NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

as a manuscript

Neklyudov Kirill

BAYESIAN APPROACH IN DEEP LEARNING: REFINEMENT OF DISCRIMINATIVE AND GENERATIVE MODELS

PhD Dissertation Summary
for the purpose of obtaining academic
degree Doctor of Philosophy in Computer
Science

Moscow — 2020

The PhD Dissertation was prepared at National Research University Higher School of Economics.

Academic Supervisor: Dmitry P. Vetrov, Candidate of Sciences, Research Professor, National Research University Higher School of Economics.

Topic of the thesis

This work employs the formalism of Bayesian statistics for refinement of existing deep learning models in various ways. Based on the doubly stochastic variational inference [1], this work proposes two probabilistic models for deep discriminative networks. The first model allows for structured sparsification of convolutional neural networks (CNNs) and, hence, their acceleration. The second allows for better uncertainty estimation in conventional CNN architectures. Further, we focus on deep generative models. Treating the problem of sampling from the MCMC perspective, current work proposes an algorithm that improves the performance of generative adversarial networks (GANs) [2]. Providing an asymptotic analysis of the proposed scheme, we introduce the implicit Metropolis-Hastings algorithm. It can be seen as an adaptation of the conventional Metropolis-Hastings algorithm [3] to the case of implicit proposal model and empirical target distribution.

Actuality of the work.

Machine learning is the scientific approach that allows for building models (algorithms) from data. Machine learning algorithms find their practical success in tasks where a solution cannot be rigorously formalized or explicitly programmed. Ones of the main areas of application are computer vision, language modeling, speech recognition. Here we provide informal description of the machine learning concept. For rigorous formulation, as well as, instances of algorithms and applications, we refer the reader to [4; 5].

The classical dichotomy in machine learning is between *supervised* learning and *unsupervised* learning. The goal of supervised learning is to build a function from objects (usually described by real-valued vectors called *features*) to *labels* (also a real-valued vectors in the most general setting). We further refer to this function as *model*. To build such a model, one usually formulates it in a parametric way, i.e. defines a function (analytical, as instance) or algorithm that outputs the *prediction* of a label given input features and all parameters values. Further, among all possible values of parameters, we select a configuration that is the most suitable for our task. The process of selection is called *training*. In order to perform training, one usually defines a *loss function* on the joint space of predictions and labels. The loss function defines a measure of prediction ”correctness”, as instance, it equals to zero if the prediction is correct and grows with an amount of its error. Given such function, and the *training* empirical data (subset of objects with labels from the general set of objects), the process of training can be formulated as an optimization problem on the space of parameters. That is, we need to find such configuration of parameters that yields a minimum of the loss function on the training set. Usually this configuration is obtained by gradient-based optimization methods.

The general goal of the unsupervised learning is to infer an underlying structure of the given data. Saying differently, for every object (with feature description) in the data we need to infer a latent variable that somehow describes this object. However, a precise notion of the structure (space of latent variables) significantly depends on the nature of the data, and intended use of the latent variables. To provide the reader with more intuition, we briefly describe several instances of unsupervised learning tasks. Clustering task is one of the most common tasks in unsupervised learning, and can be informally described as ”inferring labels without labels in the training data”. That is, given a training set, we need to assign a label to every object, where labels are discrete variables, and objects with the same label form a cluster. Sometimes we also need to define the space of labels (its cardinality and interior relations). Another popular approach in unsupervised learning is auto-encoders. They operate by encoding an object into a latent space and then decoding the obtained latent representation back to the original space, trying to exactly restore the original object. The latent space usually have some desired property. For instance, an auto-encoder can perform compression, by mapping original objects to a low-dimensional latent space. In the context of the current work, generative models are the most important instances of unsupervised learning. In generative models, we assume that the observed empirical data (training dataset) is the set of samples from some unknown distribution. Under this assumption, the goal is to recover this unknown distribution by building a model that can sample from this distribution. The most popular approaches of learning such models are contrastive divergence [6], variational auto-encoders [7; 8], generative-adversarial networks [2], normalizing flows [9].

The crucial point in developing machine learning methods and their applications is the choice of feature representation for objects. Ones of the most complex objects to represent are images, texts, sounds, due to their high-dimensionality and complex underlying structure. For this kind of data, before the watershed paper [10], representations were built using an expert knowledge of the corresponding area. As instance, in computer vision, SIFT [11] and HOG [12] features were conventional representations of images. Deep learning [13; 14] have automated the process of constructing feature representations. The main idea of this approach is to build a model with multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning models greatly rely on artificial neural networks, which are also known to be universal approximators. However, the ability to approximate any function is not sufficient for construction of good model. It also should depicts the nature of the data, for instance, CNNs [15] are shift-invariant, what make them well-suited for images, and LSTM [16] prevents gradient saturation for long sequences, what make them well-suited for texts.

In this work we greatly rely on the formalism of Bayesian reasoning [17]. To provide an instance of Bayesian model, let us consider a supervised learning task. Then, as well as in general formulation, we have a model that defines a *likelihood*, e.g. the probability of correct label given features and parameter values. Besides the model, there is also a *prior* distribution on the parameters that depicts our background knowledge about the task. Given the prior and the likelihood, during the training, we want to find not a single configuration of the parameters, but a distribution in the space of the parameters (called *posterior*). This approach provides a range of benefits, and further we list some of them. First of all, if we can infer the posterior accurately, then we will have large ensemble of models (weights are proportional to the density of posterior) that can improve a performance compared to single model. Secondly, we can incorporate our task-specific knowledge into the model using the prior distribution. Thirdly, we can perform an incremental learning by depicting a knowledge about previously seen data in the prior distribution. To highlight the main aspects of Bayesian inference we provide a relation of it with the maximum entropy principle [18], which is the most general approach of inferring a model from data. Such relation is precisely described by Jaynes [17]:
"Bayesian and maximum entropy methods differ in another respect. Both procedures yield the optimal inferences from the information that went into them, but we may choose a model for Bayesian analysis; this amounts to expressing some prior knowledge — or some working hypothesis — about the phenomenon being observed. Usually, such hypotheses extend beyond what is directly observable in the data, and in that sense we might say that Bayesian methods are — or at least may be — speculative. If the extra hypotheses are true, then we expect that the Bayesian results will improve on maximum entropy; if they are false, the Bayesian inferences will likely be worse."

