Cartography Active Learning

Mike Zhang and **Barbara Plank**
Department of Computer Science
IT University of Copenhagen
{mikz, bapl}@itu.dk

Abstract

We propose Cartography Active Learning (CAL), a novel Active Learning (AL) algorithm that exploits the behavior of the model on individual instances *during training* as a proxy to find the most informative instances for labeling. CAL is inspired by data maps, which were recently proposed to derive insights into dataset quality (Swayamdipta et al., 2020). We compare our method on popular text classification tasks to commonly used AL strategies, which instead rely on post-training behavior. We demonstrate that CAL is competitive to other common AL methods, showing that training dynamics derived from small seed data can be successfully used for AL. We provide insights into our new AL method by analyzing batch-level statistics utilizing the data maps. Our results further show that CAL results in a more data-efficient learning strategy, achieving comparable or better results with considerably less training data.

1 Introduction

Active Learning (AL) is a widely-used method to tackle the time-consuming and expensive collection and manual labeling of data. In recent years, many AL strategies were proposed. The simplest and most widely used is *uncertainty sampling* (Lewis and Gale, 1994; Lewis and Catlett, 1994), where the learner queries instances that it is most uncertain about. Uncertainty sampling is myopic: it only measures the information content of a single data instance. Alternative AL algorithms instead focus on selecting a *diverse* batch (Geifman and El-Yaniv, 2017; Sener and Savarese, 2018; Gissin and Shalev-Shwartz, 2019; Zhdanov, 2019) or to estimate the *uncertainty* distribution of the learner (Houlsby et al., 2011; Gal and Ghahramani, 2016). However, these methods are usually limited in their notion of informativeness, which is tied to post-training model uncertainty and batch diversity.

Recently, Swayamdipta et al. (2020) introduced *data maps*, to visualize the behaviour of the model on individual instances during training (*training dynamics*). The plotted data maps (Figure 1) reveal distinct regions in a dataset: groups of *ambiguous* instances useful for high performance and linked to high informativeness, *easy-to-learn* instances which aid optimization, and *hard-to-learn* instances which frequently correspond to mislabeled or erroneous instances.

We propose *Cartography Active Learning* (CAL), which automatically selects the the most informative instances that contribute optimally to model learning. To do so, we leverage a largely ignored source of information: insights derived *during training*, i.e., training dynamics derived from *limited* data maps (see Section 4) to choose informative instances at the boundary of ambiguous and hard-to-learn instances. We hypothesize that this region is where the model will *learn the most* from. Data maps provide the additional benefit that we can use them to measure informativeness of a batch with straightforward metrics and visualize dataset properties. These distinct regions in the data maps have their own respective statistics. Therefore, as a second research question we investigate whether data map statistics help to assess why some AL algorithms work better than others.

Contributions In this paper, our contributions are twofold. (1) We present Cartography Active Learning, a novel AL algorithm that exploits data maps for AL. We compare our results against other competitive and widely used AL algorithms and outperform them in early AL iterations. (2) Additionally, we leverage the data maps to inspect what instances AL methods select. We show that our approach optimally selects informative instances avoiding only *hard-to-learn* and *easy-to-learn* cases, which leads to better AL and comparable or better results than full dataset training.

2 Related Work

AL has seen many usage scenarios in the Natural Language Processing (NLP) field (Shen et al., 2018; Lowell et al., 2019; Ein-Dor et al., 2020; Margatina et al., 2021). The perspective of AL is that if a model is allowed to select the data from which it will *learn the most*, it will achieve comparable (or better) performance with less training instances (Siddhant and Lipton, 2018), and at the same time addressing the costly labeling process with a human annotator.

A popular scenario is pool-based active learning (Lewis and Gale, 1994; Settles, 2009, 2012), which assumes a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$. Most AL algorithms start similarly: a model is fit to $\mathcal{L}$ to get access to $P_\theta(y \mid \boldsymbol{x})$, then apply a query strategy to get the best scored instance from $\mathcal{U}$, label this instance and add it to $\mathcal{L}$ in an iterative process.

Common Strategies A commonly used query strategy is *uncertainty sampling* (Lewis and Gale, 1994; Lewis and Catlett, 1994). In this approach, the learner queries the instances which it is least certain about. There are two popular approaches. (1) Uncertainty sampling based on entropy (Shannon, 1948; Dagan and Engelson, 1995), it uses the entropy of the label distribution as a measure for the uncertainty of the model on an instance. (2) Uncertainty sampling based on which best labeling is the least confident (Culotta and McCallum, 2005).

Batch-mode Active Learning It is inefficient and time-consuming to obtain sampled queries one by one for annotation in the context of Deep Neural Networks (DNNs). In a real-world setting, consider having multiple annotators available. One can exploit this setting and label the instances in batches and parallel. Batch-mode AL allows the learner to query instances in groups. To assemble the optimal batch, one can greedily pick the top-k examples according to an instance-level acquisition function suitable for DNNs. There are many works on ways for making neural network posteriors accurately represent the confidence on a given example. One popular example is stochastic regularisation techniques such as dropout during inference time, known as the Monte Carlo Dropout technique (Houlsby et al., 2011). Gal and Ghahramani (2016) refer to this as Bayesian Active Learning by Disagreement (BALD). This allows us to consider the model as a Bayesian neural network and calculate approximations of uncertainty estimates by analyzing its multiple predictions. However, if the information of these top-k examples is similar, this will result in the model not generalizing well over the dataset. Therefore, alternative approaches take the *diversity* of a batch into account.

