Analysis of priors for Multiplicative Normalizing Flows in Bayesian neural networks

Akash R. Komarlu Narendra Gupta
University of Amsterdam
akashrajkn@gmail.com

Emiel Hoogeboom
University of Amsterdam
e.hoogeboom@gmail.com

Abstract

In this paper, we explore Multiplicative Normalizing flows in Bayesian neural networks with different prior distributions over the network weights. The prior over the parameters can not only influence how the network behaves, but can also affect the uncertainty calibration and computational efficiency. The Bayesian framework also offers an added advantage of compression when sparsity-inducing priors are used. We experiment with uniform, Cauchy, log uniform, Gaussian, and standard Gumbel priors on predictive accuracy and predictive uncertainty.

1 Introduction

In recent years Deep Neural Networks (DNNs) have revolutionized the field of artificial intelligence [1]. Despite their robustness and expressivity, vanilla DNNs trained with maximum likelihood or MAP procedures have two major shortcomings: over-fitting when limited data is available and over-confidence in predictions. In real world applications such as predicting whether a patient has a disease or not, a high confidence for wrong predictions could be fatal.

The Bayesian framework naturally addresses the aforementioned problems. Instead of optimizing a point estimate for the parameters (equivalent to delta distributions), a posterior distribution over the weights is computed. A network trained with `MNIST` predicts a wrong label with high confidence when adversarial examples are used. A Bayesian neural network on the other hand gives a more realistic predictive distribution with low confidence for unseen data points. Because the true posterior is almost always intractable, approximations to the posterior distribution are often used. Some of the methods include Markov Chain Monte Carlo with Hamiltonian Dynamics [2], distilling Stochastic Gradient Descent with Langevin Dynamics or deterministic techniques such as variational inference [3, 4].

In this paper, we use the Multiplicative Normalizing Flows (MNF) architecture proposed in [5]. Posterior distribution over the weight matrices is approximated by using a stochastic gradient variational inference [6]. By implementing auxiliary random variables and normalizing flows, the flexibility of the approximate posterior distribution can be improved by a large extent. The remaining part of the paper is organized as follows. Section 2 gives an overview of Bayesian Neural networks and the MNF architecture. In section 3 we outline the different experiments conducted and discuss the results, conclusion and future work thereafter. In appendix, we report KL divergence terms between the various priors and Gaussian distribution.

2 Model

2.1 Bayesian Neural Networks

In Bayesian Neural Networks (BNN), the task of training a network is seen as a problem of inference [2, 7]. Bayes' theorem is used to assign a probability density to each point w in the parameter space of the neural network. This posterior predictive distribution can be used to obtain better uncertainties about the model.

Let $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$ be the data observed, $\{x^*, y^*\}$ be the new data point. Due to the nature of the architecture, the posterior does not have an simple analytical form. Stochastic gradient variational inference [6, 8] is adopted to obtain a posterior approximation over the weight matrices. Let $p(w)$ and $q_\phi(w)$ denote prior and approximate posterior distributions over the weight parameter w. Using Jensen's inequality, we can write the data log likelihood marginalized with respect to weights as:

$$\begin{aligned} \log p(y^*|x^*) &= \log \mathbb{E}_{q_\phi(w)} \left[\frac{p(y^*|x^*, w)p(w|\mathcal{D})}{q_\phi(w)} \right] \\ &\geq \mathbb{E}_{q_\phi(w)} \left[\log \frac{p(y^*|x^*, w)p(w|\mathcal{D})}{q_\phi(w)} \right] \end{aligned} \tag{1}$$

Using equation 1, we can derive the lower bound:

$$\begin{aligned} \mathcal{L}(\phi) &= \mathbb{E}_{q_\phi(w)} \left[\log p(y^*|x^*, w)\right] + \mathcal{KL}(q_\phi(w)||p(w|\mathcal{D}) \\ &= \mathbb{E}_{q_\phi(w)} \left[\log p(y^*|x^*, w)p(w|\mathcal{D})\right] + \mathbb{H}[q_\phi(w)] \end{aligned} \tag{2}$$

where ϕ denotes the parameters of the variational posterior, $\mathbb{H}$ denotes the entropy and $\mathcal{KL}$ is the Kullback-Leibler divergence.

