Learning Weight Uncertainty with Stochastic Gradient MCMC for Shape Classification

Chunyuan Li, Andrew Stevens, Changyou Chen, Yunchen Pu, Zhe Gan, Lawrence Carin
Duke University
{cl319, ajs104, cc448, yp42, zg27, lcarin}@duke.edu

Abstract

Learning the representation of shape cues in 2D & 3D objects for recognition is a fundamental task in computer vision. Deep neural networks (DNNs) have shown promising performance on this task. Due to the large variability of shapes, accurate recognition relies on good estimates of model uncertainty, ignored in traditional training of DNNs, typically learned via stochastic optimization. This paper leverages recent advances in stochastic gradient Markov Chain Monte Carlo (SG-MCMC) to learn weight uncertainty in DNNs. It yields principled Bayesian interpretations for the commonly used Dropout/DropConnect techniques and incorporates them into the SG-MCMC framework. Extensive experiments on 2D & 3D shape datasets and various DNN models demonstrate the superiority of the proposed approach over stochastic optimization. Our approach yields higher recognition accuracy when used in conjunction with Dropout and Batch-Normalization.

1. Introduction

Shape is an important representation in visual object recognition. Shape characterizes the boundary/surface of objects, as opposed to other properties such as color, illumination or texture. In computer vision, shapes exist primarily as 2D binary image silhouettes and 3D models of external surfaces. Fig. 1 shows examples of shape data. The availability of large public-domain databases of 2D image and 3D models (*e.g.* ImageNet [61] and ShapeNet [11]) has generated demand for leveraging shape information to find semantic categories of objects. The success of recognition hinges on the geometric and topological representation of shape properties. Traditional recognition methods use hand-crafted features to maintain representation invariance under certain classes of transformations. This has lead to excellent performance in specific domains, *e.g.*, spectral geometry and topological persistence methods for non-rigid shapes [8, 45].

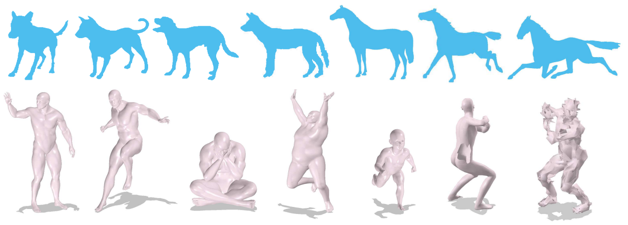
Figure 1. Variability in 2D shapes (top row) and 3D shapes (bottom row). 3D shapes are visualized as projected 2D views. Object articulation can be seen in the horse examples. Change of view is shown in the dog examples for 2D data, and the two rightmost human body shapes show modeling error in 3D data.

Shapes in the real-world, however, manifest rich within-class variability. This variability is due to, for example, object articulation, change of viewpoint in 2D, and modeling error in 3D (see examples in Fig. 1). Shape variability inhibits "shallow" engineered representations from generalizing to a broad range of domains and tasks. Recently, deep neural networks (DNNs) have gained popularity in computer vision and pattern recognition [12, 63]. They can learn "deep" representations from either raw data or traditional features to achieve higher discriminative power.

DNNs are usually trained using stochastic optimization methods such as stochastic gradient descent (SGD) [60]. A regularizer imposed on the model parameters can be viewed as the log of a prior on the *distribution* of the parameters, with such a prior connected to a Bayesian perspective. From this standpoint optimizing the cost function may be viewed as a maximum *a posteriori* (MAP) estimate of model parameters. The MAP solution is a single point estimate, ignoring model uncertainty [6, 26]. Such a point estimation often makes over-confident predictions on test data [34, 78], especially when the data has significant variability.

A principled way to incorporate uncertainty during learning is to use Bayesian inference [51, 55]. Unfortunately, exact Bayesian learning of DNNs is generally intractable. One approximate learning procedure is Markov Chain Monte Carlo (MCMC), which can produce sample-based approximations to the posterior [55]. An advantage of MCMC is its asymptotic consistency with the true pos-

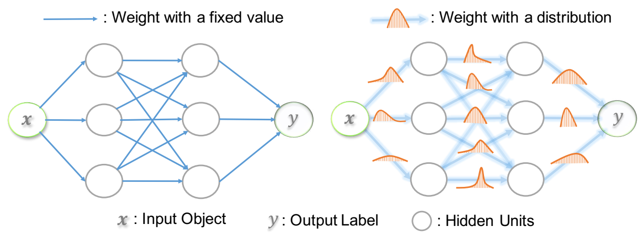

Figure 2. Illustration of Bayesian DNNs with a 2-layer model. All weights in Bayesian DNNs are represented as distributions using SG-MCMC (right figure); rather than having fixed values (left figure), as provided by classical stochastic optimization methods. The SG-MCMC learns correlated uncertainty jointly on all parameters, where (right) associated marginal distributions are depicted.

terior distribution. In fact, MCMC gained extensive attention a decade ago in computer vision [82], with broad applications ranging from image segmentation [71] and human pose estimation [40] to feature correspondence [19]. Despite appealing theoretical properties and excellent empirical results, there is a gap between the limited scalability of conventional MCMC, and an increasingly massive amount of visual/geometric data. This paper seeks fill this gap, based on recent MCMC developments, which have interesting connections to optimization-based approaches.

Specifically, in this paper, we leverage recent advances in stochastic gradient Markov Chain Monte Carlo (SG-MCMC) algorithms [15, 20, 41, 42, 78] to train DNNs. One merit of this family of algorithms is that they are highly scalable. As with an iteration of SGD, SG-MCMC only requires the evaluation of the gradient on a small mini-batch of data. It has been shown that SG-MCMC methods converge to the true posterior by using a slowly-decreasing sequence of step sizes [14, 69]. This means that instead of training a single network, SG-MCMC trains an ensemble of networks, where each network has its weights drawn from a shared posterior distribution. A schematic comparison of optimization and Bayesian learning of DNNs is shown in Fig. 2.

The contributions of this paper are summarized as follows: *i*) A unified Bayesian treatment is provided for DNNs, whose weight uncertainty is learned via SG-MCMC. *ii*) We provide insights on the interpretation of Dropout [64] and DropConnect [74] from the perspective of SG-MCMC, which also allows the use of Batch-Normalization [29]. *iii*) Applications to a wide range of shape classification problems demonstrate the advantages of SG-MCMC over optimization.

2. Related Work

A number of recent works [6, 26] demonstrate the utility of Bayesian learning of DNNs, and advocate the incorporation of uncertainty estimates during model training to improve robustness and performance. In the deep learning literature, two regularization schemes have been developed, Dropout [64] and DropConnect [74]. These schemes help prevent overfitting of the DNN by adding noise to local hidden units and global weights, respectively. In fact, it is possible to view Dropout as an approximate Bayesian learning technique that incorporates uncertainty during learning [22, 33]. Complementary to that, we show later that DropConnect can be viewed as a variant of SG-MCMC.

