Efficient Failure Pattern Identification of Predictive Algorithms

Bao Nguyen[1,2] **Viet Anh Nguyen**[3]

[1]School of Information and Communication Technology, Hanoi University of Science and Technology, Vietnam
[2]College of Engineering & Computer Science, VinUni-Illinois Smart Health Center, VinUniversity, Vietnam
[3]The Chinese University of Hong Kong

Abstract

Given a (machine learning) classifier and a collection of unlabeled data, how can we efficiently identify misclassification patterns presented in this dataset? To address this problem, we propose a human-machine collaborative framework that consists of a team of human annotators and a sequential recommendation algorithm. The recommendation algorithm is conceptualized as a stochastic sampler that, in each round, queries the annotators a subset of samples for their true labels and obtains the feedback information on whether the samples are misclassified. The sampling mechanism needs to balance between discovering new patterns of misclassification (exploration) and confirming the potential patterns of classification (exploitation). We construct a determinantal point process, whose intensity balances the exploration-exploitation trade-off through the weighted update of the posterior at each round to form the generator of the stochastic sampler. The numerical results empirically demonstrate the competitive performance of our framework on multiple datasets at various signal-to-noise ratios.

1 INTRODUCTION

Over the past few years, algorithmic predictive models have claimed many successful stories in real-world applications, ranging from healthcare and finance to jurisdiction and autonomous driving. These successes often take place in an invariant environment where the training data and the test data come from sufficiently similar distributions. If this similarity condition does not hold, then it is well-known that the performance of the algorithmic prediction can deteriorate significantly in the deployment phase. This performance deterioration may trigger subsequent concerns, especially in consequential domains such as self-driving cars and healthcare, where the algorithmic predictions may affect system reliability and human safety. When a predictive model performs unsatisfactorily in a *systematic* manner, then it is called a failure pattern. For example, if an object detection system fails to capture the object systematically under low-light conditions, then it is a failure pattern. Detecting and correcting the failure patterns is arguably one of the most daunting challenges in developing the future analytics systems.

Detecting failure patterns is also beneficial for many steps in the life-cycle of an analytics product. For example, it is very common to develop analytics systems using data collected from one country, say the United States, but the systems can be deployed in another country, say Canada. Before field deployment, the systems need to undergo intensive fine-tuning and recalibration, and information regarding the failures can significantly reduce the time and efforts spent on this step. Further, as foundation models are increasingly used as a building block of future analytics systems, assessing the blindspots of these pre-trained models is of crucial importance for product development, deployment and continuous performance evaluation.

Detecting failure patterns, unfortunately, requires the true outcome or label of the data samples. If the dataset is cleanly labeled, then the problem of failure pattern detection can be simplified to a search problem. The situation becomes more cumbersome if we have access to only an *un*labeled dataset. It is thus natural herein to incorporate the role of a human annotator into the detection procedure. However, in many applications that require high-skilled annotators, such as healthcare, it is time-consuming and cost-intensive to query the annotator. Given a dataset containing *un*labeled samples, we are interested in designing an efficient routine to query these samples for their true outcome or label, so that we can identify as many failure patterns as possible with minimal annotation queries.

Contributions. We propose in this paper a directed sam-

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

pling mechanism with the goal of detecting failure patterns from an unlabeled dataset. This mechanism has two main components:

- a Gaussian process to model the predictive belief about the misclassification probability for each unlabeled sample,
- a determinantal point process sampling that balances the trade-off between exploration and exploitation by taking a mixture between a similarity matrix (reflecting exploration) and a posterior probability of misclassification matrix (reflection exploitation).

Ultimately, we propose a human-machine collaborative framework that relies on the machine's proposed queries and the corresponding annotator's feedback to detect failure patterns, and consequently improve the reliability of the predictive algorithm in variant environments.

This paper unfolds as follows: In Section 2, we survey existing efforts to identify failure patterns in a given dataset. Section 3 formalizes our problem and introduces the necessary notations. Section 4 depicts our approach using Gaussian processes to build the surrogate for the Value-of-Interest, which represents the belief about the misclassification probability for the unlabeled samples. Section 5 focuses on the determinantal point process sampler, which blends the Value-of-Interest (highlighting exploitation) with a diversity term (highlighting exploration). Section 6 discusses how we can choose the bandwidth, which is critical given the unsupervised nature of the failure identification problem. Finally, Section 7 provides extensive numerical results to demonstrate the superior performance of our directed sampling mechanism in detecting failure patterns for various datasets.

In almost all related work about failure identification, a failure mode (termed as "slice" in Eyuboglu et al. [2022]) of a pre-trained classifier on a dataset is a subset of the dataset that meets two conditions: (i) the classifier performs poorly when predicting samples in that subset, and (ii) the subset captures a distinct concept or attribute that would be recognizable to domain experts. If the true labels of all samples in that subset are available, criterion (i) can be easily confirmed using popular performance metrics for classification tasks. However, condition (ii) is subjective and implicit. For instance, two medical experts may interpret a group of misclassified brain scan images in two different ways, and both interpretations may be reasonable and acceptable. For unlabeled data, it is difficult to employ the existing definitions of the failure patterns.

Arguably, a failure mode is a subjective term that depends on the machine learning task, on the users, and on the datasets. We do not aim to provide a normative answer to the definition of a failure mode in this paper. Our paper takes a pragmatic approach: given a choice of failure pattern definition of users, we develop a reasonable method that can efficiently find the failure patterns from the unlabeled data.

Notations. We use $\mathbb{R}^d$ to denote the space of d-dimensional vectors, and $\mathbb{S}_+^d$ to denote the set of d-by-d symmetric, positive semidefinite matrices. The transposition of a matrix A is denoted by $A^\top$, and the Frobenius norm of a matrix A is denoted by $\|A\|_F$.

