Introspective Neural Networks for Generative Modeling

Justin Lazarow*
Dept. of CSE, UCSD
jlazarow@ucsd.edu

Long Jin*
Dept. of CSE, UCSD
lojin@ucsd.edu

Zhuowen Tu
Dept. of CogSci, UCSD
ztu@ucsd.edu

Abstract

We study unsupervised learning by developing a generative model built from progressively learned deep convolutional neural networks. The resulting generator is additionally a discriminator, capable of "introspection" in a sense — being able to self-evaluate the difference between its generated samples and the given training data. Through repeated discriminative learning, desirable properties of modern discriminative classifiers are directly inherited by the generator. Specifically, our model learns a sequence of CNN classifiers using a synthesis-by-classification algorithm. In the experiments, we observe encouraging results on a number of applications including texture modeling, artistic style transferring, face modeling, and unsupervised feature learning. [1]

1. Introduction

Supervised learning techniques have made a substantial impact on tasks that can be formulated as a classification/regression problem [43, 11, 5, 27]. Unsupervised learning, where no task-specific labeling/feedback is provided on top of the input data, still remains one of the most difficult problems in machine learning but holds a bright future since a large number of tasks have little to no supervision.

Popular unsupervised learning methods include mixture models [9], principal component analysis (PCA) [24], spectral clustering [38], topic modeling [4], and autoencoders [3, 2]. In a nutshell, unsupervised learning techniques are mostly guided by the minimum description length principle (MDL) [35] to best reconstruct the data whereas supervised learning methods are primarily driven by minimizing error metrics to best fit the input labeling. Unsupervised learning models are often generative and supervised classifiers are often discriminative; generative model learning has been traditionally considered to be a much harder task than discriminative learning [12] due to its intrinsic learning complexity, as well as many assumptions and simplifications made about the underlying models.

[1] * equal contribution.

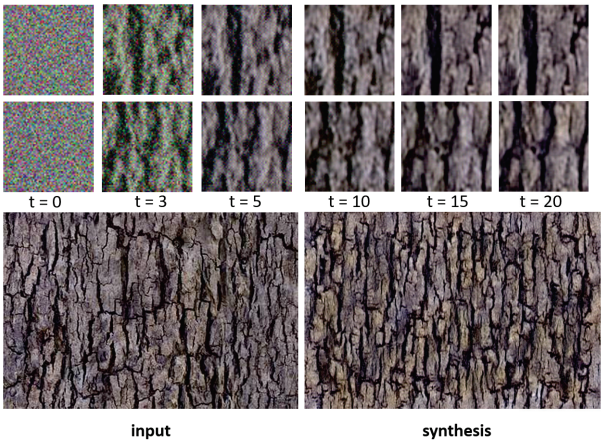

Figure 1. The first row shows the development of two 64×64 pseudo-negative samples (patches) over the course of the training process on the "tree bark" texture at selected stages. We can see the initial "scaffold" created and then refined by the networks in later stages. The input "tree bark" texture and a synthesized image by our INNg algorithm are shown in the second row. This texture was synthesized by INNg using 20 CNN classifiers each with 4 layers.

Generative and discriminative models have traditionally been considered distinct and complementary to each other. In the past, connections have been built to combine the two families [12, 28, 41, 22]. In the presence of supervised information with a large amount of data, a discriminative classifier [26] exhibits superior capability in making robust classification by learning rich and informative representations; unsupervised generative models do not require supervision but at a price of relying on assumptions that are often too ideal in dealing with problems of real-world complexity. Attempts have previously been made to learn generative models directly using discriminative classifiers for density estimation [45] and image modeling [40]. There is also a wave of recent development in generative adversarial networks (GAN) [14, 34, 37, 1] in which a discriminator helps a generator try not to be fooled by "fake" samples. We will discuss in detail the relations and connections of our model with these existing literature in the later sections.

In [45], a self supervised boosting algorithm was proposed to train a boosting algorithm by sequentially adding features as weak classifiers on additionally self-generated negative samples. Furthermore, the generative discrimi-

native modeling work (GDL) in [40] generalizes the concept that a generative model can be successfully modeled by learning a sequence of discriminative classifiers via self-generated pseudo-negatives.

Inspired by the prior work on generative modeling [51, 45, 40] and development of convolutional neural networks [27, 26, 13], we develop an image modeling algorithm, introspective neural networks for generative modeling (INNg) that can be used simultaneously as a generator and a discriminator, consisting of two critical stages during training: (1) pseudo-negative sampling (synthesis) — a generation of samples considered to be positive examples and (2) a CNN classifier learning stage (classification) for self-evaluation and model updating from the previous synthesis. There are a number of interesting properties about INNg worth highlighting:

- **CNN classifier as generator**: No special conditions on the CNN architecture are needed in INNg and existing CNN classifiers can be directly made into generators, if trained properly.
- **End-to-end self-evaluation and learning**: Perform end-to-end "introspective learning" to self-classify between synthesized samples (pseudo-negatives) and the training data, to approach the target distribution.
- **All backpropagation**: Our synthesis-by-classification algorithm performs efficient training using backpropagation in both stages: the sampling stage for the input images and the classification training stage for the CNN parameters.
- **Model-based anysize-image-generation**: Since we model the input image, we can train on images of a given size and generate an image of a larger size while maintaining coherence of the entire image.
- **Agnostic to various vision applications**: Due to its intrinsic modeling power being at the same time generative and discriminative, INNg can be adopted to many applications in computer vision. In addition to the applications shown here, extension of the objective (loss) function within INNg is expected to work for other tasks such as "image-to-image translation" [21].

