ECAI 2023
K. Gal et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230402

Understanding and Improving Neural Active Learning on Heteroskedastic Distributions

Savya Khosla[a], Chew Kin Whye[b], Jordan Ash[c], Cyril Zhang[c], Kenji Kawaguchi[b] and Alex Lamb[c]

[a]University of Illinois Urbana-Champaign
[b]National University of Singapore
[c]Microsoft Research NYC

Abstract. Models that can actively seek out the best quality training data hold the promise of more accurate, adaptable, and efficient machine learning. Active learning techniques often tend to prefer examples that are the most difficult to classify. While this works well on homogeneous datasets, we find that it can lead to catastrophic failures when performed on multiple distributions with different degrees of label noise or heteroskedasticity. These active learning algorithms strongly prefer to draw from the distribution with more noise, even if their examples have no informative structure (such as solid color images with random labels). To this end, we demonstrate the catastrophic failure of these active learning algorithms on heteroskedastic distributions and propose a fine-tuning-based approach to mitigate these failures. Further, we propose a new algorithm that incorporates a model difference scoring function for each data point to filter out the noisy examples and sample clean examples that maximize accuracy, outperforming the existing active learning techniques on the heteroskedastic datasets. We hope these observations and techniques are immediately helpful to practitioners and can help to challenge common assumptions in the design of active learning algorithms. Our code is available at this URL.

1 Introduction

In an active learning setup, a model has access to a pool of labeled and unlabeled data. After training on the available labeled data, a selection rule is applied to identify a batch of k unlabeled examples to be labeled and integrated into the training set before repeating the process. Under this paradigm, data is considered to be abundant, but label acquisition is costly. An active learning algorithm aims to identify unlabeled examples that, once labeled and used to fit model parameters, will elicit the most performant hypothesis possible given a fixed labeling budget. To fulfill this objective, a selection criteria generally follows two heuristics: (1) select diverse examples and (2) select examples where the model has a high degree of uncertainty.

The presence of noise is an unavoidable problem that corrupts real-world datasets [41] and is detrimental to the performance of classifiers directly trained on them. Active learning can seek to combat this problem through a robust data-selection pipeline that effectively filters out the noisy data and select the most informant examples for machine learning, allowing us to efficiently leverage the abundance of data we have at our disposal.

In our study, we explore the performance of the active learning algorithms on *heteroskedastic distributions*, where the training data consists of a mixture of distinct distributions with different degrees of noise. One use case of active learning on heteroskedastic distributions is in reinforcement learning, where the agent gathers its own training data through its actions and obtains the feedback from the environment. Often, the environment has heteroskedastic noise. For example, certain actions like "opening a box with a question mark symbol" have random outputs, whereas other actions like "moving left/right" have deterministic outputs. Another possible use case of our study could be for training LLMs using techniques like self-instruct [42]. Recent works [42, 37, 45] have shown that curating a dataset using large language models could help generate more diverse training examples. However, this method of curating data is inherently noisy, and therefore the data pipeline uses a critical filtering step to remove redundant and non-informative examples. We believe that the findings in this paper could also help reduce the dependency on clean data allowing practitioners to train models on larger sets.

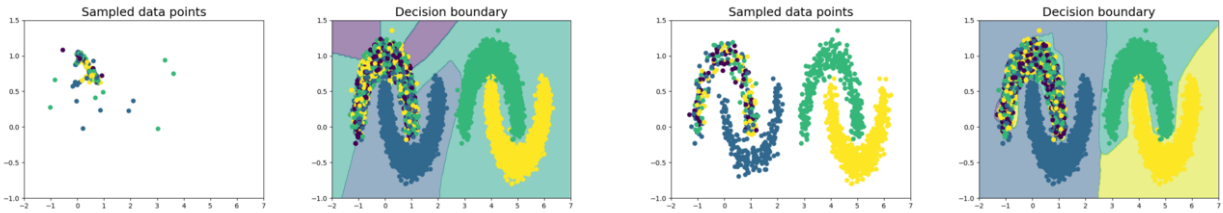
We find that preferring examples with high uncertainty often works well on homogeneous datasets but can lead to catastrophic failure when training on heteroskedastic distributions. Uncertainty-based active learning algorithms typically rely on notions of model improvement that are unable to disambiguate aleatoric uncertainty from epistemic uncertainty, thereby over-selecting examples for which the model is unconfident but which are unlikely to improve the current hypothesis. We produce a generalization bound that explains why this phenomenon occurs, which seems to superficially contradict previous theory [22] that showed training only on examples with high loss could generalize as well as training on randomly selected examples.

Further, we show that this inefficiency in the data-selection process can be mitigated in three ways.

1. **Favoring diversity over uncertainty.** As we demonstrate empirically in this work, active learning algorithms that promote the selection of a diverse set of examples can efficiently filter out the data points with heteroskedastic noise since the noisy examples come from the same distribution and therefore have similar feature representations.

2. **Leveraging high confidence examples in the unlabeled pool of data.** We show that the examples in the unlabeled pool for which the model is highly confident can promote better feature learning, thereby improving the performance of active learning algorithms even in the presence of heteroskedastic noise in the dataset.

3. **Encouraging the selection of examples for which model's representations change over training iteration.** We show that the model's representation for examples with heteroskedastic noise con-



(a) Least-Confidence Sampling: Test accuracy = 53.67%

(b) LHD Sampling: Test accuracy = 84.68%

Figure 1: We construct *Four-Moons Dataset*, a toy dataset with heteroskedastic noise. In this dataset, we have four classes - blue, green, yellow, and purple. The data points belonging to the purple class (top-left moon) are assigned uniformly random labels, while the points in other moons has no noise. The least-confidence sampling (uncertainty-based algorithm) selects examples almost exclusively from the noisy class (top-left moon), resulting in a poor decision boundary. The proposed LHD algorithm, on the other hand, promotes selection of clean data points and almost perfectly solves the classification problem.

verges quickly to a suboptimal solution. This can be used as a helpful signal to filter out these noisy examples.

The change in model's representation is measured using the difference between a conventionally-trained model and an exponential moving average (EMA) of its iterates. For the noisy examples in the dataset, both the conventionally-trained model and the EMA model converges quickly to the suboptimal solution. Consequently, the EMA difference is nearly zero for these examples, and thus, it can be used as a helpful signal to filter out the noise from the dataset. Further, as we show later in the paper, the EMA difference is maximum for examples that are difficult to classify (but not noisy), thereby promoting the selection of challenging yet clean data.

The toy example with heteroskedastic noise in Figure 1 shows how the least-confidence sampling (uncertainty-based algorithm) selects examples almost exclusively from the noisy class (top-left moon), resulting in an extremely poor decision boundary. The Coreset algorithm (diversity-based algorithm) and the proposed LHD algorithm promotes selection of clean data points and almost perfectly solves the classification problem.