2 RELATED WORK

Detecting failure patterns of a machine learning algorithm on a given dataset is an emergent problem in the literature. d'Eon et al. [2022] formulates a single-stage optimization problem to identify the systematic error. Sohoni et al. [2020] mitigates hidden stratification by detecting sub-class labels before solving a group distributionally robust optimization problem(GDRO). Eyuboglu et al. [2022] provides an evaluation method for slice discovery methods (SDMs): given a trained classifier and a labeled dataset of the form $(x_i, y_i)_{i=1}^N$, it outputs a set of slicing functions that partitions the dataset into overlapping subgroups on which the model underperforms. Nushi et al. [2018] proposes a hybrid approach to analyze the failures of AI systems in a more accountable and transparent way. The hybrid approach involves two main phases: (1) machine analysis to identify potential failure modes, and (2) human analysis to validate and explain the identified failure modes. Singla et al. [2021] concentrates on understanding the failures of deep neural networks. The paper proposes a method for robust feature extraction, which aims to identify the most important features that contribute to the output of a deep neural network. Polyzotis et al. [2019] proposed SliceFinder that automatically identifies subsets of the data that cause the model to fail or perform poorly. It repeatedly draws random subsets of the data and re-trains the model on each subset, then measures the model's performance. Finally, it employs a statistical test to identify data slices that are significantly different from the overall distribution. Sagadeeva and Boehm [2021] build a method using linear algebra techniques to speed up the slice-finding process for identifying problematic subsets of data that cause a model to fail.

Our paper is also closely related to the field of active learning. Active learning examines how to obtain the largest amount of performance gains by labeling as few samples as possible. Comprehensive surveys on active learning can be found in Ren et al. [2021] and Budd et al. [2021]. There is, however, a critical difference between the settings of our paper and those of active learning: in our paper, we focus on identifying the failure pattern for a fixed (invariant) classifier, whereas active learning focuses on improving the model performance with model retraining after each batch of recommendations. The numerical experiments in Section 7 demonstrate empirically that active learning algorithms do not perform competitively at the failure identification task.

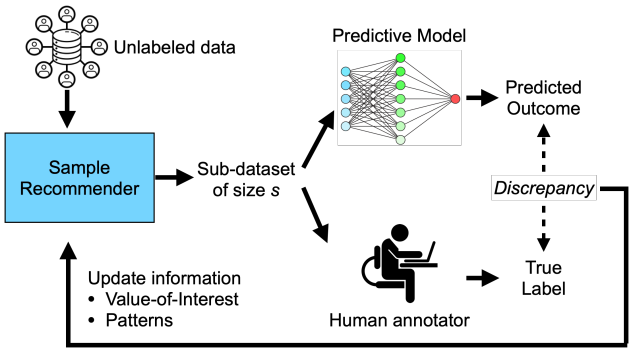

Figure 1: Schematic view of our solution: We build a sampling mechanism to recommend the next s unlabeled samples to be labeled by the annotators. The misclassification information is fed back to update the sampling density. Throughout this sequential recommendation process, the predictive model does not change.

3 PROBLEM STATEMENT

We suppose that a user is given a classification algorithm and a set of unlabeled dataset. The classifier is denoted by $\mathcal{C} : \mathcal{X} \to \mathcal{Y}$, which admits a feature space $\mathcal{X} = \mathbb{R}^d$ as input, and outputs labels in the set $\mathcal{Y} = \{1, \ldots, C\}$. The unlabeled dataset consists of N samples, which can be represented by N vectors $x_i \in \mathbb{R}^d$ for $i = 1, \ldots, N$. For each sample x_i, the predicted label (termed the pseudolabel) that is recommended by the algorithm is denoted by $\widehat{y}_i = \mathcal{C}(x_i)$; while its true label is denoted by y_i^{true}. The i-th sample in the dataset is accurately classified if $\widehat{y}_i = y_i^{\text{true}}$, and it is misclassified if $\widehat{y}_i \neq y_i^{\text{true}}$.

The main object of this paper is *not* the individual misclassified samples. Our goal in this paper is to study the group effect of misclassified samples: when they are sufficiently close to each other, they form a failure pattern or a failure cluster, and together, they signal a systematic error of the predictive algorithm on the given dataset. To give a formal definition of a failure pattern, we need to construct a graph representation of the data. To do this, we suppose the available data can form an abstract undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes whereas each node represents one sample from the dataset, and $\mathcal{E}$ is the set of edges. Let $\mathcal{G}^{\text{mis}} = (\mathcal{V}^{\text{mis}}, \mathcal{E}^{\text{mis}})$ be the pruned graph of $\mathcal{G}$ after removing the nodes representing samples that are classified accurately. We can now rigorously define a general failure pattern.

Definition 3.1 (M-C failure pattern)**.** Given a graph $\mathcal{G}^{\text{mis}}$, an integer size threshold M, and a connectivity criterion C, a failure pattern is a subgraph $\mathcal{G}^{\text{fail}} = (\mathcal{V}^{\text{fail}}, \mathcal{E}^{\text{fail}})$ of $\mathcal{G}^{\text{mis}}$ that satisfies the connectivity criterion C and the cardinality of the set $\mathcal{V}^{\text{fail}}$ is at least M.

Our framework gives the user tremendous flexibility in specifying the problem and leveraging our tool to identify the related failure patterns:

- First, the user can specify the semantic similarity between two nodes represented by an edge connecting them. Moreover, this user-based graph construction can capture the inherently subjective characteristics of the definition of failure, which depends on the user's perspective and the domain of data. For a concrete example, if the user is confident that the embedding space is rich enough to capture the similarity between samples, then one possible approach is to construct $\mathcal{G}$ by the mutually nearest neighbor graph under the Euclidean distance between embedding representations.
- Secondly, users can specify preferred connectivity criteria among samples in a failure pattern. For instance, the criterion of "completeness" can be employed as a candidate for C. This criterion is one of the most stringent: it mandates that each misclassified node must exhibit an edge (indicating semantic similarity) with every other misclassified node in the pattern. However, users can adjust the stringency by employing looser criteria, such as "Super-κ connectivity" or "connected connectivity".
- Third, the users can specify M which depicts the amount of evidence required in order to confirm a failure pattern. A large value of M means a high level of evidence required (represented by a high number of clustered misclassified samples) to pinpoint a pattern. Moreover, one can also see that given a fixed graph of misclassified samples $\mathcal{G}^{\text{mis}}$, the number of failure patterns in $\mathcal{G}^{\text{mis}}$ is non-increasing in M. Taking this perspective, one can also view M as a parameter that is *inverse*-proportional to the user's degree of risk aversion: a low value of M indicates that the user believes there are many misclassified patterns in the dataset.

