Active SG-MCMC with data augmentation

Chinmay Shirore
CICS, UMass Amherst
cshirore@umass.edu

Abstract

Most of the deep learning literature is focused on improving the accuracy of deep learning models over large datasets. But many modern problems present the challenge of learning under data scarcity and hence designing methods tailored to handle such situations are necessary. Active learning and data augmentation are 2 methods that have shown promising results in deep learning literature but have not been used together in bayesian literature. We propose to study the effect of combining these methods for SG-MCMC based Bayesian Deep Learning(BDL) models[3,4] to improve their performance when labeled data is limited. The uncertainty expressed by BDL models can influence the data acquisition process and hence provides a natural setup for active learning. Data augmentation is a relatively simple and fast way to increase the data and we see that using it with active learning results in better generalisation. We present our results on a synthetic dataset and the CIFAR-10 dataset.

1 Introduction

Deep learning methods are data hungry and lack the ability to learn from insufficient data without over fitting. This means that they generalize poorly to new tasks and in practical applications where data is not readily available these models end up over-fitting to the available data. Several methods exist that look at ways to efficiency of the existing data or add new labelled data for training. The most common method that leverages the large amounts of unlabelled data by using an oracle to label some instances from the data pool is called Active Learning[15]. This method employs sampling strategies to acquire subsets of the unlabeled data that may prove to be most informative for the model. Another method called Data Augmentation tackles our original problem in a different way by generating training samples from existing labeled data. It is a cheaper alternative to acquiring new data as we do not need to annotate new data or process the unlabelled data for sampling.

We look at ways to combine both these methods to extract better performance from the model while ensuring the model does not over fit in the process. Bayesian Learning provides a natural setup for this problem as they provide in-built protection against over fitting and a probabilistic interpretation for model uncertainty. This model uncertainty can be used for active learning to acquire informative samples. To this end we look at the SG-MCMC models like SGLD[3] and SGHMC[4] that provide a better approximation of the posterior distribution compared to the more widely used MC-Dropout method. We look at the effects of using BALD or BatchBALD style sampling with SG-MCMC methods and ways to add augmented data for training in this setup. Our results show that, our method does provide greater protection against over fitting while achieving equivalent performance with our baseline. We perform our experiments on synthetic data sets and the well-known CIFAR-10 classification Data set.

2 Novelty Statement

To the best of our knowledge, this is the first work that compares the performance of SG-MCMC methods for Active learning in such settings. We also provide a novel method to incorporate data generated via data augmentation into the active learning pipeline and study the effect of changing the temperature parameter on the same.

3 Related work

Generating new training samples by adding noise or perturbation is one of the most common methods of performing Data Augmentation(DA). But data generated by these methods is not guaranteed to be informative. If the noise added is too little then the model might just be looking at the same data again leading to over fitting, on the other hand a completely noise data point isn't very helpful either. [14] tackle this problem by treating the newly generated points as being missing data points to be generated from the distributions learnt from the training set. They use Generative Adversarial Networks (GANs) to produce the new data points. In our method, we look at popular DA methods that have been well studied before and are known to provide meaningful transformations over the data, using it with Active learning ensures that we only look at informative samples within this newly generated pool of data.

Bayesian Active learning[16] has been well studied and data acquisition functions such as BALD[7] and BatchBALD[8] are known to work well in active learning settings. We aim to combine these 2 useful methods for learning in constrained data settings for Bayesian Models. Such a combination has been studied previously in supervised deep learning setup where [12] and [13] have shown promising results.

In bayesian deep learning, this idea of augmenting generated data for active learning has been looked at in [11]. They use GANs to generate data from existing labeled data such that it is not correctly classified by the classifier resulting in an informative sample. So the generator would generate data points with large augmentation from the original data point such that the classifier changes it's prediction. This is an effective method for generating data points that lie on the edge of the predictive distribution, however, they suggest jointly training the generator and the classifier and thus it increases the training time for their method. To reduce the training time they use a pre-trained GAN model which may not always be available and training it may again lead to increased use of resources. Our method differs from theirs as we add random perturbations to the existing data and create a pool of synthetic data and hence these samples are fairly cheap to obtain. We then pick the most informative samples by computing the model uncertainty over these newly generated samples and picking the most informative via some sampling strategy like BALD. But, there has not been much work in implementing these methods for SG-MCMC methods and is thus the focus of our work.

