Cooperative Training of Fast Thinking Initializer and Slow Thinking Solver for Conditional Learning

Jianwen Xie*, Zilong Zheng*, Xiaolin Fang, Song-Chun Zhu, *Fellow, IEEE,* and Ying Nian Wu

Abstract—This paper studies the problem of learning the conditional distribution of a high-dimensional output given an input, where the output and input may belong to two different domains, e.g., the output is a photo image and the input is a sketch image. We solve this problem by cooperative training of a fast thinking initializer and slow thinking solver. The initializer generates the output directly by a non-linear transformation of the input as well as a noise vector that accounts for latent variability in the output. The slow thinking solver learns an objective function in the form of a conditional energy function, so that the output can be generated by optimizing the objective function, or more rigorously by sampling from the conditional energy-based model. We propose to learn the two models jointly, where the fast thinking initializer serves to initialize the sampling of the slow thinking solver, and the solver refines the initial output by an iterative algorithm. The solver learns from the difference between the refined output and the observed output, while the initializer learns from how the solver refines its initial output. We demonstrate the effectiveness of the proposed method on various conditional learning tasks, e.g., class-to-image generation, image-to-image translation, and image recovery. The advantage of our method over GAN-based methods is that our method is equipped with a slow thinking process that refines the solution guided by a learned objective function.

Index Terms—Deep generative models; Cooperative learning; Energy-based models; Langevin dynamics; Conditional learning.

1 Introduction

1.1 Background and motivation

When we learn to solve a problem, we can learn to directly map the problem to the solution. This amounts to fast thinking, which underlies reflexive or impulsive behavior, or muscle memory, and it can happen when one is emotional or under time constraint. We may also learn an objective function or value function that assigns values to candidate solutions, and we optimize the objective function by an iterative algorithm to find the most valuable solution. This amounts to slow thinking, which underlies planning, search or optimal control, and it can happen when one is calm or have time to think through.

In this paper, we study the supervised learning of the conditional distribution of a high-dimensional output given an input, where the output and input may belong to two different domains. For instance, the output may be an image, while the input may be a class label, a sketch, or an image from another domain. The input defines the problem, and the output is the solution. We also refer to the input as the source or condition, and the output as the target.

We solve this problem by learning two models cooperatively. One model is an initializer. It generates the output directly by a non-linear transformation of the input as well as a noise vector, where the noise vector is to account for variability or uncertainty in the output. This amounts to fast thinking because the conditional generation is accomplished by direct mapping. The other model is a solver. It learns an objective function in the form of a conditional energy function, so that the output can be generated by optimizing the objective function, or more rigorously by sampling from the conditional energy-based model, where the sampling is to account for variability and uncertainty. This amounts to slow thinking because the sampling is accomplished by an iterative algorithm such as Langevin dynamics [1], which is an example of Markov chain Monte Carlo (MCMC) [2], [3]. We propose to learn the two models jointly, where the initializer serves to initialize the sampling process of the solver, and the solver refines the initial solution by an iterative algorithm. The solver learns from the difference between the refined solution and the observed solution, while the initializer learns from the difference between the initial solution and the refined solution.

Figure 1 conveys the basic idea. The algorithm iterates two steps, a solving step and a learning step. The solving step consists of two stages: **Initialize**: The initializer generates the initial solution according to the given condition by direct mapping, such as ancestral sampling. **Solve**: The solver refines the initial solution according to the same condition by an iterative algorithm, such as Langevin sampling, which minimizes the objective function. The learning step also consists of two parts: **Learn-mapping**: The initializer updates its mapping by learning from how the solver refines its initial solution, for the purpose of providing better initial solution for the solver in the next iteration. **Learn-objective**: The solver updates its objective function by shifting its high value region from the refined solution to the observed solution, for the sake of matching the refined solution to the observed one in terms of value in the next iteration.

Figure 2(a) illustrates Learn-mapping step. In the Initialization

- *J. Xie is with the Cognitive Computing Lab, Baidu Research, Bellevue, WA 98004, USA. E-mail: jianwen@ucla.edu*
- *Z. Zheng is with the Department of Computer Science, University of California, Los Angeles, CA 90095, USA. E-mail: z.zheng@ucla.edu*
- *X. Fang is with the Department of Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. E-mail: xiaolinf@csail.mit.edu*
- *S.-C. Zhu is with Tsinghua University and Peking University, Beijing, China. E-mail: sczhu@stat.ucla.edu*
- *Y. N. Wu is with the Department of Statistics, University of California, Los Angeles, CA 90095, USA. E-mail: ywu@stat.ucla.edu*
- ** indicates equal contributions.*

Manuscript received 24 Dec. 2019; revised 29 Jan. 2021; accepted 9 Mar. 2021.

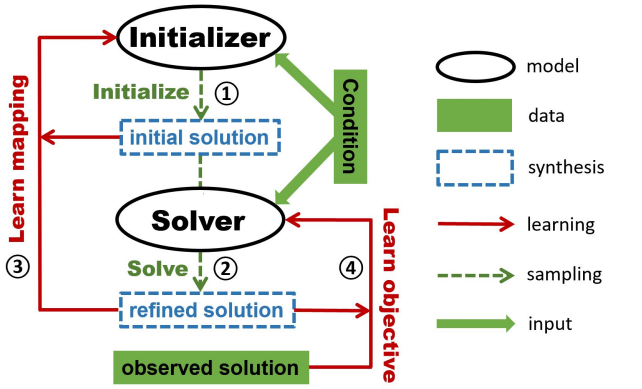
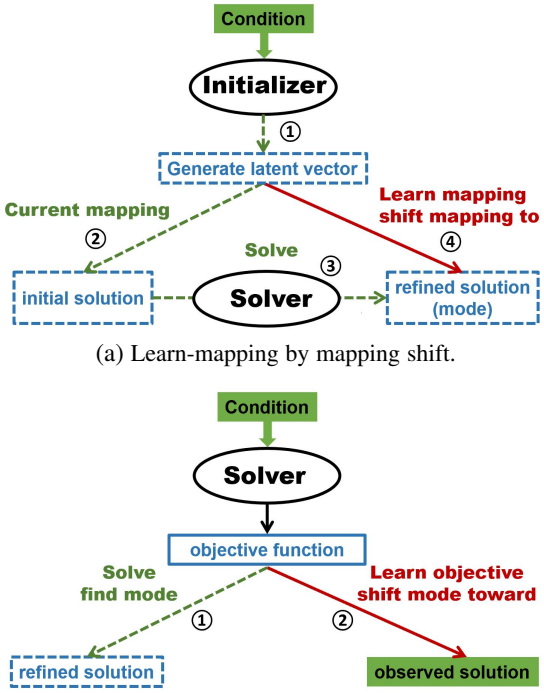


Fig. 1. Diagram of fast thinking and slow thinking conditional learning. Given a condition, the initializer initializes the solver, which refines the initial solution. The initializer provides the initial solution via direct mapping (see ①), i.e., ancestral sampling, which is a fast thinking process, while the solver refines the initial solution via Langevin sampling that optimizes the objective function (②), which is a slow thinking process. The initializer learns the mapping from the solver's refinement (see ③), while the solver learns the objective function by comparing to the observed solution (see ④).



(a) Learn-mapping by mapping shift.

(b) Learn-objective by objective shift.

Fig. 2. Learning step. (a) **Learn-mapping by mapping shift**: In the initialize stage, the initializer generates the latent noise vector (see ①), and maps it along with the input condition to the initial solution (see ②). The solver outputs the refined solution after refining the initial solution (see ③). The learning of the initializer is to shift its mapping from the initial solution toward the refined solution (see ④). (b) **Learn-objective by objective shift**: In the solve stage, the solver finds high value region or mode in its objective function via an iterative algorithm (see ①). Those modes corresponds to the refined solution. The learning of the solver is to shift the high value region or mode of its objective function from the refined solution toward the observed solution (see ②).

step, the initializer generates the latent noise vector, which, together with the input condition, is mapped to the initial solution. In the Learn-mapping step, the initializer updates its parameters so that it maps the input condition and the latent vector to the refined solution, in order to absorb the refinement made by the solver. Because the latent vector is known, it does not need to be inferred and the learning is easy. In other words, keeping the same mapping source, the initializer shifts its mapping target from the initial solution toward the refined solution.

Figure 2(b) illustrates Learn-objective step. In the Solve step, the solver finds the refined solution at high value region around a mode of the objective function. In the Learn-objective step, the solver updates its parameters so that the objective function shifts its high value region around the mode toward the observed solution, so that in the next iteration, the refined solution will get closer to the observed solution.

The solver shifts its mode toward the observed solution, while inducing the initializer maps the input condition and the latent vector to its mode. Learning an initializer is like mimicking "how", while learning a solver is like trying to understand "why" in terms of goal or value underlying the action.

Why slow thinking solver? The reason we need a solver in addition to an initializer is that it is often easier to learn the objective function than learning to generate the solution directly, since it is always easier to demand or desire something than to actually produce something directly. Because of its relative simplicity, the learned objective function can be more generalizable than the learned initializer. For instance, in an unfamiliar situation, we tend to be tentative, relying on slow thinking planning rather than fast thinking habit.

Efficiency. Even though we use the wording "slow thinking", it is only relative to "fast thinking". In fact, the slow thinking solver is usually fast enough, especially if it is jumpstarted by fasting thinking initializer, and there is no problem scaling up our method to big datasets. Therefore the time efficiency of the slow thinking method is not a concern.

Student-teacher v.s. actor-critic. We may consider the initializer as a student model, and the solver as a teacher model. The teacher refines the initial solution of the student by a refinement process, and distills the refinement process into the student. This is different from the actor-critic relationship in (inverse) reinforcement learning [4], [5], [6] because the critic does not refine the actor's solution by a slow thinking process.

Cooperative learning v.s. adversarial learning. Our framework, belonging to cooperative learning [7], [8], jointly learns a conditional energy-based model as the slow thinking solver and a conditional generator as the fast thinking initializer. This is essentially different from the conditional generative adversarial net (cGAN) [9], [10], [11], where a conditional discriminator is simultaneously learned to help train the conditional generator. Our framework simultaneously trains both models and keeps both of them after training, while cGAN discards its discriminator once the generator model is well trained. In other words, our framework trains both the slow thinking solver (i.e., the energy-based model) and the fast thinking initializer (i.e., the generator), while cGAN only desires a fast thinking model (i.e., the generator). Thus, the advantage of our method over cGAN is that our method is equipped with a refinement process guided by the learned energy-based model.

We apply our learning method to various conditional learning tasks, such as class-to-image generation, image-to-image translation, image inpainting, etc. Our experiments show that the proposed method is effective compared to other methods, such as those based on GANs [9].

Amortized computation and temporal difference learning. The solver is an iterative computing process. The initializer is an amortization of this process. The learning of the initializer can be considered temporal difference learning, where the finite steps of refinements produce the temporal difference to be distilled into the initializer.

Learning from external and internal data. The learning of the conditional energy function is from the training data, which we may call the external data. The learning of the initializer can be considered as learning from the internal data produced by the computational process of the solver.

Policy, value, and control. The initializer is similar to a policy network. The solver is similar to an iterative optimal control or planning process based on a value network. The conditional energy function is similar to a cost function.

