Abstract
In active learning, new labels are commonly acquired in batches. However, common acquisition functions are only meant for one-sample acquisition rounds at a time, and when their scores are used naively for batch acquisition, they result in batches lacking diversity, which deteriorates performance. On the other hand, state-of-the-art batch acquisition functions are costly to compute. In this paper, we present a novel class of stochastic acquisition functions that extend one-sample acquisition functions to the batch setting by observing how one-sample acquisition scores change as additional samples are acquired and modelling this difference for additional batch samples. We simply acquire new samples by sampling from the pool set using a Gibbs distribution based on the acquisition scores. Our acquisition functions are both vastly cheaper to compute and out-perform other batch acquisition functions.

1. Introduction
Active Learning is essential for increasing label- and thus cost-efficiency in real-world machine learning applications, especially when they use deep learning. In active learning, we have access to a huge reservoir of unlabelled data \( \{ x_i^{\text{pool}} \} \) in a pool set and iteratively select samples from this pool set to be labeled by an oracle (e.g., human experts) to increase the model performance as quickly as possible.

Usually, acquisition functions are used to score all pool samples and the highest scorer is acquired. The goal of an acquisition function is to score the most informative samples the highest. Common acquisition functions include predictive entropy, variation-ratios, and BALD (in Bayesian Active Learning) (Gal et al., 2017; Houlsby et al., 2011).

In practice, batch acquisitions are used: instead of acquiring samples individually, samples are acquired in batches without retraining the model in-between. Commonly, the top \( B \) highest-scoring samples are acquired from the pool set. However, this approach suffers from a lack of diversity within the batches as samples that are considered informative initially might be very similar and thus not help the model gain performance because of this duplication.

We introduce novel stochastic acquisition functions, which take into account that, as we add samples to an acquisition batch, the one-samples scores ought to be updated and would diverge from the initial scores more and more.

Instead of using costly exact computations in a Bayesian setting to estimate the change in scores (Kirsch et al., 2019), this new class of stochastic acquisition functions selects batch samples stochastically using the one-sample acquisition scores.

Our new acquisition functions perform better than BatchBALD, which directly estimates the joint MI, while being cheaper to compute for the relevant batch sizes. We also compare favorably with other batch acquisition functions.

Figure 1. On RepeatedMNIST (≈ MNISTx2), our Softmax acquisition functions outperform the top-k one and perform as well as the Batch versions of BALD and EPIG-BALD (Kirsch et al., 2019; Anonymous, 2021) while requiring a fraction of the compute and scaling to higher batch sizes.
functions which are also expensive to compute. This is depicted in Figure 1.

2. Related Work

BatchBALD (Kirsch et al., 2019) solves the issue of BALD acquiring redundant samples in the batch setting by explicitly capturing the correlations between samples using their joint predictive probability. While this works well for small acquisition batch sizes, it suffers from noisy estimates. This requires many MC dropout samples and has high compute and memory overhead due to a combinatorial explosion even for modest acquisition batch sizes > 10. Needing many samples to capture the joint predictive probability also means that BatchBALD cannot be used with deep ensembles and is limited to implicit approximate Bayesian models that can be sampled from consistently.

Our contribution does not have this limitation and cheaply scales to higher acquisition batch sizes as we show empirically.

3. Background

In the following section, we introduce active learning and acquisition functions, as well Bayesian methods that we will use for analysis. We note that our approach does not require Bayesian model explicitly. We only follow a Bayesian approach to motivate our novel stochastic acquisition functions.

Bayesian Deep Learning. The model parameters are treated as a random variable \( \Omega \) with prior distribution \( p(\omega) \). We denote the training set \( \mathcal{D}_{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}})\} \subseteq \mathcal{D}_{\text{pool}}, \) where \( \{(x_i^{\text{train}})\} \subseteq \mathcal{D}_{\text{pool}} \) are the input samples and \( \{y_i^{\text{train}}\} \subseteq \mathcal{D}_{\text{pool}} \) the labels or targets.

The probabilistic model is as follows:

\[
p(y, x, \omega) = p(y \mid x, \omega) \ p(\omega) \ p(x),
\]

where \( x \) and \( y \) are outcomes for the random variables \( X \) and \( Y \) denoting the input and label, respectively.

To include multiple labels and inputs, we expand the model to joints of random variables \( \{x_i\} \subseteq \mathcal{X} \) and \( \{y_i\} \subseteq \mathcal{Y} \) obtaining

\[
p(\{y_i\}, \{x_i\}, \omega) = \prod_{i \in \mathcal{I}} p(y_i \mid x_i, \omega) \ p(x_i) \ p(\omega).
\]

We are only interested in discriminative models and thus do not explicitly specify \( p(x) \).

