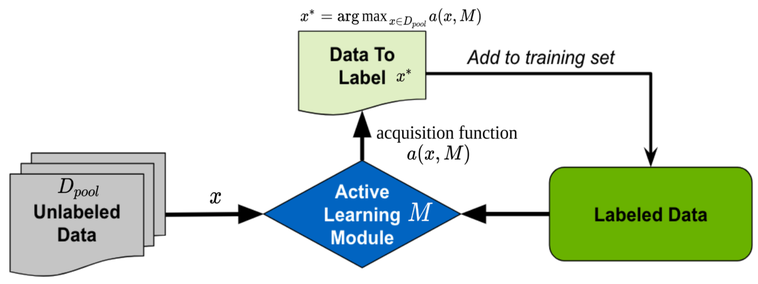
The Relevance of Bayesian Layer Positioning for Model Uncertainty in Deep Bayesian Active Learning

Jiaming Zeng[1], Adam Lesnikowski[2], Jose M. Alvarez[2]
[1] Stanford University | [2] NVIDIA

Active Learning

Active learning (AL) is a label-efficient framework in which the system incorporates information from the model being trained to optimally select data to label next. While AL is important in machine learning, scaling to high-dimensional data and deep neural networks is a major challenge with relatively scarce existing literature. We improve the memory and computational demands of Bayesian uncertainty approaches for AL by proposing a hybrid Bayesian and deterministic architecture able to capture the uncertainty information needed for AL, and demonstrate on an image classification challenge.



Bayesian Neural Networks

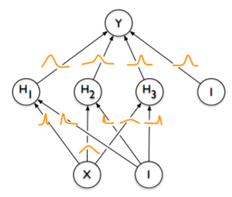
Problem Setup:

Data : $D = \{\mathbf{X}_i, y_i\}_{i=1}^N$ where $\mathbf{X}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$

Prior : $p(\mathbf{w}) \sim N(\mu, \Sigma)$

Posterior : $p(\mathbf{w}|\mathbf{X}, y) = \frac{p(y|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(y|\mathbf{X})}$ ← *Model evidence: Difficult to compute*

Prediction : $p(y^*|\mathbf{x}^*, \mathbf{X}, y) = \int p(y^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathbf{X}, y)d\mathbf{w}$



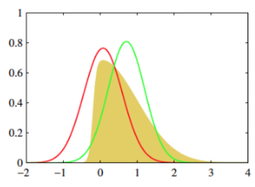
Variational Inference: Approximate intractable distribution ***p(w|X,y)*** with simpler distribution ***q(w)***.

posterior distribution for the weights → $q^*(\mathbf{w}) = \arg\min_{q(\mathbf{w}) \in D} KL(q(\mathbf{w})||p(\mathbf{w}|x)) = \arg\min_{q(\mathbf{w}) \in D} \mathbb{E}\left(\frac{\log q(\mathbf{w})}{\log p(\mathbf{w}|\mathbf{X})}\right)$

Minimizing KL is equivalent to maximizing *Evidence Lower Bound (ELBO)*

$$
\begin{aligned}
-ELBO &= \mathbb{E}\left(\frac{\log q(\mathbf{w})}{\log p(\mathbf{w}|\mathbf{X})}\right) - \log p(\mathbf{X}) \\
&= \mathbb{E}(\log q(\mathbf{w})) - \mathbb{E}(\log p(\mathbf{w})p(\mathbf{X}|\mathbf{w})) \\
&= -\mathbb{E}(\log p(\mathbf{X}|\mathbf{w})) + \mathbb{E}\left(\log \frac{q(\mathbf{w})}{p(\mathbf{w})}\right)
\end{aligned}
$$

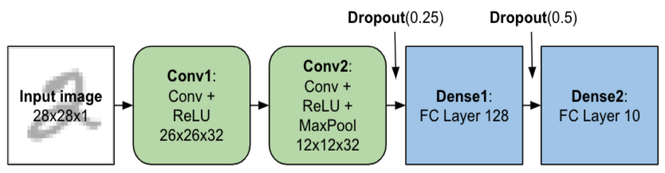
↑ Expected Neg Log Likelihood ↑ Prior KL



