Reducing the Cost of Test Data Labelling for Deep-Learning Systems: An Empirical Study

by

Taoyu Wu

Thesis submitted to the University of Ottawa

in partial fulfillment of the requirements for the degree of

Master of Computer Science

in

School of Electrical Engineering and Computer Science

University of Ottawa

Ottawa, Ontario, Canada

Abstract

Deep learning (DL) systems have achieved remarkable success across various domains, including healthcare, autonomous driving, and facial recognition. However, ensuring the reliability of these systems requires comprehensive testing with accurately labeled datasets, which is a resource-intensive and costly process, especially for large-scale data. This thesis presents an empirical study aimed at reducing the cost of test data labeling for deep-learning systems through automated and semi-automated methods.

We propose two approaches: Human-in-the-Loop labeling and Pseudo Labeling. The Human-in-the-Loop approach combines machine learning with human expertise, intervening only in challenging cases to ensure high labeling accuracy while minimizing human effort. The Pseudo Labeling method, on the other hand, iteratively refines labels generated by the model through repeated training cycles, offering a more automated alternative to manual labeling.

Through extensive experiments across seven diverse datasets, we evaluated these approaches against vision-based deep-learning baseline models, including VGG16, ResNet50, and ViT_13B, focusing on the trade-offs between human effort and labeling accuracy. Our results demonstrate that the Human-in-the-Loop approach achieves near-human labeling accuracy (up to 99%) while requiring less human effort than traditional methods. Conversely, Pseudo Labeling performs well in scenarios with minimal human intervention but

generally does not outperform Human-in-the-Loop or baseline models when larger amounts of labeled data are available.

This research provides valuable insights into balancing the cost of labeling with the need for high accuracy in deep-learning system testing. The findings highlight the potential of Human-in-the-Loop as a practical solution for scenarios requiring high precision and efficiency, while also recognizing the limitations of automated methods like Pseudo Labeling. Future work could explore hybrid models and further optimizations to reduce the overall cost of test data labeling while maintaining high system reliability.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my professors, Prof. Mehrdad Sabetzadeh and Prof. Shiva Nejati. Throughout my graduate studies, they provided me with patient guidance and generous support during times of difficulty, enabling me to successfully complete this thesis. They not only broadened my horizons and enhanced my skills but also contributed significantly to my personal growth. I would also like to thank the members of my lab, Baharin Aliashrafi Jodat and Mohammad Hossein Amini, who offered invaluable support and assistance, allowing me to learn many new things from them.

Finally, I want to extend my deepest thanks to my parents. Their support and encouragement in my academic pursuits enabled me to complete this unforgettable journey. Moreover, their passion for and dedication to academia have inspired my own love for scholarly work, motivating me to continue on this path.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Deep learning (DL) systems have achieved remarkable success across various applications, including healthcare, autonomous driving, and facial recognition. These systems benefit from the neural network's powerful ability to automatically learn complex feature distributions from large amounts of training data. However, using large numbers of parameters to fit target functions also gives deep neural networks a "black-box" nature, posing significant challenges to system reliability [23, 28, 44]. To ensure the reliability of these systems, it is essential to test them thoroughly with diverse and representative test datasets. A key requirement for effective testing is having accurately labeled test datasets.

For DL testing, datasets must be accurately labeled. Although manual labeling ensures high accuracy, it is costly and labor-intensive, particularly when large volumes of test data are needed.

While some interactive methods have optimized the manual labeling process [20, 28],

manual labeling becomes impractical due to its heavy demand on human, time, and financial resources as data requirements increase. In fields such as medical image annotation and product defect detection, where domain-specific expertise is required, the difficulty and cost of obtaining high-quality labels are further amplified.

Schmelzer et. al. state that the market for third-party data labeling solutions grew from $150M in 2018 to over $1B by 2023. For every $1 spent on third-party data labeling, an additional $2 is spent internally on supporting or enhancing these labeling efforts. Furthermore, the data preprocessing market alone is projected to grow from $500M in 2018 to $1.2B by 2023 [53]. These figures highlight the significant financial burden of manual labeling, reinforcing the need to explore alternative methods to reduce associated costs and effort while maintaining high labeling accuracy.

