Performance Analysis of Deep Neural Maps

Boren Zheng and Lutz Hamel

Dept. of Computer Science and Statistics
University of Rhode Island
Kingston, RI 02881–2018
boren_zheng@uri.edu, lutzhamel@uri.edu

Abstract. Deep neural maps are unsupervised learning and visualization methods that combine autoencoders with self-organizing maps. An autoencoder is a deep artificial neural network that is widely used for dimension reduction and feature extraction in machine learning tasks. The self-organizing map is a neural network for unsupervised learning often used for clustering and the representation of high-dimensional data on a 2D grid. Deep neural maps have shown improvements in performance compared to standalone self-organizing maps when considering clustering tasks. The key idea is that a deep neural map outperforms a standalone self-organizing map in two dimensions: (1) Better convergence behavior by removing noisy/superfluous dimensions from the input data, (2) faster training due to the fact that the cluster detection part of the DNM deals with a lower dimensional latent space. Traditionally, only the basic autoencoder has been considered for use in deep neural maps. However, many different kinds of autoencoders exist such as the convolutional and the denoising autoencoder and here we examine the effects of various autoencoders on the performance of the resulting deep neural maps. We investigate five types of autoencoders as part of our deep neural maps using three different data sets. Overall we show that deep neural maps perform better than standalone self-organizing maps both in terms of improved convergence behavior and faster training. Additionally we show that deep neural maps using the basic autoencoder outperform deep neural maps based on other autoencoders on non-image data. To our surprise we found that deep neural maps based on contractive autoencoders outperformed deep neural maps based on convolutional autoencoders on image data.

Keywords: autoencoder, deep neural map, self-organizing map, deep learning

1 Introduction

Deep neural maps (DNMs) [20] are unsupervised learning and visualization methods that combine autoencoders with self-organizing maps. An autoencoder (AE) is a deep artificial neural network that is widely used for dimensionality reduction and feature extraction in machine learning tasks [14]. The self-organizing map (SOM) is an artificial neural network designed for unsupervised

learning [16]. It is often used for clustering and the representation of high-dimensional data on a 2D grid. Deep neural maps have shown improvements in performance compared to standalone self-organizing maps when considering clustering tasks [22] [20].

In diverse fields such as genomic data clustering [21] and cluster analysis of massive astronomical data [15], self-organizing maps are a good clustering approach since they not only accomplish the clustering task but also provide an accessible, visual clustering representation. However, because both genomic data and astronomical data are extremely high-dimensional, convergence behaviors of SOMs tend to be very slow and erratic. In deep neural maps, an autoencoder is used to reduce the dimensionality of the original data as a preprocessing step before the training of the underlying self-organizing map begins thus improving the convergence behavior and speed of the map.

In all studies we are aware of, *e.g.* [4, 5, 20, 22], only consider a single autoencoder architecture as part of their deep neural maps. Here, we investigate five types of autoencoders as part of our deep neural maps using three different data sets illuminating the performance characteristics of the various autoencoders. The five autoencoder architectures we investigate are: (1) basic (2) sparse, (3) contractive, (4) denoising, and (5) convolutional. Here autoencoder architectures (2), (3), and (4) can be considered regularized versions of the basic autoencoder architecture (1). Architecture (5), the convolutional autoencoder, is based on convolutional deep neural networks commonly used for image processing [18]. The data sets we used in our analysis are the digits data set derived from the MNIST database [19], the landsat data set [24], and a synthetic data set that contains 16 well-defined clusters in 64-dimensional space.

We show that DNMs improve the performance of standalone SOMs along two dimensions: (1) DNMs improve convergence behavior by removing noisy/superfluous dimensions from the input data. The improved convergence behavior can be observed by superior cluster homogeneity and convergence accuracy scores. (2) DNMs train faster due to the fact that the cluster detection part of the DNM deals with a lower dimensional latent space.

It is perhaps a surprise that for non-image data a DNM with a basic autoencoder outperforms all others. For image data we found that a DNM with a contractive autoencoder performs best and not a DNM with a convolutional autoencoder as one would expect.