While identifying the failure patterns from an unlabeled dataset is becoming increasingly pertinent in the current practice of machine learning deployment, this problem is inherently challenging due to two main factors:

1. Annotation cost: to determine if a data point is misclassified, we need to have complete information about its *true* label, which is often obtained by querying a team of human annotators. Unfortunately, in many consequential domains such as healthcare and law enforcement, the cost of annotation can be enormous for a large dataset.
2. Signal-to-noise ratio: suppose that there are F failure patterns represented in the data graph $\mathcal{G}^{\text{mis}}$ and they are denoted by subgraphs $\mathcal{G}_f^{\text{fail}} = (\mathcal{V}_f^{\text{fail}}, \mathcal{E}_f^{\text{fail}})$ for $f = 1, \ldots, F$. The union $\cup_f \mathcal{V}_f^{\text{fail}}$ gives a collection of all misclassifed samples that belong to the failure patterns, and the remainder set $\mathcal{V}^{\text{mis}} \backslash \cup_f \mathcal{V}_f^{\text{fail}}$ contains all samples that are misclassified but do not belong to any failure pattern. Because the goal of the user is to identify failure

patterns, the user can view the cardinality of the union set $\cup_f \mathcal{V}_f^{\text{fail}}$ as a measure of the amount of signal in the dataset, and the cardinality of the remainder set $\mathcal{V}^{\text{mis}} \backslash \cup_f \mathcal{V}_f^{\text{fail}}$ as a measure of the amount of noise therein. As such, we observe a typical signal-to-noise ratio (SNR) indication: if the SNR is high, then the problem tends to be easy; on the contrary, if the SNR is low then the problem tends to be hard.

Unfortunately, crucial information to identify the failure patterns is not known to the user ex-ante: for example, the user does not know how many patterns there are in the dataset, nor does the user know the number of misclassified samples and the SNR. To proceed, we make the assumptions:

Assumption 3.2. We assume the followings:

- the number of unlabeled samples N in the given dataset is large and the cost of using an annotator is expensive, thus it may be prohibitive to annotate the whole dataset,
- the inference cost of the predictive algorithm is negligible for any feature $x \in \mathcal{X}$, which means we can obtain the pseudolabels $\widehat{y}_i = \mathcal{C}(x_i)$ for all data points,
- the predictive algorithm $\mathcal{C}$ remains invariant throughout the failure identification process, and thus the failure patterns do not change over time.

The first assumption is critical because if the cost of annotating the whole dataset is small, then the user does not need to use our proposed sampling mechanism because the benefit of annotating the whole dataset outweighs the resulting cost. The second assumption is reasonable in most practical settings as it requires feed-forwarding the entire given dataset through the predictive algorithm once. The last assumption ensures that our targets, the failure patterns, do not alter over time and that the collected information and the updated belief remain relevant for recommending the next batch of samples for annotation.

We are interested in a dynamic (sequential) recommendation setting that consists of T rounds. In each round $t = 1, \ldots, T$, the system recommends to the user a set of s unlabeled samples to query the annotator for the true label. After the true labels are obtained, the user can recognize which newly-annotated samples are misclassified, and the user, based on this arrival of information, can (i) identify if a new failure pattern can be identified by verifying the conditions in Definition 3.1, and then (ii) update the user's own belief about where the remaining failure pattern may locate. The posterior belief is internalized to the recommendation mechanism to suggest another set of s unlabeled samples from the remaining pool to the next round.

We note that task (i) described above is purely algorithmic: after the arrival of the misclassification information, the user can run an algorithm to search for the newly-formed maximally connected subgraphs of misclassified samples with size at least M. Task (ii), on the contrary, is more intricate because it requires a dynamic update of belief. To achieve task (ii), we will employ a Bayesian approach with Gaussian processes to form a belief about the locations of the failure patterns, and this belief will also be used for the sampling mechanism.

Remark 3.3 (Semantics of the failure pattern). Herein, the failure patterns are defined using the closeness in the feature of the samples. This is a convenient abstraction to alleviate the dependence of the definition on specific tasks (such as image, text, or speech classification). Defining patterns based on the feature space is also a common practice in failure identification, see Eyuboglu et al. [2022] and d'Eon et al. [2022].

4 GAUSSIAN PROCESS FOR THE VALUE-OF-INTEREST

In this section, we describe our construction of a surrogate function called the Value-of-Interest (VoI). Mathematically, we define $\text{VoI} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ to quantify the degree of interest assigned to each pair of feature-pseudolabel $(x_i, \widehat{y}_i)$ data point. The notion of VoI aims to capture the exploitation procedure: it emphasizes recommending samples with a high tendency (or intensity) to confirm a misclassification pattern. Thinking in this way, we aim to predict the probability that the feature-pseudolabel pair will be part of a yet-to-confirmed failure pattern. For any generic sample $(x, \hat{y})$, we model VoI using a sigmoid function of the form

$$\text{VoI}(x, \hat{y}) = \frac{1}{1 + \exp(-g(x, \hat{y}))},$$

for some function $g : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. While VoI is bounded between 0 and 1, the function g can be unbounded, and it is more suitable to model our belief about g using a Gaussian process (GP). In particular, we let $g \sim \text{GP}(m,\ \mathcal{K})$, where m is the mean function and $\mathcal{K}$ is the kernel or covariance function, both are defined on $\mathcal{X} \times \mathcal{Y}$. A GP enables us to model the neighborhood influence among different samples through the covariance function $\mathcal{K}$.

4.1 POSTERIOR UPDATE OF THE PREDICTIVE PROBABILITY

Using the Gaussian process model, we have

$$\tilde{g} = \begin{bmatrix} g(x_1, \hat{y}_1) \\ \vdots \\ g(x_N, \hat{y}_N) \end{bmatrix} \sim \mathcal{N}(0, K),$$

where for a slight abuse of notation, m is a vector of mean values and the covariance matrix $K \in \mathbb{S}_+^N$ is the Gram matrix with the components

$$K_{ij} = \mathcal{K}((x_i, \hat{y}_i), (x_j, \hat{y}_j)) \quad \forall (i, j). \tag{1}$$

We can re-arrange g into another vector $(\tilde{g}_{[t]}, \tilde{g}^*_{[t]})$ where the subvector $\tilde{g}_{[t]} = (g(x_i, \hat{y}_i))_{i \in \mathcal{I}_{[t]}}$ represents the value of $\tilde{g}$ at all samples that are queried by time t, while $\tilde{g}^*_{[t]} = (g(x_i, \hat{y}_i))_{i \in \mathcal{I}^*_{[t]}}$ represents the value of $\tilde{g}$ at all samples that are *not* queried yet by time t. By a similar decomposition of the matrix K, we can rewrite

$$\begin{bmatrix} \tilde{g}_{[t]} \\ \tilde{g}^*_{[t]} \end{bmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m_{[t]} \\ m^*_{[t]} \end{pmatrix}, \begin{bmatrix} K_{[t]} & K^*_{[t]} \\ (K^*_{[t]})^\top & K^{**}_{[t]} \end{bmatrix} \right).$$

