Towards Interpretable Image Synthesis by Learning Sparsely Connected AND-OR Networks

Xianglei Xing, Tianfu Wu, Song-Chun Zhu, and Ying Nian Wu

arXiv:1909.04324v1 [cs.CV] 10 Sep 2019

Abstract—This paper proposes interpretable image synthesis by learning hierarchical AND-OR networks of sparsely connected semantically meaningful nodes. The proposed method is based on the compositionality and interpretability of scene-objects-parts-subparts-primitives hierarchy in image representation. A scene has different types (i.e., OR) each of which consists of a number of objects (i.e., AND). This can be recursively formulated across the scene-objects-parts-subparts hierarchy and is terminated at the primitive level (e.g., Gabor wavelets-like basis). To realize this interpretable AND-OR hierarchy in image synthesis, the proposed method consists of two components: (i) Each layer of the hierarchy is represented by an over-completed set of basis functions. The basis functions are instantiated using convolution to be translation covariant. Off-the-shelf convolutional neural architectures are then exploited to implement the hierarchy. (ii) Sparsity-inducing constraints are introduced in end-to-end training, which facilitate a sparsely connected AND-OR network to emerge from initially densely connected convolutional neural networks. A straightforward sparsity-inducing constraint is utilized, that is to only allow the top-k basis functions to be active at each layer (where k is a hyperparameter). The learned basis functions are also capable of image reconstruction to explain away input images. In experiments, the proposed method is tested on five benchmark datasets. The results show that meaningful and interpretable hierarchical representations are learned with better qualities of image synthesis and reconstruction obtained than state-of-the-art baselines.

Index Terms—Sparse coding, Generator network, Alternating back-propagation, Generative models, Unsupervised learning

1 Introduction

Remarkable recent progress on image synthesis [1], [2], [3], [4], [5], [6] have been made using deep neural networks (DNNs) [7], [8]. Most efforts focus on developing sophisticated architectures and training paradigms for sharp and realistic-looking image synthesis [3], [9], [10]. Although high-fidelity images can be generated, the internal synthesizing process via DNNs is still largely viewed as a black-box, thus potentially hindering the long-term applicability in eXplainable AI (XAI) [11]. More recently, the generative adversarial network (GAN) dissection method [12] has been proposed to identify internal neurons in pre-trained GANs that show interpretable meanings using a separate annotated dataset in a post-hoc fashion.

In this paper, we focus on learning interpretable models for unconditional image synthesis from scratch with explicit hierarchical representations. By interpretable image synthesis, it means the internal image generation process can be explicitly unfolded through meaningful basis functions end-to-end learned at different layers and conceptually reflecting the hierarchy of scene-objects-parts-subparts-primitives. A scene has different types (i.e., OR) each of which consists of a number of objects (i.e., AND). This can be recursively formulated across the scene-objects-parts-subparts hierarchy and is terminated at the primitive level (e.g., Gabor wavelets-like basis functions). Figure 1 shows an example of the AND-OR tree learned from scratch for explaining a generating face image.

The hierarchy of scene-objects-parts-subparts-primitives is at the stem of image grammar models [13], [14]. The AND-OR compositionality has been applied in image vision tasks [14]. With the recent resurgence of deep neural networks (DNNs) [7], [8] and the more recent DNN-based image synthesis framework such as the widely used Generative Adversarial Networks (GANs) [1] and Variational Auto-Encoder (VAE) methods [15], [16], the hierarchy is usually assumed to be modeled implicitly in DNNs. Due to dense connections between consecutive layers in traditional DNNs, they often learn noisy compositional patterns of how entities in a layer are formed from "smaller" ones in the layer right below it.

On the other hand, the sparsity principle has played a fundamental role in high-dimensional statistics, machine learning, signal processing and AI. In particular, the sparse coding scheme [17] is an important principle for understanding the visual cortex. By imposing sparsity constraints on the coefficients of a linear generative model, [18] learn Gabor-like wavelets that resemble the neurons in the primary visual cortex (V1) from natural image patches. Since then, there are many important work on sparse coding presented in the literature before the resurgence of DNNs.

- *X. Xing is with the College of Automation, Harbin Engineering University, Heilongjiang, China, 150001.E-mail: xingxl@hrbeu.edu.cn*
- *T. Wu is with Department of Electrical and Computer Engineering and the Visual Narrative Initiative, North Carolina State University, Raleigh, North Carolina 27695.E-mail:tianfu_wu@ncsu.edu*
- *S.-C. Zhu and Y. N. Wu are with Department of statistics, University of California, Los Angeles, California 90095.E-mail: {sczhu,ywu}@stat.ucla.edu*

The work was done while the first author worked as a visiting scholar at UCLA.

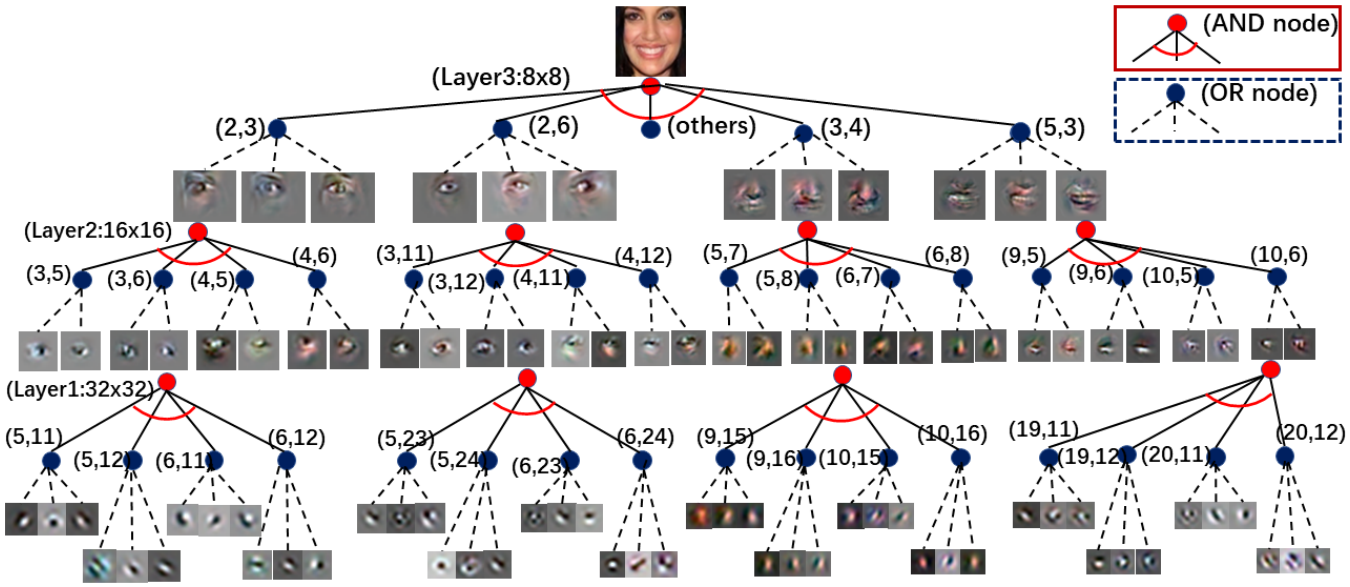

Fig. 1. An example of AND-OR tree learned for faces. For clarity, we only show 3 layers (out of the total 5 layers). On the top, we show a synthesized face at the resolution of 64×64. A 3-layer AND-OR tree is illustrated from Layer 3 (the part/composite part level) to Layer 1 (the primitive level). The spatial resolution of the feature map at Layer 3 is 8×8. The entire grid is interpreted by a AND-node. Each position (x, y) in the 8×8 grid is interpreted by an OR-node such as the eye node at the positions $(2, 3)$ and $(2, 6)$. Similarly, Layer 2 and Layer 1 can be interpreted w.r.t. the AND-OR compositions. The activated basis functions have semantically meaningful interpretations at Layer 3 and Layer 2. Layer 1 shows the learned primitives covering the classic Gabor-like wavelets and the blob-like primitives. See text for detail. Best viewed in color.

