Masked Autoencoder for Distribution Estimation on Small Structured Data Sets

Ahmad Khajenezhad, Hatef Madani, and Hamid Beigy

***Abstract*— Autoregressive models are among the most successful neural network methods for estimating a distribution from a set of samples. However, these models, such as other neural methods, need large data sets to provide good estimations. We believe that knowing structural information about the data can improve their performance on small data sets. Masked autoencoder for distribution estimation (MADE) is a well-structured density estimator, which alters a simple autoencoder by setting a set of masks on its connections to satisfy the autoregressive condition. Nevertheless, this model does not benefit from extra information that we might know about the structure of the data. This information can especially be advantageous in case of training on small data sets. In this article, we propose two autoencoders for estimating the density of a small set of observations, where the data have a known Markov random field (MRF) structure. These methods modify the masking process of MADE, according to conditional dependencies inferred from the MRF structure, to reduce either the model complexity or the problem complexity. We compare the proposed methods with some related binary, discrete, and continuous density estimators on MNIST, binarized MNIST, OCR-letters, and two synthetic data sets.**

***Index Terms*— Deep learning, density estimation, Markov random field (MRF), masked autoencoder.**

I. Introduction

THE problem of distribution estimation is to infer the density function $P(X)$ of a d-dimensional random variable $X = \langle X_1, X_2, \ldots, X_d \rangle$ from a set of independent and identically distributed observations $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$. It is a basic block for probabilistic learning and has a wide range of applications. Due to the intrinsic complexities of this problem, different models have been proposed for this task. Kernel density estimators (for continuous variables) [1] and mixture models [2] are among the most common density estimators. The problem becomes more challenging when dealing with high-dimensional data. Traditionally, estimating high-dimensional densities suffers from the necessity of having an exponentially large number of samples. The curse of dimensionality complicates the task by growing the space of estimators (i.e., increasing the number of model parameters) at an exponential rate.

Manuscript received February 13, 2019; revised October 24, 2019 and June 23, 2020; accepted September 16, 2020. Date of publication October 13, 2020; date of current version October 28, 2021. *(Corresponding author: Hamid Beigy.)*

The authors are with the Department of Computer Engineering, Sharif University of Technology, Tehran 1458889694, Iran (e-mail: khajenezhad@ce.sharif.edu; hmadani@ce.sharif.edu; beigy@sharif.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.3026572

Graphical models [3] have been widely used as density estimators for structured data. Such models reduce the volume of the space of the estimators by assuming a structural model for the underlying distribution, which can be expressed as a set of conditional independencies among different dimensions. Recent advances in generalizing restricted Boltzmann machines (RBMs) [4], approximating the parameters of pairwise Markov random fields (MRFs) [5], and structure learning in graphical models [6] are just a few examples of many studies in this field.

Another approach to solving the problem of high-dimensional density estimation is to use a representation of the data (i.e., extra variables) in a hidden space. The use of extra variables enriches the model's capacity for estimating complex densities. RBMs are well-known models of this type of density estimation [7]. However, according to their computational complexity for estimating the exact density function, RBMs are not commonly used today, but they have significant historical importance. By extending RBMs, some other density estimators have been proposed based on deep belief networks [8].

On the other hand, many successful neural density estimators have been proposed in recent years. A common approach among these methods is to use the chain rule to factorize the joint probability function to the product of conditional probabilities. Using this idea, several models have been proposed that receive a d-dimensional sample as input and estimate d conditional probabilities in the output layer. Since the ith output dimension estimates the conditional probability of the ith dimension given the preceding dimensions, it must be computed only from the first $i-1$ input dimensions. This property is called the atoregressive condition. Different neural autoregressive models have been proposed. Bengio and Bengio [9] extended the autoregressive approach of the fully visible sigmoid belief networks (FVSBNs) [10] by using single-layer neural networks instead of simple logistic regression models for estimating the conditionals. The idea of hidden representations used in RBM's was combined with the autoregressive approach in [11] and its extensions [12]–[14]. Among the autoregressive neural density estimators, the masked autoencoder for distribution estimation (MADE) [15] is a well-structured density estimator for binary variables, which alters a simple autoencoder by setting a set of masks on its connections to satisfy the autoregressive property. MADE does not rely on a hidden representation of the data, and its computation time on test data is less than the previous autoregressive neural models.

It is worth mentioning that there exist other neural network approaches to the problem of density estimation than autoregressive models, such as [16]–[19], which have used manipulated functions (normalizing flows) to transform the data into a desirable space, appropriate for density estimation, or to map a simple initial density function to the distribution of the observations. Moreover, variational autoencoders [20] can also be considered as density estimators, whereas some other generative models such as generative adversarial models [21] cannot be used for density estimation.

Although deep methods have caused significant progress in density estimation in recent years, they all rely on a large set of training observations. Therefore, they cannot be used in an application for which data collection is an expensive process (e.g., where data are collected through field observations). On the other hand, knowing structural information about the data can lead the estimators to provide better estimations. Our idea is to use this information to compensate for data shortage. In fact, we tackle the problem of estimating structured distributions by deep models using a small set of observations. We use available structural information from the data to simplify the estimation problem by either reducing the dimensionality of the regression problems that should be solved or reducing the complexity of the models we use. We use autoregressive neural models, which are among the most successful neural network methods for estimating a distribution from a set of samples and pick MADE among them, as a neat and well-known model. Although MADE is capable to estimate complicated functions, its performance depends on the number of training samples, such as other neural networks. In particular, it does not use available information about structural independencies of the data. However, structural information can resolve the network's need for a large set of training samples. In this article, we extend MADE and propose the Masked Autoencoder for Structured Distribution Estimation (MASDE) model, which imports structural information into the masking process of the MADE algorithm to improve the estimation, especially in the presence of a modest number of samples. MASDE is designed based on the idea of simplifying the regression problems that MADE solves by reducing the dimensionality of the domain spaces of those regression problems. In fact, in a sparse MRF, each conditional probability (appeared in the chain-rule factorization) might depend only on a small subset of the preceding dimensions. Considering this fact, MASDE tries to refine the paths between the input layer and the output layer in the autoencoder. Although MASDE is a general model that can be used for all MRF structures, for some data sets, it needs larger hidden layers than MADE to provide good results. Furthermore, its masking process is computationally more complex than MADE. However, for some specific sparse structures, we can consider simpler methods than MASDE for using the structural information in the masking process. One group of such structures is those that are comprised of the replication of a small structure, with local connections. We call them grid-based structures. As an example, consider a set of nodes located on a grid, in which each node is connected to the nodes in its neighborhood with a small euclidean distance. We propose another method, Masked Autoencoder for Grid Structured Distribution Estimation (MAGSDE), that suits grid-based structures. Using a small modification, MAGSDE alters the masking process of MADE to reduce the number of connections in the autoencoder. Our experimental results show the superiority of our proposed methods over MADE and some other baseline binary density estimators, especially in the case of learning with small training data.