2. Significance and related work

Our introspective neural networks generative modeling (INNg) algorithm has connections to many existing approaches including the MinMax entropy work for texture modeling [51], and the self-supervised boosting algorithm [45]. It builds on top of convolutional neural networks [27] and we are particularly inspired by two lines of prior algorithms: the generative modeling via discriminative approach method (GDL) [40], and the DeepDream code [31] and the neural artistic style work [13]. Parallels can be drawn to ideas elaborated in [16] where parameters of a distribution are learned using a (single) classifier between noise and training data. Additionally, the use of "negative" examples to bridge the gap between an unsupervised task into a supervised one is also seen in [18], although this focuses on the training of the *weights* of the network for classification rather than synthesis. The work of [31, 13], along with the Hybrid Monte Carlo literature [44], motivates us to significantly improve the time-consuming sampling process in [40] by an efficient stochastic gradient descent (SGD) process via backpropagation (the reason for us to say "all backpropagation"). Next, we review some existing generative image modeling work, followed by detailed discussions about GDL [40]; comparisons to generative adversarial networks (GAN) [14] will be provided in Section 3.7.

The history of generative modeling on image or non-image domains is extremely rich, including the general image pattern theory [15], deformable models [48], inducing features [8], wake-sleep [19], the MiniMax entropy theory [51], the field of experts [36], Bayesian models [49], and deep belief nets [20]. Each of these pioneering works points to some promising direction in unsupervised generative modeling. However the modeling power of these existing frameworks is still somewhat limited in computational and/or representational aspects. In addition, not too many of them sufficiently explore the power of discriminative modeling. Recent works that adopt convolutional neural networks for generative modeling [47] either use CNNs as a feature extractor or create separate paths [46, 42]. The neural artistic transferring work [13] has demonstrated impressive results on the image transferring and texture synthesis tasks but it is focused [13] on a careful study of channels attributed to artistic texture patterns, instead of aiming to build a generic image modeling framework. The self-supervised boosting work [45] sequentially learns weak classifiers under boosting [11] for density estimation, but its modeling power was not adequately demonstrated.

Relationship with GDL [40]

The generative via discriminative learning framework (GDL) [40] learns a generator through a sequence of boosting classifiers [11] using repeatedly self-generated samples, called **pseudo-negatives**. Our INNg algorithm takes inspiration from GDL, but we also observe a number of limitations in GDL that will be overcome by INNg: GDL uses manually specified feature types (histograms and Haar filters), which are fairly limited; the sampling process in GDL, based on Markov chain Monte Carlo (MCMC), is a big computational bottleneck. Additional differences between GDL and INNg include: (1) the adoption of convolutional networks in INNg results in a significant boost to feature learning. (2) introducing SGD based sampling schemes to the synthesis process in INNg makes a fundamental improvement to the sampling process in GDL that is otherwise slow and impractical. (3) two compromises to the algorithm, namely INNg-single (see Fig. 4) and INNg-compressed, are additionally proposed to maintain a single classifier or subset of classifiers, respectively.

Introspective Discriminative Networks [23]

In the sister paper [23], the formulation is extended to focus on the discriminative aspect — improvement of existing classifiers. Additional key differences are: **a)** the model

in [23] is usually composed of a *single* classifier with a new formulation for training a softmax multi-class classification and **b)** it is less concerned with human perceivable quality of its syntheses and instead focuses on their impact within the classification task.

3. Method

We describe below the introspective neural networks generative modeling (INNg) algorithm. We discuss the main formulation first, which bears some level of similarity to GDL [40] with the replacement of the boosting algorithm [11] by convolutional neural networks [27]. As a result, INNg demonstrates significant improvement over GDL in terms of both modeling and computational power. Whereas GDL relies on manually crafted features, the use of CNNs within INNg provides for automatic feature learning and tuning when backpropagating on the network parameters as well as an increase in computational power. Both are motivated by a formulation from the Bayes theory.

3.1. Motivation

We start the discussion by borrowing notation from [40]. Suppose we are given a set of training images (patches): $S = \{\mathbf{x}_i \mid i = 1..n\}$ where we assume each $x_i \in \mathcal{R}^m$ e.g. $m = 64 \times 64$ for 64×64 patches. These will constitute **positive** examples of the patterns/targets we wish to model. To introduce the supervised formulation of studying these patterns, we introduce class labels $y \in \{-1, +1\}$ to indicate negative and positive examples, respectively. With this, a generative model computes $p(y, \mathbf{x}) = p(\mathbf{x}|y)p(y)$, which captures the underlying generation process of $\mathbf{x}$ for class y. A discriminative classifier instead computes $p(y|\mathbf{x})$. Under Bayes rule, similar to [40]:

$$p(\mathbf{x}|y=+1) = \frac{p(y=+1|\mathbf{x})p(y=-1)}{p(y=-1|\mathbf{x})p(y=+1)} p(\mathbf{x}|y=-1), \tag{1}$$

which can be further simplified when assuming equal priors $p(y=+1) = p(y=-1)$:

$$p(\mathbf{x}|y=+1) = \frac{p(y=+1|\mathbf{x})}{1-p(y=+1|\mathbf{x})} p(\mathbf{x}|y=-1). \tag{2}$$

