Deep Bayesian Active Learning with Image Data

An Evaluation

Fahad Mansoor

Technical University Munich

Contact Information:
Department of Informatics
Technical University of Munich
Boltzmannstrasse 15, 85748 Garching
Email: `mansoor@cs.tum.edu`

Abstract

Uncertainty information opens up a new set of possibilities for neural networks. Active learning utilizes uncertainty information to query some information source to select the data it needs. In the paper under evaluation various acquisition functions allows the neural network to be trained using much lesser data. Which greatly reduces the effort required for labeling and acquiring large data sets. This evaluation tries to reproduce the results and lists the issues faced in the reproduction.

Introduction

Existing Literature [1], showed that averaging forward dropout passes during test time through the network is equivalent to Monte Carlo integration over a Gaussian process's posterior approximation. This uncertainty information can be used for Active Learning.

Acquisition functions are used in this regard to select the data which is important.The acquisition of training points comes at a cost of additional training time, as we have to average out the stochastic forward passes through the data after training it on some initial data. The models are then trained again on the new points. This greatly reduces the need for large amounts of data and allows for your model to be trained on limited sets of data.

Background

Bayesian Convolution neural networks are used which were proposed in [1]. These networks allow us to represent prediction uncertainty on image data. BCNNs have a prior probability placed over the model parameters

$$\omega \sim p(\omega) \tag{1}$$

The likelihood is defined as

$$p(y = c|x, \omega) = softmax(f^{\omega}(x)) \tag{2}$$

Dropout is used to perform approximate inference, as it was shown in [1]. Inference is done by using dropout in the training phase and also during the testing phase (called Monte Carlo dropout). This is equivalent to performing approximate variational inference where we find $q^*_\theta(\omega)$ in a tractable family which minimizes the KL Divergences to the true model posterior $p(\omega|\mathcal{D}_{train})$

The Uncertainty in the weights induces prediction uncertainty by marginalizing over the approximate posterior using Monte Carlo integration

$$p(y = c|x, \mathcal{D}_{train}) = \int p(y = c|x, \omega)p(\omega|\mathcal{D}_{train}) \tag{3}$$

$$\approx \int p(y = c|x, \omega)q^*_\theta(\omega) \tag{4}$$

$$\approx \frac{1}{T}\sum_{t=1}^{T} p(y^*|x, \hat{\omega}_t) \tag{5}$$

This is computationally tractable and provides uncertainty information.

Methodology

In this paper [2] acquisition functions were used to query the data set. The training set was divided into two sets, namely a pool set with around 40 thousand points and an initial starting train set with only 20 points which were uniformly picked from each of the labeled classes.

Additional points were acquired using acquisition functions and the model was re-trained on each acquisition iteration. To approximate the predictive uncertainty 100 iterations of stochastic forward passed were performed as a form of Monte Carlo integration over the approximate posterior.

To get the data points from the pool set the following acquisition functions were used

1. Max Entropy
2. Maximization of Mutual information
3. Maximize Variational ratios
4. Maximize Mean standard deviation
5. Random Acquisition

Implementation

The paper under review was implemented in Pytorch. Pytorch was chosen because of its flexible API and its performance. An existing implementation in Keras was used as a reference. One thing to note here is that to reduce computational complexity of stochastic forward pass (MC Dropout) 2000 randomly selected samples of the pool data were selected rather than the complete pool set and the predictive uncertainty was calculated on that randomly selected subset.

The implementation in Pytorch significantly improved the performance over the Keras implementation. For stochastic forward passes in Pytorch the model was marked as train to get dropout during the prediction over the pool set phase. The implementation can be found at [3]

Analysis of Acquisition Functions

Different acquisition function listed above use different heuristics to select data points. Images from the pool set are selected by maximizing the value of the acquisition function. Some of the images selected during different iterations using different acquisition algorithms are presented below