The rest of this article is organized as follows. In Section II, density estimation using masked autoencoders is reviewed. In Section III, the proposed MASDE method for general sparse MRFs is introduced. The proposed method for grid-based structures, MAGSDE, is introduced in Section IV. Experimental results are presented in Section V. Finally, Section VI concludes this article.

II. Density Estimation Using Masked Autoencoders

The chain rule in probability theory states that the joint probability of random variables $X_1, \ldots, X_d$ can be factorized into a set of conditional probabilities

$$P(X_1, \ldots, X_d) = P(X_1)P(X_2|X_1)\ldots P(X_d|X_1, \ldots, X_{d-1}). \tag{1}$$

MADE [15] is a neural network with a d-dimensional input and a d-dimensional output layer. When this model is fed by a binary vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_d \rangle$, it estimates the conditional terms appeared in (1), in the output layer. In fact, it provides conditions in which the ith output dimension, y_i, can be considered as a valid estimation of $P(X_i = 1|x_1, x_2, \ldots, x_{i-1})$. Therefore, the negative log-likelihood of the input $\mathbf{x}$ is equal to

$$\begin{aligned} -\log P(x_1, \ldots, x_d) &= -\sum_{i=1}^{d} \log P(X_i = x_i|x_1, \ldots, x_{i-1}) \\ &= -\sum_{i=1}^{d} (\log P(X_i = 1|x_1, \ldots, x_{i-1})^{x_i} \\ &\quad + \log P(X_i = 0|x_1, \ldots, x_{i-1})^{1-x_i}) \\ &= -\sum_{i=1}^{d} (x_i \log y_i + (1-x_i)\log(1-y_i)). \end{aligned} \tag{2}$$

Hence, the averaged log-likelihood over a set of n inputs $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}$ with outputs $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(n)}$ is equal to

$$\begin{aligned} &\frac{1}{n}\sum_{k=1}^{n} \log P(\mathbf{x}^{(k)}) \\ &= -\frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{d} \left(x_i^{(k)} \log y_i^{(k)} + \left(1 - x_i^{(k)}\right) \log\left(1 - y_i^{(k)}\right)\right). \end{aligned} \tag{3}$$

MADE uses the loss function given in (3). Minimizing this loss function is equivalent to maximizing the log-likelihood function. On the other hand, this loss function is the well-known cross-entropy loss function, which is commonly

used in autoencoders for binary data.[1] Hence, MADE is a simple autoencoder for binary data, except that its output dimensions should satisfy the necessary constraints of the conditional terms in (1). Since $P(X_i|X_1, X_2, \ldots, X_{i-1})$ is a function of $X_{<i} = \langle X_1, X_2, \ldots, X_{i-1}\rangle$, a necessary constraint for the ith output dimension of the autoencoder in order to present the ith conditional probability is to be computed only from the first $i-1$ dimensions of the input. In other words, there should be no path from the last $d-i+1$ dimensions of the input layer to the ith dimension of the output layer. This property is called the autoregressive property. It can be easily shown that the autoregressive property is also sufficient for the outputs of the autoencoder to be a valid set of conditional probabilities.[2]

MADE is constructed by masking (removing) some connections of a simple autoencoder with fully connected feed-forward layers, to satisfy the autoregressive property. The masking method works based on random labels assigned to the nodes of the neural network. First, a label from the set $\{1, 2, \ldots, d\}$ is assigned to each node in the hidden layers. The label of each node is selected uniformly at random, independent of the label of the other nodes. The ith node in the input layer is assigned label i $(1 \leq i \leq d)$. The d nodes in the output layer are also labeled in the same way as the input layer. The label of a node determines the largest input dimension that is allowed to have a path to that node. If a node in a hidden layer is labeled by k, it means that only the first k dimensions of the input are allowed to have paths to this node. For the output layer, it is a bit different. If a node in the output layer is labeled by k, it means that only the first $k-1$ dimensions of the input are allowed to have paths to that node. Using these labels, the masking process is clear; the connection between node u with label j in one layer and node v with label i in the next layer will be masked if and only if $j > i$. If v is in the output layer, this condition changes to $j \geq i$. This masking method guarantees the autoregressive property.

Note that this method can simply handle the change of order of the dimensions in the chain rule by running one permutation on the labels of the nodes in both the input layer and the output layer. Using different random labels or different orders of dimensions results in different sets of masks on the network connections. An on-the-fly ensemble of models is suggested in [15] by making several sets of masks on the network connections and using one of them in each training epoch. The model parameters (i.e., network weights) are shared between all the mask sets. The final likelihood estimation for a test input will be the average of the estimations using all different mask sets.

[1]MADE can be used for multivariate discrete data or continuous data as well. In those cases, we should consider a parametric form for the conditional probabilities in (1). For example, for continuous data, a Gaussian function or a mixture of Gaussians can be used. A different loss function than cross-entropy should be used in these cases.

[2]In fact, the other constraints $\forall i : y_i \in [0, 1]$ need also to be satisfied. It will be easily handled by using sigmoid activation functions in the output layer of the neural network.

III. Density Estimation Using Masked Autoencoders for Structured Data

In structured distributions, the graph structure of the variables declares their conditional independencies. Therefore, having a graph structure, each of the chain rule conditional terms in (1) might be presentable by a conditional probability on a smaller set of variables. In other words, for each i $(1 \leq i \leq d)$, we can assume that there is a subset $B_i \subseteq \{1, \ldots, i-1\}$ such that

$$P(X_i|X_{<i}) = P(X_i|X_{B_i}). \tag{4}$$

From now on, we call B_i as the looking-back Markov blanket of dimension i, because given B_i, X_i is independent of the other preceding dimensions. Note that B_i is different from the Markov blanket of dimension i, since, by definition, the Markov blanket of X_i is a subset of X_{-i} that shields X_i from all other dimensions, but B_i is a subset of $X_{<i}$ that shields X_i from other dimensions of $X_{<i}$ (X_{-i} stands for all dimensions except for the ith one.). Using looking-back Markov blankets, we can rewrite (1) as

$$P(X_1, \ldots, X_d) = P(X_1)p(X_2|X_{B_2}) \ldots P(X_d|X_{B_d}). \tag{5}$$

Using (5) instead of Equation (1), makes the domain spaces smaller when solving the regression problems to estimate the parameters of the conditional probabilities. This simplification becomes more helpful when we have a few training data. However, to use (5) in a density estimation autoencoder, we need a stronger property than the autoregressive property. The autoregressive property obligates the ith dimension of the output layer to be computed only from the first $i-1$ dimensions of the input layer. For a d-tuple of looking-back Markov blankets $\mathcal{B} = \langle B_1, B_2, B_3, \ldots, B_d\rangle$, we define $\mathcal{B}$-restricted autoregressive property as an obligation that for each $i \in \{1, \ldots, d\}$, the ith dimension of the output layer of the autoencoder should be computed only from the dimensions of the input layer belonging to B_i. In other words, there should be no path from the dimensions outside B_i of the input layer to the ith dimension of the output layer in the neural network. Note that assuming $B_1 = \{\}$ and $B_i = \{1, \ldots, i-1\}$ for all $i \in \{2, \ldots, d\}$ results in the equivalence of the $\mathcal{B}$-restricted autoregressive property and the autoregressive property. However, we prefer smaller looking-back Markov blankets to obtain simpler models.