The main contributions of this work are:

1. We study the performance of active learning algorithms on heteroskedastic distributions and show that algorithms that exclusively prefer uncertainty can catastrophically fail in the presence of heteroskedasticity.
2. We produce a generalization bound that explains why training only on low confidence examples can lead to poor performance in the presence of heteroskedastic noise.
3. We explore a fine-tuning-based approach that helps improve the performance of all algorithms in the presence of heteroskedasticity.
4. We propose an algorithm, hereafter referred to as LHD, that performs comparably to the existing state-of-the-art algorithms in the general setup and, when coupled with fine-tuning, outperforms all algorithms by a significant margin.

2 Related Work

Neural active learning. Active learning is an extremely well-researched area, with the richest theory developed for the convex setting [12, 10, 9]. More recently, however, there have been several attempts to tractably generalize active learning to the deep regime. Such approaches can be thought of as identifying batches of samples that cater more to either the model's predictive uncertainty or to the diversity of the selection.

In the former approach, a batch of points is selected in order of the model's uncertainty about their label. Many of these methods query samples that are nearest the decision boundary, an approach that's theoretically well understood in the linear regime when the batch size is 1 [38]. Some deep learning-specific approaches have also been developed, including using the variance of dropout samples to quantify uncertainty [15], and adversarial examples have been used to approximate the distance between an unlabeled sample and the decision boundary. In the deep setting, however, where models are typically retrained from scratch after every round of selection, larger batch size is usually necessary for efficiency purposes.

For large acquisition batch sizes, algorithms that cater to diversity are usually more effective. In deep learning, several methods take the representation obtained at the penultimate layer of the network and aim to identify a batch of samples that might summarize this space well [32, 17, 18]. Other methods promote diversity by minimizing an upper bound on some notion of the model's loss on unseen data [43, 11, 44, 23]. This approach has also been taken to a trade-off between diversity and uncertainty in deep active learning [3, 4].

Data poisoning, distributional robustness, and label noise. A related body of work seeks to obtain models and training procedures that are robust against *worst-case* perturbations to the data distribution. For recent treatments of this topic and further references, see [36, 31]. A few recent works have considered data poisoning in the active learning setting [26, 39], with defenses focusing on modifying the setting rather than the algorithm. Further, some existing works in active learning regime [24, 14] consider the presence of label noise, out-of-distribution examples, and redundancy in the dataset. Our work, however, considers the setting wherein the system suffers from low-quality labels (e.g., in medical diagnosis, where the labelers are not always adept at assigning the correct label to the example queried by the algorithm and might end up incorrectly assigning out-of-distribution examples to one of the classes in the label space).

Heteroskedasticity in machine learning. The issues of class imbalance and heteroskedasticity are of interest in the supervised learning setting [33, 8], in which various methods have been proposed to make training more robust to these distributions. Our work seeks to initiate the study of the orthogonal (but analogous) issue in the sample *selection* regime. Similar to our work, [1] considers a theoretical regression problem that explores active learning in heteroskedastic noise. However, contrary to our setup, it *leverages* heteroskedasticity in the pool of labeled data to sample more observations from the parts of the input space with large variance.

Semi-supervised active learning. Recent advances in semi-supervised learning (SSL) have demonstrated the potential of using

unlabeled data for active learning. For instance, [46] combines SSL and AL using a Gaussian random field model. [16] proposes to human label the unlabelled example for which the different augmented views result in inconsistent SSL model's predictions because such behavior indicates that the model cannot successfully distill helpful information from that unlabelled example. Similarly, [7] proposes to use a model trained using SSL to select a batch of unlabeled examples that best summarizes a pool of data pseudo-labeled by the model itself. [13, 29] leverage active learning and semi-supervised learning in succession to show incremental improvements in speech recognition and object detection, respectively. [34] learns the sampling criteria by setting up a mini-max game between a variational auto-encoder that generates latent representations for labeled and unlabeled data and an adversarial network that tries to discriminate between these representations. In this work, we experiment with a very simple SSL setup to see its effectiveness when performing active learning in the presence of heteroskedastic noise.

3 Heteroskedastic Benchmarks for Neural Active Learning

We introduce three benchmarks for active learning on heteroskedastic distributions. In all cases, we introduce an additional set of N examples with purely random labels to the original clean data. K is the number of unique noisy examples, since some of the examples are repeated.

The model is not given information on which samples are noisy/clean, but it is reasonably predictable from the example's features since the noisy datapoints are from the same distribution. This distinguishes our benchmarks from IID label noise, which is not predictable based on the example's features. These constructions are described below and summarized in Figure 2.

Noisy-Blank: We introduce N examples that are all solid black ($K = 1$) and have a random label $y \sim U(1, n_y)$, where n_y refers to the number of classes.

Noisy-Diverse: We increase the difficulty by introducing $K = 100$ different types of examples, where each type is a random solid color and has a label randomly drawn from three label choices that are unique to that color. N such noisy examples are introduced to the dataset. This benchmark is designed to make the heteroskedastic distribution more diverse while still keeping the noisy examples simple.

Noisy-Class: In our most challenging setting, we take K examples from a particular class (say, $y = 1$) and assign these examples uniformly random labels $y \sim U(1, n_y)$. We then randomly repeat these examples to give N noisy examples. In this case, the randomly labeled examples are challenging but still possible to identify.

We designed these benchmarks to easily evaluate the performance of existing algorithms. Despite being highly simplistic, these benchmarks do delineate a shortcoming of the existing active learning algorithms, and we believe that practitioners should make their active learning pipelines robust to such noisy adversaries. In this regard, we can draw an analogy with the domain of adversarial learning – while the adversary will not have access to the model weights and gradients in most realistic setups, practitioners want their pipelines to be robust to white-box adversarial attacks like Projected Gradient Descent.

Future work can be done to generate more realistic datasets with heteroskedastic noise.

4 Method

In this section, we describe (1) the baseline active learning algorithms with which we experiment, (2) LHD, an active learning algorithm that leverages EMA difference to sample examples for which the model's representation changes over training iterations, (3) the fine-tuning technique used to improve the performance of the baseline algorithms. and (4) the combination of LHD with fine-tuning.

4.1 Baselines

We review some prominent neural active learning algorithms, which act as baselines in our study.

Random sampling (RAND). Unconditional random sampling from the unlabeled pool of data.

Least confidence sampling (CONF). Confidence sampling selects the k unlabeled points for which the most likely label has the smallest probability mass [40]:

$$x^*_{\mathsf{CONF}} = \arg\min_x P_\theta(\hat{y}|x)$$

Here, $P_\theta(\hat{y}|x)$ is the probability of the predicted (most likely) class $\hat{y}$ given the input x and model parameters θ, and x^*_{CONF} is the selected batch of data points.

Margin sampling (MARG). Margin sampling selects the k points for which the difference in probability mass in the two most likely labels is smallest [30]:

$$x^*_{\mathsf{MARG}} = \arg\min_x P_\theta(\hat{y}_1|x) - P_\theta(\hat{y}_2|x)$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable classes, respectively.