Deep learning has recently been employed in shape recognition, from the perspective of two broad categories. The first is the use of deep models with stochastic hidden layers, including deep belief networks [9] or the autoencoder [80]; these methods discover latent representations of shapes from their hand-crafted features, such as spectral descriptors [2, 67]. The second category is the adoption of deterministic layers, *i.e.*, feedforward neural networks. A discriminative non-linear mapping is learned in [21] to embed traditional shape features. Moreover, the feedforward CNN has been extended to learn representations from raw shape data. Interestingly, due to the intrinsic complexity of shape geometry, the convolution operator has been applied in a variety of spaces, including multi-view projections [3, 66, 75], 3D volumes [79], curvature space [31] and meshed graphs [7]. However, none of these methods consider uncertainty during training. We extend these models with Bayesian learning, and show improved performance across a broad range of applications.

3. Deep Neural Nets: A Bayesian Perspective

3.1. Predictive Models

Assume we are given data $\mathcal{D} = \{\boldsymbol{d}_1, \cdots, \boldsymbol{d}_N\}$, where $\boldsymbol{d}_n \triangleq (\boldsymbol{x}_n, y_n)$, with input object/feature $\boldsymbol{x}_n \in \mathbb{R}^D$ and output label $y_i \in \mathcal{Y}$, with $\mathcal{Y}$ being the output discrete label space. A model characterizes the relationship from $\boldsymbol{x}$ to y with parameters $\boldsymbol{\theta}$. The parameters $\boldsymbol{\theta}$ are assigned a prior distribution $p(\boldsymbol{\theta})$. The corresponding data likelihood is $p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^N p(\boldsymbol{d}_i|\boldsymbol{\theta})$. Following Bayes rule, the posterior is $p(\boldsymbol{\theta}|\mathcal{D}) \propto p(\boldsymbol{\theta})p(\mathcal{D}|\boldsymbol{\theta})$.

For testing, given a test input $\tilde{\boldsymbol{x}}$ (with missing label $\tilde{y}$), the uncertainty learned in training is transferred to prediction, yielding the posterior predictive distribution in (1).

$$p(\tilde{y}|\tilde{\boldsymbol{x}}, \mathcal{D}) = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[p(\tilde{y}|\tilde{\boldsymbol{x}}, \boldsymbol{\theta})] = \int_{\boldsymbol{\theta}} p(\tilde{y}|\tilde{\boldsymbol{x}}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}) \ . \quad (1)$$

The predicted distribution of $\tilde{y}$ may be viewed in terms of model averaging across parameters, based on the learned $p(\boldsymbol{\theta}|\mathcal{D})$; this should be contrasted with learning a single point estimate of $\boldsymbol{\theta}$ based on $\mathcal{D}$.

3.2. From Logistic Regression to DNNs

One simple Bayesian predictive model is logistic regression (LR). For binary classification, the likelihood is:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) \triangleq g_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-(\mathbf{W}^\top \boldsymbol{x} + \boldsymbol{c})\right)} \ , \quad (2)$$

where a Gaussian prior $\mathcal{N}(0, \sigma^2)$ is usually placed for each element of the model parameters $\boldsymbol{\theta} \triangleq \{\mathbf{W}, \boldsymbol{c}\}$; the variance σ^2 imposes the amount of prior uncertainty in model parameters.

In complex real-world modeling such as shape classification, such a simple parametric model is often not expressive enough for robust generalization. DNNs extend LR by parameterizing the form of relationship from $\boldsymbol{x}$ to y, as a composition of a set of nonlinear functions (*e.g*. the sigmoid function used in LR). Specifically, an L-layer DNN for multi-class classification puts a softmax function on the output of a set of function compositions [51, 55]:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \text{softmax}\big(g_{\boldsymbol{\theta}_L} \circ \cdots \circ g_{\boldsymbol{\theta}_0}(\boldsymbol{x})\big) \ , \qquad (3)$$

where $\circ$ denotes function composition, $\text{softmax}(\boldsymbol{x}) \triangleq e^{\boldsymbol{x}}/(\sum_i e^{x_i})$. Similarly, Gaussian priors are adopted on model parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_L\}$. As a result, one may consider the LR as a "zero-layer" neural network.

Various DNN models are defined by specifying $g_{\boldsymbol{\theta}_\ell}$ for each layer. In the following, we describe two canonical DNNs: Feedforward Neural Networks (FNNs) and Convolutional Neural Networks (CNNs). These serve as major building blocks to construct more advanced DNN models (*e.g*. VGG networks [63]). By choosing prior distributions for the network weights, all DNN models can be viewed as in the family of Bayesian predictive models [72].

Feedforward Neural Networks The FNN can learn sophisticated functional representations from input/output examples [17, 26]. This can be further employed to discover discriminative feature representations [21]. Specifically, it sends the input data forward from the bottom layer to the top layer with pointwise nonlinear functions between each layer. In the ℓ-th layer, $g_{\boldsymbol{\theta}_\ell}$ in (3) can be the sigmoid function, the hyperbolic tangent, and more recently, the Rectified Linear Unit (ReLU) [23]. ReLU takes the form:

$$g_{\boldsymbol{\theta}_\ell}(\boldsymbol{x}) = \max(0, \mathbf{W}_\ell^\top \boldsymbol{x} + \boldsymbol{c}_\ell) \text{ , with } \boldsymbol{\theta}_\ell = (\mathbf{W}_\ell, \boldsymbol{c}_\ell) \ .$$

Convolutional Neural Networks The CNN is a special class of FNNs, typically applied to data with spatial or temporal covariates. The CNN employs the convolution operation at each layer of the feedforward network. CNNs are commonly used for learning deep feature representations from raw data [12, 35, 59, 63, 79]. The nonlinear function $g_{\boldsymbol{\theta}_\ell}$ in CNN is typically composed of convolution and pooling operators [38]. The CNN can take advantage of the properties of natural signals [37] such as images and shapes, which exhibit high local correlations [7, 12] and rich shared components [8, 81].

Given inputs in the form of multiple arrays $(\boldsymbol{x}_{k_{\ell-1}})_{k_{\ell-1}=1}^{K_{\ell-1}}$ from the $(\ell-1)$-th layer, for the k_ℓ-th filter bank $\mathbf{W}_\ell$ in the ℓ-th layer, the output is

$$g_{\mathbf{W}_{k_\ell}}(\boldsymbol{x}_{k_{\ell-1}}) = \text{Pool}\big(\textstyle\sum_{k_{\ell-1}} \mathbf{W}_{k_\ell}^\top * \boldsymbol{x}_{k_{\ell-1}}\big) \ ,$$

where $*$ is the convolution operator, Pool is the pooling operator (*e.g.*, max-pooling [38]). The parameters for the ℓ-th layer are $\boldsymbol{\theta}_\ell \triangleq \{\mathbf{W}_{k_\ell}\}$.