By the law of conditional distribution for joint Gaussian distributions [Murphy, 2012, §15.2.1], we have the distribution of $\tilde{g}^*_{[t]}$ conditional on $\tilde{g}_{[t]} = g^{\text{observe}}_{[t]}$ is also a Gaussian distribution:

$$\tilde{g}^*_{[t]} | g^{\text{observe}}_{[t]} \sim \mathcal{N}(m^*_t, \Sigma^*_t),$$

where the conditional mean is determined by

$$m^*_t = m^*_{[t]} + (K^*_{[t]})^\top K^{-1}_{[t]} (g^{\text{observe}}_{[t]} - m_{[t]}) \tag{2a}$$

and the conditional variance is computed as

$$\Sigma^*_t = K^{**}_{[t]} - (K^*_{[t]})^\top K^{-1}_{[t]} K^*_{[t]}. \tag{2b}$$

The vector m^*_t captures the posterior mean of the misclassification probability of samples that are not yet queried by time t.

We now discuss how to estimate the expected value of $\text{VoI}(x_i, \hat{y}_i)$ for the *un*queried sample i. Note that $\text{VoI}(x_i, \hat{y}_i)$ is a nonlinear function of $g(x_i, \hat{y}_i)$, we can use a second-order Taylor expansion of VoI around the conditional mean of g, and we obtain[1]

$$\mathbb{E}[\text{VoI}(x_i, \hat{y}_i) | g^{\text{observe}}_{[t]}] \approx \alpha_i + \frac{1}{2} \Sigma^*_{t,i} \beta_i \triangleq \gamma_{t,i}, \tag{3}$$

with α_i and β_i being computed as

$$\alpha_i = (1 + \exp(-m^*_{t,i}))^{-1}, \quad \beta_i = \alpha_i(1 - \alpha_i)(1 - 2\alpha_i).$$

In the above formulas, $m^*_{t,i}$ and $\Sigma^*_{t,i}$ are the mean and the variance component of the vector m^*_t and matrix Σ^*_t corresponding to sample i. A disadvantage of the approximation (3) is that the value $\gamma_{t,i}$ may become negative due to the possible negative value of β_i. If this happens, we can resort to the first-order approximation:

$$\mathbb{E}[\text{VoI}(x_i, \hat{y}_i) | g^{\text{observe}}_{[t]}] \approx \alpha_i,$$

which guarantees positivity due to the definition of α_i.

4.2 COVARIANCE SPECIFICATION.

Given any two samples $(x, \widehat{y})$ and $(x', \widehat{y}')$, the covariance between $g(x, \widehat{y})$ and $g(x', \widehat{y}')$ is dictated by

$$\text{Cov}(g(x, \widehat{y}), g(x', \widehat{y}')) = \mathcal{K}((x, \widehat{y}), (x', \widehat{y}')).$$

[1] See Appendix B.2 for detailed derivation.

The bivariate function $\mathcal{K}$ is constructed using a kernel on the feature-pseudolabel space $\mathcal{X} \times \mathcal{Y}$. We impose a product kernel on $\mathcal{X} \times \mathcal{Y}$ of the form

$$\mathcal{K}((x, \widehat{y}), (x', \widehat{y}')) = \mathcal{K}_{\mathcal{X}}(x, x') \mathcal{K}_{\mathcal{Y}}(\widehat{y}, \widehat{y}'). \tag{4}$$

In order to express a covariance function, we construct $\mathcal{K}$ as a positive definite kernel.

Definition 4.1 (Positive definite (pd) kernel)**.** Let $\mathcal{Z}$ be any set. A symmetric function $\mathcal{K}_{\mathcal{Z}} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is positive definite if for any natural number n and any choices of $(z_i)_{i=1}^n \in \mathcal{Z}$ and $(\alpha_i)_{i=1}^n \in \mathbb{R}$, we have

$$\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \mathcal{K}_{\mathcal{Z}}(z_i, z_j) \geq 0.$$

If $\mathcal{K}_{\mathcal{X}}$ and $\mathcal{K}_{\mathcal{Y}}$ are positive definite kernels, then $\mathcal{K}$ is also a positive definite kernel according to Schur product theorem [Schur, 1911, Theorem VII]. Thus, it now suffices to construct individual pd kernel for $\mathcal{K}_{\mathcal{X}}$ and $\mathcal{K}_{\mathcal{Y}}$. We choose $\mathcal{K}_{\mathcal{X}}$ as the Gaussian kernel

$$\mathcal{K}_{\mathcal{X}}(x, x') = \exp\big(-\frac{1}{2h^2_{\mathcal{X}}} \|x - x'\|^2_2\big), \tag{5}$$

where $h_{\mathcal{X}} > 0$ is the kernel width.

The main difficulty encountered when specifying the kernel is the categorical nature of the pseudolabel. Imposing a kernel on $\mathcal{Y}$ hence may require a significant effort to pin down a similarity value between any pair of labels. To alleviate this difficulty, we first represent each label y by the respective conditional first- and second-moments, in which the moments are estimated using the samples and their pseudolabels. More specifically, we collect all samples whose pseudolabel is y, and we estimate the feature mean vector and the feature covariance matrix as follows

$$\widehat{\mu}_y = \frac{1}{N_y} \sum_{i: \hat{y}_i = y} x_i \in \mathbb{R}^d, \quad \text{and}$$
$$\widehat{\Sigma}_y = \frac{1}{N_y} \sum_{i: \hat{y}_i = y} (x_i - \widehat{\mu}_y)(x_i - \widehat{\mu}_y)^\top \in \mathbb{S}^d_+,$$

where N_y is the number of samples with pseudolabel y. We now anchor the kernel on $\mathcal{Y}$ through the Gaussian kernel on the product space $\mathbb{R}^d \times \mathbb{S}^d_+$ of the mean-covariance representation, that is, for any generic label y and y':

$$\mathcal{K}_{\mathcal{Y}}(y, y') = \exp\big(-\frac{\|\widehat{\mu}_y - \widehat{\mu}_{y'}\|^2_2}{2h^2_{\mathcal{Y}}}\big) \exp\big(-\frac{\|\widehat{\Sigma}_y - \widehat{\Sigma}_{y'}\|^2_F}{2h^2_{\mathcal{Y}}}\big). \tag{6}$$

Notice that this kernel is specified by a single bandwidth parameter $h_{\mathcal{Y}} > 0$. By combining Jayasumana et al. [2013, Theorems 4.3 and 4.4] and by the fact that the product of pd kernels is again a pd kernel, we conclude that the kernel $\mathcal{K}$ defined using the formulas (4)-(6) is a pd kernel.