Based on Eq. (2), a generative model for the positive samples (patterns of interest) $p(\mathbf{x}|y=+1)$ can be fully represented by a generative model for the negatives $p(\mathbf{x}|y=-1)$ and a discriminative classifier $p(y=+1|\mathbf{x})$, if both $p(\mathbf{x}|y=-1)$ and $p(y=+1|\mathbf{x})$ can be accurately obtained/learned. However, this seemingly intriguing property is circular. To faithfully learn the positive patterns $p(\mathbf{x}|y=+1)$, we need to have a representative $p(\mathbf{x}|y=-1)$, which is equally difficult, if not more. For clarity, we now use $p^-(\mathbf{x})$ to represent $p(\mathbf{x}|y=-1)$. In the GDL algorithm [40], a solution was given to learning $p(\mathbf{x}|y=+1)$ by using an iterative process starting from an initial reference distribution of the negatives $p_0^-(\mathbf{x})$, e.g. a Gaussian distribution $U(\mathbf{x})$ on the entire space of $\mathbf{x} \in \mathcal{R}^m$:

$$\begin{aligned} p_0^-(\mathbf{x}) &= U(\mathbf{x}), \\ p_t^-(\mathbf{x}) &= \frac{1}{Z_t} \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} \cdot p_{t-1}^-(\mathbf{x}),\ t = 1..T \end{aligned} \tag{3}$$

where $Z_t = \int \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x}) d\mathbf{x}$. Our hope is to gradually learn $p_t^-(\mathbf{x})$ by following this iterative process of Eq. 3:

$$p_t^-(\mathbf{x}) \overset{t=\infty}{\rightarrow} p(\mathbf{x}|y=+1), \tag{4}$$

such that the samples drawn $\mathbf{x} \sim p_t^-(\mathbf{x})$ become indistinguishable from the given training samples. The samples drawn from $\mathbf{x} \sim p_t^-(\mathbf{x})$ are called **pseudo-negatives**, following a definition in [40] to indicate examples considered by the current iteration of the *model* to be positives but are, in reality, negative examples. Next, we present the practical realization of ideas from Eq. 3, namely INNg (consisting of a sequence of CNN classifiers composed to produce the process seen in Fig. 3) and, additionally, the extreme case of INNg-single that maintains a sequence consisting of single CNN classifier as seen in Fig. 4.

3.2. INNg Training

Algorithm 1 Outline of the INNg algorithm.

Input: Given a set of training data $S_+ = \{(\mathbf{x}_i, y_i = +1), i = 1..n\}$ with $\mathbf{x} \in \Re^m$.
Initialization: obtain an initial distribution e.g. Gaussian for the pseudo-negative samples: $p_0^-(\mathbf{x}) = U(\mathbf{x})$. Create $S_-^0 = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$ with $\mathbf{x}_i \sim p_0^-(\mathbf{x})$
For t=1..T
1. Classification-step: Train CNN classifier C^t on $S_+ \cup S_-^{t-1}$, resulting in $q_t(y=+1|\mathbf{x})$.
2. Update the model: $p_t^-(\mathbf{x}) = \frac{1}{Z_t} \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})} p_{t-1}^-(\mathbf{x})$.
3. Synthesis-step: sample l pseudo-negative samples $\mathbf{x}_i \sim p_t^-(\mathbf{x}), i = 1, ..., l$ from the current model $p_t^-(\mathbf{x})$ using a SGD-based sampling procedure (backpropagation on the input) to obtain $S_-^t = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$.
4. $t \leftarrow t + 1$ and go back to step 1 until convergence (e.g. indistinguishable to the given training samples).
End

As defined in the previous section, we are given an unlabeled training set by $S = \{\mathbf{x}_i \mid i = 1..n\}$ as our positive examples. Since pseudo-negatives will be added, we refer to this initial positive set as $S_+ = \{(\mathbf{x}_i, y_i = +1) \mid i = 1..n\}$ within the discriminative formulation. Additionally, we must consider the initial set of negatives bootstrapped from noise (also referred to as the initial pseudo-negative set) denoted:

$$S_-^0 = \{(\mathbf{x}_i, -1) \mid i = 1, ..., l\}$$

where $\mathbf{x}_i \sim p_0^-(\mathbf{x}) = U(\mathbf{x})$ according to a Gaussian distribution. Since each stage of the algorithm will refine this set, we define the working set for stage $t = 1..T$ as

$$S_-^{t-1} = \{(\mathbf{x}_i, -1) \mid i = 1, ..., l\}.$$

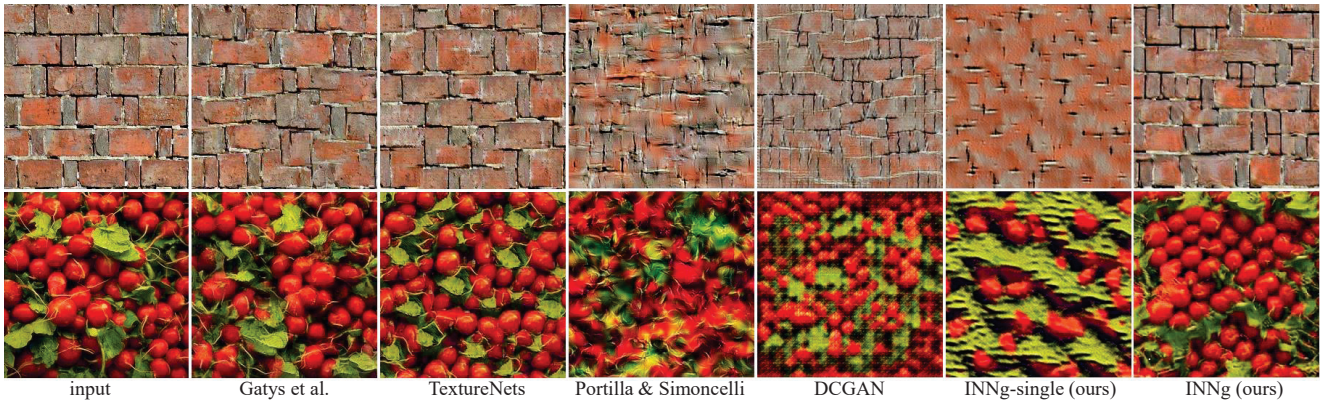

Figure 2. Texture synthesis algorithm comparison. Gatys et al. [13], Texture Nets [42], Portilla & Simoncelli [33], and DCGAN [34] results are from [42].