4 Methodology

From an information theoretic view, model uncertainty is directly affected by the informativeness of the data points used for training. To this end acquisition functions such as BALD[7] and Batch-Bald[8] have been used to find informative data points through uncertainty and diversity sampling. We aim to use these methods on BDL models such as SGLD[3] and SGHMC[4] to evaluate their performance along with different data augmentation strategies. Data augmentation has shown to improve performance for BDL in [14], however in this work, we aim to see how data augmentation and acquisition affects the performance for SG-MCMC methods. We demonstrate our experiments on the ResNet-18 model [10] with SGLD[3] and SGHMC[4], while using SGD for our baseline models.

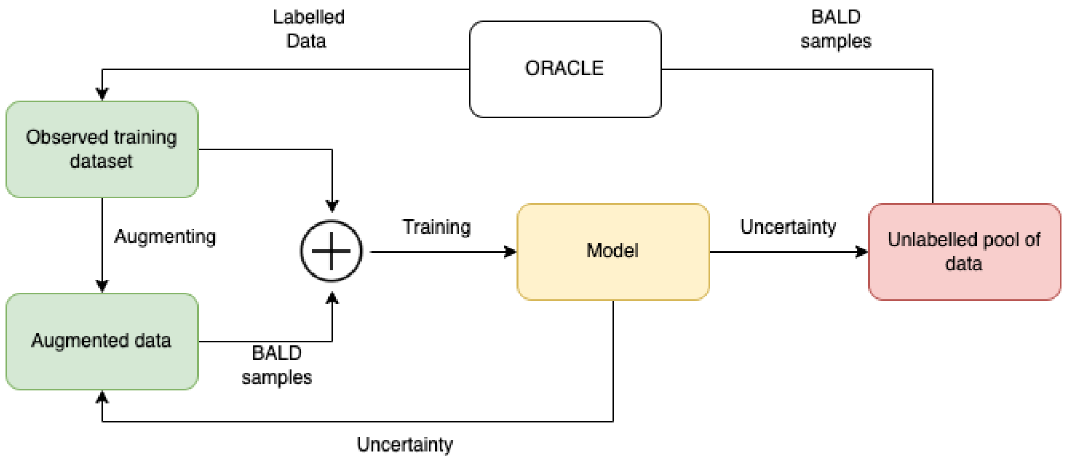

Figure 1: An overview of the Data flow showing how the model is trained in our approach. The model is initially trained on the available data in our training set. We then first augment this data to generate the Augmented data set and compute the model uncertainty over this data. A sampling approach like BALD[7] maybe used to select the most informative samples and are appended to the training batches. While on the other hand similar sampling strategy over the model uncertainty is used to obtain the most informative samples from the unlabelled pool of data which are then labeled by the oracle and added to the training set.

4.1 Active Learning

Active learning offers an efficient framework for obtaining data annotations from unlabeled data sets. But to efficiently sample from this data the model should be able to effectively quantify uncertainty over data which most deep neural networks aren't designed to do. Thus, We employ the most basic form of uncertainty sampling, namely entropy based sampling, over data from the predictions obtained from a deep neural network, here ResNet, trained via SGD optimization to obtain our baseline.

Let us denote the initial labeled data set by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathcal{X}$ are the labeled data points with labels $y_i \in \mathcal{C}$ where $\mathcal{C}$ is the set of class labels. And let the unlabeled pool of data be $\mathcal{D}_{pool}$. We also define the posterior of the parameters θ for the Bayesian Deep Learning model as $p(\theta|\mathcal{D})$. The goal of the sampling step is to obtain new samples x^* and its label y^* are added to the labeled data set for the next training iteration: $\mathcal{D} \leftarrow \mathcal{D}(x^*, y^*)$.

4.1.1 Entropy based Sampling

Entropy-based sampling is a simple strategy that selects the data points with highest entropy. This entropy is computed based on the predicted class probability as shown in this equation.

$$x^* = \underset{x \in \mathcal{D}_{pool}}{\arg\, max} - \sum_{y \in C} p(y|x, \mathcal{D}) log(p(y|x, \mathcal{D}))$$

The entropy would be highest when the predictive distribution is highly unsure and treats each output class as being equally likely for the given data. This is a very common method for uncertainty sampling and hence we use it as our baseline with SGD.