With the remarkable successes of sparse coding models, it is not unreasonable to assume that a top-down generative model of natural images should be based on the linear sparse coding model, or incorporate the sparse coding principle at all of its layers. However, developing a top-down sparse coding model that can generate, rather than merely reconstruct, realistic-looking natural image patterns has proven to be a difficult task [19], mainly due to the difficulty of selecting and fitting sparse basis functions to each image.

In this paper, we take a step forward rethinking dense connections between consecutive layers in traditional DNNs. We propose to "re-wire" them sparsely for explicit modeling of the hierarchy of scene-objects-parts-subparts-primitives in image synthesis (see Figure 1. To realize the "re-wiring", we integrate the sparsity principle in DNNs in a simple yet effective and adaptive way: (i) Each layer of the hierarchy is represented by a (over-completed) set of basis functions. The basis functions are instantiated using convolution to be translation covariant. Off-the-shelf convolutional neural architectures are then exploited to implement the hierarchy such as generator networks used in GANs. (ii) Sparsity-inducing constraints are introduced in end-to-end training which facilitates a sparsely connected AND-OR network to emerge from initially densely connected convolutional neural networks. A straightforward sparsity-inducing constraint is utilized, that is to only allow the top-k basis functions to fire at each layer (where k is a hyperparameter). By doing so, we can harness the highly expressive modeling capability and the end-to-end learning flexibility of DNNs, and the interpertability rigor of the explicit compositional hierarchy.

2 Related Works

Sparse-autoencoders [20], [21], [22] were proposed for effective feature representations and these representations can improve the performance of the classification task. The sparsity constrains are designed and encouraged by the Kullback-Leibler divergence between the Bernoulli random variables [20], l_1 penalty on the normalized features [23], and winner-take-all principle [24]. However, these methods do not have the ability of generating new data. Lee [25], [26] proposed a convolutional deep belief networks which employ sparsity regularization and probabilistic max-pooling to learn hierarchical representations. However, the learning is difficult and computationally expensive for training the deep belief nets. Zeiler [27], [28] proposed the deconvolutional networks to learn the low and mid-level image representations based on the convolutional decomposition of images under a sparsity constrain. However, for the aforementioned methods, the hierarchical representations have to be learned layer by layer, that is to first train the bottom layer of the network and then fix the learned layer and train the upper layers one by one. Moreover, the above methods usually work on the gray images or the gradient images which are preprocessed by removing the low frequency texture information and highlighting the structure information. Unlike the above method, the proposed method can directly work on the raw color images without any preprocessing. The proposed model can simultaneously learn meaningful hierarchical representations, generate realistic images and reconstruct the original images.

Our Contributions. This paper makes three main contributions to the field of generative learning: (i) It proposes interpretable image synthesis that unfolds the internal generation process via a hierarchical AND-OR network of semantically meaningful nodes. (ii) It presents a simple

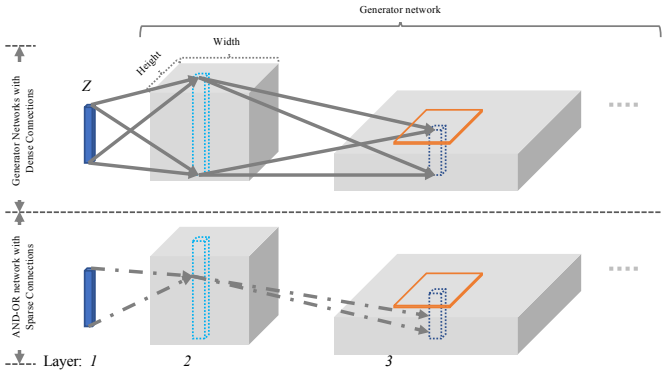

Fig. 2. *Top*: traditional generator networks with dense connections (solid arrows) between consecutive layers, which are widely used in GANs and VAE. *Bottom*: the proposed AND-OR networks with sparse connections (dashed arrows). See text for detail.

yet effective sparsity-inducing method that facilitates a hierarchical AND-OR network of sparsely connected nodes to emerge from an initial network of dense connection between consecutive layers. (iii) It shows that meaningful hierarchical representations can be learned end-to-end in image synthesis with better qualities than state-of-the-art baselines.

3 The Proposed Approach

3.1 Image Synthesis and Model Interpretability

From the viewpoint of top-down generative learning in image synthesis, we start with a d-dimensional latent code vector $Z = (z_i, i = 1, \cdots, d)$ consisting of d latent factors. We usually assume $Z \sim \mathcal{N}(0, I_d)$, where I_d represents the d-dimensional identity matrix. In GANs and VAE, generator networks are used to implement the highly non-linear mapping from a latent code vector Z to a synthesized image, denoted by Y which lies in a D-dimensional image space (i.e., D equals the product of the spatial dimensions, width and height of an image, and the number of chromatic channels such as 3 for RGB images). The generator networks are thus seen as non-linear extensions of factor analysis [29]. We have,

$$\begin{aligned} &Y = g(Z; \Theta) + \epsilon, \\ &Z \sim \mathcal{N}(0, I_d), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I_D), \quad d < D \end{aligned} \tag{1}$$

where ϵ is the observational errors assumed to be Gaussian white noises, $g(\cdot)$ represents the generator network and Θ collects parameters from all layers.

As illustrated in the top of Figure 2, dense connections between consecutive layers are learned in the vanilla generator network, which we think is the main drawback that hinders explicit model intepretability. We explore and exploit the AND-OR compositionality in image synthesis by learning to rewire the connections sparsely and to unfold the internal image generation process in an interpretable way, as illustrated in the bottom of Figure 2.

3.2 The Proposed AND-OR Network

Without loss of generality, consider a simple hierarchy of object(O)-part(P)-primitive/basis(B) that generates RGB images. Start with the latent code vector $Z \in R^d$, we have,

$$\text{Hierarchy: } Z \rightarrow \quad O \rightarrow \quad P \rightarrow \quad B \rightarrow \quad Y, \tag{2}$$

$$\text{Layer Index: } 1, \quad 2, \quad 3, \quad 4, \quad 5. \tag{3}$$

For example, Figure 2 illustrates the computing flow from Layer 1 to Layer 3.

The symbol O in the hierarchy is grounded in an internal $(H_2 \times W_2 \times d_2)$-dimensional space, which can be treated as $H_2 \times W_2$ d_2-dimensional vectors when instantiated. Similarly, the symbols P and B will be instantiated as $H_3 \times W_3$ d_3-dimensional vectors and $H_4 \times W_4$ d_4-dimensional vectors respectively. Y is a generated RGB image of size $H_5 \times W_5 \times 3$.

To better show how to facilitate the sparse connections to emerge from the dense ones, we look at the computing flow using the lens of vector-matrix multiplication [30]. In the vanilla generator network, consider a d_l-dimensional vector, U in Layer l, it connects to a set of d_{l+1}-dimensional vectors, V_i's in Layer $l+1$. Let $ch(U)$ be the set of indices of the vectors in Layer $l+1$ which connect with U (i.e., U's child nodes). We have,

$$V_i(U) = W_{d_{l+1} \times d_l} \cdot U + b, \quad i \in ch(U) \tag{4}$$

where $V_i(U)$ means the contribution of U to V_i since there may have other vectors U' in Layer l connecting to V_i too. $W_{d_{l+1} \times d_l}$ is the transformation matrix and b the bias vector. Consider Layer 1 to Layer 2 ($Z \rightarrow O$), $U = Z$ is connected with all vectors V_i's in O with different $W_{d_{l+1} \times d_l}$'s and b's. Consider Layer 2 to Layer 3 ($O \rightarrow P$), convolution is usually used, so each U only connects to vectors V_i's locally, and $W_{d_{l+1} \times d_l}$'s and b's are shared among different U's.