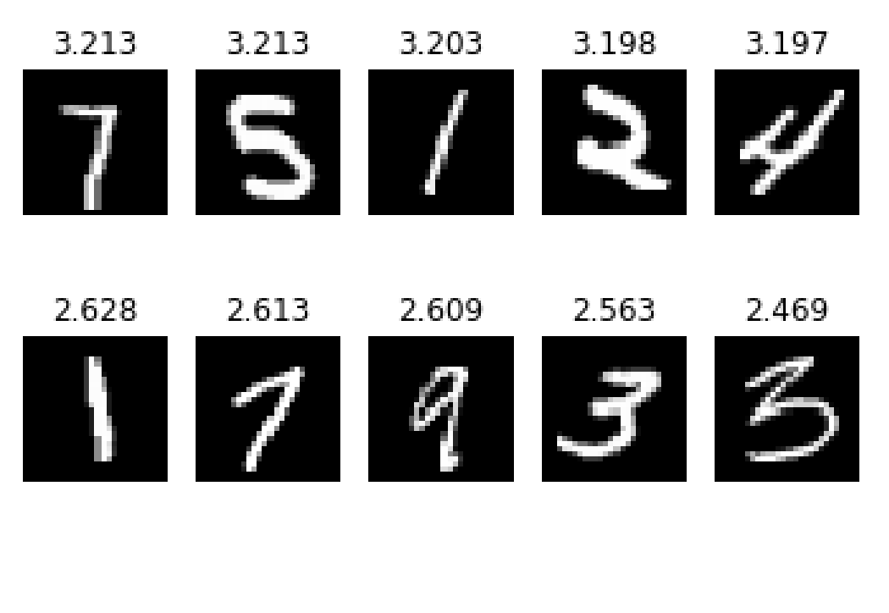

Figure 1: Acquisition function: Max Entropy, Top: the 5 selected points with entropy values on top of each image Bottom : The points with the lowest entropy

The value of the acquisition function was calculated for the predictive uncertainty on all points. The values of variational ratios acquisition function is shown below which shows the scores for all the points in the random pool subset.

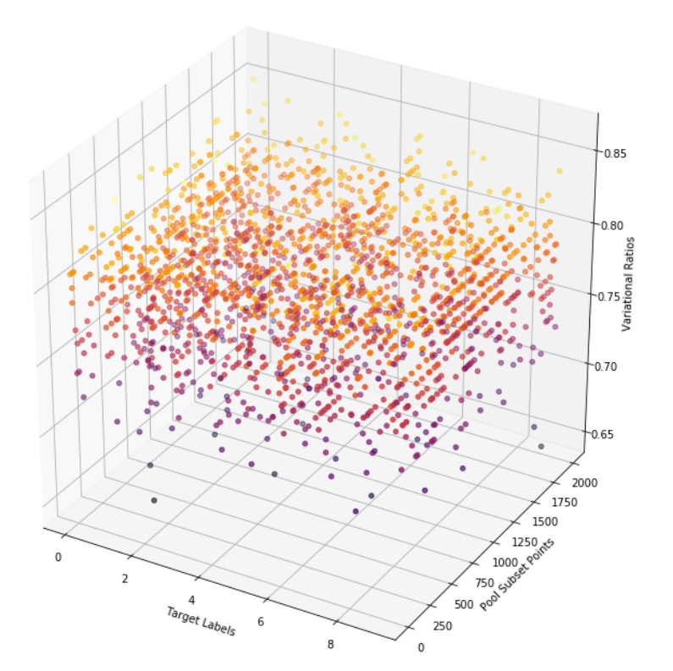

Figure 2: Variational ratios for all points of the pool subset

These points were then used for subsequent training and testing loops. The quality of the acquisitions determined the speed at which the network was trained.

Results

The results were calculated using various configurations, using the architecture followed by the paper, we were able to get to within 10 % error in 350 iterations. When using a different architecture with more pool subset points and higher dropout iterations we were able to get to within 10% in 90 acquired images. But this run took very long.

