Towards a Safer and More Reliable Selective Classifier

With Human Knowledge and Value Incorporated

Xinyue Chen

Delft University of Technology

TU Delft

Towards a Safer and More Reliable Selective Classifier

With Human Knowledge and Value Incorporated

by

Xinyue Chen

Student Name	Student Number
Xinyue Chen	5212642

Instructor:	Jie Yang
Teaching Assistant:	Agathe Balayn, Philip Lippmann, Andrea Mauri
Committee:	Geert-Jan Houben, Jie Yang, Seyran Khademi
Institution:	Delft University of Technology
Place:	Faculty of Computer Science, Delft
Project Duration:	November, 2021 - August, 2022

TU Delft

Preface

Two years is a long and short period of time. It's long enough for me to acquire all the knowledge I need to know to come up with this paper and get used to the life in a foreign country. In the meantime, it's so short that I sense I still have so much to learn and explore. But here I am, proudly presenting the output for the past 9 months and even more.

It has been a challenging but harvesting journey to work on this project, and it is sure to remark a mile stone of my master program at TU Delft, as well as an ending note to my student life. Apart from my own efforts, this work will not be accomplished without the support from the people surrounding me.

First, I'd like to thank Professor Jie Yang from WIS group, who has shown great encouragement to my academic research. He involved me in a research project about re-defining the value of machine learning, and this study is exactly inspired by and an extension of that. I also much appreciate the help from Agathe Balayn, Phillip Lippmann, Andrea Mauri, who have been continuously giving me constructive feedback and advice in every meeting we had. As this project inherited part of the work from Shahin Sharifi Noorlan, I'm grateful that he answered my questions and helped me set up the framework. Next, I would like to thank my parents, who constantly prompt me to be my best self even though we are oceans apart. The thank-you list continues with my friends and my housemates - without the meet-ups in the library, the small chats, the spontaneous walks to get snacks, this long journey would have been significantly less fun. Lastly, I'm grateful that all the people who love me and I love are living a safe and happy life - this seemingly simple life certainly takes some luck in this time. The progress of science and arts should make that type of simple life accessible to more and more people. And I hope I've made my small contribution to it.

Xinyue Chen
Delft, August 2022

Abstract

While the performance of traditional confidence-based rejectors is heavily dependent on the calibration of the pretrained model, this study proposes the concept of feature-based rejectors and the whole pipeline where such rejector can be used in. Multiple design and development decisions along the implementation are discussed in the paper - involving humans in the loop of creating the feature set that machine should use and defining value, using interpretability methods such as saliency map to extract the features that the machine actually uses, and SceneRejector is the end product of the whole process. It is relevant to the field of computer vision, and it is applied to the task of scene classification within the scope of this paper. Its performance is evaluated against baselines with accuracy, rejection rate, and a new metric named value, which more comprehensively measures the practical value of machine learning with a reject option in different use cases. Experiments have proved the concept of a feature-based rejector works and it is able to filter out unknown unknowns, which is a challenge to the confidence-based rejectors. It also creates better value than the traditional confidence-based rejectors in some cases, especially when the machine learning model is not well calibrated. Further analysis is conducted to understand the behavior of SceneRejector as well. It is discovered that SceneRejector brings better value than confidence-based rejectors when the pretrained model is not well calibrated, and that rejection rate and accuracy of the rejector is also correlated with the value.

Contents

List of Figures

List of Tables

1

Introduction

As machine learning algorithms are employed in more and more applications in real life, ranging from job recruitment [14] to public security and surveillance [32], it is influencing various aspects of people's life more and more. Despite the efficiency and convenience brought by them, machine learning models, especially the ones used in high-stake industries, are very likely to cause a negative consequence if their decisions are not properly filtered or rejected. For example, Amazon's recruiting AI proves to hold prejudice against women applicants [11], because the AI was trained with Amazon's past recruitment data, which is not balanced in terms of gender. As a result, it learned profiles that include "women" are less preferable, and the decisions it made also followed the pattern to exclude women in tech industry, which again reinforced the vicious cycle.

Hence, it has become an important and indispensable issue to safeguard the decisions made by

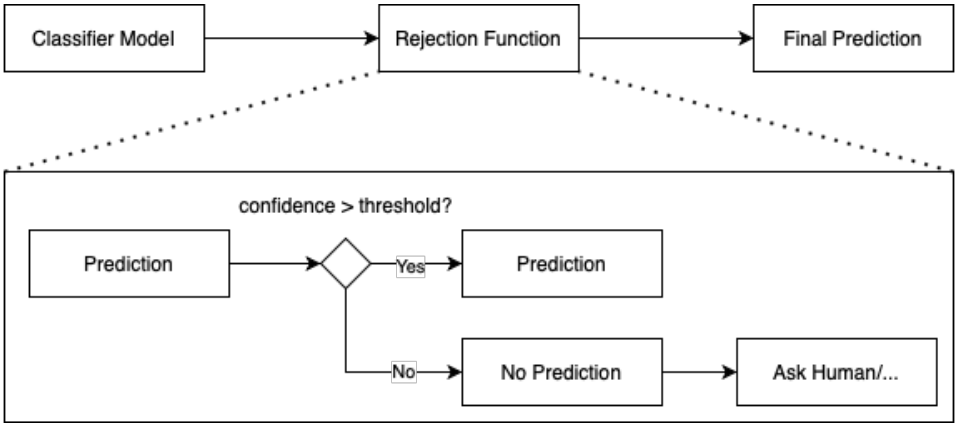

Figure 1.1: Traditional workflow to reject machine decisions

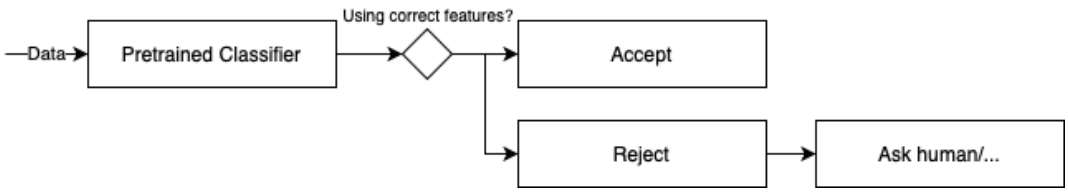

Figure 1.2: Workflow of Proposed Pipeline

machines, so that users can be protected against undesired consequences from false predictions. The traditional and most common way is to accept a machine decision if the prediction confidence exceeds a certain threshold and reject otherwise, as is shown in figure 1.1. This introduces the concept of a "selective classifier". Geifman and El-Yaniv [16] defines a selective classifier as a pair of function (f, g), where f is a standard classifier and g is a rejection function. Apart from applying an arbitrary confidence threshold, existing works [10, 16] mainly set the rejection threshold by maximizing a performance function or risk-coverage function, and the rejection threshold is still based on prediction confidence or neuron value in case of a neural network. This builds on the assumption that the model is well calibrated, which means the model's prediction confidence is very well aligned with its accuracy [31]. However, this may not always be the case, especially for neural networks [18]. Furthermore, such confidence-based rejection methods are not immune to high-confidence errors, which is also known as unknown unknowns. Unknown unknowns are one of the major challenges in reliable and responsible image recognition, and they are likely to occur when the training dataset is biased or when the classifier learned the wrong features (from a human's perspective) to make the prediction. Therefore, this paper proposes a new type of rejector that is not solely based on confidence, in a gesture to overcome the limitation of the confidence-based rejectors.

From the perspective of safeguarding machines decisions, it is critical to ensure that a model makes prediction with the knowledge that it should know. For example, a model shouldn't make a classification decision solely based on the background of the image in a bird classification task, but rather it should learn and use more features from the bird itself, including the color of feather, the shape of beak, etc. Referring back to the Amazon example mentioned previously, it would be ideal to attach a rejector to the recruitment AI so that it can reject the decisions if gender is considered during AI decision making. In short, a selective classifier should acknowledge the relevant and sufficient knowledge to make a decision, and if the utilized knowledge for a classification does not lie in that range, then the decision should be reconsidered for acceptance, so that the decision output by the classifier is both reliable and responsible. Figure 1.2 shows the underlying logic.

With this as a starting point, this paper aims to answer the following research question: how can we effectively improve the reliability and social value of a pretrained classifier, by adding to the end of the workflow a rejector that inspects the features used for prediction? In an effort to answer this question, we propose a pipeline to reject machine decisions with human in the loop, and within the scope of this project, we will use computer vision classification task to test the performance of this pipeline.

The proposed pipeline consists of 2 workflows, one being rejector training, and the other being rejector testing.

The rejector training workflow includes three components: "should know" extraction, input from pretrained classifier, and the rejector. The goal of the "should know" extraction is to find the relevant and sufficient features of each class. In other words, this step extracts what the classifier should know when making a prediction, which can be later compared with what it really knows. However, what knowledge is suitable to use while making a prediction and where should we obtain such knowledge? As humans know the potential bias a machine would learn, it makes more sense to consult humans to determine what information the machine should utilize, and thus we employ the Scalpel-HS framework [40] to implement the crowdsourcing task. The input from pretrained classifier usually consists of predicted class, prediction confidence, the featured used for prediction, which would need techniques such as SmoothGrad [42] to extract. The third and most important part is the rejector, which is a machine learning model that will learn the features, compare with the features used by the pretrained classifier for prediction, determine whether the used features are sufficient to come to a prediction, and output a reject or accept decision in the end.

The rejector testing workflow is essentially to apply the trained rejector to existing pretrained classifiers. The required input from the pretrained classifier is the same as training workflow, and the rejector will output a reject or accept decision based on that information. It's important to notice that as long as the task doesn't change, the rejector can be used for any pretrained classifier, once the it is well trained.

As for evaluation, we cannot proceed without acknowledging that this proposed pipeline aims to improve the reliability and social values of pretrained classifiers, rather than only accuracy metrics. We must separately consider the value of a wrong prediction, a correct prediction, and a reject condition, as in real life applications they are likely to have different impacts. In this sense, a classifier with a better rejection function may produce a higher value than a classifier with higher accuracy score. Hence, the pipeline will be compared with other state-of-the-art rejector baselines on both traditional metrics such as accuracy and the metric known as value, which will be further explained in section 2.5.

The goal of this study is to create a feature-based rejector and embed it in the machine learning pipeline, so that it is able to make pretrained machine learning models to make predictions or create value with the right reasons. We took an approach that involves humans in the loop to determine the relevant and sufficient features for a machine learning model to use while making a prediction and to define the value which is considered in the optimization target. As a result, this study implemented a feature-based rejector, SceneRejector, in a scene classification task. Experiments have proved that feature-based rejector is a working concept and SceneRejector creates better value than baseline as the cost of a wrong prediction increases. High confidence errors, unknown unknowns, are successfully rejected, and SceneRejector is most suitable to use when the pretrained classifier is not well calibrated.

2

Literature Review

This section will review the existing works that align with the scope of this paper, and it will show the gap that this study tries to fill. First, the definition and root cause of unknown unknowns, which is the type of error that this paper aims to eliminate, will be introduced, along with existing methods to deal with them. Second, it will present the motivation and definition of a selective classifier, its taxonomy, and how this study differs from other existing selective classifiers to safeguard machine decisions. Third, as the human-in-the-loop feature extraction of "what a model should know" is part of the proposed pipeline, the methods that other studies leverage to extract semantic features from images are also reviewed. Fourth, apart from "should know", it is also important to acknowledge what the model really knows, and thus multiple studies about model interpretability are included. Last but not least, evaluation metrics to measure the performance of the selective classifier are discussed. In short, the contribution of this paper includes a selective classifier that 1) aims to eliminate unknown unknowns; 2) comes with a feature-based rejector, in the context of neural networks; 3) involves human in the loop to determine domain requirements.

2.1. Unknown Unknowns and Calibration

In scientific research, there are usually three types of knowledge: known known, known unknown, and unknown unknown. Known known refers to the things we know that we know; known unknown refers to the things we know that we don't know; and unknown unknown refers to the things we don't know that we don't know [26]. Extending these concepts to the field of machine learning, known known and known unknown denote high-confidence correct predictions and low-confidence wrong predictions, respectively. They both share the common ground: confidence is an indicator of the accuracy; in other words, the machine learning model is well calibrated. On the other hand, unknown unknown denotes high-confidence errors, which stem from poor calibration.

Methods to deal with low-confidence errors have been extensively studied, e.g. a large amount of works about active learning attempt to identify informative cases to find the errors of the model [5]. There are many sampling strategies devoted to finding the most relevant samples to eliminate model

errors, including uncertainty sampling [25], query-by-bagging [9], etc. However, one trait shared by all of the strategies is that they try to improve model accuracy by enhancing the performance in the area that the model is known to be prone to failure, i.e. known unknowns. Furthermore, in the context of selective classifier, the regular confidence-based rejector can easily filter out this type of error, as confidence is a strong indicator of the error.

On the other hand, unknown unknown is a critical challenge for model performance, especially for the high-stake applications, because the errors cannot be detected with the information provided by the model. There are two ways to work around unknown unknowns: 1) calibrate prediction confidence and use the same strategy as low-confidence errors [47, 35]; 2) detect unknown unknowns and reject them [2, 40]. The reliability of prediction confidence is important for applications in sensitive domains [30], and there exists several post-processing methods to calibrate output from classification algorithms. Zadrozny and Elkan's work [47] has shown that binning is able to significantly improve naive Bayesian probability estimates and that smoothing by m-estimation and a new variant of pruning, curtailment, can improve decision tree probability estimates. Platt Scaling [35] can calibrate the output probabilities by passing them through a learned sigmoid function as well.

Nevertheless, the scope of this paper is focused on the second approach, which is to reject the unknown unknowns, and this approach is not restricted to the type of algorithm used. Levin et al. [24] examined and analyzed the network parameters responsible for erroneous decisions and found several types of mistakes that might have high confidence: 1) the target object is confused with another object in the image; 2) the salient pixels are focused on the target object features which confuse the network (e.g. confusing a leopard and jaguar by only focusing on the pattern of fur); 3) the model uses background information to make a prediction; 4) when there's a noisy label, the network is "more correct" than the target label. The first three types of unknown unknowns can be summarized as not using both sufficient and relevant features to make the prediction, and this is what this study aims to tackle. Attenberg et al. [2] designed a game-like crowdsourcing task to engage human intelligence to collect samples that they believe might be difficult for the machine, thus being able to detect unknown unknowns. Apart from methods to proactively elicit unknown unknowns, comparing the model's saliency map with the requirement specifications can also characterize unknown unknowns. Sharifi Noorian et al. [40] involved human in both requirement specifications and machine interpretations, namely deriving the features the machine learning model should know and really knows. It is proven effective and informative to characterize unknown unknowns by juxtaposing the two sets of features.

Although this study seeks to eliminate unknown unknowns, it does not explicitly detect or characterize them. However, this process is implicitly embedded in the rejector's functions. To the best of my knowledge, this study is the first to apply a selective classifier to handle unknown unknowns.

