Stochastic Bouncy Particle Sampler for Bayesian Neural Networks

Ethan Goan [1] **Clinton Fookes** [1]

Abstract

Due to their perceived computational cost, Markov chain Monte Carlo (MCMC) methods have seen little recent application to Bayesian Neural Networks (BNNs). We show here that new continuous time MCMC methods can alleviate this cost, and allow for efficient sampling within BNNs. We propose a simplified version of the Stochastic Bouncy Particle Sampler, making it suitable to perform inference on both dense and convolutional networks. We introduce a new Python package that leverages modern GPU acceleration, allowing for flexible posterior distributions to be found without prohibitive time and compute restrictions.

1. Introduction

Variational inference (VI) (Graves, 2011; Gal & Ghahramani, 2016; Blundell et al., 2015; Rezende et al., 2014; Blum et al., 2015; Wen et al., 2018) and optimization based methods (Wilson & Izmailov, 2020; Maddox et al., 2019) have recently dominated the field of Bayesian Neural Networks (BNNs). The appeal of these methods for BNNs can be linked to their amenability to performing inference with existing optimization schemes, and their generally lower compute cost. VI permits mini-batch sampling during inference, whilst traditional Markov chain Monte Carlo (MCMC) methods require evaluating the complete likelihood. The reduced compute cost over traditional MCMC comes at the expense of inference accuracy. VI places an assumed distribution of the parameters of a model, which is then optimized to best match the true posterior by maximizing the evidence lower bound (ELBO). The most common form of posterior approximation is a mean-field approach, where each parameter is assumed to be independent of others and Gaussian distributed, yet both of these assumptions are often violated for BNNs. This is illustrated in Figure 1, where we see strong correlation between the parameters and multi-modal estimated marginal distribution of a single parameter.

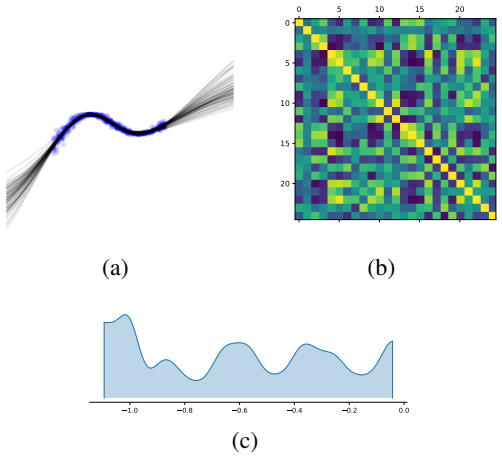


Figure 1. Example of correlations between the parameters in the first layer of a BNN for a simple regression task. Plot (a) samples of the predictive posterior from the proposed method, (b) correlation between all parameters on the first layer, (c) kernel density estimate for a single parameter.

Recent work has addressed sub-sampling schemes for MCMC based inference such as Stochastic Gradient Langevin Dynamics (SGLD), though it has been found that naive sampling introduces bias away from the target posterior (Teh et al., 2016; Vollmer et al., 2015). This work explores a new set of "exact" inference methods to address this issue based on Piecewise Deterministic Markov Processes (PDMPs) (Davis, 1984), in particular, the Bouncy Particle Sampler (BPS) (Bouchard-Côté et al., 2018). The BPS is able to maintain the true posterior as its invariant distribution during inference whilst permitting sub-sampling of the likelihood. This property is attractive for BNNs which are typically high dimensional in terms of their parameters, and the increasingly large data sets needed for complex perception tasks. However, the application of the BPS for exact inference is limited to a small class of models, as it requires a strict upper bound on the event rate that dictates when deterministic dynamics are updated.

A recently proposed method (Pakman et al., 2017) introduced the Stochastic Bouncy Particle Sampler (SBPS), a

*Equal contribution [1]School of Electrical Engineering and Robotics, Queensland University of Technology. Correspondence to: Ethan Goan <ej.goan@qut.edu.au>.