4. Scalable Learning with SG-MCMC

We describe a general framework for SG-MCMC algorithms in this section, but recommend [78, 41] for more intuitive explanations for the employed algorithms. SG-MCMC are derived from a corresponding stochastic differential equation (SDE), represented in the general form of an Itô diffusion:

$$\mathrm{d}\boldsymbol{\Psi}_t = F(\boldsymbol{\Psi}_t)\mathrm{d}t + \delta(\boldsymbol{\Psi}_t)\mathrm{d}W_t \ , \qquad (4)$$

where $\boldsymbol{\Psi}_t \in \mathbb{R}^K$ is the model state, W_t is Brownian motion, and t is the (continuous) time index. The model state could represent model parameters $\boldsymbol{\theta}$ [78], or an augmented state space (*e.g.*, with momentum [15]), in which case we discard the auxiliary variables to perform the desired marginalization given samples from the diffusion [15, 20]. The function $F : \mathbb{R}^K \to \mathbb{R}^K$ is the deterministic drift, and $\delta : \mathbb{R}^K \to \mathbb{R}^{K \times K}$ is the diffusion matrix. It has been shown that with appropriate functions F and δ, the stationary marginal distribution $\rho(\boldsymbol{\theta})$, of an Itô diffusion (4) has a marginal distribution equal to the posterior distribution [14, 50].

In the Bayesian setup in Section 3.1, this means $\rho(\boldsymbol{\theta}) \propto \exp(-U(\boldsymbol{\theta}))$, where U is the unnormalized negative log-posterior:

$$U \triangleq -\left(\sum_{i=1}^{N} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right) . \qquad (5)$$

To deal with large-scale datasets, an unbiased *stochastic gradient* is computed for each mini-batch of data during the learning procedure, *e.g.*

$$\tilde{U}_t \triangleq -\left(\frac{N}{|\mathcal{S}_t|} \sum_{i \in \mathcal{S}_t} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})\right) \qquad (6)$$

for the t-th minibatch. Here $\mathcal{S}_t \subset \{1, 2, \cdots, N\}$ chooses the mini-batch data points, and $|\cdot|$ is the cardinality of a set.

4.1. SG-MCMC Algorithms

DNNs often exhibit pathological curvature and saddle points in their parameter spaces [18]. Efficient learning requires a sampler to adapt random walks according to the local geometry of the parameter space (highly related concepts have been considered in optimization-based methods, *e.g.* RMSprop [70] and Adam [32], as discussed further below). A variant of SG-MCMC that incorporates geometry information is the *stochastic gradient Riemannian Langevin dynamics* (SGRLD). It specifies an Itô diffusion as:

$$\tilde{F} = -\mathbf{G}^{-1}\nabla_{\boldsymbol{\theta}}\tilde{U}(\boldsymbol{\theta}) - \Gamma(\boldsymbol{\theta}), \text{ and } \delta = \sqrt{2}\mathbf{G}^{-\frac{1}{2}} \; . \quad (7)$$

There are many choices for $\mathbf{G}$, one of which is $\mathbf{G} \triangleq G(\boldsymbol{\theta}) = \mathcal{I}_{\boldsymbol{\theta}}$, the Fisher information metric [57], and $\Gamma_i(\boldsymbol{\theta}) = \sum_j \frac{\partial G_{ij}^{-1}(\boldsymbol{\theta})}{\partial \theta_j}$ describes how the curvature of the manifold defined by $G(\boldsymbol{\theta})$ changes for small changes in $\boldsymbol{\theta}$. For the many models of interest, direct computation of the Fisher information metric is impractical.

In SG-MCMC, one draws samples from (4), which usually does not have an analytic form. As a result, an ϵ-discretization of the continuous SDE can be used to approximate (4), where ϵ represents the step size. The approximate samples $\boldsymbol{\theta}_t$ are then collected via a Markov Chain with the following updates, with $\mathbf{I}$ being the identity matrix:

$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \epsilon\tilde{F}(\boldsymbol{\theta}_{t-1}) + \sqrt{\epsilon}\delta(\boldsymbol{\theta}_{t-1})\boldsymbol{\xi}, \; \boldsymbol{\xi} \sim \mathcal{N}(0, \mathbf{I}) \; . \quad (8)$$

In the testing stage, these samples are used to construct a sample-based uncertainty estimation to the posterior predictive distribution defined in (1):

$$p(\tilde{y}|\tilde{\boldsymbol{x}}, \mathcal{D}) \approx \frac{1}{T}\sum_{t=1}^{T} p(y|\boldsymbol{x}, \boldsymbol{\theta}_t) \; . \quad (9)$$

SGLD SGRLD reduces to the *stochastic gradient Langevin dynamics* (SGLD) [78] when $\mathbf{G} = \mathbf{I}$ (thus $\Gamma(\boldsymbol{\theta})$ vanishes). This "vanilla" SGLD is the first attempt in line of work of SG-MCMC. We can see that Robbins-Monro type algorithms (*e.g.* SGD) [60] which stochastically optimize a likelihood, can be combined with Langevin dynamics [56] where Gaussian noise is injected during parameter updates.

In the SGLD algorithm, we define the stochastic gradient evaluated at position $\boldsymbol{\theta}_t$ with mini-batch $\mathcal{S}_t$ as:

$$\tilde{\boldsymbol{g}}_t \triangleq \tilde{g}(\boldsymbol{\theta}_t; \mathcal{S}_t) = \frac{1}{|\mathcal{S}_t|}\sum_{i \in \mathcal{S}_t} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}_t) \; . \quad (10)$$

Consequently, one SGLD update on model parameters consists of 2 parts: (*i*) an SGD-like update, and (*ii*) Gaussian noise injection, making the algorithm explore the parameter space, rather than converging to a MAP solution. The procedure is summarized in Algorithm 1. We note that SGLD updates all parameters with the same step size, resulting in slow mixing when the components of $\boldsymbol{\theta}$ have different curvature, which is particularly true in the case of DNNs.