4.1.2 BALD

The sampled informative points x^* are given by using the acquisition function $a(x, \mathcal{M})$,

$$\begin{aligned} x^* &= \underset{x \in \mathcal{D}_{pool}}{\arg\, max}\, a(x, \mathcal{M}) \\ &= \underset{x \in \mathcal{D}_{pool}}{\arg\, max}\, \mathcal{H}[y|x, \mathcal{D}] - \mathbb{E}_{\theta \sim p(\theta|\mathcal{D})}[\mathcal{H}[y|x, \theta]] \end{aligned}$$

Here, the $\mathcal{H}[y|x, \mathcal{D}]$ represents the entropy of the prediction $p(y|x, \mathcal{D})$ and $\mathcal{H}[y|x, \theta]$ represents the entropy of the distribution $p(y|x, \theta)$. For the mutual information represented by the difference between these two values is to be high, the entropy of the model prediction must be high(i.e. the model is uncertain about the prediction) and the entropy over the distribution $p(y|x, \theta)$ is low (i.e it is low when the individual model samples from the posterior are certain about their prediction). Hence, this value is maximised when multiple models are confident about their own prediction but disagree with each other.

Sampling using BALD is very expensive in terms of time and hence we use **BatchBALD**[8] which jointly scores multiple data points by computing the mutual information between the joint of multiple points and model parameters. In our experiments we use BatchBald with our SG-MCMC methods for acquiring new samples from the unlabelled pool of data.

4.2 Data Augmentation Techniques

Data augmentation(DA) is widely used in Computer vision and there are various ways to augment image data. DA operates on individual data points and by adding perturbations can effectively alleviate the problem of overfitting. We look at common operations for augmenting images such as:

1. Random Vertical Flip
2. Random Horizontal Flip
3. Gaussian Blur
4. Random Invert

However, one drawback of DA is that it does not model the information between different samples in different categories and hence is not good at increasing the characteristic information between various samples. Hence, via active learning we only choose data augmentations that the model is most uncertain about, making it more likely for the model to pick images that it feels it has not seen before and boosts the performance of DA. To this end, we maintain a pool of augmented images that are derived from the labeled images in the training pool. As more images are added to the training pool from the unlabeled set, the size of our augmented pool would go on increasing as well.

4.3 Bayesian Deep Learning

As seen in the Active learning section, it is important for a model to be able to accurately quantify the uncertainty over the data points for efficient active learning. And hence we must look at methods that perform accurate sampling from the parameter posterior. From the collected samples from our posterior we approximate the posterior predictive distribution in this manner:

$$\begin{aligned} p(y = c|x, \mathcal{D}) &= \int p(y = c|x, \theta) p(\theta|\mathcal{D}) d\theta \\ &\approx \frac{1}{T} \sum_t p(y = c|x, \theta_t) \end{aligned}$$

The larger the number of samples the better the approximation of the full posterior predictive distribution. The MC-Dropout method is most commonly used for this purpose however SG-MCMC methods are known to provide a more accurate approximation of the posterior and hence we look at ways to utilise them for our setting.

4.3.1 SG-MCMC

Stochastic Gradient-Markov Chain Monte Carlo refers to a family of sampling methods used to draw from the parameter posterior and perform inference over mini-batches of data. They are capable of providing samples from near a mode in the posterior distribution and given enough time also lead to proper mixing across various modes successfully approximating the posterior.
Here we look at 2 popular SG-MCMC sampling methods for the posterior distribution over parameters θ which is given by $p(\theta|\mathcal{D}) \propto exp(-U(\theta))$ and the potential energy is given by $U(\theta) = -logp(\mathcal{D}|\theta) - logp(\theta)$. To perform stochastic updates over a large dataset $\mathcal{D}$, mini-batch of data are used and the gradients are computed over each mini-batch.

Stochastic Gradient Langevin Dynamics(SGLD) is an SG-MCMC algorithm proposed by [3] that adds Gaussian Noise to these stochastic gradients, and the samples are updated as follows:

$$\theta_k = \theta_{k-1} - \alpha_k \nabla \tilde{U}(\theta_k) + \sqrt{2\alpha_k}\epsilon_k$$

here, $\tilde{U}(\theta_k)$ is the minibatch approximate of $U(\theta_k)$ and α_k is the learning rate. Stochastic Gradient Hamiltonian Monte Carlo(SGHMC) is an extension of the well-known HMC method[4] that incorporates a momentum term to the SGLD update. The SGHMC update equation is as follows:

$$\theta_k = \theta_{k-1} + v_{k-1} v_k = v_{k-1} - \alpha_k \nabla \tilde{U}(\theta_k) - \eta v_{k-1} + \sqrt{2(\eta - \hat{\gamma})\alpha_k}\epsilon_k$$

Here, $1 - \eta$ is the momentum term and $\hat{\gamma}$ is the noise estimate.
Now that we have all the necessary pieces for the pipeline, let's look at the dataset used to test our approach.