Bayesian Active Learning by Disagreements (BALD). Here, the objective is to select k points that maximize the decrease in expected posterior entropy [20]:

$$x^*_{\mathsf{BALD}} = \arg\max_x H[\theta|D] - \mathbb{E}_{y\sim p(y|x,D)}[H[\theta|y, x, D]]$$

where $H[\cdot]$ represents entropy, θ represents model parameters, and D represents the datatset.

Coreset sampling (CORESET). The Coreset algorithm is a diversity-based approach that aims to select a batch of representative points, as measured in penultimate layer space of the current state of the model [32]. We refer to the function for computing this penultimate layer as $h(x)$. It proceeds in these steps on each acquisition round:

(1) Given a set of existing selected unlabeled examples and labeled examples x^*_{CORESET} and a set of indices of these selected examples s.

(2) Select an example with the greatest distance to its nearest neighbor in the hidden space

$$u = \arg\max_{i\in[n]\setminus\mathbf{s}} \min_{j\in\mathbf{s}} \Delta(h(\mathbf{x}_i), h(\mathbf{x}_j))$$

(3) Set $s = s \cup \{u\}$ and $x^*_{\mathsf{CORESET}} = x^*_{\mathsf{CORESET}} \cup \{X_u\}$.

(4) Repeat this in an active learning round until we reach the acquisition batch size.

Batch Active learning by Diverse Gradient Embeddings (BADGE). BADGE is a hybridized approach, meant to strike a balance between uncertainty and diversity. The algorithm represents data in a hallucinated gradient space before performing diverse selection using the k-means++ seeding algorithm [4]. It proceeds with these steps on each acquisition round:

(1) Compute hypothetical labels $\hat{y}(x) = h_{\theta_t}(x)$ for all unlabeled examples.

(2) Compute gradient embedding for each unlabeled example

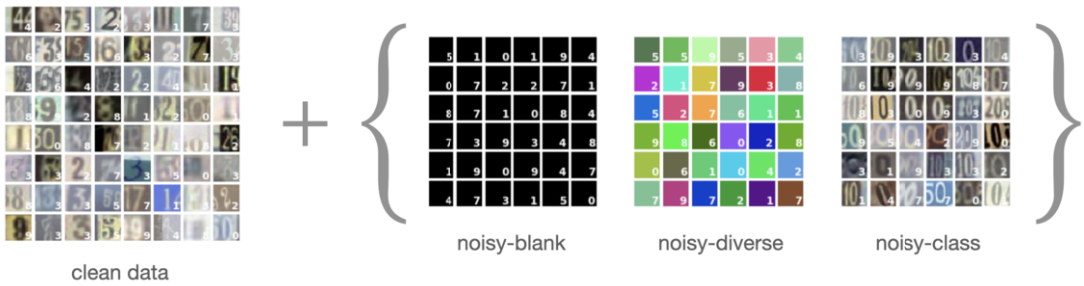


Figure 2: The heteroskedastic distributions proposed in this paper. The SVHN dataset is corrupted with randomly labeled examples, with (1) black images, (2) diverse coloured images, and (3) images from one class.

$$g_x = \frac{\partial}{\partial \theta_{\text{out}}} \ell(f(x;\theta), \hat{y}(x))|_{\theta=\theta_t}$$

where θ_{out} refers to the parameters of the output layer.

(3) Use k-means++ over the gradient embedding vectors g_x over all unlabeled examples to select a batch of examples x^*_{BADGE}.

4.2 LHD: *Increasing Sampling Where Representations Change Across Training Iterations*

For *noisy* examples, conflicting gradients result in the model converging quickly to a suboptimal solution and undergoing little change throughout the training. For the *clean* examples, the model converges slowly to an optimal solution, undergoing changes throughout the training (Figures 5 and 6 in the Appendix at this link substantiate this claim). Therefore, by encouraging the selection process in active learning to select the examples for which the model converges slowly, we can maximize the sampling of *clean* examples, improving the performance in the heteroskedastic setting.

To measure the convergence rate, in addition to the main model F_θ, we introduce an exponential moving average (EMA) of the model F_β in the training pipeline. The EMA model has the same architecture as the main model but uses a different set of parameters β, which are exponentially moving averages of θ. That is, at epoch t, $\beta_{t+1} \leftarrow \alpha \cdot \beta_t + (1-\alpha) \cdot \theta_t$, for some choice of decay parameter α.

The convergence rate for a training example is captured as the state difference between the main and the EMA model, which is measured in two ways:

(1) **Loss Difference** Δl: The absolute difference between the loss values of an example from the EMA model and the main model:

$$\Delta l = \mid l_{\text{ema}} - l_{\text{main}} \mid$$

For the unlabeled examples, we assume the prediction of the EMA model as the ground truth for loss computation.

(2). **Hidden State Difference** $\Delta \mathbf{h}$: The difference between the hidden feature representation from the penultimate layer of the EMA model and the main model:

$$\Delta \mathbf{h} = \mathbf{h}_{\text{ema}} - \mathbf{h}_{\text{main}}.$$

Δl will be low for *noisy* examples because both l_{main} and l_{ema} are high throughout the training. Similarly, Δl will be low for *simple-clean* examples because both l_{main} and l_{ema} are low for most part of the training. For *difficult-clean* examples, however, the main model converges slowly, which leads to an even slower convergence of the EMA model. As a result, at some point in the training, we get a low l_{main} but a high l_{ema} for *difficult-clean* examples, resulting in a high Δl. Using a similar line of reasoning, we can infer that $\Delta \mathbf{h}$ will have a smaller magnitude for the *noisy* examples and *simple-clean* examples, and a larger magnitude for *difficult-clean* examples. Further, examples that are similar to one another will have similar $\Delta \mathbf{h}$, and a diversity-based sampling technique that operates on $\Delta \mathbf{h}$ will promote sampling of a diverse batch of examples.

We use Δl and $\Delta \mathbf{h}$ to obtain the final **State Difference** $\mathbf{lh}$ as:

$$\mathbf{lh} = \Delta l \cdot \Delta \mathbf{h}.$$

A training example with a small $||\mathbf{lh}||_2$ has a small state difference between the main and EMA model, meaning the example had converged very fast early on in the training, indicating that it is probably a *noisy* example. (This can be seen from Figure 7a in the Appendix).

Algorithm 1 describes LHD in detail. The state difference, $\mathbf{lh}$, is computed for all unlabeled examples. Then, we use the k-means++ seeding algorithm [2] over all the $\mathbf{lh}$ embeddings, which selects a batch of diverse examples that have a high magnitude of $\mathbf{lh}$.

Algorithm 1 LHD: **L**oss and **H**idden state **D**ifference sampling

Require: Main model F_θ, EMA model F_β, unlabeled pool of examples U, initial number of examples M, number of active learning iterations T, decay parameter α, cross-entropy loss function $\text{CE}(\cdot)$, one-hot function $\text{OH}(\cdot)$.