2.2. Selective Classifier

Selective classifiers, which output predictions selectively, are also known as machine learning with a reject option. The motivation for selective prediction involves lower error rate, better user trust, and more reliable and responsible performance in the end [21]. Mathematically, a selective classifier can be presented with the following function:

$$m(x) = \begin{cases} \emptyset, & \text{if } r(x) = 1 \\ h(x), & \text{otherwise} \end{cases}$$

where m is the selective classifier, r is the rejector to output binary reject decision, and h is the predictor, i.e. the original pretrained classifier.

Kilian et al. [21] summarize three types of architecture of selective classifier: separate rejector, dependent rejector, and integrated rejector. Separate rejector is placed before the predictor, and it decides what data samples to pass to the predictor. It is often used to filter data outliers. This architecture saves computation cost because predictor makes fewer decisions. On the other hand, as rejector and predictor do not share information, the rejector cannot learn from the predictor's misclassifications, thus resulting in sub-optimal rejection performance. Dependent rejector relies on the output of the predictor, and it accounts for predictor's information during rejection. Oftentimes it's dependent on confidence metrics from the predictor, and the quality of such metric can greatly influence the rejector performance. Furthermore, this architecture enables using rejector as an extension of an existing model. The integrated architecture integrates the rejector and predictor into a single model by treating the

rejection as a special case with an additional output $\emptyset$. In this way, both components are optimized, but a new algorithm is required and the rejector is not extensible. The rejector of this study aligns with the second architecture, because it is trained with information from the predictor, including the salient objects in each prediction.

While there are plenty of existing studies on selective classifiers, they are mainly focused on learning algorithms such as support vector machine [15, 27], random forest [6], nearest neighbors [20], etc. However, the reject option is rarely discussed in the context of neural networks or deep neural networks [16]. Cordella et al. [10] proposed a method to evaluate the classification reliability and to provide a reject option to neural networks. In their studies, an optimal reject threshold value is determined by maximizing a performance function which takes into account the requirement of the application and measures the classification quality regarding error rate and rejection rate. Rejection happens when the difference between the neurons with first and second highest values is below the threshold, and experiments have shown the threshold achieves the best trade-off between reject rate and error rate, improving the classification reliability. Geifman and El-Yaniv [16] approaches the classification reliability challenge from an alternative risk-coverage view. They introduced a practice to learn "a rejection function that will guarantee with high probability a desired error rate" and maximizes the coverage in the meantime. The rejection function is based on prediction confidence, so this method is still about setting a confidence-based threshold to reject samples in essence. Similarly, Stefano et al. [12] also base a reject option on reliability evaluator such as confidence, and a sample is rejected if this metric is below a threshold, which is defined as the one which maximizes the effectiveness function of cost coefficients.

Confidence-based rejectors dominates the majority of machine learning models with a reject option, and there are very limited studies on non-reliability metric based rejectors, especially feature-based rejectors. The study of Rajalakshmi and Aravindan [36] presents URL-based classifiers along with rejection framework and feature extraction techniques in later stages. The workflow extracts more and costlier features at each stage, and it only proceed to the next stage if the output from the previous stage is rejected. This practice saves computation cost and allows the rejector to filter out predictions with insufficient features. It is important to note that the feature-based rejector of this paper does not only eliminate predictions made with insufficient features, but also the ones made with irrelevant or wrong features. Moreover, instead of text classification, this study is focused on image classification. In short, our proposed rejector tries to fill the gap of feature-based rejector to neural networks, and it is also among the first to filter unknown unknowns, and to involve human in the loop to determine domain requirements, i.e. what is required for the model to know.

2.3. Feature Extraction with Crowdsourcing

Facilitating algorithm reliability and thus improvement, feature engineering plays a critical part in a successful real-world application of machine learning, because oftentimes stakeholders prefer applications that use human-interpretable features. Features of different dimensions can be extracted from training images, such as texture level features including intensity and smoothness [4] and semantic level features like the existence of certain objects [44]. The proposed rejection pipeline is designed to only focus on the semantic features due to the nature of the task of scene classification - it only requires this level of granularity because it is the object, rather than the image pattern, in a scene that determines the category. As for other image classification tasks, textual level features might be helpful as well. In the scope of this project, this section will only introduce semantic level feature extraction techniques from existing research.

It is currently not feasible to automatically extract features on a semantic level, and thus incorporating the crowd into a hybrid workflow can greatly facilitate the exploration of the feature space that are largely unreachable by machine [8]. According to [13], crowdsourcing is defined as "a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task." The practice of crowdsourcing is commonly used for data labeling for machine learning datasets, and is more and more used for machine interpretation with human-understandable concepts [28].

While what this pipeline requires from the crowd is not simply extracting features in the image, but the deterministic features that can distinguish class from class, we also look for crowdsourcing workflows that can meet this requirement during literature research. Cheng et al. [8] proposes Flock,

which involves crowdsourcing to extract useful features to improve ML performance. Considering the crowd may have minimal domain expertise, Flock presents two contrasting images from the positive and negative class and ask the crowd which one belongs to the positive class and why. The reason are then automatically clustered. Another crowd task is then designed to aggregate each cluster and generate the feature of each cluster. The generated features for the same cluster are then fed into a voting task to determine which feature is most representative. In this way, Flock can detect the deterministic features of each class, although it only applies to binary classification. AdaFlock [44] is a variation of Flock that aims to improve the classification accuracy by iteratively applying the Flock workflow. Each crowd worker is shown three positive and three negative examples randomly selected from the training dataset and is asked to provide a feature definition. Afterwards, in each iteration the crowd workers are required to define features helpful for correctly classifying the misclassified examples in the previous iteration. As a result, the classifier accuracy is improved with each iteration. Zou et al. [48] introduce another crowdsourcing workflow that works in a intuitively similar way as Flock, by finding differences and similarities in a set of images. They construct a binary feature tree with "two-out-of-three" queries, where they present a crowd worker with a triple of examples and ask them to name a feature common to any two out of the three examples. The triple is randomly sampled from unresolved queries, and a query is consider resolved if there is a known distinguishing feature for the query. Once a feature is discovered, another crowd task is used to label the remaining data according to the feature. In the next round, the new "two-out-of-three" queries are sampled from unresolved triples with the same least common ancestor. This method can elicit various features without repetition.

Apart from feature extraction from contrasting images, Sharifi Noorian et al. [40] proposes the Scalpel-HS framework to generate descriptions of what a model should know and really knows in a scene classification task. As for the "should know" task, it aims to "understand the salience of each object and relation in identifying the scene in the given image". It first extracts scene graphs of the images, and it then asks the crowd to verify the automatically generated relations and rate the relevance of each relation within the class, on a scale from 1 to 20. The crowd workers can also add missing relations that they consider to be relevant to the scene. The "really knows" task's goal is to find out what object or relation significantly contribute to the prediction made by the machine learning model. In this task, the crowd workers need to annotate the highlighted areas in extracted saliency maps, which facilitates to understand machine behaviors. They also define the relations among the objects. Experiments have proven that the comparison between the results from the two tasks can efficiently detect unknown unknowns. Furthermore, this framework also reduces the cognitive load compared with previous crowdsourcing workflow. Hence, this project chooses the Scalpel-HS framework to conduct feature extraction regarding what a machine should know.

2.4. Model Interpretability

Having discussed how to extract the features that a machine learning model should learn and use, it's also necessary to understand what the machine really knows in order to tell whether its knowledge falls into the desired range, which is mentioned previously. Features used to make predictions are usually hidden in millions of parameters, and they are not always comprehensible to human, or at least not obvious to human's mental representation of the world, because human perception of the world mainly consists of observable properties [1]. Furthermore, the lack of machine interpretability can lead to huge risks and user distrust in high-stake domains such as health and security. Therefore, it has become more and more critical to understand the behavior of machine learning models. Thus, human can mitigate the cost of a wrong prediction if we can investigate and discover what has gone wrong during the learning process. For example, Levin et al. [24] identified and analyzed the network parameters that are responsible for erroneous predictions, and they showed that pruning those parameters often contributes to a better model performance.

Plenty of studies have proposed different techniques to interpret machine decisions, and they can be classified in two categories: one is to build inherently interpretable models, and the other is to interpret machine decisions post-hoc. This section will focus on post-hoc methods. In the field of natural language processing or text classification, the LAMA (LAnguage Model Analysis) Probe is one of the first and most popular interpretation techniques, and it is originally applied in the study of Petroni et al. [34] to evaluate relational knowledge contained in large pretrained language model such as BERT. Following studies [22, 46, 33] continue to optimize the technique and elicit machine knowledge in more

domains.

While it is quite complicated to encapsulate and interpret the features learned by language models, interpretation of computer vision models can be more straightforward, as it is easier to visualize image predictions. Saliency map [41] is one of the most extensively studied and widely used methods to interpret image classification models. It highlights the pixels that make an important contribution to model prediction with regard to a specific predicted class. To be more specific, an image classification model has a class activation function for each class, and the gradient of the activation function for each pixel is related the extent to which the pixel will influence the classification decision. Hence, saliency map is essentially a heatmap of the gradient values [41]. There are several methods to generate saliency map [41, 39, 43], and SmoothGrad [42] is used in this project, as it can generate less noisy results.

An extra step has to take place to interpret machine predictions, as saliency map only gives explanation on a pixel level instead of a human-comprehensible semantic level. In order to obtain semantic-level interpretation, human is usually involved in the process. For example, Kim et al. [23] proposed testing with concept activation vectors (TCAV), which can "quantify the the model's sensitivity to the user-given concept for the predicted class". Balayn et al. [3] adopted the approach of crowdsourcing to annotate the saliency map for high-fidelity and scalable machine interpretation. The SECA framework is designed to ask crowd workers to annotate the entity and attribute of the highlighted areas in a way that requires minimum cognitive efforts. The "really knows" task from [40], mentioned in the previous section, works in a similar way. Aggregating the crowdsourced results, SECA framework can show the important features used for the prediction of each class. Apart from the methods that require human efforts, Ghorbani et al. [17] introduced the ACE algorithm, which can automatically provide concept-based explanations. Images are segmented with different resolutions and then clustered. TACV score of each clustered concept is calculated to find out important features of each class.

Compared with interpretation methods with human in the loop, current automatic processes are prone to low fidelity and the performance is dependent on the quality of segmentation and clustering. Nevertheless, this project still uses the automatic method to discover the features leveraged to make a classification, which is to match the highlighted area in saliency map to bounding boxes derived from a pretrained object detection model. Admittedly, such approach is sub-optimal in that it heavily relies on the quality of object detection model and the only information available is the entity without any attributes. However, the goal of incorporating a interpretation method in the pipeline is not to fully understand complex model behavior, but rather to simply check whether the feature used by the model falls into the range of knowledge that a model should know. Therefore, automatic interpretation is sufficient in the use case of this pipeline, and it also increases the workflow efficiency and saves the cost of crowdsourcing.

2.5. Evaluation of Machine Learning Models and Rejectors

As the objective of this project is to build a more responsible and reliable machine learning pipeline, traditional evaluation metrics such as accuracy are no longer sufficient to evaluate the performance of the pipeline. Meanwhile, a few position paper [37] have recently challenged the predominance of accuracy metrics, including precision and recall, in the realm of evaluating the quality of machine learning models. Despite accuracy metrics should remain an important evaluation method in machine learning research, it is critical to acknowledge that the gap between machine learning usage in laboratory and enterprise is usually not due to low accuracy, but rather uncertainty of its reliability. In high-stake domains such as medicine, where there is very low fault tolerance, an error made by a 99% accurate model can cause huge damage, and therefore it is very critical to safeguard each decision made by machine. As mentioned in section 2.2, confidence is often used as the indicator to the reliability of a machine decision. Nevertheless, confidence is not always a reliable, and calibration metrics such as expected calibration error (ECE) [29] are introduced as a measurement of calibration. ECE segments the probability interval into a fixed number of bins and assigns each predictions to the according bin. The calibration error is then defined as the expectation of the difference between accuracy mean and confidence mean in each bin (see equation below).

$$ECE = \sum_{b=1}^{B} \frac{n_b}{N} |acc(b) - conf(b)|$$

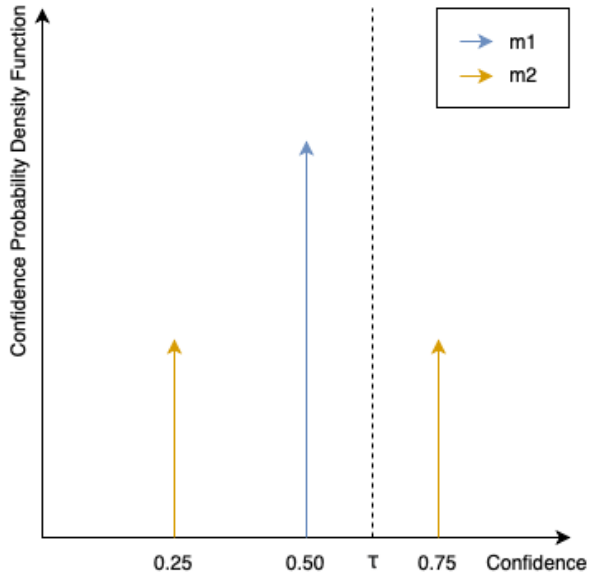

Figure 2.1: Probability density function of two models over the same dataset

While ECE is widely used to measure calibration, it has several issues [30]. ECE is computed only with the probability of the predicted class, and hence the metric is not able to accurately assess the calibration error for other classes. Furthermore, the confidence distribution is not uniform but the binning is evenly spaced, which causes only a few bins to make major contribute to the final ECE score. New calibration metrics have been introduced to deal with the above-mentioned issues. Static Calibration Error (SCE) is a simple extension of ECE in the multi-class setting. Adaptive Calibration Error (ACE) creates uneven confidence intervals by having equal number of predictions in each bin, so that the metric can focus on the regions where the majority of predictions are made, while paying less attention to regions with fewer predictions.

