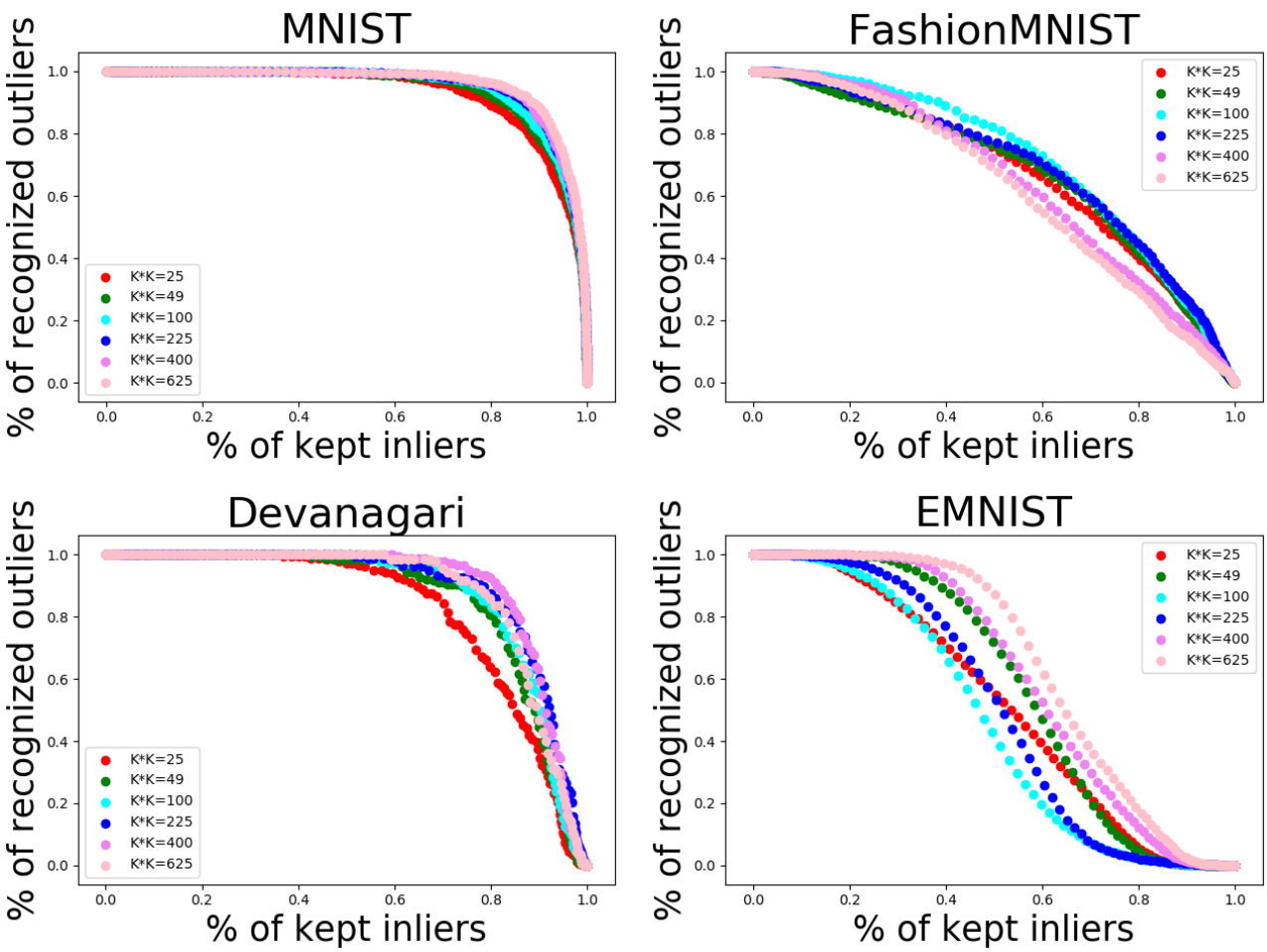

Fig. 9: Energy-based outlier detection for various map sizes. Shown is the percentage of rejected outliers plotted against the percentage of accepted inliers. Upper row, left to right: MNIST, FashionMNIST. Lower row, left to right: Devanagari, EMNIST.

quantization errors than periodic boundary conditions can reach, especially for small map sizes.