Batch-aware Query Strategies Instead of greedily choosing the examples that maximize some score, one can instead try to find a batch that is as diverse as possible. One recently proposed effective strategy is Discriminative Active Learning (DAL; Gissin and Shalev-Shwartz, 2019). This approach aims to select instances from $\mathcal{U}$ that make $\mathcal{L}$ representative of $\mathcal{U}$. In other words, the idea is to train a separate model to classify between $\mathcal{L}$ and $\mathcal{U}$. Then, to use that model to choose the instances which are most confidently classified as being from $\mathcal{U}$. If $\mathcal{U}$ and $\mathcal{L}$ become indistinguishable, the learner has successfully closed the data gap between $\mathcal{U}$ and $\mathcal{L}$. DAL was proposed for computer vision and was recently successfully used in NLP (Ein-Dor et al., 2020). Alternative diversity AL strategies exist, such as core-set, which often rely on heuristics (Sener and Savarese, 2018; Geifman and El-Yaniv, 2017).[1]

3 Cartography Active Learning

The key idea of CAL is to use model-independent measures, from fitting the model on the seed data $\mathcal{L}$, by using data maps (Swayamdipta et al., 2020) for AL. Data maps help identify characteristics of instances within the broader trends of a dataset by leveraging their training dynamics (i.e., the behavior of a model *during training*, such as mean and standard deviation of confidence and correctness with respect to the gold label). These model-dependent measures reveal distinct regions in a data map, by and large, reflecting instance properties (see Figure 1 and details below on *easy-to-learn*, *ambiguous*, and *hard-to-learn* instances). Training dynamics encapsulate information of data quality that has been largely ignored in AL: the sweet spot of instances at the boundary of *hard-to-learn* and *ambiguous* instances, which are quick to label while providing informative samples, as shown in full data training (Swayamdipta et al., 2020).

[1]Contemporary to our work and both appearing at EMNLP 2021, Margatina et al. (2021) proposed CAL (Contrastive Active Learning), another AL method with the same acronym to our approach. The main difference to Cartography AL is using data points that are similar in the model feature space, while optimizing for maximally disagreeing predictive likelihoods.

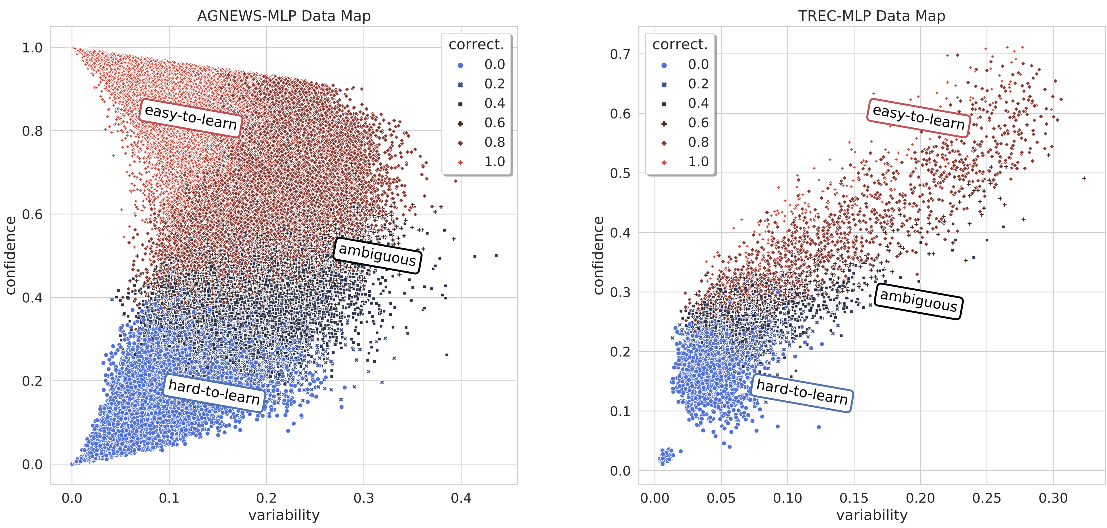

Figure 1: **Full Data Maps for AGNews & TREC.** AGNews (120,000 instances) on the left, and TREC (5,452 instances) on the right, both w.r.t. an MLP training for ten epochs. The x-axis shows **variability** and the y-axis the **confidence**. The colors and shapes indicate the **correctness**.

In the next part, we introduce training characteristics, first showing the resulting data maps on the full data. Then we introduce CAL, which proposes to learn a data map from the seed labeled data $\mathcal{L}$ and identifying regions of instances with a binary classifier, inspired by DAL (Gissin and Shalev-Shwartz, 2019). To identify these regions, we require: (1) a data map can be learned from limited data, and (2) a classifier to identify informative instances. The full algorithm, illustrated in Algorithm 1, is described later.

Mapping the Data Formally, the training dynamics of instance i are defined as the statistics calculated over E epochs. These statistics are then used as the coordinates in the plot. The following statistics are calculated, **confidence**, **variability**, and **correctness**, following the notation of Swayamdipta et al. (2020):

$$\hat{\mu}_i = \frac{1}{E}\sum_{e=1}^{E} p_{\boldsymbol{\theta}^{(e)}}(y_i^* \mid \boldsymbol{x}_i) \tag{1}$$

Confidence[2] (Equation 1) is the mean model probability of the gold label (y_i^*) across epochs. Where $p_{\boldsymbol{\theta}^{(e)}}$ is the model's probability with parameters $\boldsymbol{\theta}^{(e)}$ at the end of the e^{th} epoch.

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{e=1}^{E}\left(p_{\boldsymbol{\theta}^{(e)}}(y_i^* \mid \boldsymbol{x}_i) - \hat{\mu}_i\right)^2}{E}} \tag{2}$$

Then, **variability** (Equation 2) is calculated as the standard deviation of $p_{\boldsymbol{\theta}^{(e)}}(y_i^* \mid \boldsymbol{x}_i)$, the spread across epochs E.

$$\hat{\phi}_i = \frac{1}{E}\sum_{e=1}^{E} \mathbf{1}(\hat{y}_i = y_i^* \mid \boldsymbol{x}_i) \tag{3}$$

Last, **correctness** (Equation 3) is denoted as the fraction of times the model correctly labels instance $\boldsymbol{x}_i$ across epochs E.

