2023-08-01

Increasing the efficiency and accuracy of collective intelligence methods for image classification

Md Mahmudulla Hassan
University of Texas at El Paso

INCREASING THE EFFICIENCY AND ACCURACY OF COLLECTIVE INTELLIGENCE METHODS FOR IMAGE CLASSIFICATION

MD MAHMUDULLA HASSAN

Doctoral Program in Computer Science

APPROVED:

Olac Fuentes, Ph.D., Chair

Martine Ceberio, Ph.D.

Adolfo R. Escobedo, Ph.D.

Stephen L. Crites, Ph.D.
Dean of the Graduate School

to my son

Arish Hassan

INCREASING THE EFFICIENCY AND ACCURACY OF COLLECTIVE INTELLIGENCE METHODS FOR IMAGE CLASSIFICATION

by

MD MAHMUDULLA HASSAN

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

August 2023

Acknowledgements

The completion of this research would not have been possible without the invaluable contributions and support of numerous individuals whose dedication and guidance played a crucial role in shaping this work.

At the forefront, I owe my deepest gratitude to my advisor, Dr. Olac Fuentes. It is an often used cliché, but in this case, it is no overstatement to say that without the consistent guidance, tutelage, support, unparalleled knowledge, and encouragement of my advisor, this thesis would never have existed. I consider myself exceptionally fortunate to have had the opportunity to learn from him. I am truly grateful for the mentorship that has shaped my academic journey and personal growth.

I extend my heartfelt appreciation to Dr. Adolfo Escobedo for being invaluable to my Ph.D. committee. I am deeply grateful for his expertise and guidance throughout my research journey. His vast knowledge in the field has provided crucial insights and perspectives, enriching the quality of my work.

Dr. Martine Ceberio has been an exceptional member of my Ph.D. committee, and I am immensely appreciative of her invaluable contributions to my research. Her profound understanding of the subject matter and her rigorous approach to academic inquiry have been truly inspiring.

I also extend my heartfelt appreciation to Dr. Suman Sirimulla, whose unwavering support at the beginning of my Ph.D. journey provided me with the encouragement and freedom to explore ideas, propelling me forward in this significant research endeavor.

I am also grateful to my colleagues from the Vision and Learning Lab, who have contributed directly or indirectly to my learning process, fostering an environment of collaboration and camaraderie.

Beyond academia, the love and support of my family have been my pillars of strength. I express my heartfelt gratitude to my mother, Asma Begum, my brother, Md Jubaer, and my sister, Farhana Nasrin, for their unending support and encouragement during this journey.

I want to express my deepest and most heartfelt gratitude to my wife, Sharmin Akter, for her love and constant support throughout this journey. Her presence has been a pillar of strength during the long nights and early mornings, and her unwavering belief in me has been a source of motivation.

To my son, Arish, I am immensely grateful for the joy and inspiration he has brought into my life. His presence has made me stronger and better every day, reminding me of the importance of this endeavor and giving me the determination to persevere.

Additionally, I want to thank The University of Texas at El Paso Computer Science Department professors and staff for all their hard work and dedication, providing me with the means to complete my degree and prepare for a career as a computer scientist.

To all those mentioned and the countless others who have contributed to my academic and personal growth, I offer my sincerest thanks. Your support has been the cornerstone of this dissertation, and I am deeply grateful for the impact you have had on my life.

Abstract

Collective intelligence has emerged as a powerful methodology for annotating and classifying challenging data that pose difficulties for automated classifiers. It works by leveraging the concept of "wisdom of the crowds" which approximates a ground truth after aggregating experts' feedback and filtering out noise. However, challenges arise when certain applications, such as medical image classification, security threat detection, and financial fraud detection, demand accurate and reliable data annotation. The unreliability of experts due to inconsistent expertise and competencies, coupled with the associated cost and time-consuming judgment extraction, presents additional challenges.

Input aggregation is the process of consolidating and combining multiple individual judgments, feedback, or annotations obtained from a diverse group of experts to arrive at a single representative decision or prediction. In this dissertation, we introduce diverse deep learning techniques to enhance the accuracy of input aggregation methods and optimize task assignments among experts. We demonstrate that incorporating the outputs of an automated classifier as additional features improves traditional input aggregation methods. We also show that the accuracy of these methods can be further improved by adding meaningful image features learned by self-supervised models. The additional features reduce the requirements of collecting a large number of inputs from human labelers. We also investigate how task assignments can be optimized for groups of experts that possess varying degrees of expertise and diverse competency areas. We show that experts' competencies and samples' complexity can be modeled simultaneously and that optimization algorithms can leverage deep learning models to perform an optimal selection of experts. To train and evaluate the deep learning models, we propose a novel algorithm for generating a large dataset of synthetic X-ray images. The dataset works as a test bed for conducting comprehensive testing and validation of our proposed methodologies.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Collective intelligence refers to the shared knowledge, insights, and problem-solving capabilities that emerge from the collaboration and contributions of a group of individuals. It has emerged as a useful technique for obtaining experts' feedback and labeling unknown samples, leveraging the collective intelligence of a group of individuals to accomplish tasks or solve problems [2]. Its accuracy often surpasses automated methods, proving more effective, particularly in tasks that require domain knowledge of the subject matter. While recent advancements in deep neural networks have demonstrated significant improvements over conventional computer vision techniques in various domains, it is important to acknowledge that Deep Learning remains an evolving field [3]. Progress in this realm predominantly revolves around collecting large datasets, model training, and subsequent predictions. However, the effectiveness of these models largely depends on supervised learning, which requires a significant amount of labeled data. As a result, utilizing these models can often be costly and impractical for many real-world applications [4].

While the use of collective intelligence facilitates accurate labeling, extracting expert judgment poses challenges in terms of cost and time, particularly when dealing with a large number of tasks. Moreover, certain applications necessitate experts with profound subject-specific knowledge, such as medical image classification, galaxy classification, or identifying threats in X-ray images. In such instances, a small group of experts can be employed to procure the required labels through input aggregation. Input aggregation refers to the process of combining multiple individual judgments, feedback, or annotations obtained from a diverse group of human participants to arrive at a single representative decision or prediction. The goal of input aggregation is to leverage the collective intelligence

of the group of experts and obtain a more accurate and reliable result than what could be achieved by any individual participant alone. However, challenges arise when the performance of these experts exhibits significant variations due to inconsistent expertise and competencies. Consequently, the need arises for efficient methods that improve feedback quality and allocation of the available resources, ensuring that the quality of feedback remains uncompromised while preventing the overburdening of specific experts within the group. The following section discusses the challenges related to aggregating experts' feedback and optimizing expert selection in collective intelligence systems.

1.1 Challenges

Input Aggregation

Research in the past has demonstrated that by aggregating collective judgments through appropriate methods, results often surpass the performance achieved by individuals. This notion has found successful application in various areas, including image classification [5] and semantic segmentation [6].

However, input aggregation faces several challenges despite the potential advantages. One major limitation is the reliance on the wisdom of the crowd assumption, which assumes that the aggregated feedback will lead to an accurate ground truth prediction. While this is generally true for simple and straightforward tasks, the performance of input aggregation tends to degrade when dealing with complex tasks requiring specialized domain knowledge. In such cases, some experts may lack enough domain knowledge, leading to less reliable feedback.

Another challenge lies in the purely heuristic nature of collective intelligence-based techniques, such as Majority Voting (MV) [7], Max-Margin Majority Voting [8], Domain-Weighted Majority Voting [9], and Proxy Voting [10]. These methods typically rely on a single form of feedback from the crowd, which may not always be comprehensive or representative of the true ground truth. This reliance on a singular type of input can lead

to limitations in the accuracy and reliability of the aggregation process.

Furthermore, the effectiveness of crowdsourcing-based methods is significantly influenced by the size and diversity of the crowd. While large crowds can lead to more reliable aggregated judgments, smaller crowds may not produce sufficiently accurate results. This presents a challenge in determining the appropriate crowd size for a given task and optimizing the aggregation process accordingly.

Extracting meaningful features becomes a crucial aspect of input aggregation because it seeks to address the limitations imposed by the inherent noise and variance present in crowdsourced data. By refining and improving the extracted labels, the input aggregation methods can attain higher accuracy and robustness in their predictions. It is crucial to have precise and informative features when working with complex samples and specialized domain tasks to ensure accurate predictions.

Given these challenges, it becomes evident that there is a need for novel algorithms and methodologies to improve input aggregation, address the limitations of crowdsourcing-based techniques, and increase the overall reliability and accuracy of the process compared to the existing methods. By overcoming these challenges, input aggregation can become a more robust and effective tool for obtaining valuable insights and predictions from crowdsourced data.

Optimization of Expert Selection

The competencies of experts generally vary and exhibit inconsistencies due to differences in their domain knowledge, experience, and work environments. Particularly when dealing with complex samples, divergent opinions among experts can emerge, emphasizing the importance of obtaining accurate and reliable expert judgments for real-world applications. For instance, in medical image analysis, the precise evaluation of a sample that may indicate the presence of a cancerous tumor necessitates the involvement of expert professionals. Meanwhile, less experienced professionals may effectively handle simpler samples, contributing to reduced labeling costs.

To ensure efficient utilization of available expertise, it is important to simultaneously model both the experts' competencies and the samples' complexity. This modeling process applies to a set of labeling tasks assigned to a small group of experts. By doing so, the optimization of expert selection aims to enhance task assignment [11] and avoid sub-optimal allocation of valuable resources. A well-designed expert selection process can lead to improved accuracy and efficiency in real-world applications that rely on expert opinions for decision-making and analysis.

1.2 Scope of the Thesis

In this thesis, we aim to advance the understanding and implementation of the methods associated with collective intelligence, contributing to improving feedback collection mechanisms. Through an in-depth analysis of input aggregation, we seek to develop robust and scalable approaches that use experts' feedback and accommodate diverse groups of experts and input types. Additionally, by exploring optimized expert selection, we strive to address the inconsistencies arising from differences in expertise, experience, and work environment among experts. Our objective is to develop algorithms that improve the accuracy and reliability of collective intelligence methods, particularly in scenarios where accurate assessments are critical, such as in medical image analysis, security thread detection, and financial fraud detection.

1.3 Thesis Contributions

This thesis makes the following contributions:

- **Improvement of Input Aggregation Methods:** We demonstrate the potential for enhancing the performance of input aggregation methods by leveraging an automated image classifier. This approach aims to improve the process of aggregating feedback from crowdsourced annotations, leading to more accurate and reliable predictions.

- **Enhancement of the Feature Set:** To further improve the performance of input aggregation methods, we propose a Deep Learning-based technique that extracts additional features and includes those into the process. These enriched features can significantly enhance the quality and informativeness of the features collected through different input elicitation methods, enabling more effective training of input aggregation algorithms.

- **Development of a Task-Assignment Algorithm:** We propose a novel task-assignment algorithm to address resource and expert allocation. This algorithm seeks to optimize the assignment of labeling tasks among a group of experts, ensuring that each task is assigned to the most suitable expert, thus maximizing the overall accuracy of the labeling process.

- **Generation of a Large Synthetic X-ray Dataset:** A large image dataset is required for effectively training and testing deep learning algorithms for this research. So, as an additional contribution, we propose a novel algorithm to generate a large synthetic dataset consisting of X-ray images. This extensive dataset forms the cornerstone for training and evaluating our proposed algorithms, facilitating rigorous testing and validation of our methodologies.

The methodologies developed in this thesis are specifically applied and tailored for inspection tasks, with a primary focus on X-ray image analysis. However, it is important to note that our research's fundamental principles and techniques have the potential for broader applications in more general scenarios. The insights gained from addressing the challenges in inspection-related tasks can serve as a foundation for adapting and extending these collective intelligence-based methods to a wide range of domains where accurate and reliable data annotation is important. As a result, the findings presented in this thesis offer valuable contributions not only to inspection tasks but also to the wider field of collective intelligence and its applications in diverse fields of study.

1.4 Outline

The organization of the document is as follows. Chapter 2 provides relevant background information along with related works in input aggregation methods and task assignment optimization. It lays the foundation for understanding the context and significance of the subsequent chapters. Chapter 3, describes the improved aggregation method using automated classifiers. Chapter 4 introduces our methodology for generating additional features through a Deep Learning algorithm. Chapter 5 presents the process of creating a synthetic dataset of X-ray images and proposes a task-assignment algorithm. Finally, Chapter 6 provides a comprehensive summary of the research findings, discussions of their significance, and implications. Moreover, we outline potential avenues for future research and development in the field of input aggregation and related areas.

Chapter 2

Related Works

Input Aggregation Methods

In recent years, crowdsourcing has found widespread application in a variety of image labeling/classification tasks, spanning from simple visual identification tasks to those demanding domain expertise. Numerous studies have capitalized on crowdsourcing to annotate large-scale datasets involving subjective analysis, such as conceptualized images [12], scene-centric images [13], and publicly available general-purpose images [14, 15]. Additionally, crowdsourcing techniques have been successfully tailored to address complex visual labeling/classification contexts requiring specialized domain knowledge, such as identifying fish and plants [16, 17], endangered species through camera trap images [18], locations of targets [19], land covers [20], and sidewalk accessibility [21].

Low cost and rapid processing capabilities have made crowdsourcing useful for classifying CT images in medical applications. Noteworthy tasks in this domain have included identifying malaria-infected red blood cells [22], detecting clinical features of glaucomatous optic neuropathy [23], categorizing dermatological features [24], labeling protein expression [25], and various other medical image analysis tasks [26, 27].

Addressing the technical challenges associated with maximizing the benefits of crowdsourcing is important despite its effectiveness in handling high work volumes. A major obstacle is figuring out how to combine different sources of information that may contradict each other in order to create a precise representation. This requires implementing reliable methods for evaluating and estimating judgments. The quality of predictions heavily relies on the method used to consolidate crowdsourced inputs [28]. Consequently, numerous

works have focused on developing algorithms to tackle this task, drawing inspiration from computational social choice, a field dedicated to the rigorous analysis and design of data aggregation mechanisms [29]. One such method is Majority Voting (MV) [7], which is popular for its simplicity. Ipeirotis et al. [30] proposed the Expectation-Maximization (EM) method that presents a framework for managing quality on Amazon Mechanical Turk that includes techniques such as cost-sensitive classification and massive redundancy. Karger et al. [31] consider a general model of crowdsourcing tasks and pose the problem of minimizing the total price (i.e., number of task assignments) that must be paid to achieve a target overall reliability. The study shows that the proposed algorithm significantly outperforms majority voting and is asymptotically optimal through comparison to an oracle that knows the reliability of every worker. GLAD (Generative model of Labels, Abilities, and Difficulties) is proposed by Whitehill, J. et al. [32], presenting a probabilistic model to infer labels from images that outperform MV. Quoc et al. [33] shows the EM [30] method is more accurate than the existing methods, while MV [7] is best in terms of computation time.

However, one promising direction that hasn't been explored yet involves data-driven approaches that improve the performance of input aggregation methods. Deep neural networks have achieved breakthrough performances [34, 35, 36] in image classification and object detection tasks over the past few years. In combination with the expert's feedback, these algorithms can improve the performance of the aggregation methods significantly.

Self-Supervised Features

Besides relying on experts' feedback, the input aggregation methods can also make use of additional features of the samples to perform better. The performance of the machine learning-based input aggregation methods heavily relies on the quality and informativeness of the input features. In scenarios where the available features lack discriminative power or fail to capture relevant information, the accuracy of the aggregation process may be significantly affected. Therefore, there is a need for techniques that can enhance the quality

of the feature sets and improve the overall performance of input-aggregation methods.

Deep learning models have demonstrated remarkable success in various domains, primarily due to their ability to learn rich and hierarchical representations from raw data. However, the widespread adoption of deep learning approaches is often hindered by the requirement of large-scale labeled datasets for training [37]. To address this challenge, self-supervised learning [38] has emerged as a promising alternative. Self-supervised learning tasks involve training models to predict certain aspects of the input data without explicit supervision. Deep learning models can effectively learn useful representations from unlabeled data by formulating tasks such as image colorization, image inpainting, or image rotation prediction.

One popular framework for self-supervised learning is the Simple Framework for Contrastive Learning of Representations (SimCLR) introduced by Chen et al. [1]. SimCLR maximizes the agreement between differently augmented views of the same image and learns representations that capture the underlying structure of the data. By leveraging large-scale unlabeled datasets, SimCLR has shown significant improvements in representation learning compared to traditional supervised learning approaches.

In addition to SimCLR, various other self-supervised learning methods have been proposed in the literature. For example, the Momentum Contrast (MoCo) framework introduced by He et al. [39] utilizes a queue of negative samples and a momentum update mechanism to enhance the learned representations. Another approach, the Bootstrap Your Own Latent (BYOL) method proposed by Grill et al. [40], leverages two copies of the same neural network and a target network to learn representations without negative pairs.

These self-supervised learning methods aim to capture meaningful and generalizable features from unlabeled data, enabling deep learning models to extract high-level representations that can be beneficial for downstream tasks. By leveraging the intrinsic structure and patterns within the data, self-supervised learning provides a way to learn useful features without the need for extensive labeled data.

The learned representations in self-supervised learning have been shown to transfer well to various domains. In computer vision tasks, self-supervised features have been successfully

employed for image classification [1], object detection [41], and semantic segmentation [42]. Similarly, in natural language processing, self-supervised methods such as BERT [43] have demonstrated improved performance in tasks like text classification, named entity recognition, and sentiment analysis.

While input-aggregation methods and self-supervised learning have individually garnered significant attention, their combination remains relatively unexplored. The potential synergy between these two research areas offers promising avenues for improving the performance and robustness of deep learning models. By incorporating self-supervised features learned through a SimCLR model into existing input-aggregation methods, it is possible to bridge the gap between unsupervised representation learning and input feature aggregation. This integration has the potential to enhance the discriminative power of the aggregated features, enabling more accurate and reliable predictions.

Optimized Task Assignment

Optimized task assignment has been an active research area for many years, and a wide range of approaches have been proposed to solve this problem. Traditional optimization techniques [44, 45] involve formulating the problem as a mathematical model with constraints and an objective function and then using optimization algorithms to find the optimal solution. For example, the Hungarian algorithm [46] is a well-known algorithm for solving the task assignment problem, which uses a cost matrix to assign tasks to agents to minimize the total cost.