1: Labeled dataset $S \leftarrow M$ examples drawn uniformly at random from U together with their labels y.
2: Train an initial main model F_{θ_1} on S while updating the initial EMA model F_{β_1} using $\beta_1 \leftarrow \alpha \cdot \beta_1 + (1-\alpha) \cdot \theta_1$ in each training iteration
3: **for** $t = 1, 2, ..., T$ **do**
4: Optionally fine-tune F_{θ_t} and F_{β_t} using Algorithm 2
5: **for** all examples $x \in U \setminus S$ **do**
6: $y_{\text{pseudo}_x} = \text{OH}(F_{\beta_t}(x))$
7: $l_{\text{ema}_x} = \text{CE}(F_{\beta_t}(x), y_{\text{pseudo}_x})$ and $\mathbf{h}_{\text{ema}x} = \hat{F_{\beta_t}}(x)$, where $\hat{F_{\beta_t}}(\cdot)$ represents a function to extract penultimate layer
8: $l_{\text{main}_x} = \text{CE}(F_{\theta_t}(x), y_{\text{pseudo}_x})$ and $\mathbf{h}_{\text{main}x} = \hat{F_{\theta_t}}(x)$, where $\hat{F_{\theta}}(\cdot)$ represents a function to extract penultimate layer
9: $\Delta l_x = \mid l_{\text{ema}_x} - l_{\text{main}_x} \mid$ and $\Delta \mathbf{h}_x = \mathbf{h}_{\text{ema}x} - \mathbf{h}_{\text{main}x}$
10: $\mathbf{lh}_x = \Delta l_x \cdot \Delta \mathbf{h}_x$
11: **end for**
12: $S_t \leftarrow$ a subset of $U \setminus S$ using the k-means++ seeding algorithm on $\{\mathbf{lh}_x : x \in U \setminus S\}$ and query their labels
13: $S \leftarrow S \cup S_t$
14: Train $F_{\theta_{t+1}}$ on S by minimizing $E_S[\text{CE}(F_{\theta_t}(x), y)]$ and in each training iteration, update $F_{\beta_{t+1}}$ using $\beta_{t+1} \leftarrow \alpha \cdot \beta_t + (1-\alpha) \cdot \theta_t$
15: **end for**

Algorithm 2 Fine-tuning algorithm for improved example selection

Require: Main model F_θ, EMA model F_β, unlabeled pool of examples U, labeled pool of examples S, number of fine-tuning iterations T, decay parameter α, confidence threshold γ, data augmentation function $\mathrm{A}(\cdot)$, cross-entropy loss function $\mathrm{CE}(\cdot)$, one-hot function $\mathrm{OH}(\cdot)$.

Initialize $V \leftarrow \{\}$ a set of example selected for fine-tuning
Initialize $y \leftarrow \{\}$ pseudo-labels for the fine-tuning examples
for all examples $x \in U \setminus S$ **do**
  Compute confidence $c_x = \max(F_{\theta_t}(x))$
  if $c_x > \gamma$ **then**
    $y \leftarrow y \cup \{\mathrm{OH}(F_{\beta_t}(x))\}$
    $V \leftarrow V \cup \{\mathrm{A}(x)\}$
  end if
end for
for $t = 1, 2, ..., T$ **do**
  Train model $F_{\theta_{t+1}}$ on V by minimizing $E_V[\mathrm{CE}(F_{\theta_t}(x), y)]$ and in each training iteration, update $F_{\beta_{t+1}}$ using $\beta_{t+1} \leftarrow \alpha \cdot \beta_t + (1-\alpha) \cdot \theta_t$
end for

4.3 *Fine-tuning: Using Unlabeled Data for Tackling Heteroskedasticity*

Traditionally, active learning algorithms train models on a small set of labeled data and use a large pool of unlabeled data only for sampling informative examples. However, we conjecture that the unlabeled pool can be efficiently leveraged to improve the performance of active learning algorithms even in the presence of heteroskedasticity. To this end, we experiment with an extremely simple semi-supervised learning technique (similar to [35]) to aid active learning.

After training the model on the labeled data points, we sample a batch of examples from the unlabeled pool for which the model is highly confident. Using the predicted labels for these examples as the ground truth, we fine-tune the model on strongly augmented versions of these confident examples. Since the model is inherently less confident in the noisy examples, most of the examples used for fine-tuning are clean. This way, we leverage the information in already well-classified clean examples, and the model learns more discriminative feature representations. Algorithm 2 outlines the fine-tuning procedure.

Since the fine-tuning technique allows us to exploit the information in the unlabeled data pool, all active learning methods benefits from the use of fine-tuning. Additionally, since the fine-tuning technique improves the quality of the representations, active learning algorithms that rely on these representations for the data selection obtains a more substantial benefit, especially in the earlier rounds where labeled data is scarce.

4.4 *LHD with Fine-tuning*

Even though the LHD method is able to filter out the noisy examples, it can still struggle to differentiate between the *simple-clean* examples - examples from the original/clean data that are very easy to classify, and the *difficult-clean* examples - examples from the original/clean data that are challenging for the model to classify.

When we add the fine-tuning method on top of LHD, the average $||\mathbf{lh}||_2$ for the *difficult-clean* examples becomes much higher than the $||\mathbf{lh}||_2$ for *simple-clean* examples. This can be seen from Figure 7b in the Appendix.

Essentially, the fine-tuning method trains on the examples that the model is confident in, which are mostly the *simple-clean* examples. Therefore, the model will converge quickly to an optimal solution for the *simple-clean* examples, undergoing little change later in training. On the other hand, since the model is not fine-tuned on the *difficult-clean* examples, it takes a longer time to learn the correct solution, converging slowly to an optimal solution and undergoing changes throughout the training.

So, by encouraging the selection of examples for which the model converges slowly (the idea behind LHD) and fine-tuning the model on confident examples (the idea behind fine-tuning), we can maximize the sampling of *difficult-clean* examples, thereby improving the performance of our active learning algorithm in heteroskedastic settings.

Empirically analyzing Δl in this setting further substantiates the abovementioned claim. For *noisy* examples, both l_{main} and l_{ema} are high throughout the training, resulting in a small Δl. For *simple-clean* examples, both l_{main} and l_{ema} are low throughout the training, resulting in a small Δl. For *difficult-clean* examples, since the loss decreases slowly, l_{main} will be low while l_{ema} is high, resulting in a large Δl. On similar lines, $||\Delta\mathbf{h}||_2$ has a larger value for *difficult-clean* examples and a smaller value for *simple-clean* and *noisy* examples.

