BOOSTING GENERATIVE MODELS BY LEVERAGING CASCADED META-MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

A deep generative model is a powerful method of learning a data distribution, which has achieved tremendous success in numerous scenarios. However, it is nontrivial for a single generative model to faithfully capture the distributions of the complex data such as images with complicate structures. In this paper, we propose a novel approach of cascaded boosting for boosting generative models, where meta-models (i.e., weak learners) are cascaded together to produce a stronger model. Any hidden variable meta-model can be leveraged as long as it can support the likelihood evaluation. We derive a decomposable variational lower bound of the boosted model, which allows each meta-model to be trained separately and greedily. We can further improve the learning power of the generative models by combing our cascaded boosting framework with the multiplicative boosting framework.

1 INTRODUCTION

The past decade has witnessed tremendous success in the field of deep generative models (DGMs) in both unsupervised learning (Goodfellow et al., 2014; Kingma & Welling, 2013; Radford et al., 2015) and semi-supervised learning (Abbasnejad et al., 2017; Kingma et al., 2014; Li et al., 2018) paradigms. DGMs learn the data distribution by combining the scalability of deep learning with the generality of probabilistic reasoning. However, it is not easy for a single parametric model to learn a complex distribution, since the upper limit of a model's ability is determined by its fixed structure. If a model with low capacity was adopted, the model would be likely to have a poor performance. Straightforwardly increasing the model capacity (e.g., including more layers or more neurons) is likely to cause serious challenges, such as vanishing gradient problem (Hochreiter et al., 2001) and exploding gradient problem (Grosse, 2017).

An alternative approach is to integrate multiple weak models to achieve a strong one. The early success was made on mixture models (Dempster et al., 1977; Figueiredo & Jain, 2002; Xu & Jordan, 1996) and product-of-experts (Hinton, 1999; 2002). However, the weak models in such work are typically shallow models with very limited capacity. Recent success has been made on boosting generative models, where a set of meta-models (i.e., weak learners) are combined to construct a stronger model. In particular, Grover & Ermon (2018) propose a method of multiplicative boosting, which takes the geometric average of the meta-model distributions, with each assigned an exponentiated weight. This boosting method improves performance on density estimation and sample generation, compared to a single meta-model. However, the boosted model has an explicit partition function, which requires importance sampling (Rubinstein & Kroese, 2016) for an estimation. In general, sampling from the boosted model is conducted based on Markov chain Monte Carlo (MCMC) method (Hastings, 1970). As a result, it requires a high time complexity of likelihood evaluation and sample generation.

Rosset & Segal (2003) propose another method of additive boosting, which takes the weighted arithmetic mean of meta-models' distributions. This method can sample fast, but the improvement of performance on density estimation is not comparable to the multiplicative boosting, since additive boosting requires that the expected log-likelihood and likelihood of the current meta-model are better-or-equal than those of the previous boosted model (Grover & Ermon, 2018), which is difficult to satisfy. In summary, it is nontrivial for both of the previous boosting methods to balance well between improving the learning power and keeping the efficiency of sampling and density estimation.

To address the aforementioned issues, we propose a novel boosting framework, called cascaded boosting, where meta-models are connected in cascade. The framework is inspired by the greedy layer-wise training algorithm of DBNs (Deep Belief Networks) (Bengio et al., 2007; Hinton et al., 2006), where an ensemble of RBMs (Restricted Boltzmann Machines) (Smolensky, 1986) are converted to a stronger model. We propose a decomposable variational lower bound, which reveals the principle behind the greedy layer-wise training algorithm. The decomposition allows us to incorporate any hidden variable meta-model, as long as it supports likelihood evaluation, and train these meta-models separately and greedily, yielding a deep boosted model. Finally, We demonstrate that our boosting framework can be integrated with the multiplicative boosting framework (Grover & Ermon, 2018), yielding a hybrid boosting with an improved learning power of generative models. To summary, we make the following contributions:

- We propose a boosting framework to boost generative models, where meta-models are cascaded together to produce a stronger model.
- We give a decomposable variational lower bound of the boosted model, which reveals the principle behind the greedy layer-wise training algorithm.
- We finally demonstrate that our boosting framework can be extended to a hybrid model by integrating it with the multiplicative boosting models, which further improves the learning power of generative models.

2 APPROACH

In subsection 2.1, we review the current multiplicative boosting (Grover & Ermon, 2018). Then, we present our cascaded boosting. We first figure out how to connect meta-models, and then propose our boosting framework with its theoretical analysis. Afterwards, we discuss the convergence of our cascaded boosting.

2.1 MULTIPLICATIVE BOOSTING

Grover & Ermon (2018) introduced multiplicative boosting, which takes the geometric average of meta-models' distributions, with each assigned an exponentiated weight α_i as

$$P_n(x) = \frac{\prod_{i=0}^{n} M_i^{\alpha_i}(x)}{Z_n}, \tag{1}$$

where $M_i(x)$ ($0 \quad i \quad n$) are distributions of meta-models, which are required to support likelihood evaluation, $P_n(x)$ is the distribution of the boosted model and Z_n is the partition function. The first meta-model M_0 is trained on the empirical data distribution p_D which is defined to be uniform over the dataset D. The other meta-models M_i ($1 \quad i \quad n$) are trained on a reweighted data distribution p_{D_i} as

$$\max_{M_i} \mathbf{E}_{p_{D_i}} [\log M_i(x)], \tag{2}$$

where $p_{D_i} \propto \left(\frac{p_D}{P_{i-1}}\right)^{\beta_i}$ with $\beta_i \in [0, 1]$ being the hypermeter.