The posterior parameter distribution \( p(\omega \mid \mathcal{D}_{\text{train}}) \) is determined via Bayesian inference. We obtain \( p(\omega \mid \mathcal{D}_{\text{train}}) \) using Bayes’ law:

\[
p(\omega \mid \mathcal{D}_{\text{train}}) \propto p(\{y_i^{\text{train}}\} \mid \{x_i^{\text{train}}\}, \omega) \ p(\omega).
\]

which allows for predictions by marginalizing over \( \Omega \):

\[
p(y \mid x, \mathcal{D}_{\text{train}}) = \mathbb{E}_{\omega \sim p(\omega \mid \mathcal{D}_{\text{train}})} [p(y \mid x, \omega)].
\]

Exact Bayesian inference is intractable in deep learning, and we use variational inference for approximate inference using an distribution \( q(\omega) \). We determine \( q(\omega) \) by minimizing the following KL divergence:

\[
\text{KL}(q(\omega) \mid \mid p(\omega \mid \mathcal{D}_{\text{train}})) = \mathbb{E}_{q(\omega)} [-\log p(\{y_i^{\text{train}}\} \mid \{x_i^{\text{train}}\}, \omega)]
\]

\[
+ \text{KL}(q(\omega) \mid \mid p(\omega)) + \log p(\mathcal{D}_{\text{train}}) \geq 0.
\]

We can use the local reparameterization trick and Monte-Carlo dropout for \( q(\omega) \) in a deep learning context.

Active Learning. In active learning, we have access to an unlabelled pool set \( \mathcal{D}_{\text{pool}} = \{x_i^{\text{pool}}\} \subseteq \mathcal{D}_{\text{pool}} \). We select samples from it and ask an oracle for labels. We then add those newly labeled samples to the training set, retrain our model, and repeat this process until the model satisfies our performance requirements.

Samples acquired in batches to avoid retraining the model all the time. We score possible candidate batches \( x^{\text{acq}} \) with the acquisition batch size \( b \) using an acquisition function \( a(\{x_i^{\text{acq}}\}, p(\Omega \mid \mathcal{D}_{\text{train}})) \) and pick the highest scoring one:

\[
\arg \max_{\{x_i^{\text{acq}}\} \subseteq \mathcal{D}_{\text{pool}}} a(\{x_i^{\text{acq}}\}, p(\Omega \mid \mathcal{D}_{\text{train}}))
\]

Many acquisition functions only support scoring one sample at a time. We canonically extend such an acquisition function \( a_{\text{single}}(x^{\text{acq}}, p(\Omega \mid \mathcal{D}_{\text{train}})) \) to batches by summing over the individual batch candidates:

\[
a(\{x_i^{\text{acq}}\}, p(\Omega \mid \mathcal{D}_{\text{train}})) := \sum_i a_{\text{single}}(x_i^{\text{acq}}, p(\Omega \mid \mathcal{D}_{\text{train}})).
\]

Finding the highest-scoring batch for one-sample acquisition functions is equivalent to selecting the highest scorers, which is common practice (Gal et al., 2017).

Indeed, BALD was originally introduced as a one-sample acquisition function of the expected information gain between the prediction \( Y^{\text{acq}} \) for an input \( x^{\text{acq}} \) and the model parameters \( \Omega \):

\[
a_{\text{BALD}}(x^{\text{acq}}, p(\Omega \mid \mathcal{D}_{\text{train}})) := [Y^{\text{acq}}, \Omega \mid x^{\text{acq}}, \mathcal{D}_{\text{train}}].
\]

In BatchBALD (Kirsch et al., 2019), this one-sample case was canonically extended to the batch acquisition case using the expected information gain between the joint of the predictions \( \{Y_i^{\text{acq}}\} \) for the batch candidates \( \{x_i^{\text{acq}}\} \) and the model parameters \( \Omega \):
A Simple Baseline for Batch Active Learning with Stochastic Acquisition Functions

\[ a_{\text{BatchBALD}}(\{x_i^{\text{acq}}\}, p(\Omega \mid D^{\text{train}})) := \]

\[ = I(\{Y_i^{\text{eval}}\}; \Omega \mid \{x_i^{\text{acq}}\}, D^{\text{train}}] \]

\[ = I[Y_1^{\text{acq}}, \ldots, Y_b^{\text{acq}}; \Omega \mid \{x_1^{\text{acq}}, \ldots, x_b^{\text{acq}}\}, D^{\text{train}}]. \]

**Notation.** Instead of \( \{Y_i^{\text{eval}}\}, \{x_i^{\text{eval}}\} \), we will write \( Y^{\text{eval}} \)

\( x^{\text{eval}} \) and so on to to cut down on notation. Like above, all
terms can be canonically extended to sets by substituting the joint. We provide the full derivations in the appendix. Also
note again that lower-case variables like \( y^{\text{eval}} \) are outcomes
and upper-case variables like \( Y^{\text{eval}} \) are random variables,
with the exception of the datasets \( D^{\text{pool}}, D^{\text{train}}, \) etc, which
are sets of outcomes.