Denote by $pr(V_i)$ the set of indices of vectors in Layer l connecting with V_i. In the vanilla generator network,

$$V_i = act(\sum_{U \in pr(V_i)} V_i(U)), \tag{5}$$

where $act(\cdot)$ stands for activation function such as the ReLU function [8].

In the proposed method, we compute V_i by,

$$V_i = S(act(\sum_{U \in pr(V_i)} V_i(U)); k_{l+1}), \tag{6}$$

where $S(\cdot; k_{l+1})$ is the sparsity-inducing function. From symbol Z to O, we apply the sparsity-inducing function along the d_{l+1} dimension and retain the top k_{l+1} out of d_{l+1} elements in the resulting vector in terms of the element values. In the subsequent layers, we apply it along the spatial domain across the d_{l+1} dimensions individually. By doing so, the resulting vectors at different location will have different sparsity ratios. The k_l's are hyper-parameters. We usually set $d_l > d_{l+1}$ and $k_l < k_{l+1}$, that is, Layer l has higher sparsity degree than lower Layer $l+1$.

With sparsity-inducing functions, image synthesis is fundamentally changed in terms of representation. The internal

generation process is also much easier to unfold. The proposed AND-OR network is emerged from the vanilla dense connected generator network. We can rewrite Eqn. 1 as,

$$\begin{aligned} &Y = g(Z; \Theta, \mathbf{k}) + \epsilon, \\ &Z \sim \mathcal{N}(0, I_d), \quad \epsilon \sim \mathcal{N}(0, \sigma^2 I_D), \quad d < D \end{aligned} \tag{7}$$

where the sparsity hyper-parameters $\mathbf{k} = \{k_l; l = 1, \cdots, L\}$. We summarize the proposed AND-OR network for image synthesis as follows.

Layer 1 to Layer 2: $Z \rightarrow O$. The latent code vector Z is represented by a root OR-node (non-terminal symbol),

$$Z \xrightarrow{\text{OR}} z_1 | z_2 | \cdots | z_i | \cdots, \quad z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I_d), \tag{8}$$

where $a|b$ denotes OR switching between symbols a and b (i.e., instantiated latent code vectors that generate different object images).

Each instantiated latent code vector z_i is then mapped to an object instance AND-node O_i. The object instance AND-node O_i represents the object-part decomposition in the lattice Λ_2 (of size $H_2 \times W_2$). We have,

$$O_i \xrightarrow{\text{AND}} o_{i,1} \cdot o_{i,2} \cdot \cdots \cdot o_{i,j} \cdot \cdots \cdot o_{i,N_P} \tag{9}$$

where $a \cdot b$ represents the composition between symbols a and b. N_P is the number of part symbols. The object-part decomposition is usually done in the spatial domain. For example, if the support domain for o_i's is 4×4, we will have at most 16 parts. We could use 4 parts if we further divide the 4×4 domain into 2×2 blocks.

Each $o_{i,j}$ is then represented by an OR-node in the d_2-dimensional vector space indicating the sparse selection among $\binom{d_2}{k_2}$ candidates. When instantiated, we have part AND-node $o_{i,j}(k_2)$.

Layer 2 to Layer 3: $O \rightarrow P$. Each part AND-node $o_{i,j}(k_2)$ is decomposed into a number of M child part type OR-nodes,

$$o_{i,j}(k_3) \xrightarrow{\text{AND}} P_{i,j,1} \cdot P_{i,j,2} \cdot \cdots \cdot P_{i,j,M}, \tag{10}$$

where M is determined by the kernel size when convolution is used to compute Layer 3 from Layer 2.

Similarly, each part type OR-node $P_{i,j,t}$ is grounded in the d_3-dimensional vector space indicating the sparse selection among $\binom{d_3}{k_3}$ candidates. When instantiated, we have part-primitive AND-node. Then, the AND-OR is recursively formulated in the downstream layers. Now, let us look at Figure 2 again, for each instantiated z_i, we can follow the sparse connections and visualize the encountered kernel symbols (see Figure 1).

3.3 Learning and Inference

The proposed AND-OR network can still utilize off-the-shelf end-to-end learning framework since the sparsity-inducing terms do not change the formulation (Eqn. 7). We adopt the alternating back-propagation learning framework proposed in [29].

Denote by $\{Y_i, i = 1, \ldots, N\}$ the training dataset consisting of N images (e.g., face images). The learning objective is to maximize the observed data log-likelihood,

$$\begin{aligned} L(\Theta) &= \frac{1}{N} \sum_{i=1}^{N} \log P(Y_i; \Theta, \mathbf{k}) \\ &= \frac{1}{N} \sum_{i=1}^{N} \log \int P(Y_i, Z_i; \Theta, \mathbf{k}) dZ_i. \end{aligned} \tag{11}$$

where the latent vector Z_i for an observed data Y_i is integrated out, and $P(Y_i, Z_i; \Theta, \mathbf{k})$ the complete-data likelihood. The gradient of $L(\Theta)$ is computed as follows,

$$\begin{aligned} &\frac{\partial}{\partial \Theta} \log P(Y; \Theta, \mathbf{k}) \\ = &\frac{1}{P(Y; \Theta, \mathbf{k})} \frac{\partial}{\partial \Theta} \int P(Y, Z; \Theta, \mathbf{k}) dZ \\ = &\mathrm{E}_{P(Z|Y;\Theta,\mathbf{k})} \left[\frac{\partial}{\partial \Theta} \log P(Y, Z; \Theta, \mathbf{k}) \right]. \end{aligned} \tag{12}$$

In general, the expectation in Eqn.12 is analytically intractable. Monte Carlo average is usually adopted in practice with samples drawn from the posterior $P(Z|Y; \Theta, \mathbf{k})$ by the Langevin dynamics,

$$Z_{\tau+1} = Z_\tau + \frac{\delta^2}{2} \frac{\partial}{\partial Z} \log P(Z_\tau, Y; \Theta, \mathbf{k}) + \delta \mathcal{E}_\tau, \tag{13}$$

where τ indexes the time step, δ is the step size, and $\mathcal{E}_\tau$ denotes the noise term, $\mathcal{E}_\tau \sim \mathrm{N}(0, I_d)$.

Based on Eqn. 7, the complete-data log-likelihood is computed by,

$$\begin{aligned} &\log p(Y, Z; \Theta, \mathbf{k}) = \log \left[p(Z) p(Y|Z; \Theta, \mathbf{k}) \right] \\ &= -\frac{1}{2\sigma^2} \|Y - g(Z; \Theta, \mathbf{k})\|^2 - \frac{1}{2} \|Z\|^2 + C \end{aligned} \tag{14}$$

where C is a constant term independent of Z and Y. It can be shown that, given sufficient transition steps, the Z obtained from this procedure follows the joint posterior distribution [31]. For each training example Y_i, we run the Langevin dynamics in Eqn.13 to get the corresponding posterior sample Z_i. The sample is then used for gradient computation in Eqn.12. The parameters Θ are then learned through Monte Carlo approximation,

$$\begin{aligned} &\frac{\partial}{\partial \theta} L(\Theta) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \Theta} \log p(Y_i, Z_i; \Theta, \mathbf{k}) \\ &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\sigma^2} (Y_i - g(Z_i; \Theta, \mathbf{k})) \frac{\partial}{\partial \Theta} g(Z_i; \Theta, \mathbf{k}). \end{aligned} \tag{15}$$