Table 1: Classification accuracy on CIFAR-10 with a Resnet model after 10 rounds of active learning. (+*FT* represents Fine-tuning)

Method	Clean	Noisy-Blank	Noisy-Diverse	Noisy-Class
RAND	41.09 ± 0.43	38.27 ± 0.27	32.70 ± 0.18	32.11 ± 0.42
CONF	38.27 ± 1.48	32.88 ± 0.25	39.53 ± 0.08	27.04 ± 0.26
MARG	37.86 ± 4.33	39.16 ± 0.83	26.34 ± 3.25	30.40 ± 2.46
BALD	43.31 ± 1.34	46.90 ± 0.39	35.64 ± 0.32	31.65 ± 0.30
CORESET	41.81 ± 1.17	47.19 ± 2.03	47.33 ± 3.31	40.31 ± 1.51
BADGE	41.52 ± 1.04	47.84 ± 0.38	42.40 ± 1.66	39.33 ± 1.74
LHD	43.70 ± 1.10	46.28 ± 1.69	41.25 ± 1.39	40.43 ± 1.92
Method	**Clean**	**Noisy-Blank**	**Noisy-Diverse**	**Noisy-Class**
RAND + *FT*	59.64 ± 0.63	45.45 ± 0.72	40.31 ± 1.03	43.12 ± 0.82
CONF + *FT*	61.40 ± 1.39	58.66 ± 2.07	57.09 ± 2.79	49.78 ± 2.35
MARG + *FT*	61.91 ± 0.46	58.33 ± 0.01	52.94 ± 2.11	52.46 ± 24.18
BALD + *FT*	66.64 ± 0.66	65.31 ± 0.20	44.34 ± 1.34	53.00 ± 0.40
CORESET + *FT*	64.00 ± 0.90	61.13 ± 0.31	59.25 ± 0.06	53.01 ± 0.15
BADGE + *FT*	64.40 ± 0.76	64.30 ± 1.26	61.36 ± 0.57	53.78 ± 0.28
LHD + *FT*	75.35 ± 0.21	75.78 ± 0.99	70.50 ± 0.56	64.34 ± 0.10

Table 2: Classification accuracy on SVHN with a Resnet model after 10 rounds of active learning. (+*FT* represents Fine-tuning)

Method	Clean	Noisy-Blank	Noisy-Diverse	Noisy-Class
RAND	80.06 ± 0.20	80.35 ± 0.46	71.49 ± 1.30	53.93 ± 0.98
CONF	77.84 ± 2.45	75.84 ± 2.68	76.41 ± 1.70	30.00 ± 4.71
MARG	79.11 ± 0.77	79.84 ± 1.57	53.82 ± 3.59	38.45 ± 3.97
BALD	78.76 ± 1.69	84.69 ± 2.57	69.10 ± 4.13	43.06 ± 0.76
CORESET	80.71 ± 0.78	86.97 ± 0.94	86.61 ± 0.21	65.52 ± 0.40
BADGE	80.48 ± 0.81	86.90 ± 1.46	84.41 ± 1.01	59.98 ± 0.55
LHD	78.50 ± 0.99	86.97 ± 0.12	82.98 ± 0.67	66.01 ± 0.76
Method	**Clean**	**Noisy-Blank**	**Noisy-Diverse**	**Noisy-Class**
RAND + *FT*	90.60 ± 0.67	85.9 ± 0.38	71.49 ± 1.30	53.93 ± 0.98
CONF + *FT*	89.92 ± 0.81	89.85 ± 1.03	83.31 ± 2.40	57.28 ± 1.02
MARG + *FT*	89.73 ± 1.01	90.76 ± 0.45	84.74 ± 2.45	68.12 ± 3.79
BALD + *FT*	91.10 ± 0.23	90.90 ± 0.36	67.21 ± 1.34	63.73 ± 4.80
CORESET + *FT*	87.81 ± 1.58	89.21 ± 0.59	87.46 ± 0.24	70.92 ± 0.36
BADGE + *FT*	92.49 ± 1.12	91.27 ± 0.87	88.76 ± 0.63	71.50 ± 0.22
LHD + *FT*	94.26 ± 0.12	93.54 ± 0.23	90.92 ± 0.84	75.51 ± 1.09

5 Experiments

We experimented with different active learning algorithms, datasets, and noising benchmarks. In all experiments, we started with 2000

labeled points and queried $k = 1000$ examples in each round of active learning, for a total of 10 rounds. We evaluate the performance of two benchmarking datasets - (1) CIFAR10 [25], which contains colored images belonging to 10 classes, and (2) SVHN [28], which contains images of street numbers on houses. In all cases, the data pool comprises 80% noisy data points and 20% clean data points. Experiments were conducted on the ResNet18 architecture. Our implementation uses the BADGE's codebase [4].

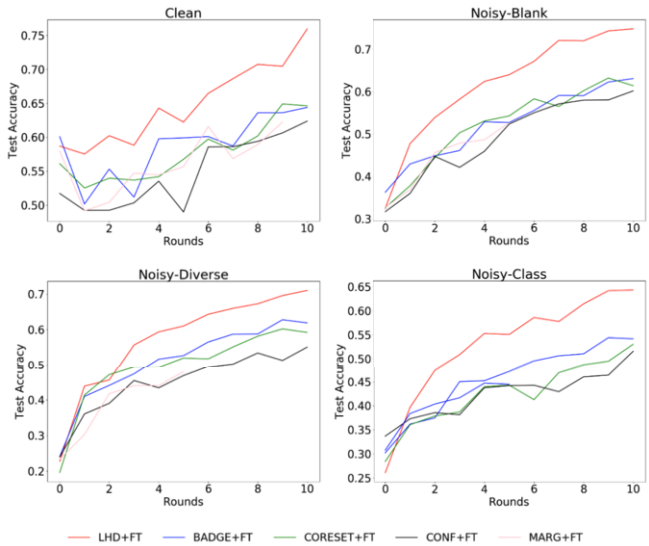


Figure 3: Performance over multiple rounds of active learning acquisition on CIFAR10 dataset. The training is complemented with fine-tuning. LHD outperforms other techniques by a significant margin.

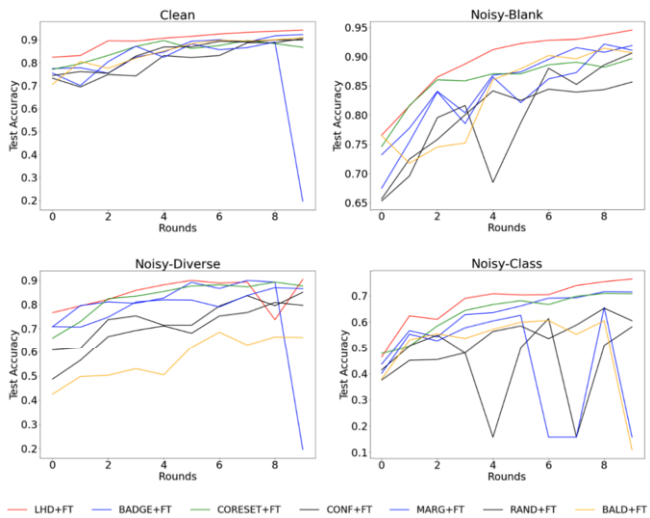


Figure 4: Performance over multiple rounds of active learning acquisition on SVHN dataset. The training is complemented with fine-tuning. LHD outperforms other techniques by a significant margin.

