OPTIMIZING CONNECTIVITY THROUGH NETWORK GRADIENTS FOR THE RESTRICTED BOLTZMANN MACHINE

Anonymous authors
Paper under double-blind review

ABSTRACT

Leveraging sparse networks to connect successive layers in deep neural networks has recently been shown to provide benefits to large scale state-of-the-art models. However, network connectivity also plays a significant role on the learning performance of shallow networks, such as the classic Restricted Boltzmann Machines (RBM). Efficiently finding sparse connectivity patterns that improve the learning performance of shallow networks is a fundamental problem. While recent principled approaches explicitly include network connections as model parameters that must be optimized, they often rely on explicit penalization or have network sparsity as a hyperparameter. This work presents a method to find optimal connectivity patterns for RBMs based on the idea of network gradients (NCG): computing the gradient of every possible connection, given a specific connection pattern, and using the gradient to drive a continuous connection strength parameter that in turn is used to determine the connection pattern. Thus, learning RBM parameters and learning network connections is truly jointly performed, albeit with different learning rates, and without changes to the objective function. The method is applied to the MNIST and other datasets showing that better RBM models are found for the benchmark tasks of sample generation and input classification. Results also show that NCG is robust to network initialization, both adding and removing network connections while learning.

1 INTRODUCTION

While most neural network architectures adopt a fully connected network between units of successive layers, it has been long recognized that network connectivity plays a fundamental role in the model, not only reducing the number of parameters but also leading to a more accurate model or to faster learning (Reed, 1993; Blalock et al., 2020). This finding has recently reemerged in the context of deep neural networks, and while classic architectures such as ResNet (He et al., 2016) and BERT (Devlin et al., 2019) have millions of parameters that must be learned, recent works indicate that only a small fraction is necessary for the model to attain a similar performance under an equivalent training effort (Blalock et al., 2020).

Most works on leveraging network connectivity to improve the model focus on deep neural networks or large scale networks. However, connectivity patterns play a fundamental role even on simple two-layer networks such as Restricted Boltzmann Machines (RBMs). While the reduction on the absolute number of parameters may be small, recent works observe that an effective connectivity pattern can still yield superior learning curves, learning faster and better (Mocanu et al., 2018; de Oliveira & Figueiredo, 2022). In fact, the connectivity of an RBM can be interpreted as a hyperparameter that influences its performance, just as the number of neurons (another hyperparameter) in its hidden layer (Fischer & Igel, 2014; Côté & Larochelle, 2016).

Finding the best network connectivity for a given neural network is not a trivial problem, given its dependence on the input (training data), the discrete nature of the connections, and the exponentially large space of possible connection patterns (there are 2^{n^2} different networks between two layers with n units each). A common approach to tackle this problem is to construct the connectivity pattern while training the network, starting with a dense network and using some pruning strategy to remove

connections in a sequence of rounds (train and prune) (Han et al., 2015; Frankle & Carbin, 2019). A less explored yet more principled approach is to explicitly include the network connections as parameters that must be optimized in the model (Savarese et al., 2020; Chen et al., 2021; Zhou et al., 2021). Intuitively, the model should jointly learn the optimal network weights and network connections during training.

However, the discrete nature of the network connections poses a challenge to widely used continuous optimization frameworks such as (stochastic) gradient descent, since discrete connection parameters have no derivatives (and thus, no gradient). To circumvent this problem, recent approaches adopt continuous variables and functions to represent network connections. Moreover, in order to drive the model towards sparse networks, many incorporate an explicit penalization term (representing the number of connections) into the objective function. Also, the discrete network connectivity is often only determined at the end of training (or a round), and does not evolve jointly with the optimization of other parameters. In contrast, this work proposes a novel method tailored to RBMs based on the notion of "network gradients".

In a nutshell, the Network Connectivity Gradient (NCG) method computes the gradient for every possible network connection for any given connectivity pattern. Moreover, NCG uses a continuous parameter to represent the strength of every possible network connection which is updated according to the gradient as any other model parameter. Finally, the network strength is thresholded to yield a discrete connectivity pattern *during* optimization (i.e., at each training iteration) which in turn determines how information (probabilities) and gradients flow on the model during training.

Intuitively, the network gradient indicates the relevance of each possible connection given the current connection pattern. This gradient drives the connection strength parameter which in turn determines if a connection should be present or absent, effectively adjusting the connection pattern as the model is trained. Thus, if the initial connectivity pattern is too sparse or too dense, NCG will enable or disable connections early during training, respectively. In essence, NCG truly learns the network connectivity jointly with other RBM parameters, albeit with possibly different learning rates. Note that no changes are required to the objective function of the RBM.

Beyond proposing NCG, this work evaluates the method using the MNIST data set on two orthogonal tasks often used to assess RBMs: sample generation (average NLL is the performance metric) and input classification (accuracy is the performance metric). For the classification task, two other data sets from the UCI Evaluation Suite (Dua & Graff, 2017) are considered. In both tasks NCG shows a superior learning curve, both learning faster and more accurately than a classic fully connected RBM. The evaluation also shows that NCG removes and adds network connections during training, indicating its effectiveness in searching for optimal network patterns and robustness to initialization. Comparison with static patterns and the SET method (Mocanu et al., 2018) (also designed for optimizing the network connectivity of RBMs) indicate the superiority of NCG, especially during the early phases of training.

The remainder of this work is organized as follows: Section 2 has a cursory discussion of related works; Section 3 imparts a brief explanation of the RBM; Section 4 presents the NCG method proposed in this work; Section 5 shows the experimental results; and Section 6 has the concluding remarks.

2 Related Work

Recent developments in Neural Architecture Search (NAS) focus on the design of effective network architectures for deep neural networks targeted to solve a given task. The design spaces often consider the structure of layers through which information flows as well as the type of operation (aggregation/activation) applied by each layer (Elsken et al., 2019). Optimization problems that consider different design spaces for the network can be formulated and solved using different techniques (Liu et al., 2019; Fang et al., 2020). However, the focus of such approaches is on the macro scale organization of the network, and not on the fine connectivity pattern of consecutive layers (other than a small set of pre-defined connectivity patterns, such as different convolutions).

Independently of NAS, the idea of removing (pruning) connections between two adjacent network layers has recently reemerged in the context of deep neural networks. Pruning can improve the model's learning curve (learning faster or better) and drastically reduce the number of model parameters (Reed,

1993; Blalock et al., 2020; Liang et al., 2021). Finding the optimal connectivity pattern for two adjacent layers is not a trivial task. Most approaches start with dense networks and iterate in rounds of training the model parameters and using the parameter values (and the input samples) to prune network connections. Pruning can also be performed before training starts (Lee et al., 2019; de Jorge et al., 2021).