Preconditioned SGLD As alluded to above, directly incorporating $\mathcal{I}_{\boldsymbol{\theta}}$ is intractable in DNNs due to the infeasibility of representing the Fisher information matrix. *Preconditioned SGLD* (pSGLD) [41] instead uses adaptive preconditioners borrowed from stochastic optimization [70, 32] to construct a feasible and effective $\mathbf{G}$. It equalizes the gradient with the following construction of $\mathbf{G}$, so that the step size is adaptive among different dimensions:

Algorithm 1: SGLD algorithm

```
Initialize: Random $\boldsymbol{\theta}_1$;
for $t = 1, 2, \ldots, T$ do
    %Estimate gradient from minibatch $\mathcal{S}_t$
    $\tilde{\boldsymbol{g}}_t \leftarrow \frac{1}{|\mathcal{S}_t|}\sum_{i \in \mathcal{S}_t} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}_t)$;
    %Parameter update
    $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \epsilon\mathbf{I})$;
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \frac{\epsilon}{2}\Big[\Big(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + N\tilde{\boldsymbol{g}}_t\Big)\Big] + \boldsymbol{\xi}_t$;
end
```

Algorithm 2: Practical pSGLD algorithm

```
Input: Default settings for the tested models:
        $\epsilon = 1\times10^{-3}, \lambda = 10^{-8}, \beta_1 = 0.99$.
Initialize: $\boldsymbol{v}_0 \leftarrow \mathbf{0}$, random $\boldsymbol{\theta}_1$;
for $t = 1, 2, \ldots, T$ do
    %Estimate gradient from minibatch $\mathcal{S}_t$
    $\tilde{\boldsymbol{g}}_t \leftarrow \frac{1}{|\mathcal{S}_t|}\sum_{i \in \mathcal{S}_t} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{d}_i|\boldsymbol{\theta}_t)$;
    %Preconditioning
    $\boldsymbol{v}_t \leftarrow \beta_1\boldsymbol{v}_{t-1} + (1-\beta_1)\tilde{\boldsymbol{g}}_t \odot \tilde{\boldsymbol{g}}_t$;
    $\mathbf{G}_t^{-1} \leftarrow \text{diag}\Big(\mathbf{1} \oslash \big(\lambda\mathbf{1} + \boldsymbol{v}_t^{\frac{1}{2}}\big)\Big)$;
    %Parameter update
    $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \epsilon\mathbf{G}_t^{-1})$;
    $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \frac{\epsilon}{2}\Big[\mathbf{G}_t^{-1}\Big(\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + N\tilde{\boldsymbol{g}}_t\Big)\Big] + \boldsymbol{\xi}_t$;
end
```

$$\boldsymbol{v}_t \leftarrow \beta_1\boldsymbol{v}_{t-1} + (1-\beta_1)\tilde{\boldsymbol{g}}_t \odot \tilde{\boldsymbol{g}}_t \; , \quad (11)$$

$$\mathbf{G}_t^{-1} \leftarrow \text{diag}\Big(\mathbf{1} \oslash \big(\lambda\mathbf{1} + \boldsymbol{v}_t^{1/2}\big)\Big) \; , \quad (12)$$

where $\{\boldsymbol{v}_t\}$ are intermediate variables, β_1, λ are hyper parameters, $\odot$ is an element-wise product, and $\oslash$ element-wise division. By transforming the landscape of the parameter space to be more uniformly curved, it is possible for the sampler to move much faster. The effect of $\Gamma(\boldsymbol{\theta})$ is not needed in our case since $\boldsymbol{\theta}_t$ contributes little in the construction of $\boldsymbol{v}_t$ (thus the derivative w.r.t. $\boldsymbol{\theta}_t$ is small), we thus exclude it from the practical procedure, shown in Algorithm 2 [41]. pSGLD maintains faster mixing with trivial overhead per iteration than SGLD.

4.2. From Stochastic Optimization to SG-MCMC

SG-MCMC methods are closely related to conventional stochastic optimization. The main difference is the injection of Gaussian noise during parameter updates. With the correct amount of noise, the algorithm becomes a posterior sampling method, instead of a MAP optimization method. For example, SGLD has been shown to be SGD with $\mathcal{N}(0, \epsilon\mathbf{I})$ [78]; similarly pSGLD can be viewed as RMSprop with $\mathcal{N}(0, \epsilon\mathbf{G}^{-1})$ [41], and SGHMC is momentum SGD with injecting Gaussian noise [15]. In fact, if we anneal the system temperature of the Itô diffusion to zero, in the limit SG-MCMC becomes stochastic optimization [13].

Prediction using stochastic optimization methods often involves finding an optimum point estimation of the parameters on the training dataset, *e.g.*, by maximum likelihood or MAP, then using this estimate for testing. This has the disadvantage that it does not account for any uncertainty in the parameters—which will underestimate the variance of the predictive distribution. Whereas in Bayesian learning, (9) captures a better representation of the uncertainties in the learning process, and helps prevent model overfitting [34].

4.3. SG-MCMC Interpretation of DropConnect

In deep learning, *Dropout/DropConnect* [27, 74] have been proposed to improve model generalization by explicitly adding noise during DNN learning. Our SG-MCMC model learning provides a principled Bayesian interpretation for DropConnect.

Regular (binary) DropConnect adds noise to *global* network weights, by setting a randomly selected subset of weights to zero within each layer:

$$\boldsymbol{z} = g((\boldsymbol{\xi}_0 \odot \boldsymbol{\theta})\boldsymbol{h}), \;\; \boldsymbol{\xi}_0 \sim \mathsf{Bernoulli}(1-p) \; , \qquad (13)$$

where $\boldsymbol{h}$ and $\boldsymbol{z}$ are the input and output layers, and p is the probability that the weight is dropped. Binary Dropout instead randomly selects *local* hidden units:

$$\boldsymbol{z} = \boldsymbol{\xi}_0 \odot g(\boldsymbol{\theta h}), \;\; \boldsymbol{\xi}_0 \sim \mathsf{Bernoulli}(1-p) \; , \qquad (14)$$

It was shown in [64] that binary Dropout has a Gaussian approximation $\boldsymbol{\xi}_0 \sim \mathcal{N}(1, \frac{p}{1-p})$, called *Gaussian Dropout* [76] with identical regularization performance and faster convergence. We note that the SGD update with Gaussian DropConnect shares the same form as SGLD:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\xi}_0 \odot \boldsymbol{\theta}_t + \frac{\epsilon}{2}\nabla_{\boldsymbol{\theta}} f = \boldsymbol{\theta}_t + \frac{\epsilon}{2}\nabla_{\boldsymbol{\theta}} f + \boldsymbol{\xi}_0' \; , \qquad (15)$$

where $\boldsymbol{\xi}_0' \sim \mathcal{N}(0, \epsilon\mathbf{V})$, $\mathbf{V} = \frac{p}{\epsilon(1-p)}\text{diag}(\boldsymbol{\theta}_t^2)$ and $\nabla_{\boldsymbol{\theta}} f = \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}_t) + N\tilde{\boldsymbol{g}}$; (15) shares a similar update rule as the modified SGLD [73] with a different $\mathbf{V}$.