Vector-valued initializer and scalar-valued conditional energy function. The initializer learns a mapping from an input to a high-dimensional output. The solver learns a scalar-valued conditional energy function. It is much easier to learn a scalar-valued function than a high-dimensional vector-valued mapping, so that the iterative refinement process guided by the learned energy function improves the initializer.

Contributions. This paper proposes a novel method for supervised learning of high-dimensional conditional distributions by learning a fast thinking initializer and a slow thinking solver. We show the effectiveness of our method on conditional image generation and recovery tasks. Perhaps more importantly,

- We propose a different method for conditional learning than GAN-based methods. Unlike GANs, our method has a learned value function (i.e., the energy function in the conditional energy-based model) to guide a slow thinking process to refine the solution of the initializer (i.e., conditional generator). We demonstrate the benefit of such a refinement on various image synthesis tasks.
- The proposed framework is generic and can be applied to a broad range of artificial intelligence problems that can be modeled via a conditional learning framework, e.g., inverse optimal control, etc. The interaction between the fast thinking initializer and the slow thinking solver can be of interest to cognitive science.
- This is the first paper to study conditional learning via a model-based Initializer-solver framework. It is fundamental and important to AI community.

1.2 Related work

The following themes are closely related to our research. We will briefly review each of them and connect them with our work.

Conditional adversarial learning. Generative Adversarial Networks (GANs) [9] proposed by Goodfellow et al. have demonstrated promising results of image generation in [12], which belongs to unconditional learning, in which no supervision signals are used. With the success of adversarial learning, the conditional version of GAN (i.e., conditional GAN or cGAN) [13] has become a popular framework for supervised conditional learning, and it has been successfuly appplied to different scenarios that can be modeled in the context of conditional learning. For example, [11], [14] use conditional GANs for image synthesis based on class labels. [13], [15] study text-conditioned image synthesis. Other examples include image-to-image translation [10], semantic-image-to-photo translation [16], super-resolution [17], and video-to-video synthesis [18], etc. Our work studies similar problems. The major difference between the conditional GAN and our method is that ours is based on a conditional energy function that serves as an objective function and an iterative algorithm, which is the Langevin dynamics guided by this objective function. This iterative process corresponds to slow thinking. Existing adversarial learning methods do not involve this slow thinking refinement process.

Cooperative learning. Just as the conditional GAN is inspired by the original GAN [9], our learning method is inspired by the recent work of generative cooperative networks (CoopNets) [7], [8], where the models are unconditioned. Specifically, the CoopNets framework consists of an unconditional energy-based model and an unconditional latent variable model, and jointly trains both models via MCMC teaching [7], where the latent variable model learns to initialize the MCMC sampling of the energy-based model. While unconditioned generation is interesting, conditional generation and recovery is much more useful in applications. It is also much more challenging because we need to incorporate the input condition into both the initializer and the solver. Thus our method is a substantial generalization of the CoopNets [7], and our extensive experiments convincingly demonstrate the usefulness of our method, which in the meantime provides a different methodology from GAN-based methods. Our work is the first to study conditional cooperative learning, and propose the fast thinking and slow thinking framework as a conditional version of CoopNets.

Conditional random field. The objective function and the conditional energy-based model can also be considered a form of conditional random field [19]. Unlike traditional conditional random field, our conditional energy function is defined by a trainable deep network, and its MCMC sampling process is jumpstarted by a non-iterative initializer.

Energy-based generative neural nets. Our slow thinking solver is related to energy-based generative neural nets [20], [21], [22], [23], [24], [25], [26], which are energy-based models (EBMs) with energy functions parameterized by deep neural nets, and trained by MCMC-based maximum likelihood learning. [20] is the first to learn EBMs parametrized by modern ConvNets by maximum likelihood estimation via Langevin dynamics, and also investigates ReLU [27] with Gaussian reference in the proposed model that is called generative ConvNet. [21] proposes a multi-grid sampling and learning method for training generative ConvNets. The spatial-temporal generative ConvNet proposed in [22], [23] further generalizes the generative ConvNet of images in [20] to modeling dynamic patterns, e.g., videos or image sequences, by parameterizing the energy function with a bottom-up spatial-temporal ConvNet. [24], [28] develops a volumetric version of the energy-based generative neural net, which is called generative VoxelNet, for 3D object patterns. Recently, [25] investigates training the energy-based generative ConvNet with a short-run MCMC. All models mentioned above are unconditioned EBMs, while our solver is a conditioned EBM jointly trained with a conditional latent variable model serving as an approximate conditional sampler.

Inverse reinforcement learning. Our method is related to inverse reinforcement learning and inverse optimal control [4], [5], where the initializer corresponds to the policy, and the solver corresponds to the planning or optimal control. Unlike the action space in reinforcement learning, the output in our work is of a much higher dimension, a fact that also distinguishes our work from common supervised learning problem such as classification. As a result, the initializer needs to transform a latent noise vector (along with

an input condition) to generate the initial solution, and this is different from the policy in reinforcement learning, where the policy is defined by the conditional distribution of action given state, without resorting to a latent vector.

Unsupervised conditional learning. Some methods study unsupervised conditional learning, where the inputs and outputs are unpaired in the training set. For example, CycleGAN [29] jointly trains two GANs and enforces a cycle-consistency regularization between them to learn a two-way translator between two image collections in the absence of paired examples. AlignFlow [30] adopts normalizing flow models [31], [32] to solve this problem. Recently, CycleCoopNets [33] tackles the unpaired translation problem based on the framework of cooperative learning. Our work belongs to supervised conditional learning, where the correspondence between source domain and target domain is given and used as supervision during training.

2 COOPERATIVE CONDITIONAL LEARNING

Let Y be the D-dimensional output signal of the target domain, and C be the input signal of the source domain, where "C" stands for "condition". C defines the problem, and Y is the solution. Our goal is to learn the conditional distribution $p(Y|C)$ of the target signal (solution) Y given the source signal C (problem) as the condition. We shall learn $p(Y|C)$ from the training dataset of the pairs $\{(Y_i, C_i), i = 1, ..., n\}$ with the fast thinking initializer and slow thinking solver.

2.1 Slow thinking solver

The solver is based an objective function or value function $f(Y, C; \theta)$ defined on (Y, C). $f(Y, C; \theta)$ can be parametrized by a bottom-up convolutional network (ConvNet) where θ collects all the weight and bias parameters. Serving as a negative energy function, $f(Y, C; \theta)$ defines a joint energy-based model [20]:

$$p(Y, C; \theta) = \frac{1}{Z(\theta)} \exp\left[f(Y, C; \theta)\right], \tag{1}$$

where $Z(\theta) = \int \exp\left[f(Y, C; \theta)\right] dY dC$ is the normalizing constant.

Fixing the source signal C, $f(Y, C; \theta)$ defines the value of the solution Y for the problem defined by C, and $-f(Y, C; \theta)$ defines the conditional energy function. The conditional probability is given by

$$\begin{aligned} p(Y|C; \theta) &= \frac{p(Y, C; \theta)}{p(C; \theta)} = \frac{p(Y, C; \theta)}{\int p(Y, C; \theta) dY} \\ &= \frac{1}{Z(C, \theta)} \exp\left[f(Y, C; \theta)\right], \end{aligned} \tag{2}$$

where $Z(C, \theta) = Z(\theta)p(C; \theta)$. The learning of this model seeks to maximize the conditional log-likelihood function

$$L(\theta) = \frac{1}{n} \sum_{i=1}^{n} \log p(Y_i|C_i; \theta), \tag{3}$$

whose gradient $L'(\theta)$ is

$$\sum_{i=1}^{n} \left\{ \frac{\partial}{\partial \theta} f(Y_i, C_i; \theta) - \mathrm{E}_{p(Y|C_i, \theta)} \left[\frac{\partial}{\partial \theta} f(Y, C_i; \theta) \right] \right\}, \tag{4}$$

where $\mathrm{E}_{p(Y|C;\theta)}$ denotes the expectation with respect to $p(Y|C, \theta)$. The identity underlying (4) is $\frac{\partial}{\partial \theta} \log Z(C, \theta) = \mathrm{E}_{p(Y|C,\theta)} \left[\frac{\partial}{\partial \theta} f(Y, C; \theta) \right]$.

The expectation in (4) is analytically intractable and can be approximated by drawing samples from $p(Y|C, \theta)$ and then computing the Monte Carlo average. This can be solved by an iterative algorithm, which is a slow thinking process. One solver is the Langevin dynamics for sampling $Y \sim p(Y|C, \theta)$. It iterates the following step:

$$Y_{\tau+1} = Y_\tau + \frac{\delta^2}{2} \frac{\partial}{\partial Y} f(Y_\tau, C; \theta) + \delta U_\tau, \tag{5}$$

where τ indexes the time steps of the Langevin dynamics, δ is the step size, and $U_\tau \sim \mathrm{N}(0, I_D)$ is Gaussian white noise. D is the dimensionality of Y. A Metropolis-Hastings acceptance-rejection step can be added to correct for finite δ. The Langevin dynamics is gradient descent on the energy function, plus noise for diffusion so that it samples the distribution instead of being trapped in the local modes.

For each observed condition C_i, we run the Langevin dynamics according to (5) to obtain the corresponding synthesized example $\tilde{Y}_i$ as a sample from $p(Y|C_i, \theta)$. The Monte Carlo approximation to $L'(\theta)$ is

$$L'(\theta) \approx \frac{\partial}{\partial \theta} \left[\frac{1}{n} \sum_{i=1}^{n} f(Y_i, C_i; \theta) - \frac{1}{n} \sum_{i=1}^{n} f(\tilde{Y}_i, C_i; \theta) \right]. \tag{6}$$

We can then update $\theta^{(t+1)} = \theta^{(t)} + \gamma_t L'(\theta^{(t)})$.

Objective shift: The above gradient ascent algorithm is to increase the average value of the observed solutions versus that of the refined solutions, i.e., on average, it shifts high value region or mode of $f(Y, C_i; \theta)$ from the generated solution $\tilde{Y}_i$ toward the observed solution Y_i.

The convergence of such a stochastic gradient ascent algorithm has been studied by [34].

2.2 Fast thinking initializer

The initializer is of the following form:

$$X \sim \mathrm{N}(0, I_d),\ Y = g(X, C; \alpha) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D), \tag{7}$$

where X is the d-dimensional latent noise vector, and $g(X, C; \alpha)$ is a top-down ConvNet defined by the parameters α. The ConvNet g maps the observed condition C and the latent noise vector X to the signal Y directly. If the source signal C is of high dimensionality, we can parametrize g by an encoder-decoder structure: we first encode C into a latent vector Z, and then we map (X, Z) to Y by a decoder. Given C, we can generate Y from the conditional generator model by direct sampling, i.e., first sampling X from its prior distribution, and then mapping (X, Z) into Y directly. This is fast thinking without iteration.

We can learn the initializer from the training pairs $\{(Y_i, C_i), i = 1, ..., n\}$ by maximizing the conditional log-likelihood $L(\alpha) = \frac{1}{n} \sum_{i=1}^{n} \log p(Y_i|C_i, \alpha)$, where $p(Y|C, \alpha) = \int p(X)p(Y|C, X, \alpha)dX$. The learning algorithm iterates the following two steps. (1) Sample X_i from $p(X_i|Y_i, C_i, \alpha)$ by Langevin dynamics. (2) Update α by gradient descent on $\frac{1}{n} \sum_{i=1}^{n} \|Y_i - g(X_i, C_i; \alpha)\|^2$. See [35] for details.