A more principled yet less explored approach explicitly includes the network connectivity as a parameter of the model, making the network connectivity a part of the optimization problem. This often requires increasing the number of parameters and modifying the objective function to induce pruning. A prominent example is Continuous Sparsification (Savarese et al., 2020), that uses continuous parameters and continuous functions to approximate the discrete nature of network connections, and adds a penalization term to the objective function. The discrete network connectivity is determined at the end of training rounds. UGS (Chen et al., 2021) deploys a similar approach tailored to Graph Neural Networks (GNN). DNW (Wortsman et al., 2019) does not add penalization to the objective function, and keeps k edges with the largest weight magnitude, discovering good sparse subnetworks in predefined architectures. Another recent approach is SR-STE (Zhou et al., 2021) where the (discrete) connectivity pattern is updated at each training iteration. However, the method assumes that each input unit is connected to exactly k output units and thus sparsity is predefined (k is a hyperparameter).

All prior works above focus on deep neural networks. However, network connectivity also plays a fundamental role on simple two-layer networks, including the Restricted Boltzmann Machine (RBM), a principled and probabilistic model that has been widely explored and applied in literature (Fischer & Igel, 2014; Decelle & Furtlehner, 2021). RBMs' hyperparameters have a significant impact on the model's performance which has prompted different methods to best choose them given a task and input data (Hinton, 2012; Papa et al., 2015; Côté & Larochelle, 2016). A prominent example is the infinite RBM (Côté & Larochelle, 2016), a variation where the number of hidden units (a hyperparameter in the classic model) is an explicit model parameter that is determined during training.

The connectivity between layers have also been investigated in shallow networks. For example, a recent work showed that manually crafted connectivity patterns can lead to significantly better learning performance on RBMs (de Oliveira & Figueiredo, 2022). Moreover, the Sparse Boltzmann Machines (SBM) is a model where the connectivity is a sparse two-layer tree-like network (Chen et al., 2017). Different tree-like networks can be learned from data and then used as hyperparameters when training the RBM, generating models that are less likely to overfit and that have better interpretability with respect to the dense RBM. In a more recent work, the network connectivity of the RBM is learned during training along with other model parameters. Their approach (called SET method) removes connections with the smallest weights and adds the same number of randomly chosen new connections at each training round Mocanu et al. (2018). Thus, network sparsity is predefined (a hyperparameter). In contrast, NCG (to be presented) learns the connectivity and sparseness during training using the gradients of the unmodified objective function of the model. This is the first method of this kind specifically designed for RBMs, to the best of our knowledge.

3 The Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM), first proposed under the name *Harmonium* (Smolensky, 1986), is an energy-based model for unsupervised learning. It is a classic neural network model that has been applied to a number of different tasks. While initially designed for sample generation (Roux et al., 2011; Decelle & Furtlehner, 2021; Tang et al., 2012), it has been used for classification tasks (Tieleman, 2008; Larochelle et al., 2012) and also as preprocessing method for downstream tasks (Midhun et al., 2014). Furthermore, RBMs are the building blocks of Deep Belief Networks (DBN), which find numerous applications in modern problems (Salakhutdinov & Murray, 2008; Qiang et al., 2020).

An RBM is a probabilistic model composed of two layers of binary units: one visible $\mathbf{x}$ of size X, representing the data, and one hidden (or latent) $\mathbf{h}$ of size H, that extracts characteristics and increases learning ability. The two layers are fully connected through undirected weighted connections in a bipartite network. Figure 1 shows the example of an RBM network with $X = 4$ and $H = 5$.

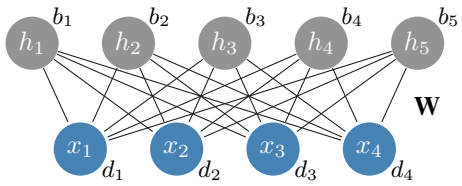


Figure 1: The RBM network for 4 visible and 5 hidden units.

Each configuration $(\mathbf{x}, \mathbf{h})$ has the following associated energy: $E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{d} - \mathbf{h}^{\mathrm{T}}\mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{H,X}$ is the weight matrix of the layers' connections (w_{ij} is the weight between visible unit x_j and hidden unit h_i), $\mathbf{d} \in \mathbb{R}^X$ is the visible units' bias vector (d_j is the bias for x_j) and $\mathbf{b} \in \mathbb{R}^H$ is the hidden units' bias vector (b_i is the bias for h_i). $\mathbf{W}$, $\mathbf{d}$ and $\mathbf{b}$ are the model parameters, subsequently denoted by $\theta = (\mathbf{W}, \mathbf{d}, \mathbf{b})$. The probability distribution of the RBM is defined as $P_\theta(\mathbf{x}, \mathbf{h}) = Z^{-1}e^{-E(\mathbf{x},\mathbf{h})}$, with Z being the normalization constant (or partition function). Note that this equation is in general not tractable due to the very large number of configurations (2^{X+H}).

3.1 Training the Restricted Boltzmann Machine

The RBM is typically trained to minimize the Negative Log-Likelihood (NLL) of the available data set, which is equivalent to maximizing the Log-Likelihood. In this case, the average NLL is often adopted in order to simplify the learning procedure. Given a data set $\{x^{(t)}\}_{t=1,\dots,T}$ with T samples, the average NLL of the model is simply $\frac{1}{T}\sum_{t=1}^{T} - \ln P_\theta(\mathbf{x}^{(t)})$.

The RBM is trained by applying Stochastic Gradient Descent (SGD) (Bottou, 2010). Due to the intractability of the normalization constant, training methods such as Contrastive Divergence (CD) (Hinton, 2002) approximate the gradient with the following expression:

$$\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \mathbb{E}_{\mathbf{h}}\left[\nabla_\theta E(\mathbf{x}, \mathbf{h}) \,\middle|\, \mathbf{x} = \mathbf{x}^{(t)}\right] - \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \mathbb{E}_{\mathbf{h}}\left[\nabla_\theta E(\mathbf{x}, \mathbf{h}) \,\middle|\, \mathbf{x} = \tilde{\mathbf{x}}^{(t)}\right], \tag{1}$$

where $\mathcal{B}$ corresponds to a batch of samples randomly chosen from the data, a widely applied heuristics (Hinton, 2012); and $\tilde{\mathbf{x}}^{(t)}$ is a random sample of the RBM given its parameters. Note that equation 1 requires generating a random sample from the RBM distribution for each data sample $\mathbf{x}^{(t)}$, which is done applying k steps of Gibbs Sampling on the model, starting from the data sample $\mathbf{x}^{(t)}$.

Calculating the corresponding expectations for each model parameter w_{ij}, b_i, d_j, the resulting parameter update rules are given by:

$$\mathbf{W} \leftarrow \mathbf{W} + \alpha\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\left(\hat{\mathbf{h}}(\mathbf{x}^{(t)})\mathbf{x}^{(t)^{\mathrm{T}}} - \hat{\mathbf{h}}(\tilde{\mathbf{x}}^{(t)})\tilde{\mathbf{x}}^{(t)^{\mathrm{T}}}\right) \tag{2}$$

$$\mathbf{b} \leftarrow \mathbf{b} + \alpha\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\left(\hat{\mathbf{h}}(\mathbf{x}^{(t)}) - \hat{\mathbf{h}}(\tilde{\mathbf{x}}^{(t)})\right) \tag{3}$$

$$\mathbf{d} \leftarrow \mathbf{d} + \alpha\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}}\left(\mathbf{x}^{(t)} - \tilde{\mathbf{x}}^{(t)}\right) \tag{4}$$

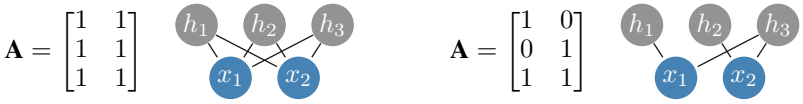
where $\alpha > 0$ is the learning rate hyperparameter and $\hat{\mathbf{h}}(\mathbf{x}) = \boldsymbol{\sigma}(\mathbf{b} + \mathbf{W}\mathbf{x})$, with $\boldsymbol{\sigma}(\cdot)$ being the element-wise operation of $\sigma(y) = \frac{1}{1+e^{-y}}$.

