Learning Probabilistic Models from Generator Latent Spaces with Hat EBM

Mitch Hill
OPPO US Research Center
mitch.hill@innopeaktech.com

Erik Nijkamp
Salesforce Research
erik.nijkamp@salesforce.com

Jonathan Mitchell
University of California, Los Angeles
jcmitchell@ucla.edu

Bo Pang
Salesforce Research
b.pang@salesforce.com

Song-Chun Zhu
University of California, Los Angeles
sczhu@stat.ucla.edu

Abstract

This work proposes a method for using any generator network as the foundation of an Energy-Based Model (EBM). Our formulation posits that observed images are the sum of unobserved latent variables passed through the generator network and a residual random variable that spans the gap between the generator output and the image manifold. One can then define an EBM that includes the generator as part of its forward pass, which we call the Hat EBM. The model can be trained without inferring the latent variables of the observed data or calculating the generator Jacobian determinant. This enables explicit probabilistic modeling of the output distribution of any type of generator network. Experiments show strong performance of the proposed method on (1) unconditional ImageNet synthesis at 128×128 resolution, (2) refining the output of existing generators, and (3) learning EBMs that incorporate non-probabilistic generators. Code and pretrained models to reproduce our results are available at https://github.com/point0bar1/hat-ebm.

1   Introduction

Generator networks [20, 12] which transform a latent distribution to a complex observed distribution (e.g., images, videos) are found across different deep generative models. One limitation of generator networks is the difficulty of obtaining an explicit representation of the probability distribution defined by the output of the network after transformation of the latent space. For generators from Generative Adversarial Networks (GANs) [12, 32] and Variational Autoencoders (VAEs) [20, 33] where the latent states corresponding to realistic images follow a trivial distribution (e.g., isotropic Gaussian), the difficulty of obtaining image space probabilities lies in calculating the log determinant of the generator Jacobian which is needed to perform density change-of-variables. For other generator models such as a deterministic autoencoder, there might not be a natural way to generate probabilistic samples from the latent space that correspond to realistic images. When the latent space has smaller dimension than the image space, there is further difficulty in describing the generator image distribution since the distribution measure must be confined to the manifold of the generator output.

This work proposes a method for using a generator network as the foundation for an Energy-Based Model (EBM). The generator network is concatenated with a *hat network* that takes an image input
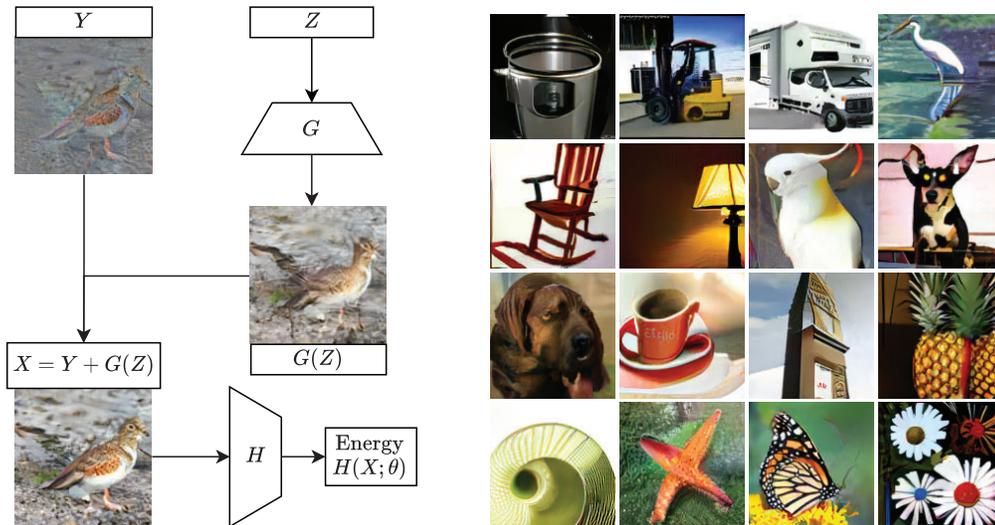
Figure 1: *Left:* The Hat EBM model takes a joint input (Y, Z) where Y is a residual image and Z is a latent vector. An image is generated using $X = Y + G(Z)$ for a generator $G(Z)$, and the image is passed to the hat network $H(X; \theta)$ to obtain the energy of the pair (Y, Z). This allows for principled probabilistic learning which can incorporate the latent space of any generator. *Right:* Unconditional ImageNet 128×128 samples generated by a Hat EBM.

and outputs a scalar. Before the generator output is fed into the hat network, the generated image is adjusted by adding a residual image that spans the gap between the generator output and the image manifold. The total function, including the generator, addition of the residual image, and the hat network, is called the *Hat EBM*. This formulation allows us to define an EBM which can incorporate the generator latent space as part of its image space density and MCMC sampling process. Figure 1 displays a diagram of the Hat EBM and selected Hat EBM samples for unconditional ImageNet at resolution 128×128. Our method is general and applies to any generator model, in contrast to existing methods for converting generator networks to EBMs which only apply to a specific generator model like a GAN [5] or VAE [38]. Additionally, the learning method does not require inference of latent states of the observed data as used in related works [20, 39, 15, 30].

The Hat EBM formulation enables applications with frozen pretrained generator networks such as learning an EBM to refine samples from generators with probabilistic latent spaces (e.g. GANs) and synthesizing samples using non-probabilistic generators (e.g. deterministic autoencoders). We also propose a self-contained learning method that extends cooperative learning of EBM and generator networks [39] to achieve high-quality synthesis. Our main contributions are summarized below.