Researchers have explored machine learning techniques to solve task assignment problems. For example, reinforcement learning has been used to find optimized task assignments in various settings, such as job scheduling and vehicle routing [47]. In reinforcement learning, an agent learns to make decisions by interacting with its environment and receiving feedback through rewards or penalties. The agent's goal is to maximize its long-term reward, often defined as a function of its task assignments. In more complex scenarios, such as multi-agent systems and decentralized decision-making [48], reinforcement learning has been used to

learn task assignment policies directly from data without relying on handcrafted optimization algorithms [49].

In addition to machine learning techniques, researchers have explored evolutionary algorithms, swarm intelligence, and other optimization techniques to solve the task assignment problem. For example, genetic algorithms have been used to find optimized task assignments in resource allocation and supply chain management [50]. Li et al., [51], proposed an improved genetic algorithm for multi-agent task allocation with time window constraints. The authors established a mathematical model for task allocation and analyzed the constraint problem of the time window, using the penalty function method to handle the constraint condition. Additionally, they incorporate the improved Large Neighborhood Search (LNS) into the local search to increase population diversity and demonstrate the effectiveness of the proposed algorithm through simulations.

Researchers have also investigated other methods for solving the task assignment problem. For example, some studies have focused on using optimization techniques specific to particular tasks or agents. For instance, in a transportation context, the task assignment problem may involve assigning delivery tasks to a fleet of vehicles. Researchers have proposed optimization models that consider factors such as vehicle capacity, travel time, and customer preferences [52]. Similarly, in a manufacturing context, the task assignment problem, in ant colony optimization, artificial ants search for an optimized path between a source and a destination, considering factors such as machine capabilities, production rates, and production schedules [53].

Another approach to solving the task assignment problem is to use swarm intelligence algorithms inspired by the collective behavior of social insects such as ants, bees, and termites. In swarm intelligence algorithms, a population of agents collaborates to find an optimized solution to a problem. For example, in ant colony optimization, artificial ants search for an optimized path between a source and a destination by leaving and following pheromone trails [54]. In particle swarm optimization, a population of particles move through a problem space and adjust their positions based on their own experience and that

of their neighbors [55]. Swarm intelligence algorithms have been used to solve optimization problems, including task assignment, scheduling, and routing [56].

However, one potential area that remains relatively unexplored involves a data-driven approach to assess the complexity of samples and correlate it with experts' performance. Deep learning algorithms excel at learning intricate relationships between inputs and outputs of any objective function, making them suitable for modeling the association between image complexity and experts' capabilities. By leveraging these algorithms to determine the success probability of expert-sample pairs, we can obtain valuable insights into the probable performance of a group of experts. Once the cost matrix is established, an optimization algorithm, such as the Hungarian Algorithm, can be applied to achieve an optimized task assignment. This integration of data-driven techniques and optimization methods holds significant potential for improving task assignments and optimizing resource allocation in various domains.

Chapter 3

Improving Input Aggregation Methods Using Automated Classifiers

In recent years, computer vision approaches based on machine learning (ML) and, in particular, those based on deep convolutional neural networks have demonstrated significant performance improvements over conventional approaches for image classification and annotation [34, 57, 58]. However, these algorithms generally require a large, diverse set of annotated data to generate accurate classifications. Large amounts of annotated data are not always available, especially for tasks where producing high-quality meta-data is expensive, such as image-based medical diagnosis [59] and pattern recognition in geospatial remote sensing data [37, 60]. In addition, ML algorithms are often sensitive to perturbations in the data for complex visual tasks, such as object detection in cluttered backgrounds and detection of adversarial examples [61, 62], due to the high dimensionality and variability of the feature space of the images.

Crowdsourcing has received significant attention in various domain-specific applications as a complementary approach to image classification. Its growth has been accompanied by the emergence of online crowdsourcing platforms (e.g., Amazon Mechanical Turk, Prolific), which are widely employed to recruit and compensate human participants for annotating and classifying data that are difficult for machine-only approaches. In general, crowdsourcing works by leveraging the concept of the "wisdom of the crowd" [63], with which the judgments or predictions of multiple participants are aggregated to sift out the noise, and to approximate a ground truth better [64]. Numerous studies over the last decade have established that, under the right circumstances and with the proper aggregation methods, the collective

judgment of multiple non-experts is more accurate than those of almost any individual, including well-informed experts. This concept of using groups to make collective decisions has been successfully applied to many visual tasks ranging from simple classification and annotation [5] to complex real-world applications, including assessment of damages caused by natural disasters [65], and segmentation of biomedical images for diagnostic purposes [6].

However, high amounts of richly annotated data are inaccessible in various situations, and/or obtaining them is prohibitively costly. Yet, when fewer data are available, ML methods provide a natural mechanism for incorporating multiple crowdsourced inputs in different forms since they are designed for classification based on input features. Previous works have used a single form of input (i.e., mostly binary classification labels provided by participants) as a feature for ML algorithms on visual classification tasks. However, the vast majority have not considered obtaining additional inputs from any data-driven algorithm to improve the classification performance. This work investigates how the performance of crowdsourcing-based image classification tasks can be improved using an automated image classifier and a variety of user-provided inputs.

To pursue these objectives, we design several experiments that elicit a diversity of inputs on each classification task: binary classification (1 = positive or 0 = negative); target object's location; level of confidence in the binary response (on a scale from 0-100%); guess of what the majority of participants' binary classification is on the same task; and level of the perceived difficulty (on a scale from 0 to 1) of the binary classification task (on a discrete scale). We use the elicited inputs as features for ML algorithms to harness the benefits of both collective human intelligence and machine intelligence. The results indicate that integrating diverse forms of input elicitation, including self-reported confidence values, can improve the accuracy of crowdsourced computation. Then we design and implement an automated image classification method based on the ResNet-50 neural network architecture [66] by training it on multiple datasets ranging from 10,000 to 90,000 image samples. The outputs of this automated classifier are used as additional features within the crowdsourcing-based ML algorithms. These additional results demonstrate that this hybrid image classification

approach can provide more accurate predictions, especially for larger datasets.

3.1 Crowdsourcing-based ML classification

This section introduces input elicitations and describes how they can be utilized within a crowdsourcing-based ML classifier. The elicitation methods (Section 3.1.1) were introduced by Yasmin et al. [67, 68], our collaborators from Arizona State University. They also designed the activities and conducted the experiments on human participants, which are described in this section.

Consider the image label aggregation problem where a set of images I will be labeled by a set of participants P. Without loss of generality, assume each image and participant has a unique identifier, that is, $I = \{i_1, i_2, ..., i_n\}$ and $P = \{p_1, p_2, ..., p_m\}$, where n and m represent the total number of images and participants, respectively. For each image $i_k \in I$, the objective is to infer the binary label $y_k \in \{0, 1\}$, where $y_k = 1$ if the specified target object is present in the image (i.e., positive image) and $y_k = 0$ (i.e., negative image) otherwise. Since each worker may label only a subset of the images in these experiments, let $P(i_k) \subseteq P$ be the set of participants who complete the labeling task of image $i_k \in I$. In contrast to most crowdsourced labeling tasks, where only a single label estimate is elicited per classification task, each participant is asked to provide multiple inputs from the following five options in the featured experiments.

1. The first input is their binary response $l_k^j \in \{0, 1\}$ (i.e., classification label) indicating the presence/absence of the target object in image i_k.

2. The second input is a coordinate-pair (u_k^j, v_k^j) indicating the target object's location (elicited only when $l_k^j = 1$).

3. The third input is a numeric value $c_k^j \in [0, 100]$ indicating the degree of confidence in the binary response l_k^j.

4. The fourth input is another binary choice $g_k^j \in \{0, 1\}$ indicating what p_j estimates the binary response assigned by the majority of participants to i_k is; this input is referred to in this study as the Guess of Majority Elicitation (GME).

5. The fifth input is a discrete rating $d_j^k \in \{1, 2, 3, 4\}$, whose values are mapped from four linguistic responses—1:"not at all difficult", 2: "somewhat difficult", 3: "very difficult", and 4:"extremely difficult"—indicating, in increasing order, the perceived difficulty of task i_k.

The remainder of this section describes how shallow machine learning models are built using different features to generate predictions before integrating the models with the deep learning–based automated classifier.

3.1.1 Features for Crowdsourcing-based ML Methods

Seven features were extracted from the five input elicitations discussed at the beginning of this section for use with the ML classifiers. These features are described in the ensuing paragraphs.

- **Binary Choice Elicitation (BCE)**: For each image $i_k \in I$, the Binary Choice Elicitation values are divided into two sets: one containing the participants with response $l_k^j = 1$ and the other containing participants with response $l_k^j = 0$. The number of participants in each set can be used as an input feature within an ML classifier. However, since the number of participants can vary from image to image in practical settings, it is more prudent to use the relative size of the sets. Note that these relative sizes are complements of each other; that is, the fraction of participants who chose $l_k^j = 1$ as their binary choice label can be determined by subtracting from 1.0 the fraction of participants who chose $l_k^j = 0$. Therefore, to remove redundancy and co-linearity within the features, only one of these values is used as an input and

is given as:

$$x_k^1 = \frac{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 1)}{|P(i_k)|}$$

where x_k^1 is the fraction of participants who answer that the target object is present in image i_k.

- **Spatial Elicitation (SE)**: A clustering-based approach is implemented to identify participants whose location coordinates (u_k^j, v_k^j)—elicited only when they specify that the target object is present—are close to each other. For each image $i_k \in I$, participants with binary choice label $l_k^j = 1$ are divided into multiple clusters using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [69]. The reason for choosing this algorithm is twofold. First, DBSCAN can identify groups of points that are close to each other but form arbitrary shapes; since the target images have varying shapes and sizes, this is what one would expect to see in a single image if all collected data points were overlaid onto it. Second, this clustering algorithm can easily mark as outliers/noise the points in low-density areas, i.e., coordinate points that have a significant distance from each other. After clustering, the fraction of participants belonging to the largest cluster is used as an input feature within the ML classifiers. For image i_k, this input feature can be expressed as:

$$x_k^{SE} = \frac{\max\limits_{r \in R_k} n_r}{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 1)},$$

where n_r is the number of participants in cluster r and R_k is the set of clusters identified by DBSCAN for image i_k.

- **Confidence Elicitation (CE)**: Although previous works have explored using confidence scores to improve the annotation quality of crowdsourced data [70], very few have incorporated this input within a machine learning model. The confidence values are divided into two sets based on the l_k^j, and the corresponding averages are used as

additional features for the ML classifier. For image $i_k \in I$, these two input features can be expressed as

$$x_k^{conf,\,1} = \frac{\sum\limits_{p_j \in P(i_k)} c_k^{j*} \mathbb{1}(l_k^j = 1)}{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 1)}; \text{and}$$

$$x_k^{conf,\,0} = \frac{\sum\limits_{p_j \in P(i_k)} c_k^{j*} \mathbb{1}(l_k^j = 0)}{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 0)}.$$

The confidence values are rescaled linearly between 0 and 100 before incorporating them within the features.

- **Guess of Majority Elicitation (GME)**: Similar to BCE, GME is converted into a single feature based on the number of participants whose g_k^j response value is 1 and is written as

$$x_k^{GME,\,1} = \frac{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(g_k^j = 1)}{|P(i_k)|}.$$

- **Perceived Difficulty Elicitation (PDE):** Previous research has shown that a task's perceived difficulty level can be used to some extent to improve the quality of annotation. In most cases, the difficulty level is set based on inputs from experts, that is, participants with specialized knowledge for the task at hand [71], or it is estimated from the classification labels collected from participants [31]. Unlike these works, the featured experiments gather the perceived difficulty of each task directly from each participant to evaluate the reliability of this information and its potential use within ML classifiers. For each image $i_k \in I$, the difficulty elicitation values d_k^j are divided into two sets: one for the participants with response $l_k^j = 1$, and the other for the remaining participants with response $l_k^j = 0$. The mean values from each set are then used as additional features for the ML classifier; these two input features can be expressed as

$$x_k^{PDE,\,1} = \frac{\sum\limits_{p_j \in P(i_k)} d_k^j \mathbb{1}(l_k^j = 1)}{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 1)}; \text{and}$$

$$x_k^{PDE,\,0} = \frac{\sum\limits_{p_j \in P(i_k)} d_k^j \mathbb{1}(l_k^j = 0)}{\sum\limits_{p_j \in P(i_k)} \mathbb{1}(l_k^j = 0)}.$$

3.1.2 Experiment Design

Before introducing the components of the experiment design, we describe the *MPEG-7 Core Experiment CE-Shape-1 Test Set* [72], which is the source data from which the featured crowdsourcing activities are constructed. The dataset comprises black-and-white images of diverse shapes and objects, including animals, geometric shapes, common household objects, etc. The dataset consists of $1,200$ objects/shapes (referred to here as *templates*) divided into 60 object/shape classes, each containing 20 members. Figure 3.1 provides representative templates from 12 of these classes.

Images across all experiments were generated with a $1,080 \times 1,080$ pixel beige background (RGB values $(245, 245, 220)$). The rotation of all object templates follows the uniform distribution $U(0, 360)$. The remaining parameters are specific to each experiment. In experiment sets C and D, we use images from four difficulty levels, "very difficult", "difficult", "average", and "easy", with densities of 90, 100, 115, and 150, respectively. See Table 3.1.

3.1.3 Description of Activities

For the crowdsourcing activities, we designed two studies, each of which elicits multiple forms of input from participants to complete a number of image classification tasks. A user interface was designed and implemented to perform the two studies, which differ based on the subsets of input elicitations tested and the class balance ratios of the image datasets (more details are provided later in this subsection). The interfaces were developed in HTML

Table 3.1: Experiment Sets C and D sample images

Sets C and D Examples	**Positive**	**Negative**
Very Difficult Density: 90 Scale: $T(0.25, 0.35, 0.40)$ Color: - $(31, 28, 28)$ - $(20, 92, 163)$ - $(89, 135, 28)$ - $(196, 130, 23)$ Transparency: $U(150, 200)$		
Difficult Density: 100 Scale: $T(0.25, 0.35, 0.40)$ Color: - $(31, 28, 28)$ - $(20, 92, 163)$ - $(89, 135, 28)$ - $(196, 130, 23)$ Transparency: $U(150, 200)$		
Average Density: 115 Scale: $T(0.25, 0.35, 0.40)$ Color: - $(31, 28, 28)$ - $(20, 92, 163)$ - $(89, 135, 28)$ - $(196, 130, 23)$ Transparency: $U(150, 200)$		
Easy Density: 150 Scale: $T(0.25, 0.35, 0.40)$ Color: - $(31, 28, 28)$ - $(20, 92, 163)$ - $(89, 135, 28)$ - $(196, 130, 23)$		

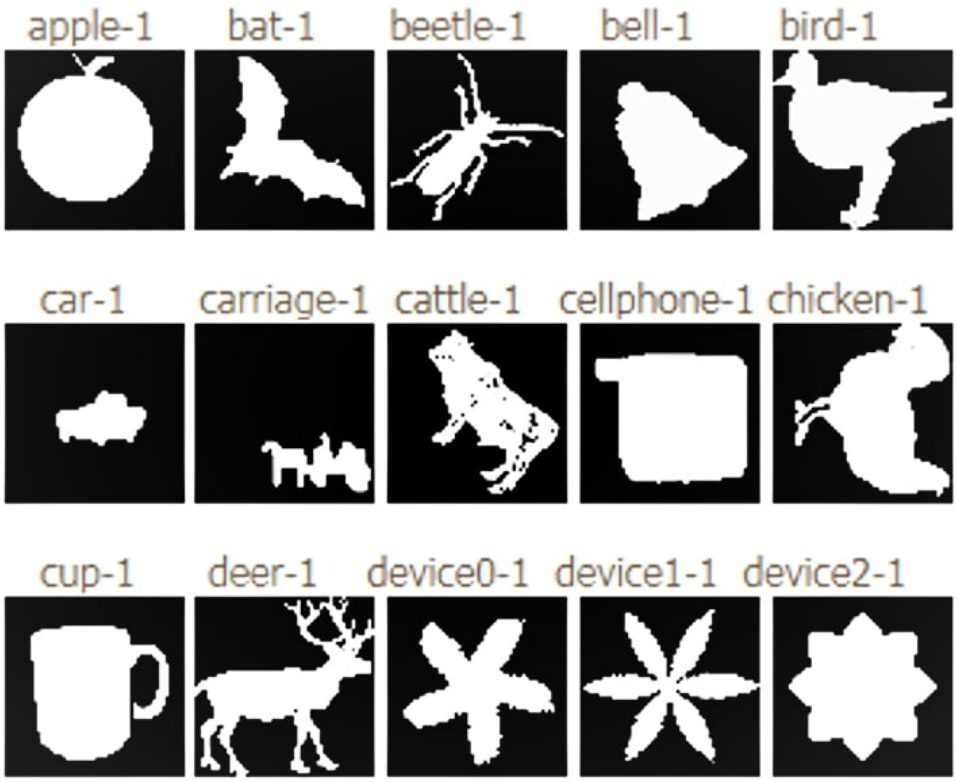

Figure 3.1: Object/shape templates from the MPEG-7 Core Experiment CE-Shape-1 Test Set

and JavaScript and then deployed using Amazon Mechanical Turk (MTurk). Participants were first briefed about the nature of the study and shown a short walk-through video explaining the interface. Afterward, participants proceeded to the image classification tasks shown in a randomized order. After completing an experiment, participants were disallowed to participate in further experiments to ensure we had unique responses.

Figures 3.2 and 3.3 provide examples of the user interfaces, both of which instituted a 60-second time limit to view each image before it was hidden; participants were prompted to provide their inputs during or after the viewing time to proceed.

If the participant completed the inputs before the time limit, they were allowed to proceed to the next image; on the other hand, if the time limit was reached, the image was hidden from view, but participants could take as much time as they needed to finish

providing their inputs. Participants were not allowed to proceed to the next image until they provided input. The time limit was imposed to ensure the scalable implementation of a high number of tasks. In particular, the goal is to develop activities that capture enough quality input from participants while mitigating potential cognitive fatigue. In preliminary experiments, we found that participants rarely exceeded 45 seconds. In the featured studies (described in the next two paragraphs), the full 60 seconds were utilized in only 7% of the tasks, with an average time of around 27 seconds. The number of image classification tasks given to the participants varied by experiment, ranging from 16 to 40 images (see Table 3.2 for details). Based on findings of prior studies with shared characteristics, we deemed this number of tasks reasonable and not cognitively burdensome to participants. For instance, [73] performed a visual identification crowdsourcing study where participants were assigned up to 80 tasks, each of which took a median time of 29.4 seconds to complete. The authors found that accuracy decreased negligibly for this workload (i.e., twice as large as in the featured studies).