Grover & Ermon (2018) show that the expected log-likelihood of the boosted model P_i over the dataset D will not decrease (i.e., $\mathbf{E}_{p_D} [\log P_i(x)] \quad \mathbf{E}_{p_D} [\log P_{i-1}(x)]$) if Equation 2 is maximized. The multiplicative boosting succeeds in improving the learning power of generative models. Compared to an individual meta-model, it has better performance on density estimation and generates samples of higher quality. However, importance sampling and MCMC are required to evaluate the partition function Z_n and generate samples respectively, which limits its application in occasions requiring fast density estimation and sampling. To overcome these shortcomings, we propose our cascaded boosting framework.

2.2 HOW TO CONNECT META-MODELS

In multiplicative boosting, distributions of meta-models are connected by multiplication, leading to the troublesome partition function. To overcome this problem, we connect meta-models in cascade. Suppose we have n meta-models $\{m_i(x_i, h_i)\}_{i=1}^{n}$, where x_i is the visible variable, h_i is the hidden

variable and $m_i(x_i, h_i)$ is their joint distribution. These meta-models can belong to different families (e.g. RBMs and VAEs (Variational Autoencoders) (Kingma & Welling, 2013)), as long as they have hidden variables and support likelihood evaluation. We replace x_i with h_{i-1} and connect meta-models in a top-down style to construct a boosted model as

$$p_k(x, h_1, \ldots, h_k) := m_k(h_{k-1}, h_k) \prod_{i=1}^{k-1} m_i(h_{i-1}|h_i), \tag{3}$$

where $h_0 = x$ and $1 \leq k \leq n$. This formulation avoids the troublesome partition function and we can sample from the boosted model in a top-down style using simple ancestral sampling.

The boosted model allows us to generate samples hereby. Then, we build the approximation of the posterior distribution, which allows us to do inference. We connect meta-models in a bottom-up style to construct the approximation of the posterior distribution as

$$q_k(h_1, \ldots, h_k|x) := \prod_{i=1}^{k} m_i(h_i|h_{i-1}), \tag{4}$$

where $h_0 = x$ and $1 \leq k \leq n$. Since $q_j(h_1, \ldots, h_j|x)$ is exactly the marginal distribution of $q_k(h_1, \ldots, h_k|x)$ when $j < k$, we can omit the subscript, thereby writing q_k as q. The approximation of the posterior distribution makes an assumption of conditional independence: $h_i \perp h_1, \ldots, h_{i-2}|h_{i-1}$ $(3 \leq i \leq k)$, thereby leading to the advantage that we don't need to re-infer the whole boosted model after incorporating a new meta-model m_k: we only need to infer h_k from $m_k(h_k|h_{k-1})$ conditioned on h_{k-1} inferred previously.

2.3 Decomposable Variational Lower Bound

Supposing D is the training set, we give a decomposable variational lower bound L_k to the marginal likelihood $\mathbf{E}_D[\log p_k(x)]$ of the boosted model p_k, as illustrated in Theorem 1.

Theorem 1. *Let $\{m_i(h_{i-1}, h_i)\}_{i=1}^n$ be n meta-models, $p_k(x, h_1, \ldots, h_k)$ be the boosted model constructed from $\{m_i\}_{i=1}^k$, and $q(h_1, \ldots, h_{k-1}|x)$ be the approximate posterior constructed from $\{m_i\}_{i=1}^{k-1}$, then we have:*

$$\begin{aligned} \mathbf{E}_D[\log p_k(x)] - L_k(m_1, \ldots, m_k) &:= \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)} \left[\log \frac{p_k(x, h_1, \ldots, h_{k-1})}{q(h_1, \ldots, h_{k-1}|x)} \right] \\ &= \sum_{i=1}^{k} l_i(m_1, \ldots, m_i), \end{aligned} \tag{5}$$

where $l_1(m_1) = \mathbf{E}_D[\log m_1(x)]$ and

$$\begin{aligned} l_i(m_1, \ldots, m_i) = &\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{i-1}|x)}[\log m_i(h_{i-1})] \\ &- \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{i-1}|x)}[\log m_{i-1}(h_{i-1})] \quad (2 \leq i \leq k). \end{aligned} \tag{6}$$

Proof: see Appendix A.

The lower bound L_k is decomposed to k terms l_i $(1 \leq i \leq k)$, where L_k is only related to $m_1, \ldots, m_k$ and l_i is only related to $m_1, \ldots, m_i$. Specifically, l_1 is the marginal likelihood of the first meta-model m_1. l_i $(2 \leq i \leq k)$ is the difference between the marginal likelihood of the observable variable of m_i and the hidden variable of m_{i-1}.