From the analysis of the final results for CIFAR-10 and SVHN shown in Table 1 and Table 2 respectively, we draw 3 main conclusions.

Firstly, we see that the uncertainty-based techniques (CONF and MARG) perform sub-optimally on the noisy data, even worse than random sampling. Methods that factor in diversity has much better performance. This results are supported by Table 3 and Table 4 (Appendix), which shows the percentage of clean (non-noisy) examples which are selected over the course of training by different active learning algorithms. The uncertainty-based techniques sample mainly noisy examples, whereas diversity based methods select mainly clean examples. The proposed LHD method has comparable performance to the other diversity based methods.

Secondly, when the fine-tuning technique is added, we observe a significant and consistent boost in the performance of all algorithms across all settings.

Lastly, with fine-tuning, the proposed LHD algorithm outperforms all other techniques. We analyze the performance of the techniques with fine-tuning throughout the 10 active learning rounds, as shown in Figure 3 and Figure 4, which shows that LHD outperforms all the other techniques throughout active learning rounds. Furthermore, Figure 4 demonstrates catastrophic failure of uncertainty-based active learning techniques in certain rounds of active learning, which happens due to over sampling of noisy examples in those rounds.

Additional experiments for different levels of noise can be found in Table 5, and Table 6 (ref. Appendix) shows the time taken by different active learning algorithms to complete 10 rounds of acquisition on a Tesla V100 GPU. We see that LHD takes much less time compared to the diversity-based techniques, viz. BADGE and CORESET.

6 Theoretical Analysis of Confidence Sampling in Heteroskedastic Setting

While previous work [22] has suggested that selecting high-loss examples can accelerate training, our experiments show that selecting examples with the lowest prediction confidence can fail catastrophically on heteroskedastic distributions. In this section, we provide a theoretical account of why training only on high loss examples (which would have low confidence given a well-calibrated model) can lead to poor performance on heteroskedastic distributions.

6.1 Notation

Let $\mathcal{D} = ((x_i, y_i))_{i=1}^n$ be a training dataset of n samples where $x_i \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$ is the input vector and $y_i \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$ is the target vector for the i-th sample. A standard objective function is

$$L(\theta; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^n L_i(\theta; \mathcal{D})$$

where $\theta \in \mathbb{R}^{d_\theta}$ is the parameter vector of the prediction model $f(\cdot\,; \theta) : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$, and $L_i(\theta; \mathcal{D}) := \ell(f(x_i; \theta), y_i)$ with the function $\ell : \mathbb{R}^{d_y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is the loss of the i-th sample.

Similar to the notation of order statistics, we first introduce the notation of ordered indexes: given a model parameter θ, let $L_{(1)}(\theta; \mathcal{D}) \geq L_{(2)}(\theta; \mathcal{D}) \geq \cdots \geq L_{(n)}(\theta; \mathcal{D})$ be the decreasing values of the individual losses $L_1(\theta; \mathcal{D}), \ldots, L_n(\theta; \mathcal{D})$, where $(j) \in \{1, \ldots, n\}$ (for all $j \in \{1, \ldots, n\}$). That is, $\{(1), \ldots, (n)\}$ as a perturbation of $\{1, \ldots, n\}$ defines the order of sample indexes by loss values. Whenever we encounter ties on the values, we employ an arbitrary fixed tie-breaking rule in order to ensure the uniqueness of such an order.

Denote $r_i(\theta; \mathcal{D}) = \sum_{j=1}^n \mathbb{1}\{i = (j)\}\gamma_j$ where (j) depends on $(\theta, \mathcal{D})$. Given an arbitrary set $\Theta \subseteq \mathbb{R}^{d_\theta}$, we define $\mathfrak{R}_n(\Theta)$ as the (standard) Rademacher complexity of the set $\{(x, y) \mapsto \ell(f(x; \theta), y) : \theta \in \Theta\}$:

$$\mathfrak{R}_n(\Theta) = \mathbb{E}_{\bar{\mathcal{D}}, \xi} \left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \xi_i \ell(f(\bar{x}_i; \theta), \bar{y}_i) \right]$$

where $\overline{\mathcal{D}} = ((\bar{x}_i, \bar{y}_i))_{i=1}^n$, and $\xi_1, \ldots, \xi_n$ are independent uniform random variables taking values in $\{-1, 1\}$ (i.e., Rademacher variables). Given a tuple $(\ell, f, \Theta, \mathcal{X}, \mathcal{Y})$, define M as the least upper bound on the difference of individual loss values:

$$|\ell(f(x;\theta), y) - \ell(f(x';\theta), y')| \leq M$$

for all $\theta \in \Theta$ and all $(x, y), (x', y') \in \mathcal{X} \times \mathcal{Y}$. For example, $M = 1$ if ℓ is the 0-1 loss function. We can then write:

$$\hat{\mathfrak{R}}_n(\Theta) = \mathbb{E}_\xi \left[\sup_{\theta\in\Theta} \frac{1}{n} \sum_{i=1}^n \xi_i \ell(f(x_i;\theta), y_i)\right].$$

6.2 Preliminaries

The previous paper [22] proves that the stochastic optimization method that uses a gradient estimator that is purposely biased toward those samples with the current top-q losses (i.e., ordered SGD) implicitly minimizes a new objective function of $L_q(\theta; \mathcal{D}) = \frac{1}{q}\sum_{j=1}^n \gamma_j L_{(j)}(\theta; \mathcal{D})$, for any $\mathcal{D}$ (including $g(\mathcal{D})$), in the sense that such a gradient estimator is an unbiased estimator of a (sub-) gradient of $L_q(\theta; \mathcal{D})$, instead of $L(\theta; \mathcal{D})$. Accordingly, the top-q-biased stochastic optimization method converges in terms of L_q instead of L.

Building up on this result, we consider generalization properties of the top-q-biased stochastic optimization with the presence of additional label noises in training data. We want to minimize the expected loss, $\mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f(x;\theta), y)]$, by minimizing the training loss $L_q(\theta; g(\mathcal{D}))$, where $g(\mathcal{D}) = ((g_i^x(x_i), g_i^y(y_i)))_{i=1}^n$ is potentially corrupted by arbitrary noise and corruption effects within arbitrary fixed functions g_i^x and g_i^y for $i = 1 \ldots, n$, where $(x_i, y_i) \sim \mathcal{P}$. Thus, we want to analyze the generalization gap:

$$\mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f(x;\theta), y)] - L_q(\theta; g(\mathcal{D}))$$

[22] showed the benefit of the top-q-biased stochastic optimization method in terms of generalization when g_i^x and g_i^y are identity functions and thus when the distributions are the same for both expected loss and training loss. In contrast, in our setting, the distributions are different for expected loss and training loss with potential noise corruptions through g_i^x and g_i^y.