By reparameterizing the marginal log-likelihood, approximate posterior inference can be treated as a straightforward optimization problem. The random sampling from continuous distributions $q(.)$ of the lower bound in equation (2) can be reparameterized [6] in terms of noise variables ϵ and deterministic functions $f(\phi, \epsilon)$:

$$\mathcal{L} = \mathbb{E}_{p(\epsilon)} \left[\log p(y^*|x^*, f(\phi, \epsilon)) + \log p(f(\phi, \epsilon)) - \log q_\phi(f(\phi, \epsilon))\right] \tag{3}$$

Off the shelf stochastic gradient ascent techniques can be used to optimize this problem. For BNNs, the straightforward approach of mean field with independent normal distributions for each weight limits the approximation capability. It can be improved by introducing more complex distributions or using normalizing flows [8] which is described in the next section.

2.2 Multiplicative Normalizing Flows

In this section, we briefly introduce Multiplicative normalizing flows [5]. The basic idea of a normalizing flow (NF) is as follows. Given a random variable, x that is invertible and follows a simple distribution $p(x)$, the probability density of the transformed variable y can be calculated as

$$p_T(y) = p(x) \left| det \left(\frac{\partial T(x)}{\partial x} \right) \right|^{-1} \tag{4}$$

where T denotes the invertible mapping. Complex tractable distributions can be constructed by applying the equation 4 successively. Applying a NF to vanilla BNN does not scale well with the weight matrix W. To ameliorate the detrimental effects of curse of dimensionality, auxiliary random variables [9] are used. The auxiliary random variables introduce latent variables into the posterior making it more flexible. The posterior is parameterized with the following process by exploiting multiplicative noise in neural networks:

$$z \sim q_\phi(z); \qquad W \sim q_\phi(W|z) \tag{5}$$

With this reparameterization, the approximate posterior becomes a mixture distribution,

$$q(W) = \int q(W|z)\, q(z)\, dz \tag{6}$$

Assuming fully factorized Gaussians for weights, we have

$$q_\phi(W|z) = \prod_{i=1}^{D_{in}} \prod_{j=1}^{D_{out}} \mathcal{N}(z_i \mu_{ij}, \sigma_{ij}^2) \qquad \text{for a fully-connected layer} \tag{7}$$

$$q_\phi(W|z) = \prod_{i=1}^{D_h} \prod_{j=1}^{D_w} \prod_{k=1}^{D_f} \mathcal{N}(z_k \mu_{ijk}, \sigma_{ijk}^2) \qquad \text{for a convolutional layer} \tag{8}$$

where D_{in} and D_{out} are input and output dimensionality respectively for the fully-connected layers and D_h, D_w, D_f represent height, width and number of filters for each kernel in the convolutional layers. In both the cases, local optima due to large variance gradients can be avoided by removing the effect of z on the variance. The advantage of this formulation is the smaller dimensionality of of the auxiliary random variable z compared to the weight vector W. NFs can be applied to $q(z)$ without worrying about the curse of dimensionality. The flexibility of the posterior can be improved by controlling the flow of q; this can achieve a better posterior approximation and eventually a better performance. For the type of posterior explained above, algorithms 1 and 2 describe a forward pass for a fully-connected layer and a convolutional layer respectively.