4 Network Connectivity Gradient

The classic RBM considers a fully connected network between its input and hidden layers. However, this is not necessarily the best connectivity pattern for learning a model for a particular task, prompting the investigation of other patterns.

Let $\mathbf{A} \in \mathbb{B}^{H,X}$ denote a binary matrix that represents a given connectivity pattern for the RBM, in the sense that $a_{ij} = \mathbf{A}[i, j] = 1$ if hidden unit h_i is connected to input unit x_j, or $a_{ij} = \mathbf{A}[i, j] = 0$ otherwise. Figure 2 shows examples of the adjacency matrix $\mathbf{A}$ for two connectivity patterns. Note that it is generally intractable to enumerate them even in the case of small models (2^{HX} possibilities

of **A**). In order to incorporate **A** into the model, the weights in matrix **W** must be zero on entries where a connection is not present. Thus, let $\mathbf{C} = \mathbf{W} \odot \mathbf{A}$ denote the acting weights of the model where $\odot$ is the element-wise matrix product such that $c_{ij} = \mathbf{C}[i, j] = w_{ij}a_{ij}$. The classic model parameters can be learned as before by using matrix **C** instead of **W** to compute the gradients.



(a) Dense Network (b) Sparse Network

Figure 2: Two examples of adjacency matrices **A** and the corresponding RBM networks: the classic fully connected network (a), and a model with the connections $x_1 - h_2$ and $x_2 - h_1$ suppressed (b).

The novelty of the proposed method lies on computing a gradient for each possible element (connection) of **A**. This can be analytically derived as with the other RBM parameters, where θ in equation 1 also includes **A**. Note that the gradient for a connection (i, j) can be non-zero even when $a_{ij} = 0$. This is a key aspect in the methodology here proposed, since it provides a gradient for absent connections and consequently the possibility for them to be enabled (or permanently disabled).

However a_{ij} is binary, and thus the usual continuous optimization framework that leverages the gradient to update its value do not apply. To circumvent this limitation, a continuous parameter denoting the connectivity strength is introduced in the model, and represented by $\mathbf{A}' \in [0, 1]^{H,X}$ such that $0 \leq a'_{ij} = \mathbf{A}'[i, j] \leq 1$. Thus, the connection strength can be updated using the corresponding gradient (but saturating at 0 or 1), and the binary connection becomes a function of the connection strength. In particular, a simple threshold (step) function is used to determine the presence or absence of a connection. This idea leads to the following two-step update rule for the connection parameters:

$$\begin{aligned} a'_{ij} &\leftarrow a'_{ij} + \frac{\alpha_A}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \left[\hat{h}_i(\mathbf{x}^{(t)}) w_{ij} x_j^{(t)} - \hat{h}_i(\tilde{\mathbf{x}}^{(t)}) w_{ij} \tilde{x}_j^{(t)} \right] \\ a_{ij} &\leftarrow \mathbb{1}\left[a'_{ij} \geq \gamma \right] \end{aligned} \tag{5}$$

where γ is the hyperparameter that denotes the threshold for enabling/disabling a connection based on the connections strength, $\mathbb{1}[\cdot]$ corresponds to the indicator/step function, and α_A to the connectivity learning rate. The method is called Network Connectivity Gradient (NCG) and jointly learns the connectivity pattern and classic model parameters for the RBM. Note that α_A allows to decouple the learning rate of model parameters from the connectivity pattern.

A fundamental aspect in continuous optimization frameworks such as SGD is the initialization of the parameters that must be optimized. Being parameters, the connectivity pattern and connection strength must also be initialized. A common initialization in the context of RBM (and other models) is choosing random and small values for the parameters. Thus, each possible connection is initialized as active ($a_{ij} = 1$) with probability p or inactive $a_{ij} = 0$) with probability $1 - p$, in which p is a hyperparameter.

Once the initial connection pattern has been determined, the connection strengths must also be defined. While initializing $a'_{ij} = a_{ij}$ is a possible initialization, this leads to connection strengths that are either 0 or 1 which may require too many iterations in order to cross the threshold to enable or disable the connection, respectively. To avoid this cold start, connection strengths are randomly initialized so that $a'_{ij} = \mathrm{U}(0, \gamma)\, \mathbb{1}\left[a_{ij} = 0\right] + \mathrm{U}(\gamma, 1)\, \mathbb{1}\left[a_{ij} = 1\right]$, where $\mathrm{U}(a, b)$ is the continuous uniform random value in the interval $[a, b]$. Note that the random value of the connection strength depends on the threshold γ for enabling/disabling the connection. Intuitively, a random value is chosen in the segment corresponding to the connection being absent (range $[0, \gamma]$) or present (range $[\gamma, 1]$).

5 Empirical Evaluation

Two tasks will be considered to evaluate the learning performance of the RBM under the NCG methodology: sample generation (average NLL is the performance metric) and image classification (accuracy is the performance metric). The main data set used is MNIST [1], a frequently used benchmark

[1] Data set available at `http://yann.lecun.com/exdb/mnist/`.

in computer vision and the RBM literature (Fischer & Igel, 2014). Also, two data sets from the UCI Evaluation Suite (Dua & Graff, 2017) are used: Mushrooms and Connect-4 [2].

The MNIST data set consists of gray-scale square images of 28×28 pixels. It has two separate sets of data: a train set, with 60 k samples; and a test set, which has 10 k samples. The images were converted into black and white in order to be directly used as input to the RBM. Each pixel was assigned a black color with probability proportional to its darkness (gray-scale) in the original image, a methodology commonly adopted (Salakhutdinov & Murray, 2008; Côté & Larochelle, 2016). Some examples of the resulting data are shown in figure 3.

Figure 3: Examples of MNIST data set images after conversion to binary.

For MNIST, each image in the data set has 784 pixels, each of which corresponds to a visible unit of the RBM. The experiments use 500 hidden units, and training was achieved using CD with 10 steps of Gibbs sampling (CD-10). The learning rate for the model parameters was set to $\alpha = 0.1$ and mini batches of 50 random samples. The connectivity learning rate was $\alpha_A = 0.5$. No momentum or weight decay were used. The RBM weight parameters were initialized with null biases and small random weights, uniformly distributed in the interval $[-1, 1]$. For the connection threshold in NCG, $\gamma = 0.5$ was adopted as this is the midpoint value in the possible range for the connection strengths, not favoring either a more sparse ($\gamma > 0.5$) or dense network ($\gamma < 0.5$). During training, one epoch corresponds to one iteration over the entire training data set with the model's parameters being updated at every batch.