- We introduce a method for defining a Hat EBM that incorporates a generator network as part of its forward pass. This EBM includes the generator latent space as part of MCMC sampling.
- We show that our method can refine samples from pretrained GAN generators and sample from the latent space of deterministic autoencoders which are originally incompatible with sampling.
- We propose a self-contained Hat EBM learning method that trains both a generator and energy network from scratch. This enables us to achieve an FID score of 29.2 on unconditional ImageNet at resolution 128×128, demonstrating the EBMs are competitive with state-of-the-art generative models on complex and high resolution datasets.

2 Related Work

This section highlights important related methods for EBM learning. A more thorough discussion of related work can be found in Appendix I.

EBM. An EBM defines an unnormalized density or equivalently a Gibbs-Boltzmann distribution. Prototypes include exponential family distributions, Boltzmann machines [1, 34], and the FRAME (Filters, Random field, And Maximum Entropy) model [44]. Recent work has introduced the EBM

with a ConvNet potential [43, 42]. This dramatically increases the model capacity and enables strong image synthesis performance [29, 9] and adversarial robustness [16]. Several works investigate training an EBM in tandem with an auxiliary model. Kim and Bengio [19] jointly train an EBM and generator without MCMC by using samples from the generator as direct approximations of the EBM density. The EGAN [7] builds on this method by introducing a maximum entropy objective to improve generator training. A similar approach is explored by the VERA model [14]. Cooperative learning [41] trains the EBM and generator by using the generator to initialize samples needed to train the EBM and uses reconstruction loss between generator and EBM samples to learn the generator. The Flow Contrastive EBM [11] learns an EBM using Noise Contrastive Estimation with an auxiliary flow model.

Latent Space EBM. EBMs in the data space can be highly multi-modal, and MCMC sampling can be difficult [40, 29, 9]. Recent works [30, 31] explore learning an EBM in latent space, which is then mapped to the data space with a learned generator. The energy landscape in the latent space is smoother and less multi-modal because it occupies much lower dimensional space and stands upon an expressive generator. These works define a prior EBM in the latent space as a correction of the non-informative uniform prior or isotropic Gaussian prior. To learn the model, one needs to infer the posterior of the latent variables. Posterior inference given such a complicated model is non-trivial. One needs to either design a sophisticated amortized inference network or run expensive MCMC. Our model also defines an EBM in the latent space, while its learning does not need posterior inference, making the learning much simpler and more scalable. Several works [36, 5, 2] leverage a pretrained GAN to define an EBM in the latent space of the generator with a correction based on the discriminator, and shows improved synthesis quality over the pretrained GAN. The VAEBM [38] uses a pretrained VAE to facilitate EBM learning. Our model can likewise be utilized to improve the quality of images from pretrained GAN or VAE generators. Our method is however more general since it can be used to obtain realistic samples from any pretrained generator, including non-probabilistic generators from deterministic autoencoders.

3 Formulation of Hat EBM

This section presents the formulation of the Hat EBM energy function and the proposed learning procedure. We first review the fundamental equations of EBM learning. Then we introduce two variants of the Hat EBM: one that learns a joint distribution over the latent space and residual image space, and one that learns a distribution of residual images conditional on a fixed latent state from a known latent distribution. Finally, we propose a method for learning the hat network and generator network of a Hat EBM simultaneously so that our model can be used for self-contained image generation without the need for a pretrained generator.

3.1 Review of EBM Learning

We briefly review the main components of EBM learning following the standard method derived from works such as [17, 44, 40]. A deep EBM has the form

$$p(x; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp\{-U(x; \theta)\} \tag{1}$$

where $U(x; \theta)$ is a deep neural network with weights θ and $\mathcal{Z}(\theta)$ is the intractable normalizing constant. Given a true but unknown data density $q(x)$, Maximum Likelihood learning uses the objective $\arg\min_\theta D_{KL}(q(x) \| p(x; \theta))$, which can be minimized using the stochastic gradient

$$\nabla\mathcal{L}(\theta) \approx \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta U(X_i^+; \theta) - \frac{1}{n}\sum_{i=1}^{n}\nabla_\theta U(X_i^-; \theta) \tag{2}$$

where the positive samples $\{X_i^+\}_{i=1}^n$ are a set of data samples and the negative samples $\{X_i^-\}_{i=1}^n$ are samples from the current model $p(x; \theta)$. To obtain the negative samples for a deep EBM, it is common to use MCMC sampling with the K steps of Langevin equation

$$X^{(k+1)} = X^{(k)} - \frac{\varepsilon^2}{2}\nabla_{X^{(k)}} U(X^{(k)}; \theta) + \varepsilon V_k, \tag{3}$$

where ε is the step size and $V_k \sim N(0, I)$. The Langevin trajectories are initialized from a set of states $\{X_{i,0}^-\}_{i=1}^n$ obtained from a certain initialization strategy.