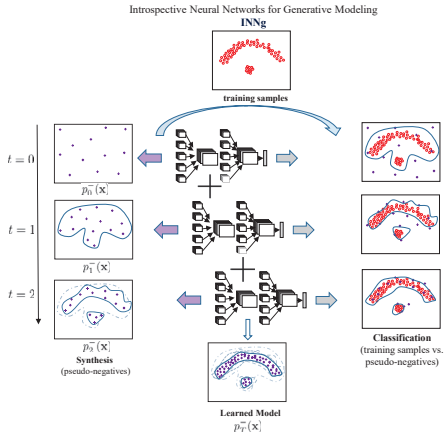

Figure 3. Schematic illustration of the pipeline of INNg. The top figure shows the input training samples shown in red circles. The bottom figure shows the pseudo-negative samples drawn by the learned final model. The left panel displays pseudo-negative samples drawn at each time stamp t. The right panel shows the classification by the CNN on the training samples and pseudo-negatives at each time stamp t.

to include the pseudo-negative samples (l of them) self-generated by the model after stage t. We then train the model at each stage t to obtain

$$q_t(y=+1|\mathbf{x}), \quad q_t(y=-1|\mathbf{x}) \tag{5}$$

over $S_+ \cup S_-^t$ resulting in the classifier C^t. Note that q is an approximation to the true p due to limited samples drawn from $\Re^m$. At each time t, we then compute an approximation to (elaborated in Section 3.2.2)

$$p_t^-(\mathbf{x}) = \frac{1}{Z_t}\frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})}p_{t-1}^-(\mathbf{x}), \tag{6}$$

where $Z_t = \int \frac{q_t(y=+1|\mathbf{x})}{q_t(y=-1|\mathbf{x})}p_{t-1}^-(\mathbf{x})d\mathbf{x}$. Then, we can draw new samples

$$\mathbf{x}_i \sim p_t^-(\mathbf{x})$$

to produce the stages's pseudo-negative set:

$$S_-^{t+1} = \{(\mathbf{x}_i, -1), i = 1, ..., l\}. \tag{7}$$

Algorithm 1 describes the learning process. The pipeline of INNg is shown in Fig. 3, which consists of: (1) a synthesis step and (2) a classification step. A sequence of CNN classifiers is progressively learned. With the pseudo-negatives being gradually generated, the classification boundary gets tightened and approaches the target distribution.

3.2.1 Classification-step

The classification-step can be viewed as training a classifier on the training set $S_+ \cup S_-^t$ where $S_+ = \{(\mathbf{x}_i, y_i = +1), i = 1..n\}$. $S_-^t = \{(\mathbf{x}_i, -1), i = 1, ..., l\}$ for $t \geq 1$. In practice, we also keep a subset of pseudo-negatives from earlier stages to increase stability. We use a CNN as our base classifier. When training a classifier C^t on $S_+ \cup S_-^t$, we denote the parameters to be learned in C^t by a high-dimensional vector $\mathsf{W}_t = (\mathbf{w}_t^{(0)}, \mathbf{w}_t^{(1)})$ which might consist of millions of parameters. $\mathbf{w}_t^{(1)}$ denotes the weights on the top layer combining the features $\phi(\mathbf{x}; \mathbf{w}_t^{(0)})$ and $\mathbf{w}_t^{(0)}$ carries all the internal representations. Without loss of generality, we assume a sigmoid function for the discriminative probability

$$q_t(y|\mathbf{x}; \mathsf{W}_t) = 1/(1 + \exp\{-y < \mathbf{w}_t^{(1)}, \phi(\mathbf{x}; \mathbf{w}_t^{(0)}) >\}). \tag{8}$$

Both $\mathbf{w}_t^{(1)}$ and $\mathbf{w}_t^{(0)}$ can be learned by the standard stochastic gradient descent algorithm via backpropagation to minimize a cross-entropy loss with an additional term on the pseudo-negatives:

$$\mathcal{L}(\mathsf{W}_t) = -\sum_{(\mathbf{x}_i,+1)\in S_+}^{i=1..n} \ln q_t(+1|\mathbf{x}_i; \mathsf{W}_t) - \sum_{(\mathbf{x}_i,-1)\in S_-^t}^{i=1..l} \ln q_t(-1|\mathbf{x}_i; \mathsf{W}_t)$$

3.2.2 Synthesis-step

In the classification step, we obtain $q_t(y|\mathbf{x}; \mathsf{W}_t)$ which is then used to update $p_t^-(\mathbf{x})$ according to Eq. (6):

$$p_t^-(\mathbf{x}) = \prod_{a=1}^{t} \frac{1}{Z_a}\frac{q_a(y=+1|\mathbf{x}; \mathsf{W}_a)}{q_a(y=-1|\mathbf{x}; \mathsf{W}_a)}p_0^-(\mathbf{x}). \tag{9}$$

In the synthesis-step, our goal is to draw fair samples from $p_t^-(\mathbf{x})$. By using SGD-based sampling, we carry out backpropagation with respect to $\mathbf{x}$ (the image space) using the CNN models making up Eq. 9. Note that the partition function (normalization) Z_a is a constant that is not dependent on the sample $\mathbf{x}$. Let

$$g_a(\mathbf{x}) = \frac{q_a(y=+1|\mathbf{x};\mathsf{W}_a)}{q_a(y=-1|\mathbf{x};\mathsf{W}_a)} = \exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) >\}, \tag{10}$$

and take its ln, which is nicely turned into the logit of $q_a(y=+1|\mathbf{x};\mathsf{W}_a)$

$$\ln g_a(\mathbf{x}) =< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) > . \tag{11}$$

Starting from an initialization of $\mathbf{x}$, the process allows us to directly increase $\sum_{a=1}^{t} < \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) >$ using gradient ascent on $\mathbf{x}$ via backpropagation to obtain fair samples subject to Eq. (9). Injecting the noise to the sampler results in a general family of stochastic gradient Langevin dynamics [44]: $\Delta\mathbf{x} = \nabla(\sum_{a=1}^{t} \ln g_a(\mathbf{x})) + \eta$ where $\eta \sim N(0,\epsilon)$ is a Gaussian distribution. In practice, to reduce the time and memory complexity, we initialize $\mathbf{x}$ drawn from $p_{t-1}^-(\mathbf{x})$, allowing us to primarily focus on the most recent model q_t with SGD sampling for $\ln g_t(\mathbf{x}) =< \mathbf{w}_t^{(1)}, \phi(\mathbf{x};\mathbf{w}_t^{(0)}) >$. Therefore, generating pseudo-negative samples does not need a large overhead.