The Mushrooms data set contains characteristics of different types of mushrooms, subdivided into edible and poisonous categories. There are 21 attributes, converted into 112 binary features, and 8124 samples (subdivided into 2 k for training the rest for testing). The Connect-4 data set contains board situations for the game of connect-4, labeled by whether the first player wins, loses, or there is a draw. There are 67557 samples (with 16 k separated for training), each with 42 board spaces, converted into 126 binary features. Experiments for these data sets used 100 hidden neurons, batch of 10 random samples, $\alpha = 0.01$ and $\alpha_A = 0.05$. Other hyperparameters were kept the same.

Although fine-tuning these parameters could improve the learning performance of the RBM, the goal here is to compare NCG with other connectivity patterns, and not necessarily obtain the best model across different hyperparameters.

5.1 Generative Results

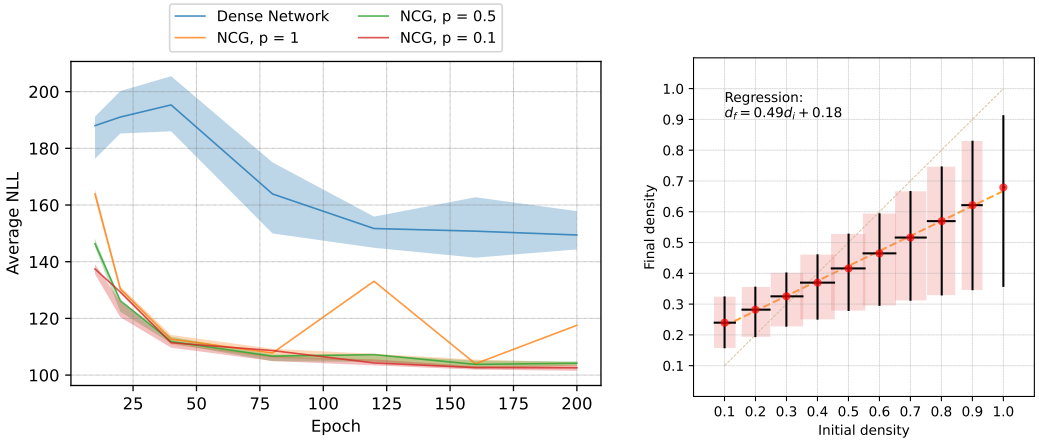
In the sample generation task, a classic generative RBM is trained as to generate random samples similar to the input examples. The performance is assessed using the average NLL across the training set. Since the exact average NLL cannot be computed due to intractability of the normalization constant, the Annealed Importance Sampling (AIS) method is used as an approximation (Salakhutdinov & Murray, 2008), with 100 runs and 14.5 k intermediate distributions.

Figure 4(a) shows the learning curve (evolution of the average NLL over the epochs) for the classic fully connected RBM and three initializations of the NCG method. Clearly, the fully connected network exhibits a significantly worse learning curve, both in terms of sample mean and variance. Interestingly, the three different initializations for NCG exhibit a very similar performance (with the exception of a few outliers for the case $p = 1$).[3] While the mean performance for $p = 0.1$ could be said to be slightly better, the overlapping quartiles show that the sparsity of the random initialization is not particularly important in this scenario. Indeed, the similar learning curves indicate that NCG can find effective networks independently of the (random) initial connectivity.

Despite the similar learning curves, the evolution of the network degrees is very different across the different initializations. Figure 5 shows the evolution of the maximum, minimum and average

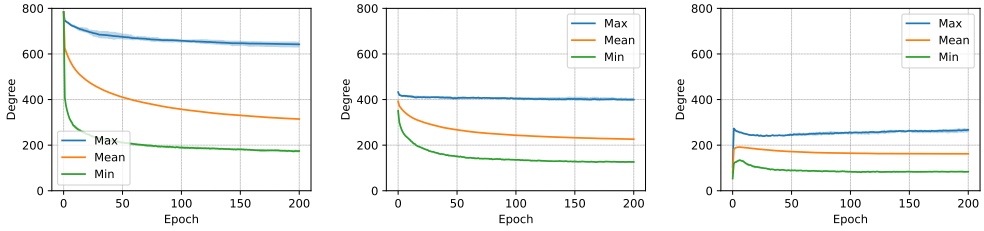
[2] Data sets available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[3] While training NCG with $p = 1$, two of the 10 runs exhibited much higher than average NLL at epoch 120 and one of the 10 runs at epoch 200.



(a) Average NLL (b) Density comparison

Figure 4: (a) Average NLL over the training epochs – lines are the means and shades the quartile uncertainty over 10 experiments; (b) Comparison between initial and final fraction of active connections for NCG training after 10 epochs – the dots are the mean over 25 experiments, the shades the uncertainty (minimum and maximum values) and the orange line the linear fit.



(a) $p = 1$ (b) $p = 0.5$ (c) $p = 0.1$

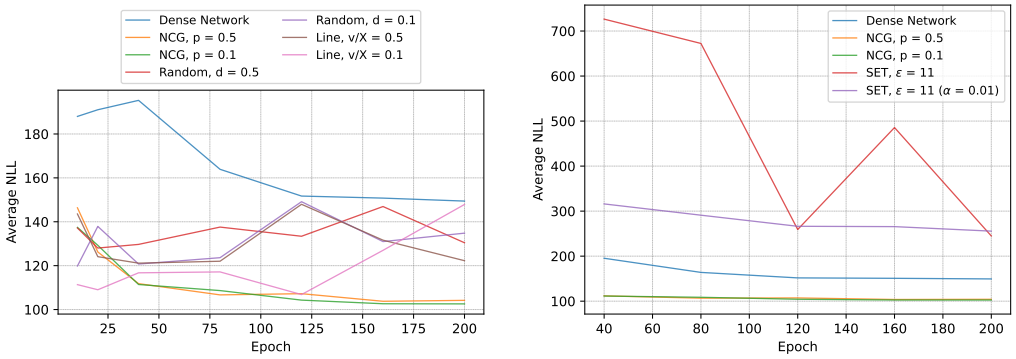
Figure 5: Degree statistics (minimum, average, maximum) of the hidden units over the NCG training epochs – lines are the sample mean and shades the sample quartile over 10 experiments.

degree of the hidden units for the three initializations. For $p = 1$ a sharp decrease is observed in all three statistics in the first 10 epochs, with the curves indicating a slight decay even aver 200 epochs, while for $p = 0.5$, the initial decrease is not as strong and the curves seem closer to convergence. Interestingly, the case $p = 0.1$ shows an increase in all three statistics in the first 10 epochs and convergence after 200 epochs. This shows that NCG can not only prune connections but also *add* connections when the network is too sparse. The similar learning curves but different network patterns indicate that the joint optimization of model parameters and network connectivity can compensate for one another, leading to similar performance even when the connectivity pattern is different.