3.4 Combining with an Energy-Based Network

It is well known that using squared Euclidean distance alone to train generator networks often yields blurry reconstruction results, since the precise location information of details may not be preserved, and the L_2 loss in the image space leads to averaging effects among all likely locations. In order to improve the quality, we utilize an energy-based network to help the generator network. The energy-based model is in the form of exponential tilting of a reference distribution of observed data,

$$P(Y; \Phi) = \frac{1}{Z(\Phi)} \exp\left[-f(Y; \Phi)\right] q(Y), \tag{16}$$

where $f(Y;\Phi)$ is parameterized by a bottom-up ConvNet which maps an image Y to the feature statistics or energy, $Z(\Phi) = \int \exp\left[f(Y;\Phi)\right] q(Y)dY = \mathrm{E}_q\{\exp[f(Y;\Phi)]\}$ the normalizing constant, and $q(Y)$ is the reference distribution such as Gaussian white noise,

$$q(Y) = \frac{1}{(2\pi\sigma^2)^{D/2}} \exp\left[-\frac{\|Y\|^2}{2\sigma^2}\right]. \tag{17}$$

Let $P(Y)$ be the underlying true data distribution. We jointly learn the generator network and the energy-based network by,

$$\min_{\theta} \max_{\Phi} KL(P(Y;\Phi)|P(Y;\Theta,\mathbf{k})) - KL(P(Y)|P(Y;\Phi)) + H(P(Y;\Theta,\mathbf{k}), P(Y;\Phi)) \tag{18}$$

where $KL(\cdot|\cdot)$ is the KL divergence and $H(P(Y;\Theta,\mathbf{k}), P(Y;\Phi))$ is the cross-entropy between the two distributions. Minimizing the first term is equivalent to maximizing the Eqn. 11. Maximizing the negative KL distance of the second term is equivalent to maximizing the following log-likelihood function,

$$L(\Phi) = \frac{1}{N}\sum_{i=1}^{N} \log P(Y_i;\Phi). \tag{19}$$

The gradient of $L(\Phi)$ is computed by ,

$$\frac{\partial}{\partial\Phi}L(\Phi) = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\Phi}f(Y_i;\Phi) - \mathrm{E}_{p(Y;\Phi)}\left[\frac{\partial}{\partial\Phi}f(Y;\Phi)\right]$$

One key result is that $\frac{\partial}{\partial\Phi}\log Z(\Phi) = \mathrm{E}_\Phi[\frac{\partial}{\partial\Phi}f(Y;\Phi)]$, where E_Φ denotes the expectation with respect to $P(Y;\Phi)$. We use the generative model to alleviate the difficulty of sampling images from the energy-based model,

$$\frac{\partial}{\partial\Phi}L(\Phi) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\Phi}f(Y_i;\Phi) - \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\Phi}f(g(Z_i;\Theta,\mathbf{k});\Phi) \tag{20}$$

The third term in Eqn.18 can be solved by,

$$\min_{\Theta} H(P(Y;\Theta,\mathbf{k}), P(Y;\Phi)) = -\mathrm{E}_{p(Y;\Theta,\mathbf{k})}\log P(Y;\Phi) = -\mathrm{E}_{Z\sim p(Z)}\log P(g(Z;\Theta,\mathbf{k});\Phi) \tag{21}$$

Then, the gradient of the cross-entropy $H(P(Y;\Theta,\mathbf{k}), P(Y;\Phi))$ w.r.t. Θ is computed by,

$$\frac{\partial}{\partial\Theta}H(\Theta) = -\frac{\partial}{\partial\Theta}\mathrm{E}_{Z\sim p(Z)}\log P(g(Z;\Theta,\mathbf{k});\Phi) \approx \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\Theta}f(g(Z_i;\Theta,\mathbf{k});\Phi) \tag{22}$$

Algorithm 1 summarizes the detail of learning and inference.

Algorithm 1 Learning and Inference Algorithm

Input:
(1) training examples $\{Y_i, i = 1, ..., N\}$
(2) network architectures and sparsity-inducing hyper-parameters $\mathbf{k}$
(3) Langevin steps $l_{\mathcal{G}}$ and learning iterations T

Output:
(1) estimated parameters Θ and Φ
(2) synthesized examples $\{\hat{Y}_i, i = 1, ..., \tilde{N}\}$

Let $t \leftarrow 0$, randomly initialize Θ and Φ.
repeat
  Step 1: For $i = 1, ..., N$, generate $\hat{Z}_i \sim \mathrm{N}(0, I_d)$, and generate $\hat{Y}_i = g(\hat{Z}_i;\Theta^{(t)},\mathbf{k})$. Update $\Phi^{(t+1)} = \Phi^{(t)} + \gamma_t \frac{\partial}{\partial\Phi}L(\Phi^{(t)})$, where $\frac{\partial}{\partial\Phi}L(\Phi^{(t)})$ is computed using Eqn. 20.
  Step 2: For each i, start from the current Z_i, run $l_{\mathcal{G}}$ steps of Langevin dynamics to update Z_i each of whhic follows Eqn. 13.
  Step 3: Update $\Theta^{(t+1)} = \Theta^{(t)} + \gamma_t \frac{\partial}{\partial\Theta}(L(\Theta^{(t)}) + H(\Theta^{(t)}))$, where $\frac{\partial}{\partial\Theta}L(\Theta^{(t)})$ is computed using Eqn. 15, and $\frac{\partial}{\partial\Theta}H(\theta^{(t)})$ is computed using Eqn. 22.
  Let $t \leftarrow t + 1$
until $t = T$

4 Experiments

In this section, we present the qualitative and quantitative results of the proposed method tested on five datasets widely used in image synthesis. The proposed method consistently obtains better quantitative performance with interpretable hierarchical representations learned. We implement the proposed method using Google's TensorFlow [1] and our source code will be released.

Datasets: We use the CelebA dataset [32], the human fashion dataset [33], the Stanford car dataset [34], the LSUN bedroom dataset [35]. We train our proposed AND-OR networks on the first 10k CelebA images as processed by OpenFace [36], 78,979 human fashion images as done in [33], the first 16k Stanford car images, and the first 100k bedroom images, all cropped to 64×64 pixels.

Baselines: We compare our model with state-of-art image synthesis methods including VAE [15], DCGAN [37], WGAN [5], CoopNet [4], CEGAN [38]), ALI [39], and ALICE [40]. We use the Fréchet Inception distance (FID) [41] for evaluating the quality of generated images. The number of generated samples for computing FID is the same as that of training set. We also compare the image reconstruction quality in terms per pixel mean square errors (MSE).

Settings: Table 1 summarizes architectures of the generator network and the energy-based network used in our experiments.

4.1 Qualitative Results

The proposed AND-OR network is capable of joint image synthesis and reconstruction. Figure 3 shows examples of reconstructed and generated face images. The top of Figure 4,

1. https://github.com/tensorflow

Fig. 3. Results of image synthesis and reconstruction on CelebA. The first two rows show the original face images, the middle two rows show the reconstruction results, and the last two rows show the generated face images. The learned AND-OR tree model is illustrated in Figure 1.

TABLE 1
Network architectures used in experiments. Upsample uses nearest neighbor interpolation. Downsample uses average pooling. LReLU is the leaky-ReLU with negative slope being 0.2. All convolution layers use kernels of size 3×3 with the number of output channles listed in $(\cdot)$. The sparsity-inducing hyper-parameter k is also given.