In the first study, seven experiments were completed and grouped into two sets: Experiment Set A (four experiments) and Experiment Set B (three experiments). Each experiment used a balanced set of images, with half containing the target template (i.e., positive images); target objects were chosen to avoid confusion with other template classes. See Table 3.2 for image generation parameters. The parameter ranges selected for Experiment Set A was designed to keep the difficulty of the classification tasks relatively moderate. On the other hand, a more complex set of parameters was selected for Experiment Set B to expand the range of difficulty. These differences are reflected in the individual performance achieved in these two experiment sets, measured by the respective average number of correct classifications obtained by participants. For Experiment Set A, individual performance averages ranged between 59% and 77% for each of the four experiments, whereas for Experiment Set B, they were between 54% and 82% for each of the three experiments.

In the second study, six experiments were conducted. These experiments were also grouped into Experiment Set C (three experiments) and Experiment Set D (three experi-

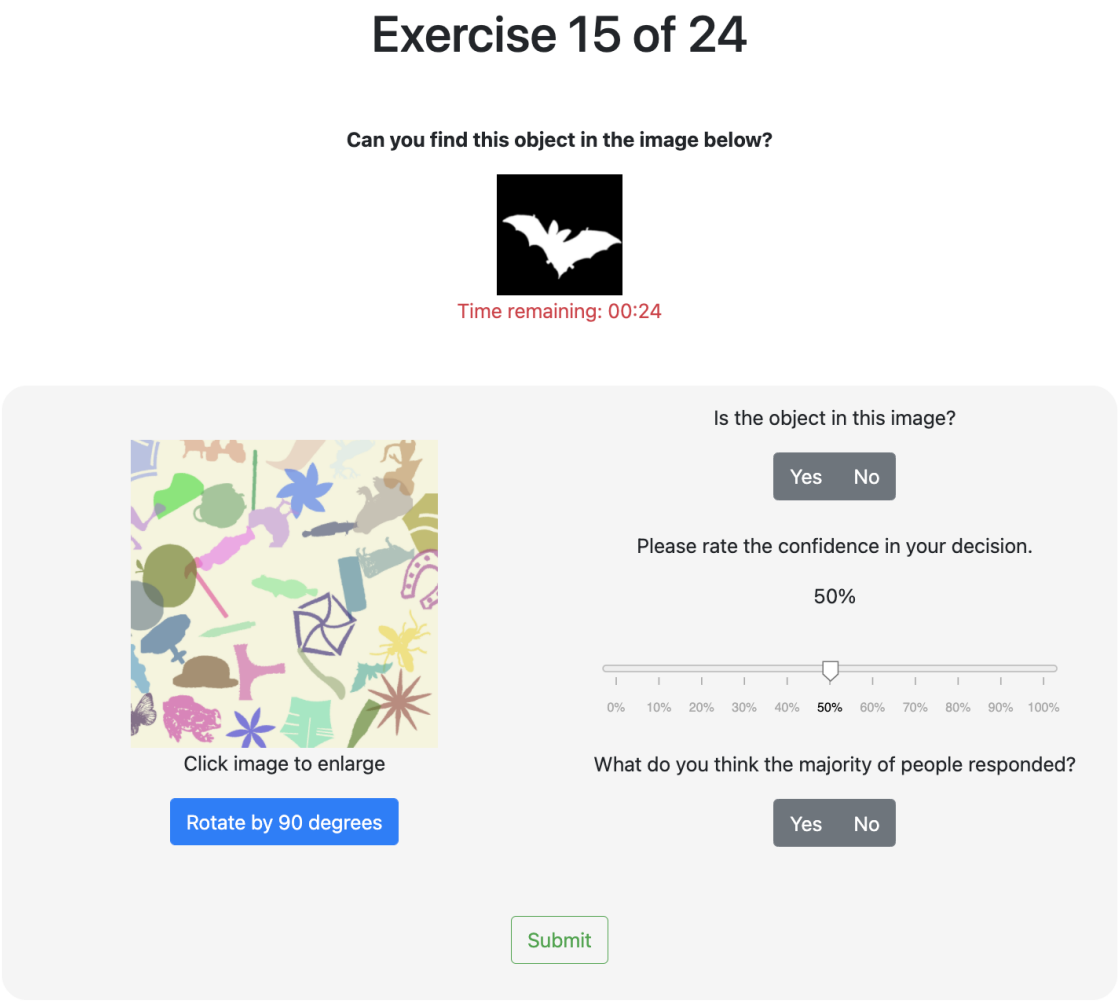

Figure 3.2: Image classification task UI for balanced dataset - the image contains bat (lower right)

ments). Each consisted of image sets with an imbalanced positive- to negative-images ratio. Experiment Set C had a 20-80 balance, meaning that 20% of the images were positive, and 80% were negative; Experiment Set D had a 10-90 balance. The results of Experiment Sets A and B revealed that *scale* and *density* are the only factors that had a statistically significant impact on individual performance. Based on this insight, we constructed a simple linear regression model with these two parameters as the predictors and *proportion of correct participants* as the responses; the model is very significant ($p < 0.001$), and its adjusted

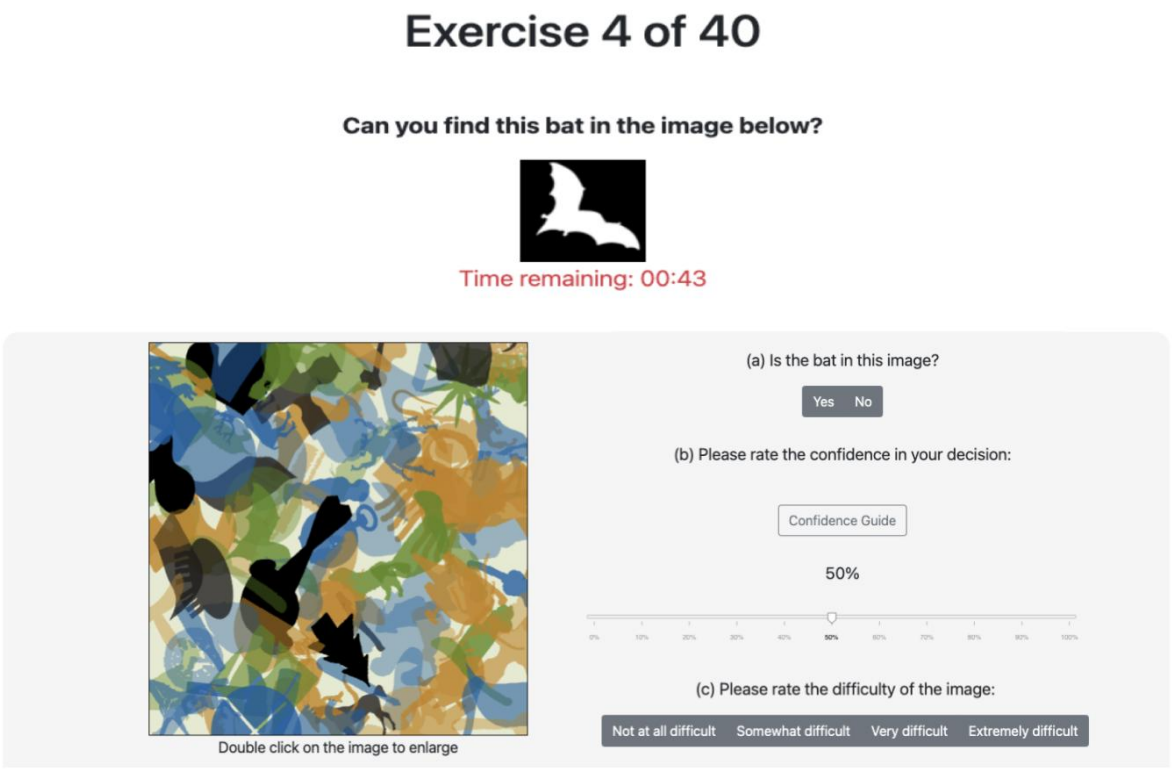

Figure 3.3: Image classification task UI for imbalanced dataset - the image contains bat (center left)

R-squared value is 0.65. The model generated image sets with an approximated difficulty level by modifying the scale and density parameters accordingly. It should be noted that the true difficulty of each image varies based on the random generation process. The model was implemented to design experiments consisting of classification tasks of reasonable difficulty, neither trivial nor impossible to complete. Images of four difficulty levels were generated for Experiment Sets C and D. At each difficulty level, the density was varied while keeping the other parameters consistent across images. This resulted in similar images but with different amounts of "clutter". The four difficulties generated were categorized as "very difficult", "difficult", "average", and "easy". Experiment Sets C and D use an even split of each difficulty (i.e., 25% of generated images from each level). Experiment Sets C and D can be construed as "more difficult" than Experiment Sets A and B due to the imbalanced nature of the set. However, the individual performance achieved in these two experiment sets, measured by the average percentage of participants with the correct classification, says otherwise. For the three respective experiments, individual average accuracy values ranged between 65% and 73% for Set C and between 58% and 72% for Set D.

Figures 3.2 and 3.3 show the user interface presented to participants in the first and second studies, respectively. For each classification task (i.e., image) in the first study, participants were asked to provide a binary response indicating whether or not a target object was present. If they responded in affirmation, they were prompted to locate the target object by clicking on it. Then, participants were asked to rate their confidence in their binary response on a scale from 0-100%. Finally, participants were asked to guess the binary response of the majority of participants. The second study asked participants similar questions as the first study. For each classification task, participants were also asked to provide a binary response indicating whether or not a target object was present and their confidence level in this response. If they responded in affirmation, however, they were prompted to locate the target object by drawing a bounding box around it; the centroid of the bounding box was used as the (x, y)-coordinates gathered from this elicitation. In replacement to the last question of the first study, participants were asked to rate the

Table 3.2: Summary of experiment image parameters

Exp.		Images	Density	Scale	Color	Transparency	Target
Set A	#1 #2 #3 #4	16	{100, 120, 140, 160}	$\{T(0.2 \pm 0.12), .., T(0.65 \pm 0.12)\}$	Discrete: {4}	$U(100, 200)$	Bat Butterfly Apple Stingray
Set B	#5 #6 #7	24	{80} {80,100,120} {100, 150}	$\{T(0.2 \pm 0.05), T(0.3 \pm 0.05)\}$ $\{T(0.2 \pm 0.05), .., T(0.4 \pm 0.05)\}$ $\{T(0.2 \pm 0.05), T(0.3 \pm 0.05)\}$	Discrete: {1,...,6} $U(10, 255)$ for R,G,& B	$U(140, 170)$	Bat Turtle Various-7
Set C	#8 #9 #10	40	{90, 100, 115, 150}	$\{T(0.25, 0.35, 0.40)\}$	Discrete: {4}	$U(150, 200)$	Bat
Set D	#11 #12 #13						

difficulty of the specific image classified based on a discrete scale. The rating choices provided were "not difficult at all", "somewhat difficult", "very difficult", and "extremely difficult". These labels were mapped to 1, 2, 3, and 4 for use in the aggregation algorithms.

3.2 Enhancement of Crowdsourcing-based ML Methods with an Automated Classifier

In order to assess the difficulty of the image classification problem presented to participants and to evaluate the potential of hybrid human-ML approaches, we developed a deep learning image classification approach that leverages large training datasets. Deep convolutional neural networks naturally extract low-level image features and then progressively learn higher-level features to generate probability distributions toward target classes, providing an end-to-end data pipeline for the required task. Our classifier is based on ResNet-50, a popular variant of ResNet architecture [66], which has shown excellent performance on multiple image classification tasks. It has been extensively used by the computer vision research community and adopted as a baseline architecture in many studies over the last few years [74].

He et al. [66] trained the ResNet-50 architecture on the ImageNet [75] dataset, which

Table 3.3: Modified version of the ResNet-50 architecture diagram

Layer Name	Output Size	Layers
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 maxpool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
conv3_x	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
conv4_x	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
conv5_x	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
fc1	1024×1	dropout, 2048-d fc, relu
fc2	256×1	dropout, 1024-d fc, relu
fc3	128×1	dropout, 256-d fc, relu
fc4	1×1	dropout, 128-d fc

consists of 1.28 million images and 1,000 different classes. We modified the fully connected layers of the standard architecture to make it compatible with the binary classification task (See Table 3.3).

To assess the classifier's performance as a function of the training set size, we generated a balanced dataset of 100,000 samples, with 10,000 samples set aside as the validation set and the rest used as the training set. The images are representative of an even mixture of the difficulty classes used to generate Experiment Sets C and D. We trained and evaluated the performance of the network using training set sizes ranging from 10,000 samples to 90,000 samples, increasing the training set size by 10,000 every iteration, totaling nine different training sessions. Each training session started from the previous session's best-performing checkpoint of the network and continued for 35 epochs. We used the Adam optimizer [76] with default parameters (learning rate = 10^{-3}, $\beta_1 = 0.99$, $\beta_2 = 0.999$) and He's method [77]

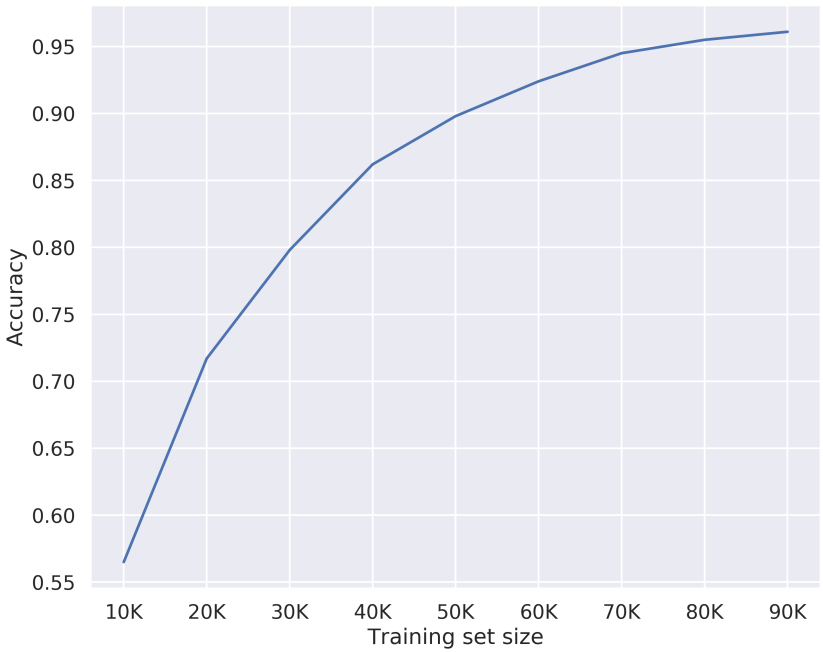

Figure 3.4: Validation accuracy vs. Training set size

to initialize the weights.

Figure 3.4 shows accuracy as a function of the training set size for the automated ResNet-50 classifier. The largest training set size of 90,000 samples leads to more than 95% accuracy on the balanced validation set (consisting of 5,000 positive and 5,000 negative samples). However, when trained on the smallest training set of 10,000 samples, the model performs only slightly better than random guessing (see Figure 3.4). The plot confirms that deep learning models almost always benefit from large datasets, given that the network has enough parameters to capture the learnable features.

We emphasize that this work does not aim to advance the state-of-the-art results for automated image classification. Instead, the automated classification method focuses on exploring the benefits and limitations of a hybrid method introduced herein that integrates the outputs of a well-known deep neural network into the crowdsourcing-based classification methods. In particular, the proposed method uses the output of the automated classifier as an additional feature of the featured ML methods. Table 3.4 summarizes the results for the

small imbalanced test sets used in Experiment Sets C and D as the training set grows larger. Due to the imbalanced nature of these test sets, this table and the rest of the analysis focus on F_1-score, false-negative rate (FNR), and area under the ROC curve (AUC). Before proceeding, it is worthwhile to mention two additional points regarding the values presented in the table. First, the input elicitation RC represents the probability value of positive classification obtained from the automated classifier when used as a feature. For example, BCE-RC indicates that both the binary elicitation inputs and the probability scores from the ResNet-50 were used as features for the ML classifiers. Second, the Combined Set C&D is created by merging the data from Experiment Sets C and D, thereby effectively doubling the size of the training set relative to the individual experiment sets.

Table 3.4 highlights those cases in which the performance of the hybrid method according to a given metric is better than both the completely automated approach (ResNet-50) and the results achieved by the crowdsourcing-based ML methods according to the best input combination. As expected, when the ResNet-50 performance is poor, using its output as a feature hurts the overall results. Conversely, when the ResNet-50 performance is near perfect, it is difficult to improve upon its performance by adding information obtained from the crowd. However, apart from those extremes, exploiting the output of the ResNet-50 is beneficial in most cases, mainly regarding F_1-score and AUC.