Algorithm 1: Forward propagation for each fully connected layer h

```
Require: $H, M_w, \Sigma_w$
    1. $Z_0 \sim q(z_0)$
    2. $Z_{T_f} = NF(Z_0)$
    3. $M_h = (H \odot Z_{T_f})M_w$
    4. $V_h = H^2 E_w$
    5. $W \sim \mathcal{N}(0,1)$
Return: $M_h + \sqrt{V_h} \odot E$
```

Algorithm 2: Forward propagation for each convolutional layer h

```
Require: $H, M_w, \Sigma_w$
    1. $z_0 \sim q(z_0)$
    2. $z_{T_f} = NF(Z_0)$
    3. $M_h = H * (M_w \odot \text{reshape}(z_{T_f}, [1,1,D_f]))$
    4. $V_h = H^2 * E_w$
    5. $W \sim \mathcal{N}(0,1)$
Return: $M_h + \sqrt{V_h} \odot E$
```

Masked real-valued non-volume preserving (realNVP) is used for the normalizing flow of $q(z)$. A major upside of using realNVP is that in addition to having a more flexible functional form, it allows evaluation of an exact and tractable log-likelihood during training. It defines a loss function in terms higher level features and does not require a fixed reconstruction cost; it is able to learn a semantic representation of the latent space whose dimension is equal to the input space [10].

$$\begin{aligned} m \sim &\text{Bern}(0.5); \quad h = \tanh(f(m \odot z_t)) \\ \mu = &g(h); \quad \alpha = \sigma(k(h)) \\ z_{t+1} = &m \odot z_t + (1-m) \odot (z_t \odot \alpha + (1-\alpha) \odot \mu) \\ \log &\left| \frac{\partial z_{t+1}}{\partial z_t} \right| = (1-m)^T \log \sigma \end{aligned} \tag{9}$$

where m is the mask, $\sigma(x) = \frac{1}{1+exp(-x)}$ and f, g, h are linear mappings. m denotes the binary mask which is sampled frequently. By introducing an auxiliary distribution $r(z|W)$ the lower bound can be made tractable,

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(z,W)} \left[\log p(y|x,z,W) + \log p(W) + \log r_\theta(z|W) - \log(q_\phi(W|z) q_\phi(z)) \right] \tag{10}$$

where θ are the parameters of the auxiliary distribution r. The performance of the model now depends on how well r approximates q.

2.3 Choice of priors

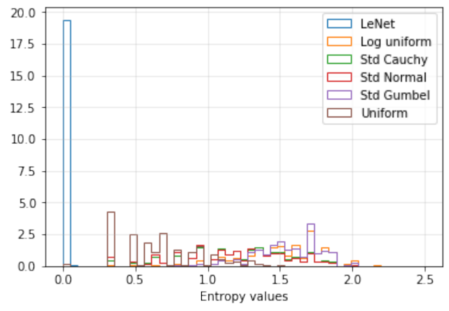
The $\mathcal{KL}$ divergence term between the prior and the approximate posterior is given below (for convenience we denote z_{T_f}, the final iterate of a normalizing flow as z):

$$-\mathcal{KL}(q(W)||p(W)) = \mathbb{E}_q \left[-\mathcal{KL}(q(W|z)||p(W)) + \log r(z|W) - \log q(z) \right] \tag{11}$$

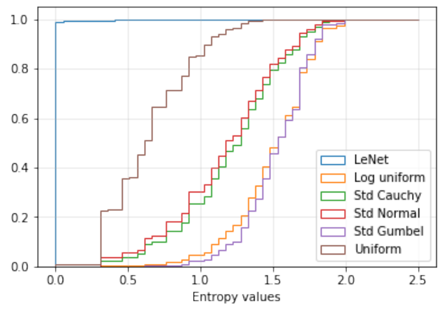
where r is the auxiliary distribution as described in the previous section. We compare the standard normal distribution used in the pilot study with various choices of priors. Appendix B and C give the derivation and $\mathcal{KL}$ divergence terms between the priors and approximate posterior.