To lower the cost of obtaining labeled data, automated data labeling methods have been widely researched and applied. These methods primarily focus on generating synthetic data with built-in labels. Common approaches include generating synthetic data using generative deep learning models, perturbing original images to create new data, or generating data from specific domain-controlled models [48, 51, 59, 60, 64, 69]. However, synthetic data often suffers from quality gaps when compared to real-world datasets. Generative models face issues such as hallucination, where generated images do not align with the desired requirements [7, 40], and perturbation-based methods struggle with controlling perturbation intensity [28]. While training datasets can tolerate these errors to some extent due to the widespread use of regularization techniques that mitigate noise, test datasets require higher data reliability. Consequently, synthetic test data cannot directly inherit labels from the original data and must be relabeled.

Some supervised or semi-supervised learning methods have also been proposed to reduce the need for manual labeling [20, 64]. However, no fully automatic or semi-automatic labeling technique can guarantee perfect accuracy, often resulting in mislabeled data.

The imperfections of automated labeling methods raise a critical question: how can we balance labeling accuracy and human effort to ensure the reliability of test data labeling while minimizing human labor costs?

To address this challenge, we propose both an automated and a semi-automated labeling method. The first method employs an active learning process to assist in labeling test inputs, with human annotators intervening in difficult cases. This hybrid approach leverages both machine learning and human expertise, ensuring careful handling of the most challenging instances. The second method uses pseudo-labeling, where initial labels are generated by the model and iteratively refined through subsequent training cycles. We conducted a comprehensive empirical analysis of these two methods, evaluating their accuracy and the human effort required. Our analysis includes a detailed comparison of their performance across various datasets and scenarios, highlighting the strengths and weaknesses of each approach.

1.1 Challenges

In view of the discussion above, the specific challenges that this thesis aims to address related to automated and semi-automated labeling are as follows:

- **Designing a High-Accuracy and Efficient Test Data Labeling Tool:** In differ-

ent application scenarios, large-scale test data labeling demands not only reliable label generation but also the minimization of human resources to ensure efficiency. However, as previously mentioned, existing methods struggle to meet both requirements simultaneously. Considering the varying demands for labeled data and available human resources in practice, we propose two methods. The first method is designed for scenarios that require high accuracy and have ample human resources. It uses an active learning process to automatically label test inputs, with human intervention for difficult cases. This hybrid method ensures that challenging data is handled by humans or enhanced models, maintaining labeling accuracy. The second method is tailored for scenarios with high accuracy requirements but limited human resources.

- **Measuring Model Performance in Terms of Labeling Accuracy and Human Effort:** Since labeling accuracy and human effort are directly related, higher human effort implies more manually labeled data, which in turn can enhance model performance by providing more training data. However, as mentioned earlier, due to the different needs in different scenarios, there are several ways that the trade-off between accuracy and human effort could be addressed. We use Pareto graphs to compare and explore these trade-offs without any bias while providing a visual representation of the performance of different approaches in various scenarios.

- **Comparing Human-in-the-Loop (HIL) and Pseudo-Labeling Methods:** Human-in-the-loop methods involve human participation in labeling, meaning human effort continues to be consumed even after the initial training. Additionally, a desired accuracy rate can be set in HIL to determine which data should be handed over to

human annotators, allowing HIL to control labeling accuracy but making it difficult to manage human effort. In contrast, pseudo-labeling consumes all the human effort during labeling but cannot control labeling accuracy. To compare the two methods, we proposed a technique to control HIL's human effort.

1.2 Contributions

This research makes several key contributions to the field of deep learning system testing and data labeling:

- **Introduction of a Hybrid Active Learning Framework:** We propose an active learning process that combines human expertise with machine learning to label test data, effectively balancing accuracy and efficiency. This framework addresses the challenge of data that is difficult for models to label by involving human annotators only in those cases, thereby reducing overall costs without compromising label quality. We also present a variation of the HIL method that adapts to scenarios with limited human effort.

- **Development of a Pseudo-Labeling Method:** We introduce a pseudo-labeling approach that iteratively improves model-generated labels through repeated training cycles. This method provides a cost-effective alternative to manual labeling, reducing the need for human intervention while maintaining an acceptable level of accuracy.