Experimental Setup

Active Learning Setup:

- Initial Training: 20 images
- # acquisitions: ~100
- Images per acquisition: 10



Acquisition Functions:

- Random (baseline)
- Max Entropy:
- Variation Ratios $\mathbb{H}[y|\mathbf{x}, \mathcal{D}] := -\sum_c p(y = c|\mathbf{x}, \mathcal{D}) \log p(y = c|\mathbf{x}, \mathcal{D})$

$$\text{variation-ratio}[x] := 1 - \max_y p(y|\mathbf{x}, \mathcal{D})$$

Training Setup:

- # of Trials: 3
- Epochs: 200
- # of Monte Carlo Samples: 100
- Experimental Architectures:
- Optimizer: ADAM
- Learning rate: 0.001
- Batch size: 64
- Initial posterior variance: $\sigma_{q(\mathbf{w})} \sim N(-3, 0.1)$

	BNN	BNN-1	BNN-2	BNN-3	BNN1	BNN2	BNN3	CNN
Conv1	Bayes	Det	Det	Det	Bayes	Bayes	Bayes	Det
Conv2	Bayes	Det	Det	Bayes	Det	Bayes	Bayes	Det
Dense1	Bayes	Det	Bayes	Bayes	Det	Det	Bayes	Det
Dense2	Bayes	Bayes	Bayes	Bayes	Det	Det	Det	Det

Table 1: Summary of combinations of Bayesian and deterministic layers in our architectures.

Estimating Model Uncertainty

BNN provides a straightforward to model the uncertainty of predictions
Approximate *p(y = c)* with Monte Carlo sampling, where *T* is the number of MC samples
Implementation:

- Perform *T* forward passes through the network and average together the distribution over classes for each pass

$$
\begin{aligned}
p(y = c|\mathbf{x}, \mathcal{D}) &= \int p(y = c|\mathbf{x}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \\
&\approx \int p(y = c|\mathbf{x}, \mathbf{w})q^*(\mathbf{w})d\mathbf{w} \\
&\approx \frac{1}{T}\sum_{t=1}^{T} p(y = c|\mathbf{x}, \hat{\mathbf{w}}_t)
\end{aligned}
$$

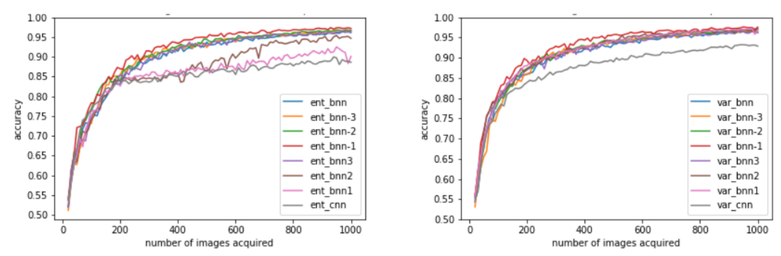
Results

Comparison of Experimental Architectures

- CNN, BNN1, BNN2, BNN3 all underperformed the BNN.
- BNN-1, BNN-2, BNN-3 all outperformed the BNN.
- BNN-1 showed the best performance.
- Having Bayesian layers closer to the output layer captures more uncertainty than having Bayesian layers closer to the input.
- Additional Bayesian layers may actually compromise the accuracy without the benefit of added uncertainty modeling.

Acquisitions	BNN	BNN-1	BNN-2	BNN-3	BNN1	BNN2	BNN3	CNN
Random	6.62%	**5.58** %	5.71%	6.37 %	6.62%	6.44%	6.59%	6.90%
Max Ent	3.67%	**2.63** %	3.22%	3.28%	7.50%	4.62%	3.58%	10.03%
Var Ratios	3.56%	**2.40** %	3.00 %	3.34%	2.70%	2.84%	3.32%	6.48%

Table 2: We compared the test error rates for our eight architectures with various hybrids of Bayesian and deterministic architectures, lower rates are better.