Apart from calibration metrics to measure machine learning reliability, we cannot proceed without acknowledging that real-life machine learning application should be evaluated with practical values as well, because the value of a correct prediction, a wrong prediction, and a rejection vary from use case to use case. It also makes sense to consider different use scenarios when comparing different machine learning models. Casati et al. [7] proposed the following formula to calculate the average value of a machine learning model g over dataset D, with a reject option:

$$V(g, D) = \rho V_r + (1 - \rho)(\alpha V_c + (1 - \alpha)V_w)$$

where V_r is the value of a rejection, V_c is the value of a correct prediction, V_w is the value of a wrong prediction, ρ is the percentage of rejection, and α is the accuracy. Consider the case we do not use machine learning at all, in other words, all the predictions are rejected. We have zero value under this assumption, and therefore we can set $V_r = 0$. We can also set $V_w = -kV_c$, where k is an indicator of how much negative value will a wrong prediction bring with respect to a correct prediction. With these simplifications, the formula then becomes:

$$V(g, D) = V_c(1 - \rho)(\alpha - k(1 - \alpha))$$

In Casati's work [7], a confidence threshold τ is selected to maximize the value of $V(g, D)$ and to be used as the rejection threshold. τ can be either empirical or theoretical. The empirical threshold is set with some tuning dataset, and the theoretical threshold can determined by the value of k. In theoretical setting, we assume the model has perfect calibration, so the accuracy α is equal to confidence c, i.e. $\alpha_\tau = \tau$. To have $V(g, D) > 0$, we need $\tau - k + k\tau > 0$, which leads to $\tau > \frac{k}{k+1}$. Therefore, the theoretical threshold is $\frac{k}{k+1}$. Finally, comparing the value of different machine learning models over different k can provide us with better insight into the model performance in different use cases.

On top of that, the value metric can also capture some qualities that ECE or accuracy score cannot capture, especially in the context of machine learning with a reject option. For example in figure 2.1, we

can assume they all have ECE = 0. $m1$ and $m2$ have the same accuracy, but $m2$ has better value than $m1$, with the rejection threshold τ.

Therefore, apart from calibration metrics and regular performance evaluation metrics for machine learning with a reject option, such as accuracy and coverage [21], value will also be a major evaluation metric in this project. We will also use the confidence-based rejector, mentioned previously in this section, as a baseline to compare with the proposed rejector.

3

Methods

3.1. Problem Setting

This section will present the intuition of the general key ideas of the solution this paper proposes. To start, a review of the goal of this paper can be useful to understand the solution:

- Instead of traditional and wide-used confidence-based rejectors, this study aims to implement a feature-based rejector to embed in a selective classifier.
- Human is involved in the loop to determine domain requirements, i.e. the sufficient and necessary features for the classifier to make a decision, which is used to train the rejector afterwards.
- The proposed rejector is able to filter out unknown unknowns.

In a gesture to implement the above mentioned functions, we propose a training pipeline with three components and a testing pipeline with two components.

In the training pipeline, the first component is "should know" extraction. It is carried out by crowdsourcing tasks, from which we can acquire the relevant and sufficient features to make a prediction, i.e. what the classifier should know. For example, the crowd workers may agree that "fridge", "stove", and "oven" are the sufficient and necessary features to predict a room as a "kitchen". The input is the second component. It is always derived from a pretrained classifier - note that we do not train a classifier in this pipeline - e.g., the prediction, the confidence, and the features that the classifier used to make a prediction (what the classifier really knows). Since this pipeline is used for training, the ground truth labels of the data fed to the classifier are known. The last component is the rejector to train. The rejector is trained with both the input and the crowdsourced results. In the scope of this project, we assume all the correct predictions should be accepted and all the wrong predictions should be rejected. Thus, the rejector can learn to reject predictions by matching the "should know" and "really knows" and comparing it with the ground truth labels.

In the testing pipeline, the two components are the input and the already trained rejector. The input also comes from the pretrained classifier, including the prediction, the confidence, and the used features. However, ground truth labels will not be used even if available. The input can be fed into the rejector,

which will output a decision of accept or reject. This pipeline is what can be embedded in a machine learning application.

3.2. Pipeline Overview

While section 3.1 introduces the main idea and intuition of a general pipeline that this study proposes, this section will give a structural overview of the proposed pipeline in the context of image classification. The training and testing pipeline follow the same structure described in section 3.1, and the three major components - "should know" extraction, input from a pretrained classifier, and a pluggable rejector - will be introduced regarding their purpose and function.

3.2.1. "Should Know" Extraction

The selective classifier rejects predictions by the pretrained classifier based on the used features in the image. Therefore, it is critical to determine what features can be counted as "relevant" and "sufficient" for each class. Including human in the loop can provide a means to answer this question, and some crowdsourcing tasks are designed to derive the features that the pretrained classifier should use when giving a prediction, from a human perspective. For example, when showing crowd workers multiple room images, they may come to a conclusion that "bed" should be more relevant than "window" in the class "bedroom". The "should know" features will then be used to train the rejector together with the input from the pretrained classifier, and this will only happen in the training pipeline.

3.2.2. Input from a Pretrained Image Classifier

The proposed rejector is pluggable to any pretrained classifier, so there is no specific requirements of the pretrained classifier, as long as it's possible to derive the necessary data from the classifier. As mentioned in section 3.1, the input obtained from a pretrained classifier ("really knows") includes the prediction, the confidence, the used features, etc. Since the proposed rejector is feature-based rather than confidence-based, we do not need the confidence but the prediction and used features. To be more specific in the context of image classification, the features are demonstrated with saliency map. Saliency map [42] is a visualization technique to interpret machine predictions - it highlights the pixels in the original image with the gradient of the class score function, thus making it more convenient for human to find what parts of the image makes relatively more contribution to the machine prediction. Apart from the visual level which is extracted from saliency map, the used features should also have a semantic level to annotate the highlighted entity. This is achieved by adding another object detection model to the pipeline, which will be explained in section 3.3. Finally, it's important to note that the input from the pretrained classifier of interest will be used in both training and testing pipeline, and the only difference is the used dataset.

3.2.3. Rejector

For a simplified explanation, a rejector is another machine learning model attached to the last layer of the pretrained classifier, with the input from the pretrained classifier and the output of a accept or reject decision. The rejector can be used for any pretrained classifiers as long as they are trained with the same task and same classes.

In the training pipeline, the rejector learns to reject and accept a prediction by comparing the "really know" and "should know" features and learning to what extent the overlap of two sets of features can lead to an accept. In other words, the input of the rejector is the features, and what the rejector aims to predict is whether the prediction is the same as the ground truth. If the prediction is equal to the ground truth, it outputs an accept.

In the testing pipeline, the rejector is ready to use. It reads the "really knows" input from the pretrained classifier and can decide whether the prediction should be rejected or not. Note that the "should know" features are not explicitly used in this pipeline because it is already learned by the rejector.

3.3. Pipeline Implementation

This section will introduce the implementation of the pipeline in detail, including all the techniques used in each component described in previous sections. As the overall workflow is shown in figure 3.1,

the training and testing pipeline both have the same components as previously described. While this pipeline is generally applicable to most image classification tasks, we will take the scene classification task as an example to better demonstrate it. In a scene classification task, an image classifier needs to predict the type of room, such as "bedroom" and "conference room", given a scene image. This study attaches a rejector to the end of such classifier to reject predictions made with insufficient or irrelevant features.

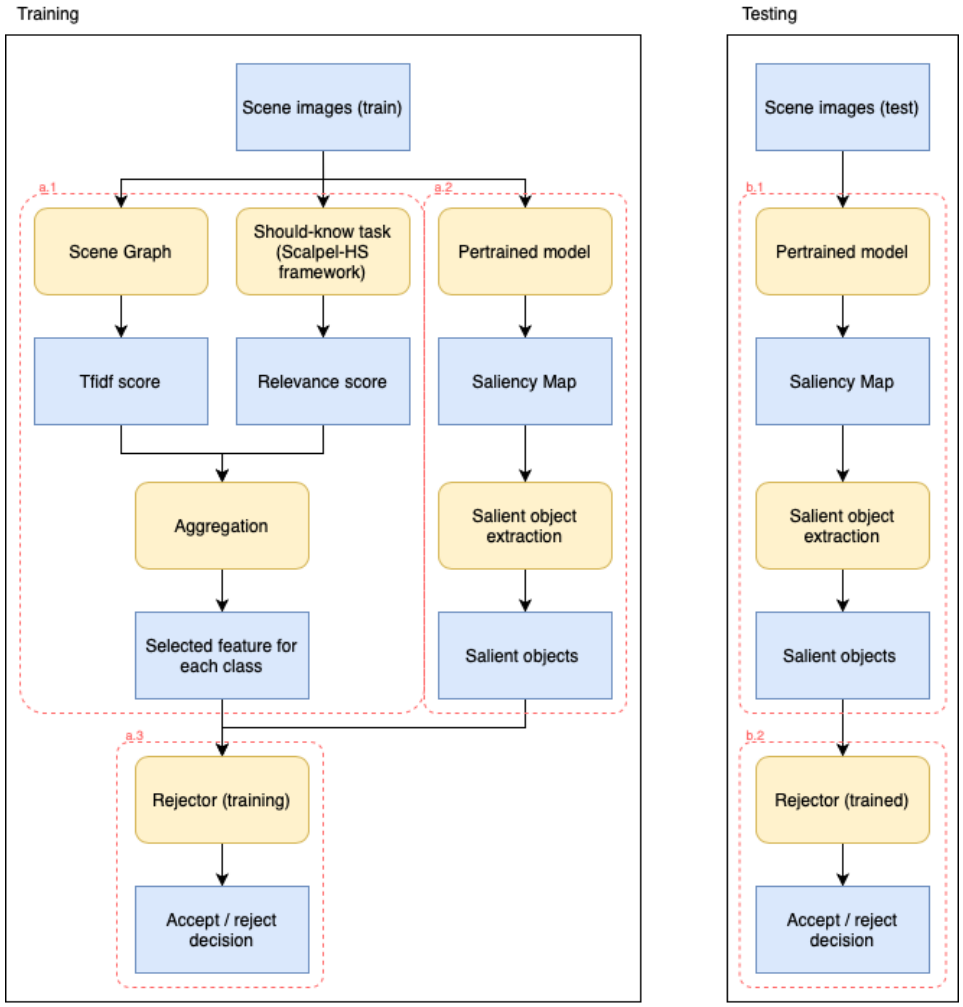

Figure 3.1: The overall workflow of the pipeline. The training pipeline includes 3 components: 1) "should know" extraction, 2) input from pretrained classifier, 3) the rejector. The testing pipeline includes only 2 components: input from pretrained classifier and the rejector.

3.3.1. "Should Know" Extraction

The training process requires the rejector to know the features of each class that the classifier should know, i.e. sufficient and relevant features to make a prediction. We need to include human in the loop to extract this type of knowledge, because in the end the end users are human and thus it is important that the machine predictions can be interpreted from the human perspective.

In summary, the "should know" extraction consists of two methods to find the relevant features of each class, which means the indispensable objects of a particular scene in this scene classification example. The first method involves a crowdsourcing task adopted from Scalpel-HS framework [40] to aggregate the relevance of the objects in each class. In the second method, an object detection model, e.g. Scene Graph [45] used in this project, is also applied to detect all the objects in the image, with which we can also estimate the importance of an object in each class by calculating its tfidf scire.

Combining the relevance from the crowdsourcing task and the tfidf score of each object in each

class, we are able to select the deterministic features which are relevant and sufficient for each class.

Scalpel-HS Framework
The Scalpel-HS framework [40] consists of two tasks, which are the "should-know" task and the "really-knows" task. In this project, only the "should-know" task is used.

Scene Graph [45] is first used to extract objects and relation in an image, and then the predictions from Scene Graph is presented to crowd workers, who is required to validate the objects and relations as well as assigning relevance scores (on a scale from 1 to 20) to each relation to indicate how important they think a relation is in a scene class. They can also add a missing object if they think it is essential in the identification of the scene in the image.

Although the co-occurrence of objects is accounted, the specific relation of objects, such as "in" or "on", is not of interest within the scope of this project. Therefore, the crowdsourced results are transformed in the following way: 1) take the two objects in one relation, and assign the relevance score of the relation to both objects; 2) normalize the relevance score of each object in each class by averaging the sum of its relevance score with the number of its occurrence. In such manner, under each class, the objects can be ranked with their normalized relevance score, and any objects with a score greater than 10, which is the median of a scale from 1 to 20, can be considered for feature selection.

However, it is found that some generic objects, e.g. "room" and "wall", have a relatively high relevance in many classes, because "room" frequently occurs in relations such as "object in room". Consequently, it is not reasonable to select features solely based on the crowdsourced results.

Tfidf
As previously mentioned in section 3.3.1, the crowdsourced relevance score alone should not be the only factor to determine the critical features of each class, thus we introduce tfidf to rule out generic objects, such as "room" and "wall".

Each class is treated as a document, which consists of all the objects detected by Scene Graph [45] from all the images within that class. Then the tfidf score is calculated for each object in each class. Objects with a tfidf score higher than a certain threshold is considered identifiable for a class and may be selected as a feature. Since different classes have different tfidf distribution patterns, the threshold for each class is separately determined by manually inspecting the corresponding objects. In this way, we are able to factor out the generic objects.

Apart from removing the generic objects, tfidf score can also serve as a complementary metric to the crowdsourced relevance score for "should know" extraction. A union of the objects with crowdsourced relevance scores above a threshold and the objects with tfidf scores above a threshold is taken to generate the final features for rejector training. The features of each calss are shown in table 3.1.

Class	Fetures
living_room	counter, vase, table, couch, light, shelf, screen, fireplace, chair, book, curtain, person, window, rug
conference_room	cup, speaker, screen, water, desk, chair, table, person, window
dorm_room	table, bed, window, chair, book, shelf, drawer, curtain, closet, basket
bedroom	bed, lampshade, couple, cats, banner, post
bathroom	sink, shower, tub, toilet, towel, tile, toiletries, mirror, counter
kindergarten_classroom	window, kid, chair, adult, flag, book, board, table
kitchen	stove, plate, oven, person, sink, grate, counter, cupboard, dishwasher, kid, apron, food, pan, jar, table

Table 3.1: Selected Features of Each Class

3.3.2. Input from a Pretrained Classifier
In the example of scene classification, the required input from the pretrained classifier is the prediction and used features. The prediction can be very easily obtained, and the used features can be acquired via the salient object extraction method.

(a) Original Image **(b)** Scene Graph Object Detection **(c)** Saliency Map **(d)** Salient Object

Figure 3.2: Salient Object Extraction

Salient Object Extraction

The salient object extraction mainly consists of two components: an object detection model and a saliency map generation method.

An object detection model, which can draw bounding boxes and annotate the bounded object, is crucial in this pipeline, and it plays an important part in extracting the salient objects when the pretrained classifier makes a prediction. For the sake of a consistent vocabulary of objects, Scene Graph [45] is used in this project, as it is also used in the crowdsourcing tasks in [40]. Since there is no human verification for the correctly predicted images, any object with a prediction confidence greater than 0.5 from Scene Graph will be adopted. Considering the error rate of Scene Graph, if the confidence threshold is set too low, the annotations will not be correct. The threshold of 0.5 is arbitrarily selected, and the results seem to be accurate by inspecting the output images from hindsight.