As described in Section II, to satisfy the autoregressive constraint in MADE, a label was assigned to each node in the network. This label determined the largest dimension in the input layer that was allowed to have paths to that node. Then, all the connections from a node with some label to a node with a larger label (or to a node in the output layer with equal label) were masked to guarantee the autoregressive constraint. The straightforward extension of this method to satisfy the $\mathcal{B}$-restricted autoregressive property can be achieved by letting the label of each node be a subset of $\{1, \ldots, d\}$ instead of a singleton member of that. This label indicates the set of input dimensions that are allowed to have paths to this node. Therefore, the nodes in the input layer should be labeled by $\{1\}, \{2\}, \ldots, \{d\}$, and the nodes in the output layer should

be labeled by $B_1, B_2, \ldots, B_d$. Nodes in the hidden layers would take random members of $\mathcal{P}(B_1) \cup \mathcal{P}(B_2) \cup \ldots \cup \mathcal{P}(B_d)$ as labels, where $\mathcal{P}(B_i)$ stands for the set of all subsets of B_i. Using these labels, the connection from a node u in a layer to node v in the next layer is held (i.e., will remain established) if the label of u is a subset of the label of v and should be masked otherwise. Since we have assigned labels $B_1, B_2, \ldots, B_d$ to the output layer nodes, this method of masking restricts the ith dimension of the output layer to be computed only from dimensions B_i of the input layer. Therefore, it is clear that $\mathcal{B}$-restricted autoregressive constraint is satisfied by this method.

Although this labeling method is general and can be used for every set $\mathcal{B}$ that satisfies the conditions given in (4), the number of possible labels for a node in the hidden layer is an exponential function of $\max_i |B_i|$. Therefore, if we select the labels of the nodes in the hidden layers uniformly at random from the set of all possible labels, we might need very large hidden layers. Otherwise, the connections between some related dimensions of the input layer and the output layer may be blocked. In what follows, we propose Masked Autoencoder for Structured Distribution Estimation (MASDE). This method still assigns subsets of $\{1, \ldots, d\}$ as labels to the nodes and uses the same way for masking connections as explained above but assigns labels to the nodes in a smarter manner than uniformly at the random selection. The labeling procedure of MASDE is as follows.

1) Assign label $\{i\}$ to the ith dimension of the input layer for $1 \leq i \leq d$.
2) Assign label B_i to the ith dimension of the output layer for $1 \leq i \leq d$.
3) Assign a random number from $\{2, \ldots, d\}$ to each node in the hidden layers. We call it the attachment number of that node. Do the assignment in a way that ensures each number in $\{2, \ldots, d\}$ is selected as the attachment number of at least one node in each layer.
   - ▹ If the attachment number of a node is k, then the label assigned to that node will be a subset of B_k. In addition, the label assigned to that node at the end of the labeling algorithm will be an extension of (or equal to) the label assigned to some node in the preceding layer whose label is a subset of B_k.
4) Assign labels to the nodes in the hidden layers, from the first hidden layer to the last one. For each node with an attachment number of k, run the following procedure.
   a) Select a node in the preceding layer whose label is a subset of B_k. There exists at least one such node in the preceding layer because there is at least one node with an attachment number of k in the preceding layer.
   b) Copy the label of the selected node from the preceding layer to the current node and add extra random members selected from B_k to its label.

Therefore, the connection from a node u in a layer to node v in the next layer is held (i.e., will remain established) if the label of u is a subset of the label of v and should be masked otherwise.

The abovementioned procedure ensures that each node in the neural network will have at least one incoming connection from the preceding layer, but there might be nodes with no connections to any nodes in the next layer. To address this issue for hidden layers, we can run an extra step (a backward pass over the hidden layers) as follows.

5) Investigate the hidden layers from the last layer to the first one.
   a) In each layer, for each node v that has no connection to the next layer, select a random node u, with label L_u, from the next layer.
   b) Since each node has at least one incoming connection, there exist some nodes in the preceding layer of v that their labels are subsets of L_u. Select one of them, say w, at random.
   c) Copy the label of w for v and add extra random members of L_u to its label.
   d) Update the masks on the connections between v and the nodes in its preceding and next layers.

The detailed labeling algorithm of MASDE is given in Algorithm 1.

It just remains to explain how we find the looking-back Markov blankets. The set of looking-back Markov blankets, $\mathcal{B}$, depends on the order of the dimensions. We prefer orders that result in small-sized looking-back Markov blankets. Having an order for the dimensions (i.e., having numbers 1 to d assigned to the dimensions), it is easy to find the looking-back Markov blankets in an MRF. For each node with number v, it suffices to run a depth-first search that does not progress when visiting nodes with numbers less than v. All the nodes with numbers less than v visited in this search should be included in B_v. Algorithm 2 shows the details of this procedure. Now, we are ready to state Theorem 1, which shows that Algorithm 2 returns the best looking-back Markov blanket for the node v. To increase readability, all proofs are given in the Appendix.

Theorem 1: The set of nodes returned by Algorithm 2 is a minimal looking-back Markov blanket of the node v, and any other valid looking-back Markov blanket of the node v should contain all the members of this set.

Similar to MADE, we run the random process of labeling multiple times to generate multiple sets of masks. We then use an on-the-fly ensemble model by using one set of masks in each training epoch. Note that all weight updates in the training process are performed on a common set of network parameters but using different masks in different epochs. The final estimate for a test input will be the average of the estimates using all different masks.

IV. Density Estimation Using Masked Autoencoders for Grid-Based Structures

Although MASDE is a general model that can be used for all MRF structures, given some data sets, large hidden layers are required to make enough paths from nodes in B_i from the input layer to node i in the output layer for all $1 \leq i \leq d$ (as we will show in the experimental results