5 Data Sets

To gain an intuitive and visual understanding of the method we look at a synthetic data set called the Spirals data set. Spirals data set is relatively hard as the data is not linearly separable but SGD can still easily fit to the data set for appropriately chosen hyper-parameters and given enough data . However, SGD fails in data scarce setting for spirals and thus provides a good opportunity for active learning to sample efficiently and lead to quicker convergence. In this data set there are 2 classes and hence is a binary classification problem. We use this data set to visually monitor the behavior of our learning approach and make decisions to set the hyper-parameters that may be used for more complex data sets later on.

We also look at the more complex real-world image classification data set CIFAR-10[2] found here. The CIFAR-10 data set consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. The data is divided into 50,000 images for training and 10,000 images for testing. We further take a random sub-sample of the training data which is 25% of the original data in size(i.e. 12,500 images for training). We see drastically worsened performance for our baseline in this setting. The rest of the data is treated as the unlabeled pool and during each active learning cycle, this data is parsed to find effective data points to acquire.

6 Experiments

In this section we elaborate on our experimental setup for both synthetic as well as CIFAR-10 data set. We look at the evaluation metrics and the baseline for comparing our results.

6.1 Data setup

We use a simple Neural Network with 2 hidden layers, both of size 10 each. The input to the network are the 2D data points in the form of $(x_i, y_i) \in D$ and the labels for each point are given by $c \in 1, 2$ as both are binary classification tasks. We use the Cross Entropy Loss criterion for training. We use the Stochastic Gradient Descent (SGD) optimization algorithm for learning with the Cosine Annealing learning rate scheduler. The initial learning rate is set to 0.01 and goes down to a minimum of 0.0001. We find these values to work best via some manual hyper-parameter tuning.

For CIFAR-10, We use the ResNet-18[10] model for all of our experiments. CIFAR-10 is a fairly complex data set and present variety of data per class which is not straight forward for the model to learn. We use ResNet as it is a widely used deep network which is known to perform well for this task and Data set. ResNet is built up of a few residual blocks that have shortcut connections for the data at any point to be passed forward down the network. We set the learning rate to 0.01 with a cyclical cosine annealing learning rate scheduler having the minimum learning rate set to 0.0001.

6.2 Active Learning

We first setup experiments to understand the performance of different methods in the active learning pipeline and then follow up by adding Data Augmentation to see its effects. There are a few hyper-parameters that we must first tune for the model to perform well in this setting.

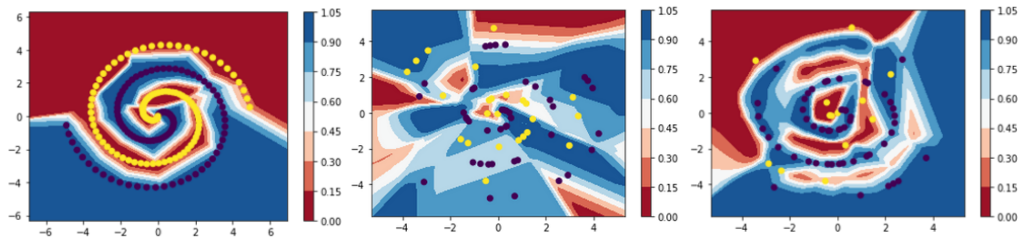

Figure 2: The first figure is the contour plot of the true posterior predictive distribution over the spirals data set. The next 2 plots are the result of running equal active learning cycles starting from a sparse subset and plotted over the held-out test set from the spirals data set. The second figure represents the predictive distribution of the model optimised by SGD with entropy based acquisition function. And, the last figure shows the posterior predictive distribution over samples from SGLD with BALD acquisition function.

Active learning Batch Size This value refers to the amount of new data points acquired during each AL cycle. Although intuitively it makes sense to acquire as much data as possible, that is not necessarily always good. If we pick a value too high, then the model may acquire more number of points that it is uncertain about at the moment, say for example, if the model hasn't seen data points from a particular class, then it may end up acquiring too many data points of this class which may not be very useful overall. Another consideration is that for certain algorithms like batchBald is that having higher Batch size increases the processing time which was a concern during the experimentation phase of this work. We see in the case of CIFAR-10 that the batch size of 512 works the best and use that for the rest of our experiments.