As a result, the noise added on the weights in the Gaussian DropConnect can be interpreted as injected noise from the Brownian motion of Langevin dynamics. Similarly, binary DropConnect can be viewed as using the Spike-and-Slab prior [30] on the weights without updating the prior parameters during inference. We can see that SG-MCMC algorithms are more adaptive than DropConnect, because the noise variance relates to the step size and the local geometry of parameter space, instead of using manually fixed values in DropConnect. Furthermore, since Dropout can be interpreted as a structured DropConnect where the weights are dropped block-wise [74], Dropout is subsumed in the SG-MCMC interpretation of DropConnect. The integration of binary Dropout with SG-MCMC can be viewed as learning weight uncertainty of mixtures of neural networks.

4.4. Accelerating with Batch-Normalization

Another method to regularize local hidden units is *Batch-Normalization* [29]. When performing gradient-based learning of a DNN, a large step size may result in exploding or vanishing gradients. In the context of the proposed Bayesian learning, the sampler could get trapped in poor local minima or on saddle points. Batch-Normalization can be readily applied in our SG-MCMC training to help address these issues. (i) It normalizes activations throughout the network, preventing small parameter changes from being amplified into larger, suboptimal activation changes in other layers. This further helps prevent the sampler in our SG-MCMC algorithms from getting stuck in the saturated regimes of nonlinearities. (ii) It allows us to use much higher step size, and thus increases the effective sample size while preventing overfitting, in which case the need for Dropout is eased.

5. Experiments

We implemented pSGLD and SGLD in Torch7 [16] using the GPU library `cudnn` [1], and present results in two main parts: (i) The uncertainty captured on weights can help the FNN to learn more discriminative representations when traditional hand-crafted features are provided. (ii) We also train CNN models and advanced variants on raw data with our approach. We show that SG-MCMC can work in conjunction with both Dropout and Batch-Normalization to improve performance on a broad range of networks and datasets. In practice, we find that it often helps convergence to anneal the variance of the noise term, and which corresponds to model averaging with the annealed distribution.

The default setting in all experiments is $\sigma^2 = 1$. Following [34], a block decay strategy for step size is used; it decreases by half after every L epochs. The hyperparameter setting and model specifications on each dataset are clarified in Supplementary Material. No data augmentation techniques [35] are applied to the datasets tested.

5.1. Feedforward Neural Networks

5.1.1 2D Digits and Animals

We first demonstrate application of the FNN on two 2D shape datasets, `MNIST` [38] and `Animal` [4].

- `MNIST` contains 60000 shapes of 2D digits, and 10000 test samples. The task is to classify the images into number ranging from 0 to 9 according to shape information.
- The `Animal` dataset [4] contains 2000 shapes describing 20 kinds of animals, including butterfly, deer, *etc*. One example for each class is displayed in Fig. 3. The dataset has much more intra-class variability since the same kind of animals may have various poses.

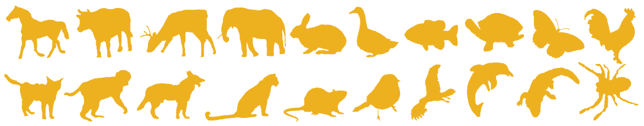
Figure 3. Example 2D shapes from `Animal` dataset.

Table 1. Test accuracy of FNN on `MNIST`.

Networks	400-400	800-800	1200-1200
Methods		**Test Error (%)**	
pSGLD + Dropout	1.36	1.26	**1.15**
SGLD + Dropout	1.45	**1.25**	1.18
RMSprop + Dropout	1.35	1.28	1.24
SGD + Dropout	1.51	1.33	1.36
pSGLD	1.45	1.32	1.24
SGLD	1.64	1.41	1.40
RMSprop	1.79	1.43	1.39
SGD	1.72	1.47	1.47
RMSspectral [10]	1.65	1.56	1.46
BPB, Gaussian [6]	1.82	1.99	2.04
BPB, Scale mixture [6]	**1.32**	1.34	1.32

A two-layer network (X-X) with ReLU is employed, where X is the number of hidden units for each layer. For `MNIST`, the test classification errors for network size 400-400, 800-800 and 1200-1200 are shown in Table 1. The inputs to the FNN are vectorized images of dimension D=784. 100 epochs are used, with $L = 20$. The thinning interval is 100, and MCMC burn-in is 1 epoch.

We compare the SG-MCMC algorithms with representative stochastic optimization methods: SGD, RMSprop and RMSspectral [10]. After tuning, the optimal step size for pSGLD and RMSprop is set as $\epsilon = 10^{-3}$, while for SGLD and SGD as $\epsilon = 5 \times 10^{-1}$. The testing error for the SG-MCMC methods are consistently lower than their corresponding stochastic optimization counterparts. This indicates that the weight uncertainty learned via SG-MCMC can improve performance.

To demonstrate that the improvements are not due just to model averaging, we collected "partially trained" models along the learning trajectory of RMSprop using the same collection scheme as pSGLD, and average over their testing evaluations. The averaged prediction gives 1.30%, improving RMSprop by 0.09%, but is still inferior to pSGLD by 0.06% on the 1200-1200 network. This is unsurprising, because pure optimization methods are often stuck in local modes, while the Brownian motion in SG-MCMC encourages the learning trajectory to explore the full space.

We also compare with a recent variational Bayesian approach to learn weight uncertainty, BPB with Gaussian and scale mixtures [6]. pSGLD provides better results on larger networks, this probably because SG-MCMC learns correlated uncertainty jointly on all parameters. Nevertheless, both approaches provide lower errors than optimization methods. It indicates learning weight uncertainty is helpful for DNNs.

The combination of Dropout with stochastic optimization and SG-MCMC is compared on the 400-400 FNN using `MNIST`. The default Dropout probability $p = 0.5$ is used. Both of Dropout and SG-MCMC show their ability to regularize learning. By integrating SG-MCMC with Dropout, we obtain lower error.

To study the role of p for both approaches, we vary p from 0 to 0.7 with a step size of 0.05, with 5 repetitions. Overall, pSGLD with Dropout provides lower errors than RMSprop with Dropout. Note that when $p = 0$, we recover pure SG-MCMC or stochastic optimization algorithms. A much lower error is provided by SG-MCMC, indicating the importance of uncertainty in recognition.

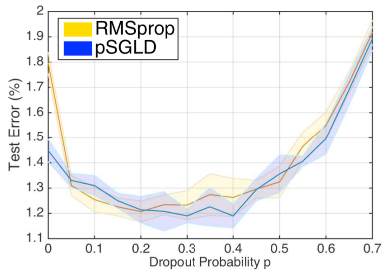

Figure 4. Dropout comparison.

For the `Animal` dataset, we use Bag-of-Contour-Fragments (BCF) [77] as input features, which have shown excellent discriminative power for 2D shape classification. We follow [77] and use a random selection of half of the shapes per class for training and the rest of shapes for testing. We run 5 repetitions and the average classification accuracy of our method is compared to other state-of-the-art methods in Table 2. A 2-layer FNN of size 800-800 with ReLU was used. It yields higher average testing accuracies than BCF with SVM [77], integration of contour and skeleton (ICS) [4], and inner distance shape contexts (IDSC) [49]. This shows that the FNN can learn a more discriminative representation based on traditional features for classification.