Algorithm 1 MASDE Labeling

procedure MASDE-LABELING($\langle B_2, \ldots, B_d \rangle$)
  for $i : 1 \rightarrow d$ **do**
    $label[0][i] \leftarrow \{i\}$ ▷ assigning label to node i in the input layer
  end for
  # assigning attachments to the nodes in the hidden layers
  for $l : 1 \rightarrow$ number of hidden layers **do**
    ind $\leftarrow$ a random subset of size $d-1$ from $\{1, 2, \ldots,$ size of the l-th layer $\}$
    for $i : 1 \rightarrow d-1$ **do**
      attachment[l][ind[i]] $\leftarrow i$
    end for
    for $i \notin$ ind **do**
      attachment[i] $\leftarrow$ random from $\{2, 3, \ldots, d\}$
    end for
  end for
  # assigning labels to the nodes in the hidden layers
  for $l : 1 \rightarrow$ number of hidden layers **do**
    for $i : 1 \rightarrow$ size of the l-th hidden layer **do**
      $a \leftarrow attachment[l][i]$
      $j \leftarrow$ random from $\{1, 2, \ldots,$ size of the $l-1$-th hidden layer $\}$ s.t. $label[l-1][j] \in B_a$
      $label[l][i] \leftarrow label[l-1][j]$
      for $k \in (B_a - label[l-1][j])$ **do**
        add k to $label[l][i]$ with probability μ_l ▷ $\mu_l = l/($ number of hidden layers $+1)$
      end for
    end for
  end for
  for i = 1:d **do**
    $label[last][i] \leftarrow B_i$ ▷ assigning label to node i in the output layer
  end for
  # backward pass
  for l : number of hidden layers $\rightarrow 1$ **do**
    for $i : 1 \rightarrow$ size of the l-th hidden layer **do**
      if $label[l][i] \not\subset label[l+1][z] \quad \forall z \in \{1, 2, \ldots,$ size of the $l+1$-th layer $\}$ **then**
        $t \leftarrow$ random from $\{1, 2, \ldots,$ size of the $l+1$-th layer $\}$
        $j \leftarrow$ random from $\{1, 2, \ldots,$ size of the $l-1$-th layer $\}$ such that $label[l-1][j] \subset label[l+1][t]$
        $label[l][i] \leftarrow label[l-1][j]$
        for $k \in (label[l+1][t] - label[l-1][j])$ **do**
          add k to $label[l][i]$ with probability μ_l ▷ $\mu_l = l/($ number of hidden layers $+1)$
        end for
      end if
    end for
  end for
end procedure

on the OCR-letters data set). Moreover, the masking process of MASDE is computationally more complex than MADE. On the other hand, a common category of MRFs in real applications are grid-based MRFs. By "grid-based," we mean a simple grid or a set of replicated small patterns connected with local connections as a grid structure. This kind of structure appears in many spatial or temporal applications, especially in geoinformatics, image processing, and computer vision tasks. For these structures, we propose another masking method that works based on the autoregressive property and does not satisfy the $\mathcal{B}$-restricted autoregressive constraint but uses a special form of the looking-back Markov blankets in these structures to reduce the number of network connections in comparison with MADE. The desirable property of these structures is that considering an appropriate order for their dimensions, we can select relatively small-sized looking-back Markov blankets of the form of

$$B_i = \{b_i, b_i + 1, b_i + 2, \ldots, i-1\} \quad \text{s.t. } b_i \geq b_{i-1} \quad \forall i \in \{1 \ldots d\}. \tag{6}$$

We will use the special form of the looking-back Markov blankets explained in (6) to propose the method. Note that a simple choice for b_i's is to set $b_i = 1$ for all i, but we prefer

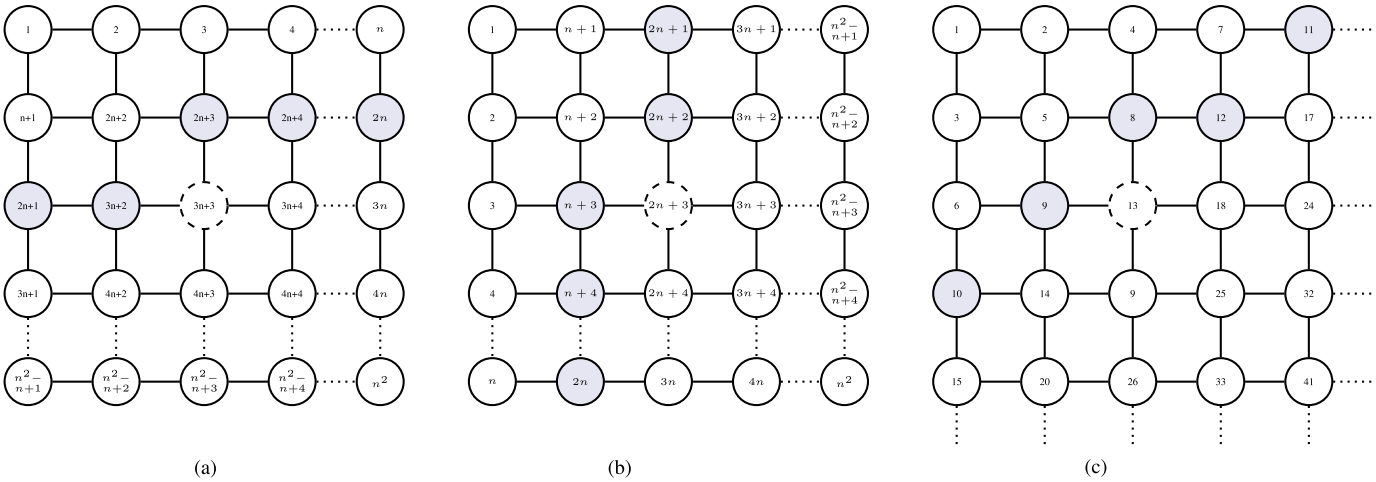

Fig. 1. Three appropriate dimension orders for a $n \times n$ grid. In each figure, the node with the dashed boundary is the intended node and the set of shaded nodes is the looking-back Markov blanket in the form of (6) for that node. (a) Horizontal order. (b) Vertical order. (c) Diagonal order.

Algorithm 2 Find Looking-Back Markov Blanket

```
procedure FIND-LOOKING-BACK-MARKOV-BLANKET(G, v)
    n ← size of G
    for i : 1 → n do
        visited[i] ← False
    end for
    DFS(v,visited,v)
    result ← {i|(i < v) & (visited[i] = True})
    return result
end procedure

procedure DFS(u, visited, v)
    visited[u] ← True
    if u < v then
        return
    end if
    for w ∈ neighbours(v) do
        if visited[w] = False then
            DFS(w,visited,v)
        end if
    end for
end procedure
```

the largest possible value for b_i, because it returns a smaller B_i and will mask more connections of the neural network in the approach that we are going to propose.

Fig. 1 shows three different appropriate orders of dimensions for a simple $n \times n$ grid that result in small-sized looking-back Markov blankets. It is clear that using any of these orders, the looking-back Markov blanket of each dimension will be of the form of (6) with the size of at most n. Furthermore, it is clear that using these orders, we have $b_i \geq b_{i-1}$ ($\forall i \in \{2, \ldots, n^2\}$). For instance, in Fig. 1(a) and (b), we have $b_i = \max(i - n, 1)$. Similar orders can be used for more complicated grid-based structures, with small modifications. As an example, in an $n \times n$ grid in which each pair of grid points are connected if their distance is less than or equal to 2, when we use horizontal or vertical orders [the same as the orders in Fig. 1(a) and (b)], we will have $b_i = \max(i - 2n, 1)$. We call these kinds of orders grid-based orders. Starting from different corners and using vertical, horizontal, or diagonal directions, we can consider 16 different grid-based orders.