To ensure this follows Langevin dynamics as elaborated in [44], Gaussian noise with an annealed variance would be added, however, we did not observe a big difference in the quality of samples in practice. Additionally, recent work in [7, 30] also show the connections and equivalence between MCMC and SGD-based sampling schemes, where the sampling bias and variance are worth further studying but pose no particular disadvantages here. We have also recently experimented on using different SGD sampling schemes (eg. early-stopping, long steps, perturbations) but did observe significant differences. This is likely partially compensated for by the inherent stochasticity introduced by the random selection of the image patches (and to what extent due to overlap) during synthesis.

3.2.3 Overall model

The overall INNg model after T stages of training becomes:

$$\begin{aligned} p_T^-(\mathbf{x}) &= \frac{1}{Z}\prod_{a=1}^{T}\frac{q_a(y=+1|\mathbf{x};\mathsf{W}_a)}{q_t(y=-1|\mathbf{x};\mathsf{W}_a)}p_0^-(\mathbf{x}) \\ &= \frac{1}{Z}\prod_{a=1}^{T}\exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) >\}p_0^-(\mathbf{x}), \end{aligned} \tag{12}$$

where $Z = \int \prod_{a=1}^{T} \exp\{< \mathbf{w}_a^{(1)}, \phi(\mathbf{x};\mathbf{w}_a^{(0)}) >\}p_0^-(\mathbf{x})d\mathbf{x}$. INNg shares a similar cascade aspect with GDL [40] where the convergence of this iterative learning process to the target distribution was shown by the following theorem in [40].

Theorem 1 *$KL[p(\mathbf{x}|y=+1)||p_{t+1}^-(\mathbf{x})] \leq KL[p(\mathbf{x}|y=+1)||p_t^-(\mathbf{x})]$ where KL denotes the Kullback-Leibler divergences, and $p(x|y=+1) \equiv p^+(x)$.*

This implies that after sufficiently many stages of INNg training, the distribution of the positives should be well approximated by the resulting cascade of classifiers modeled in Eq. 12.

3.3. INNg synthesis

While the previous section describes the training process of an INNg model (which itself uses a synthesis step in it to generate pseudo-negatives for the next stage), we consider the synthesis of a *new* sample from the fully trained model. Since a fully trained model consists of T saved classifiers, we attempt to sample through this sequence in a similar fashion to the training. Starting with some $\mathbf{x} \sim U(\mathbf{x})$, we perform gradient ascent with respect to $\mathbf{x}$ until, based off the current classifier C^t, $\mathbf{x}$ crosses the decision boundary of C^t — to now be considered a positive example. Then, C^{t+1} is loaded and the process is repeated again on the resulting $\mathbf{x}$ until it has passed through all classifiers (through C^T) in a similar manner to produce the sample. Note that we perform early stopping within each stage i.e. it was not seen to be effective to produce a transformation of $\mathbf{x}$ that has near 1.0 probability of being considered a positive, only one that is a narrow margin over this boundary. This additionally allows for a more efficient runtime of the synthesis process.

3.4. An alternative: INNg-single

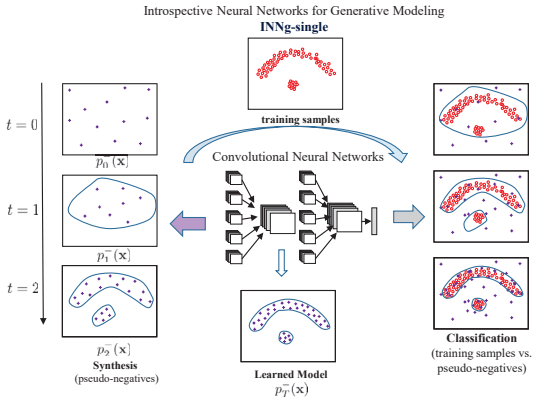

Figure 4. Schematic illustration of the pipeline of INNg-single.

We briefly present the INNg-single algorithm and show the pipeline of INNg-single is in Fig. 4. Note that we maintain a single CNN classifier throughout the entire learning process in INNg-single.

In the classification step, we obtain $q_t(y|\mathbf{x};\mathsf{W}_t)$ (similar as Eq. 8) which is then used to update $p_t^-(\mathbf{x})$ according to

Eq. (13):

$$p_t^-(\mathbf{x}) = \frac{1}{Z_t}\frac{q_t(y=+1|\mathbf{x};\mathsf{W}_t)}{q_t(y=-1|\mathbf{x};\mathsf{W}_t)}p_0^-(\mathbf{x}). \quad (13)$$

In the synthesis-step, we draw samples from $p_t^-(\mathbf{x})$. The overall INNg-single model after T stages of training becomes:

$$p_T^-(\mathbf{x}) = \frac{1}{Z_T}\exp\{< \mathbf{w}_T^{(1)}, \phi(\mathbf{x};\mathbf{w}_T^{(0)}) >\}p_0^-(\mathbf{x}), \quad (14)$$

where $Z_T = \int \exp\{< \mathbf{w}_T^{(1)}, \phi(\mathbf{x};\mathbf{w}_T^{(0)}) >\}p_0^-(\mathbf{x})d\mathbf{x}$.