When $k = 1$, there is only one meta-model and the lower bound is exactly equal to the marginal likelihood of the boosted model. So the lower bound is tight when $k = 1$. Based on the initially tight lower bound, we can further promote it by optimizing these decomposed terms sequentially, yielding the greedy layer-wise training algorithm, as discussed in subsection 2.4.

2.4 The Greedy Layer-Wise Training Algorithm

The difference between $L_k(m_1, \ldots, m_k)$ and $L_{k-1}(m_1, \ldots, m_{k-1})$ is

$$\begin{aligned} l_k(m_1, \ldots, m_k) = &\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})] \\ &- \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_{k-1}(h_{k-1})]. \end{aligned} \tag{7}$$

Algorithm 1 Cascaded Boosting

Input: dataset D; meta-models $\{m_i\}_{i=1}^{n}$
Let $p_D = \sum_{x^{(i)} \in D} \delta(x - x^{(i)})/|D|$ be the empirical distribution of D
Train m_1 by maximizing $\mathbf{E}_D[\log m_1(x)]$
$k = 2$
while $k \leq n$ **do**
  Fix $\{m_i\}_{i=1}^{k-1}$ and let $q(h_1, \ldots, h_{k-1}|x) = \prod_{i=1}^{k-1} m_i(h_i|h_{i-1})$
  Train m_k by maximizing $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})]$
  $k = k + 1$
end while
Let $p_n(x, h_1, \ldots, h_n) = m_n(h_{n-1}, h_n) \prod_{i=1}^{n-1} m_i(h_{i-1}|h_i)$
return $p_n(x, h_1, \ldots, h_n)$

To ensure the lower bound grows with m_k incorporated, we only need to ensure l_k is positive. When we train the meta-model m_k, we first fix rest meta-models $\{m_i\}_{i=1}^{k-1}$, so that $q(h_1, \ldots, h_{k-1}|x)$ is fixed and $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_{k-1}(h_{k-1})]$ is constant, and then train m_k by maximizing $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})]$. As a result, we can train each meta-model separately and greedily, as outlined in Alg. 1.

When $\{m_i\}_{i=2}^{n}$ are arbitrarily powerful learners, we can derive the non-decreasing property of the decomposable lower bound, as given in Theorem 2.

Theorem 2. *When $\{m_i\}_{i=2}^{n}$ are arbitrarily powerful learners (i.e., m_i is able to model any distribution), we have $L_1 \leq L_2 \leq \cdots \leq L_n$ during the greedy layer-wise training.*

Proof. During the kth ($2 \leq k \leq n$) round of Alg. 1, $\{m_i\}_{i=1}^{k-1}$ are fixed, so $q(h_1, \ldots, h_{k-1}|x)$ is fixed and $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_{k-1}(h_{k-1})]$ is constant. After training m_k, we have

$$\begin{aligned} \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})] = \max_p \; & \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log p(h_{k-1})] \\ & \mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_{k-1}(h_{k-1})]. \end{aligned} \tag{8}$$

Thus, $l_k(m_1, \ldots, m_k) \geq 0$ and $L_k(m_1, \ldots, m_k) \geq L_{k-1}(m_1, \ldots, m_{k-1})$. □

In practice, $l_k(m_1, \ldots, m_k)$ is likely to be negative under the following three cases:

> m_k is not well trained. In this case, $l_k(m_1, \ldots, m_k)$ is very negative, which indicates us to tune hyperparameters of m_k and retrain this meta-model.
>
> $m_{k-1}(h_{k-1})$ is close to the marginal distribution of $p_D(x)q(h_1, \ldots, h_{k-1}|x)$. In this case, $l_k(m_1, \ldots, m_k)$ will be close to zero, and we can either keep training by incorporating more powerful meta-models or just stop training.
>
> The lower bound converges. In this case, the lower bound will stop growing even if more meta-models are incorporated, which will be further discussed in subsection 2.5

For models with $m_k(h_{k-1})$ initialized from $m_{k-1}(h_{k-1})$, such as DBNs (Hinton et al., 2006), $l_k(m_1, \ldots, m_k) = 0$ after initializing m_k. In this case, we can make sure that $l_k(m_1, \ldots, m_k) \geq 0$ after training m_k.

2.5 Convergence

It's impossible for the decomposable lower bound to grow infinitely. After training m_k, if $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})]$ is maximized, then the lower bound will stop growing even if we keep incorporating more meta-models. We call this phenomenon convergence of the boosted model, which is formally described in Theorem 3.

Theorem 3. *If $\mathbf{E}_D \mathbf{E}_{q(h_1, \ldots, h_{k-1}|x)}[\log m_k(h_{k-1})]$ is maximized after training m_k, then $\forall j \geq 2 \in [k+1, n] \setminus \mathbf{Z}$, $\forall m_{k+1}, m_{k+2}, \ldots, m_j$, $L_j(m_1, \ldots, m_j) = L_k(m_1, \ldots, m_k)$.*

Proof: see Appendix B.

It indicates that it's unnecessary to incorporate meta-models as much as possible. To help judge whether the boosted model has converged, a necessary condition is given in Theorem 4.