Layer	Generator Network	Energy Based Network
1	$Z \sim \mathcal{N}(0, I_{100})$	Y, $(64 \times 64 \times 3)$
2	FC, $(4 \times 4 \times 3200)$; $k = 8$	Conv+LReLU, (64)
3	Upsample, 2 Conv+ReLU, (512) Conv+ReLU, (512); $k = \frac{8 \times 8}{4}$	Downsample, 2 Conv+LReLU, (64) Conv+LReLU, (64)
4	Upsample, 2 Conv+ReLU, (256) Conv+ReLU, (256); $k = \frac{16 \times 16}{4}$	Downsample, 2 Conv+LReLU, (128) Conv+LReLU, (128)
5	Upsample, 2 Conv+ReLU, (128) Conv+ReLU, (128); $k = \frac{32 \times 32}{4}$	Downsample, 2 Conv+LReLU, (256) Conv+LReLU, (256)
6	Upsample, 2 Conv+ReLU, (64) Conv+ReLU, (64)	Downsample, 2 Conv+LReLU, (512) Conv+LReLU, (512)
7	Conv+Tanh, (3)	FC, (1)

Figure 5 and Figure 6 shows examples for human fashion images, car images and bedroom images respectively. Both the reconstructed images and the generated images look sharp. The reconstructed images of bedroom (Figure 6) look relatively blurrier. Bedroom images usually have larger variations which may entail more complicated generator and energy-based network architectures. We use the same architectures for all the tasks.

The learned AND-OR trees on the five datasets unfold the internal generation process with semantically meaningful internal basis functions learned (emerged). To our knowledge, this is the first work in image synthesis that learn interpretable image generation from scratch. More interestingly, we observe that the primitive layers in different AND-OR trees share many common patterns similar to the Gabor wavelets and blob-like structures, which is also consistent with results in traditional sparse coding.

4.2 Quantitative Results

The FID comparsions are summarized in Table 2. The proposed method consistently outperforms the seven state-of-

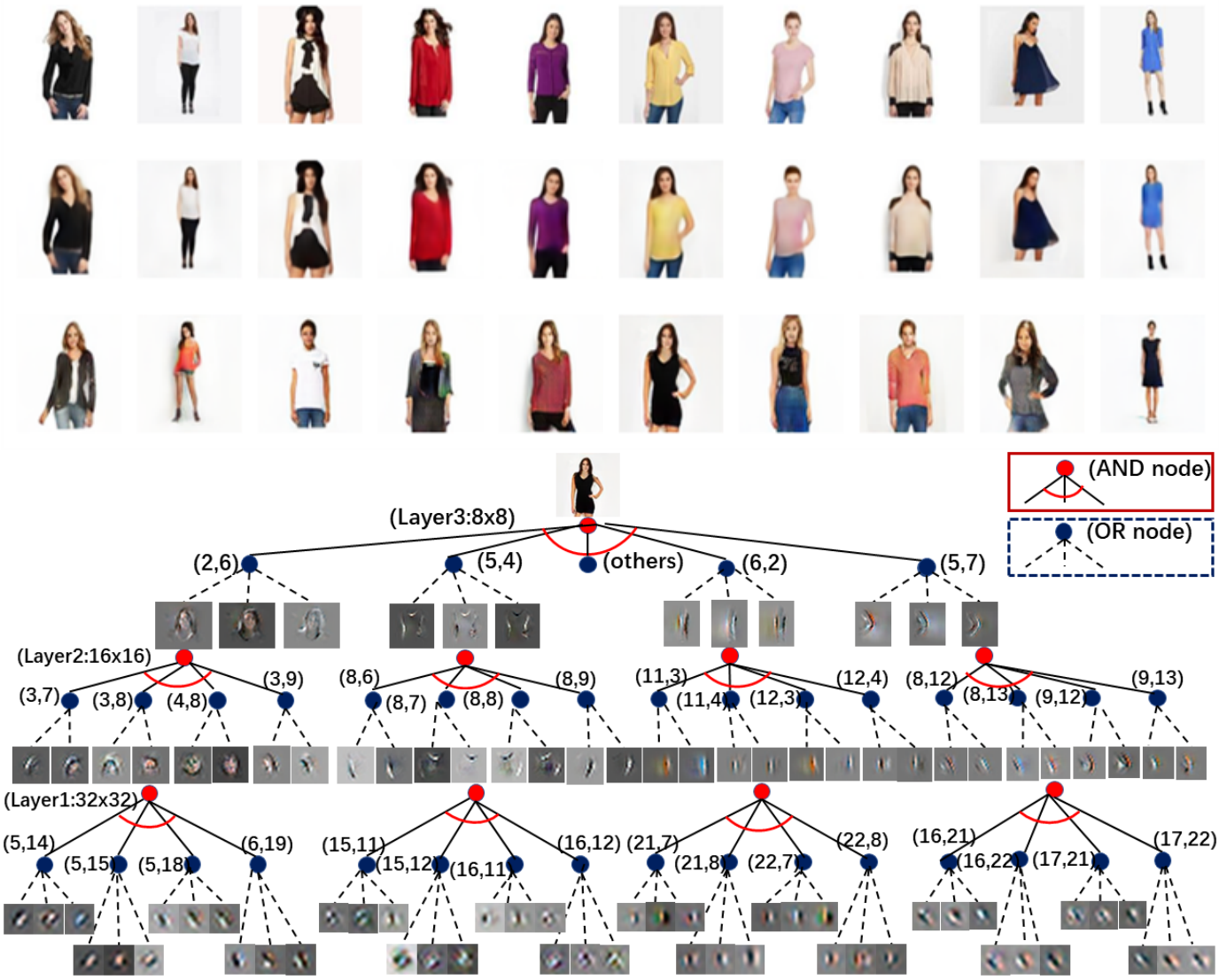

Fig. 4. Results on the human fashion dataset. Top: the three rows show original images, reconstructed images and generated images respectively. Bottom: the learned AND-OR tree model shown in the same way as Figure 1. Best viewed in color and magnification.

TABLE 2
Comparisons of the Fréchet inception distance (FID). Smaller FID is better. The last column Δ shows the improvement of our method over the runner-up method, WGAN.

Datasets\ Methods	VAE [15]	DCGAN [37]	WGAN [5]	CoopNet [4]	CEGAN [38]	ALI [39]	ALICE [40]	Ours	Δ
CelebA	45.06	19.28	*18.85*	28.49	20.62	30.53	23.17	**16.62**	2.32
HumanFashion	23.28	10.82	*10.19*	15.39	11.14	16.75	12.56	**8.65**	1.44
Standford cars	76.21	33.58	*31.62*	45.34	36.12	50.48	37.35	**28.36**	2.26
LSUN bedrooms	81.35	36.26	*33.81*	49.73	41.64	52.79	39.08	**29.70**	4.11

the-art image synthesis methods in comparisons. On the human fashion dataset, the images are nice and clean, our method obtains the least improvement by 1.44. On the bedroom dataset, the images are much more complex with large structural and appearance variations, our method obtains the biggest improvement by 4.11. We note that all the improvement are obtained with more interpretable representations learned in the form of AND-OR trees. This is especially interesting since it shows that jointly improving model performance and interpretability is possible.

We utilize per-pixel mean square error (MSE) to evaluate image reconstruction. Table 3 shows the comparisons with three state-of-the-art methods that are also capable of joint image synthesis and reconstruction (VAE [15], ALI [39], and ALICE [40]). We do not compare with the variants of GANs and CoopNets since they usually can not perform joint image reconstruction.