Remark 4.2 (Feature-label metric). Measuring the distance between two (categorical) classes by measuring the distance between the corresponding class-conditional distributions was previously used in Alvarez-Melis and Fusi [2020] and Hua et al. [2023]. Under the Gaussian assumptions therein, the 2-Wasserstein distance between conditional distributions simplifies to an explicit formula involving a Euclidean distance between their mean vectors and a Bures distance between their covariance matrices. Unfortunately, constructing a Gaussian kernel with the Bures distance on the space of symmetric positive semidefinite matrices may not lead to a pd kernel. To ensure that $\mathcal{K}_{\mathcal{Y}}$ is a pd kernel, we need to use the Frobenius norm for the covariance part of (6).

Remark 4.3 (Intuition on using pseudolabel). Exploiting the pseudolabel serves the following intuition: Suppose that an input x_i is misclassified. If there exists another input x_i' which is close to x_i, and its pseudolabel $\hat{y}_i'$ is also close to $\hat{y}_i$, then it is also likely that x_i' will also be misclassified. Thus, our construction of the VoI implicitly relies on the assumption that the pseudolabel is also informative to help predict failure patterns. If a user finds this assumption impractical, it is straightforward to remove this information from the specification of the kernel, and the construction of the Gaussian process still holds with minimal changes.

Remark 4.4 (Dimensionality reduction). If the feature space is high-dimensional (d is large), we can apply a dimensionality reduction mapping $\varphi(x_i)$ to map the features to a space of smaller dimensions $d' \ll d$. The kernel $\mathcal{K}_{\mathcal{Y}}$ will be computed similarly with $\widehat{\mu}_y$ and $\widehat{\Sigma}_y$ are all d' dimensional.

4.3 VALUE ADJUSTMENT AND PATTERN NEIGHBORHOOD UPDATE

At time t, the algorithm recommends querying the true label of the samples whose indices are in the set $\mathcal{I}_t$. If for $i \in \mathcal{I}_t$, the sample x_i is misclassified, that is $\hat{y}_i \neq y_i^{\text{true}}$, then we should update the observed probability to $\text{VoI}(x_i, \hat{y}_i) = 1$. However, this would lead to updating $g(x_i, \hat{y}_i) = +\infty$. Similarly, if x_i is classified correctly then we should update $g(x_i, \hat{y}_i) = -\infty$. To alleviate the infinity issues, we set a finite bound $-\infty < L < 0 < U < +\infty$, and we update using the rule

$$g^{\text{observe}}(x_i, \hat{y}_i) = \begin{cases} U & \text{if } x_i \text{ is misclassified,} \\ L & \text{otherwise.} \end{cases}$$

Moreover, suppose that at time t, the annotator identifies that some samples form a pattern. In that case, we intentionally re-calibrate all the values for the samples in the pattern using

$$g^{\text{observe}}(x_i, \hat{y}_i) = L \quad \text{if } i \text{ belongs to a failure pattern.}$$

In doing so, we intentionally adjust the misclassified sample i to be a correctly classified (or 'good') sample.

Remark 4.5 (Interpretation of VoI updates). The VoI aims at capturing the probability that an *un*labeled sample belongs to an *un*identified failure mode. There are three exemplary cases to guide the design of the VoI: (i) if an unlabeled sample is near the misclassified samples and those misclassified samples do not form a failure mode yet, then VoI should be high. This signifies exploitation for confirmation: it is better to concentrate on the region surrounding this unlabeled sample to confirm this failure mode; (ii) if an unlabeled sample is near the correctly-classified samples, then VoI should be low. This is an exploration process: this sample may be in the correctly-classified region, and we should query in other regions; (iii) if an unlabeled sample is near the misclassified samples and those misclassified samples already formed a failure mode, then VoI should be low. Again, this depicts an exploration process: it is better to search elsewhere for other failure modes.

5 DETERMINANT POINT PROCESS SAMPLER FOR ANNOTATION RECOMMENDATION

Determinantal Point Processes (DDPs) is a family of stochastic models that originates from quantum physics: DPPs are particularly useful to model the repulsive behavior of Fermion particles [Macchi, 1975]. Recently, DPPs have gained attraction in the machine learning community [Kulesza and Taskar, 2012, Affandi et al., 2014, Urschel et al., 2017] thanks to its successes in recommendation systems [Chen et al., 2018, Wilhelm et al., 2018, Gartrell et al., 2017] and text and video summarization [Lin and Bilmes, 2012, Cho et al., 2019, Gong et al., 2014], to name a few. Given N samples, the corresponding DPP can be formalized as follows.

Definition 5.1 (L-ensemble DPP)**.** Given a matrix $L \in \mathbb{S}_+^N$, an L-ensemble DPP is a distribution over all 2^N index subsets $J \subseteq \{1, \ldots, N\}$ such that

$$\text{Prob}(J) = \det(L_J)/\det(I + L),$$

where L_J denotes the $|J|$-by-$|J|$ submatrix of L with rows and columns indexed by J.

In this paper, we construct a DPP to help select the next batch of samples to query the annotator for their true labels. We design the matrix L that can balance between exploration (querying distant samples in order to identify *new* potential failure patterns) and exploitation (querying neighborhood samples to confirm plausible failure patterns). Given N samples, the exploration is determined by a similarity matrix $S \in \mathbb{S}_+^N$ that captures pairwise similarity of the samples

$$S_{ij} = \kappa(x_i, x_j) \quad \forall (i, j)$$

for some similarity metric κ. For example, we can use $\kappa \equiv \mathcal{K}_{\mathcal{X}}$, where $\mathcal{K}_{\mathcal{X}}$ is prescribed in (5) with the same bandwidth parameter $h_{\mathcal{X}}$.

Exploration. At time t, conditioned on the samples that are already queried $\mathcal{I}_{[t]}$, we can recover a conditional similarity matrix S_t^* for the set of *un*queried samples following [Borodin and Rains, 2005, Proposition 1.2]. Let $|\mathcal{I}_{[t]}|$ be the number of samples that have been drawn so far, then S_t^* is a $(N - |\mathcal{I}_{[t]}|)$-dimensional positive (semi)definite matrix, calculated as

$$S_t^* = ([(S + I_{\mathcal{I}_{[t]}^*})^{-1}]_{\mathcal{I}_{[t]}^*})^{-1} - I.$$

Thus, the matrix S_t^* will serve as a diversity-promoting term of the conditional DPP at time t.