Considering any permutation of dimensions, it is easy to find b_i for each $1 \leq i \leq d$. According to Theorem 1, the result of Algorithm 2 on the ith dimension is a looking-back Markov blanket of the ith dimension, and any other valid looking-back Markov blanket of i should contain all members of this looking-back Markov blanket. Therefore, it is clear that the smallest member of the looking-back Markov blanket returned by Algorithm 2 is the largest possible value for b_i.

Now, we propose the Masked Autoencoder for Grid Structured Distribution Estimation (MAGSDE) method. We guarantee that the autoregressive property is satisfied. However, we do not force the ith dimension of the output layer to depend only on B_i dimensions of the input layer, as we did in the MASDE method. Instead, according to the looking-back Markov blankets, we try to eliminate links in the network that just connect unrelated pairs of dimensions from the input layer and the output layer. MAGSDE is as follows.

1) Assign labels 1 to d to the input nodes.
2) Assign labels 1 to d to the output nodes.
3) Assign a random number from the set $\{1, 2, \ldots, d\}$ to each node in the hidden layers as its label.
4) Mask the connection from a node u with label j to node v with label i in the next layer, if $j > i$ or $j < b_i$. If v is in the output layer, the condition changes to $j \geq i$ or $j < b_i$.

Note that the labeling method of MAGSDE is the same as MADE. Also, the masking method is similar to MADE, with an extra case for masking a connection (if $j < b_i$). Therefore, every connection masked by MADE is also masked by MAGSDE. However, there are some extra connections masked by MAGSDE that MADE does not mask them. Here, using Theorem 2, we show that these extra connections just connect undesirable pairs of input and output dimensions. The proof is given in the Appendix.

Theorem 2: Consider a connection from node u to node v in the network. Assume that MAGSDE masks this connection, but MADE does not. If there exists a path from the kth input dimension to the tth output dimension in the network resulted by MADE (which are labeled as k and t, respectively) that passes through the connection from u to v then $k \notin B_t$.

Theorem 2 shows that unmasking the extra connections that are masked by MAGSDE versus MADE causes no new paths to any output dimension from its looking-back Markov blanket. Therefore, MAGSDE masks them to reduce the number of parameters of the model.

It is worth noting that MAGSDE tries to address the problem of small training size in a different way than MASDE. MAGSDE works based on reducing the complexity of the model, whereas MASDE works based on simplifying the regression problems that should be solved.

V. Experimental Results

We used two synthetic and two real data sets to evaluate the proposed methods and compared them with MADE and some other baseline methods. The synthetic data sets included two Ising models; a 4×4 grid-shaped model and a sparse 20-D MRF with less than 60 edges. We also compared different methods on the OCR-letters data set[3] (used in [7], [11], and [15] for the problem of density estimation) and the binarized MNIST (used in [15]), as two real data sets. We also compared our method with some state-of-the-art image density estimators on the (not binarized) MNIST data set.

We implemented MADE and our proposed methods in Python using Keras [22] library with Tensorflow [23] backend. All our source codes are available at http://github.com/ahmadkhajehnejad/structuredMADE.

Similar to [15], we executed an on-the-fly ensemble learning by making ten different sets of masks and using one set in each training epoch to update the parameters of the model. Also, we put direct connections from the input layer to the output layer (without violating the autoregressive property) in all the three models, as suggested in [15].

All the parameters of the synthetic data sets were sampled uniformly and independently of interval $(0, 1)$. For the synthetic data sets, since the true density function could be computed, we used the approximation of the Kullback–Leibler (KL) divergence between the estimated density for the test data and their true density as follows:

$$KL(P_t \parallel P_m) = E_{X\sim P_t}\left[\log \frac{P_t(X)}{P_m(X)}\right] \approx \frac{1}{M}\sum_{i=1}^{M} \log \frac{P_m(\mathbf{x}^{(i)})}{P_t(\mathbf{x}^{(i)})} \tag{7}$$

where P_t is the true distribution, P_m is the estimated distribution by the model, and $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ are the test data that are independent identically distributed (i.i.d) samples from P_t. We used $M = 5000$ in our experiments. Since the true distribution is unknown for real data sets, we cannot compute the KL-divergence criterion for evaluation on real data sets. Instead, we approximated the expected negative log-likelihood of the estimated distribution using an empirical average

$$E_{X\sim P_t}[-\log P_m(X)] \approx -\frac{1}{M}\sum_{i=1}^{M} \log P_m(\mathbf{x}^{(i)}) \tag{8}$$

where $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}$ are the set of test samples that are supposed to be sampled from the true distribution.

In each experiment, we start with a small training size and let it grow to investigate the performance of the methods in the presence of different sample sizes. We also set the validation size equal to one-fourth of the training size. The validation set is used for the early stopping of the training process, with a look-ahead of 20 epochs. We used Adam optimizer, proposed in [24], with learning rate $= 0.0003$ for binary data sets and 0.0001 for other cases, $\beta_0 = 0.1$, and a maximum of 2000 epochs.

For each experiment, we used a default of two hidden layers with ReLU activation functions. However, we also had a test on four and six hidden layers to investigate the impact of the number of hidden layers. We also investigated the impact of different hidden layer sizes. For the output nodes, the sigmoid activation functions were used. Due to the randomness of the methods, we report the average and standard deviation of the performance criteria over five independent executions on each training set.

Fig. 2 shows the results of MADE, MASDE, and MAGSDE on the 4×4 grid data set. It clearly shows that using different orders of dimensions for different ten masks (the middle column and the right column) improves the results of all the three methods versus using a fixed dimension order for all masks. Moreover, it shows that the proposed methods outperform MADE in all cases, especially for smaller training sizes. MASDE and MAGSDE obtain their best results when using grid-based dimension orders and wider hidden layers. On the other hand, MADE makes worse performance with wider hidden layers, especially for fixed dimension order and grid-based dimension orders. It seems that the simplifications that MASDE and MAGSDE imposed to the model have made them more robust than MADE to the size of the hidden layers.

Fig. 3 shows the superiority of MASDE against MADE on the sparse data set with 20 nodes. In order to show the impact of selecting the true looking-back Markov blankets in MASDE, we assumed an auxiliary method, the random blanket MASDE method (RBMASDE). RBMASDE is similar to MASDE, except that in this method for each dimension i, we substitute B_i (the looking-back Markov blanket of the ith dimension) with a random set $B'_i \subset \{1, 2, \ldots, i-1\}$ with the same size as B_i. The rest of the labeling and masking procedures of RBMASDE are exactly the same as MASDE such that the ith dimension of the output layer would be reachable just from dimensions in B'_i from the input layer. The results of RBMASDE are shown in the right column of Fig. 3. As anticipated, RBMASDE obtains worse results than MASDE because a random subset of $\{x_1, x_2, \ldots, x_{i-1}\}$ does not necessarily contain all the information of $P(x_i|x_1, x_2, \ldots, x_{i-1})$.