We also see that RMSprop for FNN gives a lower accuracy than pSGLD. Training LR directly for shallow representation (*i.e.*, BCF feature) with pSGLD yields 78.3%, which is significantly higher than 72.1% of RMSprop. This indicates that incorporating uncertainty in either learning representation or classification improves accuracy.

5.1.2 3D Non-rigid and Textured Shapes

Two standard 3D benchmarks from recent Eurographics SHREC contests are used, including `Textured Shape` [5] and non-rigid `Human Body Shape` datasets [58]. Exemplar shapes are displayed in Fig. 5.

- `Textured Shape` dataset is made of 572 watertight mesh models that have a geometric and texture deformation. Two tasks are considered: assignment to 16 `Geometry` classes or 13 `Texture` classes. Since the categories are reasonably differentiable, a one layer FNN with 400 ReLU units is used.

Table 2. `Animal` dataset.

Methods	Accuracy (%)
pSGLD	**84.80**± 0.35
RMSprop	84.30± 0.55
BCF + SVM [77]	83.40±1.30
ICS [4]	78.4
Distinctive Parts [46]	80.7
Bag of SIFT [47]	80.4
IDSC [49]	73.6
Class segment set [68]	69.7

Table 3. `Body Shape` and `Textured Shape`.

Datasets	`Body`	`Geometry`	`Texture`
Methods	**Accuracy (%)**		
pSGLD	**57.33** ± 3.06	**38.46**	51.49
RMSprop	56.31± 2.34	37.01	49.39
LR	51.23± 1.86	36.06	46.59
SVM	56.02± 2.94	37.09	**51.62**
KNN	46.90± 2.06	18.01	50.35

Table 4. 20 `Newsgroups`.

Methods	Accuracy (%)
pSGLD	**73.32**
RMSprop	72.52
DPFM [25]	72.67
PFM [25]	72.11
OSM [65]	69.1
DocNADE [36]	68.4
RSM [28]	67.7
LDA [54]	65.7

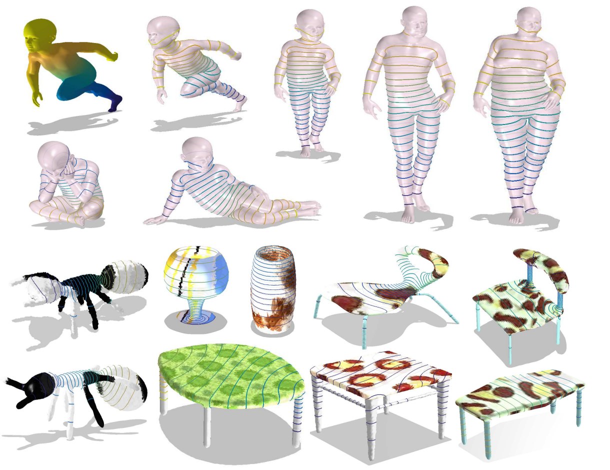
Figure 5. Example 3D shapes with ISPM. The top two rows are examples from the `Human Body Shape` dataset, the bottom two rows are examples from `Textured Shape` dataset. The shape in the top-right corner is rendered with the 2nd eigenfunction of the Laplace-Beltrami operator. It is used to construct the intrinsic partition, which is isometry-invariant and robust to shape deformation [43].

- The `Synthetic` sub-dataset of `Human Body Shape` is used for fine-grained shape classification. There are 15 different human models, each with its unique body shape and 20 different poses, resulting in a dataset of 300 models. 2 poses are for training, the rest for testing. Accuracy is averaged over 5 random train-test partitions. A 2-layer FNN with 800-800 ReLU units is used.

We follow the standard bag-of-features paradigm [8], where we extract local geometric descriptors, specifically spectral graph wavelet signature (SGWS) [44], and aggregate them with the intrinsic spatial pyramid representation (ISPM) [43]. A color histogram is embedded into ISPM to describe texture on surfaces. This pipeline has shown excellent performance in two contests [5, 58].

These features are used in FNN, support vector machine (SVM), LR and K-nearest-neighbor (KNN) for classification. The results are reported in Table 3. We emphasize that the purpose of this experiment is to compare different learning algorithms on the same input features, not to achieve state-of-the-art results. Consistent with the 2D shape results, learning deep representations with FNN shows improved performance over shallow features. The uncertainty captured by pSGLD also boosts the results.

5.1.3 General Bag-of-words setup

Besides the above *visual/geometric words* scenarios tested on 2D/3D shapes [8, 43, 77], we further show our SG-MCMC method can learn uncertainty-based discriminative representation for a general bag-of-words setup. This is demonstrated with a standard document classification corpus, 20 `Newsgroups`. This dataset has 11315 training and 7531 test documents with a vocabulary size of 2000.

An 800-800 FNN is used. Inputs of the FNN are term frequency-inverse document frequency features (tf-idf). Classification of tf-idf using the SVM yields 69.60%. We compare our model against state-of-the-art methods in Table 4, including a very recent Bayesian deep model, deep Poisson factor modeling (DPFM) [25]. We adapt the FNN as a Bayesian model to capture uncertainty in recognition using Gaussian prior with $\sigma^2 = 10$ on network weights, which is the same goal as DFPM. However, our Bayesian FNN is more desirable, because it yields higher accuracy in only 1 minute, while DFPM needs more than 1 hour. It also significantly outperforms four other methods, including supervised Latent Dirichlet allocation (LDA) [54], DocNADE [36], Replicated Softmax Model (RSM) [28] and Overreplicated Softmax Model (OSM) [65].

5.2. Convolutional Neural Networks

5.2.1 2D Digits and Caltech 101 Silhouettes

We also studied incorporating weight uncertainty in learning CNN-based representations. For 2D raw shapes, besides `MNIST`, we also consider the `Caltech 101 Silhouettes` dataset [52], containing 6364 training and 2307 test images.

On `MNIST`, we built 2 networks: one is "*LeNet*" [38], composed of a 2-layer CNN followed by a 2-layer FNN; the other is "*4-CNN*", it is similar to *LeNet* except that a 4-layer CNN is used with max-pooling after the 1st, 2nd, and 4th convolutional layers. The results are shown in Table 5. pSGLD significantly outperforms RMSprop on both networks, indicating that learning weight uncertainty in CNN models can boost results. Combined with Dropout, the performance of both SG-MCMC and optimization approaches are improved. pSGLD with Dropout reaches a testing error of 0.37%, which is lower than several state-of-the-art methods, including deeply supervised nets (0.39%) [39], stochastic pooling (0.47%) [81], Network in Network (0.47%) [48] and Maxout Network (0.45%) [24].

Table 5. Test error of CNN on MNIST, Caltech, ModelNet and Cifar10.