3.5. Model-based anysize-image-generation

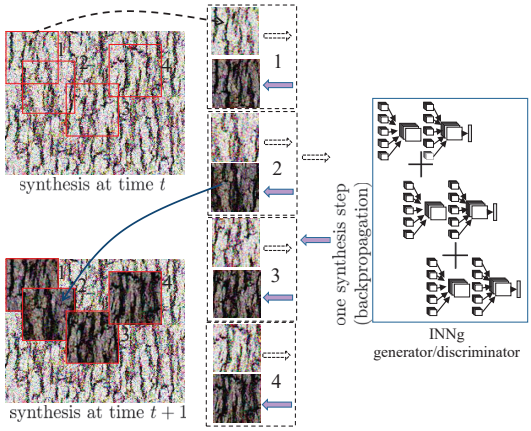

Figure 5. Illustration of model-based anysize-image-generation strategy.

Given a particularly sized image, anysize-image-generation within INNg allows one to generate/synthesize an image much larger than the given one. Patches extracted from the training images are used in the training of the discriminator. However, their position within the training (or pseudo-negative) image is not lost. In particular, when performing synthesis using backpropagation, updates to the pixel values are made by considering the average loss of all patches that overlap a given pixel. Thus, up to stage T, in order to consider the updates to the patch of $\mathbf{x}(i,j)$ centered at position (i,j) for image $\mathbf{I}$ of size $m_1 \times m_2$, we perform backpropagation on the patches to increase the probability:

$$p_T(\mathbf{I}) \propto \prod_{a=1}^{T}\prod_{i=1}^{m_1}\prod_{j=1}^{m_2} g_a(\mathbf{x}(i,j))p_0^-(\mathbf{x}(i,j)) \quad (15)$$

where $g_a(\mathbf{x}(i,j))$ (see Eq. 10) denotes the score of the patch of size e.g. 64×64 for $\mathbf{x}(i,j)$ under the discriminator at stage a. Fig. 5 gives an illustration for one stage of sampling. This allows us to synthesize much larger images by being able to enforce the coherence and interactions surrounding a particular pixel. In practice, we add stochasticity and efficiency to the synthesis process by randomly sampling these set of patches.

3.6. Model size reduction

As mentioned in Section 3.2.2 and 3.3, the process of INNg relies on keeping a snapshot of the classifier after each stage $t = 1 \ldots T$. This can pose a problem for both space and time efficiency. However, it may not be all necessary to keep each classifier around. Instead, one can pick some factor b such that the classifier is only saved every b stages. We accomplish this by: i) only saving the model to disk every b stages ii) initializing the sampling process at stage $b \times i + k$ ($i \in \mathbb{N}$ and $k < b$) from those of stage $b \times i$. We still, however, keep the pseudo-negatives around for training purposes to stabilize the process. We do find that later stages within each "mini stage" should be allowed more backpropagation steps during the synthesis step in order to compensate for the more complex optimization landscape. The image seen in Fig. 1 was generated from a reduced model ($b = 3$ and $T = 60$), resulting in only 20 classifiers. One can view this as a cascade of multiple of INNg-singles.

3.7. Comparison with GAN [14]

Next we compare INNg with a very interesting and popular line of work, generative adversarial neural networks (GAN) [14]. We summarize the key differences between INNg and GAN. Other recent algorithms alongside GAN [14, 34, 50, 6, 39] share similar properties with it.

- *Unified generator/discriminator vs. separate generator and discriminator*. INNg maintains a single model that is simultaneously a generator and a discriminator. The INNg generator therefore is able to self-evaluate the difference between its generated samples (pseudo-negatives) against the training data. This gives us an integrated framework to achieve competitive results in unsupervised and fully supervised learning with the generator and discriminator helping each other internally (not externally). GAN instead creates two convolutional networks, a generator and a discriminator.
- *Training*. Due to the internal competition between the generator and the discriminator, GAN is known to be hard to train [1]. INNg instead carries out a straightforward use of backpropagation in both the sampling and the classifier training stage, making the learning process direct. For example, all the textures by INNg shown in the experiments Fig. 2 and Fig. 6 are obtained under the identical setting without hyper-parameter tuning.
- *Speed*. GAN performs a forward pass to reconstruct an image, which is generally faster than INNg where synthesis is carried out using backpropagation. INNg is still practically feasible since it takes about 10 seconds to synthesize a batch of 50 images of 64×64 and around 30 seconds to synthesize a texture image of size 256×256, excluding the time to load the models.
- *Model size*. Since a sequence of CNN classifiers ($10 - 60$) are included in INNg, INNg has a much larger model complexity than GAN. This is an advantage of GAN over INNg. Our alternative INNg-single model maintains a single CNN classifier but its generative power is worse than those of INNg and GAN.

4. Experiments

We evaluate both INNg and INNg-single. In each method, we adopt the discriminator architecture of [34]

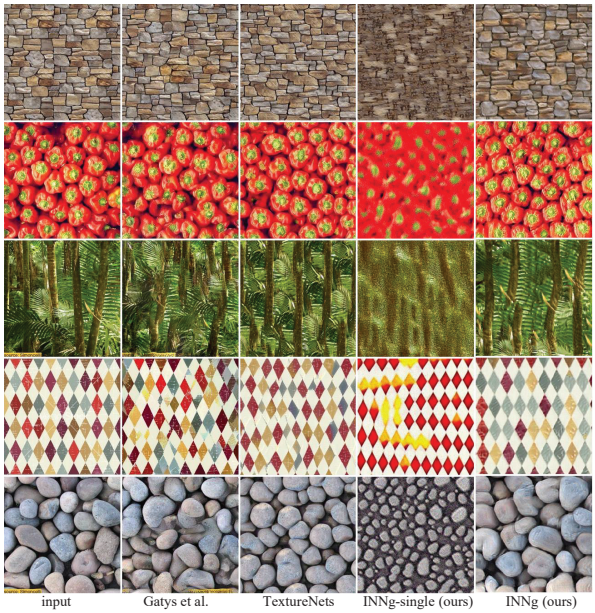

Figure 6. More texture synthesis results. Gatys et al. [13] and Texture Nets [42] results are from [42].

