A Simple Baseline for Batch Active Learning with Stochastic Acquisition Functions

Anonymous Authors[1]

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).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

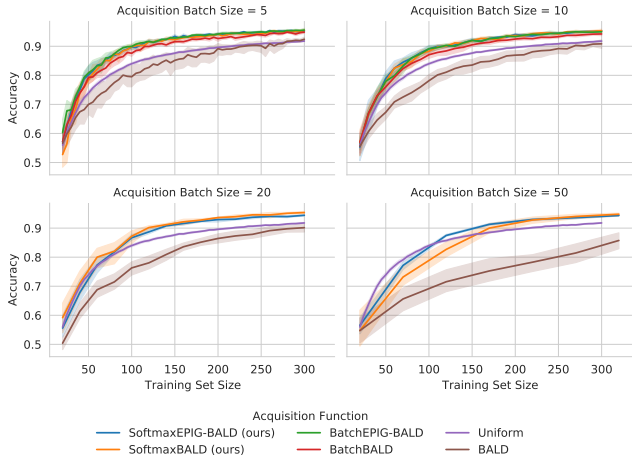

Figure 1. On RepeatedMNIST (≈ MNISTx2), our Softmax acquisition functions outperform the top-k one and perform as well as the Batch versions of BALD and EPIG-BALD(Kirsch et al., 2019; Anonymous, 2021) while requiring a fraction of the compute and scaling to higher batch sizes.

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 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 Ω with prior distribution $\mathrm{p}(\omega)$. We denote the training set $\mathcal{D}^{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}})\}_{1,\dots,i\in|\mathcal{D}^{\text{train}}|}$, where $\{x_i^{\text{train}}\}_{i\in\{1,\dots,|\mathcal{D}^{\text{train}}|\}}$ are the input samples and $\{y_i^{\text{train}}\}_{i\in\{1,\dots,|\mathcal{D}^{\text{train}}|\}}$ the labels or targets.

The probabilistic model is as follows:

$$\mathrm{p}(y, x, \omega) = \mathrm{p}(y \mid x, \omega)\ \mathrm{p}(\omega)\ \mathrm{p}(x),$$

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 $\{x_i\}_{i\in I}$ and $\{y_i\}_{i\in I}$ obtaining

$$\mathrm{p}(\{y_i\}_i, \{x_i\}_i, \omega) = \prod_{i\in I} \mathrm{p}(y_i \mid x_i, \omega)\, \mathrm{p}(x_i)\, \mathrm{p}(\omega).$$

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

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

$$\mathrm{p}(\omega \mid \mathcal{D}^{\text{train}}) \propto \mathrm{p}(\{y_i^{\text{train}}\}_i \mid \{x_i^{\text{train}}\}_i, \omega)\, \mathrm{p}(\omega).$$

which allows for predictions by marginalizing over Ω:

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

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

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}(\mathrm{q}(\omega) \parallel \mathrm{p}(\omega \mid \mathcal{D}^{\text{train}})) &= \\ = \mathbb{E}_{\mathrm{q}(\omega)}[\underbrace{-\log \mathrm{p}((\{y_i^{\text{train}}\}_i) \mid (\{x_i^{\text{train}}\}_i), \omega)}_{\text{likelihood}}] &\\ + \underbrace{\mathrm{D}_{\mathrm{KL}}(\mathrm{q}(\omega) \parallel \mathrm{p}(\omega))}_{\text{prior regularization}} + \underbrace{\log \mathrm{p}(\mathcal{D}^{\text{train}})}_{\text{model evidence}} &\geq 0.\end{aligned}$$

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

Active Learning. In active learning, we have access to an unlabelled pool set $\mathcal{D}^{\text{pool}} = \{x_i^{\text{pool}}\}_{i\in\{1,\dots,|\mathcal{D}^{\text{pool}}|\}}$. 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^{acq} with the *acquisition batch size* b using an acquisition function $a(\{x_i^{\text{acq}}\}_i, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}}))$ and pick the highest scoring one:

$$\underset{\{x_i^{\text{acq}}\}_{i\in\{1,\dots,b\}}\subseteq\mathcal{D}^{\text{pool}}}{\arg\max} a(\{x_i^{\text{acq}}\}_i, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}}))$$

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

$$a(\{x_i^{\text{acq}}\}_i, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}})) := \sum_i a_{\text{single}}(x_i^{\text{acq}}, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}})).$$

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^{acq} for an input x^{acq} and the model parameters Ω:

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

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 $\{Y_i^{\text{acq}}\}_i$ for the batch candidates $\{x_i^{\text{acq}}\}_i$ and the model parameters Ω:

$$
\begin{aligned}
&a_{\text{BatchBALD}}(\{x_i^{\text{acq}}\}_i, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}})) := \\
&\quad = \mathrm{I}[\{Y_i^{\text{acq}}\}_i; \Omega \mid \{x_i^{\text{acq}}\}_i, \mathcal{D}^{\text{train}}] \\
&\quad = \mathrm{I}[Y_1^{\text{acq}}, \dots, Y_b^{\text{acq}}; \Omega \mid x_1^{\text{acq}}, \dots, x_b^{\text{acq}}, \mathcal{D}^{\text{train}}].
\end{aligned}
$$

Notation. Instead of $\{Y_i^{\text{eval}}\}_i$, $\{x_i^{\text{eval}}\}_i$, we will write $\boldsymbol{Y}^{\text{eval}}$, $\boldsymbol{x}^{\text{eval}}$ 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^{eval} are outcomes and upper-case variables like Y^{eval} are random variables, with the exception of the datasets $\mathcal{D}^{\text{pool}}$, $\mathcal{D}^{\text{train}}$, 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:

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

where:

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

$\mathrm{H}[\Omega \mid y, x, \mathcal{D}^{\text{train}}]$ 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_{true}:

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

where we define $\mathrm{I}[\Omega; y^{\text{true}} \mid x, \mathcal{D}^{\text{train}}]$ 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 $\mathrm{I}[Y; \Omega \mid x, \mathcal{D}^{\text{train}}]$. After acquiring the labels $\boldsymbol{y}^{\text{acq}}$ for $\boldsymbol{x}^{\text{acq}}$, the expected score becomes $\mathrm{I}[Y; \Omega \mid x, \boldsymbol{y}^{\text{acq}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}]$. However, note that the above is the expected score which is different from the score when conditioned on the true labels y_{true}: $\mathrm{I}[\Omega; y_{\text{true}} \mid x, \boldsymbol{y}^{\text{acq}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}]$. As we do not assume access to the true labels $\boldsymbol{y}_{\text{true}}^{\text{acq}}$, we will use the expectation over $\boldsymbol{y}^{\text{acq}}$. The difference in acquisition scores after $\boldsymbol{y}^{\text{acq}}$ haven been acquired for $\boldsymbol{x}^{\text{acq}}$ is simply:

$$
\begin{aligned}
&\mathrm{I}[Y; \Omega \mid X, \mathcal{D}^{\text{train}}] - \mathrm{I}[Y; \Omega \mid x, \boldsymbol{y}^{\text{acq}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \\
&\quad = \mathrm{I}[Y; \Omega; \boldsymbol{y}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \\
&\quad = \mathrm{I}[Y; \boldsymbol{y}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \\
&\qquad - \underbrace{\mathrm{I}[Y; \boldsymbol{y}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \Omega, \mathcal{D}^{\text{train}}]}_{=0},
\end{aligned}
$$

where the last term is 0 because Y and all $\boldsymbol{y}^{\text{acq}}$ are mutually independent given Ω. Hence, we see that the change in score as $\boldsymbol{x}^{\text{acq}}$ are acquired is just $\mathrm{I}[Y; \boldsymbol{y}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}]$, 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

$$
\begin{aligned}
&\mathrm{H}[Y \mid X, \mathcal{D}^{\text{train}}] - \mathrm{H}[Y \mid x, \boldsymbol{y}^{\text{acq}}, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] \\
&\quad = \mathrm{I}[Y; \boldsymbol{y}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}].
\end{aligned}
$$

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

$$
\begin{aligned}
&\mathrm{I}[y_{\text{true}}; \boldsymbol{y}_{\text{true}}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}] := \\
&\quad = -\log \left(\frac{\mathrm{p}(y_{\text{true}} \mid x, \mathcal{D}^{\text{train}}) \, \mathrm{p}(\boldsymbol{y}_{\text{true}}^{\text{acq}} \mid \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}})}{\mathrm{p}(y_{\text{true}}, \boldsymbol{y}_{\text{true}}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}})} \right),
\end{aligned}
$$

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 $\mathcal{D}^{\text{train}}$, the posterior uncertainty will decrease in expectation as additional samples are acquired into $\mathcal{D}^{\text{train}}$. 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 $\mathrm{I}[y_{\text{true}}; \boldsymbol{y}_{\text{true}}^{\text{acq}} \mid x, \boldsymbol{x}^{\text{acq}}, \mathcal{D}^{\text{train}}]$, we use a Gibbs distribution over the scores from a deterministic one-sample acquisition function $a(x^{\text{acq}}, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}}))$ (either BALD or entropy based) with a temperature hyperparameter T and sample b samples without replacement for the acquisition batch:

$$
\mathrm{p}(x_i^{\text{pool}}; a) = \frac{\exp(a(x_i^{\text{pool}}, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}}))/T)}{\sum_j \exp(a(x_j^{\text{pool}}, \mathrm{p}(\Omega \mid \mathcal{D}^{\text{train}}))/T)}. \quad (2)
$$

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.