3 Experimental results

In this section we present a collection of experiments comparing the performance of different priors on the `MNIST` dataset. For all the experiments, we use the scheme proposed in [5] which is



(a)



(b)

Figure 1: Entropy of the predictive distribution for the `notMNIST` test set. The left figure is the histogram of entropy values and the right figure shows the corresponding cumulative distribution function. LeNet often tends to make wrong predictions with a high confidence.

explained briefly. Adam [11] optimizer is used with the default hyperparameter setting. We used the LeNet [12] convolution architecture with ReLU activation. Log variances were initialized by sampling from $\mathcal{N}(-9, 0.001)$. We use flows of length 2 for $q(z)$ (with 50 hidden units for each step) and $r(z|W)$ (with 100 hidden units for each step). The predictive distribution is estimated from 100 posterior samples during testing (and 1 sample during training). For the log uniform prior α values are clipped at 1 during training.

3.1 Predictive performance

We evaluate the performance on classification tasks using `MNIST` dataset. Table 1 shows the train and test accuracy of the trained models. We observe that, in general MNF network models result in a greater accuracy than LeNet. The classification accuracy is similar for the different priors considered.

Model	Prior	Accuracy	
		validation	test
LeNet	–	0.957	0.956
MNF	Standard normal	0.987	**0.992**
	Uniform	0.990	0.991
	Log uniform	0.984	0.984
	Standard Cauchy	**0.990**	0.989
	Standard Gumbel	0.985	0.987

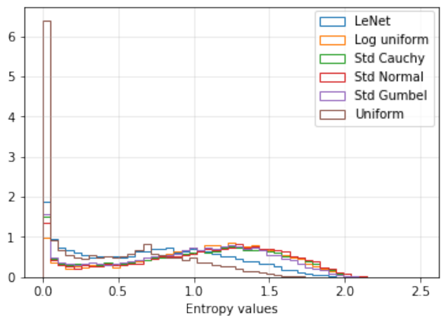
Table 1: Validation and test accuracy for `MNIST` test dataset.

3.2 Uncertainty evaluation

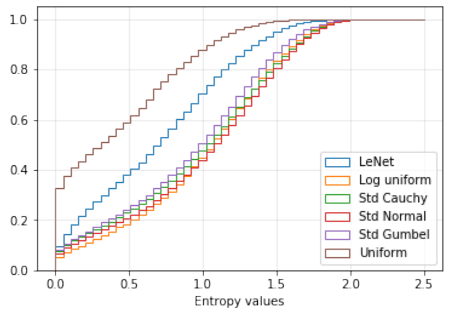
For the task of uncertainty evaluation, we use the trained network to predict the distribution for unseen classes. In real world applications, it is desirable to have higher uncertainty for unseen classes in test data. To this end, we train the models on the standard `MNIST` data. In addition to the regular test set with known classes, we also evaluate the performance of the models on a test set containing unknown classes. For the latter, we use the `notMNIST`[1] and `MNIST-rot`[2] test sets.

notMNIST The images in the `notMNIST` dataset are of the same dimensionality as `MNIST`, however they represent alphabets { A, B, ..., J }. For the test set, true conditional probabilities are not accessible. Since we know apriori that none of the classes correspond to those in the `MNIST` dataset,

[1]Available at `http://yaroslavvb.blogspot.co.uk/2011/09/notmnist-dataset.html`
[2]Available at `http://www.iro.umontreal.ca/~lisa/icml2007data`



(a)



(b)

Figure 2: Entropy of the predictive distribution for the `MNIST-rot` test set. The left figure is the histogram of entropy values and the right figure shows the corresponding cumulative distribution function.

we can expect the ideal predictive distribution to be uniform (this is also the maximum entropy distribution). Following the experiment in [13], we evaluate the entropy of the distribution and plot a histogram to evaluate the quality of the uncertainty estimates. Figure 1 (b) shows the empirical cumulative distribution function of the previously mentioned histogram. We observe that MNF with the log uniform and Gumbel priors are closest to the bottom right, which denotes that the probability of observing a high confidence is low. Contrary to this, the curve corresponding to LeNet which is very close to top-left of the plot often leads to overconfident wrong predictions. Standard Cauchy and standard normal priors give similar results as expected. In the case of a uniform distribution $\mathcal{U}(-5, 5)$, although the predictive accuracy is high, it performs worse compared to other priors for the task of predictive uncertainty.