(a) max entropy (b) variation ratios

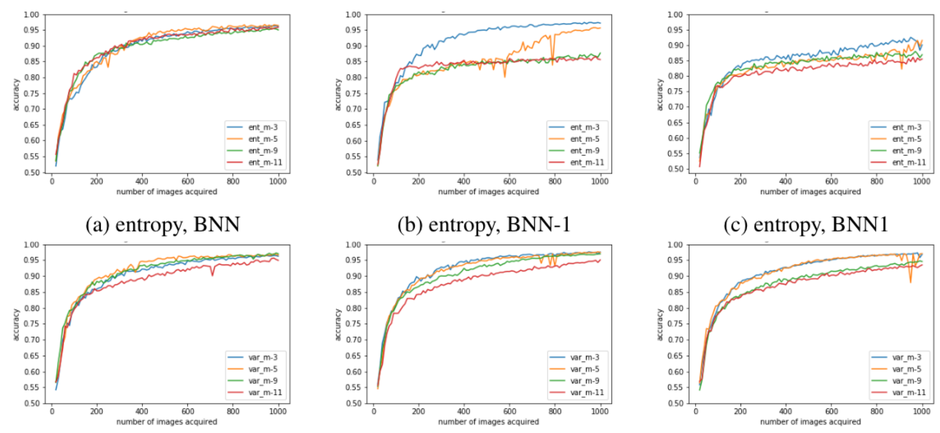
Comparison of Bayesian-ness of Prior Initialization

We tuned the initial variance means of our networks for μ = [-3, -5, -9, -11]. Lower μ values initialize the networks closer to deterministic, while higher values lead to more Bayesian networks.

- Better performance with more Bayesian values of μ closer to -3.
- Having Bayesian Dense 2 is better than having Bayesian Conv1. (i.e. the BNN-1 architecture with exactly one Bayesian layer at the end of a fully deterministic network was optimal)
- We capture the uncertainty useful for our active learning task with just one Bayesian layer at the end, which is more memory and computation friendly than fully Bayesian networks.

Acquisitions	BNN	BNN-1	BNN-2	BNN-3	BNN1	BNN2	BNN3	CNN
Random	6.07%	**5.36** %	5.71%	5.88 %	5.93%	5.76%	6.56%	6.90%
Max Ent	3.28%	**2.63** %	3.15%	2.87%	7.50%	4.62%	3.44%	10.03%
Var Ratios	2.74%	**2.38** %	2.69%	2.89%	2.70%	2.59%	2.97%	6.29%

Table 3: We compared the test error rate for all eight architectures for the optimal found μ, lower rates are better.



(a) entropy, BNN (b) entropy, BNN-1 (c) entropy, BNN1

(d) variation ratios, BNN (e) variation ratios, BNN-1 (f) variation ratios, BNN1

Conclusion

Major challenges in implementing and using Bayesian CNNs are the time and computational difficulties required in training. Our results strongly suggest that it is unnecessary to use fully Bayesian CNNs for capturing model uncertainty.

- Using only one or two Bayesian layers (BNN-1, BNN-2) near the output of a network outperforms a fully Bayesian model (BNN).
- The more Bayesian the layers are, as measured by our μ value, the more uncertainty we can capture.
- Combining deterministic CNNs' accuracy and speed with Bayesian CNNs' ability to capture uncertainty is useful for downstream tasks like active learning into an attractive hybrid architecture.
- We hope to encourage more use of Bayesian uncertainty through our novel hybrid architecture by combining the uncertainty representation of Bayesian weights with the computational parsimony of fully deterministic representations.

References:
[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424, 2015.
[2] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. arXiv preprint arXiv:1703.02910, 2017.
[3] Joshua V Dillon, Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A Saurous. Tensorflow distributions. arXiv preprint arXiv:1711.10604, 2017.