Given the aforementioned training dynamics and the obtained statistics per instance, we plot the data maps for both AGNews (Zhang et al., 2015) and TREC (Li and Roth, 2002), using all training data (Figure 1). The data map is based on a Multi-layer Perceptron (MLP). As shown by Swayamdipta et al. (2020), data maps identify three distinct regions: *easy-to-learn*, *ambiguous*, and *hard-to-learn*. The *easy-to-learn* instances are consistently predicted correctly with high confidence, these instances can be found in the upper region of the plot. The *ambiguous* samples have high variability and the model is inconsistent in predicting these correctly (middle region). The instances that are (almost) never predicted correctly, and have low confidence and variability, are referred to as *hard-to-learn* cases. This confirms findings by Swayamdipta

[2]Similar to Swayamdipta et al. (2020), we note that the term *confidence* here is the output probability of the model over the gold label as opposed to the certainty of the predicted label as commonly used in AL literature.

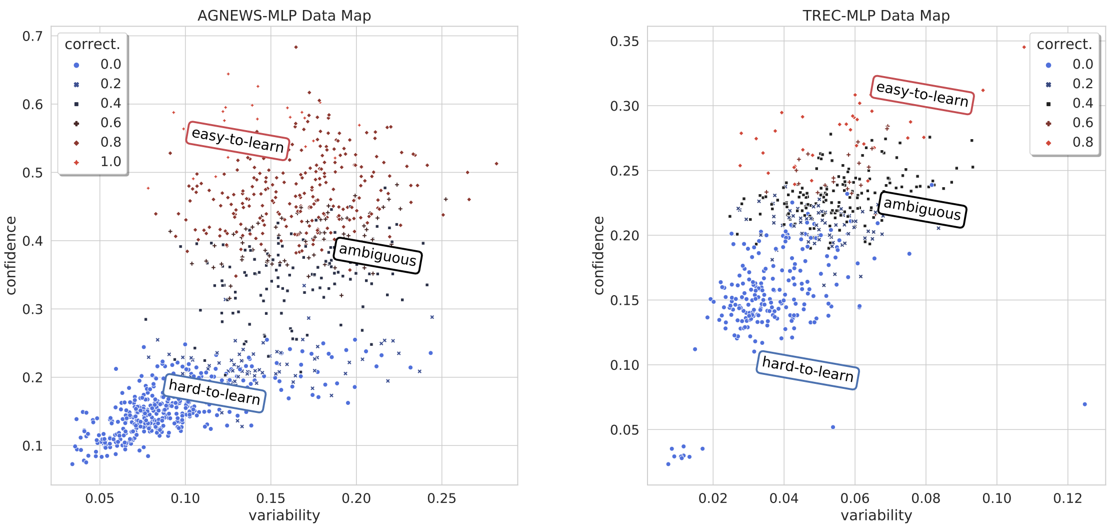

Figure 2: **Data Maps with Limited Seed Data.** Data map for the AGNews seed set (1,000 instances), and TREC seed set (500 instances). Both data maps are based on an MLP trained for ten epochs.

et al. (2020) where they show that training on the samples of these distinct regions, and in particular the *ambiguous* instances, promote optimal performance. While uncertainty-based AL mostly focus on hard-cases, CAL instead focuses on ambiguous and possibly easier instances.

Data Maps from Seed Data Given the distinct regions for data selection in the full data map in Figure 1, we first investigate whether regions are still identifiable if we have little amounts of training data, as this is a prerequisite for CAL. Figure 2 shows this for 1,000 training samples of AGNews and 500 of TREC. We can see that the data points are more scattered, where the *easy-to-learn* and *ambiguous* samples are mixed. However, it seems the *hard-to-learn* region can still qualitatively be distinguished from the other regions.

CAL Algorithm The algorithm is detailed in Algorithm 1 and described next. To select presumably informative instances, we train a binary classifier on the seed set $\mathcal{L}$ and apply it to $\mathcal{U}$ to select the instances that are the closest to the decision boundary between *ambiguous* and *hard-to-learn* instances. By visualizing a decision function in Figure 2 that separates the *hard-to-learn* region from the *ambiguous/easy-to-learn* region, we select the instances that are the closest to this boundary. In other words, selecting instances with output probability 0.5 with respect to the binary classifier. This does two things, (1) it prevents the binary classifier from selecting only *easy-to-learn* instances (low-variability, high-confidence), and (2) selecting some truly *hard-to-learn* instances (low-variability, low-confidence) which are both not optimal for learning.

Similar to DAL, for the binary classification task, we map our original input space $\mathcal{X}$ to the learned representation Ψ of the last hidden layer of an MLP (Section 4.3). These are the features used in our binary classifier (θ'). Formally, as we have three hidden layers,

$$\Psi : \mathcal{X} \rightarrow \hat{\mathcal{X}}, \text{where } \Psi = \mathbf{h}_3 = f(W_3 \cdot \mathbf{h}_2 + \mathbf{b}_3).$$

For label space $\mathcal{Y}$, we consider the binary values $\{0, 1\}$. This label depends on the *correctness*. The label $y_{\Psi(\hat{\boldsymbol{x}}_i)}$ for the learned representation of instance $\hat{\boldsymbol{x}}_i$ is labeled 1 when the *correctness* using the limited data map at epoch E is above the threshold $t_{cor} > 0.2$. We refer to these as **high-cor** cases. The samples that are rarely correct ($t_{cor} \leq 0.2$) are labeled as 0 and we refer to these as **low-cor** cases. To give a better intuition, we refer to Figure 2, where the regions of *hard-to-learn* and *ambiguous/easy-to-learn* are visually separable with this *correctness* threshold $t_{cor} = 0.2$. This threshold is empirically chosen by investigating the influence of different *correctness* thresholds on the performance of CAL in Section 5 (Table 3).