Further insight is provided by figure 4(b) which shows that NCG tends to increase the number of connections when the initial network is too sparse (up to 30% of connections initially activated) and decrease the number of connections when the initial network is too dense (40% or more connections). Note that randomness of the final density (vertical bars) is much larger than that of initial density (horizontal bars), as the final density depends on the optimization.

Despite the good performance, it is known that sparse networks can outperform their dense counterparts. Indeed, even simple patterns such as the line pattern (connecting each hidden unit to v consecutive visible units) can achieve better learning curves (de Oliveira & Figueiredo, 2022). Figure 6(a) shows the learning curves of NCG and both the line pattern and a random pattern (corresponding to a bipartite Erdös-Rényi random graph). For fairness of comparison, RBMs with the same (average) number of initial connections are created for all patterns: 50% and 10% densities. Note that NCG shows better performance than any other pattern after 50 training epochs (for both initializations) while also showing less noisy learning curves. Interestingly, all patterns outperformed the fully connected RBM.

Lastly, Figure 6(b) shows a direct comparison between NCG and SET method, trained with 2500 hidden units, sparsity parameter $\epsilon = 11$, fraction of removed edges $\zeta = 0.3$, and learning rate of 0.1



(a) Comparison with Line and Random patterns

(b) Comparison with SET

Figure 6: Average NLL over the training epochs – lines are the mean values over 10 experiments. Comparison of NCG with the Line and Random patterns (a) and with the SET method (Mocanu et al., 2018) (b).

(parameters reported in the original paper, using the publicly available code). A smaller learning rate of 0.01 was also used in order to reduce the fluctuations during learning. However, note that SET did not outperform the fully connected network (on this task), and was thus significantly outperformed by NCG (see other comparisons in the appendix D).

5.2 Classification Results

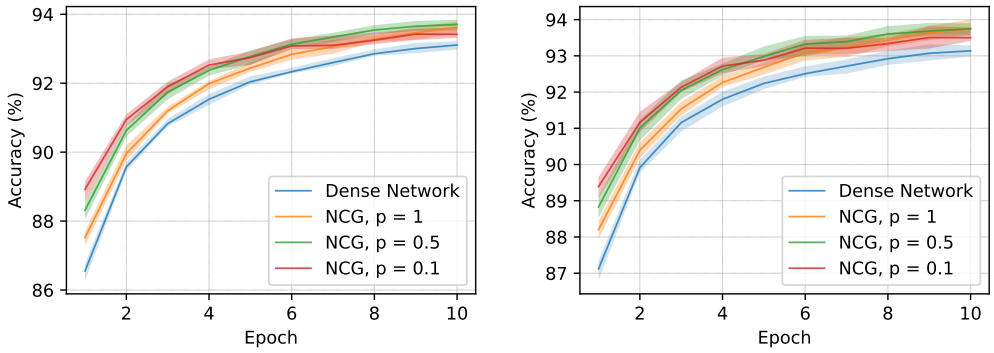
In the classification task, the RBM is trained to classify the given input (the digit in the image, for MNIST). The RBM is expanded to have additional visible (input) units in order to encode the label of the image during training (Fischer & Igel, 2014; Larochelle et al., 2012). There is one extra unit per class, and only one is activated for each input sample, the one corresponding to the sample class (the image digit, from 0 to 9, for MNIST). The connections between hidden units and the label units are fixed and not subjected to optimization, as they are crucial for the classification task. Moreover, the RBM is trained using Contrastive Divergence as in the generative task, and is not a priori aware of the classification task (no changes to the objective function).

While the generative task relied on the approximate NLL to measure the learning performance of the RBM, the classification task uses the accuracy as performance, given by the fraction of images correctly classified. Classification is performed by presenting the image to the RBM, setting each label units to 0.5, calculating the probabilities of each hidden unit being activated, and finally selecting the label unit (digit) with the higher probability of being activated. This digit is the predicted label for the image.

Figure 7 shows the evolution of the classification accuracy over the epochs for different NCG initializations for the train and test sets. Note that all three initializations generate models that are consistently better than the fully connected RBM in the early stages of training. Moreover, the performance between training and test sets are qualitative and quantitatively similar, indicating there is likely no overfitting.

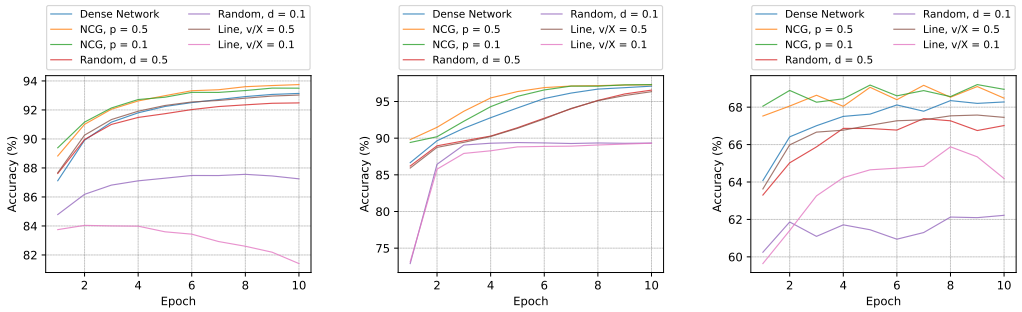
Interestingly, Figure 7 shows that accuracy is inversely proportional to the initial density during the first epochs of training: initializing the network with fewer connections yields superior accuracy in early stages of training. However, as the number of epochs increase, the accuracy between the NCG models becomes more similar. This indicates that NCG is capable of overcoming a poorly initialized connectivity pattern by adjusting both the connections and model weights. Interestingly, the degrees of hidden units of different initializations are very different after 10 epochs (see appendix A).

As with the generative task, NCG was also compared with the line and random patterns in the classification task. Figure 8 shows the accuracy of these different models for the first 10 epochs of training for both the MNIST data set as well as Mushrooms and Connect-4. For all datasets, the results indicate that NCG has a superior performance, outperforming the dense network, in particular during the beginning of training. The other patterns showed markedly worse performance, even when



(a) Train Set (b) Test Set

Figure 7: Classification accuracy over the training epochs for the train (a) and test (a) sets of the MNIST data set; lines correspond to the sample mean and shade corresponds to the sample quartile over 25 experimental runs.



(a) MNIST (b) Mushrooms (c) Connect-4

Figure 8: Test set classification accuracy for NCG, the classical RBM, and the Line and Random patterns for the MNIST (a), Mushrooms (b) and Connect-4 (c) data sets; lines are the sample means over 25 runs.

compared to the dense network. This illustrates that the connectivity pattern is more important than network sparsity.

6 Conclusion

This work presented Network Connectivity Gradient (NCG), a method tailored to RBMs that learns the optimal connectivity network jointly with other model parameters (weights and biases). NCG computes gradients for each possible network connection given a connectivity pattern. The gradients are used to drive the continuous connectivity strength parameter that in turn determines to maintain, add or remove the connection in each training epoch. NCG requires no change in RBM's objective function nor its classic optimization framework. Evaluation of NCG on a generative and classification task using the MNIST and other data sets demonstrated its effectiveness in learning better models (learning faster and better) than the dense RBM, other static patterns, and the SET method, as well as robustness with respect to its initialization.