- **Empirical Evaluation Across Multiple Datasets:** We conducted a comprehensive empirical analysis of the proposed labeling methods and baseline models across

various datasets, comparing their performance in terms of accuracy and required human effort. This evaluation provides valuable insights into the trade-offs involved in using these methods for DL system testing.

1.3 Organization

The remainder of this paper is organized as follows:

- **Chapter 2: Related Work** discusses the current research landscape in automatic labeling, active learning, and pseudo-labeling.
- **Chapter 3: Background** introduces the fundamental concepts and technologies involved in this research, including Deep Learning-based Computer Vision Models, Optimization Problems, Mixed-Integer Linear Programming, Clustering, Active Learning, Human-in-the-Loop Annotation, Pseudo Labeling, and Test Input Validity and Use.
- **Chapter 4: Methodology** describes the algorithmic frameworks of the proposed active learning and pseudo-labeling methods in detail.
- **Chapter 5: Experimental Strategy** outlines the research questions and experimental setup, including baselines, evaluation metrics, datasets, parameters, and training settings.
- **Chapter 6: Evaluation Results and Analysis** presents our experimental results and answers the research questions posed in Chapter 5.

- **Chapter 7: Conclusion and Future Work** summarizes the main findings and contributions of this research, and suggests future directions for improving the proposed methods and exploring new avenues in the field of automated labeling for DL system testing.

Chapter 2

Related Work

In this chapter, we review previous researches in the area of data labeling, active learning, and pseudo labeling.

2.1 Data Labeling

The success of DL systems has led to an increasing demand for labeled test data. Diverse and representative labeled datasets are essential to ensure the reliability of DL systems. Traditional data labeling methods rely on knowledgeable human experts to manually label real-world data. Although this approach yields reliable labeled data, it is extremely costly in terms of time and money [28, 59, 69]. To reduce the cost of obtaining labeled data, automated data labeling methods have been widely researched and applied [48, 51, 59, 60, 64, 69]. These automated methods primarily focus on generating synthetic data with built-

in labels using techniques such as Generative Deep Learning Models (GDLM), Raw Input Manipulation (RIM), and Model-Based Input Manipulation (MIM) [48].

2.1.1 Generative Deep Learning Models (GDLM)

Generative deep learning models create synthetic data by learning the latent space distribution characteristics of a specific domain. For instance, a model trained to generate images of cats will produce images labeled as "cat". Traditional generative models, such as Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), are commonly used to generate synthetic data [30, 37, 45, 70]. GANs, introduced by Goodfellow et al. in 2014, consist of a generator and a discriminator. Through adversarial training, the generator learns to produce realistic data samples, while the discriminator learns to distinguish between real and generated data, leading to continuous improvement in the generator's output [19]. VAEs, introduced by Kingma et al. in 2014, encode input data into a probabilistic latent space representation and sample from this distribution to generate new data. VAEs excel at generating smooth and continuous sample spaces for images, text, and other complex data. The advent of diffusion models has further enhanced the potential for generating high-fidelity synthetic data [42]. Diffusion models offer more stable and diverse training and generation processes [50], and they can use text prompts to guide image generation, making the images more semantically aligned with the labels [26]. However, these generative models are mainly used for training data synthesis due to the quality gap between generated images and real-world datasets, as training data can tolerate some errors [4, 25].

2.1.2 Raw Input Manipulation (RIM)

Another method for generating synthetic data involves perturbing original images. RIM techniques modify images in the original pixel space to create new inputs. For example, DeepXplore generates new images by adding various effects, such as simulating exposure, random occlusion, and random pixel value perturbation [43]. DLFuzz also belongs to the RIM category and generates new images by adding noise [21]. Since RIM techniques apply minor perturbations to existing valid inputs, the generated data is relatively homogeneous. This method is primarily used to simulate damaged input scenarios and is commonly employed in DL testing.

2.1.3 Model-Based Input Manipulation (MIM)

MIM methods generate data using domain-specific models. For instance, DeepJanus abstracts model instances from bitmaps to obtain scalable vector graphics (SVG) representations of handwritten digits. By modifying the control parameters of these SVG representations, desired digit images can be generated [47].