That is, to develop a powerful algorithm (discriminative or generative) we need to choose a model that take into account a domain-specific knowledge. In modern machine learning such models usually employ deep learning approach, e.g. CNNs for images and LSTM for sequential data. The usage of deep learning models in Bayesian reasoning coined the name of the field — Bayesian deep learning. The central technique in the Bayesian deep learning is the doubly-stochastic variational inference [1]. A seminal work of this field is Variational Dropout [19], which consider a CNN as a likelihood model in Bayesian reasoning and interprets the dropout layer as a variational approximation. Further, it was demonstrated that the log-uniform prior distribution induces sparsity in deep neural networks [20]. However, such sparsity cannot be used for acceleration of deep neural networks since it has no structure. In the thesis, we propose a model that takes the architecture of a CNN into account and induces structured sparsity, allowing for acceleration of CNNs. As well as the dropout layer, another touchpoint between conventional deep learning algorithms and Bayesian deep learning is the batch normalization layer. Considering the selection of batch as a noise source, we propose a probabilistic model for the batch normalization layer. Then, using the proposed model, we improve the performance of several models in terms of uncertainty estimation.

Bayesian deep learning significantly relies on the fact that modern deep learning models efficiently exploit a domain-specific knowledge of the task. Another way to use the deep learning models in Bayesian methods is to improve the approximate inference stage when the exact inference is infeasible and rich approximating family is needed. One of the way to perform an approximate inference is Markov Chain Monte Carlo approach, which can be used to describe the posterior distribution by samples. The choice of the MCMC algorithm is essential for such task. In practice, one usually want to obtain an algorithm which converges fast to regions of high probability and mixes rapidly between different modes. To build such an algorithm, it is reasonable to employ the rich family of deep learning models for approximation of the target distribution. One of the first instances of such algorithms are NICE-MC [21] and L2HMC [22]. In the current work, we propose an alternative approach to learning a sampler, by deriving the objective for the independent proposal in the Metropolis-Hastings algorithm. Further, we step beyond the conventional formulation of the problem and derive GANs framework using the MCMC perspective. This fact leads us to the adaptation of the Metropolis-Hastings algorithm to the case of implicit proposal and empirical target distributions. We call this adaptation the Implicit Metropolis-Hastings algorithm and provide both empirical and theoretical studies of it.

The goal of this work is improvement of modern deep learning models using the Bayesian approach. The considered improvements are performance gain and new properties of models such as sparsity and uncertainty estimation.

Key results and conclusions

The novelty of this work is that for the first time the following points are shown.

1. Bayesian probabilistic models can induce the structured sparsity in deep convolutional neural networks.
2. Batch normalization layer can be formulated as a probabilistic model that is consistent during both train and test stages.
3. Optimization of the symmetric KL-divergence leads to a better proposal distributions for the independent Metropolis-Hastings algorithm.
4. It is possible to alleviate the gap between the output distribution of an implicit generative model and target empirical distribution using an approximation of the Metropolis-Hastings acceptance test.

Theoretical and practical significance. The obtained results widen the scope of applicability of CNNs by compressing and accelerating conventional architectures. For analytical target distributions (given as unnormalized density), the work proposes a method of building the computationally efficient sampler. For the implicit generative models, such as GANs and VAEs, the work proposes the filtering procedure that demonstrates consistent empirical gains in practice. Besides that, we derive a bound on the distance between the output distribution of an implicit generative model and target empirical distribution.

Methodology and research methods. This work uses the methodology of deep learning; the toolkit of probabilistic modeling; Python; NumPy, PyTorch, Theano, Lasagne frameworks.

Reliability of the obtained results is ensured by a detailed description of the methods and algorithms, proofs of theorems, as well as description of experiments and release of the source code which facilitates reproducibility.

Main provisions for the defense:

1. The algorithm for structured sparsity of convolutional neural networks.
2. Probabilistic formulation of the batch normalization layer, and the algorithm for uncertainty estimation.

3. The adaptation of the conventional Metropolis-Hastings algorithm to the case of implicit proposal and empirical target distributions.

Personal contribution into the main provisions for the defense. In main two papers (first-tier publications) results are obtained by the author, i.e. author proposed the key scientific ideas, implemented and performed the experiments, wrote the papers. The contribution of other coauthors is review of the code of the experiments, technical help with setup of experiments, discussion of the obtained results, editing of the text of the papers, problem formulation and general supervision of research.

Publications and probation of the work

The aspirant is the main author in the two main papers on the topic of the thesis.

First-tier publications.

1. *Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, Dmitry Vetrov* Structured Bayesian Pruning via Log-Normal Multiplicative Noise // Advances in Neural Information Processing Systems 30. 2017. P. 6775–6784. Rank A* conference, indexed by SCOPUS.
2. *Kirill Neklyudov, Evgenii Egorov, Dmitry Vetrov* The Implicit Metropolis-Hastings Algorithm // Advances in Neural Information Processing Systems 32. 2019. Rank A* conference, indexed by SCOPUS.

Other publications.

1. *Andrei Atanov, Arsenii Ashukha, Dmitry Molchanov, Kirill Neklyudov, Dmitry Vetrov* Uncertainty Estimation via Stochastic Batch Normalization // In International Symposium on Neural Networks, pp. 261-269. Springer, Cham, 2019.

Reports at conferences and seminars.

1. Seminar of Bayesian methods research group, Moscow, 20 May 2017. Topic: ”Group sparsity in convolutional neural networks”.
2. “Conference on Neural Information Processing Systems 2017”, main section, Los Angeles, USA, 9 December 2016. Topic: ”Structured Bayesian Pruning via Log-Normal Multiplicative Noise”.
3. Seminar of Bayesian methods research group, Moscow, 05 October 2018. Topic: ”Metropolis-Hastings View on Variational Inference and Adversarial Training”.