3.6 Energy-based rejection strategies

It is well known that SOM prototypes approximate the distribution of input vectors[5], and that the distance between an input vector and the prototype of the BMU can be used as a measure of a priori probability for this input vector. In other words, inputs the ReST layer has rarely or never seen will produce high distances d_{ni} as defined in eqn. (1) and can thus be potentially identified. In this experiment, we wish to determine whether the same holds for the analogous quantity of ReST models, the BMU activity c_{n*}, see eqn. (1).

Concretely, we train a ReST layer with ZPC boundary conditions of different map sizes K only on the classes 1 through 9 of all image recognition problems described in Sec. 3.1 and treat samples of class 0 as outliers that should be detected by imposing a threshold θ on the quantity c_{n*}:

$$\text{class}= \begin{cases} \text{inlier} & \text{if } c_{n*} > \theta \\ \text{outlier} & \text{else} \end{cases}$$

Training and test parameters are as described in Sec. 3.2 except for K which is varied as $K = \{5, 7, 10, 15, 20, 30\}$. By logging the responses of a trained ReST layer to test samples of the various datasets, and by varying the threshold θ, we obtain rejection curves as shown in Fig. 9, each point of which represents a different trade-off between retaining inliers and suppressing background outliers.

It can be seen from Fig. 9 that the rejection strategy based on the activity map works, and increasingly well so with larger map sizes. The interesting point here is that the basic quantity $c_{n*} = \max_i c_{ni}$ used for rejection here is nothing else but the (approximate) instantaneous energy of the SOM given the current sample (since the softmax function will select only the maximal value from the sum in eqn. 4). This means that, by optimizing the energy, one directly optimizes the quantity necessary for outlier detection, which is an elegant and intuitive approach.

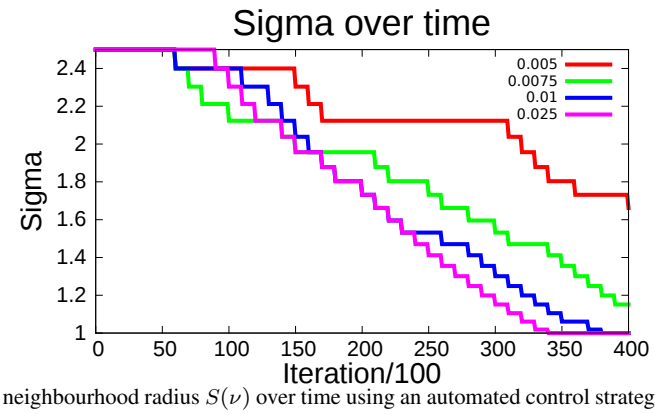

Fig. 10: Development of the ReST neighbourhood radius $S(\nu)$ over time using an automated control strategy. Experiments differ only in the value of the threshold θ_D that determines the upper limit on energy evolution that is allowed to consider the energy stationary. More restrictive values of θ_D result in slower convergence of S, and in a slower decrease of the energy function. It may be noted that the evolution of S is reminiscent of the exponential decay usually modeled when training self-organized models, only here it arises automatically from the specified control law.

3.7 Energy-based control strategies

As stated in the introduction, the behavior of the energy function can give valuable hints about controlling the time-dependent neighbourhood radius $S(\nu)$ in the ReST update rule (20). As any change in $S(\nu)$ directly impacts, through the activity map $\mathbf{a}_n$, the energy function defined in (4), changes to $S(\nu)$ should be adiabatic, i.e., small and effected only when the energy is stationary. This is the key point of the admittedly simplistic control strategy we adopt here, just to show what an energy function can do for SOM users: the detection of stationary states where no further adaptation is happening. This is effected by first exponentially smoothing the instantaneous energy values $E(\nu)$ from eqn. (4), computed from the current mini-batch or sample with index ν, subsequently computing a measure of how much the smoothed energy changed in a characteristic time interval defined by the constant τ:

$$\begin{aligned}
\hat{\mathcal{E}}_\tau(0) &= 0 \\
\hat{\mathcal{E}}_\tau(\nu) &= (1-\tau^{-1})\hat{\mathcal{E}}_\tau(\nu-1) + \tau^{-1}\mathcal{E}(\nu) \\
D(\nu) &= 1 - \frac{\hat{\mathcal{E}}_\tau(\nu)}{\hat{\mathcal{E}}_\tau(\nu-\tau)}
\end{aligned} \tag{27}$$

and subsequently imposing a threshold on $D(\nu)$ to implement the following control strategy:

$$\sigma(\nu) = \begin{cases} \max\left(\sigma(\nu-1)\xi, \sigma_\infty\right) & \text{if } 0 < D < \theta_D \\ \sigma(\nu-1) & \text{else} \end{cases} \tag{28}$$

Here, the parameters $\theta_D \in [0,1]$ which determines how strict the definition of stationarity should be, and $\xi \in [0,1]$ which determines the decrease rate for $S(\nu)$ need to be specified. The time scale parameter τ controls both the speed of exponential smoothing, as well as the time lag that is consider for the determination of stationarity. For the experiments of this section, we consider again the MNIST benchmark and train an energy-based ReST layer (with ZPC boundary conditions) using the control rule of eqn. (28), with fixed values of $\xi = 0.95$, $\tau = 1000$ and variable values of $\theta_D = \{0.005, 0.0075, 0.01, 0.025\}$. The other ReST parameters are as stated in Sec. 3.2, except for $K = 10$, $T = 40.000$, $t_0 = 10000$, $t_\infty = 30000$ and $S_\infty = 1$. The experimental interval was chosen a bit longer and the convergence times a bit less restrictive here to be able to well visualize the development of S over time as a consequence of the chosen control strategy. The control strategies differ only in the strictness of the definition of stationarity, and will therefore differ in their eagerness to trigger the next decrease step to $S(\nu)$. As can, be seen in Fig. 10, all of these strategies are effective in reducing energy over time, although at different convergence speeds.

4 Discussion

We presented the ReST model, a significant extension of the energy-based SOM model proposed in [12], characterized by the avoidance of periodic boundary conditions in the convolution operations associated with the model. We have shown that ReST layers behave just as SOMs would w.r.t. the topological organization of prototypes, that neural activities within can be successfully constrained to follow a log-normal distribution with specified parameters, and that ZPC boundary conditions achieve lower quantization errors than periodic boundary conditions, especially for small map sizes (see Sec. 3.5). We have furthermore shown that ReST layers can use the instantaneous energy value $E(\nu)$ as a criterion for rejecting outliers in a real-world visual classification problems, and that the evolution of energy over time can be used for implementing automated control strategies for the time-dependent ReST parameter $S(\nu)$ that can remove the need to tune these time dependencies. Overall, we can state that ZPC boundary conditions make energy-based ReST layer a tool that can be used as routinely as normal

SOMs, but with less parameters to be tuned and with the energy function as a clear criterion for convergence. Furthermore, there is no need to resort to exotic and often undesirable periodic boundary condition that in any case reduce the approximation capacity of any self-organized model (not just an energy-based one).

4.1 Why ZPC works

The ZPC technique we introduced here is based on a simple static re-weighting of convolution results that form the activity map a_{ij}, see Sec. 2.1. The reason why this approach works in ReST (but not, in general, for arbitrary images) lies in the fact that we ensure the temporal averages of all activities $\mathbf{a}_n$ to be homogeneous throughout the map. This is not the case for images, and thus the technique described here cannot be generalized to image processing. The temporal average activity of c_{ni} of any ReST unit (after convolution) is only dependent on the mass of the neighbourhood filter that falls inside the map, which is a quantity that depends only on that unit's position and on nothing else. The correction that fixes all map activities to homogeneous long-term values, here termed the re-weighting map χ can therefore, for fixed neighbourhood radius S, even be precomputed, but needs to be re-calculated each time S changes.