Presented at the ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning., Copyright 2021 by the author(s).

data dependent scheme that alleviates these concerns and allows for a general approach for applying the BPS for approximate Bayesian inference. Whilst suitable for approximate inference, the applicability of the SBPS in its current form is limited, due to requiring the calculation of gradients with respect to each individual input. As such, experimentation within (Pakman et al., 2017) restricts themselves to a BNN with 192 parameters and shows no results in terms of its predictive performance. We address these limitations through the following contributions,

- We simplify the SBPS method to allow for more efficient sampling, allowing for application to more complex models.
- We develop a GPU accelerated package for applying these methods to general models, allowing for evaluation of BNNs applied to challenging classification tasks.

The software package accompanying this work allows for a simple and optimised method to build models and perform approximate inference using the SBPS. MCMC methods have often been seen as computationally prohibitive for models with a large number of parameters or where modern large data sets are used. It is hoped that this work will demonstrate that approximate inference using MCMC like approaches for BNNs can be both practically feasible and offer insightful results.

2. Bouncy Particle Sampler

Following the description from (Fearnhead et al., 2018), PDMPs are defined by three key components: continuous, piecewise deterministic dynamics, an event rate and the transition kernel. The piecwise dynamics are designed to be simple enough to allow for simulation. These simple dynamics are updated according to the event rate, with the transition kernel dictating how to update the current dynamics.

Piecewise dynamics for ω within BPS are given by linear segments $\omega(t) = \omega^i + \mathbf{v}^i t$ where t is a time variable, and $\mathbf{v}$ is an auxiliary velocity parameter with a known distribution ψ, and ω^i is the starting position for this segment. Here we use the uniform measure for v such that $\mathbb{S}^{d-1} := \{\mathbf{v} \in \mathbb{R}^d : |\mathbf{v}| = 1\}$. This trajectory will continue until an event time τ^i is sampled from the inhomogeneous Poisson Process (IPP) with event rate of,

$$\lambda(\omega(t), \mathbf{v}) = \max\{0, \nabla U(\omega) \cdot \mathbf{v}^i\}. \tag{1}$$

where, $U(\omega) = -\log\Big(p(\omega)p(\mathcal{D}|\omega)\Big)$. Upon these events, the velocity is updated, or the trajectory "bounces" on the plane orthogonal to the gradient,

$$\mathbf{v}^{i+1} = \mathbf{v}^i - 2\frac{\nabla U(\omega) \cdot \mathbf{v}^i}{\|\nabla U(\omega)\|_2^2}\nabla U(\omega) = R(\omega, \mathbf{v})\mathbf{v}. \tag{2}$$

The start of this new piecewise segment will connect with the end of the previous segment, ie. $\omega^{i+1} = \omega^i + \mathbf{v}^i\tau^i$. (Bouchard-Côté et al., 2018) show that these dynamics alone can fail to target the posterior, but can be corrected by including refresh events where the velocity is sampled from its reference distribution. The rate that these refreshes occur is dictated by a homogeneous Poisson process $\lambda(\tau_{ref})$, where τ_{ref} is a hyperparameter. The complete BPS is summarised in Appendix A.

The main challenge with implementation of the BPS sampler is sampling from the event rate in Eqn (1). Analytic sampling is available only for simple models, such as linear regression with a Gaussian likelihood. For more complex models, a general approach is to appeal to the thinning method of (Lewis & Shedler, 1979), which only requires us to sample from a PP with rate $\gamma(t)$ such that $\gamma(t) > \lambda(\omega(t), \mathbf{v})$ for all $t \geq 0$. The aim of the SBPS is to find a simple $\gamma(t)$ that would be suitable for thinning.

2.1. Stochastic Bouncy Particle Sampler

The SBPS (Pakman et al., 2017) begins by creating a linear approximation for $\nabla U(\omega) \cdot \mathbf{v}^i$ by,

$$\tilde{G} = \beta_1 t + \beta_0 + \epsilon, \qquad \epsilon \sim N(0, c_t^2), \tag{3}$$

where c_t^2 is the approximated noise in $\nabla U(\omega) \cdot \mathbf{v}^i$. A prior is placed over the parameter $\beta_1 \sim \mathcal{N}(0, \sigma^2)$, and the posterior is found $p(\beta|t_i, \tilde{G}_i, c_{t_i}^2)$ using multiple samples from G at different times as the data points. From this posterior, an approximate upper bound on the event rate $\lambda(\omega(t), \mathbf{v})$ is found as,