4.3 Ablation Studies

In addition to the AND-OR tree visualization, we propose a simple method to evaluate the intepretability of learned basis functions (e.g., those at Layer 3, see Figure 1). We

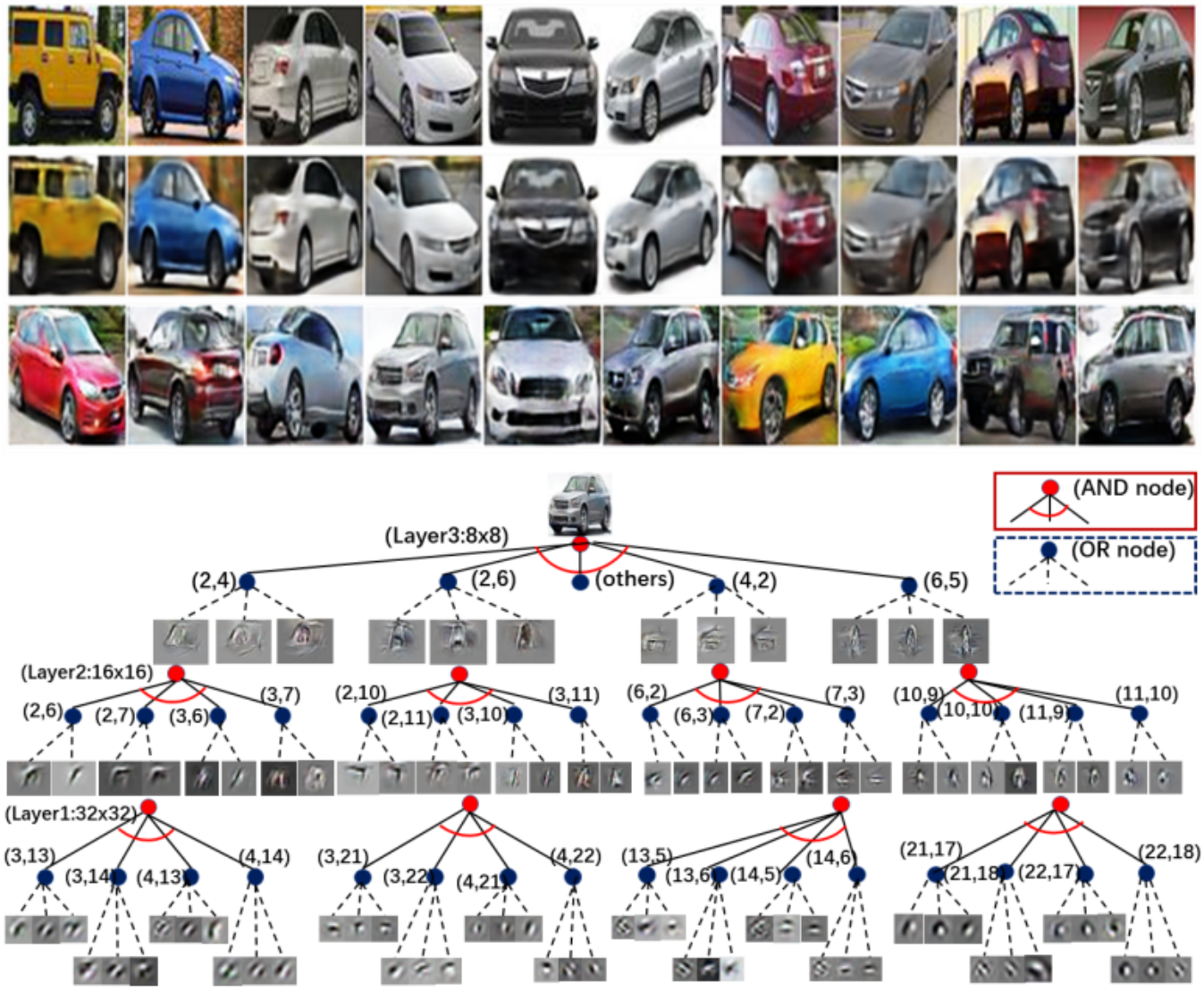

Fig. 5. Results on the Stanford car dataset. Top: the three rows show original images, reconstructed images and generated images respectively. Bottom: the learned AND-OR tree model shown in the same way as Figure 1. Best viewed in color and magnification.

TABLE 3
Comparisons of the per-pixel mean square error (MSE). Smaller MSE is better.

Datasets\ Methods	VAE [15]	ALI [39]	ALICE [40]	Ours
CelebA	0.016	0.132	0.019	**0.011**
HumanFashion	0.033	0.28	0.043	**0.024**
Standford cars	0.081	0.563	0.078	**0.054**
LSUN bedrooms	0.154	0.988	0.127	**0.097**

TABLE 4
Evaluation of interpretability. Comparisons of the matching scores using the fast normalized cross-correlation algorithm between the generator without sparsity and the proposed sparse activated generator.

Methods\Datasets	CelebA	HumanFashion	Cars	Bedroom
w/o sparsity	0.33	0.29	0.31	0.23
w/ sparsity	**0.83**	**0.81**	**0.76**	**0.72**

perform Template Matching between the learned basis functions with training images using the fast normalized cross-correlation algorithm [42]. Consider Layer 3 (a.k.a. object part level), if the learned basis functions contain meaningful local parts of the object, the matching score shall be high. We compare the Layer-3 basis functions learned with and without the proposed sparsity-inducing approach respectively (i.e., Eqn. 7 vs Eqn. 1). The results of the mean matching scores are summarized in Table 4. The proposed method significantly outperforms the counterpart. The results verify that the proposed method can learn meaningful basis functions for better model interpretability.

5 Conclusion

This paper proposes interpretable image synthesis by learning sparsely connected AND-OR networks. The proposed method is built on the vanilla generator network. The