The proposed hybrid method, which uses the results from the automated classifier as an additional input feature for the crowdsourcing-based ML methods, exhibited a robust performance. They attained maximum F_1-scores of 0.98, 0.96 0.97 and minimum FNRs of 0.04, 0.08, 0.06 for Experiment Set C, D, and Combined Set C&D, respectively, all of which represent significant improvements over what crowdsourcing-based methods achieved on a standalone basis. While these top results were associated with the automated classifier training set of 90,000 samples, impressive results were obtained using smaller datasets for Combined Set C&D, compared to Experiment Set C and D separately. As an example, incorporating the output of the automated classifier trained on 50,000 samples with the crowdsourcing-based methods for Combined Set C&D improved the F_1-score significantly

Table 3.4: Performance analysis of Crowdsourcing-based ML methods with Expanded inputs from ResNet-50

Input Elicitations	Size of Dataset	ResNet50			KNN			LR			RF			SVM-Linear		
		F_1	FNR	AUC	F_1	FNR	AUC	F_1	FNR	AUC	F_1	FNR	AUC	F_1	FNR	AUC
Experiment Set C																
BCE-CE-SE-PDE*	–	–	–	–	0.81	0.29	0.92	0.81	0.29	0.86	0.81	0.29	0.90	0.81	0.29	0.86
RC		0.36	0.21	0.67	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	10k	–	–	–	0.73	0.38	0.85	**0.82**	0.25	**0.93**	0.70	0.38	0.89	0.75	0.38	0.92
BCE-CE-RC		–	–	–	0.70	0.42	0.89	0.77	0.25	0.88	0.74	0.33	0.86	0.80	0.33	0.91
RC		0.71	0.29	0.92	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	30k	–	–	–	0.77	0.38	0.83	0.75	**0.25**	0.92	0.78	0.33	0.89	0.76	0.33	0.92
BCE-CE-RC		–	–	–	0.75	0.38	0.81	0.73	**0.25**	0.91	0.78	0.33	0.88	0.80	0.33	**0.93**
RC		0.87	0.04	0.99	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	50k	–	–	–	0.80	0.25	0.95	**0.90**	0.04	0.97	0.84	0.21	0.97	**0.88**	0.13	0.98
BCE-CE-RC		–	–	–	0.82	0.25	0.92	**0.90**	0.04	0.98	0.84	0.21	0.97	**0.88**	0.13	0.97
RC		0.90	0.08	0.99	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	70k	–	–	–	**0.91**	0.13	0.98	**0.92**	**0.04**	0.99	**0.93**	0.13	0.98	**0.94**	**0.04**	0.99
BCE-CE-RC		–	–	–	**0.91**	0.13	0.98	0.88	**0.04**	**1.00**	**0.93**	0.13	0.98	**0.94**	**0.04**	0.99
RC		0.96	0.00	1.00	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	90k	–	–	–	**0.98**	0.04	0.98	0.94	0.04	0.96	**0.98**	0.04	0.97	**0.98**	0.04	0.99
BCE-CE-RC		–	–	–	**0.98**	0.04	0.98	0.9	0.04	0.97	**0.98**	0.04	0.97	**0.98**	0.04	0.99
Experiment Set D																
BCE-CE*	–	–	–	–	0.59	0.58	0.76	0.54	0.42	0.83	0.63	0.50	0.79	0.67	0.50	0.87
RC		0.17	0.33	0.62	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	10k	–	–	–	0.59	0.58	0.67	0.44	**0.25**	0.87	0.44	0.67	0.73	0.11	0.42	0.78
BCE-CE-RC		–	–	–	0.56	0.58	0.69	0.43	0.33	0.84	0.63	0.50	0.78	0.63	0.50	0.86
RC		0.50	0.42	0.87	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	30k	–	–	–	0.50	0.67	0.67	0.43	**0.33**	0.87	0.59	0.58	0.74	0.63	0.50	0.85
BCE-CE-RC		–	–	–	0.50	0.67	0.64	0.47	0.42	**0.88**	0.56	0.58	0.8	0.67	0.50	0.87
RC		0.79	0.08	0.96	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	50k	–	–	–	0.74	0.42	0.90	0.71	0.17	0.91	0.70	0.42	0.88	**0.80**	0.17	0.96
BCE-CE-RC		–	–	–	0.70	0.42	0.91	0.69	0.17	0.90	0.74	0.42	0.86	**0.80**	0.17	0.91
RC		0.83	0.17	0.98	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	70k	–	–	–	**0.91**	0.17	0.96	**0.88**	**0.08**	0.92	**0.91**	0.17	0.94	**0.96**	**0.08**	0.92
BCE-CE-RC		–	–	–	**0.91**	0.17	0.96	**0.88**	**0.08**	0.92	**0.91**	0.17	0.93	**0.96**	**0.08**	0.92
RC		0.96	0.08	0.98	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	90k	–	–	–	0.96	0.08	0.96	0.92	0.08	0.94	0.91	0.17	0.94	0.96	0.08	0.95
BCE-CE-RC		–	–	–	0.96	0.08	0.96	0.92	0.08	0.95	0.91	0.17	0.94	0.96	0.08	0.92
Combined Set C&D																
BCE-CE*	–	–	–	–	0.68	0.47	0.83	0.73	0.33	0.9	0.72	0.42	0.85	0.76	0.39	0.9
RC		0.27	0.25	0.65	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	10k	–	–	–	0.67	0.5	0.83	0.64	0.25	0.88	0.67	0.39	0.86	0.71	0.39	**0.91**
BCE-CE-RC		–	–	–	0.69	0.44	0.86	0.68	0.25	0.9	0.69	0.44	0.84	0.76	0.39	**0.91**
RC		0.63	0.33	0.90	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	30k	–	–	–	0.71	0.42	0.87	0.66	**0.25**	**0.92**	**0.79**	**0.31**	**0.94**	0.72	0.42	**0.91**
BCE-CE-RC		–	–	–	0.72	0.42	0.84	0.64	**0.28**	**0.92**	0.75	0.39	**0.91**	0.72	0.42	**0.93**
RC		0.84	0.06	0.98	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	50k	–	–	–	**0.87**	0.19	0.96	**0.86**	0.08	0.97	**0.91**	0.11	0.96	**0.85**	0.14	0.97
BCE-CE-RC		–	–	–	**0.86**	0.22	0.94	**0.86**	0.08	0.96	**0.86**	0.22	0.96	**0.85**	0.14	0.98
RC		0.88	0.11	0.99	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	70k	–	–	–	**0.96**	**0.08**	0.97	**0.92**	**0.06**	0.94	**0.96**	**0.08**	0.96	**0.93**	**0.06**	0.97
BCE-CE-RC		–	–	–	**0.93**	**0.08**	0.97	**0.89**	**0.06**	0.97	**0.96**	**0.08**	0.96	**0.92**	**0.06**	0.97
RC		0.96	0.03	0.99	–	–	–	–	–	–	–	–	–	–	–	–
BCE-RC	90k	–	–	–	**0.97**	0.06	0.97	0.93	0.06	0.98	**0.97**	0.06	0.96	**0.97**	0.06	0.98
BCE-CE-RC		–	–	–	**0.97**	0.06	0.97	0.92	0.06	0.94	**0.97**	0.06	0.97	**0.97**	0.06	0.98

*Denotes the input combinations that achieved the best performance among the Crowdsourcing-based ML methods

(see Table 3.4). However, the hybrid approach did not show better results for Experiment Sets C and D separately over the same training set size in some cases. This can be explained by the fact that Experiment Sets C and D have fewer data points than Combined Set C&D. This attests that, while crowdsourcing-based methods supplemented with the outputs of the

automated classifier perform very well on small datasets, too few data points can negatively affect the performance of the hybrid approach.

3.3 Discussion

The results demonstrate that the automated classifier significantly improves the classical machine learning model–based input aggregation methods. When the training set is small, any of the four ML classifiers tested in this work generated dependable results for datasets of varying difficulty levels. However, when the training set is larger, integrating the inputs from the automated classifier with the crowdsourcing-based ML methods improved the results even further. Those methods achieved near-perfect FNRs thanks to a large dataset used to train the automated classifier. The F_1-score was also significantly improved through this hybrid approach. Although smaller training sets of 50,000 samples slightly reduced the performance of the automated classifier, the numbers were still better than those obtained by standalone crowdsourcing-based methods. Altogether, the results demonstrate that it is possible to obtain better classifications at a relatively low cost by including diverse inputs as features within an ML classifier.

3.4 Conclusions

Crowdsourcing-based ML classifiers are useful to aggregate expert feedback and improve data quality. However, these methods rely on the data points obtained from the experts, which could be expensive and time-consuming. This study shows that using the predicted labels of automated classifiers as additional features with no direct involvement with the experts greatly enhances the results when large datasets are available. That way, one could avoid getting a large number of data points from the experts and still produce better results.

The code used to generate the synthetic images [78] can be found at `https://github.com/O-ARE/2D-Image-Generation-HCOMP`. In addition, the code used to train and evaluate

the automated classifier [79] can be found at `https://github.com/O-ARE/2d-image-classification`.

Chapter 4

Improving input aggregation Methods Using Self-Supervised Features

4.1 Introduction

Crowdsourcing has provided a useful approach to tackling complex tasks by aggregating user inputs and harnessing the collective intelligence and diverse perspectives of a large number of individuals to solve complex tasks. Aggregating user inputs to obtain accurate labels in numerous crowdsourcing scenarios is essential for generating reliable, high-quality results. However, noisy and misleading user inputs pose a significant challenge to the aggregation process, often resulting in suboptimal outcomes.

To tackle this challenge, various input aggregation methods have been proposed in the literature, aiming to combine user inputs effectively while mitigating the impact of noise. These methods typically rely on shallow machine learning models, such as k-nearest neighbors (KNN), logistic regression, or support vector machines (SVM), to learn decision boundaries from labeled training data (as discussed in Chapter 3). However, the performance of these models heavily relies on the quality of the input features.

Deep learning models have demonstrated remarkable success in a wide range of tasks, particularly in learning rich and hierarchical representations from raw data. These models are trained on large-scale labeled datasets, enabling them to capture complex patterns and extract high-level features. However, in scenarios where labeled data are scarce or expensive to obtain, leveraging the power of deep learning becomes challenging.

In recent years, self-supervised learning has emerged as a promising alternative for

training deep learning models without the need for extensive labeled data. Self-supervised learning tasks involve training models to predict certain aspects of the input data without explicit supervision. When labeled data are scarce, deep learning models can learn useful representations of the data without the need for explicit supervision or labeled data after the models are trained to perform "pretext" tasks, such as image colorization, image in-painting, and image-context prediction. A pretext task, in the context of machine learning and deep learning, refers to a task that is formulated to help the model learn useful representations of the data without the need for explicit supervision or labeled data. The term "pretext" implies that the task is not the primary objective or end goal of the learning process but serves as a means to an end. Instead, the model learns to solve the pretext task as a form of self-supervised learning, where the training data itself provides implicit supervision. The representations learned by the model during the pretext task can be highly informative. They can subsequently be used in various downstream tasks, such as image classification, object detection, and semantic segmentation.

By leveraging self-supervised learning with pretext tasks, deep learning models can extract meaningful and generalizable features from unlabeled data, reducing the reliance on large labeled datasets. This approach has shown promise in improving the performance and efficiency of deep learning models, making them more adaptable to real-world applications with limited labeled data. Motivated by the potential advantages of self-supervised learning, we propose a deep learning-based approach to improve existing input aggregation methods in crowdsourcing. Our key hypothesis is that incorporating additional features learned by a self-supervised deep learning model will enhance the performance of shallow machine learning models used in input aggregation. We train the model on a large-scale dataset of unlabeled images to test this hypothesis to learn robust and discriminative image features. Subsequently, we integrate the learned features with the existing input features and retrain the shallow machine-learning models.

This study aims to provide empirical evidence supporting the ability of self-supervised features to enhance the performance of input aggregation methods. By demonstrating the

advantages of incorporating self-supervised features, we expect to open up new possibilities for improving the accuracy and reliability of crowdsourcing tasks. Furthermore, our findings have the potential to impact existing input aggregation methods used in this domain.

The remainder of this chapter is organized as follows. Section 4.2 describes the deep learning model used in this study, Section 4.3 presents our proposed approach, and Section 4.4 provides an insight into the dataset used to train the model. Experiments are discussed in Section 4.5, which contains the methodology employed in this research, including details about the extraction and integration of self-supervised features. The results and analysis are presented in Section 4.6. Finally, Section 4.7 concludes the chapter, summarizing the contributions and highlighting future directions of research.

4.2 SimCLR Model

The SimCLR (Simple Framework for Contrastive Learning of Representations) model incorporates various components, including the encoder, projection head, and contrastive loss function, to learn powerful representations from unlabeled data. This section explains each component and its role in the SimCLR model.

4.2.1 Encoder

The encoder serves as the backbone of the SimCLR model and is responsible for extracting informative features from the input data. In our implementation, we adopt the ResNet-50 [80] architecture as the encoder. ResNet-50 is a deep convolutional neural network that has demonstrated exceptional performance in image classification tasks. It consists of multiple convolutional layers, residual blocks, and downsampling operations. This architecture allows the model to capture low-level and high-level visual features, enabling it to learn rich representations from raw image data.

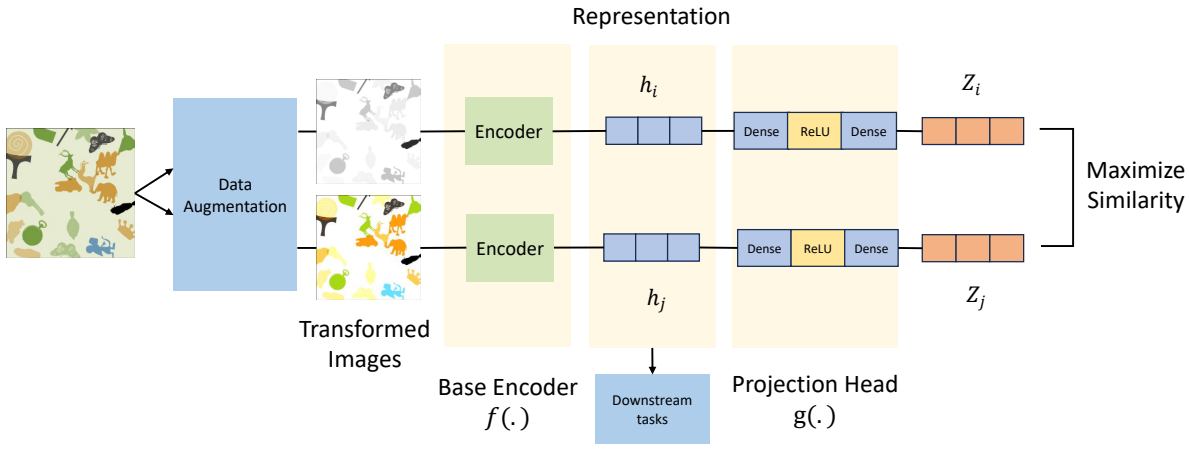

Figure 4.1: SimCLR framework. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. [1]

4.2.2 Projection Head

The projection head is a crucial component of the SimCLR model, designed to map the high-dimensional features extracted by the encoder into a lower-dimensional latent space. By reducing the dimensionality of the features, the projection head enhances the model's ability to capture semantically meaningful information and improves generalization. Typically, the projection head comprises one or more fully connected layers followed by a normalization layer. The fully connected layers act as a bottleneck, encouraging the model to learn more compact representations. The normalization layer enhances the stability and robustness of the learned features by normalizing their magnitudes.

4.2.3 Contrastive Loss Function

At the core of the SimCLR model lies the contrastive loss function, which drives the learning process by maximizing agreement between augmented views of the same sample while minimizing agreement with augmented views of different samples.

InfoNCE, where NCE stands for Noise-Contrastive Estimation, is a type of contrastive loss function used for self-supervised learning, which can be written as:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\exp(\text{sim}(z_i, c_i)/\tau)}{\sum_{j=1}^{K}\exp(\text{sim}(z_i, c_j)/\tau)}\right)$$

where:

- N represents the batch size,
- z_i refers to the representation of the anchor sample,
- c_i denotes the representation of the positive sample,
- τ is the temperature parameter that controls the sharpness of the distribution,
- $\text{sim}(z_i, c_j)$ represents a similarity function that measures the similarity between z_i and c_j,
- K is the number of negative samples used for contrastive estimation.

The combination of the encoder, projection head, and contrastive loss function enables the SimCLR model to learn powerful and transferable representations from unlabeled data. Leveraging the expressive power of the ResNet-50 encoder, the SimCLR model captures a wide range of visual features, encompassing both low-level details and high-level semantics. The projection head refines these representations by reducing their dimensionality, leading to a more compact and informative latent space. Finally, the contrastive loss function guides the learning process, compelling the model to discriminate between different samples and learn representations that effectively capture the underlying data distribution.

4.3 Proposed Approach

In this section, we present our proposed approach for improving input-aggregation methods using self-supervised features learned by a SimCLR model. The main goal of our approach is to use the self-supervised model to learn meaningful representations from unlabeled data and subsequently combine these representations into existing features obtained through input elicitation methods to enhance the performance of the input aggregation methods. Figure 4.2 provides an overview of the workflow of our proposed approach.

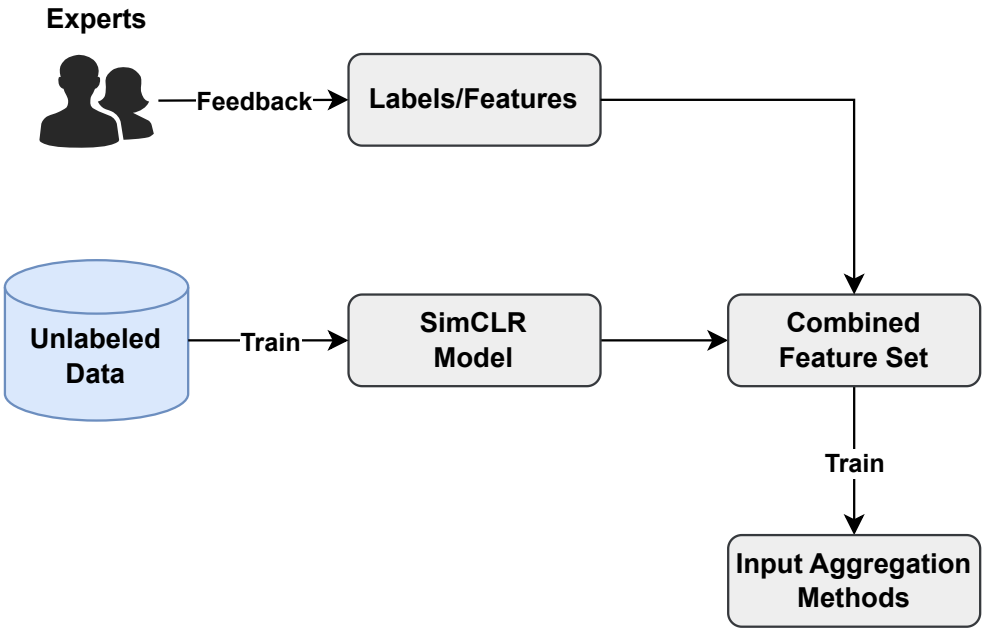

Figure 4.2: SimCLR-based feature extraction and training input aggregation methods on the combined feature set

4.3.1 SimCLR-based Feature Extraction

Our approach begins with training a SimCLR model on a large corpus of unlabeled images. SimCLR is a powerful self-supervised learning framework that has shown remarkable success in learning rich and discriminative representations. During training, the SimCLR model learns to maximize agreement between differently augmented views of the same image while

minimizing agreement with other images in the dataset. This encourages the model to capture high-level features and semantics present in the data.

Once the SimCLR model is trained, we extract the learned features from the base encoder $f(.)$, which serves as a rich representation of the input images.

4.3.2 Integration with Input-Aggregation Methods

With the self-supervised features in hand, we proceed to add them with existing features obtained through different input elicitation methods (see Section 3.1.1). Specifically, we focus on enhancing the performance of a group of machine learning models, including k-nearest neighbors (k-NN), logistic regression, Random Forest (RF), and support vector machines (SVM) classifiers.