Theorem 4. *If* $\mathbf{E}_D \mathbf{E}_{q(h_1; \ ;h_{k-1}|x)}[\log m_k(h_{k-1})]$ *is maximized after training* m_k*, then* $c_k(m_1, \ , m_k) := |\mathbf{E}_D \mathbf{E}_{q(h_1; \ ;h_k|x)}[\log m_k(h_k)] - \mathbf{E}_{m_k(h_k)}[\log m_k(h_k)]| = 0$.

Proof: see Appendix B.

We can use c_k to help us judge whether the boosted model has converged after training m_k. For meta-models such as VAEs, $m_k(h_k)$ is the standard normal distribution and $\mathbf{E}_{m_k(h_k)}[\log m_k(h_k)]$ is analytically solvable, leading to a simple estimation of c_k.

3 Hybrid Boosting

We can further consider a hybrid boosting by integrating our cascaded boosting with the multiplicative boosting. It is not difficult to implement: we can think of the boosted model produced by our method as the meta-model for multiplicative boosting. An open problem for hybrid boosting is to determine what kind of meta-models to use and how meta-models are connected, which is closely related to the specific dataset and task. Here we introduce some strategies for this problem.

For cascaded connection, if the dataset can be divided to several categories, it is appropriate to use a GMM (Gaussian Mixture Model) (Smolensky, 1986) as the top-most meta-model. Other meta-models can be selected as VAEs (Kingma & Welling, 2013) or their variants (Burda et al., 2015; Sønderby et al., 2016). There are three reasons for this strategy: (1) the posterior of VAE is much simpler than the dataset distribution, making a GMM enough to learn the posterior; (2) the posterior of a VAE is likely to consist of several components, with each corresponding to one category, making a GMM which also consists of several components suitable; (3) Since $m_{k-1}(h_{k-1})$ is a standard Gaussian distribution when m_{k-1} is a VAE, when $m_k(h_{k-1})$ is a GMM, which covers the standard Gaussian distribution as a special case, we can make sure that Equation 7 will not be negative after training m_k.

For multiplicative connection, each meta-model should have enough learning power for the dataset, since each meta-model is required to learn the distribution of the dataset or the reweighted dataset. If any meta-model fails to learn the distribution, the performance of the boosted model will be harmed. In subsection 4.5, we give a negative example, where a VAE and a GMM are connected by multiplication and the overall performance is extremely bad.

4 Experiments

We now present experiments to verify the effectiveness of our method. We first validate that the non-decreasing property of the decomposable lower bound holds in practice. Next, we give results of boosting advanced models to show that our method can be used as a technique to further promote the performance of state-of-the-art models. Then, we compare our method with naively increasing model capacity. Finally, we make a comparison between different generative boosting methods.

4.1 Setup

We do experiments on static binarized mnist (LeCun & Cortes, 2010), which contains 60000 training data and 10000 testing data, as well as the more complex celebA dataset (Liu et al., 2015), which contains 202599 face images, with each first resized to 64 64. The meta-models we use include RBMs (Smolensky, 1986), GMMs (Reynolds, 2015), VAEs (Kingma & Welling, 2013), ConvVAEs (i.e., VAEs with convolutional layers), IWAEs (Burda et al., 2015), and LVAEs (Sønderby et al., 2016), with their architectures given in Appendix C. The marginal likelihoods of RBMs are estimated using importance sampling, and the marginal likelihoods of VAEs and their variants are estimated using the variational lower bound (Kingma & Welling, 2013). All experiments are conducted on one 2.60GHz CPU and one GeForce GTX TITAN X GPU.

4.2 Validating the Non-decreasing Property of Decomposable Lower Bound

The non-decreasing property (Theorem 2) of decomposable lower bound (Equation 5) is the theoretical guarantee of the greedy layer-wise training algorithm (subsection 2.4). We validate that the non-decreasing property also holds in practice by using RBMs and VAEs as meta-models.

We evaluate the decomposable lower bound on 4 combinations of RBMs and VAEs on static binarized mnist. Since the stochastic variables in RBMs are discrete, we put RBMs at bottom and put VAEs at top. For each combination, we evaluate the lower bound (Equation 5) at different k ($1 \quad k \quad 6$) on both training and testing dataset. As shown in Figure 1, both the training and testing curves of the decomposable lower bound present the non-decreasing property. We also notice a slight drop at the end of these curves when incorporating VAEs, which can be explained by the convergence (subsection 2.5): if $\mathbf{E}_D \mathbf{E}_{q(h_1; \quad ;h_k \ _1 jx)} [\log m_k(h_k \ _1)]$ is maximized after training m_k, then the lower bound will stop growing even if we keep incorporating more meta-models.

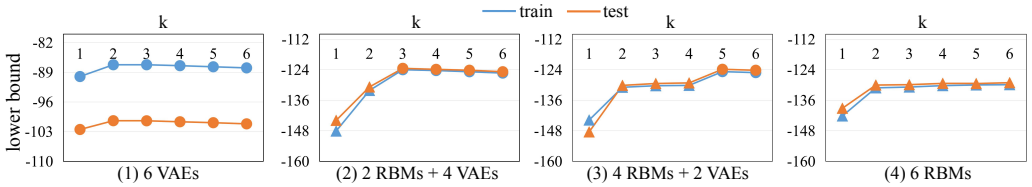


Figure 1: The lower bound (Equation 5) on different combinations of meta-models. The triangular and circular markers correspond to RBMs and VAEs respectively. **(1):** All meta-models are VAEs. After incorporating two VAEs, the lower bound becomes stable. **(2):** The first two meta-models are RBMs and the rest are VAEs. The second RBM and the first VAE greatly promote the lower bound. **(3):** The first four meta-models are RBMs and the rest are VAEs. The lower bound grows as the first two RBMs are incorporated, while the incorporation of next two RBMs doesn't help promote the lower bound. We further improve the lower bound by adding two VAEs. **(4):** All meta-models are RBMs. After incorporating two RBMs, the lower bound becomes stable.