The remaining sections of this paper are organized as follows. We give brief overviews of self-organizing maps, autoencoders, and the design of our deep neural maps in Section 2, Section 3, and Section 4, respectively. In Section 5 we detail our experiments, describe our data sets, and explain the evaluation methods we used. We discuss our results in Section 6. Finally, We summarize and propose future work in Section 7.

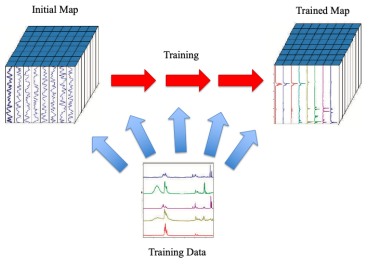

Fig. 1. Training a SOM.

2 Self-Organizing Maps

Self-organizing maps were introduced by Kohonen in the 1980's as a way to visualize high-dimensional data on a 2-D grid [16]. The 2-D grid consists of high-dimensional neurons where the dimensionality of each neuron matches the dimensionality of the training data. Figure 1 illustrates the training process of a SOM. In the initial map, the neurons are initialized with small random values and as the training data is repeatedly applied to the map the neurons are starting to take on the form of the training data. What is particularly interesting is that certain regions of the map become sensitized to certain traits in the training data. Figure 4 shows typical 2-D starburst visualizations of the final neuron map. The starbursts represent clusters in the high-dimensional training data space.

The basic SOM training algorithm can be summarized as follows [11]:

1. *Initialization*: initialize each neuron weight vector $\mathbf{m}_i$ with small random values.
2. *Selection step*: select a training data vector $\mathbf{x}_k$ from the training data.
3. *Competitive step*: find the best matching neuron $\mathbf{m}_c$ based on the Euclidean distance between the training data vector $\mathbf{x}_k$ and the neurons $\mathbf{m}_i$ on the map:
$$c = \arg\min_i(\|\mathbf{m}_i - \mathbf{x}_k\|). \tag{1}$$
4. *Update step*: update the winning neuron's $\mathbf{m}_c$ neighborhood using the following rule:
$$\mathbf{m}_i \leftarrow \mathbf{m}_i - \eta(\mathbf{m}_i - \mathbf{x}_k)h(c,i) \tag{2}$$
where $\eta(\mathbf{m}_i - \mathbf{x}_k)$ denotes the difference between a neuron and the training instance scaled by the learning rate $0 < \eta < 1$, $h(c,i)$ denotes the following loss function:
$$h(c,i) = \begin{cases} 1 & \text{if } i \in \Gamma(c), \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$
where $\Gamma(c)$ is the neighborhood of the best matching neuron m_c.

Repeat steps 2, 3, and 4 until the map has converged.

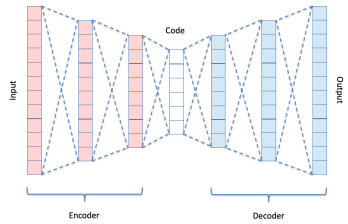

Fig. 2. The basic autoencoder architecture.

3 Autoencoders

Autoencoders are deep neural networks that conceptually consist of three parts: (a) the encoder, (b) the neurons representing the latent space encoding the compressed/encoded information, and (c) the decoder. Figure 2 shows the architecture of a basic autoencoder. Each colored column represents a layer of neurons in this illustration. Observe that both the encoder and the decoder are deep multilayer neural networks. In this basic architecture the encoder maps the input into the hidden layer representing the latent space and the decoder reconstructs the input from this hidden layer representation. An autoencoder where the latent space is of lower dimensionality than the input space is called undercomplete and we call an autoencoder for which the converse is true overcomplete. Regularization can be used to prevent autoencoders from simply copying information from the input to the latent space without learning anything useful [8].

The Basic Autoencoder: Let $\phi : X \rightarrow F$ represents the encoder part of a basic autoencoder (AE) as shown in Figure 2 that maps the input space X into a latent space F. Also, let $\psi : F \rightarrow X$ be the decoder part of a basic autencoder that maps the latent space F into the input space X. Training a basic autoencoder can now be understood as the optimization problem,

$$\arg\min_{\phi,\psi} L\big(\mathbf{x}, (\psi \circ \phi)\mathbf{x}\big), \forall \mathbf{x} \in X. \tag{4}$$