SmoothGrad [42] generates the saliency map for all the images, but there remains another step to translate those highlighted pixels to objects. First, it is necessary to have a binary criteria of "salient object", as SmoothGrad gives continuous gradient-based sensitivity maps. We manually set the hsv color boundary, which roughly covers the spectrum of highlight from orange to red, to isolate those highlighted areas. Second, bounding boxes are automatically calculated for the highlighted areas. Last, by matching the bounding boxes of the salient areas with those from object detection, we can retrieve what object is highlighted in the saliency map. Figure 3.2 demonstrates a good example to extract salient objects: the original image is of the class "bathroom", Scene Graph detects three objects from it, and the saliency map shows that the classifier used the concept of "door" when making a classification. More details about the salient object extraction method is shown in algorithm 1, and the threshold of IOU score is set to 0.15, the decision of which will be explained in chapter 4. IOU is short for intersection over union, and it is a common way to measure the overlap between two bounding boxes. Intuitively, an IOU score of 1.0 indicates complete overlapping, and an IOU score of 0.0 indicates no overlapping at all.

Algorithm 1 Salient Object Extraction

Require: Pairs of bounding boxes and corresponding objects from object detection model for each image, bounding boxes for the highlighted areas in saliency maps, and a threshold

for each image **do**
    Suppose there is a set of bounding boxes B_1 from the object detection model and a set of bounding boxes B_2 from the saliency map
    for each bounding box b_2 in B_2 **do**
        Compute the IOU scores of b_2 and each bounding box b_1 in B_1
        Retrieve the pair of b_1 and b_2 with the highest IOU score
        if IOU(b_1, b_2) > threshold **then**
            Match the object corresponding to b_1 to b_2
        else
            No match for b_2
        end if
    end for
end for

3.3.3. Rejector Training

For the sake of simplicity, we transformed the original multi-class task into a binary classification task, and thus there will be as many rejectors as classes for the different classes. Having acquired the features of each class in section 3.3.1, we can start to train the rejector for each class.

Data Preparation

In short, the data used to train the rejector has the following columns: the features (shown in table 3.1) as input, and the reject/accept label as output.

First, the image dataset, with the same number of correct and wrong predictions, is randomly shuffled, and we take train test split as 0.8. Second, for each image, the existence of each object in the features is retrieved via salient object extraction, which is later formatted into one hot encoding for input. Last, the reject/accept label is decided according to whether the image has a correct or wrong prediction from the model. Intuitively, all the wrong predictions should be rejected and vice versa. Therefore, having the same amount of correct and wrong predictions can create a balanced dataset for rejector training. It is important to acknowledge that this labeling rule is not without fault, because not all correct predictions should be accepted - some of them may be correct for the wrong reasons.

Model

We can consider the rejector as a new machine learning model to predict rejection or acceptance, and it is attached to the end of the pretrained classifier. We tried different machine learning models, including decision tree, random forest, and support vector machine. In terms of performance, they show no significantly different accuracy score, and thus we decide to use decision tree as it's more interpretable and simple. Accuracy is the metric to optimize during training.

3.3.4. Rejector Testing and Evaluation

This section will introduce how to use the trained rejector and what metrics are used to evaluate the performance of the rejector.

Rejector Application

The workflow to apply the rejector is described in figure 3.1. The output of the pretrained model, which includes the prediction and the salient objects, is formatted in the way described in section 3.3.3, and then it is used as the input to the rejector. Afterwards, the decision of the rejector is ready to be used.

Evaluation Metrics

As discussed in section 2.5, common evaluation metrics for machine learning with a reject option include accuracy and coverage [21]. Apart from that, value is also an important metric to evaluate the performance of a selective classifier, because a correct prediction, a wrong prediction, and a rejection may all have different practical values in different real world applications, and value can measure the value across different use cases. Furthermore, section 2.5 has already demonstrated the quality of a selective classifier that cannot be measured only by accuracy and coverage. Review the following formula of value, and we will find accuracy and coverage is already represented in the equation by α and ρ, respectively.

$$V(g, D) = V_c(1 - \rho)(\alpha - k(1 - \alpha))$$

It is quite obvious that plots of value over different k are also an important visualized evaluation of the performance. However, that can only be a means to qualitatively analyze and compare the performance of the rejector against the baseline. To quantitatively evaluate the rejector performance, we introduce the following metric, Value Over Baseline (VOB):

$$VOB = \frac{\int_{k=0}^{10}(V_{rejector}(k) - V_{baseline}(k))}{10}$$

In this way, we can capture the average value increase or decrease against the baseline, which is the condition without a rejector, across all use scenarios. It is important to note that VOB can be used to compare performance between different experimental conditions as well, and the value of rejector and baseline in the formula can be substituted with the value of any experimental condition.

Finally, more analysis will be conducted to observe the rejector performance in chapter 5, however, those are not solely meant to evaluate the rejector performance in application but rather observe its behavior, and therefore will not be introduced here.

4

Design Decisions

The implementation of the pipeline have several design decisions to make, including multiple parameters to adjust and features to select from. This chapter will introduce and explain the design decisions made while building the pipeline, ranging from important feature selection in rejctor training to trivial parameters to tune.

4.1. Rejector Training Data

There are several design decisions to make when sampling data for rejector training, and this section will present why certain decisions are made. Simple experiments are carried out to find out which design is better, and chapter 5 will introduce the experimental setting in detail. However, the setting is not of significance in this section, and the only thing to note here is that there are two conditions, which are false positive (FP) and false negative (FN). There are two different pretrained classifiers for the two conditions, and the datasets are also different. The reason to experiment with the two conditions is simply to provide more solid evidence for the design decision we make.

4.1.1. Including Class Labels or Not

As mentioned in section 3.3.3, the input used in rejector training is the existence of relevant and sufficient features. The features are derived from "should know" extraction, and the salient object extraction method can tell what objects in the range of relevant and sufficient features are used when the pretrained classifier makes a prediction.

In a multi-class setting, it is quite intuitive to also include the ground truth class label in the input for rejector training, because it can indicate what features the classifier should know according to different classes. Nevertheless, does this intuition still hold when we transform multi-class to binary classification? One reasonable doubt is that the rejector will only learn the class distribution rather than the used features if we include class label in rejector training. To test this hypothesis out, we conducted the following experiment. We calculated the percentage of the images with the class label of interest in the rejected set and accepted set, and this is conducted for two conditions, where we include the class

label in training and where we do not. The results are presented in table 4.1. It is important to note that we use a binary classifier, and therefore each row in the table represents a separate rejector of the binary classifier that only classifies the class of interest.

Class of Interest	CP:WP	Class Label	%class in rejected	%class in accepted
living_room	0.0741	True	0.2820	0.0000
		False	0.1111	0.0227
conference_room	0.4000	True	0.3421	0.0000
		False	0.1842	0.0737
dorm_room	0.1458	True	0.5769	0.0000
		False	0.1024	0.5000
bedroom	5.600	True	0.0000	1.000
		False	0.1729	0.1729
bathroom	2.421	True	0.0085	0.8000
		False	0.1024	0.5000
kindergarten_classroom	69.00	True	0.0000	0.8276
		False	0.1667	0.1546
kitchen	2.750	True	0.0000	0.8276
		False	0.1765	0.0808

Table 4.1: The Impact of Including Class Label in Rejector Training to Class Distribution in Rejected Set and Accepted Set (FP)

As is shown in table 4.1, different rejectors are trained with data with different ratio of correct prediction (CP) to wrong prediction (WP) of the class of interest, despite we already sampled a balanced training set with same number of CP and WP. In other words, although there are 300 CP and 300 WP, for example, in the rejector training set, among the 300 CP, there might be only 50 images of the class of interest, and there might be 200 images of the class in the totality of WP. Please note the separate binary classifiers are modified from the same multi-class classifier, the training process will be described in detail in chapter 5. Therefore, such observation is reasonable because the pretrained classifier may perform better in predicting one class than another. Among the 7 classes, 3 classes have more WP than CP. If class labels are included in the rejector training, we can observe that the rejector does not accept any prediction of that class. The same pattern is discovered for the classes with more WP than CP as well, but the rejector does not reject any prediction of those classes in that case (except for bathroom, but the percentage of the class in rejected set is also close to 0). Removing the class labels from the rejector training set, we can detect a significant change in the class distribution in the rejected and accepted set for all classes, and it is more balanced than before. This experiment is only conducted in the FP condition, but it is obvious that same results will show in the FN condition, as the rationale should apply to any condition.

Hence, it can be concluded that the hypothesis is true that including class labels in rejector training data leads the rejector to only learning the class distribution rather than the used features. As a result, class labels are not included in rejector training data.

4.1.2. Filtering Out Data Points with No Salient Objects Match or Not

When formatting the data to train the rejector, we use the technique of salient object extraction and compare the salient objects with the objects, i.e. features, obtained from "should know" extraction. During data preparation, we noticed that some images do not have any salient objects that matches the features from "should know" extraction, and those images are not minority - 26.86% in the FP dataset and 18.67% in the FN dataset. We doubt whether those data points are able to provide value for rejector training, so we conducted an experiment to find out if this is true.

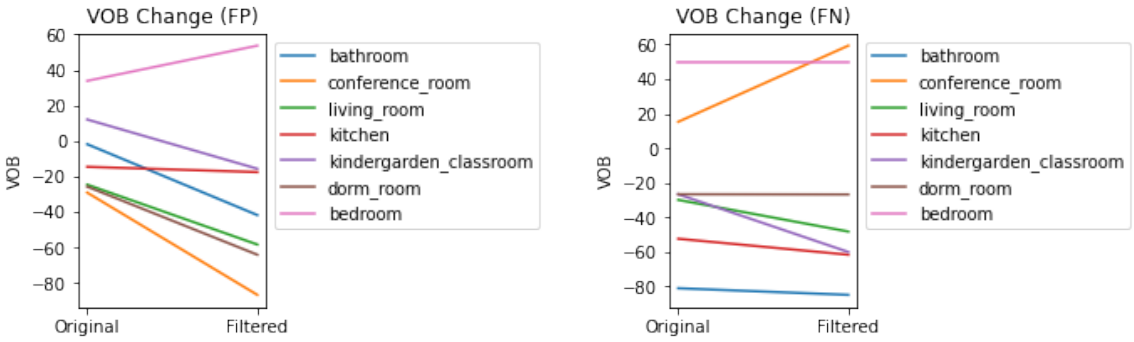

Figure 4.1: VOB Change Before and After Filtering

Figure 4.1 demonstrated the change of value over baseline (VOB, explained in section 3.3.4) before and after filtering out data points with no salient objects match. We can observe decreasing VOB is the major pattern in both FP and FN condition, which means filtering does not improve the rejector performance for most classes. This observation makes sense because the rejector can learn to reject predictions when the pretrained model doesn't highlight abt relevant features. Furthermore, more data points may also help with rejector training.

In conclusion, the experiment showed we should not filter out data points with no salient objects match in rejector training.

4.2. Parameters Tuning

Multiple parameters need to be tuned along the different components of the pipeline, and they are also small design decisions to make. This section will gather those parameters and explain why they are set to a certain value. Some parameters may be already explained in previous sections, but here we will present all of them, so that it is easier for future researchers to reproduce this project or find a better way to tune them.

Following is the list of parameters to tune, categorized by the pipeline components they belong to.

- "Should Know" Extraction
  - Scene Graph confidence threshold is set to 0.5. This is selected arbitrarily, and manual inspection proved this threshold can provide satisfactory and accurate object detection.
  - Relevance score threshold from Scalpel-HS crowdsourcing task is set to 10, because the relevance scale is from 1 to 20, and 10 is the median.
- Input from a Pretrained Classifier
  - HSV color upper and lower boundary for highlighted area is set to (90, 100, 20) and (130, 255, 255). The boundaries are set manually to cover the range from orange to red. Manual inspection has proved this is useful.
  - IOU threshold in salient object extraction is set to 0.15. Setting the IOU threshold is quite tricky, because if the threshold is set too high, there will be barely any salient object match, and if it's set too low, the match will not be accurate. We manually tried the value of 0.01, 0.1, 0.15, 0.2, 0.3, 0.5, and we inspected images for each value, compared the bounding boxes from saliency map and Scene Graph object detection. As a result, 0.15 is what is found to be the optimal value.

Within the scope of this project, the parameters are often tuned based on manual inspection of the results, and there should be a more systematic way to conduct the tuning, such as selecting the value that optimizes the final value of the rejector. However, this project is just a starting point to build a feature-based rejector, and future research is needed to tune the parameters in a way that maximizes the rejector performance.

5

Experiments and Results

This chapter will present the experimental setting in detail and evaluate the performance of our proposed rejector, SceneRejector, against the baseline and other rejectors. Further analysis will also be conducted to explain the behaviors of SceneRejector, because we believe that the performance may not be perfect as this study is simply a first step to build a feature-based, but it is important to understand why SceneRejector behaves in a certain way and in what scenario SceneRejector is useful.

5.1. Experimental Setting

5.1.1. Dataset

The dataset used is a subset of *PLACES*, which is created by [40], containing 60000 training and 1000 testing images with 9 classes equally distributed. [40] re-sampled the original training dataset to manually inject biases and thus create unknown unknowns. There are two types of bias: False Positive (FP) and False Negative (FN). FP is created by removing some objects from the training images for all classes except for the classes of interest, so that the classifier will learn to strongly associate those objects with the class of interest. On the other hand, FN is created by removing some objects from the training data of the class of interest. Table 5.1 summarizes all the objects and classes involved in unknown unknown creation. Note that the biased data set is only used for training the image classifier (more detail in section 5.1.3), and within the scope of this project we do not modify the pretrained classifier.

The data used in this project is sampled from the unbiased *PLACES* dataset. For each condition of FP and FN, there are 225 correctly classified images and 225 wrongly classified images. 225 images are selected because the wrong prediction (WP) set in FP condition is the same one used in crowdsourcing tasks in "should know" extraction, which is directly from [40] and has a size of 225. To make sure to have a balanced training set for SceneRejector, we randomly sampled the same number of correctly predicted (CP) images. As for the FN condition, both WP and CP are randomly sampled, and we sampled the same number of images to make the two conditions comparable. The WP CP distribution for each class in the two conditions is shown in figure 5.1.

Type	Class of Interest	Object
FP	kindergarden_classroom	person
	bedroom	bed
	conference_room	chair
FN	kitchen	oven
	bathroom	sink
	dining_room	wine glass
	living_room	couch/sofa
	conference_room	woman at table
	kindergarden_classroom	boy wearing shirt

Table 5.1: Manually Injected Unknown Unknowns [40]

Furthermore, a validation set of 300 images is also constructed to calculate the optimal threshold, which will be explained later in section 5.1.5. The 300 images are randomly sampled and do not overlap with the existing rejector training set.