6.3 Generalization Bound for Biased Query Samples

Theorem 1. *Let Θ be a fixed subset of $\mathbb{R}^{d_\theta}$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an iid draw of n examples $\mathcal{D} = ((x_i, y_i))_{i=1}^n$, the following holds for all $\theta \in \Theta$:*

$$\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)] \leq$$
$$L_q(\theta; g(\mathcal{D})) + 2\hat{\mathfrak{R}}_n(\Theta) + M\left(2 + \frac{s}{q}\right)\sqrt{\frac{\ln(2/\delta)}{2n}} - \mathcal{Q}_{n,q}(\Theta, g)$$

where we define the top-q-biased factor as

$$\mathcal{Q}_{n,q}(\Theta, g) := \mathbb{E}_{\bar{\mathcal{D}}}\Big[\inf_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^n \Big[\frac{r_i(\theta; g(\bar{\mathcal{D}}))n}{q} \cdot$$
$$\ell(f(g_i^x(\bar{x}_i);\theta), g_i^y(\bar{y}_i)) - \ell(f(\bar{x}_i;\theta), \bar{y}_i)\Big]\Big].$$

The expected error $\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)]$ in the left hand side of Theorem 1 is a standard objective for generalization, whereas the right-hand side contains the data corruption function g. Here, we typically have $\mathfrak{R}_n(\Theta) = O(1/\sqrt{n})$ in terms of n. For example, consider the standard feedforward deep neural networks of the form $f(x) = (\omega_T \circ \sigma_{T-1} \circ \omega_{T-1} \circ \sigma_{T-2} \cdots \sigma_1 \circ \omega_1)(x)$ where T is the number of layers, $\omega_l(a) = W_l a$ with $\|W_l\|_F \leq M_l$, and σ_l is an element-wise nonlinear activation function that is 1-Lipschitz and positive homogeneous (e.g., ReLU). Then, if $\|x\| \leq B$ for all $x \in \mathcal{X}$, using Theorem 1 of [19], we have that:

$$\hat{\mathfrak{R}}_n(\Theta) \leq \frac{B(\sqrt{2\log(2)T}+1)(\prod_{l=1}^T M_l)}{\sqrt{n}}$$

In Theorem 1, we can see that a label noise corruption g can lead to the failure of the top-q-biased stochastic optimization via increasing the training loss $L_q(\theta; g(\mathcal{D}))$ and decreasing the top-q-biased factor $\mathcal{Q}_{n,q}(\Theta, g)$. Here, if there is no corruption g (i.e., if g_i^x and g_i^y are identity functions), then we have that $\mathcal{Q}_{n,q}(\Theta, g) \geq 0$ because $\mathcal{Q}_{n,q}(\Theta, g) = \mathbb{E}_{\bar{\mathcal{D}}}[\inf_{\theta\in\Theta} L_q(\theta; \bar{\mathcal{D}}) - L(\theta; \bar{\mathcal{D}})] \geq 0$ due to $L_q(\theta; \bar{\mathcal{D}}) - L(\theta; \bar{\mathcal{D}}) \geq 0$ for any θ and $\bar{\mathcal{D}}$ when g_i^x and g_i^y are identity functions. Thus, the top-q-biased factor $\mathcal{Q}_{n,q}(\Theta, g)$ can explain the improvement of the generalization of the top-q-biased stochastic optimization over the standard unbiased stochastic optimization. However, with the presence of the corruption g, $\frac{r_i(\theta; g(\bar{\mathcal{D}}))n}{q}\ell(f(g_i^x(\bar{x}_i);\theta), g_i^y(\bar{y}_i))$ can be smaller than $\ell(f(\bar{x}_i;\theta), \bar{y}_i)$ *by fitting the corrupted noise*, resulting $\mathcal{Q}_{n,q}(\Theta, g) < 0$. This leads to a significant failure in the following sense:

The generalization gap ($\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)] - L_q(\theta; g(\mathcal{D}))$) goes to zero as n approach infinity if $\mathcal{Q}_{n,q}(\Theta, g) \geq 0$ with no data corruption, but the generalization gap no longer goes to zero as as n approach infinity if $\mathcal{Q}_{n,q}(\Theta, g) < 0$ with data corruption.

To see this, let us look at the asymptotic case when $n \to \infty$. Let Θ be constrained such that $\mathfrak{R}_n(\Theta) \to 0$ as $n \to \infty$, which has been shown to be satisfied for various models and sets Θ, including the standard deep neural networks above [6, 27, 5, 21, 19]. The third term in the right-hand side of the Equation in Theorem 1 disappears as $n \to \infty$. Thus, if there is no corruption (i.e., if g_i^x and g_i^y are identity functions), it holds with high probability that

$$\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)] \leq L_q(\theta; g(\mathcal{D})) - \mathcal{Q}_{n,q}(\Theta, g) \leq L_q(\theta; g(\mathcal{D}))$$

where $L_q(\theta; g(\mathcal{D}))$ is minimized by the top-q-biased stochastic optimization. From this viewpoint, the top-q-biased stochastic optimization minimizes the expected error for generalization when $n \to \infty$, if there is no corruption. However, if there is corruption,

$$\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)] \leq L_q(\theta; g(\mathcal{D})) - \mathcal{Q}_{n,q}(\Theta, g) \not\leq L_q(\theta; g(\mathcal{D}))$$

Hence $\mathbb{E}_{(x,y)}[\ell(f(x;\theta), y)] - L_q(\theta; g(\mathcal{D})) \not\to 0$ even in the asymptotic case. The full proof of Theorem 1 can be found in the Appendix.

7 Conclusion

Neural Active Learning is an active area of research, with many new techniques competing to achieve better results. Our work seeks to challenge the commonly held assumption that the training data is independent and identically distributed (I.I.D) for the active learning setup. We show that the uncertainty-based techniques that are competitive on homogeneous datasets with little label noise can fail catastrophically when presented with diverse heteroskedastic distributions. We also explore the different techniques (diversity-based sampling, fine-tuning, and LHD) that can be used to mitigate these failures. We believe that research has to be done exploring the various possible data distributions, since there is no guarantee that the I.I.D assumption holds for real-world data, and develop algorithms that are robust to the different data distributions.

References

[1] András Antos, Varun Grover, and Csaba Szepesvári, 'Active learning in heteroscedastic noise', *Theoretical Computer Science*, **411**(29), 2712–2728, (2010). Algorithmic Learning Theory (ALT 2008).

[2] David Arthur and Sergei Vassilvitskii, 'K-means++: The advantages of careful seeding', in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, p. 1027–1035, USA, (2007). Society for Industrial and Applied Mathematics.

[3] Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Sham Kakade, 'Gone fishing: Neural active learning with fisher embeddings', *arXiv preprint arXiv:2106.09675*, (2021).