MNIST-rot This dataset is generated from `MNIST` images rotated at an angle between 0 and 2π radians. In this test set, while the true classes are the same as `MNIST`, the input consists of previously 'known' and 'unknown' data points.

The type of uncertainty in the `MNIST-rot` dataset is different in the sense that target classes are known but the trained network does recognize the (adversarial) inputs. From Figure 2 (b) we can observe that a uniform prior makes predictions that are more over-confident than a simple LeNet. All other priors give a similar level of confidence that are not as confident as LeNet.

Prior	**Accuracy**	
	validation	test
Standard normal	0.630	0.628
Uniform	0.626	0.628
Log uniform	0.627	0.626
Standard Cauchy	**0.635**	**0.634**
Standard Gumbel	0.577	0.580

Table 2: Validation and test accuracy of the MNF model for the `CIFAR-10` dataset. Although the standard Cauchy prior performs better than others, MNF gives very low accuracy compared to LeNet.

3.3 Sparsity on weights

One way to address the problem of over-fitting and regularization is to induce sparsity to reduce the number of parameters in the deep neural network. In this experiment, following [14] we investigate the sparsity levels introduced by the log uniform and the Gumbel priors. Figure 3 shows the weight matrix of the second dense layer. The weights were pruned based on a threshold (σ) decided by resulting prediction accuracy. In the case of log uniform prior, in addition to the previously mentioned heuristic, weights corresponding to individual dropout, $\alpha_i = 1$ were pruned. For similar prediction accuracies (~ 0.95), the Gumbel prior induces a higher sparsity level of 78% whereas the log uniform prior induces a sparsity level of 47.12%. As expected the induced sparsity affects uncertainty estimates as the variance of the parameters is decreased.

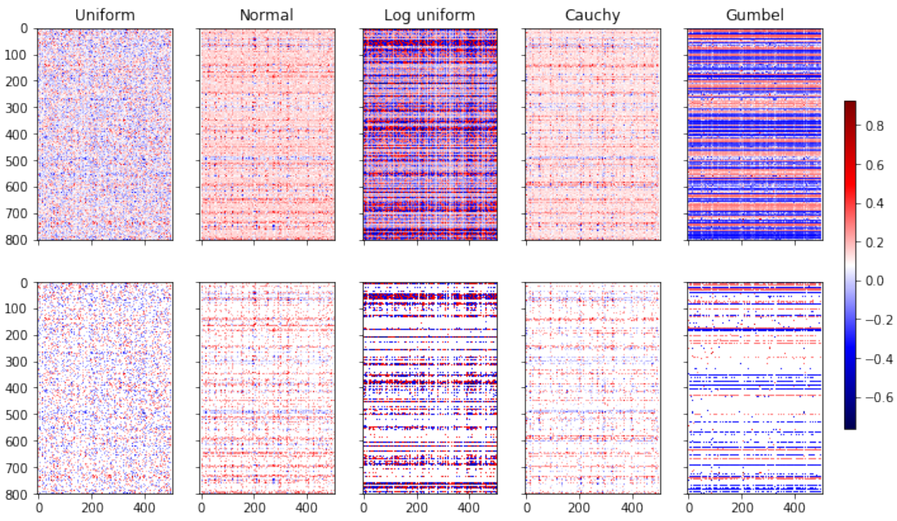


Figure 3: Heatmaps of the dense layer weights (means). Bottom row shows weight matrix after compression for accuracy of 95%.