Dataset	MNIST		Caltech		ModelNet		Cifar10	
Networks	LeNet	4-CNN	5-CNN	5-CNN-BN	Vol-CNN	View-CNN	VGG-B	VGG-B-BN
Methods	**Test Error (%)**							
RMSprop	0.65	0.62	27.52	24.40	17.06	16.13	21.04	13.52
pSGLD	0.45	0.49	26.96	25.53	16.21	15.12	20.28	14.80
RMSprop + Dropout	0.56	0.45	27.43	24.66	14.87	**14.52**	16.38	10.39
pSGLD + Dropout	**0.40**	**0.37**	**24.57**	**23.97**	**14.51**	14.62	**16.11**	**9.47**

For Caltech, we employed *5-CNN* a smaller version of the VGG networks. It is tested with and without Batch Normalization (BN). The results are shown in Table 5. Due to the large variability of shapes in this dataset, a large variance $\sigma^2 = 100$ in the prior is used. Dropout consistently provides considerable improvement for pSGLD, but not RMSprop. Using BN also improves performance.

5.2.2 3D ModelNet

We next evaluate the CNN on a large-scale 3D model dataset, Princeton ModelNet [79]. We use the 40-class setup, containing 12311 training and 2468 testing samples. We follow typical settings to constitute our experiments [66, 79]. Two CNN-based approaches are designed for 3D model classification. (i) The first is based on volume representation of 3D shapes; it performs 3D volumetric convolutions [79]. We use a $30 \times 30 \times 30$ bounding box to represent each shape, and construct "*Vol-CNN*" network: 3-layer volumetric CNN followed by 1-layer FNN with ReLU. This is the same network as [79] except for the nonlinearities. (ii) The second approach is to project the 3D model into multiple 2D depth-view-images, and perform 2D convolutions on each view [66]. We employed 12 depth-views of size 224×224 to represent a 3D shape, and adopted the "*View-CNN*" based on the CNN-M network from [12]: a 5-layer CNN followed by a 2-layer FNN with ReLU. BN is applied for both networks, and the results are shown in Table 5. In both setups, we see improved performance using pSGLD over RMSprop. When applying Dropout ($p = 0.5$) to *View-CNN*, RMSprop is better than pSGLD. Since Dropout determines the portion of parameters updated in our sequential update interpretation, further study on choosing p could alleviate the problem.

We also outperform several state-of-the-art results on this dataset, including the ShapeNets method (23.00%) [79], VoxNet (17.00%) [53] and deep panoramic views (22.37%) [62]. The only method with lower error than ours is Multi-View CNN (9.9%) [66]; note that their model is pre-trained with ImageNet [61], we did not perform any pretraining.

5.2.3 Very Deep Neural Networks

To show that the proposed method is applicable to "very deep" models and natural images, we conducted experiments on the Cifar10 [35] dataset. It contains 10 classes of 50000 RGB images of vehicles and animals, with an additional 10000 for testing. We follow [63] and use a 13-layer CNN "*VGG-B*" with a 1-layer FNN. We test 8 algorithms by integrating RMSprop and pSGLD with Dropout and BN. 300 epochs are used to collect the final results in Table 5. Step sizes for pSGLD and RMSprop with both Dropout and BN are 2×10^{-3}, all the others are 10^{-3}.

We plot the learning curves of the first 30 epochs in Fig 6. The performance between different learning algorithms on this image dataset are consistent with those on shapes. We see that pSGLD reaches lower errors than RMSprop; this is also true when they are used with Dropout. BN can improve the performance in general, and it allows us to increase the step sizes and speed up learning. Interestingly, the use of either pSGLD or Dropout slows down learning initially. This is likely due to the higher uncertainty imposed during learning, resulting in more exploration of the parameter space. Increased uncertainty, however, prevents overfitting and eventually results in improved performance.

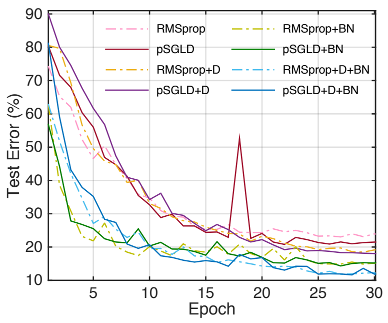

Figure 6. Learning curves on Cifar10. "D" indicates Dropout.

6. Conclusions

We provide a Bayesian framework for DNNs, where the distributions on weights are learned via SG-MCMC. The SG-MCMC is a natural extension of SGD-based optimization methods widely employed in deep learning; we leverage ideas from the SGD literature to improve our proposed SG-MCMC procedure. One may interpret DropConnect as a stochastic sampling algorithm. It further allows us to integrate Dropout and Batch-Normalization into the SG-MCMC framework, to improve the posterior sampling of DNNs. We have considered a broad range of applications to shape classification and have demonstrated the advantages of the proposed method over its optimization counterpart. The proposed method yields state-of-the-art performance on several tested models and datasets.

Acknowledgements This research was supported in part by ARO, DARPA, DOE, NGA, ONR and NSF.

References

[1] NVIDIA cuDNN – GPU accelerated deep learning. 5

[2] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV Workshops*, 2011. 2

[3] S. Bai, X. Bai, W. Liu, and F. Roli. Neural shape codes for 3D model retrieval. *PRL*. 2

[4] X. Bai, W. Liu, and Z. Tu. Integrating contour and skeleton for shape classification. In *ICCV Workshops*, 2009. 5, 6, 7

[5] S. Biasotti, A. Cerri, M. Abdelrahman, M. Aono, A. B. Hamza, M. El-Melegy, A. Farag, V. Garro, A. Giachetti, D. Giorgi, et al. SHREC14 track: Retrieval and classification on textured 3D models. In *Eurographics Workshop on 3DOR*, 2014. 6, 7

[6] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015. 1, 2, 6

[7] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *CGF*, 2015. 2, 3

[8] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM TOG*, 2011. 1, 3, 7

[9] S. Bu, Z. Liu, J. Han, J. Wu, and R. Ji. Learning high-level feature by deep belief networks for 3D model retrieval and recognition. *TMM*, 2014. 2

[10] D. Carlson, E. Collins, Y. P. Hsieh, L. Carin, and V. Cevher. Preconditioned spectral descent for deep learning. In *NIPS*, 2015. 6

[11] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[12] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 1, 3, 8

[13] C. Chen, D. Carlson, Z. Gan, C. Li, and L. Carin. Bridging the gap between stochastic gradient mcmc and stochastic optimization. In *AISTATS*, 2016. 5

[14] C. Chen, N. Ding, and L. Carin. On the convergence of stochastic gradient MCMC algorithms with high-order integrators. In *NIPS*, 2015. 2, 3

[15] T. Chen, E. B. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *ICML*, 2014. 2, 3, 4