3.2 Hat EBM: Joint Distribution of Latent and Residual Image

This section presents our method for adapting a fixed generator network $G(z)$ to be part of an EBM, which we call the *Hat EBM*. The Hat EBM defines the joint distribution of the random variable $Z \in \mathbb{R}^m$ in the m-dimensional latent space of the generator network and a random variable $Y \in \mathbb{R}^d$ in the d-dimensional image space. The joint energy has the form

$$U(Y, Z; \theta) = H(G(Z) + Y; \theta) \tag{4}$$

where $H(x; \theta)$ is a neural network that takes an image $x \in \mathbb{R}^d$ as input and returns a scalar output. The weights of H are given by θ. We call H the *hat network* because it sits atop the generator G to incorporate the generator latent space directly into the probabilistic model.

The random variable Y is meant to accommodate the gap between the output of $G(Z)$ and the image manifold, since in general we expect that $G(Z)$ contains an approximate but imperfect representation of the image distribution which can be refined by the residual state Y. In practice, we find that the majority of the appearance of a sampled image $X = G(Z) + Y$ comes from the generator output $G(Z)$ and not the residual image Y, indicating that the majority of the sampling behaviors of our model are determined by the latent space of G (see Figure 3).

An appealing aspect of the Hat EBM formulation is that we can learn the model without either calculating the log determinant of the Jacobian of $G(Z)$ as would be required for an energy of the form $U(G(Z); \theta)$, or inferring the Z latent vectors associated with observed images X as done in existing work on learning latent prior EBMs [30]. This is possible because we define the distribution of observed images X by $X = G(Z) + Y$ where the pair (Y, Z) is drawn from a true but unknown density $q(y, z)$. We can then use the Maximum Likelihood framework in Section 3.1 to learn the weights θ of the hat network $H(x; \theta)$ by minimizing the objective $\arg\min_\theta D_{KL}(q(y, z) \| p(y, z; \theta))$ where

$$p(y, z; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp\{-H(G(z) + y; \theta)\}. \tag{5}$$

One can obtain negative samples using alternating Langevin updates

$$Y^{(k+1)} = Y^{(k)} - \frac{\varepsilon_1^2}{2} \nabla_{Y^{(k)}} H(G(Z^{(k)}) + Y^{(k)}; \theta) + \varepsilon_1 V_{k,1} \tag{6}$$

$$Z^{(k+1)} = Z^{(k)} - \frac{\varepsilon_2^2}{2} \nabla_{Z^{(k)}} H(G(Z^{(k)}) + Y^{(k+1)}; \theta) + \varepsilon_2 V_{k,2} \tag{7}$$

which switch off between updates with respect to z and updates with respect to y. This sampling algorithm is essentially Metropolis-within-Gibbs since the Langevin update can be written as a Metropolis-Hastings step with Gaussian proposal, in which case (6) and (7) define a valid sampler for $p(y, z; \theta)$. Finally, updating θ can be accomplished using

$$\nabla \mathcal{L}(\theta) \approx \frac{1}{n} \sum_{i=1}^n \nabla_\theta H(X_i^+; \theta) - \frac{1}{n} \sum_{i=1}^n \nabla_\theta H(G(Z_i^-) + Y_i^-; \theta) \tag{8}$$

where X_i^+ are observed samples and the pairs (Y_i^-, Z_i^-) are obtained via MCMC. In our formulation the observed data X_i^+ are sufficient statistics for $H(x; \theta)$ and there is no need to infer the (Y_i^+, Z_i^+) pairs for the positive samples when learning the weights of the hat network.

3.3 Conditional Hat EBM: Residual Image Conditional on Latent Sample

Next we present a conditional variant of the Hat EBM. While the previous version of the Hat EBM is applicable to any generator network $G(z)$, including generators from deterministic autoencoders which cannot typically be sampled from, the conditional version of the Hat EBM is tailored towards generator networks which map a trivial latent distribution to complex signals like images. For these kinds of generators, one can use the known latent distribution as an ancestral distribution and learn a conditional distribution of the residual image given a latent sample. We emphasize that the conditional Hat EBM can be used to learn an unconditional distribution of observed images X and that the term *conditional* refers to the relationship between the latent variables Y and Z. Our experiments focus on modeling only observed images X without conditional information such as labels or captions.

Suppose we use a trivial marginal distribution $p_0(z)$ for Z and a generator G trained to produce realistic images from this latent distribution. In our experiments, p_0 is always $N(0, I)$. We can now define a conditional Hat EBM density $p(y|z; \theta) = \frac{1}{\mathcal{Z}_z(\theta)} \exp\{-H(G(z) + y; \theta)\}$ and a joint density

$$p(y, z; \theta) = \frac{1}{\mathcal{Z}_z(\theta)} p_0(z) \exp\{-H(G(z) + y; \theta)\}. \tag{9}$$

In this case, we posit that observed images X are generated according to $X = G(Z) + Y$ for some distribution $q(y, z) = p_0(z)q(y|z)$. Obtaining the negative samples (Z_i^-, Y_i^-) is done by first drawing Z_i^- from $p_0(z)$ and then obtaining $Y_i^- | Z_i^-$ by using Langevin updates on the conditional probability $p(y|z; \theta)$. Note that $p(y|z; \theta) = p(y|G(z); \theta)$ because of the form of (9). We update θ using the same equation (8) as the joint Hat EBM because X_i^+ is still a sufficient statistic for learning $H(x; \theta)$ and because we do not need to infer the Z_i^+ for X_i^+ since the prior $p_0(z)$ does not contain any model parameters. In practice we initialize Langevin sampling from $Y_0 = 0$ and $Z \sim N(0, I)$, and perform K steps of (6) to draw a residual sample Y_K while keeping Z fixed.