That is, we want to find neural networks ϕ and ψ for the en- and decoders, respectively, that minimize the loss L between the original input and the reconstructed input available as output from the decoder. Individual layers ϕ_k and ψ_l in the en- and decoder networks, respectively, can be written as the equations,

$$\begin{aligned} \phi_k(\mathbf{h}_{k-1}) &= \sigma(\mathbf{h}_{k-1} \bullet \mathbf{W}_k) = \mathbf{h}_k \\ \psi_l(\mathbf{h}_{l-1}) &= \sigma(\mathbf{h}_{l-1} \bullet \mathbf{U}_l) = \mathbf{h}_l \end{aligned} \tag{5}$$

where $\bullet$ is the dot product extended to matrices, σ is the activation function, and the matrices $\mathbf{W}_k$ and $\mathbf{U}_l$ are the weight matrices of the corresponding layers. For the input layer of the encoder network we then have,

$$\phi_0(\mathbf{x}) = \sigma(\mathbf{x} \bullet \mathbf{W}_0) = \mathbf{h}_0 \tag{6}$$

with $\mathbf{x} \in X$ and $\mathbf{h}_0$ an intermediate network representation of the input. For the output layer p of the encoder network we have,

$$\phi_p(\mathbf{h}_{p-1}) = \sigma(\mathbf{h}_{p-1} \bullet \mathbf{W}_p) = \mathbf{h}_p \tag{7}$$

where $\mathbf{h}_p \in F$ is the representation of some point $\mathbf{x} \in X$ in latent space F. That is,

$$\phi(\mathbf{x}) = (\phi_p \circ \ldots \circ \phi_1 \circ \phi_0)\mathbf{x} = \mathbf{h}_p. \tag{8}$$

Similarly for the decoder network we have,

$$\psi(\mathbf{h}_p) = (\psi_q \circ \ldots \circ \psi_{p+2} \circ \psi_{p+1})\mathbf{h}_p \approx \mathbf{x}. \tag{9}$$

with $q > p$. This optimization problem can be solved using a deep neural network library such as Keras [1].

The Sparse Autoencoder: A sparse autoencoder (SAE) only has a small number of nodes that are activated in the hidden code layer at any particular time [2]. The objective function of an SAE is the objective function of the basic AE plus a sparsity penalty term,

$$L\big(\mathbf{x}, (\psi \circ \phi)\mathbf{x}\big) + \Omega(\mathbf{h}) \tag{10}$$

where ϕ and ψ denote the encoder and decoder networks, respectively, and $\mathbf{h} = (h_1, h_2, \ldots, h_n)$ with n the dimensionality of the latent space is the output vector of output layer of the encoder network defined as $\phi(\mathbf{x}) = \mathbf{h}$. The sparsity penalty term is defined as,

$$\Omega(\mathbf{h}) = \lambda \sum_{i}^{n} |h_i|, \tag{11}$$

where λ is a hyperparameter of the autoencoder model [8].

The Denoising Autoencoder: The denoising autoencoder (DAE) does not directly add a regularization term to the objective function in order to avoid overfitting but instead adds noise to the input signal before the input is applied to the encoder network. The trick is that the output of the decoder is compared to the original signal rather than the noisy input and therefore the network has to learn to ignore noise thereby preventing it from overfitting.

Let $\hat{\mathbf{x}} = \omega(\mathbf{x})$ for all $\hat{\mathbf{x}}, \mathbf{x} \in X$. Here $\mathbf{x}$ denotes an original training instance and $\hat{\mathbf{x}}$ represents the original instance with noise added by process ω. We can now rewrite our objective function for the denoising autoencoder,

$$L\big(\mathbf{x}, (\psi_{\theta'} \circ \phi_\theta)\omega(\mathbf{x})\big), \tag{12}$$

where θ and θ' are additional parameters on the encoder and decoder networks, respectively, that are trained to minimize the average reconstruction error over the training set [8, 25].

The Contractive Autoencoder: The contractive autoencoder (CAE) adds a regularization term to the loss function of the basic autoencoder based on the Frobenius norm of the Jacobian matrix of the features of the latent space [23].