4. Theory

We start by analysing how the BALD and entropy
acquisition scores of samples change as other samples
are acquired into the training set. This will motivate our
stochastic acquisition function, which takes into account that
the computed scores are not exact within batch acquisitions.

**BALD as an Approximation.** BALD as the expected
information gain is just an expectation given the current
model’s predictions for \( x \):

\[ I[Y; \Omega \mid x, D^{\text{train}}] = H[\Omega \mid D^{\text{train}}] - H[Y, \Omega \mid x, D^{\text{train}}], \]

where:

\[ H[\Omega \mid Y, x, D^{\text{train}}] = \mathbb{E}_{p(y|x, D^{\text{train}})} H[\Omega \mid y, x, D^{\text{train}}] \quad (1) \]

\( H[\Omega \mid y, x, D^{\text{train}}] \) is equivalent to adding the label \( y \) for
\( x \) to the training set and measuring the uncertainty of the
posterior parameters. This is why BALD can be viewed as
expected reduction in posterior uncertainty. It is different
from the actual reduction in posterior uncertainty if we had
access to the true label \( y_{\text{true}} \):

\[ I[\Omega; y_{\text{true}}; x, D^{\text{train}}] := H[\Omega \mid D^{\text{train}}] - H[\Omega \mid y, x, D^{\text{train}}] \]

where we define \( I[\Omega; y_{\text{true}}; x, D^{\text{train}}] \) as a shorthand here.
BALD is the best we can do unless we have access to a
different predictor for labels.

**Change of Scores After Acquisitions.** How do the
acquisition scores for BALD change as additional samples
are acquired (on average)? The BALD score for a sample \( x \)
is \( I[Y; \Omega \mid x, D^{\text{train}}] \). After acquiring the labels \( y^{\text{acq}} \)
for \( x^{\text{acq}} \), the expected score becomes \( I[Y; \Omega \mid x, y^{\text{acq}}, x^{\text{acq}}, D^{\text{train}}] \).
However, note that the above is the expected score which is
different from the score when conditioned on the true labels
\( y_{\text{true}}: I[\Omega; y_{\text{true}}; x, y^{\text{acq}}, x^{\text{acq}}, D^{\text{train}}] \). As we do not assume
access to the true labels \( y^{\text{acq}} \), we will use the expectation
over \( y^{\text{acq}} \). The difference in acquisition scores after \( y^{\text{acq}} \)
haven been acquired for \( x^{\text{acq}} \) is simply:

\[ I[Y; \Omega \mid X, D^{\text{train}}] - I[Y; \Omega \mid x, y^{\text{acq}}, x^{\text{acq}}, D^{\text{train}}] \]

\[ = I[Y; \Omega; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}] \]

\[ = I[Y; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}] \]

\[ - I[Y; y^{\text{acq}} \mid x, x^{\text{acq}}, Y, D^{\text{train}}] = 0 \]

where the last term is 0 because \( Y \) and all \( y^{\text{acq}} \) are mutually
independent given \( \Omega \). Hence, we see that the change in
score as \( x^{\text{acq}} \) are acquired is just \( I[Y; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}] \),
which is nonnegative as a mutual information in two terms.
This is also the change in score when using the entropy as
the acquisition function as

\[ H[Y \mid X, D^{\text{train}}] - H[Y \mid x, y^{\text{acq}}, x^{\text{acq}}, D^{\text{train}}] \]

\[ = I[Y; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}]. \]

**True Change of Scores After Acquisitions.** Using the
same argument as before, we can predict the true change
given the actual label \( y_{\text{true}} \) for \( x \):

\[ I[y_{\text{true}}; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}] := \]

\[ = - \log \left( \frac{p(y_{\text{true}} \mid x, D^{\text{train}})}{p(y_{\text{true}}; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}})} \right), \]

where we again introduce a shorthand.

5. Method

As we do not know any of the true labels, we can only start
with the original acquisition scores for a model trained on
\( D^{\text{train}} \), the posterior uncertainty will decrease in expectation
as additional samples are acquired into \( D^{\text{train}} \). However,
this change depends on the model, the training set, and the
samples that are already in the acquisition batch.