$$\gamma(t) = \hat{\beta}_1 t + \hat{\beta}_0 + k\rho(t), \tag{4}$$

where $\hat{\beta}$ is the posterior mean, $\rho = \mathbf{x}\Sigma\mathbf{x}^T$, $\mathbf{x} = (t, 1)$, Σ is the posterior covariance and k is a positive hyperparameter. Following (Lewis & Shedler, 1979), event times are proposed from $\gamma(t)$, and accepted with probability $\min(1, [\lambda(\omega(t), \mathbf{v})]_+/\gamma(t))$, where $[u]_+ = \max(0, u)$.

2.2. Efficient SBPS

The efficiency of the above SBPS scheme is inhibited by the calculation of the variance parameter c_t^2. In the original SBPS, c_t^2 is calculated as[1],

$$c_t^2 = \text{Var}[\mathbf{v} \cdot \nabla \log p(\mathcal{D}|\omega)]. \tag{5}$$

[1]In (Pakman et al., 2017), and additional finite population correction term is included when subsampling the likelihood with mini-batches.

(Pakman et al., 2017) rely on computing this variance empirically. This requires finding the gradient of $\nabla \log p(\mathcal{D}|\omega)$ with respect to each data point individually. This can be an expensive operation for large models seen in BNNs or large batch sizes. For a batch size of N, computing the variance this way would require N individual gradient calls with automatic differentiation software. To reduce this cost, in the SBPS these gradient terms are found analytically. Whilst addressing the computational cost, this unfortunately severely limits the applicability of the algorithm. To address this, we simply replace c_i^2 with a constant hyperparameter. We demonstrate with this minor change, we can still generate samples efficiently. More importantly, this allows the application of the SBPS to more complex models like convolutional BNNs. To distinguish between the original SBPS, we title the application of this modification the efficient SBPS (eSBPS).

3. Experiments

We validate the performance of the eSBPS method on a number of synthetic and real data sets for regression and classification. To analyse performance for predictive tasks, the predictive posterior needs to be evaluated. In this end, we discretize samples from the continuous trajectory to allow for Monte Carlo integration,

$$p(y^*|x^*, \mathcal{D}) = \int \pi(\omega) p(y^*|\omega, x^*) d\omega, \tag{6}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} p(y^*|\omega_i, x^*) \qquad \omega_i \sim \pi(\omega). \tag{7}$$

For our experiments, we discretize the trajectory of our posterior dynamics by sampling the midpoints between update events. Experimentation is first conducted on multiple small data sets to allow us to easily visualise the predictive performance and uncertainty in our models, followed by more difficult classification tasks with BCNNs on real data sets. To enable these experiments, we deliver a Python package built upon the Tensorflow Probability library (Dillon et al., 2017), allowing for GPU acceleration and graph construction of all our models to reduce compute time. Code can be found at https://github.com/ethangoan/eSBPS.

3.1. Regression

To visualise the predictive performance and our uncertainty estimates, regression tasks were formed on synthetic data sets. For these models, a two hidden layer network was used with 20 and 10 hidden units respectively, with both using Tanh activations. All parameters are given a standard Normal prior, and the likelihood variance is known and fixed. A MAP estimate was first found using stochastic optimisation before sampling commenced. A burn-in phase of 1,000 iterations was performed, followed by a sampling phase of 2,000 iterations. Examples of the predictive posterior distribution for these models on multiple data sets is shown in Figure 2. Results from these experiments indicate promising performance in terms of predictive reasoning, with low variance seen within the range of observed data and greater variance seen in areas as the distance from the observed samples is increased.

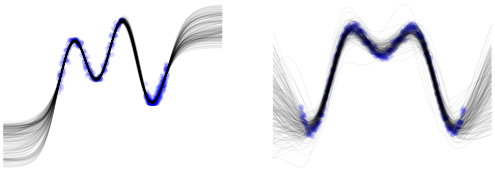


Figure 2. Examples of draws from the predictive posterior for two separate synthetic datasets using the eSBPS. Blue samples indicate data used during training, whilst the black lines indicate draws from the predictive posterior.