This regularization term will make the autoencoder more robust to perturbations of the input and is encouraged to contract the input neighborhood to a smaller output neighborhood [8]. The objective function for the CAE is,

$$L\big(\mathbf{x}, (\psi \circ \phi)\mathbf{x}\big) + \lambda \|J_f(\mathbf{x})\|_F^2, \tag{13}$$

where the regularization term is defined as,

$$\|J_f(\mathbf{x})\|_F^2 = \sum_{ij}^{n} \left(\frac{\partial h_j}{\partial x_i} \right)^2. \tag{14}$$

Additionally, as before $\mathbf{h} = (h_1, h_2, \ldots, h_n)$ with n the dimensionality of the latent space is the output vector of the hidden code layer defined as $\phi(\mathbf{x}) = \mathbf{h}$.

The Convolutional Autoencoder: A convolutional autoencoder (ConvAE) is built with convolutional layers rather than fully connected layers and hence tend to be well suited for image data sets. Here, single encoder and decoder layers are defined, respectively, as follows [9]:

$$\begin{aligned} \phi_k(\mathbf{h}_{k-1}) &= \sigma(\mathbf{h}_{k-1} * \mathbf{W}_k) = \mathbf{h}_k \\ \psi_l(\mathbf{h}_{l-1}) &= \sigma(\mathbf{h}_{l-1} * \mathbf{U}_l) = \mathbf{h}_l \end{aligned} \tag{15}$$

With respect to the basic autoencoder in Section 3 above the only thing that has changed is that the dot product operation has been replaced with a convolution operator $*$. Our remarks on network composition from above also apply here with,

$$\begin{aligned} \phi(\mathbf{x}) &= (\phi_p \circ \ldots \circ \phi_1 \circ \phi_0)\mathbf{x} = \mathbf{h}_p, \\ \psi(\mathbf{h}_p) &= (\psi_q \circ \ldots \circ \psi_{p+2} \circ \psi_{p+1})\mathbf{h}_p \approx \mathbf{x}. \end{aligned} \tag{16}$$

Where $\mathbf{x} \in X$, $\mathbf{h}_p \in F$, and q is the number of layers in the autoencoder with $p < q$.

The underlying optimization problem is to minimize the mean squared error between the input and output over all samples [9]:

$$\arg\min_{\phi,\psi} \frac{1}{|X|} \sum_{\mathbf{x} \in X} \|\mathbf{x} - (\psi \circ \phi)\mathbf{x}\|^2. \tag{17}$$

where $\mathbf{x} \in X$ represents the set of training instances in input space and $|X|$ the number of training instances.

4 Deep Neural Maps

The general architecture of our deep neural maps (DNMs) is shown in Figure 3. It consists of an autoencoder and a self-organizing map. The idea is that the autoencoder maps the input data into a latent space and the SOM is trained using this latent space. Pesteie, Abolmaesumi and Rohling have shown that this

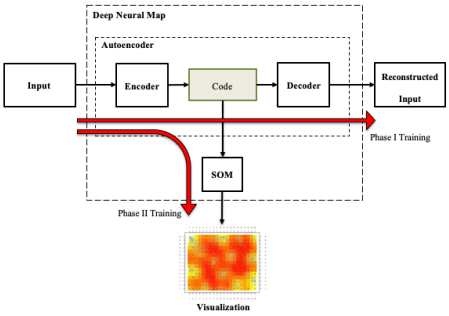

Fig. 3. Our deep neural map architecture.

architecture performs well in their experiments [20]. Similarly, Rajashekar [22] proposed an autoencoder based self-organizing map framework that uses a basic autoencoder with two hidden layers.

Training of DNMs consists of two phases shown with the red arrows in Figure 3. Phase I consists of training the autoencoder which in turn consists of solving the optimization problems discussed in the previous section. Training autoencoders is fairly fast. It takes about 200 epochs of batch size 128 to fully train the autoencoders discussed here. This is very fast compared to training a fully converged SOM which typically takes in the order of tens of thousands of iterations. Furthermore, training time of the autoencoder is amortized over the SOM model evaluation steps. Phase II is the training of the SOM using the latent space mapping of the input data.

5 Experiments

We evaluate the performance of our deep neural maps with the various different encoders on three different real-world and synthetic data sets.