3.3.1 Accuracy and compression

In real world applications, it is important for a model achievable accuracy for a given compression rate. In certain applications where processing power or memory is limited, it is desirable to get a very high compression rate with a small loss in accuracy. To this end we plot the accuracy vs compression (Figure 5) for each of the prior distributions. From the plots we observe (1) The effect it more pronounced in the dense layer 2 as compared to layer 1, and (2) the performance of uniform prior, $\mathcal{U}(-5, 5)$ is poor as expected. It reinforces the idea that choice of prior for certain tasks is important. The plots shown in Figure 5 were generated with compression only on one of the layers at a time. For the log uniform prior, the weights which had α clipped during training were also "dropped out" because they effectively were not trained.

In spite of its advantages, we found that it the compressed network behaved differently for the uncertainty evaluation. With compression (of 78%), the standard Gumbel prior did not perform well in the uncertainty evaluation task. It predicted wrong labels with a higher confidence. Figure 4 shows the predictive distribution for the compressed and uncompressed Gumbel prior. Compression

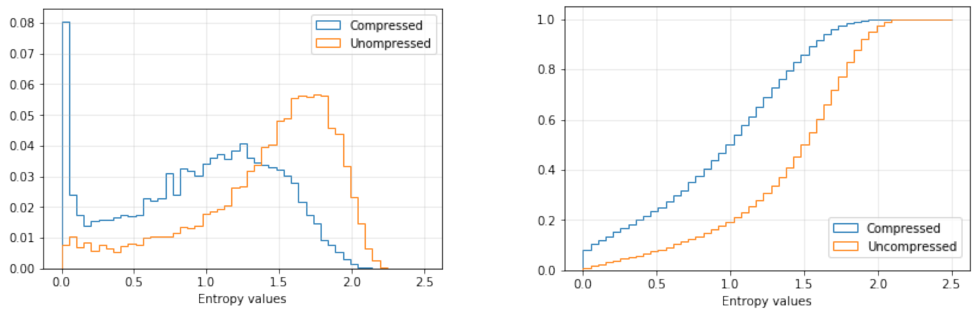
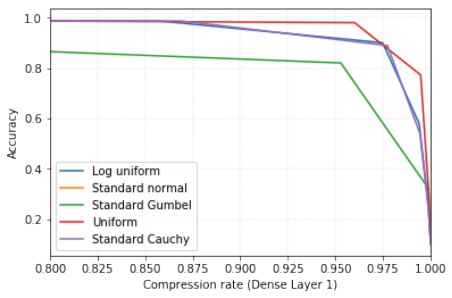


Figure 4: Entropy of the predictive distribution on the `MNIST-rot` test set using standard Gumbel prior. Left plot is the histogram of entropy values and the right shows the corresponding cumulative distribution.



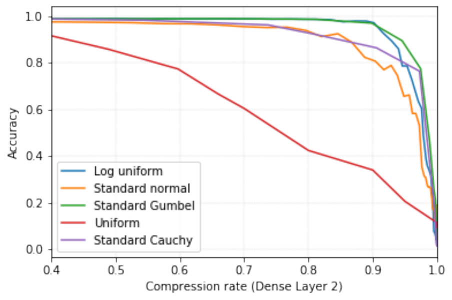
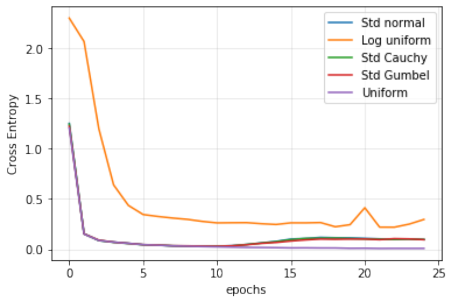
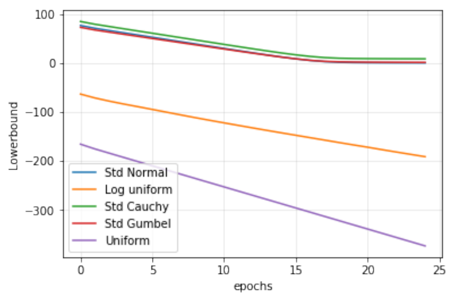


Figure 5: Accuracy of the network as compression rate is increased.