However, recent works on pruning at initialization (Lee et al., 2019; de Jorge et al., 2021) might be leverage to design more effective initial networks for NCG. Last, while NCG has been designed for RBMs, future work will reveal if its core ideas can be applied to other neural network models.

Reproducibility Statement

The codes used to produce the experiments here portrayed will be made available for final submission.

REFERENCES

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? In *Machine Learning and Systems (MLSys)*, pp. 129–146, 2020.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics (COMPSTAT)*, pp. 177–186. 2010.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International Conference on Machine Learning (ICML)*, pp. 1695–1706, 2021.

Zhourong Chen, Nevin Zhang, Dit-Yan Yeung, and Peixian Chen. Sparse boltzmann machines with structure learning as applied to text analysis. *AAAI Conference on Artificial Intelligence*, 31(1), 2017.

Marc-Alexandre Côté and Hugo Larochelle. An infinite Restricted Boltzmann Machine. *Neural computation*, 28(7):1265–1288, 2016.

Pau de Jorge, Amartya Sanyal, Harkirat S Behl, Philip HS Torr, Gregory Rogez, and Puneet K Dokania. Progressive skeletonization: Trimming more fat from a network at initialization. In *International Conference on Learning Representations (ICLR)*, 2021.

Amanda C. N. de Oliveira and Daniel R. Figueiredo. Network connectivity and learning performance on Restricted Boltzmann Machines. In *International Joint Conference on Neural Networks (IJCNN)*, 2022.

Aurélien Decelle and Cyril Furtlehner. Restricted Boltzmann Machine: Recent advances and mean-field theory. *Chinese Physics B*, 30(4):040202, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186, 2019.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10628–10637, 2020.

Asja Fischer and Christian Igel. Training Restricted Boltzmann Machines: An introduction. *Pattern Recognition*, 47(1):25–39, 2014.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1135–1143, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E Hinton. A practical guide to training Restricted Boltzmann Machines. In *Neural networks: Tricks of the trade*, pp. 599–619. 2012.

Hugo Larochelle, Michael Mandel, Razvan Pascanu, and Yoshua Bengio. Learning algorithms for the classification Restricted Boltzmann Machine. *Journal of Machine Learning Research*, 13(1): 643–669, 2012.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip Torr. SNIP: Single-shot network pruning based on connection sensitivity. In *International Conference on Learning Representations (ICLR)*, 2019.

Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.

M. E. Midhun, Sarath R Nair, V. T. Nidhin Prabhakar, and S. Sachin Kumar. Deep model for classification of hyperspectral image using Restricted Boltzmann Machine. In *International Conference on Interdisciplinary Advances in Applied Computing (ICONIAAC)*, pp. 1–7, 2014.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H. Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9(1):2383, 2018.

Joao Papa Papa, Gustavo Henrique Rosa, Kelton A. Costa, Nilceu A. Marana, Walter Scheirer, and David Daniel Cox. On the model selection of Bernoulli Restricted Boltzmann Machines through harmony search. In *Conference on Genetic and Evolutionary Computation (GECCO)*, pp. 1449–1450, 2015.

Ning Qiang, Qinglin Dong, Wei Zhang, Bao Ge, Fangfei Ge, Hongtao Liang, Yifei Sun, Jie Gao, and Tianming Liu. Modeling task-based fMRI data via deep belief network with neural architecture search. *Computerized Medical Imaging and Graphics*, 83:101747, 2020.

R. Reed. Pruning algorithms–a survey. *IEEE Transactions on Neural Networks*, 4(5):740–747, 1993.

Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning (ICML)*, pp. 872–879, 2008.

Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 11380–11390, 2020.

Paul Smolensky. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pp. 194–281. 1986.

Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Robust Boltzmann Machines for recognition and denoising. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2264–2271, 2012.

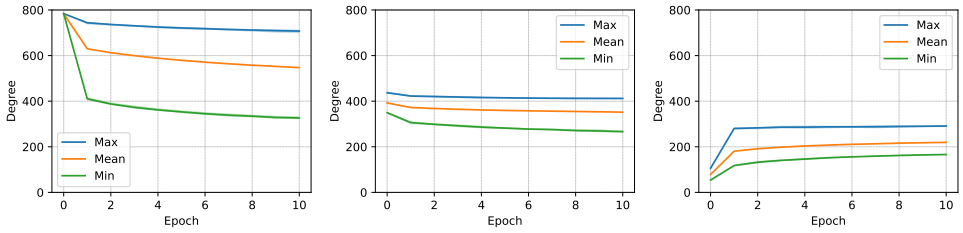
Tijmen Tieleman. Training Restricted Boltzmann Machines using approximations to the likelihood gradient. In *International Conference on Machine Learning (ICML)*, pp. 1064–1071, 2008.

Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. Discovering neural wirings. In *Advances in Neural Information Processing Systems (NIPS)*, volume 32, 2019.

Aojun Zhou, Yukun Ma, Junnan Zhu, Jianbo Liu, Zhijie Zhang, Kun Yuan, Wenxiu Sun, and Hongsheng Li. Learning N:M fine-grained structured sparse neural networks from scratch. In *International Conference on Learning Representations (ICLR)*, 2021.

A CLASSIFICATION DEGREE STATISTICS FOR MNIST

Figure 9 portrays the degree statistics (minimum, average, and maximum) of the network's hidden units over the epochs for the NCG classification experiments in the MNIST data set (Figure 7 shows the accuracy). Note that for $p = 1$ all degrees are 784 at time zero, and NCG significantly reduces the degrees of the network; the average degree is reduced by 30% after 10 epochs. On the other hand, for $p = 0.1$, NCG significantly increases the degrees of the network; the average degree is 2.5 times larger after 10 epochs. Finally, for $p = 0.5$ NCG shows a relatively small change in the degrees. Moreover, while the degrees change and converge over the epochs, the initialization density has a strong influence: the average degree of the three models after 10 epochs reflects their initial density.

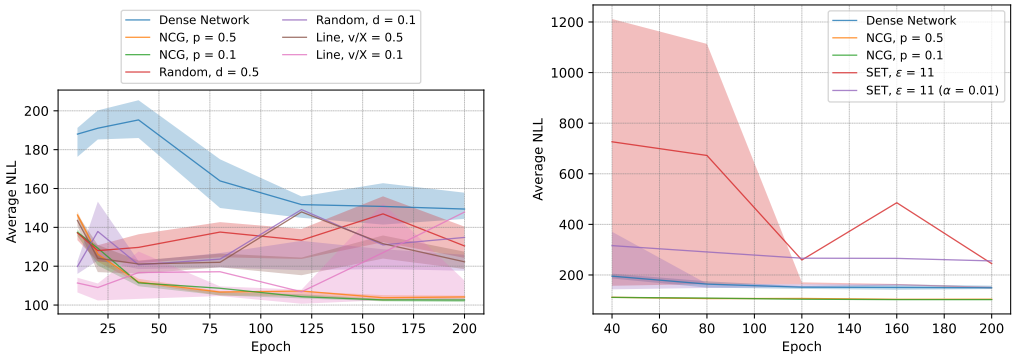


(a) $p = 1$ (b) $p = 0.5$ (c) $p = 0.1$

Figure 9: Degree statistics (minimum, average, maximum) of the hidden units over the training epochs – lines correspond to the sample mean and shades to the sample quartiles over 25 experimental runs.