Data Sets: For our experiments we used the following data sets:

(1) The **dim064** [7] [6] is a 64-dimensional synthetic data set with 1024 observations that are well separated into 16 Gaussian clusters. We split the data set as follows: 60% data for training (614 instances) and 40% data for testing (410 instances). (2) The **landsat** satellite data set from the UCI machine learning repository [3] is a real-world data set with 6435 instances and 36 attributes and 6 classes. The data set consists of the multi-spectral values of pixels in a satellite images together with their classifications. The training set contains 4435 instances, and the test set contains 2000 instances, 500 of which were used for phase II training of the DNMs. (3) The **digits** database from the UCI machine learning repository [3] is derived from the MNIST database and represents handwritten digits as 8×8 pixel images [26]. The 1797 images are stored as vectors with 64 features. For our purposes we split this data set into a training set (1437 instances) and a test set (360 instances).

Model Evaluation and Selection: As mentioned before, training a deep neural map consists of training two different models. One for the autoencoder and one for the self-organizing map.

For the autoencoders our model selection criterion was the test loss error (or reconstruction error). We chose a model architecture and the number of epochs to train a model based on minimizing that loss. In this research we found that all our autoencoder models had properly converged after 200 epochs.

Once an autoencoder has been properly trained we used its output to train the self-organizing map part of a deep neural map. Here we used the convergence accuracy [10] as a model selection criterion. For this research we selected the model with the highest convergence accuracy. For the current work we trained 40 models for each deep neural map with varying numbers of training iterations in order to compute the learning curve and select an appropriate model.

A fully trained deep neural network can then be used to produce a cluster representation of the input data on a 2D map. A typical DNM cluster presentation is shown in Figure 4. It is a heat map where deep red colors represent cluster borders and yellow/white areas represent cluster centers. The starburst graphic overlay emphasizes the cluster structure of the data [12].

Remarks on Self-Organizing Map Architectures: One of the most important hyper-parameters for SOMs is the size of the map. Here we use the following rule of thumb: *There should be as least as many neurons on the map as there are observations in the training data.* Since phase II training data consists of roughly 500 observations for all three experiments we chose a map size of 25×20 for all experiments. Another hyper-parameters is the learning rate. Here we set the learning rate fairly aggressively at .6. Finally, the POPSOM package [13] uses a constant neighborhood which contracts over the duration of the training phase.

Remarks on Autoencoder Architectures: Our basic autoencoder was implemented using a single fully-connected layer as encoder and as decoder. We added an L1 regularizer to the basic AE in order to obtain the SAE. The CAE used the same architecture as the SAE except that we used a different penalty term. We implemented the penalty term of Equation 13 as,

$$\|J_f(\mathbf{x})\|_F^2 = \sum_{ij} \left(\frac{\partial h_j}{\partial x_i} \right)^2 = \sum_j [h_j(1-h_j)]^2 \sum_i \left(\mathbf{W}_{ji}^T \right)^2 \tag{18}$$

where as before $\mathbf{x} \in X$, $\mathbf{h} = (h_1, \ldots, h_n) \in F$ is the input representation in latent space, and $\mathbf{W}$ is the weight matrix of the encoder layer [17].

We set the noise factor to be 0.5 to create noisy input for the DAE. For the models of dim064 and landsat data sets, both the encoded layer and the decoded layer of the DAE are single fully-connected layers. For the digits data set, we implemented the model as a Denoising Convolutional Autoencoder (DCAE). The encoder consists of three 2D convolutional layers followed by down-sampling layers (pooling size 2×2) and a flatten layer (encoded layer). The decoder consists of four 2D convolutional layers followed by three up-sampling layers (size 2×2), the last convolutional layer is the decoded layer.

Table 1. DNM performance with different AEs on the dim064 data set.

	SOM	DNM(AE)	DNM(SAE)	DNM(CAE)	DNM(DAE)	DNM(ConvAE)
homog.	0.92	1.00	0.83	1.00	0.98	0.93
nclust	15	16	13	16	16	15
time (sec)	20.75	1.18	0.52	0.68	1.82	1.17
conv.	0.78	0.87	0.5	0.53	0.66	0.99
dim	64	11	7	9	12	8

We utilized 1D convolutional layers, 1D max-pooling layers, and 1D upsampling layers to build the ConvAE models for the dim064 and the landsat data sets. For the model of the digits data set, the architecture of the ConvAE is the same as DCAE. However, it uses the original data as input rather than noisy data.