which takes an input size of $64 \times 64 \times 3$ in the RGB colorspace by four convolutional layers using 5×5 kernel sizes with the layers using 64, 128, 256 and 512 channels, respectively. We include batch normalization after each convolutional layer (excluding the first) and use leaky ReLU activations with leak slope 0.2. The classification layer flattens the input and finally feeds it into a sigmoid activation. This serves as the discriminator for the 64×64 patches we extract from the training image(s). Note that it is a general purpose architecture with no modifications made for a specific task in mind.

In texture synthesis and artistic style, we make use of the "anysize-image-generation" architecture by adding a "head" to the network that, at each forward pass of the network, randomly selects some number (equal to the desired batch size) of 64×64 random patches (possibly overlapping) from the full sized images and passes them to the discriminator. This allows us to retain the whole space of patches within a training image rather than select some subset of them in advance to use during training.

4.1. Texture modeling

Texture modeling/rendering is a long standing problem in computer vision and graphics [17, 51, 10, 33]. Here we are interested in statistical texture modeling [51, 46], instead of just texture rendering [10]. We train similar textures to [42]. Each source texture is resized to 256×256, used as the "positive" image in the training set; a set of 200 negative images are initially sampled from a normal distribution with $\sigma = 0.3$ of size 320×320 after adding padding of 32 pixels to each spatial dimension of the image to ensure each pixel of the 256×256 center has equal probability of being extracted in some patch. $1,000$ patches are extracted randomly across the training images and fed to the discriminator at each forward pass of the network (during training and synthesis stages) from a batch size of 100 patches — 50 random positives and negatives when training and 100 pseudo-negatives during synthesis. At each stage, our classifier is finetuned using stochastic gradient descent with learning rate 0.01 from the previous stage's classifier. Pseudo-negatives from more recent stages are chosen in mini-batches with higher probability than those of earlier stages in order to ensure the discriminator learns from its most recent mistakes as well as provide for more efficient training when the set of accumulated negatives has grown large in later stages. During the synthesis stage, pseudo-negatives are synthesized using the previous stage's pseudo-negatives as their initialization. Adam is used with a learning rate of 0.1 and $\beta = 0.5$ and stops early when the average probability of the patches under the discriminator becomes positive (across some window of steps, usually 20). We find this sampling strategy to attain a good balance in effectiveness and efficiency.

New textures are synthesized under INNg by: initializing from the normal distribution with $\sigma = 0.3$ followed by SGD based sampling via backpropagation using the saved networks for each stage, and feeding the resulting synthesis to the next stage. The number of patches is decided based on the image size to be synthesized, typically 10 patches when synthesizing a 256×256 image since this matches the average number of patches extracted per image during training. For INNg-single which consists of a single CNN classifier, SGD-based sampling is performed directly using this CNN classifier to transform the initial normal distribution to a desired texture.

Considering the results in Fig. 2, we see that INNg (60 CNN classifiers each with 4 layers) generates images of similar quality to [42], but of higher quality than those by Gatys et al. [13], Portilla & Simoncelli [33], and DCGAN [34]. In general, synthesis by INNg is usually more faithful to the structure of the input images.

More texture modeling results are provided in Fig. 6. We make an interesting observation that the "diamond" texture (the fourth row) generated by INNg (same as that used in Fig. 2) shows to preserve the near-regular patterns much better than the other methods. In the bottom row of Fig. 6, the "pebbles" synthesis of INNg captures the size variation as well as the variation in color and shading, better than TextureNets [42] and Gatys et al. [13].

4.2. Artistic style transfer

We also attempt to transfer artistic style as shown in [13]. However, our architecture makes no use of additional networks for content and texture transferring task uses a loss functions during synthesis to minimize

$$-\ln p(\mathbf{I}_{style} \mid \mathbf{I}) \propto \gamma \cdot ||\mathbf{I}_{style} - \mathbf{I}||_2 - (1-\gamma) \cdot \ln p^{-}_{style}(\mathbf{I}_{style}),$$

Figure 7. Artistic style transfer results using the "Starry Night" and "Scream" style on the image from Amsterdam.

where $\mathbf{I}$ is an input image and $\mathbf{I}_{style}$ is its stylized version, and $p^{-}_{style}(\mathbf{I})$ denotes the model learned from the training style image. We include a L_2 fidelity term during synthesis, weighted by a parameter γ, making $\mathbf{I}_{style}$ not too far away from the input image $\mathbf{I}$. We choose $\gamma = 0.3$ and average the L_2 difference between the original content image and the current stylized image at each step of synthesis. Two examples of the artistic style transfer are shown in Fig. 7.

4.3. Face modeling

Figure 8. Generated images learned on the CelebA dataset. The first, the second, and the third column are respectively results by DCGAN [34] (using tensorflow implementation [25]), INNg-single (1 CNN classifier with 4 layers), and INNg (12 CNN classifiers each with 4 layers).

INNg is also demonstrated on a face modeling task. The CelebA dataset [29] is used in our face modeling experiment, which consists of $202,599$ face images. We crop the center 64×64 patches in these images as our positive examples. For the classification step, we use stochastic gradient descent with learning rate 0.01 and a batch size of 100 images, which contains 50 random positives and 50 random negatives. For the synthesis step, we use the Adam optimizer with learning rate 0.02 and $\beta = 0.5$ and stop early when the pseudo-negatives cross the decision boundary. In Fig. 8, we show some face examples generated by the DCGAN model [34], our INNg-single model (1 CNN classifier with 4 conv layers), and our INNg model (12 CNN classifiers each with 4 conv layers).