3.2. Binary Classification

To gain an understanding into uncertainty quality for classification tasks, we start with experimentation for logistic regression and binary classification using a dense BNN fit with the proposed eSBPS method. All parameters for these models are given a standard normal prior. The BNN used consisted of two hidden layers with 10 units each, both with Tanh activations applied. These results are shown in Figure 3[2].

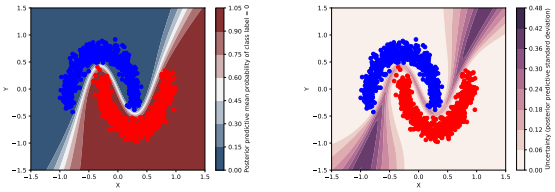


Figure 3. Binary classification with logistic regression (top) and dense BNN (bottom). The left column shows the mean classification probability, and the right column shows the variance in the predictive posterior.

These tests indicate promising performance in terms of the predictive accuracy of eSBPS and for the quality of its uncertainty estimates. To further demonstrate classification performance, we move to larger and more complicated models for performing classification on real world data sets.

[2]Code for generating figures adapted from https://twiecki.io/blog/2016/06/01/bayesian-deep-learning/

Table 1. Classification accuracy for convolutional networks fit initialized with MAP estimate, and inference performed with eSBPS. Sampling time does not include time to find MAP estimate.

DATA SET	ACCURACY	SAMPLING TIME (SECONDS)
MNIST	97.05%	376
FASHION MNIST	84.23%	307
SVHN	82.55%	1475
CIFAR-10	61.63%	1512

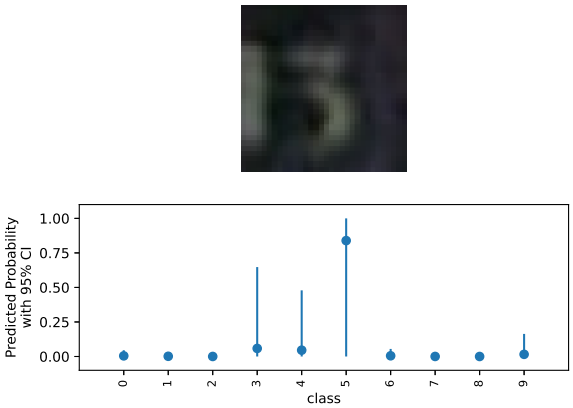
Table 2. Classification accuracy for convolutional networks with MAP estimate.

DATA SET	ACCURACY	FIT TIME (SECONDS)
MNIST	99.07%	157
FASHION MNIST	90.98%	153
SVHN	91.02%	679
CIFAR-10	63.17%	657

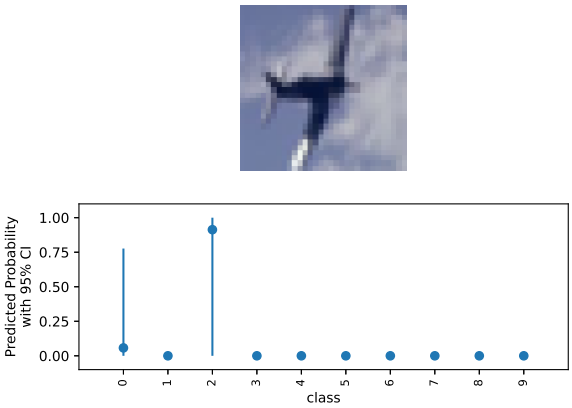
3.3. Classification

We evaluate the performance of the eSBPS on the popular MNIST (LeCun et al., 1998), Fashion MNIST (Xiao et al., 2017), CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) data sets. Descriptions of the convolutional networks can be found in Appendix B.

For each eSBPS model, we first find a MAP estimate using stochastic optimisation. This is followed by 1,000 warm-up iterations and 2,000 posterior samples. A summary of the classification accuracy for each method is listed in Table 1, where the quadratic error loss is used for final classification. Results from traditional point estimates of MAP approximations are shown in Table 2. Examples of difficult to classify samples are shown in Figure 4, with additional samples shown in Appendix C. These results demonstrate comparable performance of the probabilistic models when compared with the point estimate networks. Although the MAP estimates are faster to fit and achieve higher accuracy throughout, unlike the Bayesian counterparts they are unable to sufficiently reason about uncertainty in it's predictions. As shown in Figure 4, we see that we can often gain meaningful and intuitive uncertainty estimates for difficult to classify samples.