Exploitation. The exploitation is determined by a probability matrix $P \in \mathbb{S}_+^{N-|\mathcal{I}_{[t]}|}$. At time t, we can use $P_t^* = \mathrm{diag}(\gamma_{t,i})$, where $\gamma_{t,i}$ is the posterior probability of being misclassified defined in (3). Notice that if $\gamma_{t,i}$ is negative, we can replace $\gamma_{t,i}$ by the first-order approximation to guarantee that P_t^* is a diagonal matrix with strictly positive diagonal elements. The matrix P_t^* will induce exploitation because it promotes choosing samples with a high posterior probability of misclassification with the goal of confirming patterns.

Exploration-Exploitation Balancing Conditional DPP. We impose an additional parameter $\vartheta \in [0, 1]$ to capture the exploration-exploitation trade-off. At time t, we use a DPP with the kernel matrix L^ϑ defined as

$$L_t^\vartheta = \vartheta S_t^* + (1 - \vartheta) P_t^*$$

for some mixture weight $\vartheta \in [0, 1]$. In particular, when ϑ is equal to zero, the algorithm's approach is entirely exploitative, and its primary objective is to confirm failure patterns in the dataset. Conversely, when ϑ is equal to one, the algorithm is entirely explorative, and its main aim is to recommend a diverse set of samples from the dataset. It is worth noting that the algorithm's behavior can be adjusted by modifying the value of ϑ to achieve an appropriate trade-off between exploration and exploitation depending on the specific problem at hand. Because both S_t^* and P_t^* are positive semidefinite matrices, the weighted matrix L_t^ϑ is also positive semidefinite, and specifies a valid DPP.

Query Suggestion. We choose a set of unlabeled samples for annotation using a maximum a posteriori (MAP) estimate of the DPP specified by L_t^ϑ. We then find the s samples from the unlabeled data by solving the following problem

$$\max \left\{ \det(L_z) \ : \ z \in \{0, 1\}^{N-|\mathcal{I}_{[t]}|}, \ \|z\|_0 = s \right\}, \quad (7)$$

where L_z is a submatrix of $L_t^\vartheta \in \mathbb{S}_+^{N-|\mathcal{I}_{[t]}|}$ restricted to rows and columns indexed by the one-components of z. It is well-known that the solution to problem (7) coincides with the MAP estimate of the DPP with a cardinality constraint [Kulesza and Taskar, 2012].

Unfortunately, problem (7) is NP-hard [Kulesza and Taskar, 2012]. We thus use heuristics to find a good solution to the problem in a high-dimensional setting with a low running time. A common greedy algorithm to solve the MAP estimation problem is to iteratively find an index that maximizes the marginal gain to the incumbent set of chosen samples z. We then add the index j to the set of samples until reaching the cardinality constraint of s prototypes. This greedy construction algorithm has a complexity cost of $\mathcal{O}(s^2 N)$ time for each inference. An implementation of this algorithm is provided in Chen et al. [2018]. The greedy algorithm has been shown to achieve an approximation ratio of $\mathcal{O}(\frac{1}{s!})$[Civril and Magdon-Ismail, 2009]. Finally, to boost the solution quality, we add a 2-neighborhood local search that swaps one element from the incumbent set with one element from the complementary set. This local search is performed until no further improvement is found.

6 BANDWIDTH SELECTION

The product kernel $\mathcal{K}$ defined in (4) on the feature-pseudolabel label space requires the specification of two hyper-parameters: the bandwidth for the feature $h_\mathcal{X}$ and the bandwidth for the pseudolabels $h_\mathcal{Y}$. Given N feature-pseudolabel pairs $(x_i, \hat{y}_i)$ for $i = 1, \ldots, N$ and the kernel $\mathcal{K}$ defined as in Section 4, we denote the Gram matrix by $K \in \mathbb{S}_+^N$ with the components of K satisfying (1). If the bandwidth parameters are too small compared to the feature distance $\|x - x'\|_2$ and the pseudolabel distance $\sqrt{\|\widehat{\mu}_y - \widehat{\mu}_{y'}\|_2^2 + \|\widehat{\Sigma}_y - \widehat{\Sigma}_{y'}\|_F^2}$ in the dataset, then the matrix K tends toward an N-by-N identity matrix I_N. Notice that when K is an identity matrix, the matrix multiplication $(K^*)^\top K^{-1}$ turns into a matrix of zeros, and the updates (2) become $m_t^* = m_{[t]}^*$ and $\Sigma_t^* = K_{[t]}^{**}$. This means that all observed information from previous queries is ignored. To alleviate this, we impose a restriction on $h_\mathcal{X}$ and $h_\mathcal{Y}$ so that

$$\|K(h_\mathcal{X}, h_\mathcal{Y}) - I_N\|_F \geq \delta \|I_N\|_F \quad (8)$$

for some value of $\delta > 0$. In the above equation, we make explicit the dependence of the Gram matrix K on the hyper-parameters $h_\mathcal{X}$ and $h_\mathcal{Y}$, and the norm $\|\cdot\|_F$ is the Frobenius norm. Condition (8) imposes that the Gram matrix needs to be sufficiently different from the identity matrix, where the magnitude of the difference is controlled by δ. The next proposition provides the condition to choose $h_\mathcal{X}$ and $h_\mathcal{Y}$ to satisfy this condition.

Proposition 6.1 (Hyper-parameter values)**.** *For a fixed value of* $\delta \in (0, \sqrt{N-1})$*, the condition* (8) *is satisfied if*

$$\frac{D_\mathcal{X}}{h_\mathcal{X}^2} + \frac{D_\mathcal{Y}}{h_\mathcal{Y}^2} \leq \ln \frac{N-1}{\delta^2},$$

where $\mathcal{D}_{\mathcal{X}}$ and $\mathcal{D}_{\mathcal{Y}}$ are calculated based on the dataset as

$$D_{\mathcal{X}} = \frac{\sum_{i>j} \|x_i - x_j\|_2^2}{\binom{N}{2}}, \quad \textit{and}$$
$$D_{\mathcal{Y}} = \frac{\sum_{i>j} \|\widehat{\mu}_{\hat{y}_i} - \widehat{\mu}_{\hat{y}_j}\|_2^2 + \|\widehat{\Sigma}_{\hat{y}_i} - \widehat{\Sigma}_{\hat{y}_j}\|_F^2}{\binom{N}{2}}.$$

We suggest choosing $h_{\mathcal{X}}$ and $h_{\mathcal{X}}$ to equalize the components

$$\frac{D_{\mathcal{X}}}{h_{\mathcal{X}}^2} = \frac{D_{\mathcal{Y}}}{h_{\mathcal{Y}}^2} = \frac{1}{2} \ln \frac{N-1}{\delta^2}.$$

We also notice that the value of $\delta = \sqrt{2} \times 10^{-6}$ is reasonable for most practical cases encountered in the numerical experiments of this paper. Hence, without other mention stated, we set δ to $\sqrt{2} \times 10^{-6}$.