Active learning training epochs Active learning training epochs refers to the number of epochs spent on training over the extended data set after each acquisition. If train on the data for too long at any given point then the model may end up over fitting to this subset of the data set, which is highly undesirable early on in the process. We saw experimentally that as the amount of data goes on increasing we should ideally train for longer on the data as when the training data set size is comparatively large it is representative of the original data set. However, choosing such an iteratively increasing epoch schedule may depend on the dataset we are using and would need further experimentation. So we chose to use constant number of epochs for training between each AL cycle such that for CIFAR-10 we train for 5 epochs and for the synthetic data sets we train for 100 epochs.

6.3 Data Augmentation

We chose the most common data augmentation techniques popular in computer vision literature and see that they work well for our case. The choice of temperature parameter is of interest as we don't know the effect of augmentation on the posterior and hence we conduct experiments with the CIFAR-10 dataset and ResNet-18 Model to see the effect of temperature on the performance.

We perform the following experiments over the CIFAR-10 dataset:

1. SGD + entropy
2. SGLD + BALD
3. SGHMC + BALD
4. SGLD + BALD + DA
5. SGHMC + BALD + DA

Here we refer to the BatchBALD sampling strategy as BALD for brevity. These experiments allow us to see the impact of individual methods and with each other.

7 Results

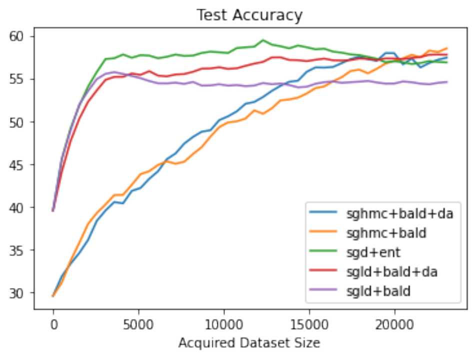

Figure 3: Test Accuracy plot comparing different methods

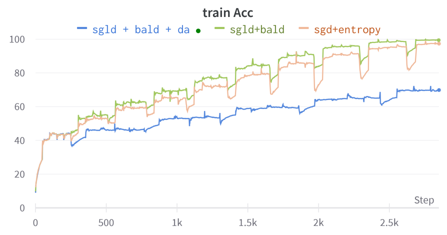

Figure 4: Train accuracy plot illustrating AL acquisitions for sgld methods

Figure 2 shows the Test accuracy over the amount of training data acquired via active learning by each method. We see that our baseline achieves the highest test accuracy at 58.92% relatively quickly with much less data but then this drops down quickly as other methods gradually improve and beat the baseline. This behavior of SGD can be explained by its training curve in figure 3, there we see that the model over fits very quickly to the data as compared to sgld with BatcBALD sampling. The best results are shown by sgld with BatchBald and Data Augmentation as it least over fits to the data and also gives better performance in terms of test accuracy.
The sudden rises in figure 3 illustrate the gain in training performance due to the newly added data. We see slightly more improvement or gain in performance per active learning update for sgld with BatchBALD and DA when compared to the baseline and sgld with just BatchBALD. This shows that our approach performs better when used with SG-MCMC methods in comparison to the baseline.
The test accuracy's achieved by each method are:

1. SGD + entropy = 58.92%
2. SGLD + BALD = 55.83%
3. SGHMC + BALD = 59.24%
4. SGLD + BALD + DA = 58.64%
5. SGHMC + BALD + DA = 59.3%

7.1 Effects of temperature

Data augmentation ends up introducing similar data points to the training set which is equivalent to introducing repeated data points, leading to the shrinking of the posterior distribution around the mode. To see the true effect of data augmentation on the posterior we run sgld with batchBALD and Data augmentation with temperatures being set to the values of 0.01, 0.5, 1 and 2. We see the best results for sgld with batchBALD and Data augmentation at temperature being set to 0.5, although the results aren't drastically varying across runs. One way to obtain better approximation of the posterior for these experiments is to use more samples than we do. We could also run for temperatures at much lower values as is done often as per the cold posterior effect to check for better performance.

8 Discussion and Conclusions

Over multiple experiments we see that adding DA with Active Learning for SG-MCMC methods achieves better performance over time while providing preventing over fitting. There are a bunch of experiments that can be performed over this setup, including a broader hyper-parameter search, larger sample collection from each SG-MCMC cycle and experiments over larger data sets and models.