For each of the shallow machine learning models, we augment the original set of input features with the learned self-supervised features from the SimCLR model. This augmentation effectively adds a 256-dimensional feature space, incorporating the additional information captured by the self-supervised features. By doing so, we aim to enrich the input data and enable the models to leverage the complementary knowledge contained within the self-supervised features.

4.3.3 Workflow Overview

The proposed workflow can be summarized as follows:

1. Train a SimCLR model on a large dataset of unlabeled images to learn self-supervised features.

2. Extract the learned self-supervised features from the base encoder $f(.)$ of the SimCLR model.

3. For each shallow machine learning model (k-NN, logistic regression, random forest, SVM), combine the original input features with the self-supervised features.

4. Perform model training and evaluation using the augmented feature set.

5. Compare the performance of the models with and without the self-supervised features to assess the improvement in input-aggregation methods.

4.4 Dataset

A collection of 100,000 images was generated using the image generation method described in Section 3.1.2, which serves as the basis for self-supervised learning. The code to generate the images is available at `https://github.com/O-ARE/2D-Image-Generation-HCOMP` [78]. The dataset consists of positive and negative samples, where positive images have a 2D image of a bat in different shapes and orientations among other objects (see Table 4.1). While the labels for these images are available during generation, the self-supervised model did not use these labels to learn discriminative image features.

The images underwent preprocessing steps to ensure their suitability for training the self-supervised model. This included data cleaning, normalization, and resizing of images to a consistent resolution to facilitate efficient training and feature extraction. For details, please see Section 3.1.2.

4.5 Experimental Setup

4.5.1 Self-supervised Training

To facilitate self-supervised learning, the SimCLR model employed augmentation techniques to increase the diversity of the training data. Images larger than the desired input size were resized to 256×256 and cropped to focus on multiple areas of the image. Additional flipping and rotation transformations were applied to introduce further variations in the data.

The model was trained using the Adam optimizer [76] with a small learning rate. The

Table 4.1: Sample images from the dataset

learning rate was dynamically adjusted if the validation loss did not exhibit consecutive improvement over a predefined number of epochs. This optimization strategy ensured effective convergence and prevented overfitting.

Throughout the training process, relevant metrics were monitored and logged periodically. The PyTorch Lightning framework facilitated model training, providing checkpointing, logging, and learning rate monitoring functionalities. Following are the hyperparameters used during the training.

Table 4.2: Hyperparameters

Hyperparameter	**Value**
Batch Size	256
Hidden Dimensions	256
Learning Rate	5e-4
Temperature	0.07
Weight Decay	1e-4
Maximum Epochs	1300

4.5.2 Feature Extraction

Once the self-supervised model was trained on the unlabeled dataset, it was utilized to extract features from the images in the target dataset. The target dataset is the dataset used in Chapter 3 to train and evaluate the shallow machine learning-based input aggregation methods. The extracted features represented the learned representations obtained from the self-supervised learning process that used image augmentation to help the SimCLR model identify discriminative features.

The self-supervised features were incorporated into the existing features that were collected via input elicitation methods (see Chapter 3) to create a comprehensive feature representation for each image in the target dataset. The input elicitation methods are:

1. Binary Choice Elicitation (BCE)
2. Spatial Elicitation (SE)
3. Confidence Elicitation (CE)
4. Guess of Majority Elicitation (GME)
5. Perceived Difficulty Elicitation (PDE)

This fusion of features aimed to leverage both the self-supervised representations and the existing features to enhance the performance of the subsequent input aggregation models.

4.5.3 Training Shallow Machine Learning Models

The aggregated features, comprising the self-supervised and existing features, were used to train shallow machine learning models. Specifically, k-nearest neighbors (KNN), logistic regression, Random Forest (RF), and support vector machines (SVM) classifiers were employed. These models learned decision boundaries based on the combined features and aimed to classify the images in the target dataset accurately.

4.6 Results

4.6.1 Self-Supervised Training

The SimCLR model demonstrated strong performance and effectively learned discriminative image representations during the self-supervised training.

Figure 4.3 showcases a t-SNE (t-distributed Stochastic Neighbor Embedding) plot that assesses the quality of the learned representations from the SimCLR model. By applying the t-SNE algorithm, the high-dimensional representations obtained from the model were projected into a two-dimensional space, revealing the distribution of the images. In the resulting t-SNE plot, we observed a slight clustering of the images into two groups representing the positive and negative classes. While perfect clustering is not expected in a 2D projection, the observed separation in the t-SNE plot indicates that the learned features encode crucial characteristics distinguishing positive and negative instances.

The t-SNE plot's clustering pattern further supports the adoption of the self-supervised training approach. The slight separation of the positive and negative images in the two-dimensional space indicates that the SimCLR model has learned meaningful representations that align with the class labels. This finding suggests that self-supervised learning has successfully captured important discriminatory information in the image representations. The observed clustering provides valuable insights into the model's ability to learn representations that facilitate subsequent classification tasks.

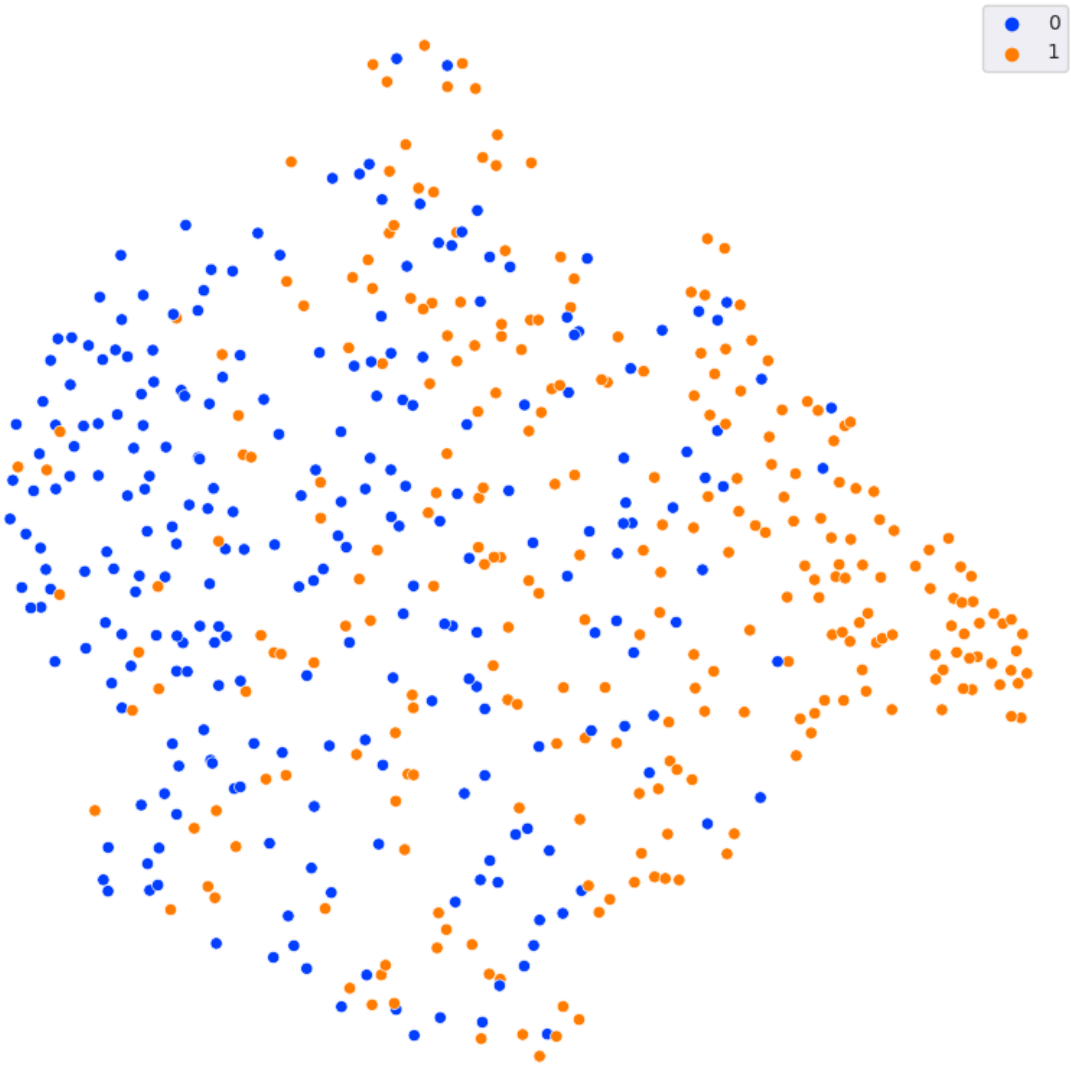

Figure 4.3: t-SNE plot of self-supervised features learned by the SimCLR model

4.6.2 Performance of the ML Models

This section presents the evaluation and comparison of various machine learning (ML) models in terms of their performance with and without the inclusion of self-supervised features. Performance metrics, F1 score, false negative rate (FNR), and area under the receiver operating characteristic curve (AUC) are used to assess the models across input aggregation methods.

k-Nearest Neighbor

Table 4.3: Performance analysis of the k-Nearest Neighbor model with and without self-supervised features

Input Elicitation	(-) Self-supervised feat.			(+) Self-supervised feat.		
	F1	FNR	AUC	F1	FNR	AUC
BCE	**0.73**	**0.38**	**0.82**	0.55	0.62	0.79
BCE-CE	**0.75**	**0.38**	**0.89**	0.59	0.58	0.80
BCE-SE	**0.81**	**0.29**	**0.83**	0.65	0.5	0.79
BCE-PDE	**0.81**	**0.29**	**0.83**	0.53	0.62	0.76
BCE-CE-SE	**0.76**	**0.33**	**0.92**	0.63	0.54	0.83
BCE-CE-PDE	**0.81**	**0.29**	**0.90**	0.63	0.54	0.82
BCE-CE-SE-PDE	**0.81**	**0.29**	**0.92**	0.68	0.46	0.79

BCE - Binary Choice Elicitation, CE - Confidence Elicitation, SE - Spatial Elicitation, PDE - Perceived Difficulty Elicitation

Table 4.3 showcases the performance analysis of the k-Nearest Neighbor (KNN) model with and without self-supervised features. Comparing the F1 scores, we observe that the KNN model without self-supervised features generally outperforms the model with self-supervised features. The F1 scores range from 0.73 to 0.81 for the KNN model without self-supervised features, while the scores range from 0.55 to 0.68 for the model with self-supervised features. These findings suggest that the inclusion of self-supervised features did not significantly enhance the KNN model's ability to balance precision and recall in the classification task.

Analyzing the false negative rate (FNR), it is noted that the model with self-supervised features exhibited higher FNR values than the model without self-supervised features. This indicates that the inclusion of self-supervised features might have negatively affected the KNN model's capability to correctly identify positive instances, leading to a higher number of false negatives.

Regarding the area under the receiver operating characteristic curve (AUC), the results indicate no consistent improvement with the incorporation of self-supervised features. This shows that the inclusion of self-supervised features did not consistently enhance the KNN

model's ability to discriminate between positive and negative instances.

It is important to note that we refrained from conducting any scaling operation on the features utilized to train the k-nearest neighbors (kNN) model. Due to the model's sensitivity to scaling, the performance of the model did not improve after adding the self-supervised features.

Logistic Regression

Table 4.4: Performance analysis of the Logistic Regression model with and without self-supervised features

Input Elicitation	(-) Self-supervised feat.			(+) Self-supervised feat.		
	F1	FNR	AUC	F1	FNR	AUC
BCE	0.78	0.33	0.79	**0.81**	**0.29**	**0.94**
BCE-CE	0.81	0.29	0.90	**0.84**	**0.25**	**0.93**
BCE-SE	0.84	0.25	0.95	**0.85**	**0.21**	**0.96**
BCE-PDE	0.76	0.33	**0.94**	**0.77**	**0.25**	0.93
BCE-CE-SE	0.81	0.29	0.92	0.81	**0.21**	**0.95**
BCE-CE-PDE	0.81	0.29	0.86	0.81	**0.21**	**0.96**
BCE-CE-SE-PDE	0.81	0.29	0.86	0.81	**0.21**	**0.96**

Table 4.4 presents the performance analysis of the Logistic Regression model with and without self-supervised features. Comparing the F1 scores, it is evident that the Logistic Regression model with self-supervised features shows equal or better performance compared to when it was trained without self-supervised features. The F1 scores range from 0.81 to 0.85 when self-supervised features are included, indicating a significant improvement compared to the range of 0.76 to 0.84 observed without self-supervised features. This enhancement suggests that leveraging self-supervised learning effectively improves the model's ability to balance precision and recall in the classification task.

Moreover, the false negative rate (FNR) is reduced when self-supervised features are incorporated. The model with self-supervised features consistently exhibits lower FNR values, indicating its improved capability to identify positive instances accurately and

minimize false negatives. This enhancement highlights the positive impact of self-supervised learning on the Logistic Regression model's ability to capture and utilize meaningful features in the classification process. With AUC scores ranging from 0.93 to 0.96, the model with self-supervised features demonstrates enhanced discriminative abilities compared to the range of 0.79 to 0.94 observed without self-supervised features.

Random Forest

Table 4.5: Performance analysis of the Random Forest model with and without self-supervised features

Input Elicitation	(-) Self-supervised feat.			(+) Self-supervised feat.		
	F1	FNR	AUC	F1	FNR	AUC
BCE	**0.73**	0.33	0.81	0.71	0.33	**0.85**
BCE-CE	**0.78**	0.33	**0.86**	0.77	0.35	0.85
BCE-SE	0.76	0.29	0.87	**0.84**	0.29	**0.90**
BCE-PDE	0.68	0.38	0.81	**0.75**	0.33	**0.84**
BCE-CE-SE	0.77	0.29	**0.90**	**0.82**	0.29	0.88
BCE-CE-PDE	0.79	0.29	0.86	**0.82**	0.29	**0.87**
BCE-CE-SE-PDE	0.81	0.29	0.90	**0.85**	**0.25**	**0.93**

Table 4.5 illustrates the performance analysis of the Random Forest model with and without self-supervised features. In terms of the F1 scores, the Random Forest model's performance improved when we added the self-supervised features. Without self-supervised features, the F1 scores range from 0.68 to 0.81; with self-supervised features, the scores range from 0.71 to 0.85. These findings suggest that the inclusion of self-supervised features led to a consistent improvement in the F1 scores of the Random Forest model.

Analyzing the false negative rate (FNR), we also observe that the model's performance improved with the inclusion of self-supervised features. Without self-supervised features, the FNR values range from 0.29 to 0.38, while with self-supervised features, the values range from 0.25 to 0.35. These results indicate that the inclusion of self-supervised features enhanced the model's ability to identify positive instances correctly and reduced the number

of false negatives.

Considering the area under the receiver operating characteristic curve (AUC), the inclusion of self-supervised features consistently led to improved performance. Without self-supervised features, the AUC scores range from 0.81 to 0.90; with self-supervised features, the scores range from 0.84 to 0.93. These findings indicate that the inclusion of self-supervised features enhanced the Random Forest model's discriminative abilities, resulting in better separability between positive and negative instances.

The observed improvements in performance resulting from the incorporation of self-supervised features can be attributed to several underlying reasons. Unlike the k-nearest neighbor model, the random forest model was able to leverage the additional features of the inherent structure of the images. By leveraging self-supervised features, the model gains a richer understanding of the images underlying patterns and variations and makes more informed decisions during the classification process, resulting in a reduction in false negatives, as observed in the lower false negative rate (FNR). The self-supervised features likely capture nuances in the images that were not effectively represented by the original input elicitation-based feature, enabling the model to better handle complex instances that were previously challenging to classify correctly.

Therefore, the results demonstrate that the inclusion of self-supervised features significantly improved the performance of the Random Forest model in terms of F1 score, false negative rate, and area under the receiver operating characteristic curve. This finding emphasizes the effectiveness of leveraging self-supervised learning to enhance the model's ability to classify instances in the classification task accurately.

Support Vector Machine

According to Table 4.6, the Support Vector Machine (SVM) consistently achieves higher scores when self-supervised features are included. Without self-supervised features, the F1 scores range from 0.73 to 0.80; with self-supervised features, the scores range from 0.81 to 0.84. The FNR values range from 0.25 to 0.38 without self-supervised features and remain

Table 4.6: Performance analysis of the Support Vector Machine with and without self-supervised features

Input Elicitation	(-) Self-supervised feat.			(+) Self-supervised feat.		
	F1	FNR	AUC	F1	FNR	AUC
BCE	0.73	0.38	0.92	**0.83**	**0.29**	**0.93**
BCE-CE	0.80	0.33	0.90	**0.84**	**0.29**	**0.94**
BCE-SE	0.84	**0.25**	0.86	0.84	0.29	**0.94**
BCE-PDE	0.77	0.38	0.90	**0.82**	**0.29**	**0.92**
BCE-CE-SE	0.81	0.29	0.88	0.81	0.29	**0.94**
BCE-CE-PDE	0.80	0.33	0.90	**0.81**	**0.29**	**0.94**
BCE-CE-SE-PDE	0.81	0.29	0.86	**0.84**	**0.25**	**0.94**

relatively consistent at 0.29 with self-supervised features.

Considering the area under the receiver operating characteristic curve (AUC), it is found that the inclusion of self-supervised features consistently improves the SVM model's discriminative abilities. Without self-supervised features, the AUC scores range from 0.86 to 0.92; with self-supervised features, the scores range from 0.93 to 0.94.

These findings indicate that the incorporation of self-supervised features leads to improved classification accuracy of the SVM model, as higher F1 scores reflect a better balance between precision and recall.

The results support the hypothesis that the inclusion of unsupervised features through self-supervised learning can enhance the performance of the evaluated classifiers. The improved AUC scores demonstrate the effectiveness of incorporating self-supervised features in enhancing the classifiers' ability to distinguish between positive and negative instances accurately.

In summary, the results of our experiments provide evidence that the inclusion of self-supervised features improves the AUC scores of the Logistic Regression, Random Forest and Support Vector Machine models, signifying enhanced discriminative power. However, for the k-Nearest Neighbor models, the impact of self-supervised features on performance is negative, with no consistent improvement observed across all input elicitation methods.

But despite the fact that KNN is sensitive to feature scaling and hence did not perform well after adding the self-supervised features, the other models' performance highlight the potential of self-supervised learning to enhance input aggregation methods in crowdsourcing scenarios, improving the reliability and accuracy of derived labels. Future research could further explore the integration of self-supervised learning in other machine learning models and investigate additional input aggregation techniques to uncover further improvements in performance.