(a) 'Three' from SVHN misclassified as 'five'.



(b) 'Airplane' from CIFAR-10 misclassified as 'bird'.

Figure 4. Example of difficult to classify samples and the predictive distributions found with eSBPS. Points represent the mean prediction for the sample. Best viewed on a computer screen.

4. Conclusion

This paper shows that advances in MCMC research can permit flexible posterior inference within BNNs for regression and classification tasks. Whilst offering meaningful uncertainty estimation in predictive distributions, the current predictive performance of these models is less than that of point estimate models. We hypothesise a few possible causes of this issue to highlight for future research. Firstly, we found the selection of the prior distribution to be crucial for meaningful inference. Secondly, our eSBPS provides a simple means for approximate sampling of event times at the expense of flexibility. The event rate for BNNs can be highly non-linear, making it difficult for a linear approximation to guarantee a strict upper bound on the true event rate. Finally, it has also been suggested that the use of applying preconditioning to the velocity can improve sampling efficiency (Pakman et al., 2017; Bierkens et al., 2020; Bertazzi & Bierkens, 2020). We highlight these areas of interest for future research as ways to further improve upon PDMP based methods to make application to BNNs more accurate, whilst also reducing their computational load.

References

Bertazzi, A. and Bierkens, J. Adaptive schemes for piecewise deterministic monte carlo algorithms. *arXiv preprint*

arXiv:2012.13924, 2020.

Bierkens, J., Grazzi, S., Kamatani, K., and Roberts, G. The boomerang sampler. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 908–918. PMLR, 2020.

Blum, A., Haghtalab, N., and Procaccia, A. D. Variational dropout and the local reparameterization trick. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2575–2583, 2015.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622. PMLR, 2015.

Bouchard-Côté, A., Vollmer, S. J., and Doucet, A. The bouncy particle sampler: A nonreversible rejection-free markov chain monte carlo method. *Journal of the American Statistical Association*, 113(522):855–867, 2018.

Clevert, D., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Davis, M. H. Piecewise-deterministic markov processes: a general class of non-diffusion stochastic models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 46(3):353–376, 1984.

Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., Patton, B., Alemi, A., Hoffman, M., and Saurous, R. A. Tensorflow distributions. *arXiv preprint arXiv:1711.10604*, 2017.

Fearnhead, P., Bierkens, J., Pollock, M., Roberts, G. O., et al. Piecewise deterministic markov processes for continuous-time monte carlo. *Statistical Science*, 33(3):386–412, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 1050–1059. JMLR.org, 2016.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. Maxout networks. In *International conference on machine learning*, pp. 1319–1327. PMLR, 2013.

Graves, A. Practical variational inference for neural networks. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pp. 2348–2356, 2011.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

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

Lewis, P. W. and Shedler, G. S. Simulation of nonhomogeneous poisson processes by thinning. *Naval research logistics quarterly*, 26(3):403–413, 1979.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. A simple baseline for bayesian uncertainty in deep learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 13132–13143, 2019.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Pakman, A., Gilboa, D., Carlson, D. E., and Paninski, L. Stochastic bouncy particle sampler. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2741–2750. PMLR, 2017.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1278–1286. JMLR.org, 2014.

Sermanet, P., Chintala, S., and LeCun, Y. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pp. 3288–3291. IEEE Computer Society, 2012.

Teh, Y. W., Thiery, A. H., and Vollmer, S. J. Consistency and fluctuations for stochastic gradient langevin dynamics. *Journal of Machine Learning Research*, 17, 2016.

Vollmer, S. J., Zygalakis, K. C., et al. (non-) asymptotic properties of stochastic gradient langevin dynamics. *arXiv preprint arXiv:1501.00438*, 2015.

Wen, Y., Vicol, P., Ba, J., Tran, D., and Grosse, R. B. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

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

Zeiler, M. D. and Fergus, R. Stochastic pooling for regularization of deep convolutional neural networks. In Bengio, Y. and LeCun, Y. (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Conference Track Proceedings*, 2013.