(a)



(b)

Figure 6: (a) Cross entropy and the (b) Lower bound in the training phase (over 25 epochs).

of weight values makes the network more confident for wrong predictions. We hypothesize that this is due to the induced sparsity in weight matrix. On further experiments, we could not establish such a relation for the log uniform prior. A future avenue of work would be to quantify the trade off between uncertainty evaluation and the selection of compression rate.

4 Discussion

We analyze the effect of prior distribution over the network performance, in particular we test the predictive accuracy and the predictive uncertainty. We have shown that while the predictive accuracy is similar, different priors can significantly improve the uncertainty estimates (in the case of uniform prior, it results in over-confident predictions). From the experiments, it can be observed that different prior distributions can capture different types of uncertainty. Among the chosen prior distributions, Gumbel and log uniform perform considerably better when test data contains unknown classes as in the case of `notMNIST`. When test data consists of adversarial examples as in the case of `MNIST-rot`, the priors give similar results. In addition to improving the uncertainty estimates, certain distributions give rise to sparsity in the weight matrix. By pruning the weights using a predefined threshold, network compression can be achieved without sacrificing the predictive accuracy.

Some avenues for future work include experimenting with more complex distributions. One of the discussions lacking in this paper is how to select prior distributions based on the task. It would also be interesting to investigate the effects of using informative prior distributions when training dataset is limited.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Radford M. Neal. Bayesian learning for neural networks. *PhD thesis*, 1995.

[3] Salimans Tim Kingma, Diederik P and Max Welling. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems*, 2015.

[4] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *arXiv preprint arXiv:1506.02142*, 2015.

[5] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[7] Pushpalatha C. Bhat and Harrison B. Prosper. Bayesian neural networks. *Statistical problems in Particle physics, astrophysics and Cosmology.*, 2005.

[8] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *Proceedings of the 32nd International Conference on Machine Learning*, 2016.

[9] Felix V Agakov and David Barber. An auxiliary vairational method. *International Conference on Neural Information Processing*, pages 561–566, 2004.

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. *ICLR*, 2007.

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[12] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick. Haffner. Gradient-based learning with matrix gaussian posteriors. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems 30*, pages 6402–6413. 2017.

[14] D. Molchanov, A. Anshuka, and D. Vetrov. Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*, 2017.

A Code Repository

The code and results for the project can be found at the following URL: `https://github.com/akashrajkn/waffles-and-posteriors`.

B Kullback-Leibler divergence

Prior	$-\mathcal{KL}$
Uniform, $\mathcal{U}(a,b)$	$\log\sigma^2 + \frac{1}{2}\log(2\pi) + \log(e) - \log(b-a)$
Standard normal	$\frac{1}{2}\left[-\log\sigma^2 + \sigma^2 + z_{T_f}^2\mu^2 - 1\right]$
Log uniform	$k_1\sigma(k_2 + k_3\log\tau) - \frac{1}{2}\log(1+\tau^{-1}) + C$
Standard Gumbel	$-\frac{1}{2}\log\sigma^2 + z_{T_f}\mu + \exp\{-z_{T_f}\mu + \frac{\sigma}{2}\} - \frac{1}{2}(1+\log(2\pi))$
Standard Cauchy	$\log\frac{\pi}{2} + \frac{1}{2}\left[-\log\sigma^2 + \sigma^2 + z_{T_f}^2\mu^2\right]$

Table 3: $\mathcal{KL}$ divergence terms used for the different priors.