4.7 Conclusions

The results of our experiments provide valuable insights into the impact of self-supervised learning on the performance of these models. Overall, the findings suggest that including self-supervised features can positively affect most of the ML models.

For the k-Nearest Neighbor model, the inclusion of self-supervised features did not significantly enhance its performance. The F1 scores remained relatively consistent, and in some cases, the model without self-supervised features outperformed the model with self-supervised features. This suggests that self-supervised features did not substantially improve the classification task's balancing precision and recall.

In contrast, the Logistic Regression model demonstrated consistent improvements with the inclusion of self-supervised features. The F1 scores increased, indicating better overall classification accuracy. The false negative rate decreased, indicating an enhanced ability to identify positive instances correctly. Additionally, the AUC scores improved, highlighting better discriminative abilities of the model. These results emphasize the potential of self-supervised learning in enhancing the Logistic Regression model's performance.

The Random Forest model showed mixed results. While the F1 scores remained relatively consistent, the inclusion of self-supervised features led to improved performance in terms of false negative rate and AUC scores. This suggests that self-supervised features contributed to better identification of positive instances and enhanced separability between positive and

negative instances in the classification process.

Similarly, the Support Vector Machine model demonstrated consistent improvements with the inclusion of self-supervised features. The F1 scores increased, indicating improved classification accuracy. The false negative rate decreased, indicating a better ability to identify positive instances. Moreover, the AUC scores improved, indicating enhanced discriminative abilities. These findings highlight the effectiveness of self-supervised learning in enhancing the performance of the Support Vector Machine model.

While our experiments provided valuable insights into the impact of self-supervised learning on input aggregation models, there are limitations to be considered. First, our study focused on a specific set of input elicitation methods and ML models, and the findings may not be generalized to all possible combinations. Further research is needed to explore the effectiveness of self-supervised features across a wider range of input elicitation methods and ML models.

Another limitation is the reliance on a specific dataset and task. Our experiments were conducted on a specific crowdsourcing dataset, and the results may vary for different datasets and tasks. It would be beneficial to conduct similar experiments on diverse datasets to validate the generalizability of the findings.

In conclusion, our study demonstrated that including self-supervised features could positively affect the performance of input aggregation models in crowdsourcing scenarios. While the impact varied across different ML models, the findings suggest that self-supervised learning has the potential to enhance classification accuracy, improve the identification of positive instances, and enhance the discriminative abilities of the models. These findings contribute to the understanding of leveraging self-supervised learning in improving input aggregation methods. Future research should focus on exploring the effectiveness of self-supervised features on a broader range of input elicitation methods, ML models, and diverse crowdsourcing tasks to further advance the field.

Chapter 5

Optimization of Task–Assignment

5.1 Introduction

The task assignment problem is a type of optimization problem in which a set of tasks must be assigned to a set of agents in a way that optimizes one or more objective functions. For example, consider an image classification task where the agents must classify the images correctly. Given a pool of unlabeled images, there is a set of tasks $T = \{t_1, t_2, \ldots, t_n\}$ for n images and a set of m agents, $A = \{A_1, A_2, \ldots, A_m\}$. Each task t_i has a certain set of requirements (e.g., time, cost), and each agent A_j has a certain set of capabilities (e.g., availability, skill level). The objective is to find an assignment of tasks to agents that maximizes the overall performance of these agents. The objective function $f(X)$ is a function that measures the quality or fairness of the assignment. It may depend on various factors, such as the quality of the assignments (e.g., task completion time, cost, quality), the fairness of the assignments (e.g., workload balance, agent preferences), or some combination of these factors. It can take different forms depending on the specific problem and the desired optimization criteria. For example, in the image classification problem, the objective function might be the overall classification accuracy of the images. Constraints can be added to the task assignment problem to satisfy certain requirements or limitations. For example, constraints might be added to ensure that each agent is assigned one task at most.

Task assignment is a fundamental problem in many fields and has many real-world applications. For example, in job scheduling, the task assignment problem could be assigning tasks to employees to maximize their productivity while minimizing the cost of overtime or the need for additional workers. In resource allocation, the task assignment problem could

be to assign resources to projects to maximize their utilization while minimizing the cost of procurement. In project management, the task assignment problem could be assigning tasks to team members to maximize the project's progress while minimizing the risk of delays.

Conventional methods employed for solving task assignment problems, such as linear programming and integer programming, have some limitations [81, 82, 83, 84, 85]. One of the main limitations of these approaches is that they require strong assumptions about the given problem and the objective function. For example, linear programming requires that the objective function is linear and the problem constraints are linear as well [86]. This limits the applicability of linear programming to problems with simple, linear structures. In contrast, the task assignment problem often involves complex, nonlinear relationships between tasks, agents, and objectives, which are difficult to capture using linear programming.

Moreover, scalability presents a further concern for these approaches [87, 88, 89]. As the problem size expands, the number of potential task assignments increases exponentially, rendering the identification of the optimal solution within a reasonable timeframe increasingly difficult. This scalability issue significantly restricts the practicality of conventional methods when confronted with large-scale problem instances.

Recently, there has been growing interest in using neural networks to solve task assignment problems. Neural networks have shown great promise in solving complex optimization problems, thanks to their ability to learn complex patterns and generalize to new scenarios [3, 90]. In particular, recent research [91, 92, 93] has shown that neural networks can be used to find optimized task assignments in a variety of settings of crowdsource-based environments.

In this study, we propose a novel approach to solving the task assignment problem using neural networks. Our approach leverages the power of deep learning to learn the agents' ability directly from data without relying on handcrafted features or assumptions. The neural networks model the task complexity and the agent's ability to perform the tasks successfully, and we use this model with the Hungarian algorithm [94] to get the optimized task assignment that maximizes the overall performance of the simulated agents.

Our proposed approach offers several advantages. It follows a simple workflow, divided into two main stages, which contributes to its practicality and ease of implementation. It is highly flexible and can be applied to real-world scenarios. It can handle complex and uncertain scenarios where traditional optimization techniques often fail. Besides, our approach is scalable, enabling it to handle large-scale task assignment problems involving tens or hundreds of agents.

5.2 Hungarian Algorithm

The Hungarian algorithm, also known as the Munkres algorithm or the Kuhn-Munkres algorithm [94], is a combinatorial optimization algorithm used to solve the assignment problem. It efficiently finds the optimal assignment of agents to tasks in a bipartite graph while minimizing the total cost or maximizing the total profit of the assignments.

The Hungarian algorithm operates based on a cost matrix, where each entry represents the cost of assigning a particular agent to a specific task. The algorithm iteratively finds a series of augmenting paths in the graph and updates the assignment to reduce the total cost. It efficiently finds the optimal solution in $O(n^3)$, where n is the number of agents or tasks.

Here is a pseudocode representation of the Hungarian algorithm:

1. Subtract the minimum value in each row from all the elements in that row.
2. Subtract the minimum value in each column from all the elements in that column.
3. Cover the zeros in the cost matrix using the minimum number of lines (horizontal or vertical lines) to cover all zeros. If the number of lines equals the number of rows or columns, a feasible assignment is found, and the algorithm terminates. Otherwise, proceed to step 4.
4. Find the minimum uncovered value in the cost matrix and subtract it from all the uncovered elements. Add it to all elements at the intersection of covered rows and columns.

5. Repeat steps 3 and 4 until a feasible assignment is found.

Below is an example probability matrix where A_i are the agents, and P_j are the corresponding probability of success for the task. The optimal assignment is obtained using the Hungarian algorithm for maximization, a variation of the standard algorithm used to find the optimal assignment.

Table 5.1: Probability Matrix

	P_1	P_2	P_3	P_4
A_1	0.82	**0.83**	0.70	0.92
A_2	**0.77**	0.37	0.49	0.92
A_3	0.11	0.35	0.05	**0.86**
A_4	0.08	0.09	**0.98**	0.23

Given the probability matrix above, image I_1 will be assigned to agent A_2, I_2 to A_1, I_3 to A_4, and I_4 to A_3.

5.3 Proposed Framework for Optimized Task Assignment

For our study, we opted for a binary classification task that requires agents to classify images featuring a target object. If the object is present in the image, the agents classify it as positive, otherwise as negative. We have n images to classify, with each agent assigned only one image per classification round. At each round, m agents evaluate m images, meaning all images will be classified over n/m rounds. No new tasks are assigned until all m agents complete their assigned tasks, ensuring equal task distribution among all agents.

To optimize the task assignment, we propose a framework that follows the following steps:

1. Train "predictor" networks on the agents' historical data on past performance.

2. Estimate the probability of success for each agent for a given task.

3. Apply the Hungarian algorithm to assign the tasks.

The phases require us to have two entities:

1. Agents - that perform the tasks. We simulate human agents using a set of deep neural networks. These networks are trained on synthetic images to recognize the target object (see Section 5.5.1).

2. Predictors - that estimate the success probability of the agents. Once the agents are trained, they are evaluated on the test set. Based on this performance data, we train a deep neural network for every agent, called "predictor", to predict the labels given a set of images. These networks receive images and the corresponding binary values that indicate failure/success for the agent as input and learn the relationship between the complexity of a given task (in this case, the image complexity) and the agent's past performance.

The benefits of using neural networks in optimized task assignments are two-fold. First, neural networks can be trained on historical data, which can capture the idiosyncrasies of each agent's performance. This can improve the accuracy of the predictions and lead to better task assignments. Second, using neural networks can significantly reduce the time complexity of the optimization problem, as the neural network can quickly compute the probabilities of success for each agent on each task.

To classify a set of unlabeled images, we use the predictors to estimate the success probability of the agents. We provide the images $(I_1, I_2, \ldots, I_n)$ as input to these predictors $(P_1, P_2, \ldots, P_m)$ and receive a probability matrix, M, where the probability in each cell, P_{ij} represents the probability of a successful classification for the agent A_i for the image I_j. The workflow is similar to the example given in Section 5.2 where Table 5.1 shows a similar cost matrix. Since this is a one-to-one assignment, at each step, we take m images and the corresponding probability matrix and apply the Hungarian algorithm to optimize for

maximum classification accuracy. According to the assignment given by the algorithm for m images, we use m simulated agents to predict the class. We continue the process until we classify all the images.

Our baseline is a randomized task assignment, assigning the images randomly to the agents. We compare the overall performance of both randomized and optimized task assignments in terms of overall accuracy.

5.4 Task Description

In this study, we used baggage inspection at the security checkpoints as an example task, involving the examination of X-ray images by multiple agents to detect unwanted objects. Given the scarcity of publicly available X-ray image datasets, we generated our own dataset of synthetic images to address this limitation.

5.4.1 Datasets

We generated multiple datasets, $D = \{d_1, d_2, \ldots, d_m\}$ for m agents. Each dataset had three sets: training, validation, and test, each containing $50,000$ X-ray images. The image generation method is described in the following section.

Synthetic X-ray Image Generation

Our synthetic X-ray images were generated using 3D objects. The 3D objects are described in STL (STereoLithography) file format, which is an openly documented format for describing the object surface as a triangular mesh. The image generation method was divided into two main steps: 1. Voxelization and 2. Packing algorithm. To simplify the packing of objects, they were voxelized after a random rotation around the x, y, and z axes. Then in the packing algorithm, an exhaustive search was performed to find suitable places for the objects in the 3D space. After that, false-colored X-ray images were generated for the top view.

Figure 5.1: 3D objects

Voxelization

The voxelization process transforms a surface-based description of an object into a volume-based description in the form of a 3D array. Objects are rotated randomly before they are voxelized, producing a fixed orientation for the object. The degree of rotation around the x, y, and z axes are drawn from a normal distribution $N(\mu = 0, \sigma = 5)$ such as:

$$\begin{aligned} \theta_{x,y} &\sim N(\mu, \sigma^2) \\ \theta_z &\sim \ \ c \ \epsilon \ \{0, 90, 180, 270\} + N(\mu, \sigma^2) \end{aligned} \tag{5.1}$$

that allows for more rotation around the z-axis compared to the other two. The output array is populated based on imaginary lines that pass through the object along with one of the three axes. We assign 1 to a voxel if the corresponding line segment is inside of an object, and 0 if the segment is outside.

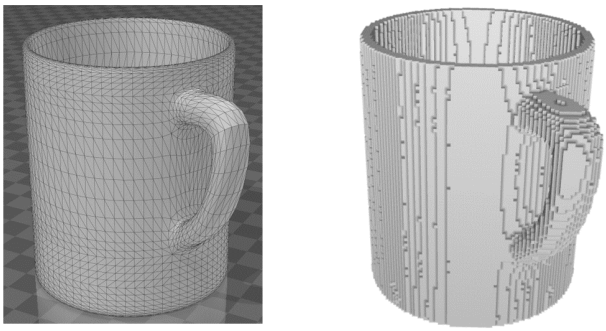

Figure 5.2: A 3D object in mesh and voxel format

Packing algorithm

The packing algorithm is designed to mimic the process of packing multiple objects in a box. Using the algorithm, we search the 3D space to find suitable locations for the objects within the box. The search space is divided into grids, with gridlines separated by a stride value specified in the input JSON file. A grid point is considered suitable for an object if it is the lowest available point and if the object does not overlap with others or go beyond the box's boundaries. However, the algorithm cannot guarantee that the same number of objects will be packed every time due to the random orientation and placement of the objects. If the packing sequence does not include the target object that the agents need to classify, it is discarded, and corresponding X-ray images will not be generated.

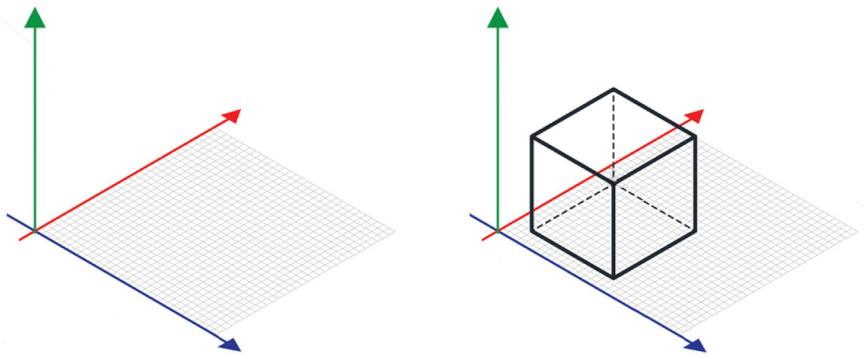

Figure 5.3: Placing voxels in the 3D space

Image generation

The X-ray inspection system comprises an X-ray source and detector, where the source emits X-rays of high and low photon energies that penetrate the baggage. The detector captures the photons and produces two energy images to analyze the material type of the objects. Due to the distinct attenuation characteristics of materials at different energy values in the spectrum, two-energy X-ray detectors can identify the atomic numbers of the objects by absorption. Subsequently, the raw X-ray image can be color-mapped with false colors that represent the material types. However, [since physical ACC. wasn't necessary and beyond ehte schope , we use an approx.] to avoid physics-based simulations that are beyond the scope of this study, we adopt a different approach to image generation. Our algorithm generates a single X-ray image by utilizing the X-ray absorption equation:

$$N/N_0 = e^{-\mu T} \tag{5.2}$$

where N_0 is the number of photons emitted by the X-ray source, N is the number of photons that pass through the object, τ is the attenuation coefficient, and T is the thickness of the object. The number of photons that pass through the object depends on the attenuation coefficient μ, which also depends on the density of the material (ρ) and mass attenuation of the material (τ). τ is experimentally obtained. A complex physics-based simulation that would produce the experimental values using the 3D objects is out of the scope of this project; hence the term $\mu(= \rho\tau)$ in the above equation is replaced by λM_c as below:

$$I = e^{-\lambda M_{c,normalized}} \tag{5.3}$$

where $I = N/N_0$, λ = decay constant. The value of λ is set to 14 for metallic objects and 4 for other types. $M_{c,normalized}$ is normalized material constant, M_c, as defined below:

Table 5.2: Material constants M_c for each RGB channel

Material Type	Material Constant (Red, Green, Blue)
Metal	(0.161, 0.486, 0.965)
Plastic	(0.957, 0.749, 0.478)
Leather	(0.318, 0.741, 0.506)
Others	(0.923, 0.341, 0.683)

$$M_{i,c,normalized} = \frac{-log(M_{i,c})}{\sqrt{\sum_j (-log M_{j,c})^2}} \tag{5.4}$$

This method produces false-colored X-ray images as in Figure 5.4.