Besides, the quality of generated samples of the boosted model also has a non-decreasing property, which is consistent with the non-decreasing property of the decomposable lower bound, as shown in Table 1. Furthermore, we can get some evidence about the convergence of the boosted model from c_k (Theorem 4). We see that c_k is the smallest when $k = 3$, which indicates that the boosted model is likely to converge after incorporating 3 VAEs and the last VAE is redundant.

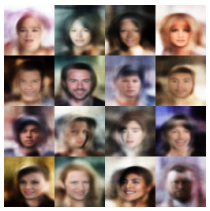
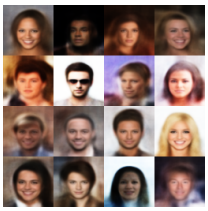
k	1	2	3	4
mnist				
lower bound (train / test)	-90.24 / -102.40	-87.44 / -100.24	-87.45 / -100.36	-87.68 / -100.59
c_k / 10^{-3} (train)	518.0	7.9	0.7	26.3
celebA				
lower bound	-6273.92	-6267.30	-6268.39	-6270.33
c_k / 10^{-3}	708.6	48.7	19.6	88.6

Table 1: Samples generated from boosted models consisting of k VAEs ($1 \quad k \quad 4$). The lower bounds (Equation 5) of these boosted models are also given. c_k (Theorem 4) is an indicator to help judge whether a boosted model has converged, where a small one supports convergence. For both mnist and celebA, the quality of generated samples has a non-decreasing property. Besides, c_k is the smallest when $k = 3$, which indicates that the boosted model is likely to converge after incorporating 3 VAEs and the last VAE is redundant.

4.3 BOOSTING ADVANCED MODELS

We show that our cascaded boosting can be used as a technique to further promote the performance of state-of-the-art models. We choose ConvVAE (i.e., VAE with convolutional layers), LVAE (Sønderby et al., 2016), IWAE (Burda et al., 2015) as advanced models, which represent current state-of-art methods. We use one advanced model and one GMM (Reynolds, 2015) to construct a boosted model, with the advanced model at the bottom and the GMM at the top. The result is given in Table 2. We see that the lower bound of each advanced model is further promoted by incorporating a GMM, at the cost of a few seconds.

	$\log p(x)$	extra time cost / s
ConvVAE	-88.41	
ConvVAE + GMM	**-87.42**	+7.85
LVAE, 2-layer	-95.73	
LVAE, 2-layer + GMM	**-95.50**	+11.76
IWAE, k=5	-81.58	
IWAE, k=5 + GMM	**-80.38**	+9.41
IWAE, k=10	-80.56	
IWAE, k=10 + GMM	**-79.20**	+8.39

Table 2: Test set performance on mnist. The density $\log p(x)$ is estimated using Equation 5. LVAE has 2 stochastic hidden layers. The number of importance weighted samples (k) for IWAE is 5 and 10. The number of components in GMM is set to 10. The extra time cost is the time cost for incorporating an extra GMM.

The performance improvement by incorporating a GMM is theoretically guaranteed: since $m_1(h_1)$ is a standard Gaussian distribution in above four cases considered in Table 2, when $m_2(h_1)$ is a GMM, which covers the standard Gaussian distribution as a special case, we can ensure that l_2 (Equation 7) will not be negative after training m_2. Besides, the dimension of hidden variable h_1 is much smaller than the dimension of observable variable h_0 for VAEs and their variants, and thus the training of m_2 only requires very little time.

4.4 COMPARISON WITH NAIVELY INCREASING MODEL CAPACITY

We compare our cascaded boosting with the method of naively increasing model capacity. The conventional method of increasing model capacity is either to add more deterministic hidden layers or to increase the dimension of deterministic hidden layers, so we compare our boosted model (Boosted VAEs) with a deeper model (Deeper VAE) and a wider model (Wider VAE). The Deeper VAE has ten 500-dimensional deterministic hidden layers; the Wider VAE has two 2500-dimensional deterministic hidden layers; the Boosted VAEs is composed of 5 base VAEs, each of them has two 500-dimensional deterministic hidden layers. As a result, all the three models above have 5000 deterministic hidden units. Figure 2 shows the results. Wider VAE has the highest lower bound, but its generated digits are usually undistinguishable. Meanwhile, the Deeper VAE is able to generate distinguishable digits, but some digits are rather blurred and its lower bound is the lowest one. Only the digits generated by Boosted VAEs are both distinguishable and sharp.

samples				
lower bound	-102.651	-140.54	-87.74	-100.91
methods	Base VAE (2 hiddens; 500-dimensional)	Deeper VAE (10 hiddens; 500-dimensional)	Wider VAE (2 hiddens; 2500-dimensional)	Boosted VAEs (5 Base VAEs)

Figure 2: Comparison between our cascaded boosting and the method of naively increasing model capacity. The lower bounds are evaluated on mnist test dataset.