While automated data labeling methods can quickly generate data, issues such as model hallucinations and the need for reliable test datasets mean that synthetic data still requires relabeling. We propose two new automated labeling methods. The first method employs an active learning pipeline to label test inputs, involving human annotators for challenging cases. This hybrid approach leverages both machine learning and human expertise to ensure careful handling of the most challenging instances. The second method employs pseudo-labeling, where initial labels are generated by the model and iteratively refined

through subsequent training cycles. We conducted a comprehensive empirical analysis of these two methods, evaluating their accuracy and required human effort.

2.2 Active Learning

Active Learning (AL) has significant potential for reducing labeling costs. AL operates on the premise that different samples in a dataset contribute variably to model improvement. To minimize labeling expenses, the goal is to select the most valuable samples for annotation. The concept of AL was first introduced by Cohn et al., who proposed the committee machine approach [12]. Subsequently, AL has been integrated with various machine learning algorithms [11, 61].

In recent years, with the advent of big data and the enhanced capabilities of deep neural networks, Pool-Based Active Learning has gained substantial attention [31]. Common pool-based AL methods include Query-by-Committee, Expected Model Change, and Uncertainty Sampling. Query-by-Committee selects samples with the highest disagreement among pre-trained classifiers (the committee) for labeling [18, 62]. Expected Model Change evaluates the expected error rate of candidate samples and selects those with lower expected error rates for labeling [8, 17]. Uncertainty Sampling defines an uncertainty score function and selects samples with the highest label uncertainty based on this function [6]. Due to its simplicity and effectiveness, Uncertainty Sampling has been widely researched.

Greg Schohn enhanced Support Vector Machines (SVM) for document classification tasks using AL heuristics [54]. BALD (Bayesian Active Learning by Disagreement) cal-

culates sample uncertainty using Bayesian principles, and Batch BALD extends this idea with a more efficient algorithm [27, 34]. With the rise of deep learning, Wang et al. incorporated deep convolutional neural networks (CNNs) into AL, developing a cost-effective uncertainty sampling method for image classification [63]. Additionally, they proposed using an Instance Uncertainty Reweighting (IUR) module to select uncertain images, enhancing the model's generalization ability [68]. Yoo and Kweon introduced a loss prediction module attached to a CNN backbone network, which predicts training image loss based on the CNN's multi-layer outputs [67]. Choi and Yi employed Bayesian rules to train a Variational Autoencoder (VAE), selecting images with the highest misclassification probabilities for the next training round, addressing class imbalance issues in AL [10]. Beluch et al. proposed an AL method using CNN ensembles for image classification, which performs well, particularly on highly imbalanced medical datasets [6].

Building on these approaches, we propose an AL method based on ensemble model confidence and linear programming. Our method uses a discriminator to filter ensemble model predictions based on confidence, submitting uncertain samples to human annotators for labeling. The newly labeled data is then used to retrain the model, enhancing its performance on the task.

2.3 Pseudo Labeling

The high cost of obtaining labeled data has also driven the development of semi-supervised learning, which combines a small amount of labeled data with a large amount of unlabeled data to train models. This approach improves the accuracy of learning tasks by leveraging

both labeled and unlabeled data [2].

Lee et al. introduced the pseudo-labeling method, where a model trained on labeled data generates pseudo labels for unlabeled data to be used in subsequent training [36]. Shi et al. proposed selecting training data based on the model's confidence in the unlabeled samples. Inspired by the concept of knowledge distillation, Xie et al. developed the noisy student model [66]. This method first uses a teacher model trained on labeled data to generate pseudo labels for unlabeled data. Then, a larger student model is trained on a combination of labeled and unlabeled data, with added noise to enhance robustness.

Rizve et al. identified that the poor performance of pseudo-labeling in semi-supervised learning is due to poorly calibrated models producing incorrect high-confidence predictions. They proposed an uncertainty-aware pseudo-label selection (UPS) framework, which improves pseudo-labeling accuracy by detecting and excluding noisy data from training [49].