B QUARTILE FIGURES

Some plots in the main body had their uncertainties removed to avoid clutter. The plots with quartile representing the uncertainty are shown here. Figure 10 shows the learning curves of NCG and the classic RBM compared to the line and random patterns as well as the SET method for the generative task, as in Figure 6.

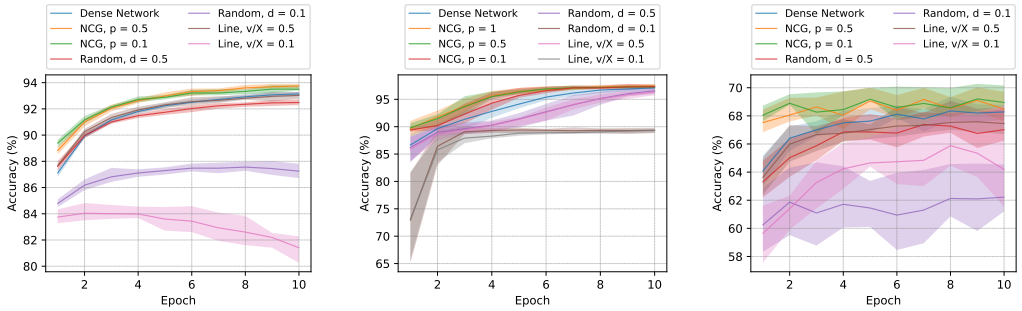


(a) Comparison with Line and Random patterns (as in figure 6(a))

(b) Comparison with SET (as in figure 6(b))

Figure 10: (Regarding figure 6) Average NLL over the training epochs – lines are the mean values and shades are the quatiles over 10 experiments. Comparison of NCG with the Line and Random patterns (a) and with the SET method Mocanu et al. (2018) (b).

Meanwhile, Figure 11 presents the classification results for training of NCG compared with the fully connected RBM, the line and the random patters, for data sets MNIST, Mushrooms and Connect-4, as in Figure 8.



(a) MNIST (b) Mushrooms (c) Connect-4

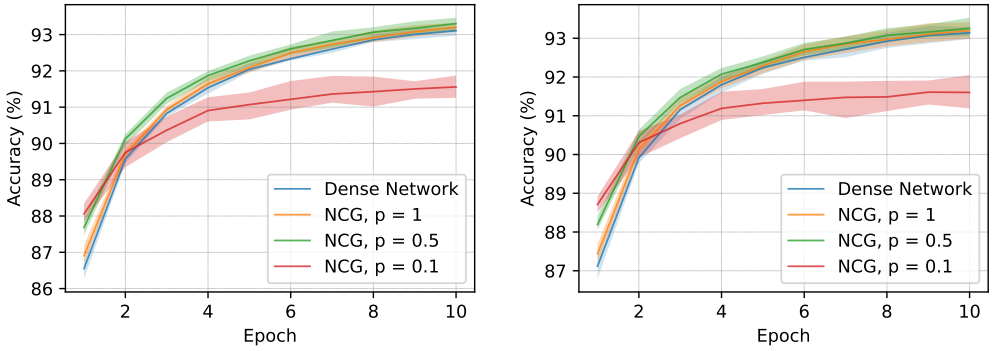
Figure 11: (Regarding figure 8) Test set classification accuracy for NCG, the classical RBM, and the Line and Random patterns for the MNIST (a), Mushrooms (b) and Connect-4 (c) data sets; lines are the sample means and shades are the quatiles over 25 runs.

C Ablation Studies

C.1 Learning Rate Analysis

Empirical experiments showed that the relatively higher connectivity learning rate $\alpha_A = 0.5$ plays an important role in allowing the network connectivity to evolve fast in the early stages of training. Intuitively, this allows NCG to quickly adjust for poor initial network patterns before other model parameters start to converge.

To illustrate this, Figure 12 shows the accuracy when using a connectivity learning rate of $\alpha_A = 0.1$, which is equal to the learning rate of other model parameters. Note the decrease in the accuracy for all three initializations for all 10 epochs (in comparison to Figure 7). Interestingly, while the performance for $p = 0.1$ is superior after 1 epoch of training (as with $\alpha_A = 0.5$), the model fails to continue improving its accuracy and falls behind the other models, including the fully connected network. Intuitively, the model cannot adjust its connection pattern fast enough and the connectivity gradient becomes subdued by other model parameters. This example highlights the importance of decoupling the learning rates when jointly optimizing network connectivity and other model parameters.



(a) Train Set (b) Test Set

Figure 12: Classification accuracy over the training epochs for the train (a) and test (b) sets of the MNIST data set with $\alpha_A = 0.1$; lines correspond to the sample mean and shade corresponds to the sample quartile over 25 experimental runs.

C.2 Contrastive Divergence Approximation

The experiments on this article applied Contrastive Divergence training using 10 steps of Gibbs Sampling (CD-10). However, changing the number of steps (and the way of obtaining the sample $\tilde{x}$

entirely) can deeply affect results. To exemplify this, some evaluations using CD-1 (only one step of Gibbs Sampling) were performed. This creates a poorer gradient approximation, which usually affects training negatively.

C.2.1 GENERATIVE RESULTS

Figure 13 portrays the learning curves for dense RBM and NCG, and Figure 14 the corresponding degree statistics. It is clear that CD-1 causes a major performance drop for all models considered: by the end of training the fully connected RBM has the average NLL around 290 in comparison to the 150 seen in Figure 4(a), and the models trained with the NCG method reach at most 150, when before the worse average did not surpass 120. The degree statistics for $p = 1$ and $p = 0.5$ appear to show less change along the epochs than what was observed for CD-10, but the differences do not appear to be significant.

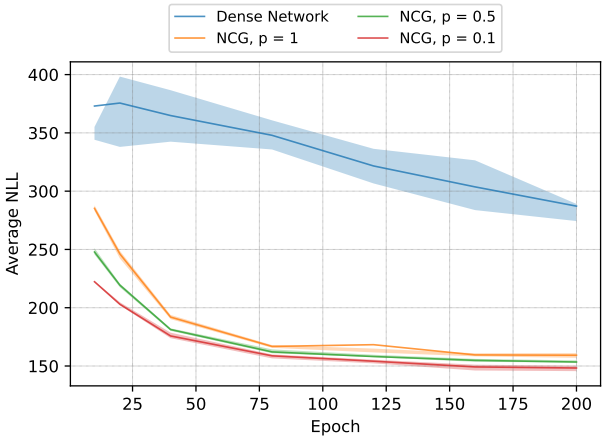
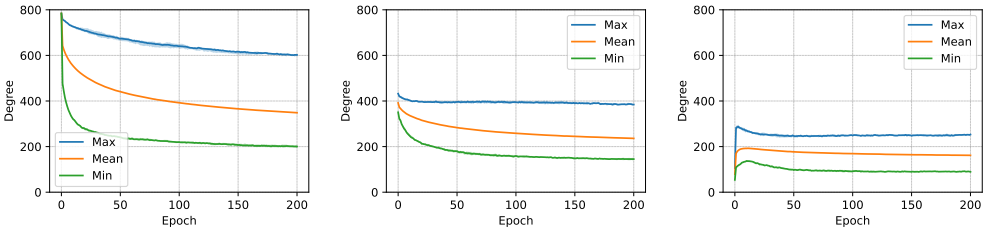


Figure 13: Average NLL over the training epochs for the MNIST dataset with CD-1 – lines correspond to the sample mean and shades to the sample quartiles over 10 experimental runs.