2.3 Cooperative training of initializer and solver

The initializer and the solver can be trained jointly as follows.

(1) The initializer supplies initial samples for the MCMC of the solver. For each observed condition input C_i, we first generate $\hat{X}_i \sim \mathrm{N}(0, I_d)$, and then generate the initial solution

$\hat{Y}_i = g(\hat{X}_i, C_i; \alpha) + \epsilon_i$. If the current initializer is close to the current solver, then the generated $\{\hat{Y}_i, i = 1, ..., n\}$ should be a good initialization for the solver to sample from $p(Y|C_i, \theta)$, i.e., starting from the initial solutions $\{\hat{Y}_i, i = 1, ..., n\}$, we run Langevin dynamics for l steps to get the refined solutions $\{\tilde{Y}_i, i = 1, ..., n\}$. These $\{\tilde{Y}_i\}$ serve as the synthesized examples from $p(Y|C_i)$ and are used to update θ in the same way as we learn the solver model in equation (6) for objective shifting.

(2) The initializer then learns from the MCMC. Specifically, the initializer treats $\{(\tilde{Y}_i, C_i), i = 1, ..., n\}$ produced by the MCMC as the training data. The key is that these $\{\tilde{Y}_i\}$ are obtained by the Langevin dynamics initialized from the $\{\hat{Y}_i, i = 1, ..., n\}$, which are generated by the initializer with *known* latent noise vectors $\{\hat{X}_i, i = 1, ..., n\}$. Given $\{(\hat{X}_i, \tilde{Y}_i, C_i), i = 1, ..., n\}$, we can learn α by minimizing $\frac{1}{n}\sum_{i=1}^{n} \|\tilde{Y}_i - g(\hat{X}_i, C_i; \alpha)\|^2$, which is a nonlinear regression of $\tilde{Y}_i$ on $(\hat{X}_i, C_i)$. This can be accomplished by gradient descent

$$\Delta\alpha \propto -(\tilde{Y}_i - g(\hat{X}_i, C_i; \alpha)\frac{\partial}{\partial \alpha} g(\hat{X}_i, C_i; \alpha). \tag{8}$$

Mapping shift: Initially $g(X, C; \alpha)$ maps $(\hat{X}_i, C_i)$ to the initial solution $\hat{Y}_i$. After updating α, $g(X, C; \alpha)$ maps $(\hat{X}_i, C_i)$ to the refined solution $\tilde{Y}_i$. Thus the updating of α absorbs the MCMC transitions that change $\hat{Y}_i$ to $\tilde{Y}_i$. In other words, we distill the MCMC transitions of the refinement process into $g(X, C; \alpha)$.

Algorithm 1 presents a description of the conditional learning with two models. See Figures 1 and 2 for illustrations.

Both computations can be carried out by back-propagation, and the whole algorithm is in the form of alternating back-propagation.

Algorithm 1 Cooperative conditional learning

Input:

(1) training examples $\{(Y_i, C_i), i = 1, ..., n\}$

(2) numbers of Langevin steps l

(3) number of learning iterations T.

Output:

(1) learned parameters θ and α,

(2) generated examples $\{\hat{Y}_i, \tilde{Y}_i, i = 1, ..., n\}$.

1: $t \leftarrow 0$, initialize θ and α.
2: **repeat**
3: **Initialization by mapping**: For $i = 1, ..., n$, generate $\hat{X}_i \sim \mathrm{N}(0, I_d)$, and generate the initial solution $\hat{Y}_i = g(\hat{X}_i, C_i; \alpha^{(t)}) + \epsilon_i$.
4: **Solve based on objective**: For $i = 1, ..., n$, starting from $\hat{Y}_i$, run l steps of Langevin dynamics to obtain the refined solution $\tilde{Y}_i$, each step following equation (5).
5: **Learn-objective by objective shift**: Update $\theta^{(t+1)} = \theta^{(t)} + \gamma_t L'(\theta^{(t)})$, where $L'(\theta^{(t)})$ is computed according to (6).
6: **Learn-mapping by mapping shift**: Update $\alpha^{(t+1)} = \alpha^{(t)} + \gamma_t \Delta\alpha^{(t)}$, where $\Delta\alpha^{(t)}$ is computed according to (8)
7: Let $t \leftarrow t + 1$
8: **until** $t = T$

In Algorithm 1, the conditional energy- model is the primary model for conditional synthesis or recovery by MCMC sampling. The conditional generator model plays an assisting role to initialize the MCMC sampling.

3 THEORETICAL UNDERPINNING

This section presents theoretical underpinnings of the model and the learning algorithms presented in the previous section. Readers who are more interested in applications and experiments can jump to the next section.

3.1 Kullback-Leibler divergence

The Kullback-Leibler divergence between two distributions $p(x)$ and $q(x)$ is defined as $\mathrm{KL}(p\|q) = \mathrm{E}_p[\log(p(X)/q(X))]$.

The Kullback-Leibler divergence between two conditional distributions $p(y|x)$ and $q(y|x)$ is defined as

$$\begin{aligned} \mathrm{KL}(p\|q) &= \mathrm{E}_p\left[\log \frac{p(Y|X)}{q(Y|X)}\right] &(9)\\ &= \int \log \frac{p(y|x)}{q(y|x)} p(x, y)dxdy, &(10)\end{aligned}$$

where the expectation is over the joint distribution $p(x, y) = p(x)p(y|x)$.

3.2 Slow thinking solver

The slow thinking solver model is

$$\begin{aligned} p(Y|C; \theta) &= \frac{p(Y, C; \theta)}{p(C; \theta)} = \frac{p(Y, C; \theta)}{\int p(Y, C; \theta)dY} \\ &= \frac{1}{Z(C; \theta)} \exp\left[f(Y, C; \theta)\right], \end{aligned} \tag{11}$$

where

$$Z(C; \theta) = \int \exp\left[f(Y, C; \theta)\right] dY \tag{12}$$

is the normalizing constant and is analytically intractable.

Suppose the training examples $\{(Y_i, C_i), i = 1, ..., n\}$ are generated by the true joint distribution $f(Y, C)$, whose conditional distribution is $f(Y|C)$.

For large sample $n \to \infty$, the maximum likelihood estimation of θ is to minimize the Kullback-Leibler divergence

$$\min_\theta \mathrm{KL}(f(Y|C)\|p(Y|C; \theta)). \tag{13}$$

In practice, the expectation with respect to $f(Y, C)$ is approximated by the sample average. The difficulty with $\mathrm{KL}(f(Y|C)\|p(Y|C; \theta))$ is that the $\log Z(C; \theta)$ term is analytically intractable, and its derivative has to be approximated by MCMC sampling from the model $p(Y|C; \theta)$.

3.3 Fast thinking initializer

The fast thinking initializer is

$$X \sim \mathrm{N}(0, I_d),\ Y = g(X, C; \alpha) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D). \tag{14}$$

We use the notation $q(Y|C; \alpha)$ to denote the resulting conditional distribution. It is obtained by

$$q(Y|C; \alpha) = \int q(X)q(Y|X, C; \alpha)dX, \tag{15}$$

which is analytically intractable.

For large sample, the maximum likelihood estimation of α is to minimize the Kullback-Leibler divergence

$$\min_\alpha \mathrm{KL}(f(Y|C)\|q(Y|C; \alpha)). \tag{16}$$

Again, the expectation with respect to $f(Y, C)$ is approximated by the sample average. The difficulty with $\mathrm{KL}(f(Y|C)\|q(Y|C;\alpha))$ is that $\log q(Y|C;\alpha)$ is analytically intractable, and its derivative has to be approximated by MCMC sampling of the posterior $q(X|Y, C;\alpha)$.

3.4 Objective shift: modified contrastive divergence

Let $M(Y_1|Y_0, C;\theta)$ be the transition kernel of the finite-step MCMC that refines the initial solution Y_0 to the refined solution Y_1. Let $(M_\theta q)(Y_1|C;\alpha) = \int M(Y_1|Y_0, C;\theta)q(Y_0|C;\alpha)dY_0$ be the distribution obtained by running the finite-step MCMC from $q(Y_0|C;\alpha)$.

Given the current initializer $q(Y|C;\alpha)$, the objective shift updates θ_t to θ_{t+1}, and the update approximately follows the gradient of the following modified contrastive divergence [7], [36]

$$\begin{aligned}&\mathrm{KL}(f(Y|C)\|p(Y|C;\theta))\\ -&\mathrm{KL}((M_{\theta_t}q)(Y|C;\alpha)\|p(Y|C;\theta)).\end{aligned} \tag{17}$$

Compare (17) with the MLE (11), (17) has the second divergence term $\mathrm{KL}((M_{\theta_t}q)(Y|C;\alpha)\|p(Y|C;\theta))$ to cancel the $\log Z(C;\theta)$ term, so that its derivative is analytically tractable. The learning is to shift $p(Y|C;\theta)$ or its high value region around the mode from the refined solution provided by $(M_{\theta_t}q)(Y|C;\alpha)$ toward the observed solution given by $f(Y|C)$. If $(M_{\theta_t}q)(Y|C;\alpha)$ is close to $p(Y|C;\theta)$, then the second divergence is close to zero, and the learning is close to MLE update.

3.5 Mapping shift: distilling MCMC

Given the current solver model $p(Y|C;\theta)$, the mapping shift updates α_t to α_{t+1}, and the update approximately follows the gradient of

$$\mathrm{KL}((M_\theta q)(Y|C;\alpha_t)\|q(Y|C;\alpha)). \tag{18}$$

This update distills the MCMC transition M_θ into the model $q(Y|C;\alpha)$. In the idealized case where the above divergence can be minimized to zero, then $q(Y|C;\alpha_{t+1}) = (M_\theta q)(Y|C;\alpha_t)$. The limiting distribution of the MCMC transition M_θ is $p(Y|C;\theta)$, thus the cumulative effect of the above update is to lead $q(Y|C;\alpha)$ close to $p(Y|C;\theta)$.

Compare (18) to the MLE (14), the training data distribution becomes $(M_\theta q)(Y|C;\alpha_t)$ instead of $f(Y|C)$. That is, $q(Y|C;\alpha)$ learns from how M_θ refines it. The learning is accomplished by mapping shift where the generated latent vector X is known, thus does not need to be inferred (or the Langevin inference algorithm can initialize from the generated X). In contrast, if we are to learn from $f(Y|C)$, we need to infer the unknown X by sampling from the posterior distribution.

In the limit, if the algorithm converges to a fixed point, then the resulting $q(Y|C;\alpha)$ minimizes $\mathrm{KL}((M_\theta q)(Y|C;\alpha)\|q(Y|C;\alpha))$, that is, $q(Y|C;\alpha)$ seeks to be the stationary distribution of the MCMC transition M_θ, which is $p(Y|C;\theta)$.