3.4   Learning the Hat Network and Generator in Tandem for Conditional Hat EBM

Both formulations of the Hat EBM above assume that a pretrained generator network $G(z)$ is available as the basis for learning the Hat EBM. In order to use the Hat EBM as a self-contained learning process for image generation, we now propose a method to learn the weights ϕ of a generator network $G(z; \phi)$ and the weights θ of a hat network $H(x; \theta)$ simultaneously. Our method is based on the cooperative learning [39] strategy. We first briefly review the cooperative learning formulation, and then present the learning formulation for the Hat EBM. A key difference between the derivations is that the original cooperative learning method requires MCMC inference of the latent variable $\hat{Z}$ associated with an MCMC sample X to train $G(z; \phi)$, while in our formulation the latent variable Z is explicitly defined and does not need to be inferred. This enables major computational savings because we can bypass the expensive MCMC inference of $\hat{Z}$.

In the original cooperative learning [39], the generator output $G(z; \phi)$ is trained to match the appearance of a Langevin chain X_K sampled from the potential $U(x; \theta)$ and initialized from the state $X_0 = G(Z_0; \phi)$ where $Z_0 \sim N(0, I)$. The model defines the conditional density of an images X given latents $Z \sim N(0, I)$ as $X|Z \sim N(G(Z; \phi), \tau^2 I)$ for some sufficiently small $\tau > 0$. Given a sampled state X_K, updating $G(z; \phi)$ requires inferring $Z|X_K$ using the latent Langevin equation

$$Z_{k+1} = Z_k - \frac{\varepsilon^2}{2} \left(Z_k + \frac{1}{2\tau^2} \nabla_{Z_k} \|G(Z_k; \phi) - X_K\|_2^2 \right) + \varepsilon V_k \tag{10}$$

before updating ϕ using the Maximum Likelihood stochastic gradient

$$\nabla \mathcal{L}_G(\phi) \approx \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2\tau^2} \nabla_\phi \|G(Z_{K,i}; \phi) - X_{K,i}\|_2^2 \tag{11}$$

where the i index denotes member i of a batch with size n. We note that the code released with the cooperative learning method does not infer the latent variable of observed images and $Z_0 \sim N(0, I)$ is used in place of Z_K in the objective (11). In accordance with this approach, we find that inferring Z_K often hurts model performance and leads to additional complication. The difficulty of inferring $Z|X_K$ and the omission of this step in practice leave an unresolved gap in the cooperative learning formulation. The Hat EBM generator update allows us to bypass the Langevin update for Z without theoretical complications.

To update the weights of a Hat EBM generator, we propose to train the generator to match the image samples produced by the current Hat EBM. Given the current generator weights ϕ_t and hat network weights θ_t at step t, we define our learnable model as $Z \sim N(0, I)$ and $X|Z \sim N(G(Z; \phi), \tau^2 I)$ and we define the true distribution of (X, Z) as $Z \sim N(0, I)$ and $X = Y_K + G(Z; \phi_t)$ where $Y_K|Z$ is drawn from K steps of Langevin sampling with Hat EBM density $p(y|z; \theta_t, \phi_t) \propto \exp\{-H(G(z; \phi_t) + y; \theta_t)\}$. Langevin updating is only used to obtain Y_K while Z remains fixed, as in the method from Section 3.3. Then ϕ can be updated using the Maximum Likelihood objective

$$\phi_{t+1} = \arg\min_{\phi} \frac{1}{2\tau^2} E_{p_0(z)p(y|z; \theta_t, \phi_t)} \left[\|G(Z; \phi) - (Y + G(Z; \phi_t))\|_2^2 \right]. \tag{12}$$
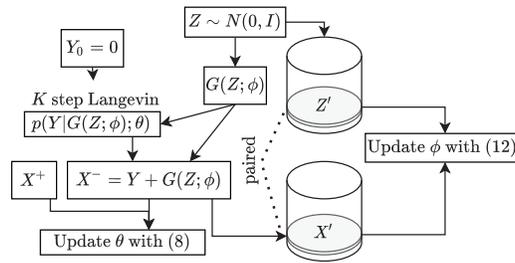
Figure 2: Visualization of tandem training method for hat network and generator. The left side illustrates training for the hat network $H(x;\theta)$. Z is drawn from a latent distribution, Y is initialized from the 0 image and updated according to $p(Y|Z;\theta,\phi) = p(Y|G(Z;\phi);\theta)$. Then data samples X^+ and negative samples $X^- = Y + G(Z;\phi)$ are used to update the weight θ of the hat network. On the right, pairs (X', Z') from past hat network updates will be drawn randomly from a bank of states to update the weight ϕ of the generator. The bank memory (X', Z') will then be overwritten by a new pair (X, Z) from the current model.

Conceptually, this loss function encourages $G(Z;\phi)$ to closely match the appearance of samples $X = Y + G(Z;\phi_t)$ created from a fixed generator $G(Z;\phi_t)$ and fixed hat network $H(x;\theta_t)$. In other words, the generator update should achieve $G(Z;\phi_{t+1}) \approx Y + G(Z;\phi_t)$ so that the updated generator absorbs the residual image $Y|Z$ from the current Hat EBM. Once the generator absorbs the current Hat EBM residual, the hat network update should learn to synthesize residual images that refine the the generator output to be more similar to the observed data. Like the original cooperative learning method, our generator is trained using only synthetic images and no observed data images are used to update ϕ.