(a) $p = 1$ (b) $p = 0.5$ (c) $p = 0.1$

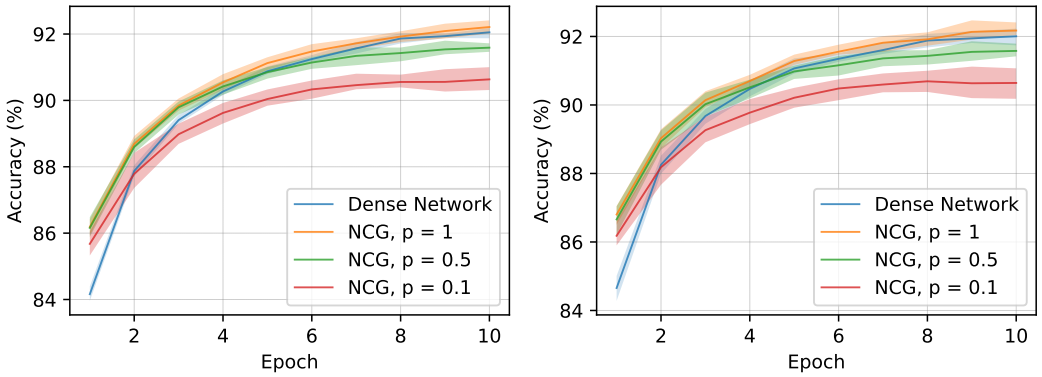
Figure 14: Degree statistics (minimum, average, maximum) of the hidden units over the training epochs. Generative results on MNIST, trained using CD-1. Lines correspond to the sample mean and shade corresponds to the sample quartiles over 10 experimental runs.

Interestingly, the NCG models' NLL increase less than the classical RBM, for which the final NLL is double the value when considering CD-10, and the relative increase in performance derived from optimizing the connectivity (NCG) is much larger.

C.2.2 CLASSIFICATION RESULTS

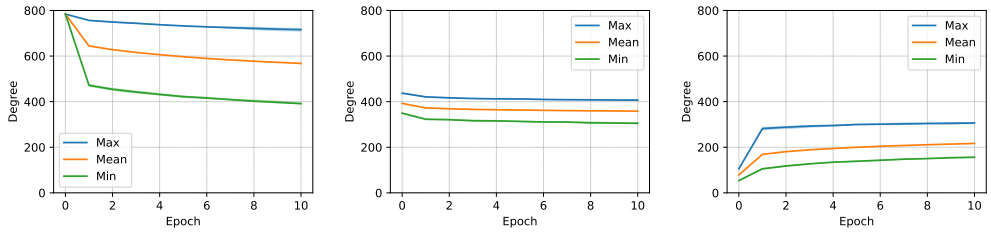
While the goal in the classification task is to maximize accuracy, the objective function used during training with aims to minimize the NLL, using CD as an approximation to the gradient. Therefore NCG trains the connectivity network for a slightly inaccurate objective, as well as all the other parameters that the traditional network trains. It stands to reason, therefore, that in worsening the approximation for the gradients, its performance will suffer.

Figure 15 shows the evolution of the accuracy over epochs for the dense RBM as well as three initializations for the NCG model, for both the train and test sets. Figure 16 presents the corresponding degree statistics evolution, giving an idea of how the connectivity changes with training. Once again, the degree statistics do not show much difference from their CD-10 counterparts, except that they suffer less change throughout training.



(a) Train Set (b) Test Set

Figure 15: Classification accuracy over the training epochs for the train (a) and test (b) sets of the MNIST data set. Trained with CD-1. Lines correspond to the sample mean and shades to the sample quartiles over 25 experimental runs.



(a) $p = 1$ (b) $p = 0.5$ (c) $p = 0.1$

Figure 16: Degree statistics (minimum, average, maximum) of the hidden units over the training epochs. Classification results on MNIST, trained with CD-1. Lines correspond to the sample mean and shades to the sample quartiles over 25 experimental runs.

Although all RBMs have a worse accuracy when training with CD-1, it is clear from the results that the relative performance between NCG and the fully connected model diminishes. In these circumstances, only NCG initializing with all connections activated ($p = 1$) manages to surpass the traditional RBM, and even then they have very close results. It is not clear that the difference is statistically significant. The $p = 0.1$ training seems to suffer the most, not showing a better accuracy even in the first epoch of training. Overall, the results indicate a very different scenario in comparison to the one observed in the generative task, for which the addition of connectivity optimization resulted only in positive results, regardless of the CD approximation used.

D SET TRAINING

As mentioned in Section 5.1, the learning curves reported for the SET method used the hyperparameters mentioned by Mocanu et al. (2018). Figure 17 shows the comparison of NCG with SET, in which SET was trained with the same parameters as the previous experiments. Note that the sparsity parameter ϵ was chosen so as to create SET networks with nominal sparsity of 50% and 10% of all possible connections activated, which corresponds to values used for NCG intialization.

This choice of hyperparameters for the SET method did not yield good performance, which shows noisy learning curves without apparent convergence. Since this scenario did show good results for SET, experiments with other hyperparameters were performed and reported in Figure 6(b). In any case, NCG showed significantly superior and more robust performance (less noisy learning curves).

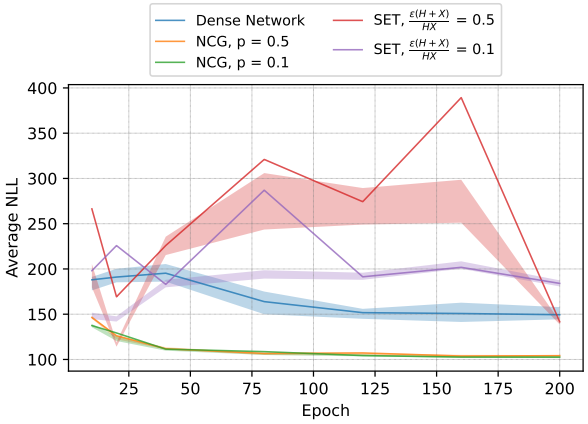


Figure 17: Average NLL over the training epochs – lines are the mean values and shades are the quatiles over 10 experiments. Comparison of NCG with the SET method.