If the learned $q(Y|C;\alpha)$ is close to $p(Y|C;\theta)$, then $(M_{\theta_t}q)(Y|C;\alpha)$ is even closer to $p(Y|C;\theta)$. Then the learned $p(Y|C;\theta)$ is close to MLE because the second divergence term in (17) is close to zero.

4 Experiments

Project page: The code and more results can be found at http://www.stat.ucla.edu/~jxie/CCoopNets/

We test the proposed framework for conditional learning on a variety of vision tasks. According to the form of the conditional learning, we organize the experiments into two parts. In the first part (Experiment 1), we study conditional learning for a mapping from category (i.e., one-hot vector) to image, e.g., image generation conditioned on image class, while in the second part (Experiment 2), we study conditional learning for a mapping from image to image, e.g., image-to-image translation. We propose a specific network architecture of our model in each experiment due to the different forms of input-output domains. Unlike the unconditioned cooperative learning framework [7], [8], the conditioned framework needs to find a proper way to fuse the condition input C into both the bottom-up ConvNet f in the solver and the top-down ConvNet g in the initializer, for the sake of capturing accurate conditioning information. An improper design can cause not only unrealistic but also condition-mismatched synthesized results.

4.1 Experiment 1: Category → Image

4.1.1 Network architecture

We start form learning the conditional distribution of an image given a category or class label. The category information is encoded as a one-hot vector. The network architectures of the models in this experiment are given as follows.

In the initializer, we can concatenate the one-hot vector C with the latent noise vector X sampled from $\mathrm{N}(0, I_d)$ as the input of the decoder $\Psi([X, C])$ to build a conditional generator $g(X, C;\alpha)$. The generator maps the input into image Y by several layers of deconvolutions. We call this setting "early concatenation". See Figure 3(1) for an illustration. We can also adopt an architecture with "late concatenation", where the concatenation happens in the intermediate layer of the initializer. Specifically, we can first sample the latent noise vector X from Gaussian noise prior $\mathrm{N}(0, I_d)$, and then decode X to an intermediate result with spatial dimension $b\times b$ by a decoder $\Psi_1(X)$. The decoder consists of several layers of deconvolutions, each of which is followed by batch normalization [37] and ReLU non-linear transformation. We then replicate the one-hot vector C spatially and perform a channel concatenation with the intermediate output. After that, we generate the target image Y from the concatenated result $[\Psi_1(X), C]$ by another decoder $\Psi_2([\Psi_1(X), C])$ that consists of several deconvolution layers. Batch normalization and ReLU layer are used between two consecutive deconvolution layers, and tanh non-linearity is added at the bottom layer. $g(X, C;\alpha)$ is the composition of Ψ_1 and Ψ_2. See Figure 3(2) for an illustration. The details of the networks will be mentioned in the section of each experiment.

To build the value function for the solver model, in the setting of "early concatenation", we first replicate the condition one-hot vector C spatially and perform a depth concatenation with image Y, and then map them to a scalar by an encoder, $\Phi([Y, C])$, that consists of several layers of convolutions and ReLU non-linear transformations. The value function $f(Y, C;\theta)$ is designed as $\Phi([Y, C]) - \|Y\|^2/2s^2$. This corresponds to an exponential tilting form in [20],

$$p(Y, C;\theta) = \frac{1}{Z(\theta)}\exp\left[\Phi(Y, C;\theta)\right]p_0(Y), \tag{19}$$

where $p_0(Y)$ is Gaussian white noise distribution, i.e., $p_0(Y) \propto \exp(-\|Y\|^2/2s^2)$, and s is a hyperparameter for the standard

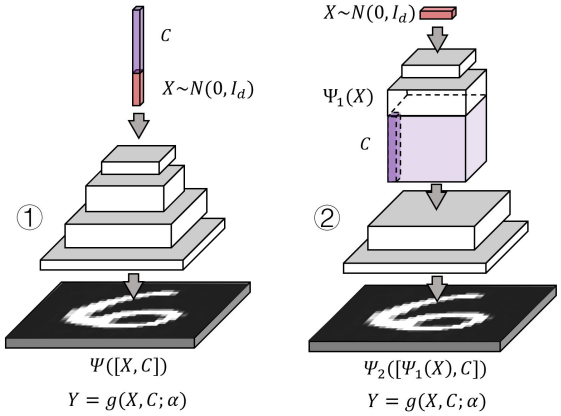


Fig. 3. Network architecture of initializer (category-to-image synthesis). (1) early concatenation: a decoder Ψ takes as input the concatenation of the condition vector C and the latent noise vector $X \sim \mathrm{N}(0, I_d)$, and outputs an image Y. (2) late concatenation: a decoder takes as input only the latent noise vector $X \sim \mathrm{N}(0, I_d)$, and outputs an image Y, in which the condition C is concatenated with the output of an intermediate layer. Ψ_2 is the sub-network after concatenation, while Ψ_1 is the sub-network before concatenation.

deviation of p_0. See Figure 4(1) for an illustration. As to the "late concatenation", we first encode the image Y to an intermediate result with spatial dimension $a \times a$ by an encoder $\Phi_1(Y)$, which consists of several layers of convolutions and ReLU non-linear transformations, and then we replicate the one-hot vector C spatially and perform a depth concatenation with the intermediate result. The value function is defined by another encoder $\Phi_2([\Phi_1(Y), C])$ plus $-\|Y\|^2/2s^2$, in which the encoder takes as input the concatenated result $[\Phi_1(Y), C]$ and outputs a scalar by performing several layers of convolutions and ReLU non-linear transformations. See Figure 4(2) for an illustration. Detailed network configuration will be discussed in the section of each experiment.

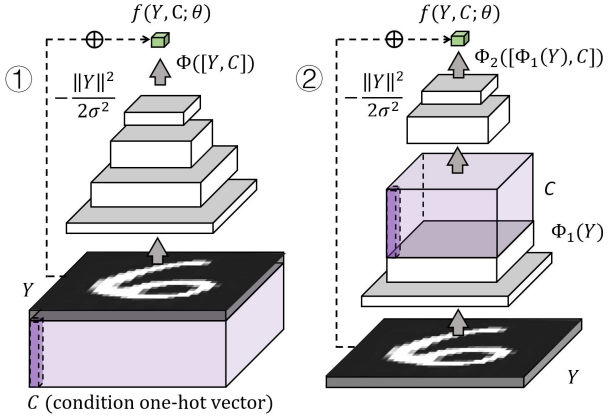


Fig. 4. Network architecture of solver (category-to-image synthesis). (1) early concatenation: an encoder Φ takes as input the depth concatenation of the spatially replicated condition vector C and the image Y, and outputs a scalar. The value function $f(Y, C; \theta)$ is defined as $\Phi([Y, C]) - \|Y\|^2/2s^2$. (2) late concatenation: an encoder takes as input only the image Y, and outputs the negative energy, in which the condition C is concatenated with the output of an intermediate layer. Φ_2 is the sub-network after concatenation, while Φ_1 is the sub-network before concatenation.

4.1.2 Conditional image generation on grayscale images

We first test our model on two grayscale image datasets, such as MNIST [38] and fashion-MNIST [39]. The former is a dataset of handwritten digit images, and the latter is a dataset of fashion product images. Each of them consists of 70,000 28×28 images, each of which is associated with a label from 10 classes. In each dataset, 60,000 examples are used for training and the rest are for testing. We learn our model on each of them respectively, conditioned on their class labels that are encoded as one-hot vectors. Since these two datasets are similar in number of classes, image size, data size, and image format (i.e., grayscale), we use the same model for them.

We adopt the setting of "early concantenation" introduced in section 4.1.1 for the initializer. To be specific, $g(X, C; \alpha)$ is a generator that maps the $1 \times 1 \times 138$ concatenated result (Note that the dimension of X is 128, and the size of C is 10.) to a 28×28 grayscale image by 4 layers of deconvolutions with kernel sizes $\{4, 4, 4, 4\}$, up-sampling factors $\{1, 2, 2, 2\}$ and numbers of output channels $\{256, 128, 64, 1\}$ at different layers. The last deconvolution layer is followed by a tanh operation, and each of the others is followed by batch normalization and ReLU operation.

We adopt the setting of "late concatenation" introduced in section 4.1.1 for the solver. Specifically, $\Phi_1(Y)$ consists of 2 layers of convolutions with filter sizes $\{5, 3\}$, down-sampling factors $\{2, 2\}$ and numbers of output channels $\{64, 128\}$. The concatenated output is of size $7 \times 7 \times 138$. (Note that the number of the output channels of Φ_1 is 128, and the size of C is 10.) $\Phi_2([\Phi_1(Y), C])$ is a 2-layer ConvNet, where the first layer has 256 3×3 filters, and the last layer is a fully-connected layer with 100 filters.

We use Adam [40] to optimize the solver with initial learning rate 0.0008, $\beta_1 = 0.5$ and $\beta_2 = 0.999$, and the initializer with initial learning rate 0.0001, $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The mini-batch size is 300. The number of paralleled MCMC chains is 300. The number of Langevin dynamics steps is $l = 16$. The step size δ of Langevin dynamics is 0.0008. The standard deviation of the residual in the initializer is $\sigma = 0.3$, and the standard deviation s of the reference distribution p_0 in the solver is 0.016. We run 1,600 epochs to train the model, where we disable the noise term in Langevin dynamics after the first 100 epochs.

Figure 5 shows some of the generated samples conditioned on the class labels after training on the MNIST dataset. Each column is conditioned on one label and each row is a different generated sample. Figure 6 shows the results for the fashion-MNIST dataset. The qualitative results show that our method can learn realistic conditional models.

To quantitatively evaluate the learned conditional distribution, we use "Fréchet Inception Distance" [41] (FID) score as a metric to measure the dissimilarity between the distributions of the observed and the synthesized examples. Specifically, we compute the distance between feature vectors extracted from observed and synthesized examples by a pre-trained Inception model [42], with the following formula

$$\mathrm{FID} = ||\tilde{\mu} - \mu||^2 + \mathrm{Tr}\left(\tilde{\Sigma} + \Sigma - 2(\tilde{\Sigma}\Sigma)^{1/2}\right),$$

where $V \sim \mathrm{N}(\mu, \Sigma)$ and $\tilde{V} \sim \mathrm{N}(\tilde{\mu}, \tilde{\Sigma})$ are the 2,048-dimensional feature vectors of the observed and synthesized examples, respectively. They are the outputs taken as the activations from the global spatial pooling layer of the Inception model. We can fit a multi-variate Gaussian to feature vectors $\{V_i\}$ and $\{\tilde{V}_i\}$ separately,

Fig. 5. Generated MNIST handwritten digits. Each column is conditioned on one class label and each row is a different synthesized sample. The size of the generated images is 28×28.

Fig. 6. Generated fashion MNIST images. Each column is conditioned on one class label and each row is a different synthesized sample. The size of the generated images is 28×28.

to obtain means $\mu, \tilde{\mu}$ and variances $\Sigma, \tilde{\Sigma}$ for the observed and synthesized distributions respectively. A lower FID score implies better qualities of the synthesized images.