One more note, further dimension reduction was achieved by dropping columns in the latent space representation that were all zeros.

6 Results

We look at the performance of DNMs with different autoencoder architectures compared to standalone SOMs for each of our data sets. Our performance analysis uses five dimensions in order to describe the performances for both standalone SOMs and DNMs:

- **homog.** – Average homogeneity of the detected clusters. This is defined as

$$homog = \frac{1}{n} \sum_{c} l_c \tag{19}$$

  where l_c is the number of majority label instances in cluster c and n is the total number of observations in the training data set.
- **nclust** – The number of clusters detected.
- **time** – Phase II training time in CPU seconds (total training time for standalone SOM).
- **conv.** – Phase II convergence accuracy (convergence accuracy for standalone SOM).
- **dim** – Dimensions of the latent space (or in the case of the SOM it is the dimensionality of the original space).

Using these criteria we validated the key advantages of deep neural maps over self-organizing maps:

1. A better clustering behavior due to the removal of noisy and/or superfluous dimensions in the input data.
2. Faster training times due to the lower dimensionality of the latent space.

Performance Analysis for Dim064: Table 1 shows the typical performance values of the standalone SOM and the phase II performances of our various DNMs when trained with the dim064 data set. We can see that the SOM has

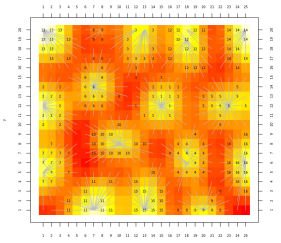

Fig. 4. The graphical output of the DNM(AE) for the dim064 data set.

an average homogeneity of .92, detected 15 clusters, took 20.75secs to converge with a convergence accuracy of .78, and was trained with 64-dimensional data. Now, in order to find the best DNM model we have to find a DNM model that:

1. Maximizes homogeneity,
2. minimizes the number of clusters,
3. minimizes training time,
4. maximizes convergence accuracy, and
5. uses the smallest number of dimensions.

Keep a couple of constraints in mind when looking for the best model: Maximizing homogeneity is more important than minimizing the number of clusters (to a certain degree - we can always achieve a homogeneity of 1 if we treat each point as a cluster) and producing better values in all the other dimensions is more important than reducing the number of dimensions.

Applying these criteria we find that the deep neural map with the basic autoencoder, DNM(AE), performs best with DNM(CAE) coming in as a close second. The DNM(AE) has a homogeneity of 1, finds all 16 clusters in the data set using a 11-dimensional latent space with a convergence accuracy of 0.87. Phase II training is about 17 times faster than training the standalone SOM.

This validates our first point above. The DNM(AE) has a better clustering behavior than the standalone SOM: A homogeneity value of 1 compared to .92 in the standalone SOM and a convergence accuracy of .87 compared to .78 in the standalone SOM. Furthermore DNM(AE) phase II training is about 17 times faster than training the standalone SOM validating our second point above.

Figure 4 shows the graphic output of the DNM(AE). The 16 homogeneous clusters are clearly visible under the starbursts and we have mapped the labels of the training instances on top of the map.

Performance Analysis for Landsat: Table 2 displays typical performance data for the landsat data set. Here we find that the standalone SOM detected 10 clusters with an average homogeneity of .78 and a convergence accuracy of .95. It took the standalone SOM 4.83 seconds to train on the 36 dimensional input data.

Applying the same analysis from the previous section to the performance of the phase II training of our various DNM architectures we again find the

Table 2. DNM performance with different AEs on the landsat data set.

	SOM	DNM(AE)	DNM(SAE)	DNM(CAE)	DNM(DAE)	DNM(ConvAE)
homog.	0.78	0.78	0.79	0.77	0.52	0.74
nclust	10	10	9	10	8	9
time (sec)	4.83	0.67	0.54	1.1	1.2	0.93
conv.	0.95	0.98	0.5	0.99	0.54	0.99
dim	36	3	2	3	2	5

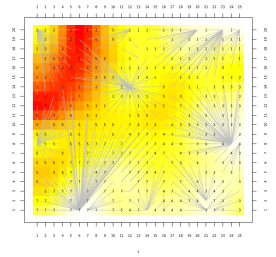

Fig. 5. The graphical output of the DNM(AE) for the landsat data set.