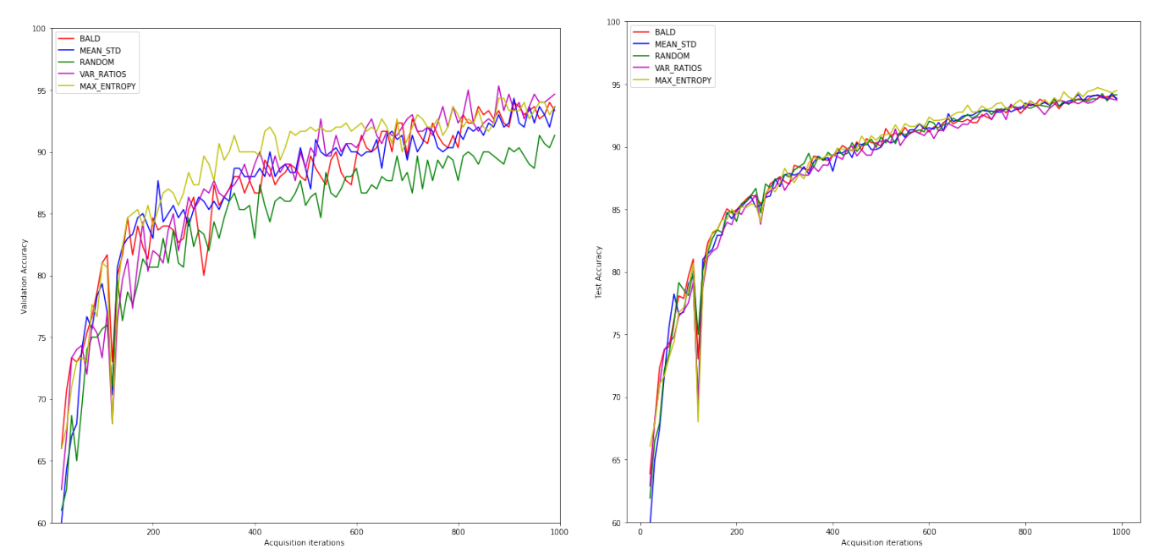

Figure 3: Left validation accuracy, Right Prediction accuracy

Configurations	10 % error	5 % error
Queries = 10 (T=20, D=50, P=2000)	370	-
Queries = 5 (T=100, D=100, P=20000)	120	550

Table 1: No of Acquired images to achieve different error levels, here T= Initial train set, D = dropout iterations, P = pool subset

The experiments with 5 queries were more interesting as they gave a better result. Here Mean of Standard deviation gives the best results out of all of the acquisition function

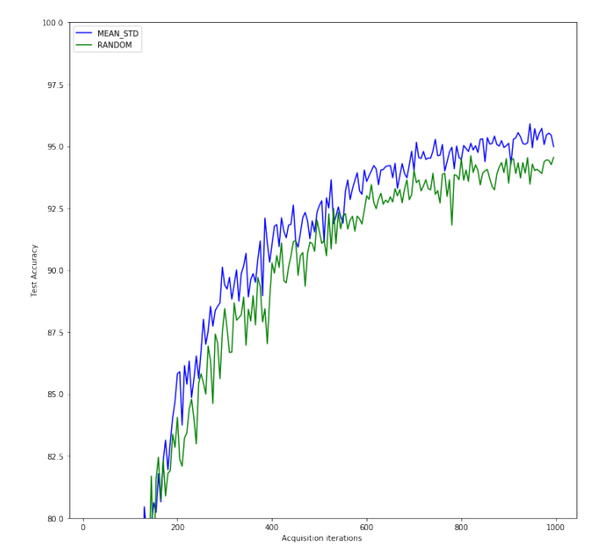

Figure 4: Test Accuracy

The reason for the bad performance of Mean Std in the paper was that the implementation of mean std was incorrect in the implementation of the acquisition function in Keras. I have emailed the author to inform him of this errata.

Issues faced and Challenges

The results of this evaluation and reimplementation weren't as good as that of the original paper, on analysis and debugging we were able to narrow this problem down. In the Keras implementation the soft-max scores for stochastic forward passes are more precise (higher probability to the expect class of the labeled data), where as the soft-max scores for stochastic forward passes in the Pytorch reimplementation are either somewhat distributed (low or same score to all classes) or they are aggregated towards one class particular class. (e.g high probability to 9's in all data points)

The training of this network, even with the Pytorch's speed improvements was very slow. A simple run with 50 dropout iterations and 100 acquisition iterations with 10 queries per iteration took over 2 hours on the CPU. A whole set of the 5 acquisition function would take around 7-8 hours to complete. And three iterations to average the results out takes well over 20 hours of training time.

We weren't able to do the medical imaging part of the paper because of the sheer amount of time required to train and lack of access to a GPU cluster.

The model had to be reinitialized after every acquisition step, this contributed to the long training time of the model.

Conclusions

- Acquisition functions can be used to significantly reduce the amount of training data needed.
- These networks take a very very long time to train because of MC Dropout and model reinitialization.
- The results weren't as impressive as the original paper this was due to Softmax scores behaving differently in Pytorch.

Forthcoming Research

Currently two topics are under consideration for research

- New acquisition functions for faster training.
- Using the uncertainty information to make dynamic child networks for classification purposes.

References

[1] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015.

[2] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

[3] Fahad. Mansoor. Dlrw Bayesian Active Learning. *GitHub repository*, 2017.