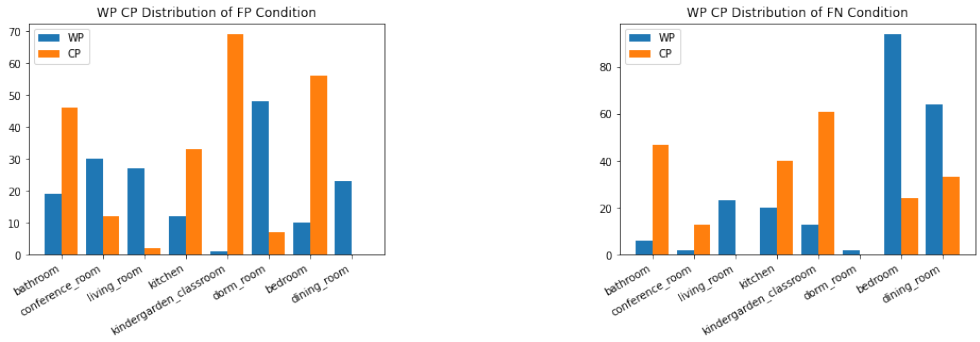

Figure 5.1: WP CP Distribution

5.1.2. Task

The task of the pretrained model is to classify the type of room given a scene image. We turned the original multi-classification task into a binary task for the eight classes, for the sake of simplicity. The eight classes include living room, conference room, dorm room, kindergarten classroom, bedroom, bathroom, kitchen, and dining room.

5.1.3. Pretrained Model

The pretrained model used for this pipeline is a state-of-the-art convolution neural network, ResNet [19], which has shown prominent performance in different types of image classification tasks. The weight of ResNet is used from [40], which is trained for 50 epochs on a biased training data. The biases come from manually injected unknown unknowns, which is explained in the previous section 5.1.1.

SmoothGrad [42] is attached to the pretrained model to generate the saliency map and visualize the pixels that have important contribution when making a prediction. The technique of salient object extraction is then applied to the saliency map, forming the input to feed to SceneRejector later in the pipeline.

As for its input to confidence-based rejectors, it is important to note that because we turned the original multi-classification task to a binary classification task

5.1.4. SceneRejector

Decision tree is used as the architecture of SceneRejector due to the fast training and easy interpretability. A SceneRejector is trained for each class in each condition, resulting in 14 SceneRejectors in total. The performance of each SceneRejector is evaluated separately. As for training, in each condition, the 550 images (225 CP and 225 WP) are randomly shuffled, and 80% of them constitute the train set, and the rest of them is used for test set. The input to train SceneRejector is the features extracted with "should know" extraction and salient object extraction, and the output is the decision of reject or accept,

represented by whether the prediction matches the ground truth. During training, accuracy is the metric to optimize. The results shown in the next section uses the data of the test set.

5.1.5. Experiment Conditions

The performance of SceneRejector will be evaluated against the following conditions, and value of each condition will be plotted over k, which is the ratio of the wrong prediction value to the correct prediction value.

Baseline
The baseline condition is simply the pretrained classifier alone, and there is no rejector attached to it.

SceneRejector
The SceneRejector condition is where the SceneRejector is attached to the pretrained classifier.

Theoretical Threshold
This condition is where a confidence-based rejector is used. For each k, we can calculate the theoretical threshold τ with the following equation. Refer to section 2.5 to see how the formula is derived. Note that the theoretical threshold is valid only when the pretrained model has perfect calibration, which means $ECE = 0$.

$$\tau = \frac{k}{k+1}$$

Theoretical Threshold + SceneRejector
This condition is where we combine the confidence-based theoretical threshold rejector with SceneRejector. In other words, the prediction from the pretrained model goes through the two abovementioned rejectors. The order to apply the rejectors does not matter, because the results will be the same.

Optimal Threshold
This condition also uses a confidence-based rejector, but the confidence threshold is set by finding an empirical threshold that optimizes value in a validation set, the construction of which is described in section 5.1.1.

Optimal Threshold + SceneRejector
This condition is where we combine the confidence-based optimal threshold rejector with SceneRejector. It is essentially applying two rejectors to the pretrained classifier, and the order does not matter.

5.2. Results

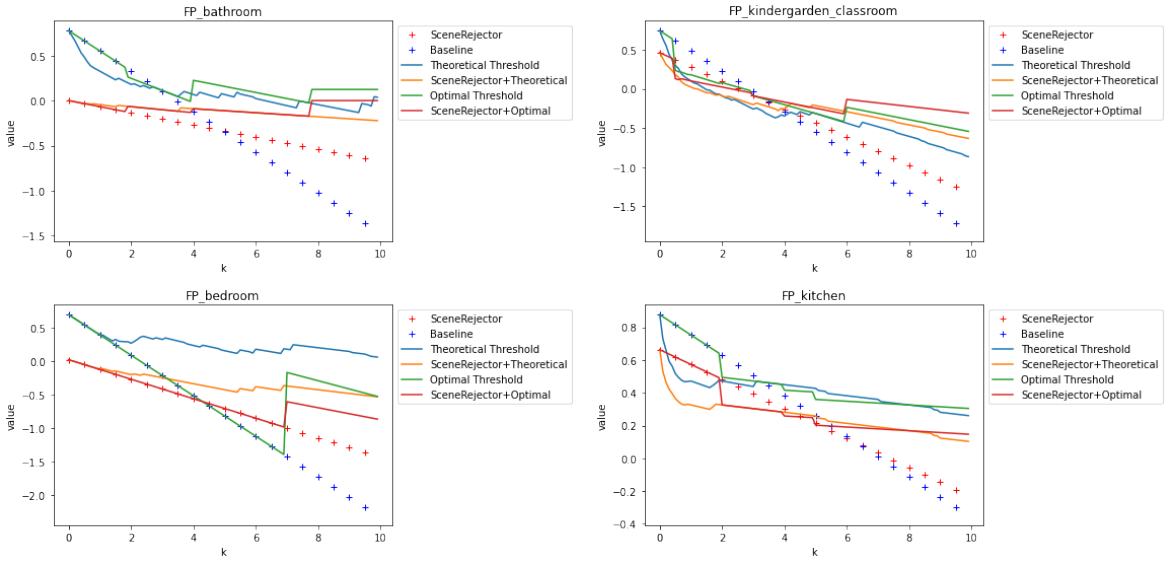

Figure 5.2: FP Results

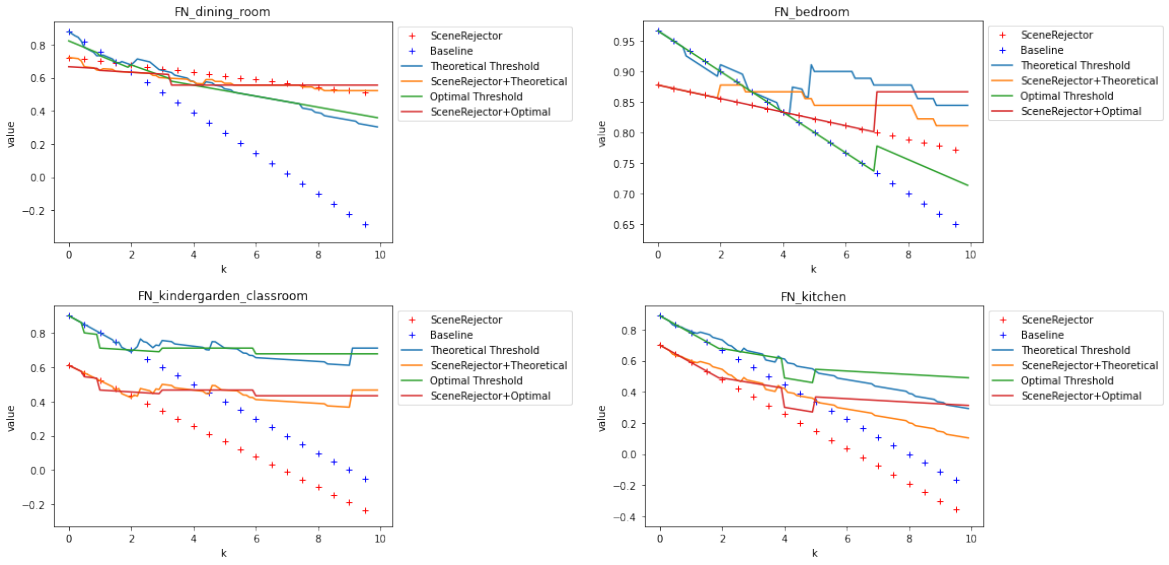

Figure 5.3: FN Results

While there are eight classes in total, here only present the results of four handpicked representative classes for each condition, and the complete results can be found in appendix A. For FP condition, we present the results of bathroom, kindergarten classroom, bedroom, and kitchen. For FN condition, we present the results of dining room, bedroom, kindergarten classroom, and kitchen.

As is described in section 5.1.5, there are 6 conditions. The 6 conditions can be sorted into 3 groups: 1) baseline and SceneRejector, 2) Theoretical Threshold and SceneRejector+Theoretical Threshold, 3) Optimal Threshold and SceneRejector+Optimal Threshold.

Comparing from an intra-group perspective, group 1 demonstrates that the slope of SceneRejector value is always less steep than the baseline value, and this is true for both conditions. In other words, as k increases and the penalty of a wrong prediction becomes larger, the decrease in SceneRejector value is not as much as in baseline value. Therefore, although in some cases SceneRejctor value is smaller than baseline value for all k ranging from 0 to 10 (e.g. in figure 5.3), an intersection point between SceneRejector and baseline can still be expected for greater k. That is to say, in use cases where a wrong prediction can induce large penalty, SceneRejector creates greater value than the baseline. Group 2 only stands when the pretrained classifier has perfect calibration. The ECE score of the pretrained classifier is quite small for all classes, so we can still evaluate the rejector performance with the theoretical threshold. From figure 5.2 and 5.3, we can observe that it is almost always true that the addition of SceneRejector does not improve the value of the confidence-based rejctor with theoretical threshold, except for a few outliers (kindergarten classroom in FP and dining room in FN). Recall that SceneRejector is built to filter out unknown unknowns, high-confidence errors. Therefore, this observation makes sense because a feature-based rejector is not necessary at all when the pretrained classifier is perfectly calibrated. Comparison with group 3 also shows the same pattern - in most cases, adding SceneRejector does not improve the value of the confidence-based rejector with optimal threshold. The ECE score of the pretrained classifier is quite small due to the conversion to binary classification, as the original ECE score of the pretrained multi-class classifier is 0.3959 for FP and 0.3797 for FN, but the ECE score of the modified binary classifier is mostly below 0.1 (details in appendix A). Therefore, the pretrained classifier is already well calibrated, so the addition of SceneRejector cannot improve the performance of a confidence-based classifier. Likewise, we can deduce that SceneRejector can contribute to a better value if the ECE score of the pretrained classifier is high. The relation between the value brought by SceneRejector and ECE score will be further discussed in section 5.3.

We can evaluate the rejector performance with an inter-group view as well. Whether it's theoretical threshold or optimal threshold, the confidence-based rejector is almost always no worse than baseline in terms of value, except for a minority of cases where the penalty of wrong prediction is small. Similarly, adding an additional confidence-based rejector is able to improve the value of SceneRejector as well. Comparing the last four conditions against baseline, we can observe that they can all significantly improve the value of baseline and thus creating a safer and more reliable machine learning pipeline.

Since the performance of group 2 is only valid when the pretrained model has perfect calibration, we can omit it and conclude group 3 gives the best performance, especially when k becomes greater. The best rejector is dependent on how well the pretrained model is calibrated, and in the presented results, the confidence-based rejector with optimal threshold gives the best performance in 5 cases and ScneRejector+Optiamal Threshold performs the best in 3 cases.

In summary, following observations are made from the value evaluation in figure 5.2 and 5.3.

- Compared with baseline, SceneRejector creates better value as the penalty of a wrong prediction gets greater.
- The addition of SceneRejector to a confidence-based rejector (with theoretical threshold or optimal threshold) does not improve value, because the pretrained classifier is already well calibrated.
- The confidence-based rejectors can significantly improve value compared with baseline.
- The addition of confidence-based rejector can improve the value of SceneRejector.
- All the experimental conditions can significantly improve the value of baseline, especially when the penalty of a wrong prediction becomes greater. The best rejector is in group 3, and the presented results shows confidence-based rejector with optimal threshold gives the best performance, due to the good calibration of the pretrained classifier.

5.3. Analysis

Despite the significant value increase that SceneRejector brings to the baseline, results from the previous do not show a recognizable evidence that the feature-based SceneRejector performs better than other confidence-based rejectors, which contradicts our assumption that feature-based rejectors should perform better than confidence-based rejectors in case of unknown unknowns.

First of all, we have to acknowledge the limitations of the experiments: 1) the dataset we use is quite small, which is 550 images for each condition; 2) more importantly, the conversion to binary classification may canceled the effect of the manually injected unknown unknowns, as the pretrained classifier itself has a high ECE score but it has quite good calibration for each class if we convert it to a binary classifier. For example, a "kitchen" image is misclassified as "kindergarden_classroom" in FP condition because there is a person in the "kitchen" image. When we convert it to binary classification, the prediction is only wrong if the class is "kitchen" or "kindergarden_classroom". In other words, the prediction is correct for the classifier of "bedroom" class, because either "kitchen" or "kindergarden_classroom" belongs to the "non-bedroom" class, and therefore a misclassification of "kitchen" for "kindergarden_classroom" does not matter in this case. The same situation can occur in the FN condition as well.

In spite that the desired results are not observed due to several limitations, we can still observe some cases where the performance of SceneRejector exceeds that of a confidence-based rejector, for example, "dining_room" SceneRejector in FN condition. What matters is how we can explain such behaviors and find out the factors that can lead to a better performance, so that the insight might shed some light on in what situation it is suitable to use SceneRejector and if is there anything can be done to improve SceneRejector value when training rejector. Therefore, a matrix of Pearson correlation coefficient between multiple variables is created in figure 5.4, and table 5.2 explains what each variable means. The variables can be classed in two groups: the first 4 variables describe the quality of the pretrained model and rejector training set, which can be viewed as independent variables, and the rest describe the resulting behavior of SceneRejector, which can be seen as dependent variables. The motivation to select those variables will be explained in the following sections. Furthermore, four VOB metrics are used to describe value, the first three can measure the value brought by the addition of SceneRejector in different scenarios, and VOB4 compares the value of SceneRejector and the value of a confidence-based rejector.