We also investigated the impact of the number of hidden layers on different methods when using grid-based dimension

[3]See http://ai.stanford.edu/~btaskar/ocr/

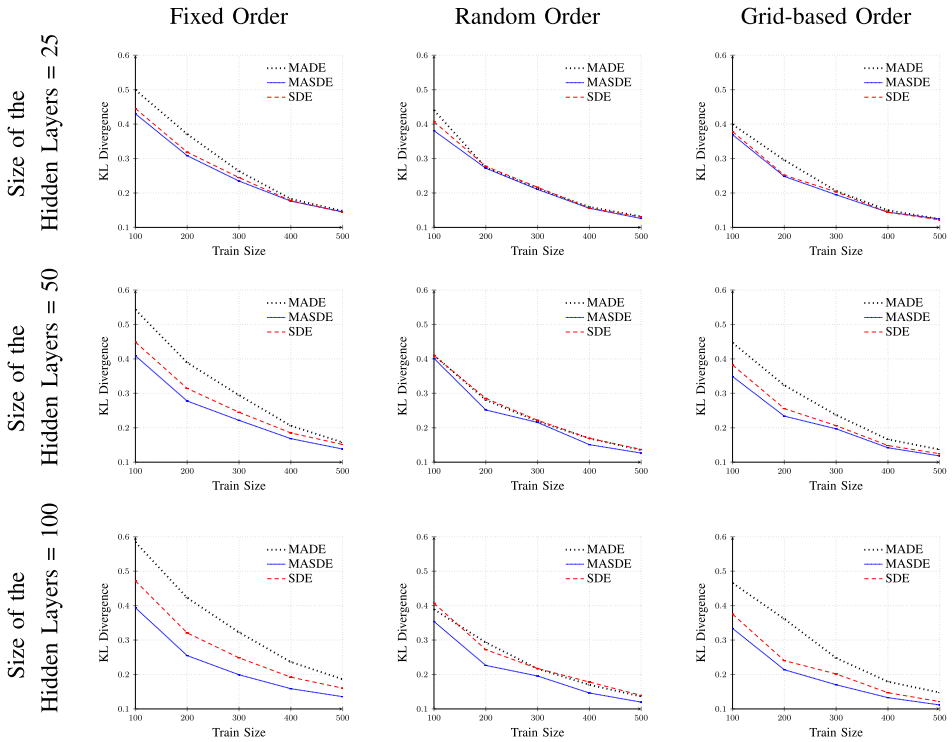

Fig. 2. Results of MADE, MASDE, and MAGSDE on the 4×4 grid data set, using a fixed (grid-based) order of dimensions (left), ten uniformly random permutations of dimensions (middle) and ten random selections among the 16 possible grid-based orders (right).

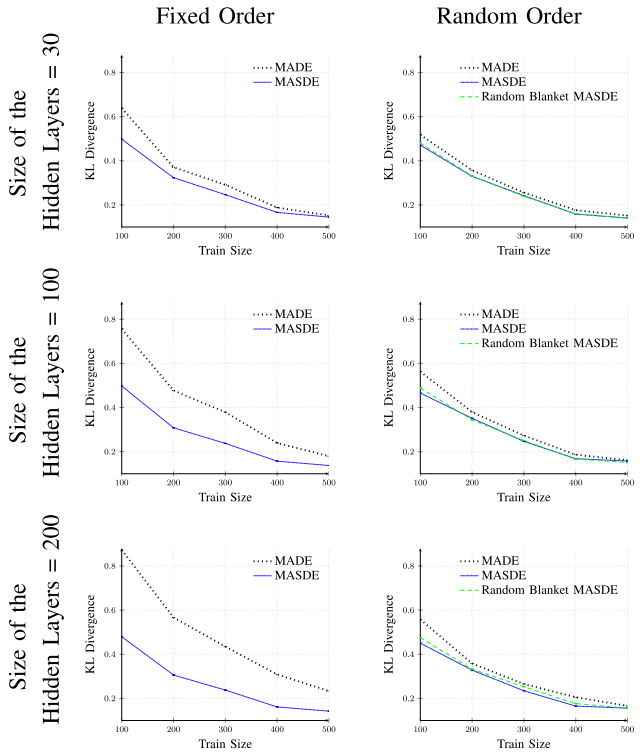

Fig. 3. Results of MADE, MASDE, and MASDE with random subsets instead of the looking-back Markov blankets, on the sparse 20 dimensional data set, using a fixed order of dimensions (left) and ten uniformly random permutations of dimensions (right).

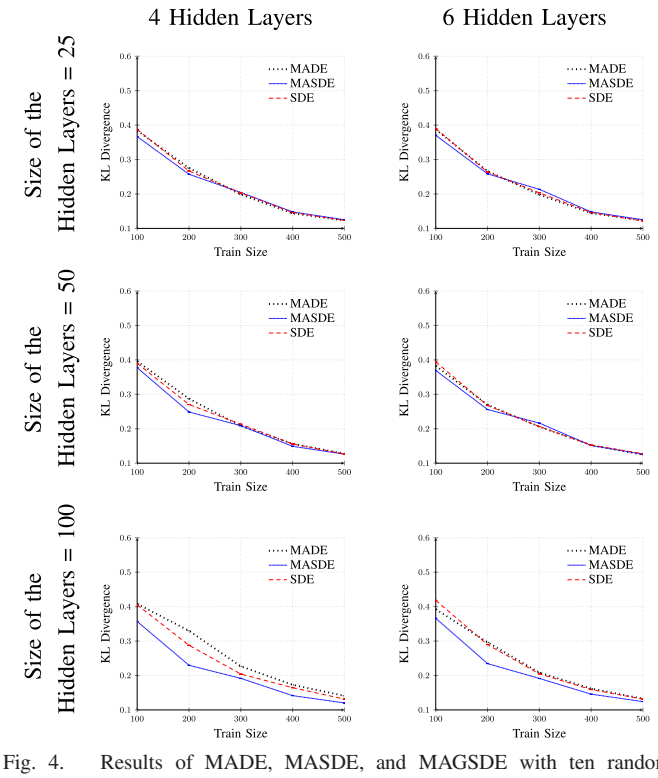

Fig. 4. Results of MADE, MASDE, and MAGSDE with ten random selections among the 16 possible grid-based dimension orders, on the 4×4 grid data set with four (left) and six (right) hidden layers.

orders on the 4×4 grid data set. As shown in Fig. 4, the performance of MASDE and MAGSDE does not have a considerable change by increasing the number of hidden layers, but MADE has improved and also has become more robust to the changes in the size of the hidden layers, despite that the number of its parameters has been increased (comparing the right column of Fig. 2 with Fig. 4). It shows that the

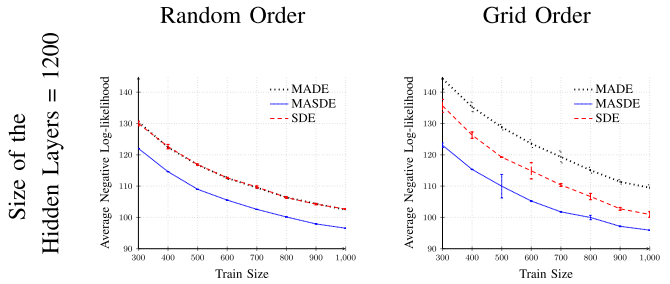

Fig. 5. Results of MADE, MASDE, and MAGSDE on binarized MNIST, using ten uniformly random permutations of dimensions (left) and ten random selections from the 16 possible grid-based orders (right).