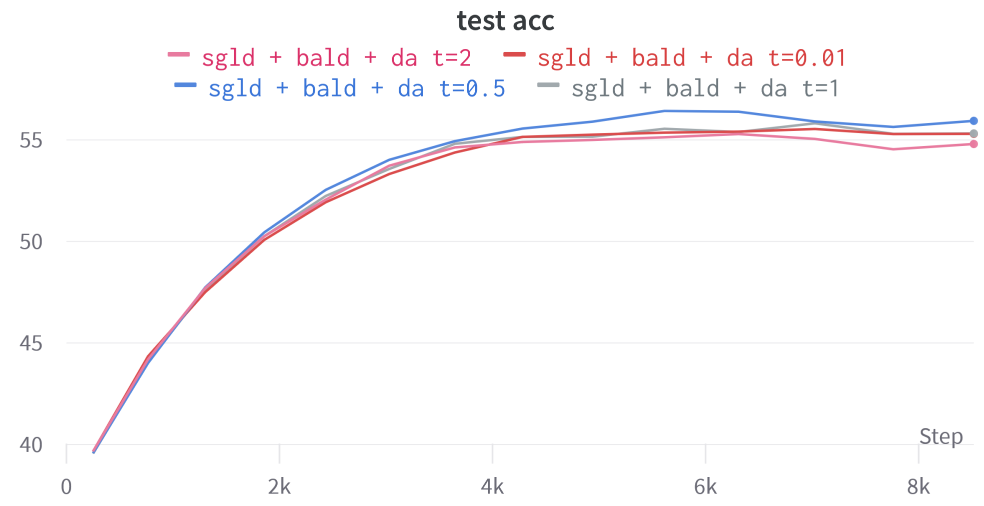

Figure 5: Test accuracy for SGLD trained with Active Learning and Data Augmentation over CIFAR-10 at temperatures 0.01, 0.5, 1 and 2. Although minute we see some performance improvement consistently across time steps with temperature at 0.5.

References

[1] Auer, P., Herbster, M., & Warmuth, M. K. (1995). Exponentially many local minima for single neurons. *Advances in neural information processing systems*, 8.

[2] Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. , , 32–33.

[3] Welling, M., & Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)* (pp. 681-688).

[4] Chen, T., Fox, E., & Guestrin, C. (2014, June). Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning* (pp. 1683-1691). PMLR.

[5] Zhang, R., Li, C., Zhang, J., Chen, C. & Wilson, A. G. (2020). Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning.. *ICLR*

[6] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E. & Weinberger, K. Q. (2017). Snapshot Ensembles: Train 1, get M for free.. *CoRR*, abs/1704.00109.

[7] Houlsby, N., Huszar, F., Ghahramani, Z. & Lengyel, M. (2011). Bayesian Active Learning for Classification and Preference Learning. *CoRR*, abs/1112.5745.

[8] Kirsch, A., Van Amersfoort, J., & Gal, Y. (2019). Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32.

[9] D'Angelo, F., & Fortuin, V. (2021). Repulsive deep ensembles are bayesian. *Advances in Neural Information Processing Systems*, 34.

[10] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[11] Tran, T., Do, T. T., Reid, I., & Carneiro, G. (2019, May). Bayesian generative active deep learning. In *International Conference on Machine Learning* (pp. 6295-6304). PMLR.

[12] Kim, Y. Y., Song, K., Jang, J., & Moon, I. C. (2021). LADA: Look-Ahead Data Acquisition via Augmentation for Deep Active Learning. *Advances in Neural Information Processing Systems*, 34.

[13] Ma, Y., Lu, S., Xu, E., Yu, T., & Zhou, L. (2020, September). Combining Active Learning and Data Augmentation for Image Classification. In *Proceedings of the 2020 3rd International Conference on Big Data Technologies* (pp. 58-62).

[14] Tran, T., Pham, T., Carneiro, G., Palmer, L., & Reid, I. (2017). A Bayesian Data Augmentation Approach for Learning Deep Models. *Advances in neural information processing systems*, 30.

[15] Houlsby, N., Huszar, F., Ghahramani, Z. & Lengyel, M. (2011). Bayesian Active Learning for Classification and Preference Learning. *CoRR*, abs/1112.5745.

[16] Gal, Y., Islam, R., & Ghahramani, Z. (2017, July). Deep bayesian active learning with image data. In International Conference on Machine Learning (pp. 1183-1192). PMLR.