C Derivation of KL Divergence term between log uniform and Gaussian

Let weight $w_i = |w_i|s_i$, where s_i is the sign bit and $|w_i|$ is the absolute value of w_i. The distribution of sign-bit on $\{-1,1\}$ is $Bernoulli(0.5)$. The log uniform distribution is defined as,

$$p(\log|w_i|) \propto c \qquad\qquad \{c \text{ is a constant}\}$$

This parameterization of the log uniform prior is known in statistics literature as Jeffreys' prior for the normal distribution. It is a non-informative prior for the normal distribution parameter space. Using the change of variables formula,

$$p(|w_i|) = p(\log|w_i|)\frac{d\log|w_i|}{d|w_i|} = \frac{p(\log|w_i|)}{|w_i|}$$

$$\log p(|w_i|) \propto \log c - \log|w_i|$$

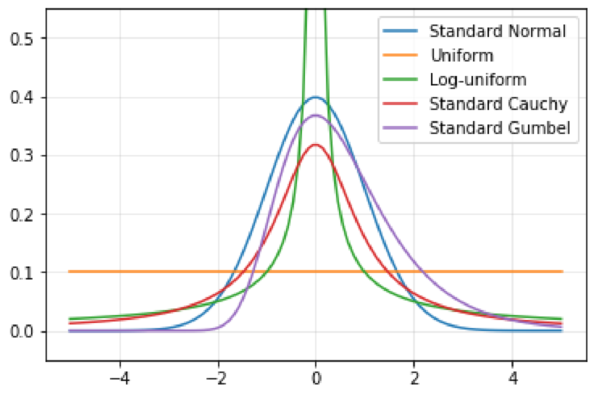


Figure 7: Prior distributions considered for the experiments.

The KL divergence between the two distributions can be broken down to two terms as,

$$-\mathcal{KL}(q(s_i)||p(s_i)) = \log(0.5) - Q(0)\log Q(0) - [1-Q(0)]\log[1-Q(0)]$$

$$-\mathcal{KL}(q(|w_i|)||p(|w_i|)) = \log c - \mathbb{E}_{q(w_i|w_i<0)}\left[\log\frac{q(w_i)}{Q(0)} - \log|w_i|\right] - \mathbb{E}_{q(w_i|w_i>0)}\left[\log\frac{q(w_i)}{1-Q(0)} - \log|w_i|\right]$$

$$\begin{aligned}-\mathcal{KL}\left(q(w_i)||p(w_i)\right) &= -\mathcal{KL}(q(s_i)||p(s_i)) - \mathcal{KL}(q(|w_i|)||p(|w_i|))\\ &= \log c + \log(0.5) - \mathbb{E}_{q(w_i|w_i<0)}\left[\log q(w_i) - \log|w_i|\right] - \mathbb{E}_{q(w_i|w_i>0)}\left[\log q(w_i) - \log|w_i|\right]\\ &= \log c + \log(0.5) - \mathcal{H}(q(w_i)) - \mathbb{E}_{q(w_i)}\log|w_i|\end{aligned}$$

Using the parameterization as specified above,

$$\begin{aligned}\epsilon_i &\sim \mathcal{N}(1,\alpha_i)\\ &= 1+\alpha_i\xi_i && \{\xi_i\sim\mathcal{N}(0,1)\}\\ w_i &= \theta_i\epsilon_i\\ &= \theta_i+\theta_i\alpha_i\xi_i\\ &= m_i+v_i\xi_i\end{aligned}$$

We now have, $\theta_i = m_i$ and $\alpha_i = \frac{v_i}{m_i}$. We use the approximation given in [14],

$$\mathcal{KL}(q(W)||p(W)) = k_1\sigma(k_2+k_3\log\tau) - \frac{1}{2}\log(1+\tau^{-1}) + C$$

where $k_1 = 0.63576, k_2 = 1.87320, k_3 = 1.48695$, $\sigma(.)$ is the sigmoid function and τ is the variance (optionally, set $C = -k_1$).