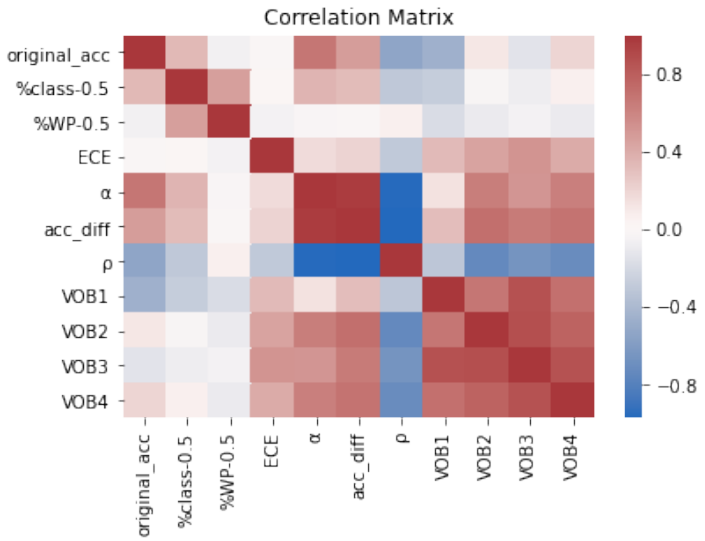

Figure 5.4: Variable Correlation Matrix

Variable	Meaning
original_acc	The accuracy of the pretrained classifier
%class-0.5	The absolute value of the difference between the percentage of samples of the class of interest and 0.5, the percentage in a balanced binary rejector training dataset
%WP-0.5	Within the sample set of the class of interest, the absolute value of the difference between the percentage of WP samples and 0.5, the percentage of WP in a dataset with a balanced WP CP distribution
ECE	The ECE score of the pretrained classifier
α	The accuracy of the accepted set given by SceneRejector
acc_diff	The difference between α and original_acc
ρ	The rejection rate of SceneRejecor
VOB1	$\int_k (value(SceneRejector) - value(Baseline))/10$
VOB2	$\int_k (value(SceneRejector + Theoretical) - value(TheoreticalThreshold))/10$
VOB3	$\int_k (value(SceneRejector + Optimal) - value(OptimalThreshold))/10$
VOB4	$\int_k (value(SceneRejector) - value(OptimalThreshold))/10$

Table 5.2: Meaning of Variables

7 data points are used from the FP condition, and the class "dining_room" is not used because it does not have CP in rejector training class. 6 data points are used from the FN condition, "living_room" and "dorm_room" are not used for the same reason. The correlation matrix is calculated with the 13 data points, and we combine the two conditions for analysis because the type of unknown unknown does not influence the performance of the rejector, as the rejector targets at unknown unknowns regardless of the type. Furthermore, more data points can lead to a more comprehensive understanding of the rejector behavior.

When interpreting the correlation coefficient, we follow a conventional approach [38], shown in the following table.

Absolute Value	Interpretation
0.00 - 0.09	Negligible correlation
0.10 - 0.39	Weak correlation
0.49 - 0.69	Moderate correlation
0.70 - 0.89	Strong correlation
0.90 - 1.00	Very strong correlation

Table 5.3: Approach to interpret Pearson correlation coefficient

Overall, figure 5.4 presents a quite clear pattern. Different ways to measure the value brought by SceneRejector are strongly correlated, so we can collectively see VOB1, VOB2, VOB3, and VOB4 altogether when talking about value. The following sections will introduce some important observed correlations between the variables and their possible implications.

5.3.1. α and acc_diff - their influence on value, and factors that affect them

In the formula of value, α stands for the accuracy of the rejector's accepted samples. Intuitively, the greater α, or the difference between α and original accuracy is, the better the value brought by the rejector should be.

As is shown in figure 5.4, α is moderately and positively correlated with VOB2 ($r = 0.6380$), VOB2 ($r = 0.5149$), and VOB4 ($r = 0.6376$). In the meantime, it is weakly and correlated with VOB1 ($r = 0.1424$), which we believe is an outlier. The variable of acc_diff can better capture the relation, because acc_diff shows the accuracy improvement after applying the rejector. acc_diff has a moderate or strong positive correlation with VOB2 ($r = 0.7220$) and VOB3 ($r = 0.6579$), VOB4 ($r = 0.6941$), and VOB1 ($r = 0.3178$) is an outlier again in this case.

The two observations reveals that the accuracy improvement, acc_diff, is correlated with the value brought by SceneRejector, regardless of the baseline that SceneRejector is compared against. Figure 5.5 can further support this.

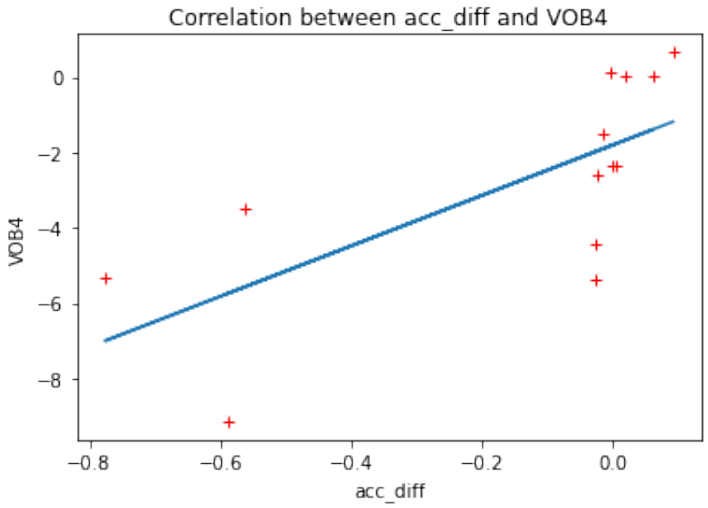

Figure 5.5: Correlation between acc_diff and VOB4

Since acc_diff influences the rejector value to a large extent, it is natural to ask what factors of the rejector training data can affect the accuracy improvement? One independent variable is found to be likely correlated. The accuracy of the pretrained classifier has a moderate correlation with α ($r = 0.6767$) and a weak correlation with acc_diff ($r = 0.4819$). If the accuracy of the pretrained classifier is high, then it is likely that a functioning rejector will not include more wrong predictions in the accepted set than the original set. Nonetheless, the correlation between original accuracy and acc_diff seems to be no better than chance. For example, if the dataset used in one condition contains more unknown unknowns than the other, which is the type of error that SceneRejector specializes in filtering, then the accuracy improvement will be more obvious in that condition, despite the two conditions may have the same original accuracy. Further research is needed for confirmation.

In summary, first, acc_diff has a strong positive correlation with the value brought by SceneRejector. Second, the accuracy of the pretrained classifier is correlated with α, but acc_diff is likely not.

5.3.2. ρ - its influence on value

A very strong negative correlation can be observed between α and ρ ($r = -0.9648$), which means if SceneRejector rejects less, the accuracy of the accepted set will be higher. Given that the original accuracy of the pretrained classifier is quite high for all the classes, it is reasonable that the bigger the rejection rate is, the more possible that correct predictions end up in the rejected set. After all,

a functional rejector should not reject a lot of the predictions if the original accuracy is already close to optimal. Apart from α, ρ is also found to be very strongly correlated with acc_diff ($r = -0.9736$). Furthermore, there is a strong or moderate correlation between ρ and value as well ($r = -0.3124$ for VOB1, $r = -0.7401$ for VOB2, $r = -0.6632$ for VOB3, $r = -0.7148$ for VOB4). It is more clearly demonstrated in figure 5.6.

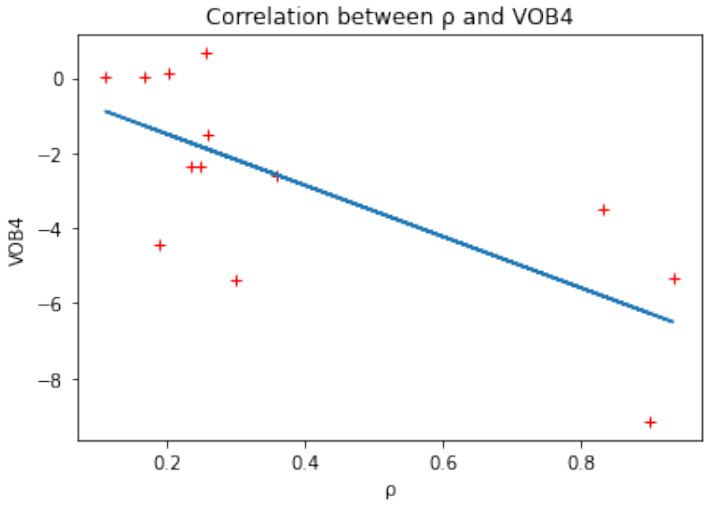

Figure 5.6: Correlation between ECE and VOB4

In summary, rejection rate ρ presents a strong negative correlation with value. Nonetheless, we need to cautiously view this correlated relation, because we do not need to apply a rejector in the first place if all we want is to minimize rejection rate. In other words, ρ is not some parameter we should tune to achieve better performance, but rather a resulting phenomenon of the quality of the pretrained classifier: if the original accuracy is high, a functional rejector will have small ρ; and if the original accuracy is low, then a functional rejector is very likely to have a greater rejection rate.

5.3.3. %class and %WP

The equally distributed dataset is now unbalanced because the original task is converted to a binary classification task, and there are multiple consequences - the ECE and accuracy of the original pretrained classifier is sure to change, but does this have any effect on training SceneRejector?

"%class-0.5" calculates the distance from a balanced class distribution for the class of interest, e.g. if there are 10% samples of the "kitchen" class when training SceneRejector for "kitchen", then the distance is 0.4. "%WP-0.5" calculates the distance from a balanced WP CP distribution for the class of interest. Continue on the previous example, within the 10% of "kitchen" samples, if 30% of the predictions are wrong, then the distance is 0.2. Figure 5.4 does not show a strong correlation between value and the two metrics that measure the distance from a balanced distribution. The correlation coefficient is below 0.1 for VOB2, VOB3, and VOB4. VOB1 has a weak correlation with "%class-0.5" ($r = -0.2763$) and "%WP-0.5" ($r = -0.1825$).

In summary, the observation shows the unbalanced dataset does not influence the performance of SceneRejector.

5.3.4. Pretrained classifier calibration - suitable use case for SceneRejector

Compared with confidence-based rejectors, the performance of feature-based rejectors is not restricted to the calibration of the pretrained classifier. Therefore, it is easy to make an assumption that feature-based rejectors will outperform confidence-based rejectors, if the pretrained classifier is not well calibrated, i.e. it has a high ECE score.

We can observe a positive correlation between ECE and the four value metrics in figure 5.4, and they show a weak or moderate correlation ($r = 0.3315$ for VOB1, $r = 0.4454$ for VOB2, $r = 0.5243$ for VOB3, $r = 0.4075$ for VOB4). The correlation is not as strong as expected, and there are two possible reasons. First, SceneRejector is not very well trained given the limited training dataset. Second, the

range of ECE from the pretrained classifiers is quite small, as shown in figure 5.7, and more data points are needed to come up with a comprehensive conclusion.

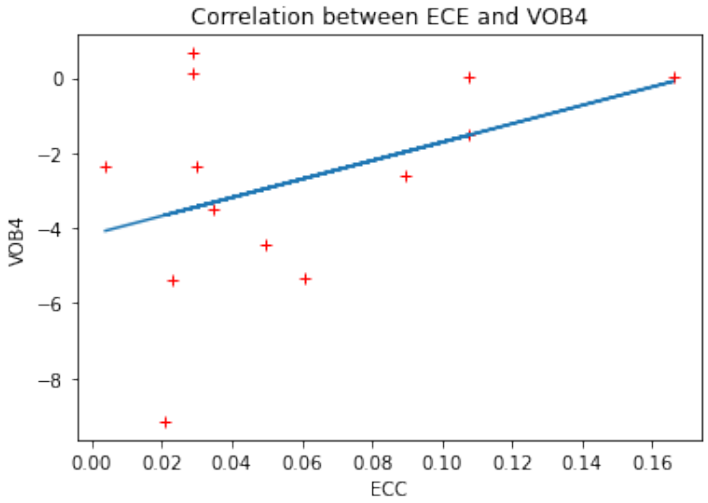

Figure 5.7: Correlation between ECE and VOB4

In summary, a positive correlation exists between ECE and the value metrics; however, the correlation is not as strong as expected due to experiment limitations, and more research is required to explore the relation between ECE and the value of feature-based rejectors.

5.4. Observation From the Rejected Set

Apart from value analysis, we also qualitatively measure the performance of SceneRejector by taking a look at the samples in its rejected set.

Condition	Class	Accuracy of Rejected Set
FP	bathroom	0.4458
	kindergarden_classroom	0.2813
	bedroom	0.4595
	kitchen	0.1364
FN	dining_room	0.3478
	bedroom	0.2000
	kindergarden_classroom	0.3704
	kitchen	0.2353

Table 5.4: Accuracy of Rejected Set in Different Cases

Table 5.4 shows the prediction accuracy of the rejected set, and we can observe that the accuracy is significantly lower than the original accuracy, meaning SceneRejector is functional in rejecting prediction errors. Furthermore, as previously mentioned, feature-based rejector should be able to reject high-confidence errors, so figure 5.8 is plotted to show the confidence distribution of the rejected errors. The majority of prediction confidence is clustered towards the right, and there are a lot of prediction errors with confidence greater than 0.8 in most cases. Therefore, we can also conclude that SceneRejector is successful in rejecting high-confidence errors, i.e. unknown unknowns. Last but not least, figure 5.9 shows some concrete examples of unknown unknowns rejected by SceneRejector. The ground truth of the two images is both "kitchen", and the pretrained classifier classifies them as "dining_room", with the prediction confidence of 0.79 and 0.83. As is shown in the saliency map, the pretrained classifier highlighted objects such as chairs and tables, which is not among the relevant and sufficient features of "kitchen", and therefore SceneRejector is able to filter these errors.

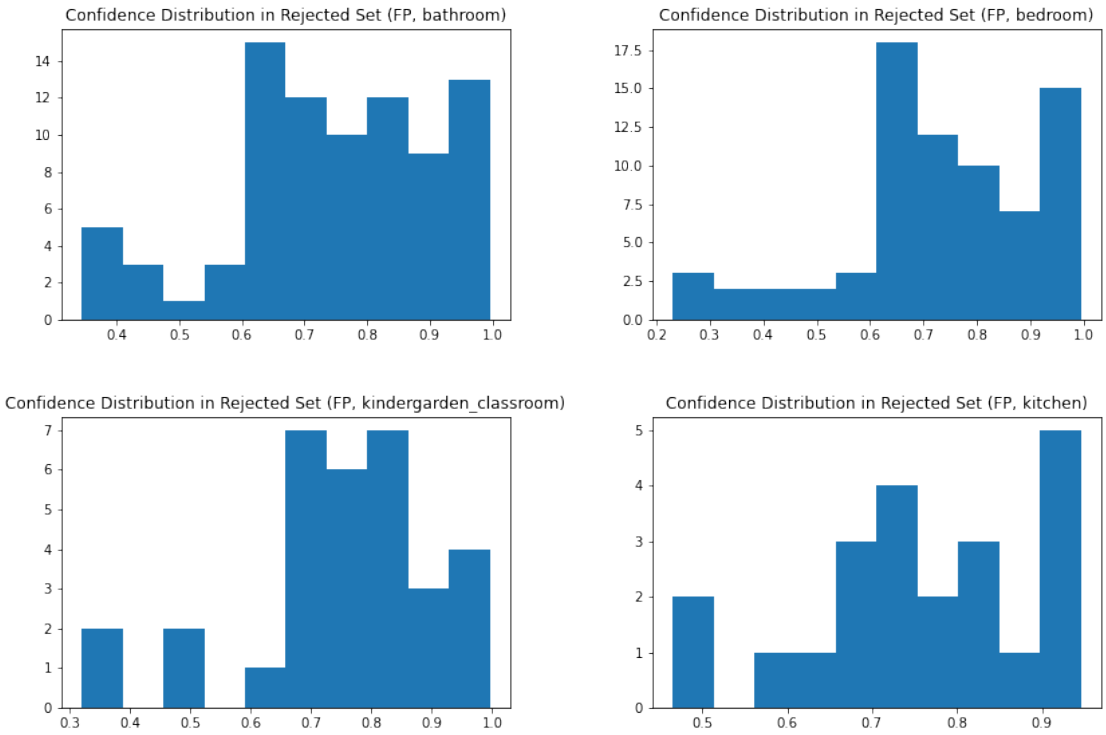

Figure 5.8: Confidence Distribution of Rejected Set

Figure 5.9: Saliency Map of Rejected Images

In summary, the observation from the rejected set confirms: 1) SceneRejector is a functional rejector; 2) SceneRejector is specialized in and able to reject high-confidence errors, i.e. unknown unknowns; 3) SceneRejector wields features used by pretrained classifier to reject predictions.