5 Conclusion and future work

More elaborate control strategies for time-dependent SOM parameters could very probably be devised (for example, there is a clear dependency between θ_D and τ in Sec. 3.7 that can be used to eliminate one parameter) so as to render these strategies fully automatic, removing the need for any parameter tuning during SOM training. Another, very interesting perspective, both conceptually and from an applied perspective, is the adaptation of the local neighbourhood functions $\mathbf{g}_i$. Since the $\mathbf{g}_i$ are included in the energy function, there is no reason why they should not be optimized by gradient descent as well, although a L_1 constraint would have to be imposed, and the learning rates for prototype and neighbourhood adaptation would probably have to be decoupled. In this way, outlier detection as performed in Sec. 3.6 may become much more powerful since each neuron i could consider only a learned subset of its neighbours for the computation of c_{ni} instead of performing a predefined and unspecific weighted summation. The energy function of eqn. (4) gives rise to a Hebbian-type of learning rule for the $\mathbf{g}_i$ which is in itself interesting from a biological modeling perspective. This potentially increases outlier detection capacity of the ReST model will in turn benefit our work on incremental learning algorithms [8].

6 Conflict of interest

The authors declare that they have no conflict of interest.

References

1. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016. 6
2. S. Acharya. Deep Learning Based Large Scale Handwritten Devanagari Character Recognition. 2015. 8
3. K. Bunte, S. Haase, M. Biehl, and T. Villmann. Stochastic neighbor embedding (sne) for dimension reduction and visualization using arbitrary divergences. *Neurocomputing*, 90:23 – 45, 2012. Advances in artificial neural networks, machine learning, and computational intelligence (ESANN 2011). 2
4. G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. EMNIST: Extending MNIST to handwritten letters. *Proceedings of the International Joint Conference on Neural Networks*, 2017-May:2921–2926, 2017. 8
5. M. Cottrell, J.-C. Fort, and G. Pagès. Theoretical aspects of the som algorithm. *Neurocomputing*, 21(1):119–138, 1998. 11
6. E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: ordering, convergence properties and energy functions. *Biological cybernetics*, 67(1):47–55, 1992. 1, 2
7. A. Flexer. On the use of self-organizing maps for clustering and visualization. *Intelligent Data Analysis*, 5(5):373–384, 2001. 2
8. A. Gepperth and C. Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, pages 1–11, 2016. 2, 13
9. I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 2
10. T. Graepel, M. Burger, and K. Obermayer. Self-organizing maps: Generalizations and new optimization techniques. *Neurocomputing*, 21(13):173 – 190, 1998. 2
11. T. Heskes. Energy functions for self-organizing maps. In *Kohonen maps*, pages 303–315. Elsevier, 1999. 2
12. T. M. Heskes and B. Kappen. Error potentials for self-organization. In *Neural Networks, 1993., IEEE International Conference on*, pages 1219–1223. IEEE, 1993. 1, 2, 4, 9, 11
13. B. Jähne. *Digital image processing*. Springer Verlag Berlin, Heidelberg, New York, 6 edition, 2005. 6
14. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982. 1, 2
15. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982. 4, 9
16. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Apllied to Document Recognition. 1998. 8
17. M. Lefort, T. Hecht, and A. Gepperth. Using self-organizing maps for regression: the importance of the output function. In *European Symposium On Artificial Neural Networks (ESANN)*, 2015. 5
18. L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 2
19. S.-L. Shieh and I.-E. Liao. A new approach for data clustering and visualization using self-organizing maps. *Expert Systems with Applications*, 39(15):11924 – 11933, 2012. 2
20. V. Tolat. An analysis of kohonen's self-organizing maps using a system of energy functions. *Biological Cybernetics*, 64(2):155–164, 1990. 1, 2
21. L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014. 2
22. J. Vesanto. Som-based data visualization methods. *Intelligent Data Analysis*, 3(2), 1999. 2

23. H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. pages 1–6, 2017. 8