4. Machine Learning Seminar at Lebedev Physical Institute, 19 February 2019. Topic: ”How Neural Networks Help MCMC and How MCMC Helps Neural Networks”.
5. “International Symposium on Neural Networks”, oral presentation, Moscow, 10 July 2019. Topic: “Uncertainty Estimation via Stochastic Batch Normalization”.
6. “Conference on Neural Information Processing Systems 2019”, main section, Vancouver, Canada, 11 December 2019. Topic: ”The Implicit Metropolis-Hastings Algorithm’.

Volume and structure of the work. The thesis contains an introduction, contents of publications and a conclusion. The full volume of the thesis is 64 pages.

1 Content of the work

1.1 Structured Bayesian Pruning via Log-Normal Multiplicative Noise

The first chapter describes Structured Bayesian Pruning model, which is able to induce an arbitrary pattern of structured sparsity on neural network parameters or intermediate data tensors. It uses stochastic variational inference to tune its parameters in a Bayesian way. Moreover, the chapter describes a proper analog of sparsity-inducing log-uniform prior distribution [20; 23] that allows us to formulate a correct probabilistic model and avoid the problems that come from using an improper prior. This way we can obtain a novel Bayesian method of regularization of neural networks that results in structured sparsity. Additionally, the proposed model can be represented as a separate dropout-like layer that allows for a simple and flexible implementation with almost no computational overhead, and can be incorporated into existing neural networks.

Given a probabilistic model $p(y \mid x,\theta)$ we want to tune parameters θ of the model using training dataset $\mathcal{D} = \{(x_i,y_i)\}_{i=1}^N$. The prior knowledge about parameters θ is defined by prior distribution $p(\theta)$. Using the Bayes rule we obtain the posterior distribution $p(\theta \mid \mathcal{D}) = p(\mathcal{D} \mid \theta)p(\theta)/p(\mathcal{D})$. However, computing posterior distribution using the Bayes rule usually involves computation of intractable integrals, so we need to use approximation techniques. One of the most widely used approximation techniques is the variational inference. Under this approach the unknown distribution $p(\theta \mid \mathcal{D})$ is approximated by a parametric distribution $q_\phi(\theta)$ by minimization of the Kullback-Leibler divergence $\mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta \mid \mathcal{D}))$. Minimization of the KL divergence is equivalent to maximization of the variational lower bound $\mathcal{L}(\phi)$.

$$\mathcal{L}(\phi) = L_D(\phi) - \mathrm{KL}(q_\phi(\theta) \,\|\, p(\theta)), \tag{1}$$

$$\text{where } L_D(\phi) = \sum_{i=1}^{N} \mathbb{E}_{q_\phi(\theta)} \log p(y_i \,|\, x_i, \theta) \tag{2}$$

The proposed model can be formulated as a single dropout-like layer with an input vector $x \in \mathbb{R}^I$ that represents one object with I features, and an output vector $y \in \mathbb{R}^I$ of the same size. The input vector x is usually supposed to come from the activations of the preceding layer. The output vector y would then serve as an input vector for the following layer. We follow the general way to build dropout-like layers (3). Each input feature x_i is multiplied by a noise variable θ_i that comes from some distribution $p_{noise}(\theta)$. For example, for Binary Dropout $p_{noise}(\theta)$ would be a fully factorized Bernoulli distribution with $p_{noise}(\theta_i) = \text{Bernoulli(p)}$, and for Gaussian dropout it would be a fully-factorized Gaussian distribution with $p_{noise}(\theta_i) = \mathcal{N}(1, \alpha)$.

$$y_i = x_i \cdot \theta_i \qquad\qquad \theta \sim p_{noise}(\theta) \tag{3}$$

Further, we follow a Bayesian treatment of the variable θ. To obtain a sparse solution, one can choose the prior distribution $p(\theta)$ to be a fully-factorized truncated log-uniform distribution. For the approximation family, we can choose the truncated log-normal distribution. Then the final model of the layer can be formulated as follows.

$$y_i = x_i \cdot \theta_i \quad p(\theta_i) = \mathrm{LogU}_{[a,b]}(\theta_i) \quad q(\theta_i \,|\, \mu_i, \sigma_i) = \mathrm{LogN}_{[a,b]}(\theta_i \,|\, \mu_i, \sigma_i^2) \tag{4}$$

The experiments show that the proposed model leads to high group sparsity level and significant acceleration of convolutional neural networks with negligible accuracy drop. We demonstrate the performance of our method on LeNet and VGG-like architectures using MNIST and CIFAR-10 datasets. Results for VGG-like architectures on CIFAR-10 dataset are presented in Table 1. For each architecture, we report the number of retained neurons and filters, and obtained acceleration. We also demonstrate that optimization w.r.t. the full set of variational parameters (μ, σ) leads to improving model quality and allows us to perform sparsification in a more efficient way, as compared to tuning of only one free parameter that corresponds to the noise variance. As a nice bonus, we show that Structured Bayesian Pruning network does not overfit on randomly labeled data, that is a common weakness of non-bayesian dropout networks.

Table 1: Comparison of different structured sparsity inducing techniques (SparseVD [20]) on VGG-like architectures on CIFAR-10 dataset. StructuredBP stands for the original SBP model, and StructuredBPa stands for the SBP model with KL scaling. k is a width scale factor that determines the number of neurons or filters on each layer of the network (width$(k) = k \times$ original width)

k	Method	Error %	Units per Layer	CPU	GPU	FLOPs
1.0	Original	7.2	64 – 64 – 128 – 128 – 256 – 256 – 256 – 512 – 512 – 512 – 512 – 512 – 512 – 512	1.00×	1.00×	1.00×
	SparseVD	7.2	64 – 62 – 128 – 126 – 234 – 155 – 31 – 81 – 76 – 9 – 138 – 101 – 413 – 373	2.50×	1.69×	2.27×
(ours)	StructuredBP	7.5	64 – 62 – 128 – 126 – 234 – 155 – 31 – 79 – 73 – 9 – 59 – 73 – 56 – 27	2.71×	1.74×	2.30×
(ours)	StructuredBPa	9.0	44 – 54 – 92 – 115 – 234 – 155 – 31 – 76 – 55 – 9 – 34 – 35 – 21 – 280	3.68×	2.06×	3.16×
1.5	Original	6.8	96 – 96 – 192 – 192 – 384 – 384 – 384 – 768 – 768 – 768 – 768 – 768 – 768 – 768	1.00×	1.00×	1.00×
	SparseVD	7.0	96 – 78 – 191 – 146 – 254 – 126 – 27 – 79 – 74 – 9 – 137 – 100 – 416 – 479	3.35×	2.16×	3.27×
(ours)	StructuredBP	7.2	96 – 77 – 190 – 146 – 254 – 126 – 26 – 79 – 70 – 9 – 71 – 82 – 79 – 49	3.63×	2.17×	3.32×
(ours)	StructuredBPa	7.8	77 – 74 – 161 – 146 – 254 – 125 – 26 – 78 – 66 – 9 – 47 – 55 – 54 – 237	4.47×	2.47×	3.93×