Acquisition Size	5		10		20		50	
Acquisition Function	90% Acc	95% Acc	90% Acc	95% Acc	90% Acc	95% Acc	90% Acc	95% Acc
BALD	172.5/200/212.5	∞	217.5/230/270	∞	280/280/∞	∞	∞	∞
EPIG-BALD	170/210/220	∞	220/220/240	∞	280/280/∞	∞	∞	∞
BatchBALD	108.75/115/117.5	260/282.5/∞	130/140/160	300/∞/∞			—	
BatchEPIG-BALD	90/110/110	205/215/240	110/120/120	260/270/270			—	
SoftmaxBALD (ours)	102.5/105/108.75	248.75/262.5/273.75	110/110/130	240/260/260	120/120/140	260/280/300	170/170/220	320/320/∞
SoftmaxEPIG-BALD (ours)	100/110/110	225/230/230	110/110/110	270/280/290	140/140/140	∞	170/170/170	∞

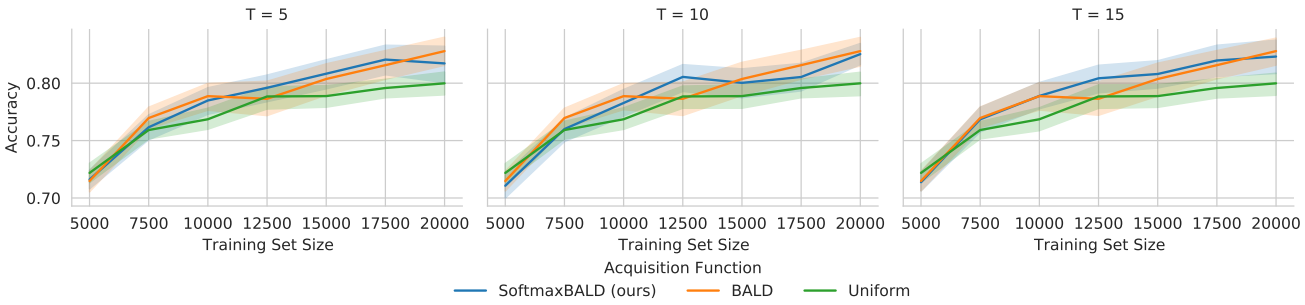

Figure 2. Ablation: SoftmaxBALD for different temperatures. 30 trials each.

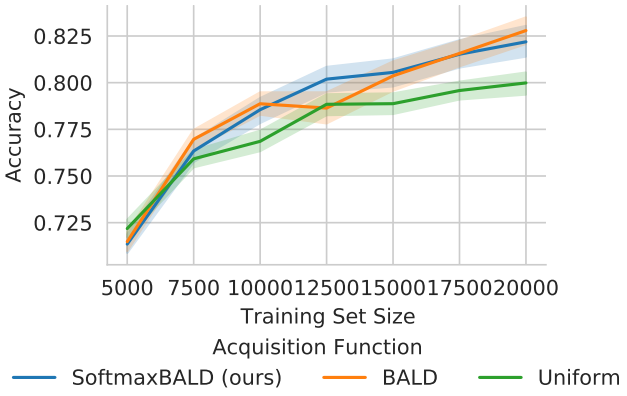

Figure 3. On CIFAR-10, our SoftmaxBALD acquisition function performs about as well as BALD with acquisition batch size 2500 (from an initial training set size of 5000). 30 trials each.

RepeatedMNIST (Kirsch et al., 2019), and a ResNet-18 (He et al., 2016) for CIFAR-10 (Krizhevsky et al., 2009). We use MC dropout with LeNet-5 (Gal et al., 2017) and for ResNet-18, we change the classifier be deeper and add MC dropout to it as described in Kirsch et al. (2019). We also follow the training regime describe there.

RepeatedMNIST = MNISTx2. RepeatedMNIST (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-BALDoutperforms EPIG-BALDand performs as well as BatchEPIG-BALD, where (Batch)EPIG-BALDcomputes 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.

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-BALDon MNIST, and no worse than than BALD even with very large acquisition sizes on CIFAR-10.

References

Anonymous. Active learning under pool set distribution shift and noisy data. *ICML Workshop Submission*, 2021.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7024–7035, 2019.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

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