4 Experimental Setup

We focus on pool-based active learning. Once trained on a seed set $\mathcal{L}$, we begin the simulated AL loop by iteratively selecting instances based on

Algorithm 1: Cartography Active Learning

input: Labeled seed set $\mathcal{L}$, Unlabeled set $\mathcal{U}$, Total budget K, Number of queries n, Correctness threshold $t_{cor} = 0.2$;
for *i = 1, ..., n* **do**
  $\Psi(\mathcal{L})$, $\Psi(\mathcal{U}) \leftarrow$ train main classifier θ on $\mathcal{L}$, get representations of $\mathcal{L}$ and $\mathcal{U}$;
  $\hat{\mu}$, $\hat{\sigma}$, $\hat{\phi} \leftarrow$ get data map statistics of $\mathcal{L}$ with θ;
  $P_{\theta'} \leftarrow$ train binary classifier θ' on $\Psi(\mathcal{L})$ with $y_{\Psi(\hat{\boldsymbol{x}}_i)} = \begin{cases} 1, & \text{if } \hat{\phi}_i > t_{cor} \\ 0, & \text{else} \end{cases}$
  for *j=1, ...,* $\frac{K}{n}$ **do**
    $\hat{\boldsymbol{x}} \leftarrow \underset{\boldsymbol{x} \in \Psi(\mathcal{U})}{\operatorname{argmin}} |0.5 - P_{\theta'}(\hat{y} = 1 \mid \boldsymbol{x})|$;
    $\mathcal{L} \leftarrow \mathcal{L} \cup \hat{\boldsymbol{x}}$;
    $\mathcal{U} \leftarrow \mathcal{U} \backslash \hat{\boldsymbol{x}}$;
  end
  reset parameters θ and θ';
  return $\mathcal{L}, \mathcal{U}$
end

the scoring of an acquisition function. We take the top-50 instances, following prior work (Gissin and Shalev-Shwartz, 2019; Ein-Dor et al., 2020). The selected instances are shown the withheld label and added to the labeled set $\mathcal{L}$ and removed from $\mathcal{U}$. We evaluate the performance of the trained model on a predefined held-out test set. We run 30 AL iterations, over five random seeds, and report averages over these runs.

4.1 Datasets

Dataset	Train	Test	Classes	Seed set size
AGNews	120,000	7,600	4	1,000
TREC	5,452	500	6	500

Table 1: **Datasets.** Statistics of the two datasets.

In our AL setup, we consider two popular text classification tasks, namely **AGNews** (Zhang et al., 2015) and **TREC** (Li and Roth, 2002). The AGNews task entails classifying news articles into four classes: world, sports, business, science/technology. For TREC, the task is to categorize questions into one of six categories based on the subject of the question, such as questions about locations, persons, concepts, et cetera. Statistics of the data can be found in Table 1. We start with a seed set size of 1,000 for AGNews and 500 for TREC, both are stratified. This means after the AL iterations we will have 2,500 labeled instances for AGNews and 2,000 for TREC. Our motivation here is to keep the AL simulation realistic. We assume enough annotation budget to initially annotate 500–1,000 samples. Then, in every AL iteration annotate an additional 50 samples, which seems manageable for an annotator. Finally, we run 30 AL iterations to give a good overview of the performance of the acquisition functions over the iterations towards convergence.

4.2 Acquisition Functions

We consider five acquisition functions. We opt for a random sampling baseline (**Rand.**), four existing acquisition functions, and our proposed CAL algorithm. We chose these as they are state-of-the-art and cover a spectrum of acquisition functions (uncertainty, batch-mode and diversity-based).

Least Confidence (LC; Culotta and McCallum, 2005) It takes

$$\underset{\boldsymbol{x} \in \mathcal{U}}{\operatorname{argmax}}\, 1 - P_\theta(\hat{y} \mid \boldsymbol{x})$$

of the predictive (e.g. softmax) distribution as the model's uncertainty, and chooses instances with lowest predicted probability.

Max-Entropy (Ent.; Dagan and Engelson, 1995) Another popular example is entropy based sampling. Instances are selected according to the function

$$\underset{\boldsymbol{x} \in \mathcal{U}}{\operatorname{argmax}} \; - \sum_{y \in \mathcal{Y}} P_\theta(y \mid \boldsymbol{x}) \log_2 \big(P_\theta(y \mid \boldsymbol{x})\big)$$

and again, based on the *a posteriori* probability distribution.

Bayesian Active Learning by Disagreement (BALD; Houlsby et al., 2011; Gal and Ghahramani, 2016) This approach entails applying dropout at test time, then estimating uncertainty as the disagreement between outputs realized via multiple passes through the model. We use the Monte Carlo Dropout technique on ten inference cycles, with the max-entropy acquisition function.

Discriminative Active Learning (DAL; Gissin and Shalev-Shwartz, 2019) This approach poses AL as a binary classification task, it uses a separate binary classifier as a proxy to select instances that make $\mathcal{L}$ representative of the entire dataset (i.e., making the labeled set indistinguishable from the unlabeled pool set). The input space for the binary classifier is task-agnostic. One maps the original input space $\mathcal{X}$ to a learned representation $\hat{\mathcal{X}}$ as the input space, with label space $\mathcal{Y} = \{l, u\}$ referring to labeled and unlabeled. In the original paper, the learned representation is defined as the logits of the last hidden layer of the main classifier which solves the original task. Formally, it selects the top-k instances that satisfy

$$\underset{\boldsymbol{x} \in \mathcal{U}}{\operatorname{argmax}} \hat{P}_\theta(\hat{y} = u \mid \Psi(\boldsymbol{x}))$$

where $\hat{P}_\theta$ is the trained binary classifier given the learned representations Ψ of instances $\boldsymbol{x}$.

4.3 Configurations

This work uses two models for the AL setup solving the classification tasks. All the code is open source and available to reproduce our results.[3]

Main Classifier We use a Multi-layer Perceptron (MLP), with three $d_{\text{emb}} = 300$ ReLu layers, dropout probability $p = 0.3$, minimize cross-entropy, AdamW optimizer (Loshchilov and Hutter, 2017), with a learning rate of $1\text{e}{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1\text{e}{-8}$.