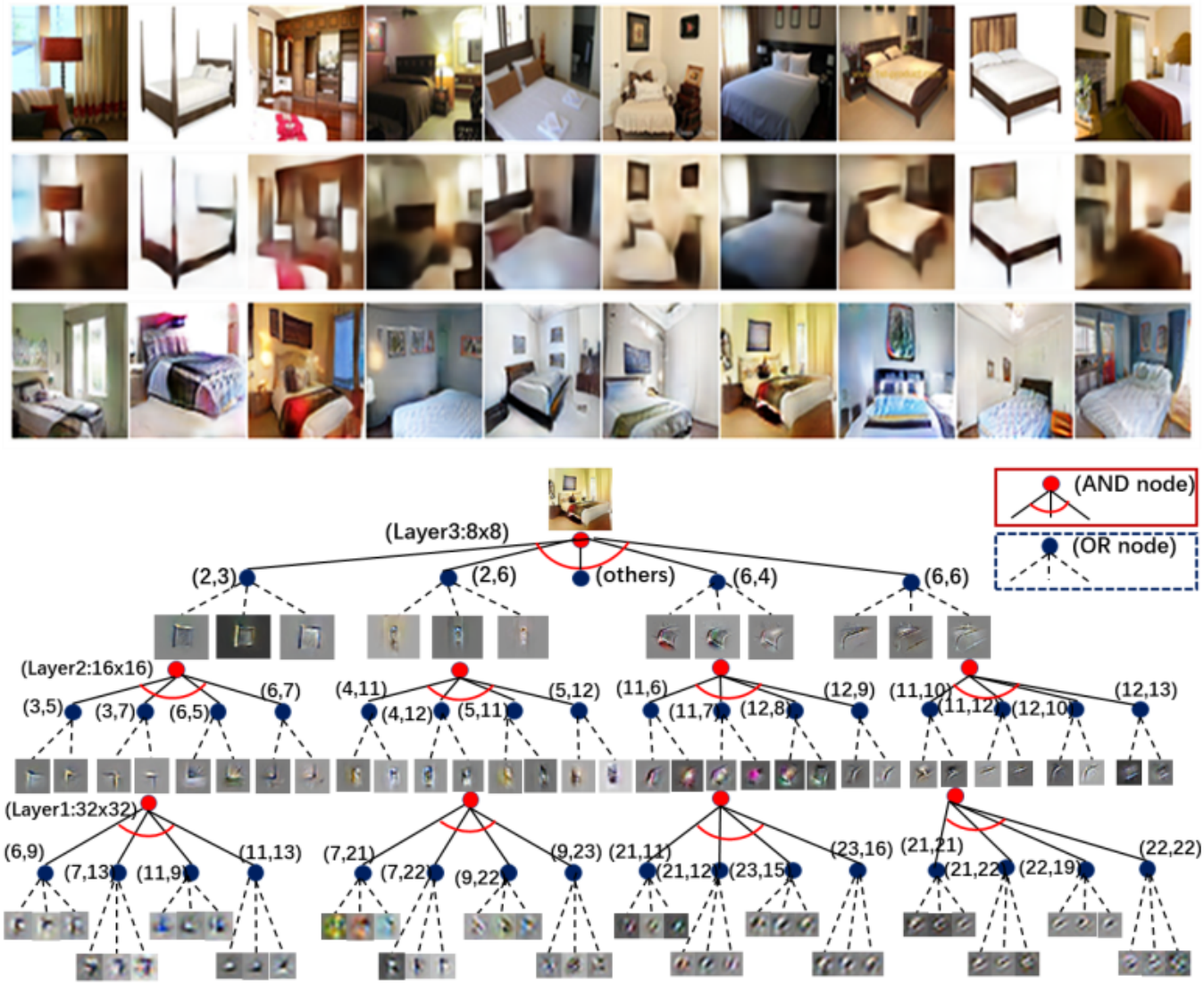

Fig. 6. Results on the LSUN bedroom dataset. Top: the three rows show original images, reconstructed images and generated images respectively. Bottom: the learned AND-OR tree model shown in the same way as Figure 1. Best viewed in color and magnification.

AND-OR network of sparsely connected nodes emerges from the original densely connected generator network when sparsity-inducing terms are introduced. In training, we further combine with energy-based networks and pose the learning problem under MLE. The resulting AND-OR networks are capable of joint image synthesis and reconstruction. In experiments, the proposed method is tested on five benchmark datasets. The results show that meaningful and interpretable hierarchical representations are learned with better qualities of image synthesis and reconstruction obtained than seven state-of-the-art methods.

Appendix A
Computing and Visualization of the Basis Functions

A.1 Basis functions and sparse representations

Suppose $Y(x)$ denotes an image defined on the spatial domain $\mathbb{D}$, where $x = (x_1, x_2)$ denotes a two dimensional vector which indexes the coordinates of pixels. $Y(x)$ can be treated as a two-dimensional function defined on $\mathbb{D}$. $Y(x)$ can also be treated as a vector if we fix an ordering for the pixels. Suppose $D = \mathbb{D}$ counts the number of pixels in $\mathbb{D}$, then D is the dimensionality of the vector Y.

A linear basis function (or basis vector) is a local image patch, which is utilized to represent image intensities. Let $\{B_k(x), k = 1, \ldots, K\}$ be a set of prototype basis functions, e.g., wavelets. Suppose that each B_k is supported on a local domain $\mathbb{S}$ centered at the origin, B_k can be shift or translate spatially to a position x_t to get a translated copy of B_k as $B_{k,x_t} = B_k(x - x_t)$. $B_{k,x_t}(x)$ can be treated as a locally supported function defined on $x \in \mathbb{D}$. $B_{k,x_t}(x)$ can also be treated as a vector of the same dimensionality as Y. The basis functions are about top-down linear representation (top-down means from the coefficients to the image) of an image Y by

$$Y(x) = \sum_{k,x_t} s_{k,x_t} B_{k,x_t}(x) + \epsilon(x)$$

where S_{k,x_t} are the coefficients, and $\epsilon(x)$ is the residual image. When most of the s_{k,x_t} are equal to zero, the resulting representation is also called sparse coding.

The traditional one layer sparse coding model generalizes the prior distribution of the coefficients vector to the signal in factor analysis from a Gaussian noise vector to a sparse vector. And typically comprise two steps: (1) Learning a dictionary of basis functions $\{B_{k,x_t}\}$, where k index a finite collection of prototype functions $\{B_k, k = 1, \ldots, K\}$, that B_{k,x_t} is a spatially translated copy of B_k. (2) Inferring the latent factor (S),

$$B^*, S^* = \arg\min_{B,S} \| Y - BS \|_F^2 \qquad \| s_{x_t} \|_0 \leq K$$

K-SVD [43] is a typical algorithm for learning the dictionary of basis functions B_k, while convex relaxation L_1 minimization or greedy methods such as matching pursuit (OMP) [44] can be employed to infer the sparsification s_{k,x_t}, i.e., the selection of the basis functions. We will develop a hierarchical top-down sparse coding model that can generate (in addition to merely reconstruct) the realistic natural image patterns.

A.2 Inducing unified hierarchical sparse coding model from sparsely connected generator

A sparsity-inducing function is introduced to develop an explainable generative model. At each layer of the top-down generator network, the sparsity-inducing function only selects the top k coefficients to be active and forces all the other coefficients to be zeros.

Suppose there are L deconvolutional layers, the featuremap at the ith layer is fm^i, and after sparsity, the survived activations constitute a sparse tensor, fms^i. For the ith layer,

$$fm^{i+1} = r(g(S(fm^i, K^{i+1}), W^i))$$

where we denote the deconvolutional operation as $g(\cdot, W^i)$, the parameters at the ith layer as W^i, the ReLU operation as $r(\cdot)$, and the sparsity-inducing function as $S(\cdot, K^{i+1})$. The sparse activations are determined by the input latent vector and the learned parameters (weights and bias) of the generator. A different latent code vector will generate a different parsing tree. The ReLU and the sparsity-inducing function can be seen as the switches or masks that partitioning the space. For an instantiated latent code vector Z, remember the masks of the chosen elements of both the sparsity-inducing and $ReLU$ functions. Then, these nonlinear functions will be changed to linear functions,

$$fm^{i+1} = g^i(fm^i \odot ms^i) \odot mr^i$$

where $g^i(\cdot) = g(\cdot, W^i)$ is the deconvolutional function, ms^i denotes the mask matrix of the sparsity-inducing function, mr^i denotes the mask matrix of the $ReLU$ function, and $\odot$ denotes the dot product between two matrices. The final generated image can be formulated as,

$$\begin{aligned} Y &= tanh(fm^{L+1}) = tanh(g^L(fm^L \odot ms^L) \\ &= tanh(g^L(g^{L-1}(fm^{L-1} \odot ms^{L-1}) \odot mr^{L-1} \odot ms^L) \\ &= tanh(\phi^L \cdot \phi^{L-1} \cdots \phi^i \cdots \phi^1(fm^1)) \end{aligned}$$

$$\phi^i(fm^i) = \begin{cases} g^i(fm^i \odot ms^i) \odot mr^i & \text{i=1,2,..., L-1} \\ g^i(fm^i \odot ms^i) & \text{i=L} \end{cases}$$

Let the kth sparse activation be $fm^i \odot ms^i_k$, where $ms^i_k, k \in 1, \ldots, K$ is the mask matrix only selects the kth sparse activation. Then $fm^i \odot ms^i = \sum_{k=1}^K fm^i \odot ms^i_k$.

$$\begin{aligned} Y &= tanh(\phi^L \cdot \phi^{L-1} \cdots \phi^i \cdots \phi^1(fm^1)) \\ &= tanh(\phi^L \cdot \phi^{L-1} \cdots \phi^i(\sum_{k=1}^K fm^i \odot ms^i_k)) \\ &= tanh(\psi^i(\sum_{k=1}^K fm^i \odot ms^i_k)) \\ &= tanh(\sum_{k=1}^K \psi^i(fm^i \odot ms^i_k)) \end{aligned}$$

where $\psi^i(\cdot) = \phi^L \cdot \phi^{L-1} \cdots \phi^i(\cdot)$ is a linear function (composed with a serious of linear function). For the ith layer, the value of the K^i sparse activations can be interpreted as the sparse coefficients, $s^i_k, k \in \{1, \ldots, K^i\}$. The sparse activations in the ith layer is equivalent to,

$$\sum_{k=1}^K fm^i \odot ms^i_k = \sum_{k=1}^K s_k \times ms^i_k \tag{23}$$

And we have

$$\begin{aligned} \psi^i(\sum_{k=1}^K fm^i \odot ms^i_k) &= \psi^i(\sum_{k=1}^K s^i_k \times ms^i_k) \\ &= \sum_{k=1}^K s_k \times \psi^i(ms^i_k) \end{aligned} \tag{24}$$