Since straightforwardly increasing the model capacity is likely to cause serious challenges, such as vanishing gradient problem (Hochreiter et al., 2001) and exploding gradient problem (Grosse, 2017), it often fails to achieve the desired results on improving models' learning power. Our boosting method avoids these challenges by leveraging the greedy layer-wise training algorithm.

4.5 COMPARISON BETWEEN DIFFERENT GENERATIVE BOOSTING METHODS

We make a comparison between our cascaded boosting, multiplicative boosting and hybrid boosting. The result is given in Table 3. The hybrid boosting produces the strongest models, but the time cost of density estimation and sampling is high, due to the troublesome partition function. Our cascaded boosting allows quick density estimation and sampling, but its boosted models are not as strong as the hybrid boosting. It is also worth note that the multiplicative connection of one VAE and one GMM produces a bad model, since the learning power of a GMM is too weak for directly learning the distribution of mnist dataset and the training time of a GMM is long for high dimensional data.

		$\log p(x)$	time cost / s		
			train	density estimate	sample
cascaded	VAE+VAE	-99.53	223.85	0.42	0.13
	VAE+GMM	-98.13	116.33	0.14	0.12
multiplicative	VAE*k*VAE	-95.72	225.21	50.91	543.82
	VAE*k*GMM	-506.76	2471.60	130.65	480.95
hybrid	(VAE+GMM)*k*VAE	-94.28	225.23	125.20	1681.77
	(VAE+GMM)*k*(VAE+GMM)	-93.94	226.86	147.82	2612.59

Table 3: Comparison between different boosting methods on mnist. The '+' represents the cascaded connection and the '*k*' represents the multiplicative connection. The density $\log p(x)$ is estimated on the test set, using Equation 5 for cascaded connection and importance sampling for multiplicative connection respectively. The sampling time is the time cost for sampling 10000 samples.

5 RELATED WORK

Deep Belief Networks. Our work is inspired by DBNs (Hinton et al., 2006). A DBN has a multi-layer structure, whose basic components are RBMs (Smolensky, 1986). During training, each RBM is learned separately, and stacked to the top of current structure. It is a classical example of our cascaded boosting, since a group of RBMs are cascaded to produce a stronger model. Our decomposable variational lower bound reveals the principle behind the training algorithm of DBNs: since $m_k(h_{k-1})$ is initialized from $m_{k-1}(h_{k-1})$ for DBNs, $l_k = 0$ (Equation 7) after the initialization. After training m_k by maximizing $\mathbf{E}_D \mathbf{E}_{q(h_1; \ldots; h_{k-1}|x)} [\log m_k(h_{k-1})]$, we can make sure that $l_k \geq 0$, assuring the non-decreasing property of the decomposable lower bound (Equation 5).

Deep Latent Gaussian Models. DLGMs (Deep Latent Gaussian Models) are deep directed graphical models with multiple layers of hidden variables (Burda et al., 2015; Rezende et al., 2014). The distribution of hidden variables in layer k conditioned on hidden variables in layer $k+1$ is a Gaussian distribution. Rezende et al. (2014) introduce an approximate posterior distribution which factorises across layers. Burda et al. (2015) introduce an approximate posterior distribution which is a directed chain. Our work reveals that the variational lower bound of Burda et al. (2015) can be further decomposed and optimized greedily and layer-wise.

Other methods of boosting generative models. Methods of boosting generative models have been explored. Previous work can be divided into two categories: *sum-of-experts* (Figueiredo & Jain, 2002; Rosset & Segal, 2003; Tolstikhin et al., 2017), which takes the arithmetic average of meta-models' distributions, and *product-of-experts* (Hinton, 2002; Grover & Ermon, 2018), which takes the geometric average of meta-models' distributions.

6 CONCLUSION

We propose a framework for boosting generative models by cascading meta-models. Any hidden variable meta-model can be incorporated, as long as it supports likelihood evaluation. The decomposable lower bound allows us to train meta-models separately and greedily. Our cascaded boosting can be integrated with the multiplicative boosting. In our experiments, we first validate that the non-decreasing property of the decomposable variational lower bound (Equation 5) holds in practice, and next further promote the performance of some advanced models, which represent state-of-the-art methods. Then, we show that our cascaded boosting has better performance of improving models' learning power, compared with naively increasing model capacity. Finally, we compare different generative boosting methods, validating the ability of the hybrid boosting in further improving learning power of generative models.

REFERENCES

M Ehsan Abbasnejad, Anthony Dick, and Anton van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 781–790. IEEE, 2017.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pp. 153–160, 2007.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Mario A. T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (3):381–396, 2002.

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*, pp. 2672–2680, 2014.

Roger Grosse. Lecture 15: Exploding and vanishing gradients. *University of Toronto Computer Science*, 2017.

Aditya Grover and Stefano Ermon. Boosted generative models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

Geoffrey E Hinton. Products of experts. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470)*, volume 1, pp. 1–6. IET, 1999.