1.2 Uncertainty Estimation via Stochastic Batch Normalization

In the second chapter, we investigate Batch Normalization technique and propose its probabilistic interpretation. Batch Normalization layer is an essential part of every deep convolutional architectures. In our work, we treat Batch Normalization as a stochastic layer and propose a way to ensemble batch-normalized networks. The straightforward technique, however, ends up with high memory and computational cost. We, therefore, propose Stochastic Batch Normalization (SBN) — an efficient and scalable approximation technique.

We consider a supervised learning problem, with a dataset $D = \{(x_i, y_i)\}_{i=1}^N$. The goal is to train the parameters θ of the predictive likelihood $p_\theta(y \,|\, x)$, modelled by a neural network. To solve this problem stochastic optimization methods with a mini-batch gradient estimator usually are used [24].

Batch Normalization attempts to preserve activations of all layers with zero mean and unit variance. In order to do that it uses the mean $\mu(\mathcal{B})$ and variance $\sigma^2(\mathcal{B})$ over the mini-batch $\mathcal{B}$ during training and accumulated statistics on the inference phase:

$$\mathrm{BN}^{\text{train}}_{\gamma,\beta}(x_i) = \frac{x_i - \mu(\mathcal{B})}{\sqrt{\sigma^2(\mathcal{B}) + \epsilon}} \cdot \gamma + \beta \qquad \mathrm{BN}^{\text{test}}_{\gamma,\beta}(x_i) = \frac{x_i - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}} \cdot \gamma + \beta \quad (5)$$

where γ, β are the trainable Batch Normalization parameters (scale and shift) and ϵ is a small constant, needed for numerical stability. Note that during training mean and variance are computed over a randomly picked batch ($\mu(\mathcal{B})$, $\sigma(\mathcal{B})$), while during testing the exponentially smoothed statistics ($\hat{\mu}$, $\hat{\sigma}^2$) are used. We further address this inconsistency by the proposed probabilistic model.

Note from (5) that forward pass through the batch-normalized network depends not only on x_i but on the entire batch $\mathcal{B}$ as well. This dependency can be reinterpreted in terms of mini-batch statistics $\mu(\mathcal{B}), \sigma(\mathcal{B})$:

$$p_\theta(y_i \,|\, x_i, \mathcal{B}_{\backslash i}) = p_\theta(y_i | x_i, \mu(\mathcal{B}), \sigma(\mathcal{B})), \tag{6}$$

where $\mathcal{B}_{\backslash i}$ is a batch without x_i. Due to the stochastic choice of mini-batches during training, for a fixed x_i $\mathcal{B}_{\backslash i}$ is a random variable, so mini-batch statistics can be treated as a random variables. The conditional distribution $p_\theta(\mu, \sigma \,|\, x_i, \mathcal{B}_{\backslash i})$ is the product of two Dirac delta functions, centered at $\mu(\mathcal{B})$ and $\sigma(\mathcal{B})$, since statistics are deterministic functions of the mini-batch, and the distribution of mean and variance given x_i is an expectation over mini-batch distribution. During inference we average the distribution $p_\theta(y|x, \mu, \sigma^2)$ over the normalization statistics:

$$p_\theta(\mu, \sigma | x_i) = \mathop{\mathbb{E}}_{\mathcal{B}_{\backslash i}} \delta_{\mu(\mathcal{B})}(\mu) \delta_{\sigma(\mathcal{B})}(\sigma) \qquad p_\theta(y|x) = \mathop{\mathbb{E}}_{p_\theta(\mu,\sigma|x)} p(y|x, \mu, \sigma) \tag{7}$$

In the work we show that during training, Batch Normalization (5) performs the unbiased one-sample MC estimation of a gradient of a lower bound to the marginal likelihood (7). Thus, such probabilistic model corresponds to Batch Normalization during training. However, on test phase Batch Normalization uses exponentially smoothed statistics $\mathbb{E}\,\mu \approx \hat{\mu}, \mathbb{E}\,\sigma \approx \hat{\sigma}$, which can be seen as a biased approximation of (7):

$$\mathop{\mathbb{E}}_{p_\theta(\mu,\sigma|x_i)} p(y_i | x_i, \mu, \sigma) \approx p_\theta(y|x, \mathbb{E}\,\mu, \mathbb{E}\,\sigma)$$

Straightforward MC averaging can be used for better unbiased estimation of (7), however, it is inefficient in practice. Indeed, to draw one sample from the distribution over statistics (7) we need to pass an entire mini-batch through the network. So, to make MC averaging for single test object, we need to perform several forward passes with different mini-batches sampled from the training data. To address this drawback we propose Stochastic Batch Normalization.

To reduct memory and computational cost of straightforward MC estimation, we propose to approximate the distribution of Batch Normalization statistics $p_\theta(\mu, \sigma \,|\, x_i)$ with a fully-factorized parametric approximation $p_\theta(\mu, \sigma \,|\, x_i) \approx r(\mu) r(\sigma)$. We parameterize $r(\mu)$ and $r(\sigma)$ in the following way:

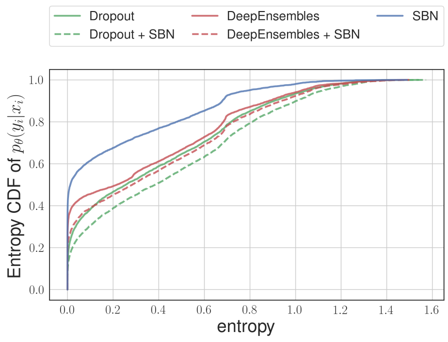
$$r(\mu) = \mathcal{N}(\mu | \mathrm{m}_\mu, \mathrm{s}_\mu^2) \qquad r(\sigma) = \mathrm{Log}\mathcal{N}(\sigma | \mathrm{m}_\sigma, \mathrm{s}_\sigma^2) \tag{8}$$

Such approximation works well in practice. In the chapter we show that it accurately fits the real marginals. Since approximation no longer depends on the training data, samples for each layer can be computed without passing the entire batch through the network and it is possible to make prediction in an efficient way. Practically, to make prediction for one test object you can duplicate object K times and pass entire batch through the network *only once*, sampling independent statistics for each copy and average the results for such "virtual batch". This procedure is K times faster in comparison to the straightforward Monte Carlo estimation.