Binary Classifier This model is suited for the binary classification task of DAL and CAL. In this case it is a single $d_{\text{emb}} = 300$ ReLu layer. We minimize the cross-entropy as well. We use the AdamW optimizer with a learning rate of $5\text{e}{-5}$. For more details regarding reproducibility, we refer to Section 7.

Training the Binary Classifier For both the binary classification task of DAL and CAL, we empirically determined that setting the number of epochs for the binary classifier to 30 yielded good results. In the context of DAL, for AGNews it reaches around 98% accuracy during training, and for TREC it achieves around 89% accuracy. In contrast, with CAL, we start with little amounts of data for a binary classifier to train on. In the early stage of the AL iterations, the binary classifier does not achieve a high accuracy for both AGNews and TREC (around random). After it reaches the fifth or sixth AL iteration it starts to properly distinguish the **low-cor/hig-cor** samples, as it probably has enough samples to learn from. It achieves around 65–75% accuracy for AGNews, and towards 85% accuracy for TREC. The classification accuracy on AGNews seems low. However, further tuning of the binary classifier (e.g., increasing the number of epochs) slightly increases binary classification accuracy, but did not result in better performance for the overall AL setup.

Significance Recently, the Almost Stochastic Order test (ASO; Dror et al., 2019)[4] has been proposed to test statistical significance for DNNs over multiple runs. Generally, the ASO test determines whether a stochastic order (Reimers and Gurevych, 2018) exists between two models or algorithms based on their respective sets of evaluation scores. Given the single model scores over multiple random seeds of two algorithms A and B, the method computes a test-specific value (ϵ_{min}) that indicates how far algorithm A is from being significantly better than algorithm B. When distance $\epsilon_{min} = 0.0$, one can claim that A stochastically dominant over B with a predefined significance level. When $\epsilon_{min} < 0.5$ one can say $A \succeq B$. On the contrary, when we have $\epsilon_{min} = 1.0$, this means $B \succeq A$. For $\epsilon_{min} = 0.5$, no order can be determined. We took 0.05 for the predefined significance level α and measure it across all 30 AL iterations.

5 Results & Analysis

We plot the accuracy of the AL algorithms (Section 4.2) on each dataset in Figure 3. For AGNews and TREC, all AL strategies except DAL outperform the random baseline. CAL is statistically dominant over BALD (AGNews) and DAL (AGNews, TREC), and competitive with LC and Entropy (Table 2). This shows that CAL reaches strong results. CAL (illustrated as `cartography` in the figure with a red-dotted line).

Why does CAL work? To gain insight on why CAL works better, we investigate the average statistics of each selected batch of samples using the data maps. In Figure 4, we check the mean confidence, variability and correctness over each selected batch of 50 for sampling strategies Random, LC, DAL, and CAL for both AGNews and TREC. The statistics of the instances are extracted after the selected top–50 batch is added to the seed

[3] https://github.com/jjzha/cal

[4] Implementation of Dror et al. (2019) can be found at https://github.com/Kaleidophon/deep-significance (Ulmer, 2021)

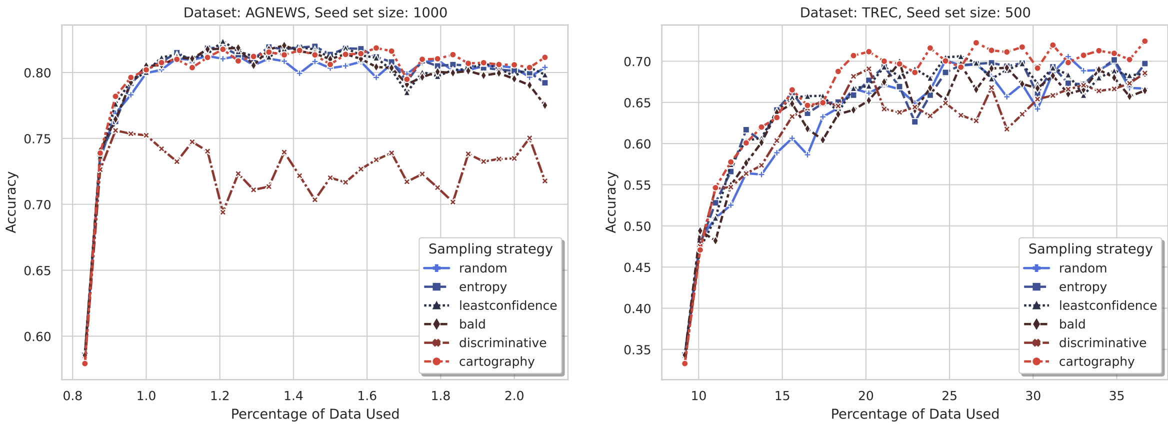

Figure 3: **Performance AL strategies.** Performance of the various AL strategies in terms of accuracy. The accuracy shown over the AL iterations is the average over five random seeds. Note that for both datasets we added the same number of instances to the seed set (+1,500 instances). The x-axis correspond to the fraction of the total size of the respective dataset.

	Rand.	LC	Ent.	BALD	DAL	CAL
Rand.		1.00	1.00	1.00	*0.01*	1.00
LC	**0.00**		1.00	1.00	*0.01*	1.00
Ent.	**0.00**	**0.00**		1.00	*0.01*	1.00
BALD	**0.00**	**0.00**	0.00		*0.01*	1.00
DAL	0.99	0.99	0.99	0.99		1.00
CAL	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	

	Rand.	LC	Ent.	BALD	DAL	CAL
Rand.		1.00	1.00	1.00	1.00	1.00
LC	**0.00**		1.00	0.78	0.57	1.00
Ent.	**0.00**	**0.00**		0.87	0.60	1.00
BALD	**0.00**	*0.22*	*0.13*		1.00	1.00
DAL	**0.00**	*0.43*	*0.40*	1.00		1.00
CAL	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	