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

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2013.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Chongxuan Li, Jun Zhu, and Bo Zhang. Max-margin deep generative models for (semi-) supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 40(11):2762–2775, 2018.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pp. 827–832, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Saharon Rosset and Eran Segal. Boosting density estimation. In *Advances in neural information processing systems*, pp. 657–664, 2003.

Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons, 2016.

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE, 1986.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pp. 3738–3746, 2016.

Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. Adagan: Boosting generative models. In *Advances in neural information processing systems*, pp. 5424–5433, 2017.

Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1):129–151, 1996.

A PROOF OF THEOREM 1

Proof. Using $q(h_1, \quad , h_{k\ 1}|x)$ as the approximate posterior, we have a variational lower bound $L_k(m_1, \quad , m_k)$ for $\log p_k(x)$:

$$\log p_k(x) \quad L_k(m_1, \quad , m_k) := \mathbf{E}_{q(h_1; \quad ; h_{k\ 1}|x)} \left[\log \frac{p_k(x, h_1, \quad , h_{k\ 1})}{q(h_1, \quad , h_{k\ 1}|x)} \right],$$

with

$$\frac{p_k(x, h_1, \quad , h_{k\ 1})}{q(h_1, \quad , h_{k\ 1}|x)} = \frac{m_k(h_{k\ 1}) \prod_{i=1}^{k\ 1} \frac{m_i(h_{i\ 1}, h_i)}{m_i(h_i)}}{\prod_{i=1}^{k\ 1} \frac{m_i(h_{i\ 1}, h_i)}{m_i(h_{i\ 1})}} = m_1(x) \prod_{i=1}^{k\ 1} \frac{m_{i+1}(h_i)}{m_i(h_i)}.$$

Thus, the lower bound is equal to:

$$\begin{aligned}
&\mathbf{E}_{q(h_1; \quad ; h_{k\ 1}|x)} \left\{ \log \left[m_1(x) \prod_{i=1}^{k\ 1} \frac{m_{i+1}(h_i)}{m_i(h_i)} \right] \right\} \\
=& \log m_1(x) + \sum_{i=1}^{k\ 1} \mathbf{E}_{q(h_1; \quad ; h_i|x)} \left[\log \frac{m_{i+1}(h_i)}{m_i(h_i)} \right] \\
=& \log m_1(x) + \sum_{i=1}^{k\ 1} \left\{ \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_{i+1}(h_i)] \quad \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_i(h_i)] \right\}.
\end{aligned}$$

Thus,

$$\log p_k(x) \quad \log m_1(x) + \sum_{i=1}^{k\ 1} \left\{ \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_{i+1}(h_i)] \quad \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_i(h_i)] \right\}.$$

Take the expection with respect to dataset D, we have

$$\begin{aligned}
&\mathbf{E}_D [\log p_k(x)] \\
&\mathbf{E}_D [\log m_1(x)] + \sum_{i=1}^{k\ 1} \left\{ \mathbf{E}_D \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_{i+1}(h_i)] \quad \mathbf{E}_D \mathbf{E}_{q(h_1; \quad ; h_i|x)} [\log m_i(h_i)] \right\} \\
=& \sum_{i=1}^{k} l_k(h_1, \quad , h_k).
\end{aligned}$$

□

B PROOF OF THEOREM 3 AND THEOREM 4

Proof. Let $q_i(x, h_1, \quad , h_i) := p_D(x) q(h_1, \quad , h_i|x)$ $(1 \quad i \quad n)$, where n is the number of meta-models. Since $q_i(x, h_1, \quad , h_i)$ is exactly the marginal distribution of $q_j(x, h_1, \quad , h_j)$ when $1 \quad i < j \quad n$, we can omit the subscript, thereby writing q_i as q.

When $\mathbf{E}_D \mathbf{E}_{q(h_1; \quad ; h_{k\ 1}|x)} [\log m_k(h_{k\ 1})] = \mathbf{E}_{q(h_{k\ 1})} [\log m_k(h_{k\ 1})]$ is maximized after training m_k, we have $m_k(h_{k\ 1}) = q(h_{k\ 1})$ *a.e.*.

For any $j \in [k+1, n] \setminus \mathbf{Z}$ and any $m_{k+1}, m_{k+2}, \quad , m_j$, given i $(k+1 \quad i \quad j)$, we have

$$\begin{aligned}
l_i(m_1, \quad , m_i) =& \mathbf{E}_D \mathbf{E}_{q(h_1; \quad ; h_{i\ 1}|x)} [\log m_i(h_{i\ 1})] \quad \mathbf{E}_D \mathbf{E}_{q(h_1; \quad ; h_{i\ 1}|x)} [\log m_{i\ 1}(h_{i\ 1})] \\
=& \mathbf{E}_{q(x, h_1; \quad ; h_{i\ 1})} [\log m_i(h_{i\ 1})] \quad \mathbf{E}_{q(x, h_1; \quad ; h_{i\ 1})} [\log m_{i\ 1}(h_{i\ 1})] \\
=& \mathbf{E}_{q(h_{k\ 1})} \mathbf{E}_{q(h_k; \quad ; h_{i\ 1}, h_{k\ 1})} [\log m_i(h_{i\ 1})] \\
& \mathbf{E}_{q(h_{k\ 1})} \mathbf{E}_{q(h_k; \quad ; h_{i\ 1}, h_{k\ 1})} [\log m_{i\ 1}(h_{i\ 1})] \\
=& \mathbf{E}_{m_k(h_{k\ 1})} \mathbf{E}_{q(h_k; \quad ; h_{i\ 1}, h_{k\ 1})} [\log m_i(h_{i\ 1})] \\
& \mathbf{E}_{m_k(h_{k\ 1})} \mathbf{E}_{q(h_k; \quad ; h_{i\ 1}, h_{k\ 1})} [\log m_{i\ 1}(h_{i\ 1})].
\end{aligned}$$