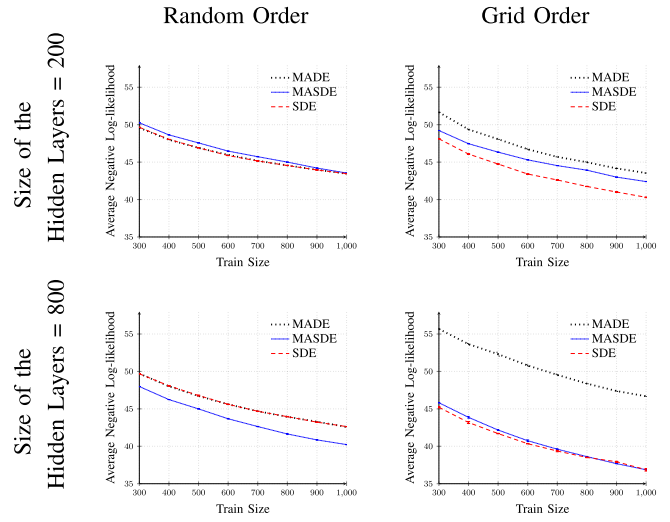

Fig. 6. Results of MADE, MASDE, and MAGSDE on OCR-letters, using ten uniformly random permutations of dimensions (left) and ten random selections from the 16 possible grid-based orders (right).

number of network parameters is not an accurate measure of the complexity of MADE.

Figs. 5 and 6 show the results of different methods on binarized MNIST and OCR-letters data sets, respectively. MNIST images are 28×28 pixels and OCR images are 16×8 pixels. Since our proposed methods need to know the structure, in MNIST, we considered that each pixel is connected to all the pixels of a 9×9 square surrounding that pixel. In OCR, we considered a square of size 3×3 as the neighbors of each pixel. We observe from Figs. 5 and 6 that on both data sets, when we run MADE, uniform random dimension orders are more proper than grid-based orders. Although MAGSDE cannot make any improvement against MADE when using uniformly random dimension orders, when we use grid-based orders, MAGSDE can drastically improve MADE. This improvement on the OCR data set is large enough to outperform MADE with uniformly random orders, but on MNIST, it is not. As can be seen from Figs. 5 and 6, MASDE outperforms MADE if the size of the hidden layers is large enough. For MNIST, with 784 dimensions, a size of $1200 \approx 784 \times 1.5$ for hidden layers is sufficient enough to let the MASDE method outperform MADE, but for OCR data set, with 128 dimensions, MASDE needs hidden layers larger than $200 \approx 128 \times 1.5$ to beat MADE.

TABLE I
KL ON GRID 4×4 DATA SET

Algorithm	100 samples	10000 samples
MoBernoullis (k=10)	0.67	0.07
Centered RBM (h=100, 100 CD steps)	0.53	0.17
Orderless NADE (2 hidden layers)	0.52	0.40
FVSBN	0.46	0.006
MADE (random order)	0.39	0.006
MASDE (grid-based order)	0.33	0.006
MAGSDE (grid-based order)	0.37	0.006

TABLE II
KL ON THE SPARSE 20-D DATA SET

Algorithm	100 samples	10000 samples
MoBernoullis k=20	0.72	0.13
Centered RBM h=200, 100 CD steps	0.64	0.47
Orderless NADE 2 hidden layers	0.65	0.56
FVSBN	0.55	0.008
MADE random order	0.55	0.01
MASDE random order	0.45	0.009

We also compared the proposed methods with some other binary density estimators in two cases of using small and large training data. We selected the baseline models according to the experiments reported in [15]. We used a mixture of multivariate Bernollis, a centered RBM model [25], an autoregressive model with sigmoid conditional probabilities which was inspired by an FVSBN used in [10], the orderless neural autoencoder density estimator (NADE) [14], and the original MADE [15]. We used the validation data in all methods (except for centered RBM) to retain them from overfitting. We again state that we claim the superiority of our proposed methods just for small training sets, because in the presence of large training sets, deep models can estimate the density accurately without knowing any extra structural information. Tables I–IV shows the results of these comparisons. We observe that except for MAGSDE with grid-based dimension orders on MNIST, the proposed methods outperform all the other methods when trained on the small data sets. Furthermore, their performance is approximately equal to MADE on the large data sets. This confirms that the simplifications that we apply on the proposed methods make them proper for learning with small data, without restricting their capacity to estimate complex functions in the presence of large data sets. It is worth noting that since MASDE performs better with large hidden layers, we used two hidden layers of 1200 units with ten different masks when testing on the binarized MNIST data set. We also used the same settings for MADE and MAGSDE for the sake of fairness. These settings were different from the settings used in [15, Sec. 6.2] (two hidden layers of size 800 and 32 masks). We also tested our implementation of MADE with the same settings as in [15]

TABLE III
NLL ON THE BINARIZED MNIST DATA SET

Algorithm	300 samples	50000 samples
MoBernoullis k=500	196.79	133.29
Centered RBM h=1200, 100 CD steps	175.61	86.06
Orderless NADE 2 hidden layers	146.11	85.19
FVSBN	132.09	75.56
MADE random order	130.55	68.59
MASDE grid-based order	123.06	72.99
MAGSDE grid-based order	135.65	65.34

TABLE IV
KL ON THE OCR DATA SET

Algorithm	200 samples	32152 samples
MoBernoullis k=500	61.39	44.05
Centered RBM h=800, 100 CD steps	65.97	42.74
Orderless NADE 2 hidden layers	51.77	39.56
FVSBN	51.56	40.83
MADE random order	49.71	30.08
MASDE grid-based order	45.82	30.05
MAGSDE grid-based order	45.21	27.49

TABLE V
NLL ON THE (NOT BINARIZED) MNIST DATA SET

Algorithm	100 samples	300 samples	1000 samples
MADE random order	1313.00	1102.66	1041.31
MASDE grid-based order	1013.01	917.26	889.44
MAGSDE grid-based order	1102.75	1013.59	920.44
RealNVP	1254.42	1019.00	783.75
PixelCNN++ random order	695.12	677.64	627.60

on 50 000 samples from binarized MNIST and obtained a better negative loglikelihood (75.48) than [15] (86.64).