Table 2: **Almost Stochastic Order Scores of AGNews (left) & TREC (right).** ASO scores expressed in ϵ_{min} The significance level $\alpha = 0.05$ is adjusted accordingly by using the Bonferroni correction (Bonferroni, 1936). **Bold** numbers indicate stochastic dominance and *cursive* means that one algorithm is better than the other, e.g., for AGNews, LC (row) is stochastically dominant over the random baseline (column) with ϵ_{min} value of 0.00). Numbers are obtained across all 30 AL iterations

set. Once the main model is trained again on the increased seed set, we obtain the statistics of the previously added batch of 50. Figure 4 shows that `leastconfidence` and `cartography` has the higher variability amongst the other methods (middle graph). There is a similar signal as the findings of Swayamdipta et al. (2020), the *ambiguous* samples that the model *learns the most* from are usually the instances that have the highest variability and average confidence—as we can see in Figure 4 (especially pronounced on AGNews), these are the instances CAL selects. In contrast, LC selects instances that have relatively low confidence and low variability in the early stages, but catches up in later ones. This suggests that LC chooses only *hard-to-learn* instances at the start. In general, CAL seems to select more *informative* samples as opposed to the random strategy.

DAL seems to start by choosing *ambiguous* samples with high variability. However, it quickly picks mostly **high-cor** samples in later iterations, in contrast to CAL. Consequently, the performance for DAL drops as it leads to picking the *easy-to-learn* samples. Picking too many *easy-to-learn* instances results in worse optimization (Swayamdipta et al., 2020). The drop for DAL is visible in Figure 3, which shows that CAL and DAL are close at start, and the accuracy of DAL then drops.

t_{cor}	0.0	$\leq$ **0.2**	≤ 0.4	≤ 0.6	≤ 0.8
AGNews	0.789	0.811	0.796	0.788	0.781
TREC	0.676	0.724	0.701	0.706	–

Table 3: **Influence of the Correctness Threshold.** Influence of the correctness threshold t_{cor} on the final average accuracy score per dataset over the five random seeds. t_{cor} considers from what boundary we should consider **low-cor** cases. In **bold** is what threshold we are using.

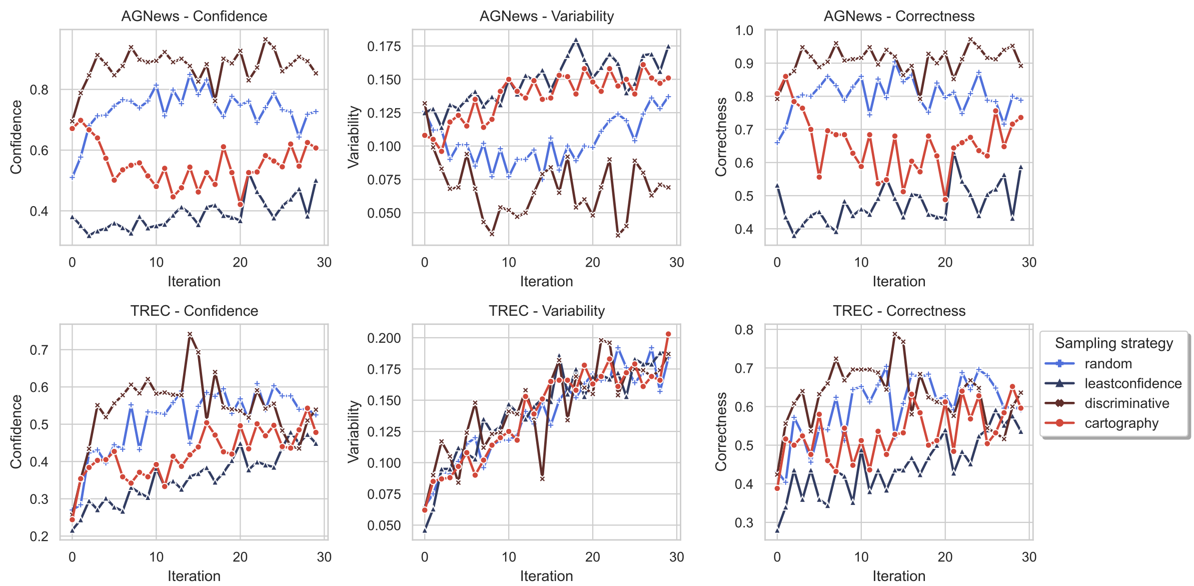

Figure 4: **Average Statistics AGNews & TREC.** Values correspond to the mean statistics (Section 3) of instances over the five random seeds. We calculate the statistics of each top-50 batch *after* being added to the seed set. Therefore, we only have statistics of 29 runs, as in the 30th run we stop the AL cycle.

What is the influence of the *correctness* threshold? Here we investigate how changing the correctness threshold for the binary classification task impacts accuracy. As shown in Table 3, the selected threshold worked well for the two datasets studied. In particular, the accuracy does not drop substantially on AGNews if we move the *correctness* threshold to a higher value. Swayamdipta et al. (2020) indicates that the *ambiguous* region contains the instances with the highest variability. For AGNews, these instances have a *correctness* range from 0.2–0.8 (as seen in Figure 1). Therefore, we assume that most of these instances are informative for the model. In the case of TREC, the final accuracy also stays similar with different *correctness* thresholds and does not drop substantially. We find in the full data map for TREC that the area with $t_{cor} = \{0.0, 0.4\}$ thresholds is more dense compared to $t_{cor} = 0.2$ (details in Section 8). In other words, there are more samples, but also ones that are detrimental to performance. We note that the *correctness* threshold could vary for different datasets and models.

Do AL strategies select the same instances for labeling? We measured the overlap between the batches selected by each pair of strategies (Random, LC, DAL, CAL) on AGNews and TREC (Table 4). The batch overlap for CAL is low, with the highest overlap being 7 instances for AGNews with CAL ∩ Rand. followed by CAL ∩ DAL with 4 overlapping instances. The highest overlap for TREC is 114 instances for CAL ∩ DAL. Note this is the total overlap over five seeds. These results indicate that the AL algorithms choose different instances.