Besides, we have

$$p_j(h_{k\ 1}, \quad , h_j) = m_j(h_{j\ 1}, h_j) \prod_{i=k}^{j\ 1} m_i(h_{i\ 1} j h_i)$$

and

$$q(h_k, \quad , h_{j\ 1} j h_{k\ 1}) = \prod_{i=k}^{j\ 1} m_i(h_i j h_{i\ 1}).$$

Let $q(h_k, \quad , h_{j\ 1} j h_{k\ 1})$ be the approximate posterior of $p_j(h_{k\ 1}, \quad , h_{j\ 1})$, according to Theorem 1, we have

$$\begin{aligned}
&\mathbf{E}_{m_k(h_{k\ 1})}[\log p_j(h_{k\ 1})] \quad \mathbf{E}_{m_k(h_{k\ 1})}[\log m_k(h_{k\ 1})] \\
&\qquad + \sum_{i=k+1}^{j} \left\{ \begin{array}{l} \mathbf{E}_{m_k(h_{k\ 1})} \mathbf{E}_{q(h_{k\ \ ;h_{i\ 1}} j h_{k\ 1})}[\log m_i(h_{i\ 1})] \\ \quad \mathbf{E}_{m_k(h_{k\ 1})} \mathbf{E}_{q(h_{k\ \ ;h_{i\ 1}} j h_{k\ 1})}[\log m_{i\ 1}(h_{i\ 1})] \end{array} \right\} \\
&= \mathbf{E}_{m_k(h_{k\ 1})}[\log m_k(h_{k\ 1})] + \sum_{i=k+1}^{j} l_i(m_1, \quad , m_i)
\end{aligned}$$

Since

$$\mathbf{E}_{m_k(h_{k\ 1})}[\log p_j(h_{k\ 1})] \quad \mathbf{E}_{m_k(h_{k\ 1})}[\log m_k(h_{k\ 1})],$$

we have $\sum_{i=k+1}^{j} l_i(m_1, \quad , m_i) \quad 0$, and thereby $L_j(m_1, \quad , m_j) \quad L_k(m_1, \quad , m_k)$.

Finally, we have

$$\begin{aligned}
\mathbf{E}_D \mathbf{E}_{q(h_1; \quad ;h_k j x)}[\log m_k(h_k)] &= \mathbf{E}_{q(x;h_1; \quad ;h_k)}[\log m_k(h_k)] \\
&= \mathbf{E}_{q(h_{k\ 1})} \mathbf{E}_{q(h_k j h_{k\ 1})}[\log m_k(h_k)] \\
&= \mathbf{E}_{m_k(h_{k\ 1})} \mathbf{E}_{m_k(h_k j h_{k\ 1})}[\log m_k(h_k)] \\
&= \mathbf{E}_{m_k(h_k)}[\log m_k(h_k)].
\end{aligned}$$

Thus, $c_k(m_1, \quad , m_k) := j \mathbf{E}_D \mathbf{E}_{q(h_1; \quad ;h_k j x)}[\log m_k(h_k)] \quad \mathbf{E}_{m_k(h_k)}[\log m_k(h_k)] j = 0.$ □

C ARCHITECTURES OF META-MODELS

The architectures of VAEs, ConvVAEs, IWAEs and LVAEs are given in this part.

C.1 ARCHITECTURES OF VAES

All VAEs have two deterministic hidden layers for both generation, and inference and we add batch normalization layers (Ioffe & Szegedy, 2015; Sønderby et al., 2016) after deterministic hidden layers. The dimension of deterministic hidden layers is set to 500 and 2500, and the dimension of stochastic hidden variables is set to 20 and 100, for experiments on mnist and celebA respectively.

C.2 ARCHITECTURES OF CONVVAES

The ConvVAE has one 500-dimensional deterministic hidden layer and one 50-dimensional stochastic hidden variable, with four additional convolutional layers LeCun et al. (1998). All convolutional layers have a kernel size of 4 4 and a stride of 2. Their channels are 32, 64, 128 and 256 respectively. We add batch normalization layers after deterministic hidden layers.

C.3 ARCHITECTURES OF IWAES

The IWAE has two 500-dimensional deterministic hidden layers and one 50-dimensional stochastic hidden variable. The number of importance sampling is set to 5 and 10.

C.4 ARCHITECTURES OF LVAES

The LVAE has four 1000-dimensional deterministic hidden layers and two 30-dimensional stochastic hidden variables. We add batch normalization layers after deterministic hidden layers.