As we generally do not have enough information about the
sample correlations to meaningfully model \( I[y_{\text{true}}; y^{\text{acq}} \mid x, x^{\text{acq}}, D^{\text{train}}] \), we use a Gibbs
distribution over the scores from a deterministic one-sample acquisition function
\( a(y^{\text{acq}}, p(\Omega \mid D^{\text{train}})) \) (either BALD or entropy based) with
a temperature hyperparameter \( T \) and sample \( b \) samples
without replacement for the acquisition batch:

\[ p(x_{i}^{\text{pool}}, a) = \frac{\exp(a(x_{i}^{\text{pool}}, p(\Omega \mid D^{\text{train}}))/T)}{\sum_{j} \exp(a(x_{j}^{\text{pool}}, p(\Omega \mid D^{\text{train}}))/T)}. \]  

As this essentially applies a Softmax to the acquisition
scores, we call this stochastic variant of BALD SoftmaxBALD.

6. Experiments

**Setup.** We follow the experimental setup detailed in Kirsch
et al. (2019) and Anonymous (2021) for the experiments.
We use a LeNet-5 for MNIST (LeCun et al., 1998) and
Table 1. 25%/50%/75% quantiles for reaching 90% and 95% accuracy on MNIST. 5 trials each.

<table>
<thead>
<tr>
<th>Acquisition Size</th>
<th>90% Acc 5</th>
<th>90% Acc 10</th>
<th>90% Acc 20</th>
<th>90% Acc 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>BALD</td>
<td>172.5/200/212.5</td>
<td>∞</td>
<td>217.5/230/270</td>
<td>280/280/∞</td>
</tr>
<tr>
<td>EPIG-BALD</td>
<td>170/210/220</td>
<td>∞</td>
<td>220/220/240</td>
<td>280/280/∞</td>
</tr>
<tr>
<td>BatchBALD</td>
<td>108.75/115/117.5</td>
<td>260/282.5/∞</td>
<td>300/300/∞</td>
<td>∞/∞/∞</td>
</tr>
<tr>
<td>BatchEPIG-BALD</td>
<td>90/110/110</td>
<td>205/215/240</td>
<td>300/300/∞</td>
<td>∞/∞/∞</td>
</tr>
<tr>
<td>SoftmaxBALD (ours)</td>
<td>102.5/105/108.75</td>
<td>248.75/262.5/273.75</td>
<td>240/260/260</td>
<td>170/170/170</td>
</tr>
<tr>
<td>SoftmaxEPIG-BALD (ours)</td>
<td>100/110/110</td>
<td>225/230/230</td>
<td>270/280/290</td>
<td>170/170/170</td>
</tr>
</tbody>
</table>

Figure 2. Ablation: SoftmaxBALD for different temperatures. 30 trials each.

On CIFAR-10, our SoftmaxBALD acquisition function performs about as well as BALD with acquisition batch size 2500 (from an initial training set size of 5000). This is depicted in Figure 3. An ablation with different temperature parameters is shown in Figure 2.

Figure 3. On CIFAR-10, our SoftmaxBALD acquisition function performs about as well as BALD with acquisition batch size 2500 (from an initial training set size of 5000). 30 trials each.

7. Conclusion & Limitations

We have introduced an incredibly simple method to turn deterministic one-sample acquisition functions into stochastic batch acquisition functions that perform surprisingly well, performing on par with the much costlier to compute BatchBALD and BatchEPIG-BALD on MNIST, and no worse than than BALD even with very large acquisition sizes on CIFAR-10.

Repeated MNIST = MNISTx2. Repeated MNIST (Kirsch et al., 2019), which contains additional redundancy by repeated MNIST training samples twice with a small amount of Gaussian noise added, is challenging for one-sample acquisition functions which select the highest scorers. High redundancy can push one-sample acquisition functions to perform worse than uniform sampling. Only BatchBALD (Kirsch et al., 2019) fixed this in the case of BALD.

Our stochastic SoftmaxBALD outperforms BALD, which acquires the top-k highest scoring samples, strongly on all acquisition batch sizes (5, 10, 20, 50) and performs as well as BatchBALD. Note that BatchBALD scores take a long time to compute for larger acquisition batch sizes and suffer from precision issues for acquisition batch sizes larger than 10 (depending on the number of MC samples). Similarly, SoftmaxEPIG-BALD outperforms EPIG-BALD and performs as well as BatchEPIG-BALD, where (Batch)EPIG-BALD computes two (Batch)BALD scores and is thus even more expensive to compute (Anonymous, 2021). The accuracy plots are shown in Figure 1 and Table 1.

CIFAR-10. On CIFAR-10, with large batch sizes, StochasticBALD is smoother but otherwise performs on par when used with acquisition batch size 2500 from an initial training set of 5000. This is depicted in Figure 3. An ablation with different temperature parameters is shown in Figure 2.
References