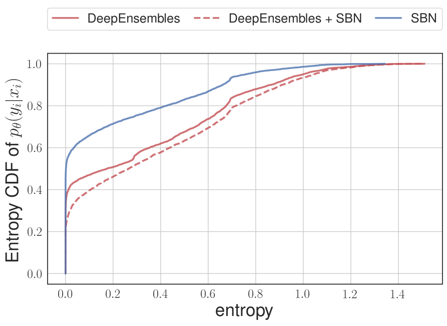
To adjust parameters $\{m_\mu, s_\mu, m_\sigma, s_\sigma\}$ we minimize the KL-divergence between distribution induced by Batch Normalization (7) and our approximation $r(\mu)r(\sigma)$ for each object:

$$D_{\mathrm{KL}}\left({}^{1}/_{N}\sum_{i=1}^{N} p_\theta(\mu, \sigma \,|\, x_i) \,||\, r(\mu)r(\sigma)\right) \longrightarrow \min_{m_\mu, s_\mu, m_\sigma, s_\sigma}$$

Since r belongs to the exponential family, this minimization problem is equal to moment matching and does not require gradients computation. In our implementation we simply use exponential smoothing to approximate the sufficient statistics of mean and variance distributions. It can be done for any pre-trained batch-normalized network.



(a) Results for VGG-11



(b) Results for ResNet-18

Figure 1: Empirical CDF of entropy for out-of-domain data. VGG-11 and ResNet-18 on five classes of CIFAR10, hidden during training. SBN corresponds to model with all Batch Normalization layers replaced by Stochastic Batch Normalization. The more to the right and the lower, the better.

To show that our method scales to deep convolutional architectures well, we perform experiments on VGG-like and ResNet architectures. We split CIFAR10 dataset into two datasets (CIFAR5), and plot the empirical CDF in Fig. 1. We trained networks on randomly chosen 5 classes and evaluated predictive uncertainty on the remaining.

1.3 Metropolis-Hastings View on Variational Inference and Adversarial Training

A significant part of MCMC methods can be considered as the Metropolis-Hastings (MH) algorithm with different proposal distributions. From this point of view, the problem of constructing a sampler can be reduced to the question — how to choose a proposal for the MH algorithm? To address this question, we propose to learn an independent sampler that maximizes the acceptance rate of the MH algorithm, which, as we demonstrate, is highly related to the conventional variational inference. For Bayesian inference, the proposed method compares favorably against alternatives to sample from the posterior distribution. Under the same approach, we step beyond the scope of classical MCMC methods and deduce the Generative Adversarial Networks (GANs) framework from scratch, treating the generator as the proposal and the discriminator as the acceptance test.

The acceptance rate of the MH algorithm is tightly connected with detailed balance. In the extreme case, when the acceptance rate achieves its maximum value, distributions $p(x')q(x \,|\, x')$ and $p(x)q(x' \,|\, x)$ must coincide (up to sets of zero measure) in the joint space of the previous point x and the proposed point x'. For such a case, we can say that the detailed balance condition holds:

$$p(x')q(x \,|\, x') = p(x)q(x' \,|\, x) \quad \forall x, x'. \tag{9}$$

It turns out that the acceptance rate defines how far distributions $p(x')q(x \,|\, x')$ and $p(x)q(x' \,|\, x)$ are, or how well the detailed balance condition is satisfied for a proposal distribution $q(\cdot \,|\, \cdot)$. We formalize this connection by introducing the following theorem.

Theorem 1. *For a random variable* $\xi = \frac{p(x')q(x \,|\, x')}{p(x)q(x' \,|\, x)}, x \sim p(x), x' \sim q(x' \,|\, x)$

$$\begin{aligned} \mathrm{AR} = \mathbb{E}_\xi \min\{1, \xi\} = 1 - \frac{1}{2}\mathbb{E}_\xi|\xi - 1| = \\ = 1 - \mathrm{TV}\Big(p(x')q(x \,|\, x') \Big\| p(x)q(x' \,|\, x)\Big), \end{aligned} \tag{10}$$

where TV *is the total variation distance.*

This reinterpretation in terms of total variation allows us to lower bound the acceptance rate via Pinsker's inequality

$$\mathrm{AR} \geq 1 - \sqrt{\frac{1}{2} \cdot \mathrm{KL}\Big(p(x')q(x \,|\, x') \Big\| p(x)q(x' \,|\, x)\Big)}. \tag{11}$$

We suggest using the acceptance rate or its lower bound as an objective for learning a proposal distribution. Note that doing so for a Markov proposal may result in a trivial solution $q(x' \mid x) = \delta(x' - x)$ that yields the maximal acceptance rate. That happens, since detailed balance and the acceptance rate does not take the autocorrelation of proposed samples into account. In this work, we enforce zero autocorrelation and exclude the trivial solution by considering independent proposals.

For independent proposals the lower bound from (11) can be rewritten in terms of symmetric KL-divergence between $p(\cdot)$ and $q(\cdot)$

$$\text{AR} \geq 1 - \sqrt{\frac{1}{2}\Big(\text{KL}(q(x)\|p(x)) + \text{KL}(p(x)\|q(x))\Big)}, \tag{12}$$

which has its global maximum at $q(x) = p(x)$; hence, at maximal acceptance rate $\text{AR} = 1$. In the work, we demonstrate that the obtained lower bound relates the proposed approach with the variational inference and GANs. For Bayesian inference, the obtained lower bound could be preferable to the acceptance rate since one can estimate it using only minibatches of data.