To compute FID score, we sample 10,000 examples from the learned conditional distribution by first sampling the class label C from the uniform prior distribution, and X from $\mathrm{N}(0, I_d)$, then the initializer and the solver model cooperatively generate the synthesized example from the sampled C and X. Table 1 shows a comparison of FID scores of different methods on two datasets. Our method achieves better results than other conditional and unconditional baseline methods in terms of generation quality evaluated by FID. Those baselines include GAN-based, flow-based, and variational inference methods.

Figure 7 displays some examples of the synthesized images at different training epochs along with the corresponding FID scores. The images shown are generated by the solver. The images at the same position of 5×5 image matrix of different training epochs share the same condition C, i.e., the class label. We can find that as the cooperative training progresses, the synthesized images become more and more realistic and the FID scores become lower and lower. Additionally, the learned connection between the condition (i.e., class label) and the target (i.e., image) becomes more and more accurate in the sense that when the model converges, even though the appearances of the synthesized images vary at different epochs, they are always consistent with their input conditions.

TABLE 1
The Fréchet Inception Distance (FID) scores of different models trained on MNIST and fashion-MNIST datasets, the smaller the FID, the better the performance.

	Model	MNIST	fashion-MNIST
unconditional	GLO [43]	49.60	57.70
	VAE [44]	21.85	69.84
	BEGAN [45]	13.54	15.90
	EBGAN [46]	11.10	41.32
	GLANN [47]	8.60	13.10
	WGAN [48]	7.07	28.17
	LSGAN [49]	6.75	14.72
	DCGAN [12]	4.54	8.22
	InfoGAN [50]	28.09	-
	GLF [51]	5.80	10.30
conditional	CGlow [52]	29.64	-
	CAGlow [52]	26.34	-
	VCGAN [53]	-	13.8
	CVAE-GAN [54]	-	15.9
	CVAE [55]	20.00	36.64
	ACGAN [56]	12.55	49.11
	CGAN [11]	5.91	11.92
	CCoopNets (ours)	**4.50**	**8.20**

We study the influences of different choices of some hyper-parameters, such as the number of dimension d of the latent space X in the initializer, the number of Langevin refinement steps l, and the step size δ of each Langevin. Figure 8 depicts the influences of varying d, l and δ, respectively, while training on fashion-MNIST dataset. Each curve represents the testing FID scores over training epochs. We observe that (1) the quality of synthesis decreases with decreasing d. (2) the more the number of MCMC refinement steps, the stabler the learning process, and the more time-consuming the refinement process of the solver. With a small l, e.g., 1 or 8, the cooperative learning tends to fail easily at the early stage of training because the slow-thinking solver distills an insufficient refinement process to the initializer such that the latter can not provide good initial solutions for the former. Figure 8(b) shows that the learning curves for $l = 1$ (in blue) and $l = 8$ (in orange) are terminated early due to failures occurred during training. Table 2 shows a comparison of computational time per epoch with different numbers of Langevin steps l and different numbers of latent dimensions d. A choice of $l = 16$ or 32 appears reasonable. The influence of d on the computational time is not significant. (3) A large Langevin step size allows the model to learn faster to generate high quality images, at the cost of arriving on a sub-optimal synthesis of images. A smaller Langevin step size may allow the model to generate more realistic images but it may take more Langevin steps.

4.1.3 Conditional image generation on Cifar-10

We also test the proposed framework on Cifar-10 [57] object dataset, which contains 10-class 60,000 training images of 32×32 pixels. Compared with the MNIST dataset, Cifar-10 contains training images with more complicated visual patterns.

As to the initializer, we adopt the "late concatenation" setting. Specifically, $\Psi_1(X)$ is a decoder that maps 100-dimensional X (i.e., $1 \times 1 \times 100$) to an intermediate output with spatial dimension 8×8 by 2 layers of deconvolutions with kernel sizes

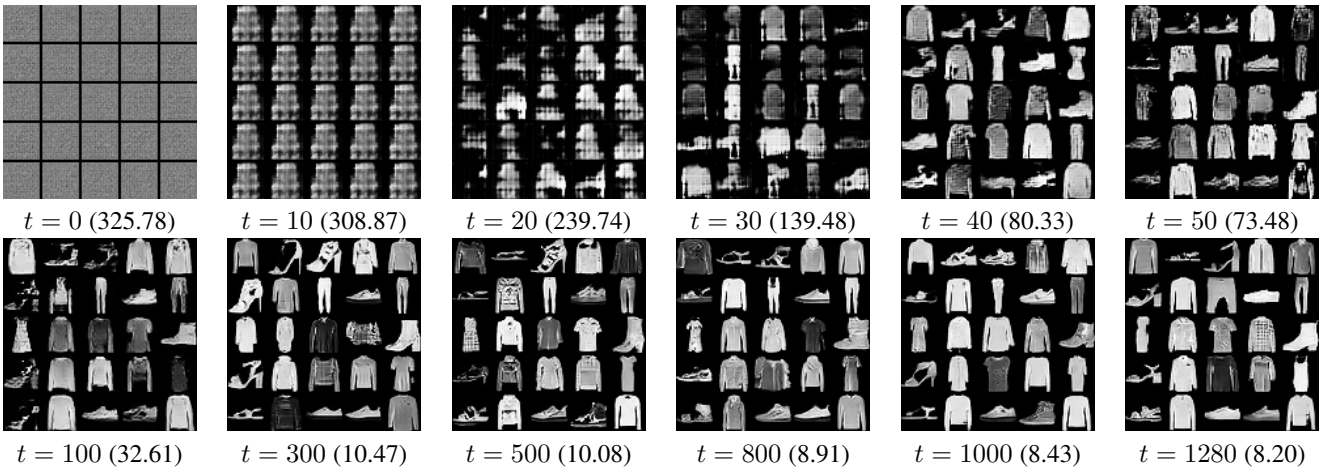
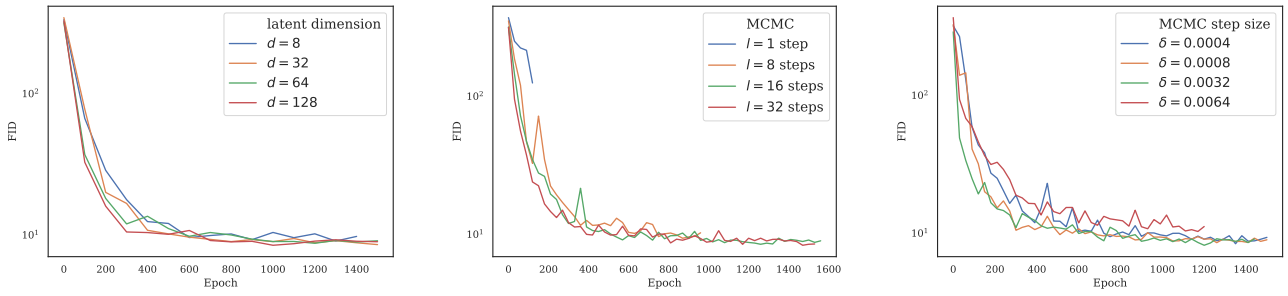


Fig. 7. Image generation by the models at different training epochs. For each epoch t, 25 examples of synthesized images are displayed. The numbers in parentheses are the corresponding FID scores that reflect the qualities of the synthesized images. The images at the same position of image matrix of different training epochs are generated from the same condition.



(a) number of latent dimension (b) number of Langevin steps (c) step size of Langevin refinement

Fig. 8. Model analysis on fashion-MNIST dataset. (a) Influence of the number of latent dimension d of the fast-thinking initializer. We set $l = 16$ and $\delta = 0.0008$. (b) Influence of the number l of Langevin refinement steps by the slow-thinking solver. We set $d = 64$ and $\delta = 0.0008$. (c) Influence of the step size δ of Langevin refinement of the slow-thinking solver. We set $d = 128$ and $l = 16$.

TABLE 2
Comparison of computational time (in seconds) per epoch with different numbers of Langevin refinement steps and different numbers of latent dimensions for class-conditioned image generation on fashion-MNIST dataset. The running times were recorded in a PC with an Intel i7-6700k CPU and a Titan Xp GPU.

	$l = 1$	$l = 8$	$l = 16$	$l = 32$	$l = 64$
$d = 8$	8.98	20.38	26.88	46.74	86.93
$d = 32$	9.23	20.21	27.04	46.95	86.95
$d = 64$	9.12	20.10	27.55	47.22	87.06
$d = 128$	9.37	20.50	27.76	48.62	86.92

$\{4, 5\}$, up-sampling factors $\{1, 2\}$ and numbers of output channels $\{256, 128\}$ at different layers from top to bottom, respectively. The condition C is a 10-dimensional one-hot vector to represent the class. $\Psi_2([\Psi_1(X), C])$ is a generator that maps the $8 \times 8 \times 138$ concatenated result to a $32 \times 32 \times 3$ image by 2 layers of deconvolutions with kernel sizes $\{5, 5\}$, up-sampling factors $\{2, 2\}$ and numbers of output channels $\{64, 3\}$ at different layers.

We adopt the "late concatenation" setting for the solver. $\Phi_1(Y)$ consists of 2 layers of convolutions with filter sizes $\{5, 3\}$, down-sampling factors $\{2, 2\}$ and numbers of output channels $\{64, 128\}$. The concatenated output is of size $8 \times 8 \times 138$. $\Phi_2([\Phi_1(Y), C])$ is a 2-layer bottom-up ConvNet, where the first layer has 256 3×3 filters, and the last layer is a fully connected layer with 100 filters.

We use the Adam for optimization. The initial learning rates for the solver and initializer are 0.002 and 0.0064, respectively. The joint models are trained with mini-batches of size 300. The number of paralleled MCMC chains is also 300. The number of Langevin dynamics steps is 8. The step size δ of Langevin dynamics is 0.0008. We run 2,000 epochs to train the model, where we disable the noise term in Langevin dynamics in the last 1,500 ones.

Figure 9 shows the generated object patterns. Each row is conditioned on one category. The first two columns display some typical training examples, while the rest columns show generated images conditioned on labels. We evaluate the learned conditional distribution by computing the inception scores of the generated examples. Table 3 compares our framework against two baselines, which are two conditional models based on GANs. The proposed model performs better than the baselines. We also found that in the proposed method, the solution provided by the initializer is indeed further refined by the solver in terms of inception score.

4.1.4 *Disentangling style and category*

To test the inference power of the fast-thinking initializer, which is trained jointly with the slow-thinking solver, we apply the learned initializer to a task of style transfer from an unseen testing image in one caegory onto other categories. The models are first trained on SVHN [64] dataset that contains 10 classes of digits collected from street view house numbers. The network architectures of initializer

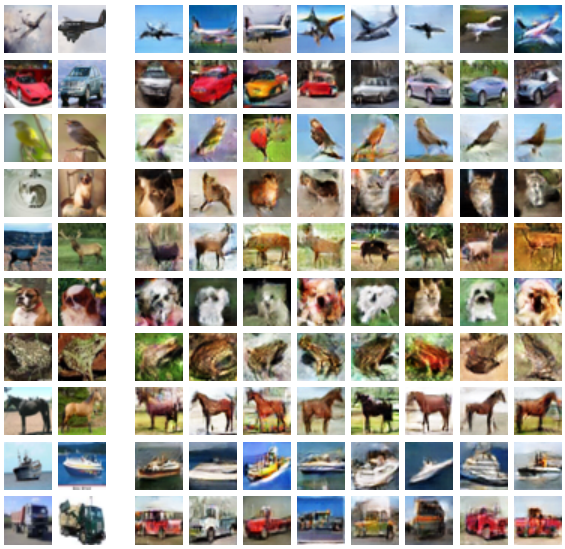


Fig. 9. Generated Cifar-10 object images. Each row is conditioned on one category label. The first two columns are training images, and the remaining columns display generated images conditioned on their labels. The image size is 32×32 pixels. The categories are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck from top to bottom.