A. BPS Algorithm

We summarise the global BPS algortihm from (Bouchard-Côté et al., 2018) in Algorithm 1.

Algorithm 1 Global Bouncy Particle Sampler (Bouchard-Côté et al., 2018)

repeat
  $\tau_{\text{bounce}} \sim \text{IPP}(\lambda(\omega(t), \mathbf{v})$ {Simulate bounce time}
  $\tau_{\text{ref}} \sim \text{PP}(\lambda_{\text{ref}})$ {Simulate refresh time}
  $\tau^{i+1} = \min(\tau_{\text{bounce}}, \tau_{\text{ref}})$
  $\omega^{i+1} = \omega^i + \mathbf{v}^i \tau^{i+1}$
  if $\tau == \tau_{\text{ref}}$ **then**
    $\mathbf{v}^{i+1} \sim \psi$ {Refresh event}
  else
    $\mathbf{v}^{i+1} = R(\omega^{i+1}, \mathbf{v}^i)\mathbf{v}^i$ {Bounce event}
  end if
until $i \geq$ Number Required Events Sampled

B. CNN Description

We describe here the networks used for the experiments in Section 3.3. For MNIST and Fashion-MNIST datasets, the LeNet5 architecture was used (LeCun et al., 1989), though the ReLU activations were replaced with ELU units (Clevert et al., 2016).

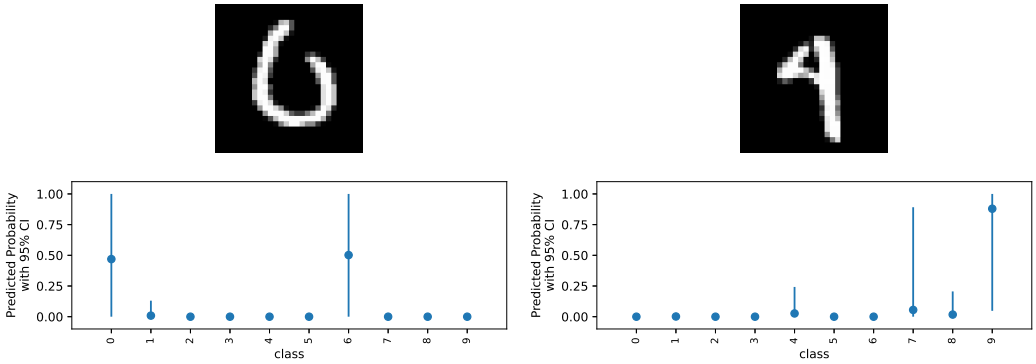
For the SVHN and CIFAR-10 datasets, a larger model was required to achieve sufficient predictive performance. A description for this model is shown in Table 3. For this network, ELU activations were used after each convolutional and dense layer, excluding the final dense layer where softmax is used. Following (Goodfellow et al., 2013), CIFAR-10 data was preprocessed using global-contrast normalization and ZCA whitening. SVHN data was preprocessed using the method proposed in (Sermanet et al., 2012; Zeiler & Fergus, 2013), where local and global contrast normalization is applied. For the MNIST and Fashion-MNIST datasets, samples were standardized within ranges of $[-1, 1]$. All parameters were given a standard Normal prior.

Table 3. Architecture used for SVHN and CIFAR-10 datasets.

CONV 3×3, 16 FILTERS
CONV 3×3, 32 FILTERS
2×2 MAX POOL
CONV 3×3, 32 FILTERS
CONV 3×3, 64 FILTERS
2×2 MAX POOL
DENSE - 256
DENSE - NUMBER CLASSES
SOFTMAX