Ideal training of H and G would alternate between using the gradient of (12) until generator convergence is reached and using the gradient (8) to update the hat network. In practice we implement the minimization in (12) using only one gradient update initialized from $\phi = \phi_t$ to obtain ϕ_{t+1} rather than training until full convergence. This is done to increase training efficiency and to avoid maintaining a separate copy of generator weights for the fixed network $G(z;\phi_t)$.

In our experiments we observe that using a single gradient update with the objective (12) has limited success because the generator output can become too closely tethered to biases of the current hat network. We find the same problems with the original cooperative learning objective (11) (see Appendix F). To overcome these problems, we choose to train $G(z;\phi)$ at time $t+1$ to match the historical distribution of hat networks $H(x;\theta_{t_\ell})$ and generators $G(z;\phi_{t_\ell})$ for a selection of past epochs $t_1, t_2, \ldots t_L \leq t$ instead of training the generator to match the distribution of the current hat network $H(x;\theta_t)$ and generator $G(z;\phi_t)$. This simply involves redefining the true distribution (X, Z) by first sampling t_ℓ from $\{t_1, \ldots, t_L\}$ and then generating $Z \sim N(0, I)$ and $X_K = Y_K + G(Z;\phi_{t_\ell})$ where $Y|Z$ follows the energy $H(G(Z;\phi_{t_\ell}) + Y;\theta_{t_\ell})$. In this case the loss (12), after replacing ϕ_t with ϕ_{t_ℓ}, is a stochastic approximation of the Maximum Likelihood gradient defined by the joint distribution (t_ℓ, Z, X_K) (see Appendix F). In practice, we implement this procedure by keeping a persistent bank of 10,000 pairs (X, Z) created from past hat network updates. When updating the generator, we draw $n = 128$ pairs from the bank and replace it with a newly generated batch of $n = 128$ pairs from the current hat EBM. This ensures that the selection $\{t_1, \ldots, t_L\}$ of past epochs remains within a close range of the current epoch t with high probability. Saving the generated images X at each EBM update allows us to learn from past generator weights ϕ_{t_ℓ} without maintaining a copy of the weights. See Figure 2 for an illustration and Appendix G for a code sketch.

4 Experiments

In the subsequent empirical evaluations, we will address the following questions:

1. **Refinement:** To what extent can our method refine samples from a pretrained generator model with known prior distribution?

2. **Retrofit:** Is our method capable of turning a generator model pretrained as a deterministic autoencoder into a generative model for which samples resemble realistic images?

933

https://doi.org/10.52202/068431-0068

3. **Synthesis:** Can our method learn a generator from scratch with competitive quality of synthesis on common image datasets? Can our method be scaled up to challenging datasets such as ImageNet with competitive synthesis for unconditional sampling?

4. **Out-of-Distribution:** Can the Hat EBM be used for Out-of-Distribution (OOD) detection to distinguish between samples from the training distribution and samples from dissimilar distributions?

Figure 3 visualizes some representative sampling paths for the models trained to investigate questions 1, 2, and 3 on the CIFAR-10 dataset. The appendix contains details such as pseudocode, training parameters, and model architectures.

4.1 Refinement

In this section, we examine the problem of refining samples from a pretrained generator using a joint Hat EBM trained according to the method in Section 3.2. A pretrained SN-GAN with generator is used as a baseline generator G. We learn a Hat EBM that incorporates the generator network to refine the initial generator samples. The hat network has the exact same structure as the SN-GAN discriminator except that we remove spectral norm layers. We use fixed batch norm statistics for the generator and our energy is deterministic. The experiment is performed for the CIFAR-10 32×32 and CelebA 64×64 datasets. To evaluate our method, we compare with Discriminator Driven Latent Sampling (DDLS) [5], which takes the pretrained discriminator D learned with G and samples from the potential $U(z) = D(G(z)) + \frac{1}{2}\|z\|_2^2$. We train D and G using the Mimicry repository for reproducible GAN experiments [22] to obtain stronger baselines for $G(z)$ than those used in [5].

Table 1 shows our results. The Hat EBM can learn a hat network capable of refining a image appearance more successfully than DDLS on both CIFAR-10 and CelebA. We use the joint version of the Hat EBM where both Y and Z are updated during sampling. Sampling is initialized from $Y_0 = 0$ and $Z_0 \sim N(0, I)$. The hat networks learns to tilt Z away from its initial normal distribution to find a nearby latent vector with a more realistic appearance.

Table 1: Improvement in FID by refining samples from a fixed SN-GAN generator.

Model	CIFAR-10	CelebA
SN-GAN (baseline) [26]	18.58 ± 0.08	6.13 ± 0.03
DDLS [5]	14.59 ± 0.07	6.06 ± 0.01
Hat EBM (*Ours*)	$\mathbf{14.04 \pm 0.11}$	$\mathbf{5.98 \pm 0.02}$

We observe that the majority of the refinement is occurring in the latent space, and that the residual image Y is essentially imperceptible (see Figure 3). We also notice that training quickly becomes unstable when Y is removed from the Hat EBM (see Appendix H). In our experience, it is essential to incorporate the residual for stable learning. A possible reason for this phenomenon is that the hat network can learn to discriminate between generator images and images not from the generator, whether they are realistic or not. If so, the hat network can assign increasingly high energy to generator samples in the absence of the residual Y. Even an imperceptible Y is sufficient to prevent the hat network from easily distinguishing positive and negative samples so that learning is stable.