In this chapter, we propose algorithms for learning parameters ϕ of an independent proposal distribution $q_\phi(x)$. As objectives for optimization, we use the acceptance rate of the MH algorithm and its lower bound. For convenience, we reformulate these objectives in terms of loss functions as follows. Maximization of the acceptance rate is equivalent to minimization of the loss:

$$\mathcal{L}_{\text{AR}}(\phi) = -\text{AR} = -\mathop{\mathbb{E}}_{\substack{x \sim p(x) \\ x' \sim q_\phi(x')}} \min\left\{1, \frac{p(x')q_\phi(x)}{q_\phi(x')p(x)}\right\}. \tag{13}$$

For maximization of the lower bound, the loss function is

$$\mathcal{L}_{\text{KL}}(\phi) = \text{KL}(q_\phi(x)\|p(x)) + \text{KL}(p(x)\|q_\phi(x)) = \tag{14}$$

$$= -\mathop{\mathbb{E}}_{\substack{x \sim p(x) \\ x' \sim q_\phi(x')}} \log\left(\frac{p(x')q_\phi(x)}{q_\phi(x')p(x)}\right). \tag{15}$$

To estimate $\mathcal{L}(\phi)$, we need to evaluate the density ratio on samples from the target $x \sim p(x)$ and proposal $x' \sim q_\phi(x')$. Depending on the form in which the target distribution is given, we have different issues during the estimation of the loss function.

If the target distribution is given as an unnormalized density (we call it *the density-based setting*), we suggest using an explicit probabilistic model as

Table 2: Short discription of two settings for target distribution $p(x)$ and proposal $q(x)$.

Setting	**Density of target**	**Samples from target**	**Density of proposal**	**Density ratio**
Density-based	given	run independent MH	given	given
Sample-based	not available	given	not available	run adversarial training

a proposal to evaluate the density ratio exactly. To obtain samples from the target, in this setting we propose to run independent MH with the currently available proposal.

If the target distribution is given in the empirical form (we call it *the sample-based setting*), samples from the target and proposal distributions are available, but we cannot compute the density ratio, so we propose to approximate it via the adversarial training.

For a summary of both settings, see Table 2.

We present an empirical evaluation for both density-based and sample-based settings. In the density-based setting, the proposed algorithm compares favorably in sampling from the posterior distribution of the Bayesian logistic regression. In the sample-based setting, we demonstrate empirical gains of various GANs by sampling via the MH algorithm at the test stage.

For an illustration, we demonstrate 2d histograms of samples from mixture of six gaussians for different methods (see Fig. 2).

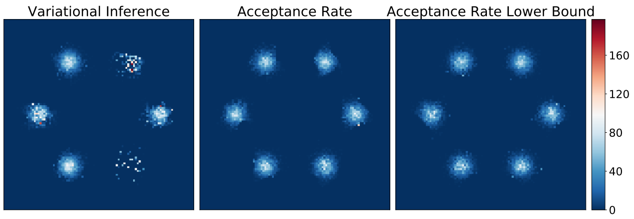


Figure 2: 2d histrograms of 25k samples from the MH algorithm with different proposals. From left to right proposals are learned by the variational inference, the acceptance rate maximization, the acceptance rate lower bound maximization.

1.4 The Implicit Metropolis-Hastings Algorithm

In the previous chapter we propose using the discriminator of a GAN to filter out unrealistic samples of the generator. Here, we generalize this idea by introducing the implicit Metropolis-Hastings algorithm. For any implicit probabilistic model and a target distribution represented by a set of samples, implicit Metropolis-Hastings operates by learning a discriminator to estimate the density-ratio and then generating a chain of samples. Since the approximation of density ratio introduces an error on every step of the chain, it is crucial to analyze the stationary distribution of such chain. For that purpose, we present a theoretical result stating that the discriminator loss upper bounds the total variation distance between the target distribution and the stationary distribution.

The Implicit Metropolis-Hastings algorithm is aimed to sample from an empirical target distribution $p(x),\ x \in \mathbb{R}^D$, while being able to sample from an implicit proposal distribution $q(x\,|\,y)$. Given a discriminator $d(x,y)$, it generates a chain of samples as described in Algorithm 1.

Algorithm 1 The implicit Metropolis-Hastings algorithm

```
input target dataset $\mathcal{D}$
input implicit model $q(x\,|\,y)$
input learned discriminator $d(\cdot,\cdot)$
  $y \sim \mathcal{D}$ initialize from the dataset
  for $i = 0 \ldots n$ do
    sample proposal point $x \sim q(x\,|\,y)$
    $P = \min\{1, \frac{d(x,y)}{d(y,x)}\}$
    $x_i \begin{cases} x, & \text{with probability } P \\ y, & \text{with probability } (1-P) \end{cases}$
    $y \leftarrow x_i$
  end for
output $\{x_0, \ldots, x_n\}$
```

We build our reasoning by first assuming that the chain is generated using some discriminator and then successively introducing conditions on the discriminator and upper bounding the distance between the chain and the target. Finally, we come up with an upper bound that can be minimized w.r.t. parameters of the discriminator. Here we consider the case of an implicit Markov proposal, but all of the derivations also hold for independent proposals.

The transition kernel of the implicit Metropolis-Hastings algorithm is

$$t(x\,|\,y) = q(x\,|\,y)\min\left\{1, \frac{d(x,y)}{d(y,x)}\right\} + \tag{16}$$

$$+\,\delta(x-y)\int dx' q(x'\,|\,y)\left(1 - \min\left\{1, \frac{d(x',y)}{d(y,x')}\right\}\right). \tag{17}$$

Further, we want the stationary distribution t_∞ of our Markov chain to be as close as possible to the target distribution p. To measure the closeness of distributions, we consider a standard metric for analysis in MCMC — the *total variation distance*

$$\|t_\infty - p\|_{TV} = \frac{1}{2}\int |t_\infty(x) - p(x)|dx. \tag{18}$$

We assume the proposal $q(x \mid y)$ to be given, but different $d(x,y)$ may lead to different t_∞. That is why we want to derive an upper bound on the distance $\|t_\infty - p\|_{TV}$ and minimize it w.r.t. parameters of the discriminator $d(x,y)$. We derive this upper bound in three steps.