	AGNews	TREC
CAL ∩ DAL	4	114
CAL ∩ LC	2	101
CAL ∩ Rand.	7	82
DAL ∩ LC	3	81
DAL ∩ Rand.	2	87
LC ∩ Rand.	2	100

Table 4: **Number of Overlapping Instances on AGNews & TREC.** The values corresponds to the total overlapping instances out of 1,500 $*$ 5 random seeds $=$ 7,500.

A popular approach for improving classification performance is combining (complementary) AL strategies. For example, Zhdanov (2019) proposed the idea to combine uncertainty sampling and diversity sampling for image classification. As DAL and CAL have few overlapping instances, they can be complementary to each other. To test this, we combined DAL and CAL using a simple heuristic, by providing both of them half of the annotation budget (i.e., take top-25 batch of each AL strategy). This resulted in an accuracy score of 0.695

for TREC and 0.789 for AGNews. This suggests that it could have a positive effect if there is a more sophisticated approach. This is an open research topic that requires further investigation.

How data-efficient is CAL in comparison to full data training? If a model is able to choose the instances that it can learn the most from, it can reach comparable results or even outperform a model trained on all data by using fewer training instances. This is noted by Siddhant and Lipton (2018), where they achieve 98–99% of the full dataset performance while labeling only 20% of the samples. The overall accuracy for AGNews trained on all data is 0.820 accuracy on test. For TREC, this results in 0.634. With CAL, we achieve around 99% of the full dataset performance while using only 2% training data for AGNews. For TREC, we outperform the full dataset performance by 0.090 accuracy (0.634 vs. 0.724), the full dataset performance is already reached by using around 14% training data. This is appealing, as active learning can provide more data-effective learning solutions.

6 Conclusion

In this paper, we introduced a new AL algorithm, Cartography Active Learning. The AL objective is transformed into a binary classification task (Gissin and Shalev-Shwartz, 2019), where we optimize for selecting the most informative data with respect to a model by leveraging insights from data maps (Swayamdipta et al., 2020). Data maps help to identify distinct regions in a dataset based on training dynamics (*hard-to-learn* and *easy-to-learn/ambiguous* instances), which have shown to play an important role in model optimization and stability in full dataset training (Swayamdipta et al., 2020). We use these insights in low-data regimes and propose CAL. Specifically for the task of Visual Question Answering, contemporary work by Karamcheti et al. (2021) use data maps for analysis into AL acquisition functions and the effect of outliers. In contrast, this work instead applies data maps directly for AL by training a discriminator on limited seed data maps to select the most informative instances. We show empirically that our method is competitive or significantly outperforms various popular AL methods, and provide intuitions on why this is the case by using training dynamics.

Acknowledgements

We thank the NLPnorth group for feedback on an earlier version of this paper — in particular Elisa Bassignana and Max Müller-Eberstein for insightful discussions. We would also like to thank the anonymous reviewers for their comments to improve this paper. Last, we also thank NVIDIA and the ITU High-performance Computing cluster for computing resources. This research is supported by the Independent Research Fund Denmark (DFF) grant 9131-00019B.

References

Carmen Banea, Di Chen, Rada Mihalcea, Claire Cardie, and Janyce Wiebe. 2014. SimCompass: Using deep learning word embeddings to assess cross-level similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 560–565, Dublin, Ireland. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Carlo Bonferroni. 1936. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *AAAI*, volume 5, pages 746–751.

Ido Dagan and Sean P Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157. Elsevier.

Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. Deep dominance - how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, Florence, Italy. Association for Computational Linguistics.

Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active Learning for BERT: An Empirical Study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7949–7962, Online. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33nd International Conference on Machine Learning,* *ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1050–1059. JMLR.org.

Yonatan Geifman and Ran El-Yaniv. 2017. Deep active learning over the long tail. *arXiv preprint arXiv:1711.00941*.

Daniel Gissin and Shai Shalev-Shwartz. 2019. Discriminative active learning. *arXiv preprint arXiv:1907.06347*.

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

Siddharth Karamcheti, Ranjay Krishna, Li Fei-Fei, and Christopher Manning. 2021. Mind your outliers! investigating the negative impact of outliers on active learning for visual question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7265–7281, Online. Association for Computational Linguistics.

David D Lewis and Jason Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. 2019. Practical obstacles to deploying active learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China. Association for Computational Linguistics.

Katerina Margatina, Giorgos Vernikos, Loïc Barrault, and Nikolaos Aletras. 2021. Active learning by acquiring contrastive examples. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 650–663.

Nils Reimers and Iryna Gurevych. 2018. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Burr Settles. 2009. Active learning literature survey.

Burr Settles. 2012. Active learning. *Synthesis lectures on artificial intelligence and machine learning*, 6(1):1–114.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Yanyao Shen, Hyokun Yun, Zachary C. Lipton, Yakov Kronrod, and Animashree Anandkumar. 2018. Deep active learning for named entity recognition. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Aditya Siddhant and Zachary C. Lipton. 2018. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2904–2909, Brussels, Belgium. Association for Computational Linguistics.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Dennis Ulmer. 2021. deep-significance: Easy and Better Significance Testing for Deep Neural Networks. Https://github.com/Kaleidophon/deep-significance.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Fedor Zhdanov. 2019. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*.

Appendix

7 Reproducibility

We initialize the main model with English $d_{\text{emb}} = 300$ FastText embeddings (Bojanowski et al., 2017) and keep it frozen during training and inference, we sum the embeddings over the sequence of tokens as motivated by Banea et al. (2014). For AGNews we impose a maximum sequence length of 200 and a batch size of 64. For TREC, a maximum sentence length of 42 and a batch size of 16. We run both models for 30 epochs, with no early stopping. Per AL iteration, we do a weight reset on all models. We average our results over five randomly generated seeds (398048, 127003, 259479, 869323, 570852). All experiments were ran on an NVIDIA® A100 SXM4 40 GB GPU and an AMD EPYC™ 7662 64-Core CPU. Specifically for CAL, a single AL batch (50) selection iteration takes 11 seconds on average assuming TREC. For AGNews, one AL iteration takes 62 seconds on average. Both runtimes are with respect to the models depicted in Section 4.3 and hardware mentioned above.