4.2 Retrofit

In this section, we incorporate a non-probabilistic generator network $G(z)$ into a probabilistic Hat EBM model. This essentially allows us to sample from the latent space of $G(z)$ to find latent samples whose mapping corresponds to a realistic image. Like the results in Section 4.1, the residual image has small norm in the image space and most of the appearance of the sampled images comes from $G(z)$. This happens naturally without the need to coerce Y to be close to 0 by including a prior term such as $p_0(y) \propto \exp\{-\frac{1}{2\sigma^2}\|y\|_2^2\}$, although including a prior term can further limit growth of Y.

The autoencoder generator $G(z)$ is pretrained as the second half of a standard inference network and generator network pairing. An image X is fed into the inference network I and converted into a latent state $Z = I(X)$, which is then decoded to obtain a reconstruction $\hat{X} = G(Z)$. The inference network and generator are learned jointly using the MSE loss $\|\hat{X} - X\|_2^2$. To keep the latent space mapping
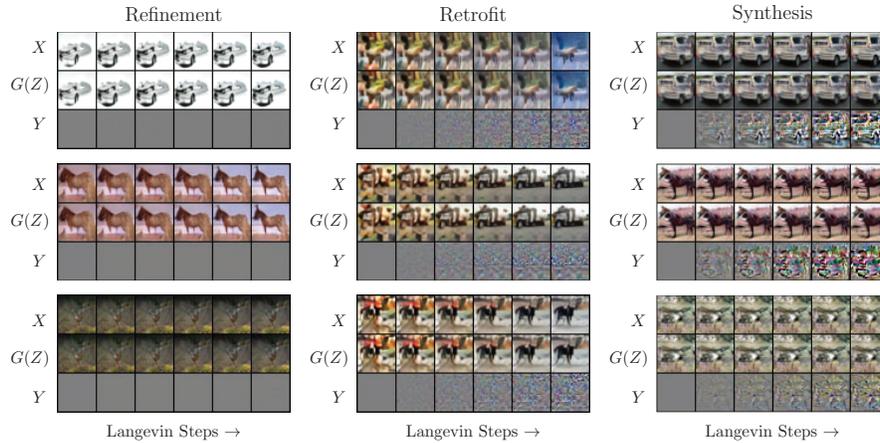
934

Figure 3: Visualization of shortrun Langevin paths used by each kind of Hat EBM. The same trajectories are used for both training and test-time generation. In each grouping, the three rows correspond to image X, generator output $G(Z)$, and residual image Y. All paths are initialized with residual $Y_0 = 0$ which corresponds to a grey image. For visualization purposes, all Y images have been magnified by a factor of 5. The Refinement paths (*left*) use the joint Hat EBM to refine the appearance of a pretrained SN-GAN generator. The residual Y is barely noticeable, but still is essential for stable learning. The Retrofit paths (*center*) use the joint Hat EBM to sample using a non-probabilistic autoencoder generator. The residual Y is somewhat noticeable after magnification but most of the image appearance comes from $G(Z)$. The Synthesis paths (*right*) use the conditional Hat EBM to sample refine a generator learned in tandem. $G(Z)$ is fixed during sampling. The residual Y plays a significant role in refining the generator appearance.

of $I(X)$ numerically stable, we project the m-dimensional raw output of the inference network to the sphere around the origin with radius $\sqrt{m}$ so that $\|I(X)\|_2 = \sqrt{m}$. More sophisticated methods such as perceptual and adversarial loss could have been used to train the autoencoder, but we use MSE loss to keep our implementation minimal. We observe that when Z is a vector-shaped latent state, it can be difficult to learn reconstructions $\hat{X}$ with sharp appearance even for simple datasets like CIFAR-10. To obtain better reconstructions and therefore a latent space with more realistic mappings to the image space, we use image-shaped latent states Z. The details of our autoencoder networks can be found in Appendix E. When Z is an image shaped latent, we treat it exactly the same as a vector latent in the learning and sampling algorithms.

We experiment with assimilating an autoencoder generator into a Hat EBM potential for the CIFAR-10 dataset. Our results are visualized in Figure 3 with additional results in the Appendix K. We train the Hat EBM using shortrun learning in the latent and image space by initializing $Y_0 = 0$ and $Z_0 \sim N(0, I)$ and using $K = 100$ MCMC steps of (6) and (7) from initialization during both training and testing evaluation to generate samples. During the Langevin dynamics, image appearance is refined mostly in the latent space. Our best model achieves a solid FID score of 26.01 ± 0.09. This demonstrates that Hat EBM can learn a probabilistic model over a non-probabilistic latent space.

4.3 Synthesis

In this section, we use the conditional Hat EBM formulation from Section 3.4 to learn a hat network and generator network from scratch for self-contained synthesis. We explore synthesis for unconditional CIFAR-10 at resolution 32×32, CelebA at resolution 64×64, and unconditional ImageNet at resolution 128×128. While recent generative models show promising results for class conditional sampling, unconditional sampling with high quality synthesis remains a significant challenge. We find especially strong results for unconditional ImageNet synthesis using the Hat EBM. This demonstrates strong potential of our synthesis method for learning with highly diverse unstructured datasets.