When a transition kernel satisfies the minorization condition, the Markov chain converges "fast" to the stationary distribution. We formalize this statement in the following Proposition.

Proposition 1. *Consider a transition kernel $t(x \mid y)$ that satisfies the minorization condition $t(x \mid y) > \varepsilon\nu(x)$ for some $\varepsilon > 0$, and distribution ν. Then the distance between two consequent steps decreases as:*

$$\|t_{n+2} - t_{n+1}\|_{TV} \leq (1 - \varepsilon) \|t_{n+1} - t_n\|_{TV}, \tag{19}$$

where distribution $t_{k+1}(x) = \int t(x \mid y) t_k(y) dy$.

To guarantee minorization condition of our transition kernel $t(x \mid y)$, we require the proposal $q(x \mid y)$ to satisfy minorization condition with some constant ε and distribution ν (note that for an independent proposal, the minorization condition holds automatically with $\varepsilon = 1$). Also, we limit the range of the discriminator as $d(x,y) \in [b,1]\ \forall x,y$, where b is some positive constant that can be treated as a hyperparameter of the algorithm. These requirements imply

$$t(x \mid y) \geq b q(x \mid y) > b\varepsilon\nu(x). \tag{20}$$

Taking the target distribution $p(x)$ as the initial distribution $t_0(x)$ of our chain $t(x \mid y)$, we reduce the problem of estimation of the distance $\|t_\infty - p\|_{TV}$ to the problem of estimation of the distance $\|t_1 - p\|_{TV}$:

$$\|t_\infty - p\|_{TV} \leq \frac{1}{b\varepsilon} \|t_1 - p\|_{TV}. \tag{21}$$

However, straightforward estimation of $t_1(x)$ results in a biased estimation of $\|t_1 - p\|_{TV}$, since the expectation is inside of a nonlinear function. To overcome this problem, we upper bound this distance in the following proposition.

Proposition 2. *For the kernel $t(x \mid y)$ of the implicit Metropolis-Hastings algorithm, the distance between initial distribution $p(x)$ and the distribution $t_1(x)$ has the following upper bound*

$$\|t_1 - p\|_{TV} \leq 2 \left\| q(y \mid x)p(x) - q(x \mid y)p(y) \frac{d(x,y)}{d(y,x)} \right\|_{TV}, \tag{22}$$

where the TV-distance on the right side is evaluated in the joint space $(x,y) \in \mathbb{R}^D \times \mathbb{R}^D$.

To upper bound the TV-distance $\|\alpha - \beta\|_{TV}$ via KL-divergence $\mathrm{KL}(\alpha\|\beta)$ one can use well-known Pinsker's inequality:

$$2\,\|\alpha - \beta\|_{TV}^2 \leq \mathrm{KL}(\alpha\|\beta). \tag{23}$$

However, Pinsker's inequality assumes that both α and β are distributions, while it is not always true for function $q(x\,|\,y)p(y)\frac{d(x,y)}{d(y,x)}$ in (22). In the following proposition, we extend Pinsker's inequality to the case when one of the functions is not normalized.

Proposition 3. *For a distribution* $\alpha(x)$ *and some positive function* $f(x) > 0\ \forall x$ *the following inequality holds:*

$$\|\alpha - f\|_{TV}^2 \leq \left(\frac{2C_f + 1}{6}\right)(\widehat{\mathrm{KL}}(\alpha\|f) + C_f - 1), \tag{24}$$

where C_f *is the normalization constant of function* f*:* $C_f = \int f(x)dx$*, and* $\widehat{\mathrm{KL}}(\alpha\|f)$ *is the formal evaluation of the KL divergence*

$$\widehat{\mathrm{KL}}(\alpha\|f) = \int \alpha(x)\log\frac{\alpha(x)}{f(x)}dx. \tag{25}$$

Summing up the results, we obtain the final upper bound as follows.

$$\|t_\infty - p\|_{TV}^2 \leq \frac{1}{b^2\varepsilon^2}\,\|t_1 - p\|_{TV}^2 \leq \tag{26}$$

$$\leq \frac{4}{b^2\varepsilon^2}\left\|q(y\,|\,x)p(x) - q(x\,|\,y)p(y)\frac{d(x,y)}{d(y,x)}\right\|_{TV}^2 \leq \tag{27}$$

$$\leq \left(\frac{4+2b}{3\varepsilon^2 b^3}\right)\Bigg(\underbrace{\mathbb{E}_{\substack{x\sim p(x)\\ y\sim q(y\,|\,x)}}\left[\log\frac{d(y,x)}{d(x,y)} + \frac{d(y,x)}{d(x,y)}\right]}_{\text{loss for the discriminator}} - \tag{28}$$

$$-1 + \mathrm{KL}\Big(q(y\,|\,x)p(x)\Big\|q(x\,|\,y)p(y)\Big)\Bigg)$$

We present an empirical evaluation of the proposed algorithm and theory for both independent and Markov proposals. In both cases sampling via the implicit MH algorithm is better than the straightforward sampling from a generator. For independent proposals, we validate our theoretical result by demonstrating monotonous improvements of the sampling procedure throughout the learning of the discriminator (see Fig. 3). Further, the implicit MH algorithm with a Markov proposal compares favourably against competitors with analogues with independent proposals.

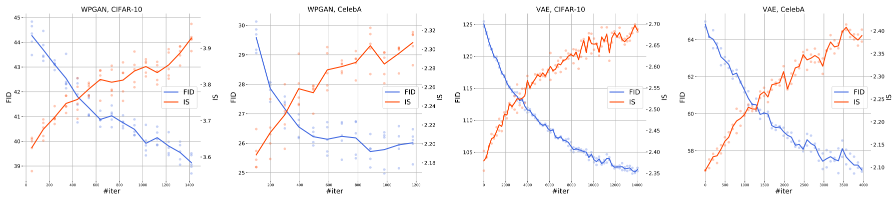


Figure 3: Monotonous improvements in terms of FID and IS for the learning of discriminator by CCE. During iterations, we evaluate metrics 5 times (scatter) and then average them (solid lines). For a single metric evaluation, we use 10k samples. Higher values of IS and lower values of FID are better. Performance for the original generator corresponds to 0th iteration of a discriminator.