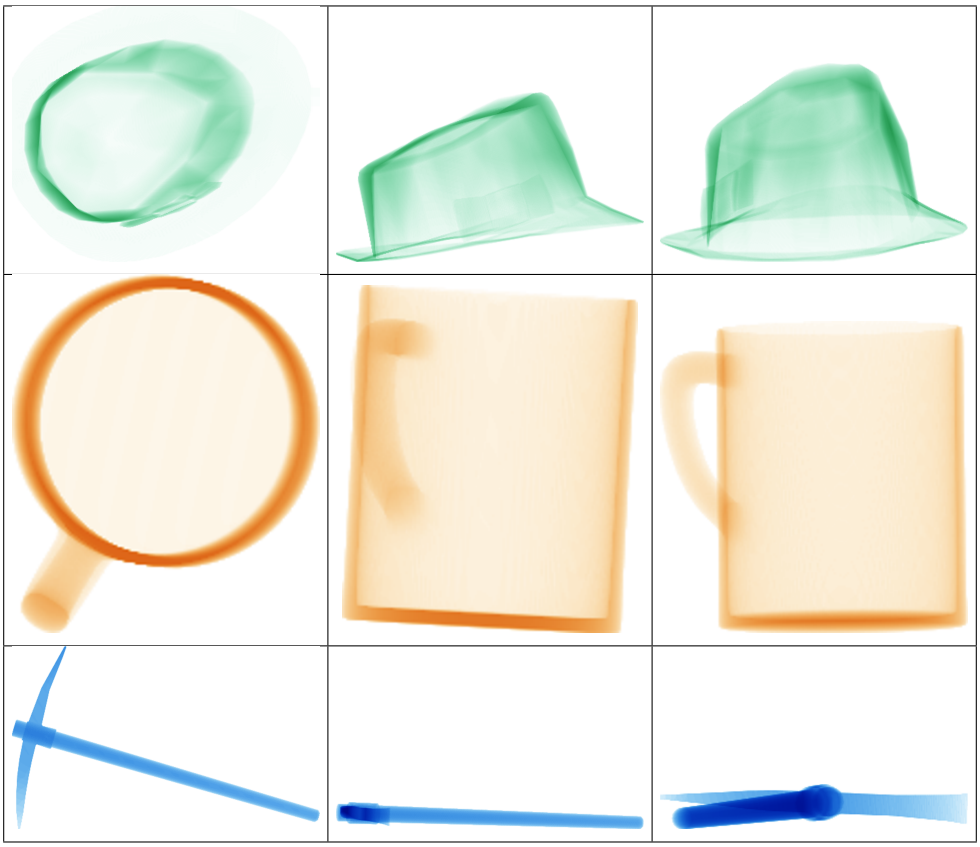

Figure 5.4: X-ray images of individual objects from different views

The first row shows a hat from three different perspectives. The second and the third row have a mug and a pickaxe. The hat is considered to be made of leather, the mug is

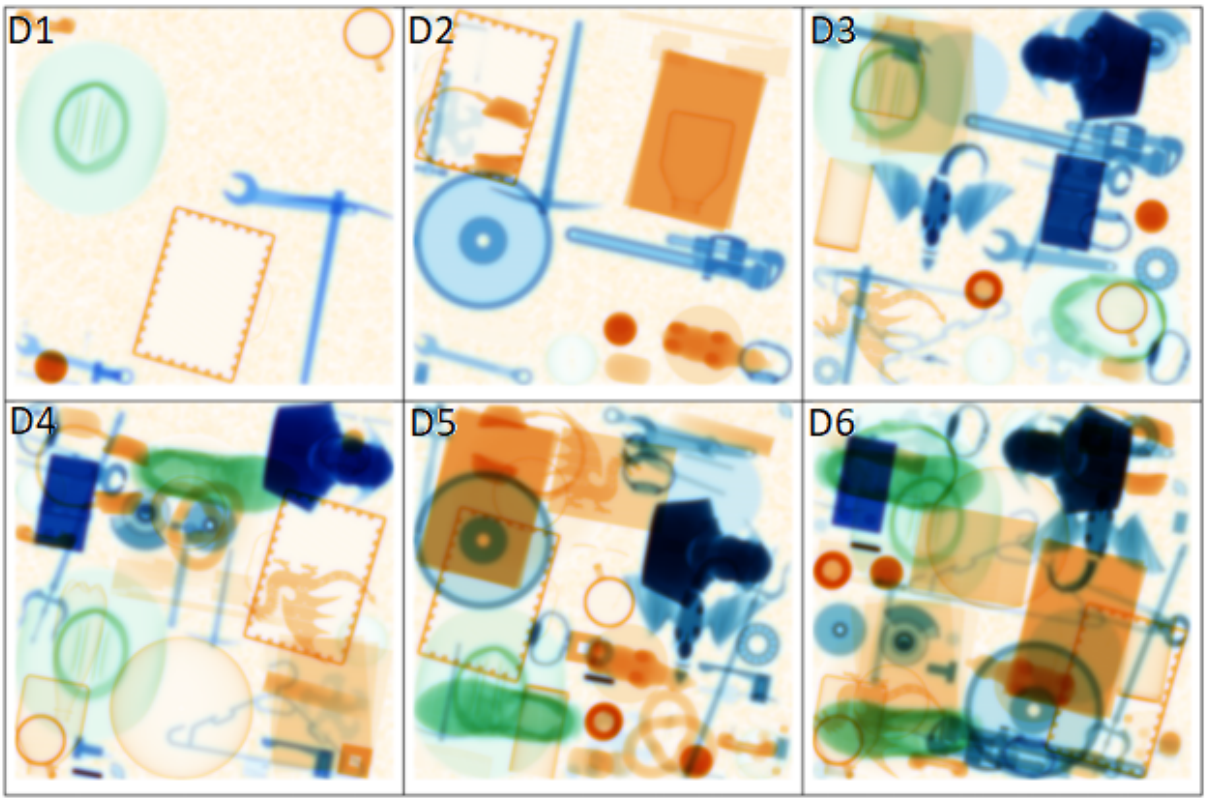

Figure 5.5: Synthetic X-ray images from all the datasets

made of plastic, and the pickaxe is made of metal. The constants defined in Table 5.2 contribute to different colors that represent material types.

For this study, we generated six datasets in total to train a maximum of six agents. The datasets' images differ in object clutteredness where d_1 is the least cluttered and d_m is the most cluttered. Figure 5.5 shows representative image samples from all the datasets.

The code to generate the X-ray images is available here `https://github.com/hassanmohsin/xray` [95].

5.5 Experiments

5.5.1 Agent Training

We hypothesize that optimized task assignment is most useful when the group of agents is diverse in terms of their level of expertise. Each agent should specialize in recognizing certain image patterns and features for the image classification task. As a starting step, we developed a baseline neural network architecture for our most naive agent, which performs slightly better than random guessing. For the rest of the agents, we relied upon off-the-shelf networks that are available in the literature. Considering the complexity level of the tasks, we chose Residual Neural Networks [96] and its subsequent variations that have shown significantly better performance over the other contemporary methods in the image classification tasks. Table 5.3 shows the network architectures used for each agent.

Table 5.3: Agent network architectures and corresponding datasets used for training

Agent	Network Architecture	Dataset
A_1	Baseline	d_1
A_2	ResNet-18	d_2
A_3	ResNet-34	d_3
A_4	ResNet-50	d_4
A_5	ResNet-101	d_5
A_6	ResNet-152	d_6

The baseline network is shown in Figure 5.6 below.

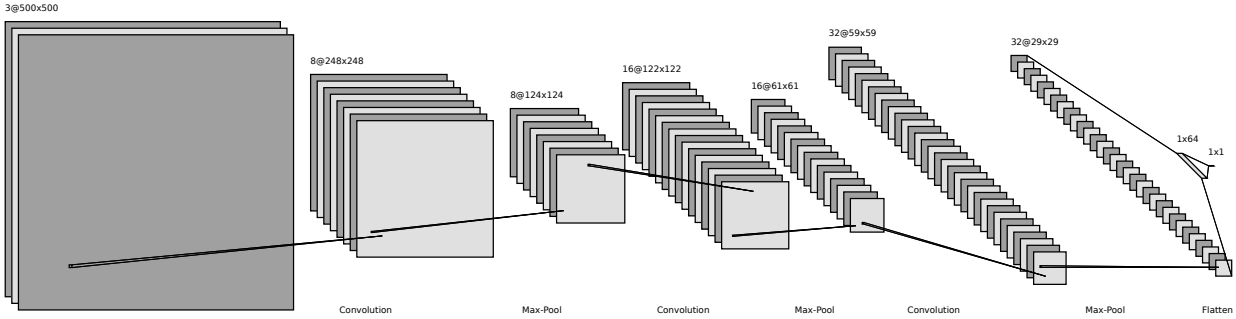

Figure 5.6: Baseline architecture

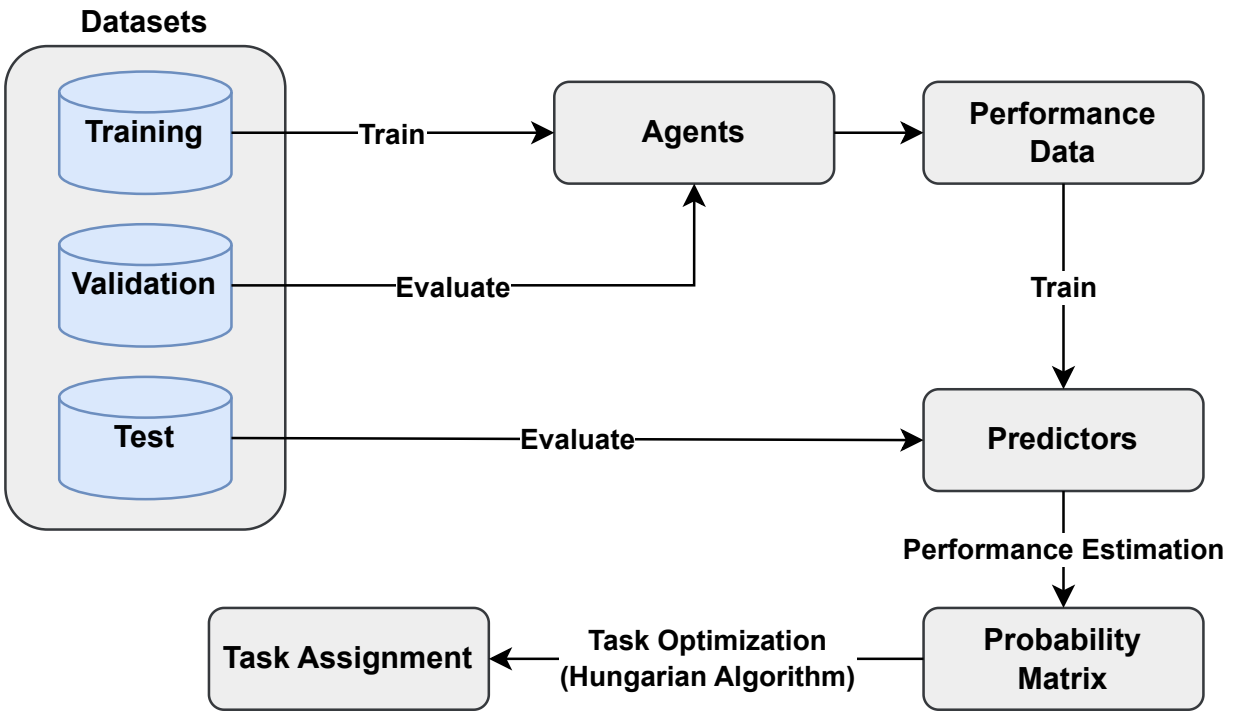

Figure 5.7: Experimental setup

We trained a total of six neural networks, each on 50,000 images from their corresponding datasets. The training set was balanced, meaning the number of positive and negative samples was the same. The training parameters for the agent models are given in Table 5.4.

Table 5.4: Training parameters for agent models

Agent model	Batch size	Epochs	Learning rate
Baseline	512	30	0.001
ResNet-18	128	30	0.001
ResNet-34	64	30	0.0001
Resnet-50	32	30	0.0001
ResNet-101	64	30	0.0001
ResNet-152	32	30	0.0001

Starting from a very simple baseline model, we increased the complexity of the networks as we trained more models. More complex neural networks have the ability to learn image features better and perform well in the classification task. Therefore, we had some agents

that were more "skilled" at the given task compared to others. This ensured that our diverse group of agents performed at different skill levels. Although the models were trained for 30 epochs, producing 30 different versions of each model, we chose the best accuracy on the validation set consisting of 300,000 images from all six datasets.

5.5.2 Predictor training

We used the baseline network architecture for all the predictor models since our goal was not to build the best possible predictor networks. Instead, our focus was to show that a neural network-based approach can be used for skill estimation of the agents and that the probability of success given by these networks can be used to optimize the task assignment. Therefore, we did not explore other network architectures or perform any hyperparameter search.

5.5.3 Task optimization

The unlabeled set contained 300,000 images, with each dataset contributing 50,000 images from their respective test sets. We employed randomized and optimized task assignment methods to classify these images, assign them to agents, and obtain their predictions. We then compared the overall classification accuracy of the agents. The number of tasks was equal to the number of images in the unlabeled set, and we divided these tasks into N/m rounds.

In each round, we used predictor networks to estimate the probability of success for m agents on m images. We then passed the resulting probability matrix (as shown in Table 5.1) to the Hungarian algorithm to obtain a one-to-one assignment. We subsequently used the agent models to predict the images' labels based on the Hungarian algorithm's assignments.

For the randomized assignment, we randomly sampled m images from the set and assigned them to the agents for prediction in each round. After obtaining all the predicted labels, we compared their performance with the optimized assignments.

5.6 Results

In this section, we present the results obtained from the experiments conducted with the agents (A_1 to A_6) performing image classification tasks on different datasets (D_1 to D_6). Table 5.5 displays the individual performance of each agent on the respective datasets, as well as the mean performance across all datasets.

Table 5.5: Agents' performance (in accuracy) on all the datasets

	D1	D2	D3	D4	D5	D6	Mean
A1	**0.986**	0.911	0.816	0.726	0.685	0.710	0.806
A2	0.996	**0.999**	0.992	0.879	0.714	0.689	0.878
A3	0.928	0.995	**0.998**	0.993	0.972	0.910	**0.966**
A4	0.856	0.848	0.956	**0.993**	0.978	0.938	0.928
A5	0.670	0.784	0.920	0.966	**0.954**	0.946	0.873
A6	0.592	0.654	0.658	0.971	0.980	**0.988**	0.870

The diagonal values in Table 5.5 represent the expected maximum performance of each agent when evaluated on their corresponding datasets. These values indicate the best possible performance an agent can achieve when classifying images from a dataset similar to the one it was trained on. As expected, the agents generally performed better on their own test datasets, and the diagonal values serve as an upper bound for their individual performance.

We anticipate the agents to exhibit good performance on datasets with simpler images compared to the ones they were trained on. For instance, agent A_5 is expected to perform well on dataset D_5, and we would expect similar performances on datasets D_1 to D_4 since these datasets consist of less complex images than D_5. Consequently, in the 6×6 matrix of Table 5.5 (shaded area), we expect the highest values along the diagonal cells, relatively close values to the diagonal in the lower triangular matrix, and lower values in the upper triangular matrix.

However, it is intriguing to observe a few lower values in the lower triangular matrix, as seen with agents A_5 and A_6. On the other hand, there are a few higher values in the

upper triangular matrix, particularly for agents A_3 and A_4. This suggests that some more "capable" agents struggled to classify simpler images, indicating a failure to generalize on easier tasks, while some less "capable" agents successfully classified more complex images than those they were trained on. As a result, agent A_3 achieved the best performance in terms of mean accuracy, followed by agent A_4, making them the top-performing agents overall.

Table 5.6: Agents' performance when the tasks are randomly assigned vs. when they are network assigned

	Random assignment	**Dataset-based assignment**	**Assigned by predictors**
A1	0.806	0.986	1.000
A2	0.878	0.999	1.000
A3	0.966	0.998	0.999
A4	0.928	0.993	0.999
A5	0.873	0.954	0.997
A6	0.870	0.988	0.996
Mean	**0.887**	**0.986**	**0.999**

Moving on to Table 5.6, we compare the random assignment performance, dataset-based assignment performance, and the actual performance achieved when the network assigned the tasks to the agents.

Initially, the tasks are randomly assigned to the agents without considering their historical performance. As seen in the previous table, agents A_3 and A_4 outperformed the other agents, while agent A_1 exhibited the lowest performance. This outcome is reasonable as agent A_1 had a relatively simple network architecture, and it was assigned tasks that were more complex in nature.

The dataset-based assignment is obtained from the diagonal values of the previous table, representing the upper bound of their individual performances. If the agents were to be evaluated on images similar to those they were trained on, their best performance would be expected, as indicated in this column.

On the other hand, when the agents were optimally assigned tasks using the network-

provided success probability and the optimization algorithm, we observed that their performance surpassed the dataset-based assignment performance, which was theoretically the upper bound for their individual performance. This finding aligns with what we observed in the previous table, indicating that some agents demonstrated the ability to classify images that were supposedly more complex than the ones they were trained on. This adaptability shows the capability of certain agents to generalize well and accurately classify images with varying levels of complexity.

Moreover, the results highlight that not all agents would consistently perform well when faced with tasks that differ in terms of complexity. Even the most capable agents may struggle due to various unknown factors. Hence, it is crucial to optimize task assignments for a group of experts with diverse levels of expertise. By doing so, we can leverage their varied skills effectively and achieve improved overall performance in image classification tasks.

5.7 Discussion

The results demonstrate the effectiveness of utilizing neural networks for task assignment optimization in achieving better performance in image classification tasks. Notably, the performance of "novice" agents can be significantly enhanced when they are assigned tasks that align with their level of expertise. By ensuring task assignments are tailored to individual competencies, the group's collective performance can be substantially elevated. In this section, we discuss the implications of these results and the limitations of our approach.

One of the key benefits of our approach is its scalability. Traditional approaches to task assignment suffer from scalability issues, particularly when the number of agents and tasks is large. Our proposed approach leverages the power of neural networks to learn the agents' level of expertise, estimate the probability of success for an unknown task and use the Hungarian algorithm, making it a viable solution for large-scale task assignment problems. Furthermore, the use of neural networks allows for the incorporation of learnable

complex features and constraints related to the agents' historical performance, enabling a more flexible and robust approach to skill estimation.

However, some limitations to our approach should be addressed in future research. One limitation arising from using complex neural networks is the need for large amounts of training data which might be scarce. Our experiments used a balanced dataset to train the neural networks. The performance of our approach on unbalanced datasets is unclear, and further research is needed to investigate this issue. Another limitation is the choice of evaluation metrics. While accuracy provides a comprehensive measure of the agents' performance, other metrics may be more appropriate for certain types of tasks, such as outlier detection and spam detection. Furthermore, we used a group of simulated agents that have a diverse level of expertise in classification tasks; the results were not verified by humans. Besides, by simulating human agents, it is impossible to capture all sorts of human behavior while performing the tasks that could affect their performance.

5.8 Conclusions

We proposed an approach to task assignment optimization using a two-step method, the estimation phase, which uses the neural networks trained on historical performance, and the assignment phase, which uses the Hungarian algorithm and applies it to image classification tasks. Traditional approaches to task assignment suffer from limitations such as scalability and inefficiency. Our proposed approach addresses these limitations by leveraging the power of neural networks to learn the agents' historical performance data and optimize the assignment of tasks to agents.

The results of our experiments show that the approach is effective in assigning images to the most appropriate agent among a set of available agents. The agents achieved high accuracy, especially the novice ones who can classify only the simpler images, demonstrating the approach's effectiveness in optimizing task assignments.

Our proposed approach can potentially be extended to other types of tasks beyond image

classification. It could also be used in a variety of settings, such as distributed computing, crowdsourcing, and multi-robot systems. However, further research is needed to explore the performance of the approach on other types of tasks and optimize the neural network's hyperparameters to achieve better results.

Chapter 6

Conclusions and Future Work

Throughout this thesis, we explored how deep learning techniques can enhance various aspects of collective intelligence methods for image classification tasks. Our research has focused on improving input aggregation methods, incorporating self-supervised features, and optimizing task assignments. We have gained valuable insights and drawn important conclusions by investigating these areas.