Our Hat EBM models use SN-GAN architectures for sizes 32×32, 64×64, and 128×128, where the discriminator architecture is used for the hat network. During learning, we keep the generator batch norm parameters fixed to mean 0 and variance 1. We remove all spectral norm layers from the hat network. Training parameters can be found in Appendix J. For ImageNet models, we found that

935

Table 2: Comparison of FID scores among representative generative models. For CIFAR-10 and Celeb-A, EBM models are above the dividing line and other generative models are below. For ImageNet, smaller scale nets are above the line and larger scale nets are below the line. (*=EBM, †=reimplementation)

CIFAR-10 (32×32)	
Model	FID
Hat EBM (*Ours*)*	**19.30 ± 0.15**
Improved CD EBM [8]*	25.1
VERA [14]*	27.5
GEBM [3]*†	30.1
Cooperative EBM[39]*	33.6
Flow EBM [11]*	37.3
JEM [13]*	38.4
IGEBM [9]*	40.6
DDPM [18]	**3.2**
NCSNv2[35]	10.9
BigGAN [4]	14.7
SN-GAN [26]†	18.6

CelebA (64×64)	
Model	FID
Hat EBM (*Ours*)*	**11.57 ± 0.04**
Divergence Triangle [15]*	31.9
SN-GAN [26]	**6.1**
NCSNv2[35]	10.2

ImageNet (128×128)	
Model	FID
Hat EBM (*Ours*)*	**40.24 ± 0.18**
SN-GAN [26]	65.7
SS-GAN [6]	43.9
InfoMax GAN [23]	58.9
Hat EBM, scaled (*Ours*)*	29.37 ± 0.15
SS-GAN, scaled [6]	**23.4**

annealing the generator and hat network learning rate by a factor of 10 after 250K weight updates for each network improved the FID score significantly. See Appendix K for uncurated samples from our Hat EBM models. FID results for our model and a representative selction of generative models are shown in Table 2. We were unable to reproduce the GEBM FID scores reported in [3] and we report the score obtained from a reimplementation with the official training code. For the SN-GAN FID score, we report the stronger baseline from our reimplementation in Section 4.1.

Results show strong performance of the Hat EBM compared to competing generative models across all datasets, with an especially strong performance for ImageNet. Our method significantly outperforms other EBM learning methods on CIFAR-10. The Hat EBM synthesis results are on par with the SN-GAN baseline for CIFAR-10 and CelebA, and the Hat EBM significantly outperforms SN-GAN for ImageNet. With a budget of 8 GPUs and about 2.5 days of computing, our Hat EBM achieves an ImageNet FID score of 40.0, outperforming the small-scale SS-GAN.

To our knowledge, the current state-of-the-art model for unconditional ImageNet synthesis at resolution 128×128 is the large-scale SS-GAN [6], which achieves an FID score of 23.4. This model was trained using a BigGAN architecture and 128-core TPUv3 pods. To scale up our Hat EBM, we doubled the number of channel dimensions for both the hat network and generator network from the original SN-GAN architecture and trained on 32-core TPU-v3 pods. Our best FID score for unconditional ImageNet 128×128 was 29.2, which comes within a competitive range of state-of-the-art. We believe that further scaling in future work could enable Hat EBM to match or surpass state-of-the-art. Our results decisively demonstrate the potential of EBM learning for high-quality synthesis well beyond the scale investigated in any prior EBM work.

4.4   Out-of-Distribution Detection

Experiments in this section assess Hat EBM performance on Out-Of-Distribution (OOD) detection. We use the conditional Hat EBM model trained on CIFAR-10 from Section 4.3 and calculate the energy $H(X; \theta)$ on unseen in-distribution images from the CIFAR-10 test set and images from OOD datasets which include CIFAR-100, CelebA, and SVHN. We follow standard OOD evaluation from works such as [27] which compute the AUROC metric on the energy scores of the in-distribution and OOD samples. This metric measures the ability of the Hat EBM to distinguish between in-distribution samples not seen during training and OOD samples. Following [13, 38], we expect that the energy of the OOD datasets will be higher than the energy of in-distribution test images since higher energy samples should appear with lower frequency in the learn density.

Our results are shown in Table 3. The Hat EBM shows strong performance as an OOD detection method. Among methods that are fully unsupervised, our model has the top performance across all three OOD datasets. Our method approaches the results of methods that are trained with labeled

Table 3: Comparison of OOD scores (AUROC) among representative models. We separate scores for fully unsupervised models (above the line) and models which used supervised data (below the line).

Model	SVHN	CIFAR-100	CelebA
Ours	**0.92**	**0.87**	**0.94**
IGEBM [9]	0.43	0.54	0.69
VAEBM [38]	0.83	0.62	0.77
Improved CD EBM [8]	0.91	0.83	–
JEM [13]	0.67	0.87	0.77
HDGE [24]	0.96	0.91	0.80
OOD EBM [25]	0.91	0.87	0.78
OOD EBM (fine-tuned) [25]	**0.99**	**0.94**	**1.00**

data such as HDGE [24] and the fine-tuned OOD EBM [25], although we do not yet match these scores. Overall, there is strong evidence that the Hat EBM is naturally an effective method for OOD detection, especially when supervised label information is unavailable.

5 Conclusion

Maximum Likelihood learning of EBMs poses a significant challenge: drawing negative samples from the current density model, which is often highly multi-modal. Prior art addresses this challenge by recruiting approximations of the EBM in the form of an ancestral sampling from a generator model, truncated Langevin chains, flow-based models, or lifting the EBM into the induced latent space of generator models. In contrast, our work proposes a method for absorbing any generator as a backbone of an EBM. The formulation assumes that observed images are the sum of unobserved latent variables pushed forward through the generator and a residual random variable which closes the gap between generator samples and image manifold. The hat network sits atop the generator and residual and both nets form the Hat EBM. The generator allows for efficient sampling but may only capture the coarse structure of the images, while the residuals can capture fine-grain details.