Conclusion

The main results of the work can be summarized as follows.

1. The model Structured Bayesian Pruning (SBP) is proposed, for structured sparsity of convolutional neural networks. It is a dropout-like layer that induces multiplicative random noise over the output of the preceding layer. The model has a sparsity-inducing prior over the noise variables and tunes the noise distribution using stochastic variational inference. SBP layer can induce an arbitrary structured sparsity pattern over its input and provides adaptive regularization. We apply SBP to cut down the number of neurons and filters in convolutional neural networks and report significant practical acceleration with no modification of the existing software implementation of these architectures.
2. The probabilistic formulation of the batch normalization layer, and the algorithm for uncertainty estimation are proposed. We study a probabilistic point of view and design a new algorithm that behaves consistently during training and test stages. We compare the performance of the proposed algorithm with concurrent techniques on image classification and uncertainty estimation tasks.
3. Approaching the problem of sampling from the Metropolis-Hastings perspective allows us to obtain efficient solutions for both empirical

and analytical distributions. Under this approach, we demonstrate that the natural objective for the optimization of the independent proposal is symmetric KL divergence. Compared to the variational inference, this procedure takes the forward KL divergence into account, thus fostering mass-covering. For empirical distributions, e.g., a dataset of images, optimization of symmetric KL is equivalent to the conventional GAN training. Nevertheless, our approach allows us to approximate the MH algorithm and obtain consistent improvements during the test stage.

4. We propose the implicit Metropolis-Hastings algorithm for sampling from an empirical target distribution using an implicit probabilistic model as the proposal. In the theoretical part of the paper, we upper bound the distance between the target distribution and the stationary distribution of the chain. The contribution of the derived upper bound is two-fold. We justify the heuristic algorithm proposed earlier and derive the loss functions for the case of Markov proposal. Moreover, the post-processing with the implicit Metropolis-Hastings algorithm can be seen as a justification or enhancement of any implicit model. In the experimental part of the paper, we empirically validate the proposed algorithm on the real-world datasets (CIFAR-10 and CelebA). For both tasks filtering with the proposed algorithm alleviates the gap between target and proposal distributions.

Bibliography

1. *Titsias M.*, *Lázaro-Gredilla M.* Doubly stochastic variational Bayes for non-conjugate inference // International Conference on Machine Learning. — 2014. — P. 1971–1979.
2. *Goodfellow I.*, *Pouget-Abadie J.*, *Mirza M.*, *Xu B.*, *Warde-Farley D.*, *Ozair S.*, *Courville A.*, *Bengio Y.* Generative adversarial nets // Advances in neural information processing systems. — 2014. — P. 2672–2680.
3. *Hastings W. K.* Monte Carlo sampling methods using Markov chains and their applications. — 1970.
4. *Bishop C. M.* Pattern recognition and machine learning. — springer, 2006.
5. *MacKay D. J.* Information theory, inference and learning algorithms. — Cambridge university press, 2003.
6. *Hinton G. E.* Training products of experts by minimizing contrastive divergence // Neural computation. — 2002. — Vol. 14, no. 8. — P. 1771–1800.
7. *Hinton G. E.*, *Dayan P.*, *Frey B. J.*, *Neal R. M.* The" wake-sleep" algorithm for unsupervised neural networks // Science. — 1995. — Vol. 268, no. 5214. — P. 1158–1161.
8. *Kingma D. P.*, *Welling M.* Auto-encoding variational bayes // ICLR. — 2014.
9. *Rezende D. J.*, *Mohamed S.* Variational inference with normalizing flows // arXiv preprint arXiv:1505.05770. — 2015.
10. *Krizhevsky A.*, *Sutskever I.*, *Hinton G. E.* Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. — 2012. — P. 1097–1105.
11. *Lowe D. G.* [et al.]. Object recognition from local scale-invariant features. // iccv. Vol. 99. — 1999. — P. 1150–1157.
12. *Dalal N.*, *Triggs B.* Histograms of oriented gradients for human detection //. — 2005.
13. *LeCun Y.*, *Bengio Y.*, *Hinton G.* Deep learning // nature. — 2015. — Vol. 521, no. 7553. — P. 436.
14. *Goodfellow I.*, *Bengio Y.*, *Courville A.*, *Bengio Y.* Deep learning. Vol. 1. — MIT Press, 2016.
15. *LeCun Y.*, *Boser B.*, *Denker J. S.*, *Henderson D.*, *Howard R. E.*, *Hubbard W.*, *Jackel L. D.* Backpropagation applied to handwritten zip code recognition // Neural computation. — 1989. — Vol. 1, no. 4. — P. 541–551.
16. *Hochreiter S.*, *Schmidhuber J.* Long short-term memory // Neural computation. — 1997. — Vol. 9, no. 8. — P. 1735–1780.
17. *Jaynes E. T.* Probability theory: The logic of science. — Cambridge university press, 2003.

18. *Jaynes E. T.* On the rationale of maximum-entropy methods // Proceedings of the IEEE. — 1982. — Vol. 70, no. 9. — P. 939–952.

19. *Kingma D. P.*, *Salimans T.*, *Welling M.* Variational dropout and the local reparameterization trick // Advances in Neural Information Processing Systems. — 2015. — P. 2575–2583.

20. *Molchanov D.*, *Ashukha A.*, *Vetrov D.* Variational dropout sparsifies deep neural networks // arXiv preprint arXiv:1701.05369. — 2017.

21. *Song J.*, *Zhao S.*, *Ermon S.* A-nice-mc: Adversarial training for mcmc // Advances in Neural Information Processing Systems. — 2017. — P. 5140–5150.

22. *Levy D.*, *Hoffman M. D.*, *Sohl-Dickstein J.* Generalizing Hamiltonian Monte Carlo with Neural Networks // arXiv preprint arXiv:1711.09268. — 2017.

23. *Kingma D. P.*, *Salimans T.*, *Welling M.* Variational Dropout and the Local Reparameterization Trick // Advances in Neural Information Processing Systems 28 / ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett. — Curran Associates, Inc., 2015. — P. 2575–2583.

24. *Kingma D. P.*, *Ba J.* Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. — 2014.