[4] Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal, 'Deep batch active learning by diverse, uncertain gradient lower bounds', *International Conference on Learning Representations*, (2020).

[5] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky, 'Spectrally-normalized margin bounds for neural networks', in *Advances in Neural Information Processing Systems*, pp. 6240–6249, (2017).

[6] Peter L Bartlett and Shahar Mendelson, 'Rademacher and gaussian complexities: Risk bounds and structural results', *Journal of Machine Learning Research*, **3**(Nov), 463–482, (2002).

[7] Zalán Borsos, Marco Tagliasacchi, and Andreas Krause. Semi-supervised batch active learning via bilevel optimization, 2020.

[8] Kaidi Cao, Yining Chen, Junwei Lu, Nikos Arechiga, Adrien Gaidon, and Tengyu Ma, 'Heteroskedastic and imbalanced deep learning with adaptive regularization', *arXiv preprint arXiv:2006.15766*, (2020).

[9] Kamalika Chaudhuri, Prateek Jain, and Nagarajan Natarajan, 'Active heteroscedastic regression', in *International Conference on Machine Learning*, pp. 694–702. PMLR, (2017).

[10] Kamalika Chaudhuri, Sham Kakade, Praneeth Netrapalli, and Sujay Sanghavi, 'Convergence rates of active learning for maximum likelihood estimation', in *Advances in Neural Information Processing Systems*, (2015).

[11] Yuxin Chen and Andreas Krause, 'Near-optimal batch mode active learning and adaptive submodular optimization', in *International Conference on Machine Learning*, (2013).

[12] Sanjoy Dasgupta, 'Two faces of active learning', *Theoretical computer science*, (2011).

[13] Thomas Drugman, Janne Pylkkonen, and Reinhard Kneser, 'Active and semi-supervised learning in asr: Benefits on the acoustic and language models', (2019).

[14] Pan Du, Suyun Zhao, Hui Chen, Shuwen Chai, Hong Chen, and Cuiping Li, 'Contrastive coding for active learning under class distribution mismatch', in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8907–8916, (2021).

[15] Yarin Gal, Riashat Islam, and Zoubin Ghahramani, 'Deep bayesian active learning with image data', in *International Conference on Machine Learning*, (2017).

[16] Mingfei Gao, Zizhao Zhang, Guo Yu, Sercan O. Arik, Larry S. Davis, and Tomas Pfister. Consistency-based semi-supervised active learning: Towards minimizing labeling cost, 2019.

[17] Yonatan Geifman and Ran El-Yaniv, 'Deep active learning over the long tail', *arXiv:1711.00941*, (2017).

[18] Daniel Gissin and Shai Shalev-Shwartz, 'Discriminative active learning', *arXiv:1907.06347*, (2019).

[19] Noah Golowich, Alexander Rakhlin, and Ohad Shamir, 'Size-independent sample complexity of neural networks', in *Conference On Learning Theory*, pp. 297–299. PMLR, (2018).

[20] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning, 2011.

[21] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio, 'Generalization in deep learning', *In Mathematics of Deep Learning, Cambridge University Press, to appear. Prepint available as: MIT-CSAIL-TR-2018-014, Massachusetts Institute of Technology*, (2018).

[22] Kenji Kawaguchi and Haihao Lu, 'Ordered sgd: A new stochastic optimization framework for empirical risk minimization', in *International Conference on Artificial Intelligence and Statistics*, pp. 669–679, (2020).

[23] 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*, (2019).

[24] Suraj Kothawade, Nathan Beck, Krishnateja Killamsetty, and Rishabh Iyer, 'Similar: Submodular information measures based active learning in realistic scenarios', in *Advances in Neural Information Processing Systems*, eds., M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, volume 34, pp. 18685–18697. Curran Associates, Inc., (2021).

[25] Alex Krizhevsky, 'Learning multiple layers of features from tiny images', Technical report, (2009).

[26] Jing Lin, Ryan Luley, and Kaiqi Xiong, 'Active learning under malicious mislabeling and poisoning attacks', *arXiv preprint arXiv:2101.00157*, (2021).

[27] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar, *Foundations of machine learning*, MIT press, 2012.

[28] Yuval Netzer, Tiejie Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng, 'Reading digits in natural images with unsupervised feature learning', (2011).

[29] Phill K. Rhee, Enkhbayar Erdenee, Shin Dong Kyun, Minhaz Uddin Ahmed, and SongGuo Jin, 'Active and semi-supervised learning for object detection with imperfect data', *Cognitive Systems Research*, **45**, 109–123, (2017).

[30] Dan Roth and Kevin Small, 'Margin-based active learning for structured output spaces', in *European Conference on Machine Learning*, (2006).

[31] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang, 'Distributionally robust neural networks', in *International Conference on Learning Representations*, (2019).

[32] Ozan Sener and Silvio Savarese, 'Active learning for convolutional neural networks: A core-set approach', in *International Conference on Learning Representations*, (2018).

[33] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng, 'Meta-weight-net: Learning an explicit mapping for sample weighting', *arXiv preprint arXiv:1902.07379*, (2019).

[34] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning, 2019.

[35] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li, 'Fixmatch: Simplifying semi-supervised learning with consistency and confidence', *Advances in Neural Information Processing Systems*, **33**, 596–608, (2020).

[36] Jacob Steinhardt, Pang Wei Koh, and Percy Liang, 'Certified defenses for data poisoning attacks', in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3520–3532, (2017).

[37] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[38] Gokhan Tur, Dilek Hakkani-Tür, and Robert E Schapire, 'Combining active and semi-supervised learning for spoken language understanding', *Speech Communication*, (2005).

[39] Jose Rodrigo Sanchez Vicarte, Gang Wang, and Christopher W. Fletcher, 'Double-cross attacks: Subverting active learning systems', in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1593–1610. USENIX Association, (August 2021).

[40] Dan Wang and Yi Shang, 'A new active labeling method for deep learning', in *International Joint Conference on Neural Networks*, (2014).

[41] R.Y. Wang, V.C. Storey, and C.P. Firth, 'A framework for analysis of data quality research', *IEEE Transactions on Knowledge and Data Engineering*, **7**(4), 623–640, (1995).

[42] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language model with self generated instructions, 2022.

[43] Zheng Wang and Jieping Ye, 'Querying discriminative and representative samples for batch mode active learning', *Transactions on Knowledge Discovery from Data*, (2015).

[44] Kai Wei, Rishabh Iyer, and Jeff Bilmes, 'Submodularity in data subset selection and active learning', in *International Conference on Machine Learning*, (2015).

[45] Renrui Zhang, Jiaming Han, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, Peng Gao, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention, 2023.

[46] Xiaojin Zhu, John D. Lafferty, and Zoubin Ghahramani, 'Combining active learning and semi-supervised learning using gaussian fields and harmonic functions', in *ICML 2003*, (2003).