6

Discussion and Conclusion

6.1. Discussion

While feature-based rejector is proved to be a working concept and SceneRejector does improve the value of the pretrained classifier in some use cases, its performance still has a lot of space for improvement. There are several limitations in this project, occurring in pipeline implementation, analysis, etc. This section will discuss the limitations of this study and in what way the result is influenced.

First, Scene Graph [45] is an important part of salient object extraction by acting as an object detection model. However, the Scene Graph model we used in this project is already pretrained and it can detect more classes than needed. The detection of those necessary objects, e.g. the features derived from "should know" extraction, may not have a very good accuracy because there exist many other possibly confounding classes. Furthermore, sometimes the vocabulary used by Scene Graph is different from what is used in the features from "should know" extraction, and as a result, it may induce a mismatch between the detected object and feature used by the rejector during salient object extraction. For example, an object might be detected as "countertop" by the object detection model, but the "should know" feature is "counter". Although one highlighted area in the saliency map matches the object detection bounding box of "countertop", this match will not be registered in the training data of the rejector. This project has already manually pointed different synonyms to a standard name for some features, but such manual manipulation is not able to cover all cases. Therefore, it might be beneficial to include the training of an object detection model in the rejector training. In other words, an object detection model could be trained only with the objects or features derived from "should know" extraction, and thus the issue of vocabulary mismatch can be solved and the model can focus better on learning the necessary classes. Nonetheless, such object detection model will only be used in salient object extraction and not "should know" extraction, because the necessary features are still not known at the stage to use object detection in "should know" extraction.

Second, in the aggregation stage of "should know" extraction, the final features are a combination of two sources (see figure 3.1) and then manually verified. However, the manual verification is only done by one person and possibly has an individual bias. Therefore, an optimal way to select the relevant and

sufficient features for each class would be to design another crowdsourcing task and to involve human in the loop again, so that the selected features are better validated. In some use scenarios that require domain expertise, the features that should be used by a machine learning model can be determined by the human experts or stakeholders as well.

Third, during rejector training, the reject/accept label used in training set is decided by the match between the predicted class and ground truth class, e.g., the rejector should accept a prediction if the predicted class is the same as the ground truth. Nevertheless, this should not always be the case, because a correct prediction may also stem from a wrong reason. For instance, if only a chair is highlighted by the pretrained model in a "kitchen" image which is correctly classified, the rejector should still reject it and the label used in training set should be a reject.

Fourth, SceneRejector is essentially a decision tree in this project, and we choose this model because it is fast to train and interpretable. The training of the decision tree optimizes accuracy, but it is value that we measure in the end. Therefore, the performance of SceneRejector might improve if it optimized with value.

Fifth, the parameters in the whole pipeline (in chapter 4) are often tuned by manual inspecting the results, but this is a sub-optimal approach. A more systematic way could be applied to tune the parameters in a way where value is maximized.

Sixth, though simplified the use case and created more rejectors to test, the conversion from multi-classification to binary classification does eliminate the effect of the manually injected unknown unknowns, as explained in the previous analysis section. As a result, a clear effect of SceneRejector to filter the injected unknown unknowns is yet to be seen. Furthermore, the ECE score of the pretrained classifier become smaller and about same for all classes after the conversion, which might play a part in preventing us to see a clear correlation between classifier calibration and value brought by SceneRejector.

Last but not least, this study only experimented with one task, which is scene classification. Therefore, although the concept of feature-based rejectors is proved to work, more experiments on different tasks are needed to find if the results can be generalized and if additional adjustments to the pipeline is needed given a new task. In this way, the analysis of SceneRejector behavior will have more data to support, and thus the analysis can be further validated.

6.2. Conclusion

For a safer and more responsible machine learning pipeline, confidence-based rejectors are commonly used to safeguard machine decisions. However, rejectors of this type cannot filter out high-confidence errors, i.e. unknown unknowns, and it heavily relies on the calibration of the classifier as well. This study proposed and implemented a feature-based rejector, SceneRejector, and embedded it in a selective classifier. Such rejector is pluggable to any pretrained classifier as long as the task and the classes stay the same. There is a training pipeline to train the rejector and a testing pipeline to apply the rejector. The training pipeline consists of three components: 1) input from a pretrained classifier, which means the features that the classifier uses while making a prediction; 2) "should know" extraction, which involves humans in the loop and acquires the features that the classifier should use to make a prediction; 3) rejector training, where we train a decision tree with the input and output are prepared from the previous two parts and the ground truth. The testing pipeline consists of two parts, the first of which is the same as the training pipeline, and the second of which is simply applying the trained rejector with input from the first part. Several design decisions are made based on experiments. It is important to note that human knowledge can not only be used in "should know" extraction, but also in defining the created value, even though this study did not involve humans in the latter case. Finally, more experiments and analysis proved that 1) SceneRejector is a working rejector and validates the concept of feature-based rejectors; 2) SceneRejector creates better value as the penalty of a wrong prediction increases; 3) the value of SceneRejector is positively correlated with the accuracy improvement; 4) there exists a positive correlation between the ECE score of the pretrained classifier and the value of SceneRejector, meaning SceneRejector brings better value when the pretrained classifier is not well calibrated, but more research is needed for a comprehensive conclusion; 5) The samples rejected by SceneRejector consists of quite a few high-confidence errors. To conclude, despite the limitations, this study managed to implement a feature-based rejector, the performance of which is close to the assumptions and hypothesis made in the beginning.

6.3. Future Works

With the limitations mentioned in section 6.1, further research is needed to discover possible better ways to implement a feature-based rejector and explain the current behavior of SceneRejector.

First, further research is needed to see how a feature-based rejector can adapt in other environments. Since the conversion from multi-classification to binary classification deducts the effect of injected unknown unknowns, a new SceneRejector should be built for a multi-class pretrained classifier, and such classifier is also more commonly used in real world applications. Different design decision might be made, e.g. including class label when constructing rejector training set. If SceneRejector can adapt to a multi-class scenario, then we are able to see whether its performance is even better in cases where the pretrained classifier is poorly calibrated. Furthermore, we can also better observe whether SceneRejector can reject the unknown unknowns created in the two conditions, FP and FN. Apart from that, it would be interesting to apply SceneRejector in other computer vision tasks as well, especially for high-stake use cases, e.g., medical image classification. In such use cases, the cost of a high-confidence error could be quite severe, and thus feature-based rejectors will contribute more to safeguarding machine decisions. Different approaches to build the rejector and construct the "should know" features might be required; nonetheless, this can still reveal whether the discoveries from the scene classification task will stand in a new task.

What's more, it is worth exploring other possible better ways to build a feature-based rejector. In this project we used the decision tree model to build SceneRejector, but there might be better models to use, and it would be interesting to see the performance difference. Maybe a simple rule-based rejector is enough for some use cases, and other cases require a more complex model for the rejector. It is important to understand what model is suitable to use in what task. In addition, the current way to tune parameters of different parts of the training pipeline is sub-optimal. One better way to come up with a systematic method and tune the parameters based on some resulting metrics, e.g. value against baseline.

While the human is already involved in the loop to extract the "should know" features, there are more components of the training pipeline that could do their job better if human input is utilized in the process. The current "should know" task only prompts crowd workers to assign relevance score to objects in the image, so the final aggregation of the relevant and sufficient feature set is manually validated. An additional crowdsourcing task might be helpful in determining the final "should know" feature set. The crowd will not rank objects in a single image, but on the scale of a whole class. Moreover, regarding assigning reject/accept labels during rejector training, we do not consider the situation where a prediction is correct for the wrong reasons. Hence, another new crowdsourcing task is necessary to prevent this. The task could be very simple: crowd workers only need to judge whether the accept/reject label to train the rejector is reasonable, given the accept/reject label and the corresponding reason in the form of saliency map. This improvement is especially meaningful for use cases that involves privacy or other ethical issues, because we can avoid machine using private or unethical information to make a decision.

On the same note, if a correct prediction is correct for the wrong reasons, we would also need to revise the formula of the value metric. Otherwise, those predictions will have better value if we do not reject them. Therefore, we should not have a single value for correct predictions, and the ones with good reasons and those with wrong reasons should be separated instead. However, it is challenging and costly to have all the the "wrongly correct predictions" labeled. Furthermore, the definition of "right reasons" can differ from case to case and does not always have a clear-cut boundary. Thus, this measure might only be valuable to use cases that involves sensitive private information that we should make sure not used by the machine. More research is needed to re-define value in the context of the above mentioned use cases.

Lastly, while it's important to interpret the classifier decisions, rejector interpretability also matters a lot. Since the rejector is a decision tree in this project, interpretation is rather easy. Extra tools are nice to have to visualize the tree and output the exact reason for acceptance or rejection to a certain prediction. However, if other machine learning models are used to build a rejector, different interpretability methods are needed.

References

[1] Diederik Aerts. “Quantum theory and human perception of the macro-world”. In: *Frontiers in Psychology* 5 (2014), p. 554.

[2] Joshua Attenberg, Panos Ipeirotis, and Foster Provost. “Beat the Machine: Challenging Humans to Find a Predictive Model’s “Unknown Unknowns””. In: *J. Data and Information Quality* 6.1 (Mar. 2015). ISSN: 1936-1955. DOI: 10.1145/2700832. URL: https://doi.org/10.1145/2700832.

[3] Agathe Balayn et al. “What do you mean? Interpreting image classification with crowdsourced concept extraction and analysis”. In: *Proceedings of the Web Conference 2021*. 2021, pp. 1937–1948.

[4] Pratiksha Benagi et al. “Feature Extraction and Classification of Heritage Image from Crowd Source”. In: *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*. 2018, pp. 1–5. DOI: 10.1109/ICCTCT.2018.8550898.

[5] Settles Burr. *Active learning*. 2012. DOI: 10.2200/s00429ed1v01y201207aim018. URL: https://cir.nii.ac.jp/crid/1370004230485356674.

[6] Peter Calhoun et al. “Random forest with acceptance–rejection trees”. In: *Computational Statistics* 35.3 (2020), pp. 983–999.

[7] Fabio Casati, Pierre-André Noël, and Jie Yang. “On the Value of ML Models”. In: (2021). DOI: 10.48550/ARXIV.2112.06775. URL: https://arxiv.org/abs/2112.06775.

[8] Justin Cheng and Michael S. Bernstein. “Flock: Hybrid Crowd-Machine Learning Classifiers”. In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work Social Computing*. CSCW ’15. Vancouver, BC, Canada: Association for Computing Machinery, 2015, pp. 600–611. ISBN: 9781450329224. DOI: 10.1145/2675133.2675214. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/2675133.2675214.

[9] Loris Copa et al. “Unbiased query-by-bagging active learning for VHR image classification”. In: *Image and Signal Processing for Remote Sensing XVI*. Vol. 7830. International Society for Optics and Photonics. 2010, 78300K.

[10] Luigi P. Cordella et al. “A method for improving classification reliability of multilayer perceptrons”. In: *IEEE transactions on neural networks* 6 5 (1995), pp. 1140–7.

[11] Jeffrey Dastin. *Amazon scraps secret AI recruiting tool that showed bias against women*. Oct. 2018. URL: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G.

[12] Claudio De Stefano, Carlo Sansone, and Mario Vento. “To reject or not to reject: that is the question-an answer in case of neural classifiers”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 30.1 (2000), pp. 84–94.

[13] Enrique Estellés-Arolas and Fernando González L. Guevara. “Towards an Integrated Crowdsourcing Definition”. In: *Journal of Information Science* 38 (Apr. 2012). DOI: 10.1177/0165551512437638.

[14] Evanthia Faliagka et al. “Application of Machine Learning Algorithms to an online Recruitment System”. In: *ICIW 2012*. 2012.

[15] Giorgio Fumera and Fabio Roli. “Support vector machines with embedded reject option”. In: *International Workshop on Support Vector Machines*. Springer. 2002, pp. 68–82.

[16] Yonatan Geifman and Ran El-Yaniv. *Selective Classification for Deep Neural Networks*. 2017. arXiv: 1705.08500 [cs.LG].

[17] Amirata Ghorbani et al. “Towards automatic concept-based explanations”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[18] Chuan Guo et al. *On Calibration of Modern Neural Networks*. 2017. arXiv: 1706.04599 [cs.LG].

[19] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385. URL: https://arxiv.org/abs/1512.03385.

[20] Martin E Hellman. “The nearest neighbor classification rule with a reject option”. In: *IEEE Transactions on Systems Science and Cybernetics* 6.3 (1970), pp. 179–185.

[21] Kilian Hendrickx et al. “Machine learning with a reject option: A survey”. In: *arXiv preprint arXiv:2107.11277* (2021).

[22] Zhengbao Jiang et al. *How Can We Know What Language Models Know?* 2020. arXiv: 1911.12543 [cs.CL].

[23] Been Kim et al. “Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)”. In: *International conference on machine learning*. PMLR. 2018, pp. 2668–2677.

[24] Roman Levin et al. “Where do Models go Wrong? Parameter-Space Saliency Maps for Explainability”. In: *arXiv preprint arXiv:2108.01335* (2021).

[25] David D Lewis and Jason Catlett. “Heterogeneous uncertainty sampling for supervised learning”. In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 148–156.

[26] David C. Logan. “Known knowns, known unknowns, unknown unknowns and the propagation of scientific enquiry”. In: *Journal of Experimental Botany* 60.3 (Mar. 2009), pp. 712–714. ISSN: 0022-0957. DOI: 10.1093/jxb/erp043. eprint: https://academic.oup.com/jxb/article-pdf/60/3/712/1237855/erp043.pdf. URL: https://doi.org/10.1093/jxb/erp043.