We obtain the final induced unified sparse coding formulation as,

$$\psi^i(\sum_{k=1}^K fm^i \odot ms^i_k) = \sum_{k=1}^K s^i_k \times \psi^i(ms^i_k) = \sum_{k=1}^K s^i_k \times B^i_k$$

where $B^i_k = \psi^i(ms^i_k)$ is the **basis function** of the sparse coefficient s^i_k of the $i^{th}, i \in 1, \ldots, L$ layer, (s^i_k can be computed from Eq. (23)).

The generated (or reconstruction) image can be represented as the summation of basis function $B^i_k = \psi^i(ms^i_k)$ at different layers,

$$\begin{aligned} Y &= tanh(\phi^L \cdot \phi^{L-1} \cdots \phi^i \cdots \phi^1(fm^1)) \\ &= tanh(\psi^i(\sum_{k=1}^K fm^i \odot ms^i_k)) \\ &= tanh(\sum_{k=1}^K s^i_k \times B^i_k) \end{aligned} \tag{25}$$

B^i_k can be employed to analyze each sparse activation's contribution to the whole generated image, when the layer i going from bottom to top, B^i_k contains more and more high-level semantic information of the generated images.

Acknowledgments

The work of X. Xing, S.-C. Zhu and Y. Wu is supported by DARPA XAI project N66001-17-2-4029; ARO project W911NF1810296; and ONR MURI project N00014-16-1-2007; and Extreme Science and Engineering Discovery Environment (XSEDE) grant ASC170063. The work of X. Xing is also supported by Natural Science Foundation of China No. 61703119, Natural Science Fund of Heilongjiang Province of China No. QC2017070, and Fundamental Research Funds for the Central Universities No. 3072019CFT0402. The work of T. Wu is supported in part by NSF IIS-1909644 and ARO grant W911NF1810295.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014, pp. 2672–2680. 1

[2] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv:1809.11096*, 2018. 1

[3] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *arXiv:1812.04948*, 2018. 1

[4] J. Xie, Y. Lu, R. Gao, and Y. N. Wu, "Cooperative learning of energy-based model and latent variable model via mcmc teaching," in *AAAI*, 2018. 1, 5, 6

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017, pp. 214–223. 1, 5, 6

[6] X. Xing, T. Han, R. Gao, S.-C. Zhu, and Y. N. Wu, "Unsupervised disentangling of appearance and geometry by deformable generator network," in *CVPR*, June, pp. 10 354–10 363. 1

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 1

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012, pp. 1106–1114. 1, 3

[9] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are gans created equal? a large-scale study," *arXiv:1711.10337*, 2017. 1

[10] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," *arXiv:1809.11096*, 2018. 1

[11] DARPA, "Explainable artificial intelligence (xai) program, http://www.darpa.mil/program/explainable-artificial-intelligence, full solicitation at http://www.darpa.mil/attachments/ darpa-baa-16-53.pdf," 2016. 1

[12] D. Bau, J.-Y. Zhu, H. Strobelt, Z. Bolei, J. B. Tenenbaum, W. T. Freeman, and A. Torralba, "Gan dissection: Visualizing and understanding generative adversarial networks," in *ICLR*, 2019. 1

[13] S. Geman, D. Potter, and Z. Y. Chi, "Composition systems," *Quarterly of Applied Mathematics*, vol. 60, no. 4, pp. 707–736, 2002. 1

[14] S.-C. Zhu and D. Mumford, "A stochastic grammar of images," *Found. Trends Comp. Graph. Vis.*, vol. 2, no. 4, pp. 259–362, 2007. 1

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013. 1, 5, 6, 7

[16] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016. 1

[17] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, p. 607, 1996. 1

[18] ——, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision research*, vol. 37, no. 23, pp. 3311–3325, 1997. 1

[19] H. Ishwaran and J. S. Rao, "Spike and slab variable selection: frequentist and bayesian strategies," *The Ann. of Stat.*, vol. 33, no. 2, pp. 730–773, 2005. 2

[20] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011. 2

[21] A. Makhzani and B. Frey, "K-sparse autoencoders," *arXiv:1312.5663*, 2013. 2

[22] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 27, no. 12, pp. 2486–2498, 2015. 2

[23] J. Ngiam, Z. Chen, S. A. Bhaskar, P. W. Koh, and A. Y. Ng, "Sparse filtering," in *NeurIPS*, 2011, pp. 1125–1133. 2

[24] A. Makhzani and B. J. Frey, "Winner-take-all autoencoders," in *NeurIPS*, 2015, pp. 2791–2799. 2

[25] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *ICML*. ACM, 2009, pp. 609–616. 2

[26] G. B. Huang, H. Lee, and E. Learned-Miller, "Learning hierarchical representations for face verification with convolutional deep belief networks," in *CVPR*. IEEE, 2012, pp. 2518–2525. 2

[27] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks." in *CVPR*, vol. 10, 2010, p. 7. 2

[28] M. D. Zeiler, G. W. Taylor, R. Fergus *et al.*, "Adaptive deconvolutional networks for mid and high level feature learning." in *ICCV*, vol. 1, no. 2, 2011, p. 6. 2

[29] T. Han, Y. Lu, S.-C. Zhu, and Y. N. Wu, "Alternating back-propagation for generator network." in *AAAI*, 2017, pp. 1976–1984. 3, 4

[30] R. Gao, J. Xie, S. Zhu, and Y. N. Wu, "Learning grid cells as vector representation of self-position coupled with matrix representation of self-motion," in *ICLR*, 2019. [Online]. Available: https://openreview.net/forum?id=Syx0Mh05YQ 3

[31] L. Younes, "On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates," *Stochastics*, vol. 65, no. 3-4, pp. 177–228, 1999. 4

[32] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015. 5

[33] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *CVPR*, June 2016. 5

[34] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[35] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv:1506.03365*, 2015. 5

[36] B. Amos, B. Ludwiczuk, and M. Satyanarayanan, "Openface: A general-purpose face recognition library with mobile applications," CMU-CS-16-118, Tech. Rep., 2016. 5

[37] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016. 5, 6

[38] Z. Dai, A. Almahairi, P. Bachman, E. Hovy, and A. Courville, "Calibrating energy-based generative adversarial networks," *arXiv:1702.01691*, 2017. 5, 6

[39] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *arXiv:1606.00704*, 2016. 5, 6, 7

[40] C. Li, H. Liu, C. Chen, Y. Pu, L. Chen, R. Henao, and L. Carin, "Alice: Towards understanding adversarial learning for joint distribution matching," in *NeurIPS*, 2017, pp. 5495–5503. 5, 6, 7

[41] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017, pp. 6626–6637. 5

[42] J.-C. Yoo and T. H. Han, "Fast normalized cross-correlation," *Circuits, systems and signal processing*, vol. 28, no. 6, p. 819, 2009. 7

[43] M. Aharon, M. Elad, A. Bruckstein *et al.*, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE TSP*, vol. 54, no. 11, p. 4311, 2006. 10

[44] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Info. Th.*, vol. 53, no. 12, pp. 4655–4666, 2007. 10