DNM(AE) is the front runner. We chose it over the DNM(CAE) based on the fact that it trains about twice as fast and has a slightly higher average homogeneity. When comparing the phase II training of DNM(AE) to the standalone SOM we find that it also detects 10 clusters with an average homogeneity of .78. However, its convergence accuracy of .98 is higher than the .95 of the standalone SOM and that it trains on 3-dimensional data compared to the 36 dimensions in the standalone. Most notably is that the phase II training of DNM(AE) runs about 7 times faster than training the standalone SOM.

We can observe again that our two evaluation criteria for DNMs are fulfilled. The DNM(AE) trains faster and has a higher convergence accuracy than the standalone SOM.

Figure 5 shows the map produced by our DNM(AE) for this data set. The clusters are clearly visible under the starbursts. What is interesting here is that the cluster with label 2 is set apart from all the other clusters.

Performance Analysis for Digits: Table 3 shows the performance numbers for our digits data set. Here the deep neural map with the contractive autoencoder, DNM(CAE), is clearly the winner. It has the highest average cluster homogeneity of .78, detects a reasonable number of clusters, trains about 11 times faster than the standalone SOM, and has a convergence accuracy that is about twice that of the standalone SOM. The latent space for this DNM has 12 dimensions.

Therefore, we can observe once more that our two evaluation criteria are fulfilled: The DNM(CAE) trains faster than the standalone SOM and has a better convergence behavior in terms of higher homogeneity and convergence accuracy scores than the standalone SOM.

Table 3. DNM performance with different AEs on the digits data set.

	SOM	DNM(AE)	DNM(SAE)	DNM(CAE)	DNM(DCAE)	DNM(ConvAE)
homog.	0.65	0.64	0.68	0.78	0.54	0.62
nclust	14	13	14	14	12	10
time (sec)	8.36	0.74	4.35	0.75	0.58	1.44
conv.	0.48	0.94	0.93	0.95	0.95	0.96
dim	64	12	12	12	5	11

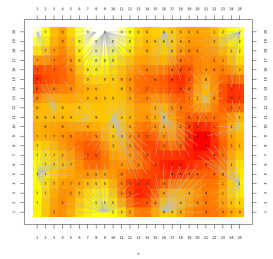

Fig. 6. The graphical output of the DNM(AE) for the digits data set.

It is a bit disappointing that none of the models detected the ten clusters due to the ten digits in the data set. However, the data set seems very noisy and the clusters that were found seem to be a reasonable approximation to the ideal clusters. Figure 6 shows the map produced by the DNM(CAE). What is perhaps noteworthy is that the digit 0 has the strongest cluster towards the upper left corner of the map. This is perhaps due to the fact that it is the most recognizable and unique digit compared to all the other digits.

Observations: In all three of our data sets we found that DNMs outperform standalone SOMs by training faster and exhibiting a better convergence behavior as defined by the average cluster homogeneity and convergence accuracy scores. To our surprise we found that deep neural maps with a convolutional autoencoder did not perform well on the digit image data. One way of interpreting this might be that the latent space here is geared towards reconstructability rather than preserving features for clustering. In order to remedy this one would have to extend the objective function to include a clustering term which would penalize latent spaces that do not take preserving features for clustering into account.

7 Conclusions and Further Work

Deep neural maps are unsupervised learning and visualization methods that combine autoencoders with self-organizing maps. Recent work has shown that deep neural maps outperform standalone self-organizing maps when considering clustering tasks. The key idea is that a deep neural map outperforms a standalone self-organizing map in two dimensions: (1) Better convergence behavior by removing noisy/superfluous dimensions from the input data, (2) faster training due to the fact that the cluster detection part of the DNM deals with a lower

dimensional latent space. However, many different kinds of autoencoders exist such as the convolutional and the denoising autoencoder and here we examined the effects of various autencoders on the performance of the resulting deep neural maps. We investigated five types of autoencoders as part of our deep neural maps using three different data sets. Overall we show that deep neural maps perform better than standalone self-organizing maps both in terms of improved convergence behavior and faster training. Additionally we show that deep neural maps using the basic autoencoder outperform deep neural maps based on other autoencoders on non-image data. To our surprise we found that deep neural maps based on contractive autoencoders outperformed deep neural maps based on convolutional autoencoders on image data.