TABLE 3
Inception scores of different models trained on Cifar-10 dataset. The larger the inception score, the better the performance.

	Model	Inception score
unconditional	PixelCNN [58]	4.60
	PixelIQN [59]	5.29
	DCGAN [12]	6.40
	WGAN-GP [60]	6.50
	ALI [61]	5.34
conditional	CGAN [62]	6.58
	Conditional SteinGAN [63]	6.35
	initializer (ours)	6.63
	solver (ours)	**7.30**

and solver are similar to those used in Section 4.1.2, except that the training images in this experiment are RGB images and they are of size 32×32 pixels. With the learned initializer, we first infer the latent variables X corresponding to that testing image. We then fix the inferred latent vector, change the category label C, and generate the different categories of images with the same style as the testing image by the learned model. Given a testing image Y with known category label C, the inference of the latent vector X can be performed by directly sampling from the posterior distribution $p(X|Y, C; \alpha)$ via Langevin dynamics, which iterates

$$X_{\tau+1} = X_\tau + sU_\tau + \frac{s^2}{2}\left[\frac{1}{\sigma^2}(Y - g(X_\tau, C; \alpha))\frac{\partial}{\partial X}g(X_\tau, C; \alpha) - X_\tau\right]. \quad (20)$$

If the category label of the testing image is unknown, we need to infer both C and X from Y. Since C is a one-hot vector, in order to adopt a gradient-based method to infer C, we adopt a continuous approximation by reparametrizing C using a softMax transformation on the auxiliary continuous variables A. Specifically, let $C = (c_k, k = 1, ..., K)$ and $A = (a_k, k = 1, ..., K)$, we reparametrize $C = v(A)$ where $c_k = \exp(a_k)/\sum_k' \exp(a_k')$, for $k = 1, ..., K$, and assume the prior for A to be $\mathrm{N}(0, I_K)$. Then the Langevin dynamics for sampling $A \sim p(A|Y, X)$ iterates

$$A_{\tau+1} = A_\tau + sU_\tau + \frac{s^2}{2}\left[\frac{1}{\sigma^2}(Y - g(X_\tau, v(A); \alpha))\frac{\partial}{\partial A}g(X, v(A_\tau); \alpha) - A\right]. \quad (21)$$

Figure 10 shows 10 results of style transfer. For each testing image Y, we infer X and C by sampling $[X, C] \sim p(X, C|Y)$, which iterates (1) $X \sim p(X|Y, C)$, and (2) $C = v(A)$ where $A \sim p(A|Y, X)$, with randomly initialized X and C. We then fix the inferred latent vector X, change the category label C, and generate images from the combination of C and X by the learned initializer. This experiment demonstrates the effectiveness of our model in style and category disentanglement.

Fig. 10. Style transfer. The trained initializer can disentangle the style and the category such that the style information can be inferred from a testing image and transferred to other categories. The first column shows testing images. The other columns show style transfer by the model, where the style latent variable X of each row is set to the value inferred from the testing image in the first column by the Langevin inference. Each column corresponds to a different category label C.

4.2 Experiment 2: Image → Image

4.2.1 Network architecture

We study learning conditional distributions for image-to-image translation by our framework. The network architectures of the models in this experiment are discussed as follows.

As to the initializer, a straightforward design is presented below: we first sample X from the Gaussian noise prior $\mathrm{N}(0, I_d)$, and we encode the condition image C via an encoder $\Phi(C)$. The image embedding $\Phi(C)$ is then concatenated to the latent noise vector X. After this, we generate target image Y by a decoder $\Psi([X, \Phi(C)])$. The initializer $g(X, C; \alpha)$ is the composition of Φ and Ψ. With Gaussian noise X, the initializer will produce stochastic outputs as a distribution. See Figure 11(1) for an illustration of the structure. However, in the initial experiments, we found that this design was ineffective in the sense that the generator learned to ignore the noise and produce deterministic outputs. Inspired by [10], we design the initializer by following a general shape of the U-Net [65] with the form of dropout [66], applied on several layers,

as noise that accounts for stochasticity in this experiment. A U-Net is an encoder-decoder structure with skip connections added between each layer j and layer $M-j$, where M is the number of layers. Each skip connection performs a concatenation between all channels at layer j and those at layer $M-j$. In the task of image-to-image translation, the input and output images usually differ in appearance but share low-level information. For example, in the case of translating sketch image to photo image, the input and output images are roughly aligned in outline except that they have different colors and textures in appearance. The addition of skip connections allow a direct transfer of low-level information across the network. Figure 11(2) illustrates the U-Net structure with dropout as the initializer for image-to-image translation.

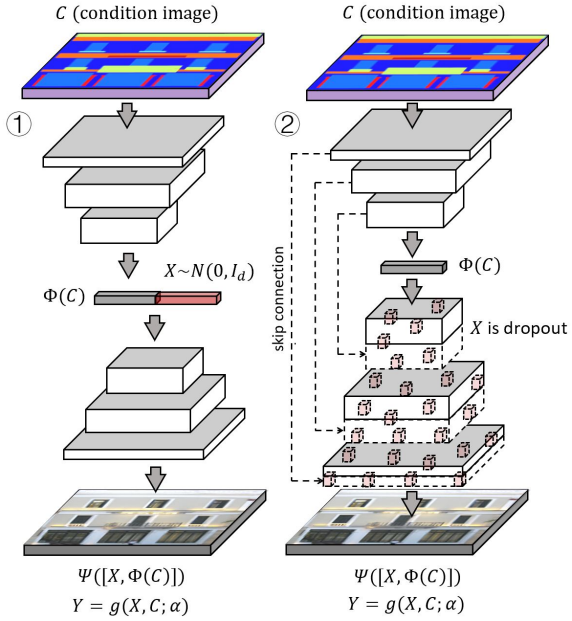


Fig. 11. Network architecture of initializer (image-to-image translation). (1) naive straightforward design: the condition image C is first encoded to a vector representation by an encoder $\Phi(C)$, and then the vector is concatenated with the Gaussian noise vector X. A decoder Ψ takes as input the concatenated vector $[X, \Phi(C)]$ and outputs the target image Y. (2) U-Net with dropout: an encoder-decoder structure (Φ is the encoder and Ψ is the decoder.), with skip connections added between each layer j and layer $M-j$, where M is the number of layers. Each skip connection concatenates all channels at layer j and those at layer $M-j$. The dropout is applied to each layer in the decoder Ψ to account for randomness X.

As to the design of the solver model, we first perform channel concatenation on target image Y and condition image C, where both images are of the same size. The value function $f(Y, C, \theta)$ is then defined by an encoder $\Phi([Y, C])$ plus $-\|Y\|^2/2s^2$, in which $\Phi([Y, C])$ maps the 6-channel "image" to a scalar by several convolutional layers. Leaky ReLU layers are added between two consecutive convolutional layers. Figure 12 shows an illustration of the network architecture of the solver.

4.2.2 *Semantic labels → Scene images*

The experiments are conducted on CMP Facade dataset [67] where each building facade image is associated with an image of architectural labels. The condition image and the target image are of the size of 256×256 pixels with RGB channels. Data are randomly split into training and testing sets.

In the initializer, the encoder Φ consists of 8 layers of convolutions with a filter size 4, a subsampling factor 2, and

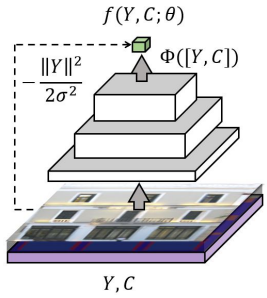


Fig. 12. Network architecture of solver (image-to-image translation). Channel concatenation is performed on the condition image C and the target image Y. The resulting 6-channel "image" is then fed into an encoder $\Phi([Y, C])$. Φ plus $-\|Y\|^2/2s^2$ serves as the value function $f(Y, C; \theta)$ in the slow-thinking solver model.

the numbers of channels $\{64, 128, 256, 512, 512, 512, 512, 512\}$ at different layers. Batch normalization and leaky ReLU (with slope 0.2) layers are used after each convolutional layer except that batch normalization is not applied after the first layer. The output of Φ is then fed into Ψ, which consists of 8 layers of deconvolutions with a kernel size 4, an up-sampling factor 2, and the numbers of channels $\{512, 512, 512, 512, 256, 128, 64, 3\}$ at different layers. Batch normalization, dropout with a dropout rate of 0.5, and ReLU layer are added between two consecutive deconvolutional layers, and a tanh non-linearity is used after the last layer. The U-Net structure used in this experiment is a connection of the encoder Φ and the decoder Ψ, along with skip connections added to concatenate activations of each layer j and layer $M-j$. (M is the total number of layers.) Therefore, the numbers of output channels of Ψ in the U-Net are $\{1024, 1024, 1024, 1024, 512, 256, 128, 3\}$. The dropout that is applied to each layer of Ψ implies an implicit latent vector X in the initializer. Such an implicit X is too complicated to infer. However, there is no need to infer this X with the cooperative training, which can get around the difficulty of the inference of any complicated forms of latent factors by MCMC teaching. It other words, in each iteration, the learning of the initializer $\Psi([X, \Phi(C)])$ is based on how the MCMC changes the initial examples generated by the initializer from the condition image C and the randomness X due to dropout.

In the solver model, we first perform a channel concatenation on target image Y and condition image C, where both images are of size $256 \times 256 \times 3$. The value function is then defined by a 4-layer encoder $\Phi([Y, C])$, which maps a 6-channel "image" to a scalar as the value score by 3 convolutional layers with numbers of channels $\{64, 128, 256\}$, filter sizes $\{5, 3, 3\}$ and subsampling factors $\{2, 2, 1\}$ at different layers (from bottom to top), and one fully connected layer with 100 single filers. Leaky ReLU layer is used between two consecutive convolutional layers.

Adam is used to optimize the solver with an initial learning rate 0.007, and the initializer with an initial learning rate 0.0001. We set the mini-batch size to be 1. The number of paralleled MCMC chains is also 1. We run 15 Langevin steps with a step size $\delta = 0.002$. The standard deviation of the residual in the initializer is $\sigma = 0.3$. The standard deviation of the reference distribution in the solver is $s = 0.016$. We run 3,000 epochs to train our model.

We adopt random jitter and mirroring for data augmentation in the training stage. As to random jitter, we first resize the input images from 256×256 to 286×286, and then randomly crop image patches with a size 256×256.

In this task, we found it beneficial to feed both the refined solutions and the observed ground truth solutions to the initializer, when we update the initializer at each iteration. The solver's job remains unchanged, but the initializer is tasked to not only learn from the solver $\{\tilde{Y}_i\}$ but also to be near the ground truth solutions $\{Y_i\}$. We add an extra $\ell 1$ loss to penalize the distance between the output of the initialzer and the ground truth solution. [10] also finds this strategy effective in training a GAN-based conditional model for image-to-image translation.