In order to compare the proposed models with recent deep models such as RealNVP [18] and PixelCNN++ [26], we used the (not binarized) MNIST data set. Similar to PixelCNN++, we used a logistic mixture model in the last layer of MADE, MAGSDE, and MASDE to precisely estimate the discrete distribution of each pixel. Table V shows the results. Since RealNVP is proposed for continuous distributions, we approximated its negative loglikelihood for discrete values of pixels using a uniform noisy sampling (see [27, Sec. 3.1]), which gives an upper bound of the true negative log-likelihood of the discrete data. Both RealNVP and Pixelcnn++ use convolutional layers, which is a way of using structural information. However, these models are adapted to images and grid-based structures and cannot be applied to other structures. On the other hand, MADE is a simpler model, and according to Table V, it cannot compete RealNVP and PixelCNN++ on image data sets. However, this experiment confirms that MASDE and MAGSDE partly compensate for MADE's weakness on small-sized image data sets.

VI. CONCLUSION

In this article, we proposed two autoregressive autoencoders for estimating structured distributions from a small set of observations. Similar to the MADE, the proposed methods mask some connections of the fully connected layers of a simple autoencoder to satisfy the autoregressive property of the network. The proposed methods also block the paths between independent dimensions from the input layer to the output layer. The independencies are induced by the known MRF structure of the distribution. The first method, Masked Autoencoder for Structured Distribution Estimation (MASDE), outperforms MADE in all the experiments; however, in some cases, it needs larger hidden layers than MADE. The second proposed method, Masked Autoencoder for Grid Structured Distribution Estimation (MAGSDE), is dedicated to the grid-based MRFs. The improvement caused by MAGSDE over MADE depends on the order of the dimensions that we use. The experimental results showed that in some data sets, MAGSDE did not improve the best results obtained by MADE. Our experiments also showed that our proposed methods can partly compensate for the shortcomings of MADE against state-of-the-art deep image density estimators when trained on small training sets.

APPENDIX

A. Proof of Theorem 1

In a Markov random field, node v is independent of node u given the subset W of nodes, if and only if each path between v and u contains at least one node in W (or is blocked by W) [3]. Algorithm 2 traverses the graph through a depth first search, starting from node v, and traces back each time it visits a node in $\{1, 2, \ldots, v-1\}$. Let us decompose the set $\{1, 2, \ldots, v-1\}$ into the visited nodes by Algorithm 2 (A) and unreached nodes (B). Notice that A is the output of the Algorithm 2. Regarding to the fact that each node in A is reachable from v using a path that passes through the nodes in $\{v+1, v+2, \ldots, d\}$, it is clear that these paths could not be blocked by any subset of $\{1, 2, \ldots, v-1\}$. Hence, any looking-back Markov blanket of v must contain all members of A. On the other hand, it is clear that each path between v and a node in B is blocked by A, otherwise, that node would be visited in the depth first search by Algorithm 2. So A is a looking-back Markov blanket for v.

B. Proof of Theorem 2

Assume that node u is labeled as l and node v is labeled as r. Since MAGSDE masks the connection, but MADE does not, we have

$$l < b_r. \tag{9}$$

From the masking condition of MADE, we know that the labels on each path in the network are non-descending. Therefore we have

$$k \leq l \leq r \leq t. \tag{10}$$

From Equations (9) and (10) it is clear that

$$k < b_r. \tag{11}$$

Also, according to the condition $b_i \geq b_{i-1}$ in Equation (6) and from Equation (10) we can infer that

$$b_r \leq b_t. \tag{12}$$

Therefore, using Equations (11) and (12) we have

$$k < b_t \tag{13}$$

According to Equation (13) and the definition of B_i in Equation (6) it is clear that $k \notin B_t$.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions which significantly improved this article. They also thank Dr. Mohammad Hossein Rohban for useful discussions. Some experiments of this research have been run using the computing service provided by the High-Performance Computing Center of IPM Institute for Research in Fundamental Sciences.

REFERENCES

[1] A. C. Davison, *Statistical Models*, vol. 11. Cambridge, U.K.: Cambridge Univ. Press, 2003, ch. 7.1.2.
[2] X. Zhuang, Y. Huang, K. Palaniappan, and Y. Zhao, "Gaussian mixture density modeling, decomposition, and applications," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1293–1302, 1996.
[3] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.
[4] Y. Liu, D. Xie, and X. Wang, "Generalized Boltzmann machine with deep neural structure," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 926–934.
[5] E. Janofsky, "Exponential series approaches for nonparametric graphical models," 2015, *arXiv:1506.03537*. [Online]. Available: http://arxiv.org/abs/1506.03537
[6] G. Lugosi, J. Truszkowski, V. Velona, and P. Zwiernik, "Structure learning in graphical models by covariance queries," 2019, *arXiv:1906.09501*. [Online]. Available: http://arxiv.org/abs/1906.09501
[7] H. Larochelle, Y. Bengio, and J. Turian, "Tractable multivariate binary density estimation and the restricted Boltzmann forest," *Neural Comput.*, vol. 22, no. 9, pp. 2285–2307, Sep. 2010.
[8] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[9] Y. Bengio and S. Bengio, "Modeling high-dimensional discrete data with multi-layer neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 400–406.
[10] B. J. Frey, G. E. Hinton, and P. Dayan, "Does the wake-sleep algorithm produce good density estimators?" in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 661–667.
[11] H. Larochelle and I. Murray, "The neural autoregressive distribution estimator," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 29–37.
[12] B. Uria, I. Murray, and H. Larochelle, "RNADE: The real-valued neural autoregressive density-estimator," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2175–2183.
[13] B. Uria, I. Murray, and H. Larochelle, "A deep and tractable density estimator," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 467–475.
[14] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle, "Neural autoregressive distribution estimation," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 7184–7220, 2016.
[15] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 881–889.
[16] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear independent components estimation," 2015, *arXiv:1410.8516*. [Online]. Available: http://arxiv.org/abs/1410.8516
[17] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1530–1538.
[18] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017.
[19] G. Papamakarios, I. Murray, and T. Pavlakou, "Masked autoregressive flow for density estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2335–2344.
[20] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2014, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[21] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[22] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://keras.io
[23] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[25] J. Melchior, A. Fischer, and L. Wiskott, "How to center deep Boltzmann machines," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 3387–3447, 2016.
[26] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017.
[27] L. Theis, A. van den Oord, and M. Bethge, "A note on the evaluation of generative models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–10.

Ahmad Khajenezhad received the B.Sc. and M.Sc. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2010 and 2012, respectively, where he is currently pursuing the Ph.D. degree.

His research interest includes learning probabilistic models for structured/distributed data.

Hatef Madani received the B.Sc. degree in computer engineering from the Kharazmi University of Tehran, Tehran, Iran, in 2014, and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, in 2018.

He is interested in image processing and deep approaches in machine learning.

Hamid Beigy received the B.Sc. and M.Sc. degrees in computer engineering from the Shiraz University, Shiraz, Iran, in 1992 and 1995, respectively, and the Ph.D. degree in computer engineering from the Amirkabir University of Technology, Iran, in 2004.

He is currently an Associate Professor with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran. His research interests include theoretical machine learning, large-scale machine learning, and social networks.