Given the results from this limited study we would choose the basic autoencoder as our autoencoder of choice in order to construct deep neural maps geared towards non-image data corroborating earlier work done by Rajashekar [22]. We would like to develop a DNM R package for use by the non-deep learning specialists.

We need to further investigate why our DNM based on a convolutional autoencoder failed to deliver good results on image data. As we mentioned above, the first step is to investigate extending the relevant objective function with a clustering term.

References

1. F. Chollet. Keras, GitHub. https://github.com/fchollet/keras, 2015.
2. Pedro Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. The master algorithm: How the quest for the ultimate learning machine will remake our world. Basic Books, New York, NY, US, 2015.
3. D. Dua and E. Karra Taniskidou. "UCI Machine Learning Repository," Irvine, CA: University of California, School of Information and Computer Science,, 2019.
4. Christos Ferles, Yannis Papanikolaou, and Kevin J Naidoo. Denoising autoencoder self-organizing map (dasom). *Neural Networks*, 105:112–131, 2018.
5. Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. Som-vae: Interpretable discrete representation learning on time series. *arXiv preprint arXiv:1806.02199*, 2018.
6. P. Franti, O. Virmajoki, and V. Hautamaki. Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1875–1881, November 2006.
7. Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets. *Applied Intelligence*, 48(12):4743–4759, December 2018.
8. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.
9. Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep Clustering with Convolutional Autoencoders. In Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy, editors, *Neural Information Processing*, Lecture Notes in Computer Science, pages 373–382. Springer International Publishing, 2017.

10. Lutz Hamel. SOM Quality Measures: An Efficient Statistical Approach. In Erzsébet Merényi, Michael J. Mendenhall, and Patrick O'Driscoll, editors, *Advances in Self-Organizing Maps and Learning Vector Quantization*, Advances in Intelligent Systems and Computing, pages 49–59, Cham, 2016. Springer International Publishing.
11. Lutz Hamel. VSOM: Efficient, Stochastic Self-organizing Map Training. In Kohei Arai, Supriya Kapoor, and Rahul Bhatia, editors, *Intelligent Systems and Applications*, volume 869, pages 805–821. Springer International Publishing, Cham, 2019.
12. Lutz Hamel and Chris W Brown. Improved interpretability of the unified distance matrix with connected components. In *Proceedings of the International Conference on Data Mining (DMIN)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2011.
13. Lutz Hamel, Benjamin Ott, Gregory Breard, Robert Tatoian, and Vishakh Gopu. popsom: Functions for Constructing and Evaluating Self-Organizing Maps, June 2019.
14. G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.
15. Woncheol Jang and Martin Hendry. Cluster analysis of massive datasets in astronomy. *Statistics and Computing*, 17(3):253–262, September 2007.
16. Teuvo Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer-Verlag, Berlin Heidelberg, 3 edition, 2001.
17. Agustinus Kristiadi. Deriving Contractive Autoencoder and Implementing it in Keras - Agustinus Kristiadi's Blog.
18. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
19. Y. LeCun and C. Cortes. MNIST handwritten digit database, 2010.
20. Mehran Pesteie, Purang Abolmaesumi, and Robert Rohling. Deep Neural Maps. *arXiv:1810.07291 [cs, stat]*, October 2018. arXiv: 1810.07291.
21. K. S. Pollard and M. J. van der Laan. Cluster Analysis of Genomic Data. In Wing Wong, M. Gail, K. Krickeberg, A. Tsiatis, J. Samet, Robert Gentleman, Vincent J. Carey, Wolfgang Huber, Rafael A. Irizarry, and Sandrine Dudoit, editors, *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, pages 209–228. Springer New York, New York, NY, 2005.
22. Deepthi Rajashekar. One-class learning with an Autoencoder Based Self Organizing Map. March 2017.
23. Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.
24. Compton J Tucker, Denelle M Grant, and Jon D Dykstra. Nasa's global orthorectified landsat data set. *Photogrammetric Engineering & Remote Sensing*, 70(3):313–322, 2004.
25. Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.
26. Lei Xu, Adam Krzyzak, and Ching Y Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22(3):418–435, 1992.