[27] Ana Madevska-Bogdanova and Dragan Nikolic. “A new approach of modifying SVM outputs”. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 6. IEEE. 2000, pp. 395–398.

[28] Swati Mishra and Jeffrey M Rzeszotarski. “Crowdsourcing and Evaluating Concept-driven Explanations of Machine Learning Models”. In: *Proceedings of the ACM on Human-Computer Interaction* 5.CSCW1 (2021), pp. 1–26.

[29] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. “Obtaining well calibrated probabilities using bayesian binning”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[30] Jeremy Nixon et al. “Measuring Calibration in Deep Learning.” In: *CVPR Workshops*. Vol. 2. 7. 2019.

[31] Yaniv Ovadia et al. *Can You Trust Your Model’s Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*. 2019. arXiv: 1906.02530 [stat.ML].

[32] Divyarajsinh N. Parmar and Brijesh B. Mehta. *Face Recognition Methods Applications*. 2014. arXiv: 1403.0485 [cs.CV].

[33] Gustavo Penha and Claudia Hauff. “What Does BERT Know about Books, Movies and Music? Probing BERT for Conversational Recommendation”. In: *Fourteenth ACM Conference on Recommender Systems*. RecSys ’20. Virtual Event, Brazil: Association for Computing Machinery, 2020, pp. 388–397. ISBN: 9781450375832. DOI: 10.1145/3383313.3412249. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/3383313.3412249.

[34] Fabio Petroni et al. *Language Models as Knowledge Bases?* 2019. arXiv: 1909.01066 [cs.CL].

[35] John Platt et al. “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods”. In: *Advances in large margin classifiers* 10.3 (1999), pp. 61–74.

[36] Ratnavel Rajalakshmi and Chandrabose Aravindan. “A Naive Bayes approach for URL classification with supervised feature selection and rejection framework”. In: *Computational Intelligence* 34.1 (2018), pp. 363–396.

[37] Burcu Sayin et al. “The Science of Rejection: A Research Area for Human Computation”. In: *arXiv preprint arXiv:2111.06736* (2021).

[38] Patrick Schober, Christa Boer, and Lothar A Schwarte. “Correlation coefficients: appropriate use and interpretation”. In: *Anesthesia & Analgesia* 126.5 (2018), pp. 1763–1768.

[39] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[40] Shahin Sharifi Noorian et al. “What Should You Know? A Human-In-the-Loop Approach to Unknown Unknowns Characterization in Image Recognition”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 882–892. ISBN: 9781450390965. DOI: 10.1145/3485447.3512040. URL: https://doi.org/10.1145/3485447.3512040.

[41] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[42] Daniel Smilkov et al. *SmoothGrad: removing noise by adding noise*. 2017. DOI: 10.48550/ARXIV.1706.03825. URL: https://arxiv.org/abs/1706.03825.

[43] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[44] Ryusuke Takahama et al. “AdaFlock: Adaptive Feature Discovery for Human-in-the-loop Predictive Modeling”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 32.1 (Apr. 2018). DOI: 10.1609/aaai.v32i1.11509. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11509.

[45] Kaihua Tang et al. *Unbiased Scene Graph Generation from Biased Training*. 2020. DOI: 10.48550/ARXIV.2002.11949. URL: https://arxiv.org/abs/2002.11949.

[46] Weizhe Yuan, Graham Neubig, and Pengfei Liu. *BARTScore: Evaluating Generated Text as Text Generation*. 2021. arXiv: 2106.11520 [cs.CL].

[47] Bianca Zadrozny and Charles Elkan. “Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers”. In: *Icml*. Vol. 1. Citeseer. 2001, pp. 609–616.

[48] James Y. Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. *Crowdsourcing Feature Discovery via Adaptively Chosen Comparisons*. 2015. DOI: 10.48550/ARXIV.1504.00064. URL: https://arxiv.org/abs/1504.00064.

A

Supplement

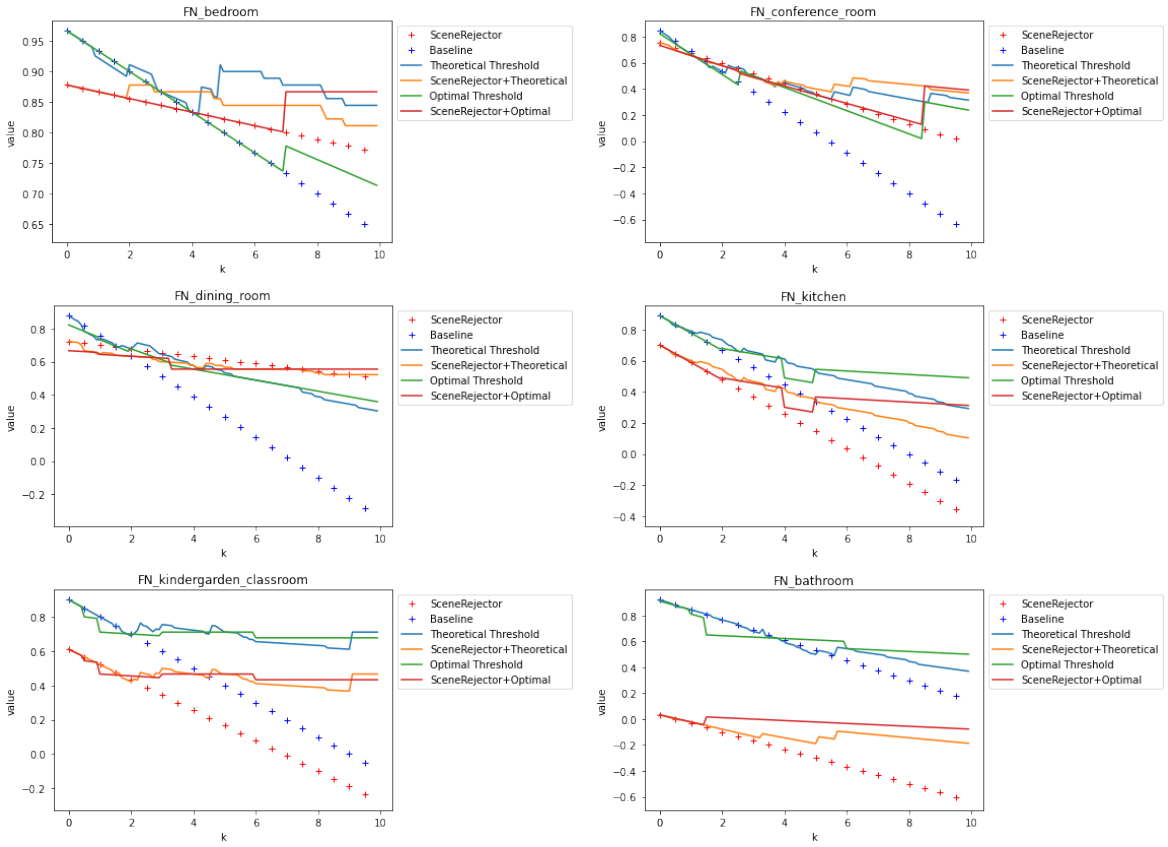

Figure A.1: FN Results (Complete)

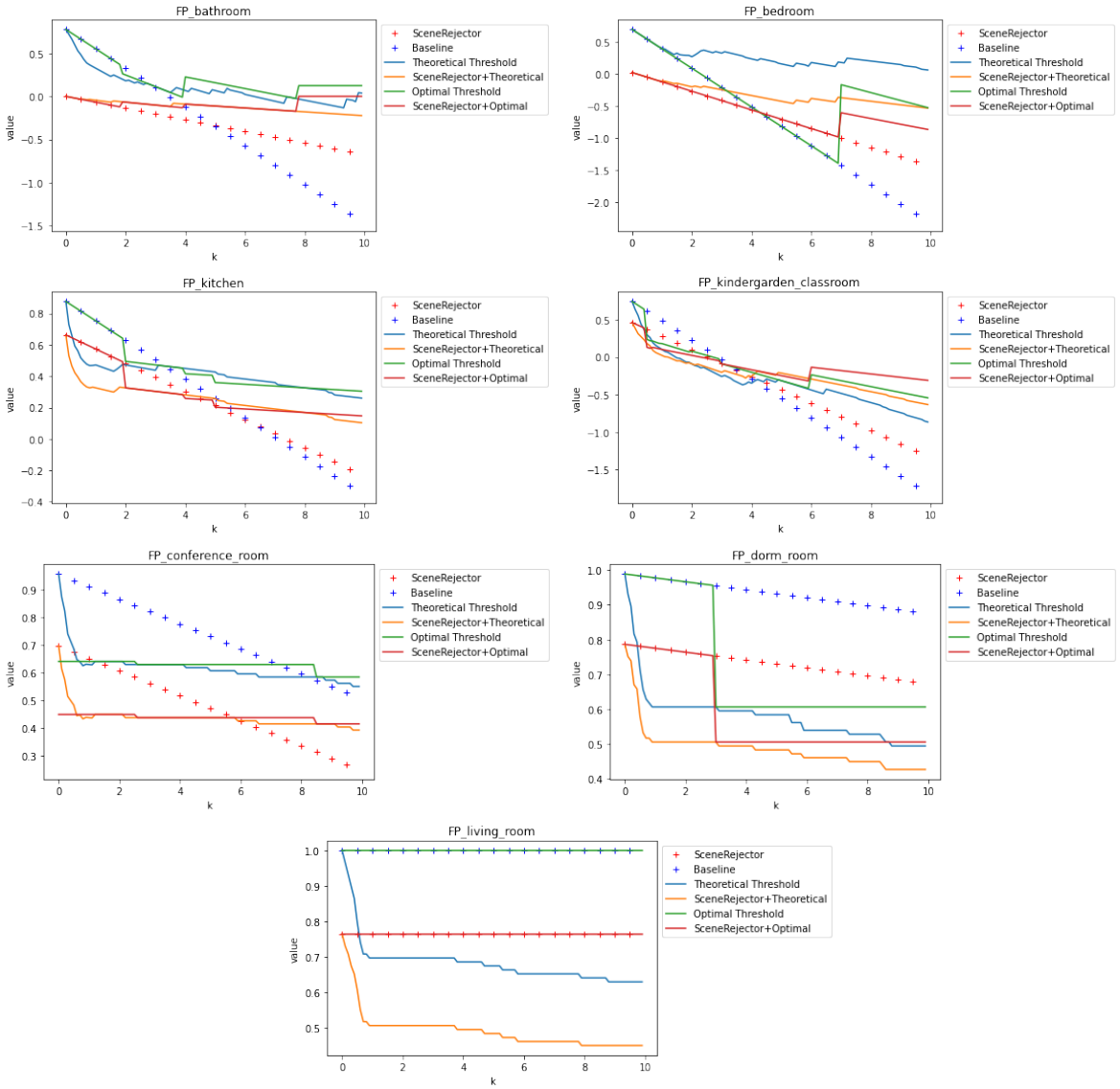

Figure A.2: FP Results (Complete)

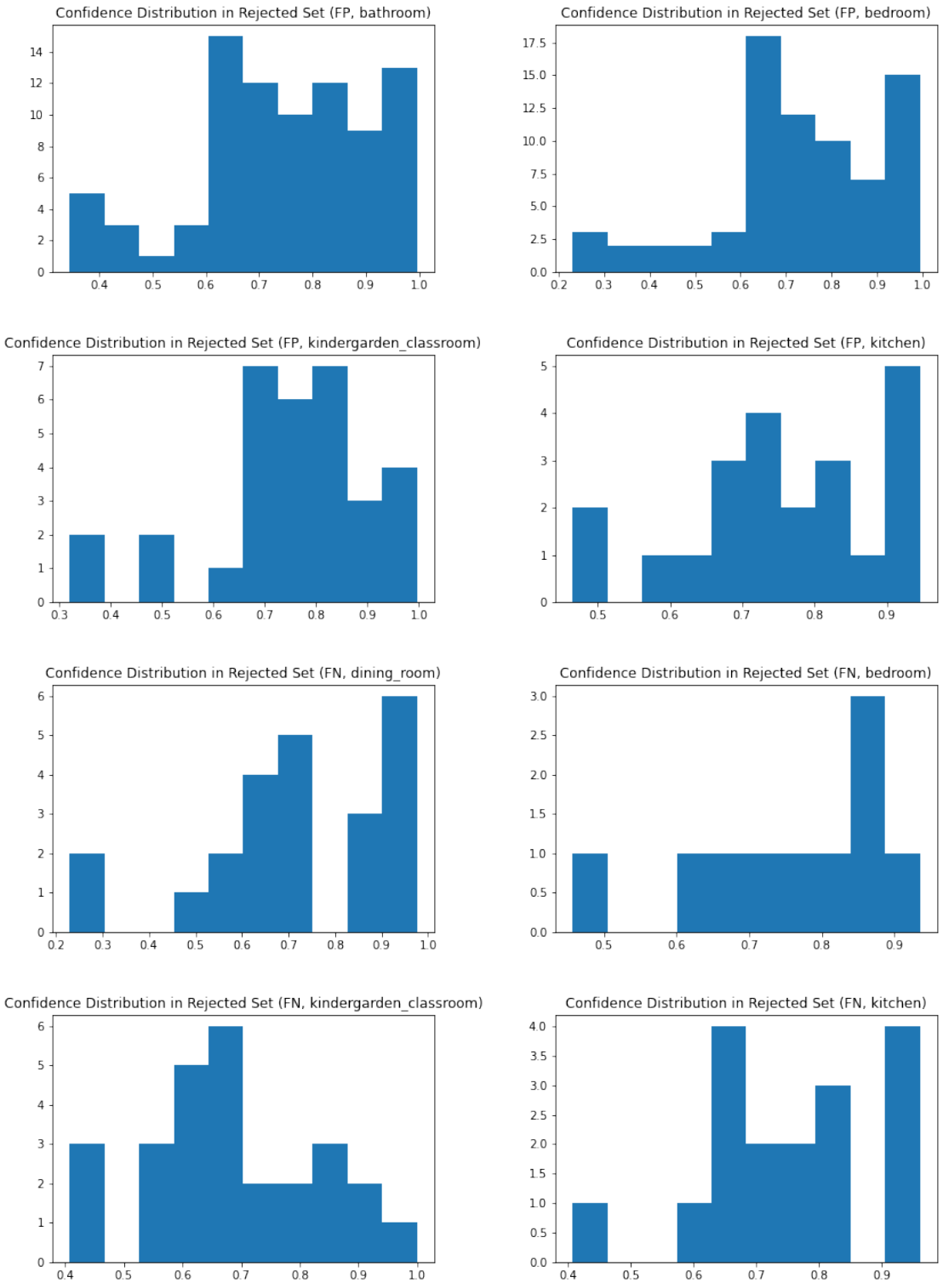

Figure A.3: Confidence Distribution of Rejected Set (Complete)