We have examined the use of automated classifiers to enhance input aggregation methods in crowdsourcing. Machine learning models for input aggregation methods rely on the features obtained through different input elicitation methods. Our study has demonstrated that by utilizing the output of automated classifiers as an additional feature, we can achieve significant improvements in the results. This approach allows us to use additional datasets and reduce the dependency on expert feedback, reducing the timeline to get accurate results.

Additionally, we have explored the inclusion of features learned by a self-supervised model in training the input aggregation methods. Through our experiments, we have observed the additional features' positive impact on the aggregation methods' performance. While the effectiveness varies across different models, we have witnessed improvements in classification accuracy, the identification of positive instances, and discriminative abilities. These findings highlight the potential of self-supervised learning as a valuable technique for enhancing the features obtained through different elicitation methods.

Next, we have focused on optimizing task assignments in crowdsourcing using a two-phase approach. By combining neural networks in the estimation phase and the Hungarian algorithm in the assignment phase, we have addressed common limitations associated with scalability and inefficiency. Our experiments on image classification tasks have shown that

our proposed approach effectively assigns tasks to the most suitable agents, resulting in higher accuracy than expected.

Through our research, we have uncovered the vast potential of deep learning techniques to transform and enhance many areas where crowd intelligence is used. We can enhance crowdsourcing tasks' efficiency, accuracy, and scalability by leveraging automated classifiers, optimizing task assignments, and incorporating self-supervised features. However, as with any field of study, there are opportunities for further exploration and improvement. We think further advancements can be made in the following areas.

- **Complex Modeling of Expert-Sample Pair:** In this thesis, we modeled the relationship between the complexity of samples and the expertise of the labelers in terms of the labelers' success probability. But there are other factors involved when the experts accomplish the tasks, such as associated cost, boredom, and fatigue. An advanced version of the modeling could consider these factors and capture a broad spectrum of the relationship between the experts and the tasks to accomplish better task optimization.

- **Reduction of Queries:** Additionally, the task optimization algorithm can be extended by assessing the informativeness of each sample for the expert and strategically selecting the most valuable samples for annotation, optimizing the overall crowdsourcing process in the context of active learning. By utilizing error probabilities and informativeness measures, researchers can develop efficient query strategies that reduce the total number of samples requiring expert feedback, effectively streamlining the data collection process and minimizing the labeling effort, and compare the performance of strategies with the existing methods such as uncertainty sampling [97] and query-by-committee [98].

- **Hybrid AI-Crowdsourcing platform:** Hybrid AI-crowdsourcing systems for image classification present a promising avenue for further research and development in the field of artificial intelligence and human-computer interaction. Building upon the

findings of agent adaptability and AI integration from the previous research, these hybrid systems can leverage the strengths of AI models and crowd workers to enhance the overall performance of image classification tasks.

Furthermore, during the literature review, we found many research works that focus on either the optimization algorithm or the approaches that are purely based on machine learning for optimal task assignment. However, only a few research works focus on data-driven approaches that are at the intersection between machine learning and task optimization techniques. Similar to what is presented in this thesis, data-driven approaches have the potential to tackle the challenges in this interdisciplinary research.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] Anita Williams Woolley, Ishani Aggarwal, and Thomas W Malone. Collective intelligence and group performance. *Current Directions in Psychological Science*, 24(6):420–424, 2015.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] M Mitchell Waldrop. What are the limits of deep learning? *Proceedings of the National Academy of Sciences*, 116(4):1074–1077, 2019.

[5] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[6] Danna Gurari, Diane Theriault, Mehrnoosh Sameki, Brett Isenberg, Tuan A Pham, Alberto Purwada, Patricia Solski, Matthew Walker, Chentian Zhang, Joyce Y Wong, et al. How to collect segmentations for biomedical images? a benchmark evaluating the performance of experts, crowdsourced non-experts, and algorithms. In *2015 IEEE winter conference on applications of computer vision*, pages 1169–1176. IEEE, 2015.

[7] Reid Hastie and Tatsuya Kameda. The robust beauty of majority rules in group decisions. *Psychological review*, 112(2):494, 2005.

[8] Tian Tian and Jun Zhu. Max-margin majority voting for learning from crowds. *Advances in neural information processing systems*, 28, 2015.

[9] Dapeng Tao, Jun Cheng, Zhengtao Yu, Kun Yue, and Lizhen Wang. Domain-weighted majority voting for crowdsourcing. *IEEE transactions on neural networks and learning systems*, 30(1):163–174, 2018.

[10] Gal Cohensius, Omer Ben Porat, Reshef Meir, and Ofra Amir. Efficient crowdsourcing via proxy voting. *arXiv preprint arXiv:1806.06257*, 2018.

[11] Ruth Urner, Shai Ben David, and Ohad Shamir. Learning from weak teachers. In *Artificial intelligence and statistics*, pages 1252–1260. PMLR, 2012.

[12] Stefanie Nowak and Stefan Rüger. How reliable are annotations via crowdsourcing: a study about inter-annotator agreement for multi-label image annotation. In *Proceedings of the international conference on Multimedia information retrieval*, pages 557–566, 2010.

[13] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in Neural Information Processing Systems*, 27:487–495, 2014.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[15] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[16] Jiyin He, Jacco van Ossenbruggen, and Arjen P de Vries. Do you need experts in the crowd? a case study in image annotation for marine biology. In *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pages 57–60, 2013.

[17] Jasper Oosterman, Archana Nottamkandath, Chris Dijkshoorn, Alessandro Bozzon, Geert-Jan Houben, and Lora Aroyo. Crowdsourcing knowledge-intensive tasks in cultural heritage. In *Proceedings of the 2014 ACM conference on Web science*, pages 267–268, 2014.

[18] Alexandra Swanson, Margaret Kosmala, Chris Lintott, Robert Simpson, Arfon Smith, and Craig Packer. Snapshot serengeti, high-frequency annotated camera trap images of 40 mammalian species in an african savanna. *Scientific data*, 2(1):1–14, 2015.

[19] Mahyar Salek, Yoram Bachrach, and Peter Key. Hotspotting—a probabilistic graphical model for image object localization through crowdsourcing. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[20] Giles Foody, Linda See, Steffen Fritz, Inian Moorthy, Christoph Perger, Christian Schill, and Doreen Boyd. Increasing the accuracy of crowdsourced information on land cover via a voting procedure weighted by information inferred from the contributed data. *ISPRS International Journal of Geo-Information*, 7(3):80, 2018.

[21] Kotaro Hara, Victoria Le, and Jon Froehlich. A feasibility study of crowdsourcing and google street view to determine sidewalk accessibility. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and accessibility*, pages 273–274, 2012.

[22] Sam Mavandadi, Stoyan Dimitrov, Steve Feng, Frank Yu, Uzair Sikora, Oguzhan Yaglidere, Swati Padmanabhan, Karin Nielsen, and Aydogan Ozcan. Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study. *PloS one*, 7(5), 2012.

[23] Danny Mitry, Kris Zutis, Baljean Dhillon, Tunde Peto, Shabina Hayat, Kay-Tee Khaw, James E Morgan, Wendy Moncur, Emanuele Trucco, and Paul J Foster. The accuracy and reliability of crowdsource annotations of digital retinal images. *Translational vision science & technology*, 5(5):6–6, 2016.

[24] Veronika Cheplygina and Josien PW Pluim. Crowd disagreement about medical images is informative. In *Intravascular Imaging and Computer Assisted Stenting and Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pages 105–111. Springer, 2018.

[25] Humayun Irshad, Eun-Yeong Oh, Daniel Schmolze, Liza M Quintana, Laura Collins, Rulla M Tamimi, and Andrew H Beck. Crowdsourcing scoring of immunohistochemistry images: Evaluating performance of the crowd and an automated computational method. *Scientific reports*, 7(1):1–10, 2017.

[26] Tan B Nguyen, Shijun Wang, Vishal Anugu, Natalie Rose, Matthew McKenna, Nicholas Petrick, Joseph E Burns, and Ronald M Summers. Distributed human intelligence for colonic polyp classification in computer-aided detection for ct colonography. *Radiology*, 262(3):824–833, 2012.

[27] Danny Mitry, Tunde Peto, Shabina Hayat, James E Morgan, Kay-Tee Khaw, and Paul J Foster. Crowdsourcing as a novel technique for retinal fundus photography classification: Analysis of images in the epic norfolk cohort on behalf of the ukbiobank eye and vision consortium. *PloS one*, 8(8):e71154, 2013.

[28] Andrew Mao, Ariel D Procaccia, and Yiling Chen. Better human computation through principled voting. In *AAAI*, 2013.

[29] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. *Handbook of computational social choice*. Cambridge University Press, 2016.

[30] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67, 2010.

[31] David Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. *Advances in neural information processing systems*, 24, 2011.

[32] David White, A Mike Burton, Richard I Kemp, and Rob Jenkins. Crowd effects in unfamiliar face matching. *Applied Cognitive Psychology*, 27(6):769–777, 2013.

[33] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. An evaluation of aggregation techniques in crowdsourcing. In *International Conference on Web Information Systems Engineering*, pages 1–15. Springer, 2013.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[37] Stephan Rasp, Hauke Schulz, Sandrine Bony, and Bjorn Stevens. Combining crowdsourcing and deep learning to explore the mesoscale organization of shallow convection. *Bulletin of the American Meteorological Society*, 101(11):E1980–E1995, 2020.

[38] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[39] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[40] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad

Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

[41] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[42] Ningyu Zhang, Xiang Chen, Xin Xie, Shumin Deng, Chuanqi Tan, Mosha Chen, Fei Huang, Luo Si, and Huajun Chen. Document-level relation extraction as semantic segmentation. *arXiv preprint arXiv:2106.03618*, 2021.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] David M Ryan and Brian A Foster. An integer programming approach to scheduling. *Computer scheduling of public transport urban passenger vehicle and crew scheduling*, pages 269–280, 1981.

[45] Osama Yaseen M Al-Rawi and Taniya Mukherjee. Application of linear programming in optimizing labour scheduling. *Journal of Mathematical Finance*, 9(3):272–285, 2019.

[46] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[47] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. In *Journal of Artificial Intelligence Research*, volume 4, pages 237–285, 1996.

[48] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

[50] Kalyanmoy Deb. Multi-objective optimization using evolutionary algorithms. *John Wiley & Sons*, 16(2):267–293, 2001.

[51] Juan Li and Ningji Fang. Improved genetic algorithm for multi-agent task allocation with time windows. In *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 6–11. IEEE, 2022.

[52] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

[53] Jung-Ug Kim and Yeong-Dae Kim. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. *Computers Operations Research*, 23(9):857–868, 1996.

[54] Marco Dorigo and Thomas Stützle. Ant colony optimization algorithms for the traveling salesman problem. 2004.

[55] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[56] Marco Dorigo. Swarm-bot: A novel type of self-assembling robot. In Kazuyuki Murase, Kosuke Sekiyama, Tomohide Naniwa, Naoyuki Kubota, and Joaquin Sitte, editors, *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment (AMiRE 2005)*, pages 3–4, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[57] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[58] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *arXiv preprint arXiv:2106.04560*, 2021.

[59] Veronika Cheplygina, Marleen de Bruijne, and Josien PW Pluim. Not-so-supervised: a survey of semi-supervised, multi-instance, and transfer learning in medical image analysis. *Medical image analysis*, 54:280–296, 2019.

[60] Bjorn Stevens, Sandrine Bony, Hélène Brogniez, Laureline Hentgen, Cathy Hohenegger, Christoph Kiemle, Tristan S L'Ecuyer, Ann Kristin Naumann, Hauke Schulz, Pier A Siebesma, et al. Sugar, gravel, fish and flowers: Mesoscale cloud patterns in the trade winds. *Quarterly Journal of the Royal Meteorological Society*, 146(726):141–152, 2020.

[61] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[62] Patrick McDaniel, Nicolas Papernot, and Z. Berkay Celik. Machine learning in adversarial settings. *IEEE Security Privacy*, 14(3):68–72, 2016.

[63] James Surowiecki. *The wisdom of crowds*. Anchor, 2005.

[64] Sheng Kung Michael Yi, Mark Steyvers, Michael D Lee, and Matthew J Dry. The wisdom of the crowd in combinatorial problems. *Cognitive science*, 36(3):452–470, 2012.

[65] Luke Barrington, Shubharoop Ghosh, Marjorie Greene, Shay Har-Noy, Jay Berger, Stuart Gill, Albert Yu-Min Lin, and Charles Huyck. Crowdsourcing earthquake damage assessment using remote sensing imagery. *Annals of Geophysics*, 54(6), 2012.

[66] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[67] Romena Yasmin, Md Mahmudulla Hassan, Joshua T Grassel, Harika Bhogaraju, Adolfo R Escobedo, and Olac Fuentes. Improving crowdsourcing-based image classification through expanded input elicitation and machine learning. *Frontiers in Artificial Intelligence*, 5:848056, 2022.

[68] Romena Yasmin, Joshua T. Grassel, Md Mahmudulla Hassan, Olac Fuentes, and Adolfo R. Escobedo. Enhancing image classification capabilities of crowdsourcing-based methods through expanded input elicitation. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 9(1):166–178, Oct. 2021.

[69] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[70] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

[71] Faiza Khan Khattak and Ansaf Salleb-Aouissi. Quality control of crowd labeling through expert evaluation. In *Proceedings of the NIPS 2nd Workshop on Computational Social Science and the Wisdom of Crowds*, volume 2, page 5, 2011.

[72] Sylvie Jeannin and Miroslaw Bober. Description of core experiments for mpeg-7 motion/shape. *MPEG-7, ISO/IEC/JTC1/SC29/WG11/MPEG99 N*, 2690, 1999.

[73] Naihui Zhou, Zachary D. Siegel, Scott Zarecor, Nigel Lee, Darwin A. Campbell, Carson M. Andorf, Dan Nettleton, Carolyn J. Lawrence-Dill, Baskar Ganapathysubramanian, Jonathan W. Kelly, and Iddo Friedberg. Crowdsourcing image analysis for plant phenomics to generate ground truth data for machine learning. *PLOS Computational Biology*, 14(7):1–16, 07 2018.

[74] Irwan Bello, William Fedus, Xianzhi Du, Ekin D Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021.

[75] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[77] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[78] Joshua T Grassel and Adolfo R Escobedo. 2D Image Generation. `https://github.com/O-ARE/2D-Image-Generation-HCOMP`, 09 2021.

[79] Md Mahmudulla Hassan and Olac Fuentes. 2D Image Classification. `https://github.com/O-ARE/2d-image-classification`, 12 2021.

[80] Brett Koonce and Brett Koonce. Resnet 50. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pages 63–72, 2021.

[81] Ying Zhen, Abdullah Khan, Shah Nazir, Zhao Huiqi, Abdullah Alharbi, and Sulaiman Khan. Crowdsourcing usage, task assignment methods, and crowdsourcing platforms: A systematic literature review. *Journal of Software: Evolution and Process*, 33(8):e2368, 2021.

[82] Antony E Phillips, Hamish Waterer, Matthias Ehrgott, and David M Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, 53:42–53, 2015.

[83] James K Ho. A successive linear optimization approach to the dynamic traffic assignment problem. *Transportation Science*, 14(4):295–305, 1980.

[84] Dimitri P Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of operations research*, 14(1):105–123, 1988.

[85] Jean-François Bérubé, Michel Gendreau, and Jean-Yves Potvin. An exact ϵ-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. *European journal of operational research*, 194(1):39–50, 2009.

[86] Yan Zhao, Jinfu Xia, Guanfeng Liu, Han Su, Defu Lian, Shuo Shang, and Kai Zheng. Preference-aware task assignment in spatial crowdsourcing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2629–2636, 2019.

[87] Abdullah Alfarrarjeh, Tobias Emrich, and Cyrus Shahabi. Scalable spatial crowdsourcing: A study of distributed algorithms. In *2015 16th IEEE International Conference on Mobile Data Management*, volume 1, pages 134–144. IEEE, 2015.

[88] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 77–90, 2010.

[89] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation: A set of best practices for high quality, economical video labeling. *International journal of computer vision*, 101:184–204, 2013.

[90] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[91] Lijun Sun, Xiaojie Yu, Jiachen Guo, Yang Yan, and Xu Yu. Deep reinforcement learning for task assignment in spatial crowdsourcing and sensing. *IEEE Sensors Journal*, 21(22):25323–25330, 2021.

[92] Mingze Wang, Yingjie Wang, Akshita Maradapu Vera Venkata Sai, Zhaowei Liu, Yang Gao, Xiangrong Tong, and Zhipeng Cai. Task assignment for hybrid scenarios in spatial crowdsourcing: A q-learning-based approach. *Applied Soft Computing*, 131:109749, 2022.

[93] Pengcheng Zhao, Xiang Li, Shang Gao, and Xiaohui Wei. Cooperative task assignment in spatial crowdsourcing via multi-agent deep reinforcement learning. *Journal of Systems Architecture*, 128:102551, 2022.

[94] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[95] Md Mahmudulla Hassan and Olac Fuentes. X-ray Image Generation from 3D objects. `https://github.com/hassanmohsin/xray`, 06 2021.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[97] Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113:113–127, 2015.

[98] Robert Burbidge, Jem J Rowland, and Ross D King. Active learning for regression based on query by committee. In *Intelligent Data Engineering and Automated Learning-IDEAL 2007: 8th International Conference, Birmingham, UK, December 16-19, 2007. Proceedings 8*, pages 209–218. Springer, 2007.

Chapter 7

Curriculum Vitae

Md Mahmudulla Hassan earned his bachelor's degree in Physics from the University of Dhaka, Bangladesh, in 2011. He pursued further studies and obtained a Master's in Physics and Computer Science from the University of Texas at El Paso in 2016 and 2018, respectively.

Continuing his academic journey, in the fall of 2018, Hassan joined the Ph.D. program in the same department, dedicating himself to research in the fields of machine learning and artificial intelligence. Under the guidance of Professor Dr. Olac Fuentes, he worked as a research assistant as well as a teaching assistant throughout his Ph.D. studies. Hassan's research interests primarily revolve around the intersection of machine learning and computer vision, with a particular focus on the application of deep learning in crowdsourcing-based methods.

During his academic tenure, Hassan had the opportunity to expand his expertise through internships at prominent technology companies. In 2019 and 2020, he interned at GoDaddy as a Software Engineer, where he led the development of a Deep Learning-based domain recommender system and an automated A/B testing platform. He spent the summer of 2022 at Microsoft as an intern and developed an automated instability detection tool for the Bing Search platform.

Email address: me@mhassan.net