7 NUMERICAL EXPERIMENTS

Datasets. For the numerical experiments, we utilize 15 real-world datasets adapted from Eyuboglu et al. [2022][2]. Each dataset in Eyuboglu et al. [2022] consists of a pre-trained classifier and a collection of data points. Each data point has three features: Activation (a 512-dimensional embedding features of the image with respect to the pre-trained classifier), True Label (an integer in the range $\{0, \ldots, C\}$ that represents the true class of the data point), Probs (a C-dimensional vector that indicates the probability of each class). By taking the argmax of the Probs vector for each data point, we can determine the predicted label (pseudolabel) for that data point.

We use the following construction of a failure pattern: Two samples are connected by an edge if each sample is in the k_{nn}-nearest neighbors of the other. Because $\mathcal{X}$ is a feature space, we measure the distance between two samples by taking the Euclidean distance between x_i and x_j. Notice that k_{nn} is a parameter that is chosen by the user. Criterion C in Definition 3.1 is chosen as maximally connected subgraphs. Further discussion about this specific selection of the user is provided in Appendix A.1. Indicating the failure mode as above requires the user to input two hyper-parameters, k_{nn} and M. The discussion about choosing values of k_{nn} and M in practical problems is in Appendix A.3.

For each dataset, we construct a dataset that is suited for the task of failure identification as follows: we choose different values of k_{nn} to construct the graph (cf. Section 3) and the evidence threshold M, then we generate the ground truth information about the true failure patterns by finding maximally connected components in $\mathcal{G}^{\text{mis}}$. Each sample in the dataset is now augmented to have four features: Activation, True Label, Pseudo Label, and Pattern, where Pattern is an integer in the range $\{-1, 1, \ldots, P\}$, where -1 means that the sample does not belong to any failure pattern, and P is the number of failure patterns in the dataset.

During the experiment, the true labels and patterns of samples are hidden: true labels are only accessible by querying the annotator, while pattern information is used to compute the performance ex-post. Our 15 generated datasets are classified into three classes based on the level of SNR: Low, Medium, and High. The details are in Appendix A.2.

Comparison. We compare the following baselines:

- Active learning algorithms: We consider two popular active learning methods, namely BADGE [Ash et al., 2019] and Coreset [Sener and Savarese, 2017]. Because the classifier is fixed in our setting, the retraining stage in these active learning algorithms is omitted.
- Uniform Sampling (US) algorithm: At iteration t with the set of remaining unlabeled samples $\mathcal{I}^*_{[t]}$, we pick a size s subset of $\mathcal{I}^*_{[t]}$ with equal probability. This algorithm is a stochastic algorithm; hence we take results from 30 random seed numbers and calculate the average.
- Five variants of our Directed Sampling (DS_ϑ) algorithm, with ϑ chosen from $\{0, 0.25, 0.5, 0.75, 1\}$. At $\vartheta = 0$, our algorithm is purely exploitative, emphasizing the confirmation of failure patterns. At $\vartheta = 1$, our algorithm is purely exploration, emphasizing recommending diverse samples from the dataset.

Throughout, the batch size is set to $s = 25$ for all competing methods. Codes and numerical results are available at `https://github.com/nguyenngocbaocmt02/FPD`.

Experiment 1 (Sensitivity). The goal of this experiment is to measure the sensitivity of different recommendation algorithms. Hereby, sensitivity is defined as the fraction between the number of queried samples until the detection of the first failure pattern in the dataset and the total number of samples. This value measures how slowly we identify the first failure pattern: a lower sensitivity is more desirable.

We observed that the two active learning algorithms have the lowest performance. We suspect that the objective of active learning algorithms is to refine the decision boundaries to obtain better performance (accuracy), whereas the primary concern herein is to isolate misclassified clusters. Thus, active learning methods may not be applicable to the problem considered in this paper.

We could see from Table 1 that the sensitivity of all methods decreases as the SNR increases. All DS variants except the extreme with $\vartheta = 1.0$ outperform the US method and active learning methods. The poor performance of DS_1.0 is attributed to the lack of an exploitative term, which is also the knowledge gathered from previous queries. Moreover, we notice that our proposed algorithm, DS_0.25 and DS_0

[2]Datasets are publicly available at `https://dcbench.readthedocs.io/en/latest`

Table 1: Benchmark of Sensitivity on different noise magnitudes. Bolds indicate the best methods.

	Noise Magnitude			
Methods	Low	Medium	High	Overall
US	0.48±0.13	0.29±0.07	0.22±0.07	0.33±0.15
DS_0	0.22±0.27	**0.04±0.02**	**0.03±0.01**	**0.11±0.18**
DS_0.25	**0.16±0.08**	0.09±0.04	0.08±0.04	**0.11±0.07**
DS_0.5	0.27±0.20	0.15±0.16	0.07±0.04	0.16±0.17
DS_0.75	0.27±0.10	0.17±0.11	0.08±0.06	0.17±0.12
DS_1.0	0.50±0.10	0.35±0.12	0.22±0.07	0.36±0.15
BADGE	0.56±0.10	0.35±0.07	0.27±0.06	0.40±0.15
Coreset	0.62±0.08	0.44±0.07	0.27±0.08	0.44±0.16

have the smallest sensitivity of 0.11 overall. While DS_0.25 achieves the highest performance in datasets with low noise magnitude, DS_0 is more effective in medium and high SNR contexts. This can be attributed to the fact that when the SNR is low, there are many noise samples. Consequently, if DS_0 gets trapped in a noisy misclassified region, it may take considerable time to confirm whether this area contains any patterns because the algorithm is purely exploitative when $\vartheta = 0$. In contrast, DS_0.25 overcomes this issue by incorporating an exploration term that avoids focusing too much on any area.

Experiment 2 (Effectiveness). This experiment aims to confirm the ability to recommend methods in detecting failure patterns subject to a limited number of annotation queries. More specifically, we allow the number of queries to be maximally 10% and 20% of the total number of samples in the dataset, and we measure effectiveness as the percentage of detected patterns up to that cut-off. A higher value of effectiveness indicates a higher capacity for failure pattern identification.