4.4. SVHN unsupervised learning

Figure 9. Generated images learned on the SVHN dataset. The first, the second, and the third column are respectively results by DCGAN [34] (using tensorflow implementation [25]), INNg-single (1 CNN classifier with 4 layers), and INNg (10 CNN classifiers each with 4 layers).

The SVHN [32] dataset consists of color images of house numbers collected by Google Street View. The training set consists of $73,257$ images, the extra set consists of $531,131$ images, and the test set has $26,032$ images. The images are of the size 32×32. We combine the training and extra set as our positive examples for unsupervised learning. Following the same settings in the face modeling experiments, we shown some examples generated by the DCGAN model [34], INNg-single (1 CNN classifier with 4 conv layers), and INNg (10 CNN classifiers each with 4 conv layers).

4.5. Unsupervised feature learning

We perform the unsupervised feature learning and semi-supervised classification experiment by following the procedure outlined in [34]. We first train a model on the SVHN training and extra set in an unsupervised way, as in Section 4.4. Then, we train an L2-SVM on the learned representations of this model. The features from the last three convolutional layers are concatenated to form a 14336-dimensional feature vector. A $10,000$ example held-out validation set is taken from the training set and is used for model selection. The SVM classifier is trained on 1000 examples taken at random from the remainder of the training set. The test error rate is averaged over 100 different SVMs trained on random 1000-example training sets. Within the same setting, our INNg model achieves the test error rate of 32.81% and the DCGAN model achieves 33.13% (we ran the DCGAN code [25] in an identical setting as INNg for a fair comparison).

5. Conclusion

Generative modeling using introspective neural networks points to an encouraging direction for unsupervised image modeling that capitalizes on the power of discriminative deep convolutional neural networks. It can be adopted for a wide range of problems in computer vision. The source code will be made available on GitHub.

Acknowledgement. This work is supported by NSF IIS-1618477, NSF IIS-1717431, and a Northrop Grumman Contextual Robotics grant. We thank Saining Xie, Weijian Xu, Jun-Yan Zhu, Jiajun Wu, Stella Yu, Alexei Efros, Jitendra Malik, Sebastian Nowozin, Yingnian Wu, and Song-Chun Zhu for helpful discussions.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[2] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. *ICML unsupervised and transfer learning*, 27, 2012.

[3] Y. Bengio and Y. LeCun. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5), 2007.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. of machine Learning research*, 3:993–1022, 2003.

[5] L. Breiman. Random Forests. *Machine Learning*, 2001.

[6] A. Brock, T. Lim, J. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[7] T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *ICML*, 2014.

[8] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE PAMI*, 19(4):380–393, 1997.

[9] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. 2nd edition, 2000.

[10] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[11] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Comp. and Sys. Sci.*, 55(1), 1997.

[12] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics, 2001.

[13] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. In *NIPS*, 2015.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.

[15] U. Grenander. *General pattern theory-A mathematical study of regular structures*. Clarendon Press, 1993.

[16] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, volume 1, page 6, 2010.

[17] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *SIGGRAPH*, pages 229–238, 1995.

[18] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh. Unsupervised discovery of nonlinear structure using contrastive backpropagation. *Cognitive science*, 30(4):725–731, 2006.

[19] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

[20] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18:1527–1554, 2006.

[21] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[22] T. Jebara. Machine learning: discriminative and generative, 2012.

[23] L. Jin, J. Lazarow, and Z. Tu. Introspective classifier learning: Empower generatively. In *arXiv preprint arXiv:*, 2017.

[24] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[25] T. Kim. DCGAN-tensorflow. https://github.com/carpedm20/DCGAN-tensorflow, 2016.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural Computation*, 1989.

[28] P. Liang and M. I. Jordan. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In *ICML*, 2008.

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

[30] S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

[31] A. Mordvintsev, C. Olah, and M. Tyka. Deepdream - a code example for visualizing neural networks. *Google Research*, 2015.

[32] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[33] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int'l j. of computer vision*, 40(1):49–70, 2000.

[34] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015.

[35] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[36] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *CVPR*, 2005.

[37] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *arXiv preprint arXiv:1606.03498*, 2016.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[39] I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf. Adagan: Boosting generative models. *arXiv:1701.02386*, 2017.

[40] Z. Tu. Learning generative models via discriminative approaches. In *CVPR*, 2007.

[41] Z. Tu, K. L. Narr, P. Dollár, I. Dinov, P. M. Thompson, and A. W. Toga. Brain anatomical structure segmentation by hybrid discriminative/generative models. *IEEE Tran. on Medical Imag.*, 2008.

[42] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016.

[43] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[44] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011.

[45] M. Welling, R. S. Zemel, and G. E. Hinton. Self supervised boosting. In *NIPS*, 2002.

[46] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *arXiv:1609.09408*, 2016.

[47] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. A theory of generative convnet. In *ICML*, 2016.

[48] A. L. Yuille, P. W. Hallinan, and D. S. Cohen. Feature extraction from faces using deformable templates. *International journal of computer vision*, 8(2):99–111, 1992.

[49] A. L. Yuille and D. Kersten. Vision as bayesian inference: analysis by synthesis? *Trends in cognitive sciences*, 10(7):301–308, 2006.

[50] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. *arXiv:1609.03126*, 2016.

[51] S. C. Zhu, Y. N. Wu, and D. Mumford. Minimax entropy principle and its application to texture modeling. *Neural Computation*, 9(8):1627–1660, 1997.