As to the computational time, compared with GAN-based method, our framework has additional $l = 15$ steps of Langevin. However, the Langevin is based on gradient, whose computation can be powered by back-propagation, so it is not significantly time-consuming. To be concrete, our method costs 32.7s, while GAN-based method costs 30.9s per epoch for training in a PC with an Intel i7-6700k CPU and a Titan Xp GPU in this experiment.

Figure 13 shows some qualitative results of generating building facade images from the semantic labels. The first column displays 5 semantic label images that are unseen in the training data. The second column displays the corresponding ground truth images for reference. The results by a baseline method, pix2pix [10], are shown in the third row for comparison. pix2pix is a conditional GAN method for image-to-image mapping. Since its generator also uses a U-Net and is paired up with a $\ell 1$ loss, for a fair comparison, our initializer adopts exactly the same U-Net structure as in [10]. The fourth to sixth columns are results generated by some variants of the conditional GAN method, including cVAE-GAN [71], cVAE-GAN++ [71] and BicycleGAN [71]. The seventh and eighth rows show the generated results conditioned on the semantic label images shown in the first row by the learned initializer and solver, respectively. We can easily observe qualitative improvements by comparing the outputs of the solver with the ones of the initializer.

We perform human perceptual tests for evaluating the visual quality of synthesized images. We randomly select 30 different human users to participate in these tests. We compare two methods in each test, where each participant is first presented two images at a time, which are results generated by two different methods given the same conditional input, and then asked which one looks more like a real image. We have total 36 pairwise comparisons in each test for each participant. We evaluate each method by the ratio that the images generated by the method are preferred. As shown in Table 4, the results generated by our method are considered more realistic by the human subjects.

TABLE 4
Human perceptual tests for image-to-image synthesis.

methods	preference ratio
CCoopNets (ours) / cVAE-GAN [71]	**0.625** / 0.375
CCoopNets (ours) / cVAE-GAN++ [71]	**0.687** / 0.313
CCoopNets (ours) / BicycleGAN [71]	**0.628** / 0.372
CCoopNets (ours) / pix2pixel [10]	**0.720** / 0.280

4.2.3 *Sketch images → Photo images*

We next test the model on CUHK Face Sketch database (CUFS) [68], where for each face, there is a sketch image drawn by an artist based on a photo of the face. We learn to recover the color face images from the sketch images by the proposed framework. The network design and hyperparameter setting are similar to the one we used in Section 4.2.2, except that the mini-batch size and the number of paralleled MCMC chains are set to be 4.

Figure 14(a) displays the face synthesis results conditioned on the sketch images. Columns 1 through 4 show some sketch images as input conditions, while columns 5 through 8 show the corresponding recovered images obtained by sampling from the learned conditional distribution. From the results, we can see that the generated facial appearance (color and texture) in each output image is not only reasonable but also consistent with the input sketch face image in the sense that the face identity in each sketch image remains unchanged after being translating to a photo image.

Figure 14(b) demonstrates the learned manifold of sketch images (condition) by showing 5 examples of interpolation. For each row, the sketch images at the two ends are first encoded into the embedding by $\Phi(C)$, and then each face image in the middle is obtained by first interpolating the sketch embedding, and then generating the images using the initializer with a fixed dropout, and eventually refining the results by the solver via finite-step Langevin dynamics. Even though there is no ground truth sketch images for the intervening points, the generated faces appear plausible. Since the dropout X is fixed, the only changing factor is the sketch embedding. We observe smooth changing of the generated faces.

We conduct another experiment on UT Zappos50K dataset [67] for photo image recovery from edge image. The dataset contains $50k$ training images of shoes. Edge images are computed by HED edge detector [69] with post processing. We use the same model structure as the one in the last experiment. Figure 15 shows some qualitative results of synthesizing shoe images from edge images.

4.2.4 *Image inpainting*

We also test our method on the task of image inpainting by learning a mapping from an occluded image (256×256 pixels), where a mask with the size of 128×128 pixels is centrally placed onto the original version, to the original image. We use Paris streetview [70] and the CMP Facade dataset. In this case, C is the observed part of the input image, and Y is the unobserved part of the image. The network architectures for both initializer and solver, along with hyperparameter setting, are similar to those we used in Section 4.2.2. To recover the occluded part of the input images, we only update the pixels of the occluded region in the Langevin dynamics.

We compare our method with some baselines, including pix2pix, cVAE-GAN, cVAE-GAN++ and BicycleGAN. Table 5 shows quantitative results where the recovery performance is measured by the peak signal-to-noise ratio (PSNR) and structural similarity measures (SSIM), which are computed between the occlusion regions of the generated example and the ground truth example. The batch size is one. Our method outperforms the baseline methods using adversarial training or variational inference in this recovery task. Table 6 reports a comparison of model complexity with the baseline methods on CMP Facade dataset in terms of number of model parameters and running time.

Figure 16 shows a comparison of qualitative results of different methods on CMP Facade dataset. Each row displays one example. The first image is the testing image with a hole that needs to be recovered. The second image is the ground truth image. The third to sixth images are the inpainting results obtained by pix2pix, cVAE-GAN, cVAE-GAN++ and BicycleGAN, respectively. The seventh and the last images are the results recovered by the initializer and the solver, respectively.

5 CONCLUSION

Solving a challenging problem usually requires an iterative algorithm. This amounts to slow thinking. The iterative algorithm

Fig. 13. Generating images conditioned on architectural labels. The first column displays 5 condition images with architectural labels. The second column displays the corresponding ground truth images for reference. For comparison, the third to sixth columns show the generated results by baselines pix2pix, cVAE-GAN, cVAE-GAN++, and BicycleGAN, respectively. The seventh and eighth columns present the generated results obtained by the learned initializer and solver respectively. The training images are of the size 256×256 pixels.

TABLE 5
Comparison with the baseline methods for image inpainting on the CMP Facade dataset and Paris streetview dataset.

Model	CMP Facades		Paris streetview	
	PSNR	SSIM	PSNR	SSIM
cVAE-GAN [71]	19.43	0.68	16.12	0.72
cVAE-GAN++ [71]	19.14	0.64	16.03	0.69
BicycleGAN [71]	19.07	0.64	16.00	0.68
pix2pix [10]	19.34	0.74	15.17	0.75
CCoopNets (ours)	**20.47**	**0.77**	**21.17**	**0.79**

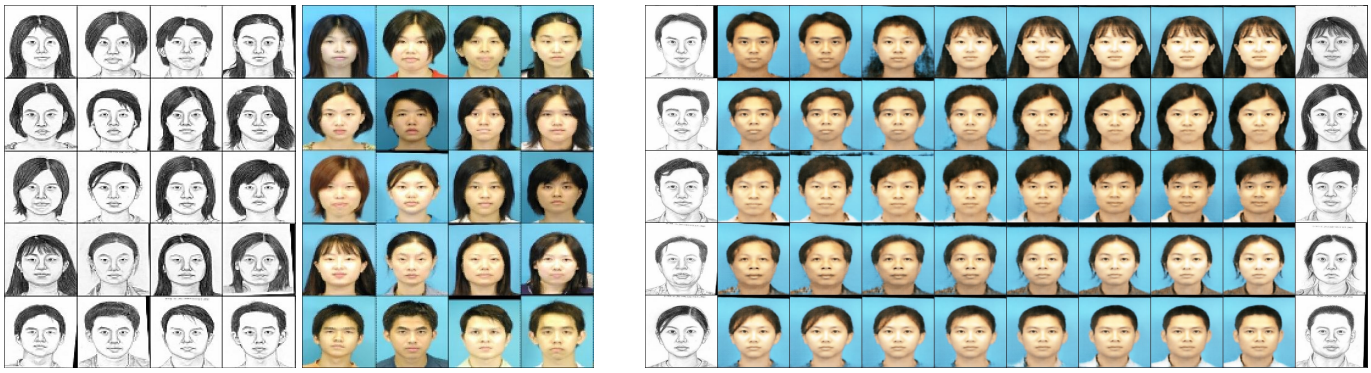
TABLE 6
Comparison of model complexity with the baseline methods for image inpainting on CMP Facade dataset.

Model	Size	Time
	♯ of parameters	sec / epoch
cVAE-GAN [71]	60.85M	12.06
cVAE-GAN++ [71]	64.30M	18.40
BicycleGAN [71]	64.30M	25.60
pix2pix [10]	57.89M	12.62
CCoopNets (ours)	55.84M	22.43

usually requires a good initialization to jumpstart it so that it can converge quickly. The initialization amounts to fast thinking. For instance, reasoning and planning usually require iterative search or optimization, which can be initialized by a learned computation in the form of a neural network. Thus integrating fast thinking initialization and slow thinking sampling or optimization is very compelling. This paper addresses the problem of high-dimensional conditional learning and proposes a cooperative learning method that couples a fast thinking initializer and a slow thinking solver. The initializer initializes the iterative optimization or sampling process of the solver, while the solver in return teaches the initializer by distilling its iterative algorithm into the initializer. We demonstrate the proposed method on a variety of image synthesis and recovery tasks. Compared to GAN-based method, such as conditional GANs, our method is equipped with an extra iterative sampling and optimization algorithm to refine the solution, guided by a learned objective function. This may prove to be a powerful method for solving challenging conditional learning problems.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. The work is supported by NSF DMS-2015577, DARPA SIMPLEX N66001-15-C-4035, ONR MURI N00014-16-1-2007, DARPA ARO W911NF-16-1-0579, DARPA N66001-17-2-4029, and XSEDE grant ASC180018.



(a) Recover faces from sketches (b) Sketch interpolation

Fig. 14. (a) Sketch-to-photo face synthesis. Columns 1 through 4: sketch images as conditions. Columns 5 through 8: corresponding face images sampled from the learned models conditioned on sketch images. (b) Sketch interpolation: Generated face images by interpolating between the embedding of the sketch images at two ends, with fixed dropout. Each row displays one example of interpolation.

Fig. 15. Results on edges $\rightarrow$ shoes generation, compared to ground truth. The first row displays the edge images. The second row shows the corresponding ground truth photo images. The last two rows present the generated results obtained by the initializer and the solver, respectively.

REFERENCES

[1] R. M. Neal, "MCMC using Hamiltonian dynamics," *Handbook of Markov Chain Monte Carlo*, vol. 2, 2011.

[2] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer, 2008.

[3] A. Barbu and S.-C. Zhu, *Monte Carlo Methods*. Springer, 2020.

[4] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2004, pp. 1–8.

[5] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, 2008, pp. 1433–1438.

[6] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4565–4573.

[7] J. Xie, Y. Lu, R. Gao, and Y. N. Wu, "Cooperative learning of energy-based model and latent variable model via MCMC teaching," in *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018, pp. 4292–4301.

[8] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu, "Cooperative training of descriptor and generator networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 42, no. 1, pp. 27–45, 2018.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5967–5976.

[11] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[12] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[13] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *International Conference on Machine Learning (ICML)*, vol. 48, 2016, pp. 1060–1069.

[14] E. L. Denton, S. Chintala, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 1486–1494.

[15] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5907–5915.

[16] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8798–8807.