The Hat EBM formulation is presented in three variations: (1) joint learning of latent and residual image for adapting any fixed generator, (2) conditional learning for generators with known prior distribution, (3) self-contained learning of both EBM and generator from scratch. Notably, the training requires neither the log determinant of the generator Jacobian or inference of latent variables, making the learning simple and scalable.

Empirical evaluations demonstrate the various capabilities of the Hat EBM: (1) strong performance for the ImageNet synthesis at 128×128 resolution with self-contained learning, (2) significant refinement of the quality of synthesis of pretrained generators on CIFAR-10 and CelebA with conditional learning, (3) retrofitting pretrained autoencoder generators with a means of sampling, (4) out-of-distribution detection with state-of-the-art performance for unsupervised models.

Acknowledgements

References

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

[2] Abdul Fatir Ansari, Ming Liang Ang, and Harold Soh. Refining deep generative models via discriminator gradient flow. In *International Conference on Learning Representations*, 2021.

[3] Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. In *International Conference on Learning Representations*, 2021.

[4] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019.

[5] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your gan is secretly an energy-based model and you should use discriminator driven latent sampling. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[6] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[7] Zihang Dai Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. Calibrating energy-based generative adversarial networks. In *International Conference on Learning Representations*, 2017.

[8] Yilun Du, Shuang Li, B. Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[9] Yilun Du and Igor Mordatch. Implicit generation and generalization with energy based models. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[10] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. Learning generative convnets via multi-grid modeling and sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[11] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. Flow contrastive estimation of energy-based models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[13] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2020.

[14] Will Sussman Grathwohl, Jacob Jin Kelly, Milad Hashemi, Mohammad Norouzi, Kevin Swersky, and David Duvenaud. No {mcmc} for me: Amortized sampling for fast and stable training of energy-based models. In *International Conference on Learning Representations*, 2021.

[15] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Divergence triangle for joint training of generator model, energy-based model, and inferential model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[16] Mitch Hill, Jonathan Craig Mitchell, and Song-Chun Zhu. Stochastic security: Adversarial defense using long-run dynamics of energy-based models. In *International Conference on Learning Representations*, 2021.

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

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[19] Taesup Kim and Yoshua Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2013.

[21] Rithesh Kumar, Anirudh Goyal, Aaron C. Courville, and Yoshua Bengio. Maximum entropy generators for energy-based models. *arXiv preprint arXiv:1901.08508*, 2019.

[22] Kwot Sin Lee and Christopher Town. Mimicry: Towards the reproducibility of gan research. *arXiv preprint arXiv:2005.02494*, 2020.

[23] Kwot Sin Lee, Ngoc-Trung Tran, and Ngai-Man Cheung. Infomax-gan: Improved adversarial image generation via information maximization and contrastive learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021.

[24] Hao Liu and Pieter Abbeel. Hybrid discriminative-generative training via contrastive learning. *arXiv preprint arXiv:2007.09070*, 2021.

[25] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[27] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019.

[28] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. On the anatomy of MCMC-based maximum likelihood learning of energy-based models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

[29] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[30] Bo Pang, Tian Han, Erik Nijkamp, Song-Chun Zhu, and Ying Nian Wu. Learning latent space energy-based prior model. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[31] Bo Pang and Ying Nian Wu. Latent space energy-based model of symbol-vector coupling for text generation and classification. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[32] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Representation Learning*, 2016.

[33] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[34] Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep boltzmann machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, 2009.

[35] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[36] Akinori Tanaka. Discriminator optimal transport. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[37] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[38] Zhisheng Xiao, Karsten Kreis, Jan Kautz, and Arash Vahdat. Vaebm: A symbiosis between variational autoencoders and energy-based models. In *International Conference on Learning Representations*, 2021.

[39] Jianwen Xie, Yang Lu, Ruiqi Gao, and Ying Nian Wu. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[40] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Yingnian Wu. A theory of generative convnet. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[41] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. Cooperative training of fast thinking initializer and slow thinking solver for conditional learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[42] Jianwen Xie, Zilong Zheng, Ruiqi Gao, Wenguan Wang, Song-Chun Zhu, and Ying Nian Wu. Learning descriptor networks for 3D shape synthesis and analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[43] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017.

[44] Song Chun Zhu, Ying Nian Wu, and David Mumford. Filters, random fields and maximum entropy (FRAME): towards a unified theory for texture modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

Checklist

1. For all authors...

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

    (b) Did you describe the limitations of your work? [Yes] See Appendix A.

    (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix B.

    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

    (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Appendix D.

    (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and pretrained models are available at `https://github.com/point0bar1/hat-ebm`.

    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See configs in code files and Appendix J.

    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Primary numerical results are FID scores. Each experiment was run 3 times and estimated standard deviation was reported.

    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

    (a) If your work uses existing assets, did you cite the creators? [Yes] Our code is annotated to acknowledge our use of existing assets. In particular, we used network structures based on existing popular models. All other code was created by the authors.

    (b) Did you mention the license of the assets? [N/A]

    (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code and pretrained models are available at `https://github.com/point0bar1/hat-ebm`.

    (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

    (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

    (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

    (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]