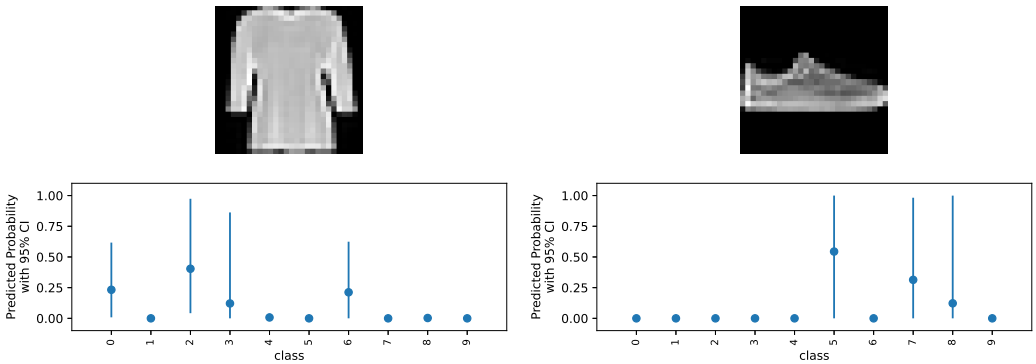
C. Additional Examples of Difficult to Classify Samples

We provide additional examples of difficult to classify samples from the MNIST, Fashion MNIST, SVHN and CIFAR-10 dataset in Figure 5. We see that when fit with the eSBPS, meaningful uncertainty estimates can be obtained for misclassified samples.



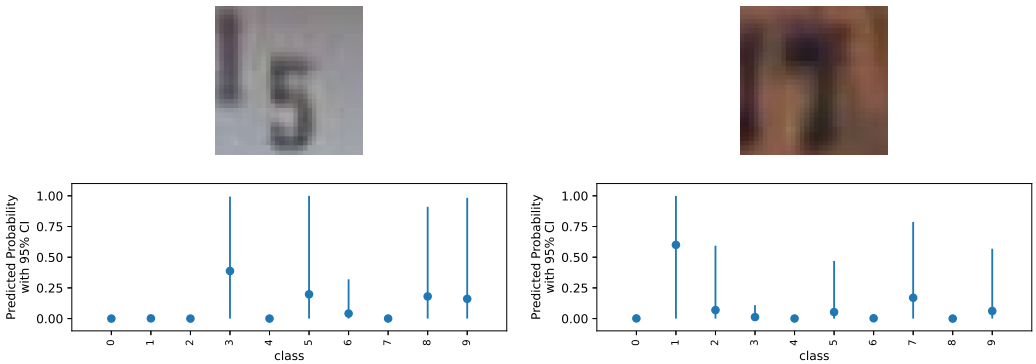
(a) 'Zero' from MNIST misclassified as 'six'.

(b) 'Four' from MNIST misclassified as 'nine'.



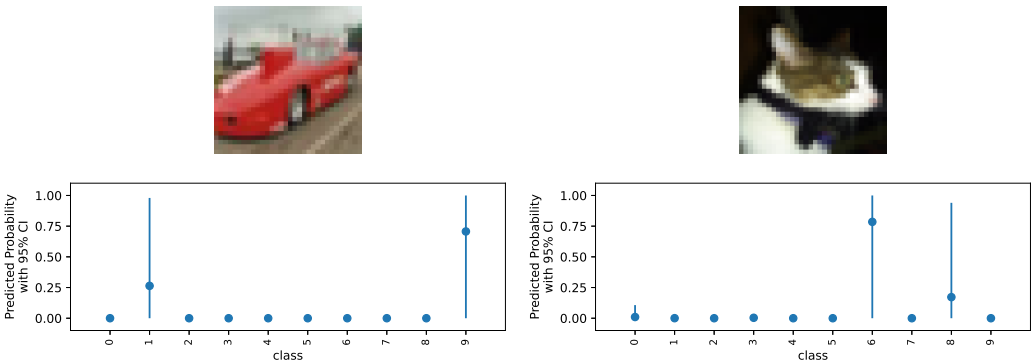
(c) 'Top' from Fashion MNIST classified as 'pullover'.

(d) 'Sneaker' from Fashion MNIST misclassified as 'sandal'.



(e) 'Five' from SVHN misclassified as 'three'.

(f) 'Seven' from SVHN misclassified as 'one'.



(g) 'Automobile' from CIFAR-10 misclassified as 'truck'.

(h) 'Cat' from CIFAR-10 misclassified as 'frog'.

Figure 5. Additional examples of difficult to classify samples and the predictive distributions found with eSBPS. Best viewed on a computer screen.