When the maximum permitted number of queries is low (e.g., 10%), there is no significant difference in the overall performance of all algorithms because the queried information about the dataset is insufficient to confirm most patterns, see Table 2. However, all versions of DS perform equally well and are more effective than the US, BADGE, and Coreset. As the number of queries increases to 20% of the dataset in Table 3, all DS variants significantly outperform US and active learning methods: the DS methods manage to detect more than a third of all failure patterns. In high SNR datasets, DS_0.5 can even detect over half of the patterns on average.

Conclusions. We proposed a sampling mechanism for the purpose of failure pattern identification. Given a classifier and a set of unlabeled data, the method sequentially suggests a batch of samples for annotation and then consolidates the information to detect the failure patterns. The sampling mechanism needs to balance two competing criteria: exploration (querying diverse samples to identify new potential failure patterns) and exploitation (querying neighborhood samples to collect evidence to confirm failure patterns). We constructed a Gaussian process to model the exploitative evolution of our belief about the failure patterns and used a DPP with a weighted matrix to balance the exploration-exploitation trade-off. The numerical experiments demonstrate that our sampling mechanisms outperform the uniform sampling method in both sensitivity and effectiveness measures.

Table 2: Benchmark of Effectiveness (at 10% of sample size) on different noise magnitudes. Bolds indicate the best methods.

	Noise Magnitude			
Methods	Low	Medium	High	Overall
US	0.00±0.00	0.00±0.01	0.02±0.08	0.01±0.05
DS_0.0	**0.10±0.09**	**0.24±0.06**	**0.38±0.10**	**0.24±0.14**
DS_0.25	**0.10±0.13**	0.18±0.19	0.22±0.19	0.17±0.18
DS_0.5	0.05±0.10	0.18±0.19	0.27±0.16	0.17±0.18
DS_0.75	0.00±0.00	0.10±0.12	0.32±0.19	0.14±0.18
DS_1.0	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
BADGE	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
Coreset	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

Table 3: Benchmark of Effectiveness (at 20% of sample size) on different noise magnitudes. Bolds indicate the best methods.

	Noise Magnitude			
Methods	Low	Medium	High	Overall
US	0.00±0.03	0.02±0.07	0.16±0.20	0.06±0.14
DS_0.0	**0.29±0.21**	**0.31±0.18**	0.38±0.1	**0.33±0.18**
DS_0.25	0.10±0.13	0.26±0.13	0.50±0.11	0.28±0.21
DS_0.5	0.14±0.13	0.27±0.25	**0.52±0.15**	0.31±0.24
DS_0.75	0.05±0.10	0.24±0.27	0.43±0.08	0.24±0.23
DS_1.0	0.00±0.00	0.05±0.10	0.15±0.20	0.07±0.14
BADGE	0.00±0.00	0.01±0.05	0.06±0.14	0.02±0.09
Coreset	0.00±0.00	0.00±0.00	0.05±0.10	0.02±0.06

Acknowledgments. We gratefully acknowledge the generous support from the CUHK's Improvement on Competitiveness in Hiring New Faculties Funding Scheme and the CUHK's Direct Grant Project Number 4055191.

References

Raja Hafiz Affandi, Emily Fox, Ryan Adams, and Ben Taskar. Learning the parameters of determinantal point process kernels. In *International Conference on Machine Learning*, pages 1224–1232. PMLR, 2014.

David Alvarez-Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439, 2020.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*, 2019.

Alexei Borodin and Eric M Rains. Eynard–Mehta theorem, Schur process, and their Pfaffian analogs. *Journal of Statistical Physics*, 121(3):291–317, 2005.

Samuel Budd, Emma C Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Medical Image Analysis*, 71:102062, 2021.

Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy MAP inference for determinantal point process to improve recommendation diversity. *Advances in Neural Information Processing Systems*, 31, 2018.

Sangwoo Cho, Chen Li, Dong Yu, Hassan Foroosh, and Fei Liu. Multi-document summarization with determinantal point processes and contextualized representations. *arXiv preprint arXiv:1910.11411*, 2019.

Ali Civril and Malik Magdon-Ismail. On selecting a maximum volume sub-matrix of a matrix and related problems. *Theoretical Computer Science*, 410(47-49):4801–4811, 2009.

Greg d'Eon, Jason d'Eon, James R. Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 1962–1981, New York, NY, USA, 2022. Association for Computing Machinery.

Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960*, 2022.

Mike Gartrell, Ulrich Paquet, and Noam Koenigstein. Low-rank factorization of determinantal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. *Advances in neural information processing systems*, 27, 2014.

Xinru Hua, Truyen Nguyen, Tam Le, Jose Blanchet, and Viet Anh Nguyen. Dynamic flows on curved space generated by labeled data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, 2023.

Sadeep Jayasumana, Richard Hartley, Mathieu Salzmann, Hongdong Li, and Mehrtash Harandi. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2013.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2-3):123–286, 2012.

Hui Lin and Jeff Bilmes. Learning mixtures of submodular shells with application to document summarization. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012.

Odile Macchi. The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122, 1975.

K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Besmira Nushi, Ece Kamar, and Eric Horvitz. Towards accountable AI: Hybrid human-machine analyses for characterizing system failure. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 6, pages 126–135, 2018.

Neoklis Polyzotis, Steven Whang, Tim Klas Kraska, and Yeounoh Chung. Slice finder: Automated data slicing for model validation. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE), 2019*, 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

Svetlana Sagadeeva and Matthias Boehm. Sliceline: Fast, linear-algebra-based slice finding for ML model debugging. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2290–2299, 2021.

J. Schur. Bemerkungen zur theorie der beschränkten bilinearformen mit unendlich vielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 1911(140): 1–28, 1911.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. Understanding failures of deep networks via robust feature extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12853–12862, 2021.

Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In *Advances in Neural Information Processing Systems*, volume 33, pages 19339–19352, 2020.

John Urschel, Victor-Emmanuel Brunel, Ankur Moitra, and Philippe Rigollet. Learning determinantal point processes with moments and cycles. In *International Conference on Machine Learning*, pages 3511–3520. PMLR, 2017.

Mark Wilhelm, Ajith Ramanathan, Alexander Bonomo, Sagar Jain, Ed H Chi, and Jennifer Gillenwater. Practical diversified recommendations on Youtube with determinantal point processes. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 2165–2173, 2018.