[16] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011. 5

[17] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*. 3

[18] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*, 2014. 3

[19] F. Dellaert, S. M. Seitz, S. Thrun, and C. Thorpe. Feature correspondence: A Markov chain Monte Carlo approach. In *NIPS*, 2000. 2

[20] N. Ding, Y. Fang, R. Babbush, C. Chen, R. D. Skeel, and H. Neven. Bayesian sampling using stochastic gradient thermostats. In *NIPS*, 2014. 2, 3

[21] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong. 3D deep shape descriptor. In *CVPR*, 2015. 2, 3

[22] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015. 2

[23] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *AISTATS*, 2011. 3

[24] I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013. 7

[25] R. Henao, Z. Gan, J. Lu, and L. Carin. Deep Poisson factor modeling. In *NIPS*, 2015. 7

[26] J. M. Hernández-Lobato and R. P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *ICML*, 2015. 1, 2, 3

[27] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012. 5

[28] G. E. Hinton and R. R. Salakhutdinov. Replicated softmax: an undirected topic model. In *NIPS*, 2009. 7

[29] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 2, 5

[30] H. Ishwaran and J. S. Rao. Spike and slab variable selection: frequentist and Bayesian strategies. *Annals of Statistics*, 2005. 5

[31] Q. Ke and Y. Li. Is rotation a nuisance in shape recognition? In *CVPR*, 2014. 2

[32] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 3, 4

[33] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *NIPS*, 2015. 2

[34] A. Korattikara, V. Rathod, K. Murphy, and M. Welling. Bayesian dark knowledge. *NIPS*, 2015. 1, 5

[35] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009. 3, 5, 8

[36] H. Larochelle and S. Lauly. A neural autoregressive topic model. In *NIPS*, 2012. 7

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 2015. 3

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. 3, 5, 7

[39] C. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *AISTATS*, 2015. 7

[40] M. W. Lee and I. Cohen. Human upper body pose estimation in static images. In *ECCV*. 2

[41] C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In *AAAI*, 2016. 2, 3, 4

[42] C. Li, C. Chen, K. Fan, and L. Carin. High-order stochastic gradient thermostats for Bayesian learning of deep models. In *AAAI*, 2016. 2

[43] C. Li and A. B. Hamza. Intrinsic spatial pyramid matching for deformable 3D shape retrieval. *IJMIR*, 2013. 7

[44] C. Li and A. B. Hamza. A multiresolution descriptor for deformable 3D shape retrieval. *The Visual Computer*, 2013. 7

[45] C. Li, M. Ovsjanikov, and F. Chazal. Persistence-based structural recognition. In *CVPR*, 2014. 1

[46] C. Li, X. You, A. Ben Hamza, W. Zeng, and L. Zhou. Distinctive parts for shape classification. In *ICWAPR*, 2011. 7

[47] K.-L. Lim and H. K. Galoogahi. Shape classification using local and global features. In *PSIVT*, 2010. 7

[48] M. Lin, Q. Chen, and S. Yan. Network in network. *ICLR*, 2014. 7

[49] H. Ling and D. W. Jacobs. Shape classification using the inner-distance. *TPAMI*, 2007. 6, 7

[50] Y. A. Ma, T. Chen, and E. B. Fox. A complete recipe for stochastic gradient MCMC. *NIPS*, 2015. 3

[51] D. J. C. MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 1992. 1, 3

[52] B. M. Marlin, K. Swersky, B. Chen, and N. D. Freitas. Inductive principles for restricted Boltzmann machine learning. In *AISTATS*, 2010. 7

[53] D. Maturana and S. Scherer. Voxnet: A 3D convolutional neural network for real-time object recognition. *IROS*, 2015. 8

[54] J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *NIPS*, 2008. 7

[55] R. M. Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995. 1, 3

[56] R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of MCMC*, 2011. 4

[57] S. Patterson and Y. W. Teh. Stochastic gradient Riemannian Langevin dynamics on the probability simplex. In *NIPS*, 2013. 4

[58] D. Pickup, X. Sun, P. L. Rosin, R. Martin, Z. Cheng, Z. Lian, M. Aono, A. B. Hamza, A. Bronstein, M. Bronstein, et al. SHREC14 track: Shape retrieval of non-rigid 3D human models. *Eurographics Workshop on 3DOR*, 2014. 6, 7

[59] Y. Pu, X. Yuan, A. Stevens, C. Li, and L. Carin. A deep generative deconvolutional image model. In *AISTATS*, 2016. 3

[60] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951. 1, 4

[61] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 1, 8

[62] B. Shi, S. Bai, Z. Zhou, and X. Bai. DeepPano: Deep panoramic representation for 3D shape recognition. *SPL*. 8

[63] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 3, 8

[64] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2, 5

[65] N. Srivastava, R. R. Salakhutdinov, and G. E. Hinton. Modeling documents with deep Boltzmann machines. *UAI*, 2013. 7

[66] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015. 2, 8

[67] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *CGF*, 2009. 2

[68] K. B. Sun and B. J. Super. Classification of contour shapes using class segment sets. In *CVPR*, 2005. 7

[69] Y. W. Teh, A. H. Thiéry, and S. J. Vollmer. Consistency and fluctuations for stochastic gradient Langevin dynamics. 2015. 2

[70] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop. *Coursera: Neural Networks for Machine Learning*, 2012. 3, 4

[71] Z. Tu and S.-C. Zhu. Image segmentation by data-driven Markov chain Monte Carlo. *TPAMI*, 2002. 2

[72] A. Vehtari, J. Ojanen, et al. A survey of Bayesian predictive methods for model assessment, selection and comparison. *Statistics Surveys*, 2012. 3

[73] S. J. Vollmer, K. C. Zygalakis, and Y. W. Teh. (Non-)asymptotic properties of stochastic gradient Langevin dynamics. Technical Report arXiv:1501.00438, 2015. 5

[74] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using DropConnect. In *ICML*, 2013. 2, 5

[75] F. Wang, L. Kang, and Y. Li. Sketch-based 3D shape retrieval using convolutional neural networks. In *CVPR*, 2015. 2

[76] S. Wang and C. Manning. Fast Dropout training. In *ICML*, 2013. 5

[77] X. Wang, B. Feng, X. Bai, W. Liu, and L. J. Latecki. Bag of contour fragments for robust shape classification. *Pattern Recognition*, 2014. 6, 7

[78] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *ICML*, 2011. 1, 2, 3, 4

[79] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015. 2, 3, 8

[80] J. Xie, Y. Fang, F. Zhu, and E. Wong. Deep shape: Deep learned shape descriptor for 3D shape matching and retrieval. In *CVPR*, 2015. 2

[81] M. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *ICLR*, 2013. 3, 7

[82] S. Zhu, F. Dellaert, and Z. Tu. Markov chain Monte Carlo for computer vision. *A tutorial at ICCV*, 2005. 2