8 Full Data Map

Figure 5 and Figure 6 show the full data maps for AGNews and TREC respectively. Identically to Swayamdipta et al. (2020), we show the density of data points in the plots. We can see a clear difference in density between the datasets. For AGNews, we can see the majority of data points have a high confidence ($\sim$0.8) and high correctness. In contrast, TREC contains plenty of instances that have low confidence ($\sim$0.3) and low correctness.

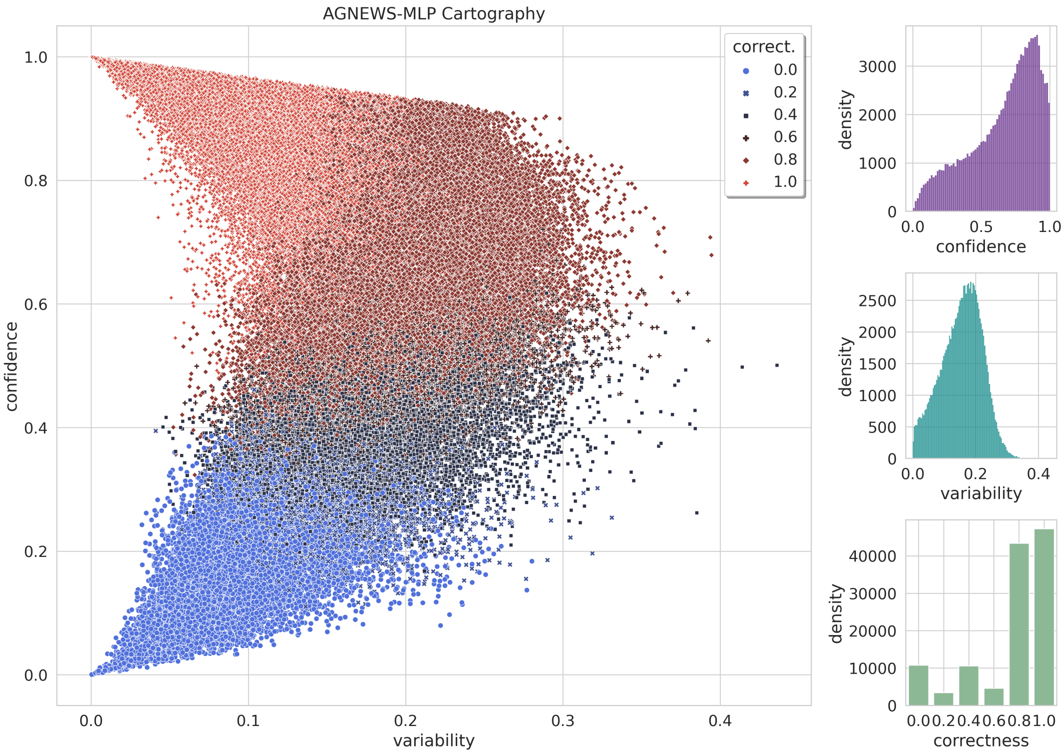

Figure 5: **Density of AGNews.** Density statistics of AGNews over ten epochs.

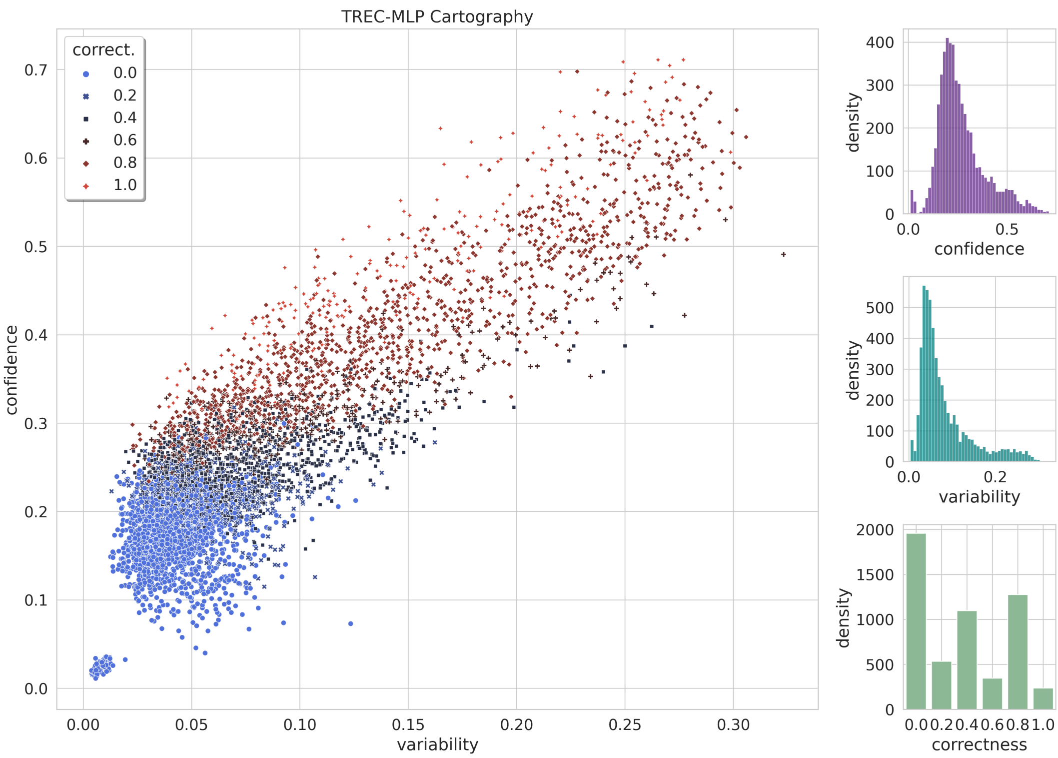

Figure 6: **Density of TREC.** Density statistics of TREC over ten epochs.