[17] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4681–4690.

[18] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, N. Yakovenko, A. Tao, J. Kautz, and B. Catanzaro, "Video-to-video synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 1152–1164.

[19] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *International Conference on Machine Learning (ICML)*, 2001, pp. 282–289.

[20] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu, "A theory of generative ConvNet," in *International Conference on Machine Learning (ICML)*, 2016, pp. 2635–2644.

[21] R. Gao, Y. Lu, J. Zhou, S.-C. Zhu, and Y. Nian Wu, "Learning generative ConvNets via multi-grid modeling and sampling," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9155–9164.

[22] J. Xie, S.-C. Zhu, and Y. N. Wu, "Synthesizing dynamic patterns by spatial-temporal generative convnet," in *CVPR*, 2017.

[23] J. Xie, S.-C. Zhu, and Y.-N. Wu, "Learning energy-based spatial-temporal generative ConvNets for dynamic patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019.

[24] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Learning descriptor networks for 3D shape synthesis and analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 8629–8638.

[25] E. Nijkamp, M. Hill, S.-C. Zhu, and Y. N. Wu, "Learning non-convergent non-persistent short-run mcmc toward energy-based model," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 5233–5243.

[26] E. Nijkamp, M. Hill, T. Han, S.-C. Zhu, and Y. N. Wu, "On the anatomy of MCMC-based maximum likelihood learning of energy-based models," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 5272–5280.

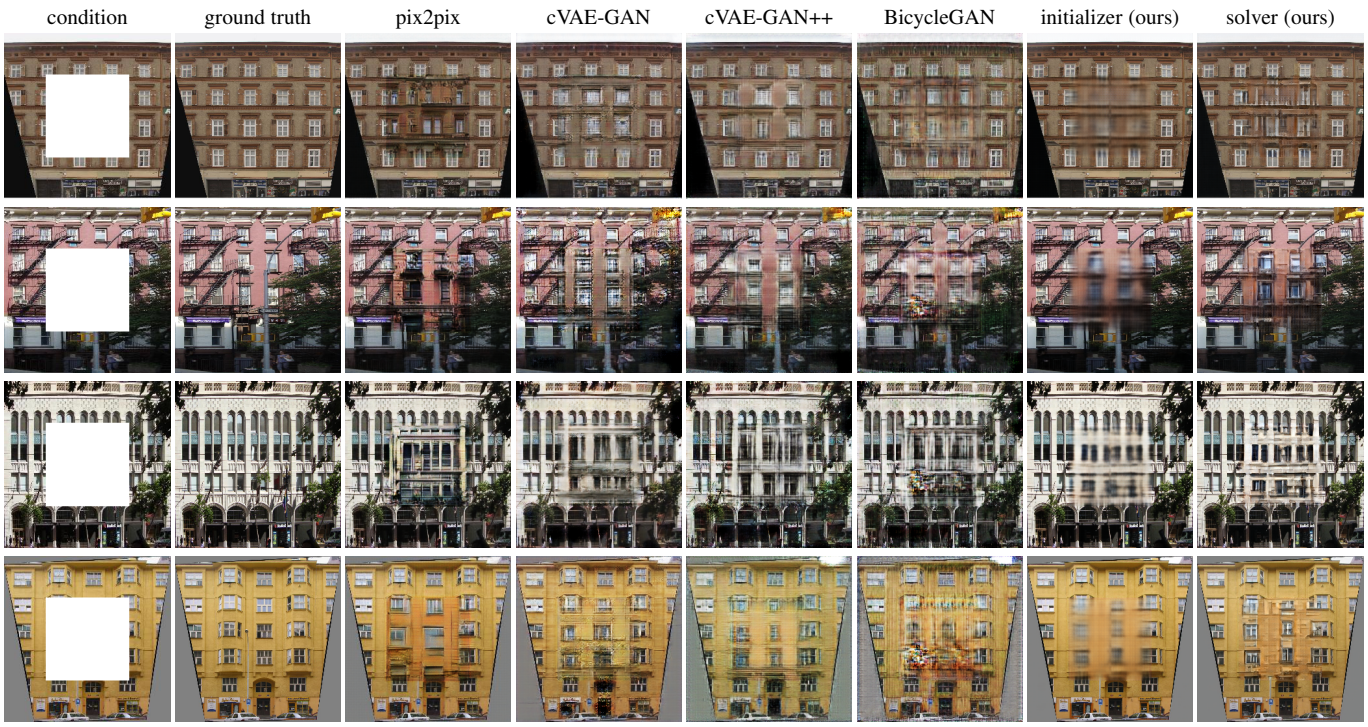


Fig. 16. Results of photo inpainting. Each row displays one example. The first image is the testing image (256×256 pixels) with a hole of 128×128 pixels that needs to be recovered, the second image is the ground truth, the third to sixth images are the results recovered by pix2pix, cVAE-GAN, cVAE-GAN++, BicycleGAN for comparison. The seventh and the last images are the results recovered by the initializer and the solver, respectively.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[28] J. Xie, Z. Zheng, R. Gao, W. Wang, S.-C. Zhu, and Y. N. Wu, "Generative voxelnet: Learning energy-based models for 3D shape synthesis and analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[29] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2223–2232.

[30] A. Grover, C. Chute, R. Shu, Z. Cao, and S. Ermon, "Alignflow: Cycle consistent learning from multiple domains via normalizing flows," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020, pp. 4028–4035.

[31] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," *arXiv preprint arXiv:1410.8516*, 2014.

[32] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations (ICLR)*, 2017.

[33] J. Xie, Z. Zheng, X. Fang, S.-C. Zhu, and Y. N. Wu, "Learning cycle-consistent cooperative networks via alternating MCMC teaching for unsupervised cross-domain translation," in *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[34] L. Younes, "On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates," *Stochastics: An International Journal of Probability and Stochastic Processes*, vol. 65, no. 3-4, pp. 177–228, 1999.

[35] T. Han, Y. Lu, S.-C. Zhu, and Y. N. Wu, "Alternating back-propagation for generator network," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017, pp. 1976–1984.

[36] G. E. Hinton, "Training products of experts by minimizing contrastive divergence." *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

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

[39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[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 *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 6626–6637.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[43] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 599–608.

[44] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[45] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[46] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," in *International Conference on Learning Representations (ICLR)*, 2017.

[47] Y. Hoshen, K. Li, and J. Malik, "Non-adversarial image synthesis with generative latent nearest neighbors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5811–5819.

[48] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 214–223.

[49] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2794–2802.

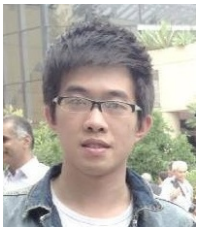
[50] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 2172–2180.

[51] Z. Xiao, Q. Yan, and Y. Amit, "Generative latent flow," *arXiv preprint arXiv:1905.10485*, 2019.

[52] R. Liu, Y. Liu, X. Gong, X. Wang, and H. Li, "Conditional adversarial generative flow for controllable image synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7992–8001.

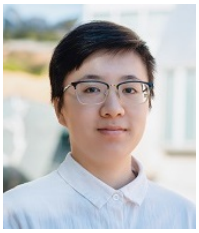
[53] M. Hu, D. Zhou, and Y. He, "Variational conditional GAN for fine-grained

controllable image generation," in *Asian Conference on Machine Learning (ACML)*, 2019, pp. 109–124.
[54] J. Bao, D. Chen, F. Wen, H. Li, and G. Hua, "CVAE-GAN: fine-grained image generation through asymmetric training," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2745–2754.
[55] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3483–3491.
[56] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *International Conference on Machine Learning (ICML)*, 2017, pp. 2642–2651.
[57] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
[58] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2016, pp. 1747–1756.
[59] G. Ostrovski, W. Dabney, and R. Munos, "Autoregressive quantile networks for generative modeling," in *International Conference on Machine Learning (ICML)*, 2018, pp. 3933–3942.
[60] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5767–5777.
[61] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. C. Courville, "Adversarially learned inference," in *International Conference on Learning Representations (ICLR)*, 2017.
[62] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 2234–2242.
[63] D. Wang and Q. Liu, "Learning to draw samples: With application to amortized MLE for generative adversarial learning," *arXiv preprint arXiv:1611.01722*, 2016.
[64] 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.
[65] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015, pp. 234–241.
[66] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014.
[67] R. Tyleček and R. Šára, "Spatial pattern templates for recognition of objects with regular structure," in *German Conference on Pattern Recognition (GCPR)*, 2013, pp. 364–374.
[68] X. Wang and X. Tang, "Face photo-sketch synthesis and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 31, no. 11, pp. 1955–1967, 2009.
[69] S. Xie and Z. Tu, "Holistically-nested edge detection," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1395–1403.
[70] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2536–2544.
[71] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 465–476.



Jianwen Xie received his Ph.D. degree in statistics from University of California, Los Angeles (UCLA) in 2016. He is currently a senior research scientist at Baidu Research USA. Before joining Baidu, he was a senior research scientist at Hikvision Research Institute USA from 2017 to 2020, and a staff research associate and postdoctoral researcher in the Center for Vision, Cognition, Learning, and Autonomy (VCLA) at UCLA from 2016 to 2017. His research interests focus on generative modeling and learning with applications in computer vision.

Zilong Zheng is currently a Ph.D. candidate in the Center for Vision, Cognition, Learning and Autonomy at the University of California, Los Angeles (UCLA). He received his B.S. degree in computer science at University of Minnesota, and B.E. degree in micro-electronic technology from University of Electronic Science and Technology of China (UESTC). His research interests lie in multimodal representation learning on computer vision and natural language.



Xiaolin Fang is currently a Ph.D. student in CSAIL at Massachusetts Institute of Technology (MIT). Before joining MIT, she received her B.Eng. degree in Computer Science and Technology from Zhejiang University, China in 2019. Her research interests lie in robotics and computer vision.

Song-Chun Zhu received Ph.D. degree from Harvard University in 1996, and is Chair Professor jointly with Tsinghua University and Peking University, director of Institute for Artificial Intelligence, Peking University. He worked at Brown, Stanford, Ohio State, and UCLA before returning to China in 2020 to launch a non-profit organization – Beijing Institute for General Artificial Intelligence. He has published over 300 papers in computer vision, statistical modeling and learning, cognition, Language, robotics, and AI. He received the Marr Prize in 2003, the Aggarwal prize from the Intl Association of Pattern Recognition in 2008, the Helmholtz Test-of-Time prize in 2013, twice Marr Prize honorary nominations in 1999 and 2007, a Sloan Fellowship, the US NSF Career Award, and ONR Young Investigator Award in 2001. He is a Fellow of IEEE since 2011. He serves as General co-Chair for CVPR 2012 and CVPR 2019.

Ying Nian Wu received his Ph.D. degree in statistics from Harvard University in 1996. He was an assistant professor in the Department of Statistics, University of Michigan from 1997 to 1999. He joined University of California, Los Angeles (UCLA) in 1999, and is currently a professor in UCLA Department of Statistics. His research interests include generative modeling, representation learning, and computer vision. He received Honorable Mention for the David Marr Prize with S. C. Zhu et al. in 1999 and 2007 for generative modeling in computer vision.