To address the issue of model hallucinations caused by poorly calibrated pseudo labels, we propose a simple yet novel improvement. By evaluating the model's performance on a validation set, we ensure the model is well-calibrated and maintain a balanced ratio of unlabeled to labeled data over a certain number of epochs.

Chapter 3

Background

This chapter presents the background on Deep Learning-based Computer Vision Models, Optimization Problems, Mixed-Integer Linear Programming, Clustering, Active Learning, Human-in-the-Loop Annotation, Pseudo Labelling, Test Input Validity and Use.

3.1 Deep Learning-based Computer Vision Models

Computer vision, a branch of artificial intelligence, focuses on enabling machines to understand visual information from the world. The goal of computer vision is to automate tasks that the human visual system can perform, such as object recognition, facial recognition, and scene understanding. Traditional computer vision models rely on handcrafted features and algorithms like Scale-Invariant Feature Transform (SIFT) [39] and Speeded-Up Robust Features (SURF) [5].

Deep Learning (DL) introduced multi-layer neural networks to extract and learn image features. These networks consist of multiple layers of neurons, each layer transforming input data into increasingly abstract representations. The primary advantage of deep learning models is their ability to learn complex features and patterns directly from data, eliminating the need for manual feature engineering.

In this section, we overview three influential deep learning-based computer vision models: the Visual Geometry Group (VGG) network [56], the Residual Network (ResNet) [24], and the Vision Transformer (ViT) [16].

3.1.1 Visual Geometry Group (VGG) Network

The Visual Geometry Group (VGG) network, developed by Simonyan and Zisserman in 2014 [56], is a highly influential convolutional neural network (CNN) architecture. VGG is known for its simplicity and depth, characterized by the use of small receptive fields (3x3 convolutional filters) and a consistent architecture across different network depths.

The VGG architecture consists of a series of convolutional layers, followed by fully connected layers. The key architectural features include:

1. **Small Convolutional Filters:** All convolutional layers use 3x3 filters with a stride of 1 and padding to maintain spatial resolution. This design choice allows the network to capture fine-grained features and deep hierarchical representations.

2. **Max-Pooling Layers:** Max-pooling layers with a 2x2 window and a stride of 2 are used to progressively reduce the spatial dimensions while retaining important

features.

3. **Fully Connected Layers:** The final layers of the network are fully connected, typically followed by a softmax layer for classification.

The most popular variants of the VGG network are VGG-16 and VGG-19, which have 16 and 19 layers, respectively. Despite their relatively simple architecture, VGG networks have achieved excellent performance on benchmarks such as ImageNet. However, they require substantial computational resources and memory, which has spurred the development of more efficient architectures in subsequent years.

3.1.2 Residual Networks (ResNet)

Residual Networks, commonly known as ResNet, are a class of deep neural networks designed to address the vanishing gradient problem, which can hinder the training of very deep networks. Introduced by He et al. in 2015 [24], ResNet uses residual learning to facilitate the training of networks with hundreds or even thousands of layers.

The key innovation in ResNet is the introduction of shortcut connections, or skip connections, that bypass one or more layers. These connections allow the network to learn residual functions with reference to the layer inputs, rather than learning unreferenced functions. Mathematically, if $\mathcal{H}(x)$ represents the desired underlying mapping, the network instead learns the residual mapping $\mathcal{F}(x) = \mathcal{H}(x) - x$, which simplifies to $\mathcal{H}(x) = \mathcal{F}(x) + x$.

The architecture of a typical ResNet block is as follows:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{\mathbf{W}_i\}) + \mathbf{x}$$

where $\mathbf{x}$ is the input to the block, $\mathcal{F}$ is the residual function (e.g., a stack of convolutional layers), and $\mathbf{W}_i$ are the weights of the layers. The output $\mathbf{y}$ is obtained by element-wise addition of the input and the output of the residual function.

ResNet architectures, such as ResNet-50, ResNet-101, and ResNet-152, have achieved state-of-the-art performance on various benchmarks, including the ImageNet dataset. The success of ResNet has spurred further research into deeper and more complex network architectures.

3.1.3 Vision Transformers (ViT)

Vision Transformers (ViT) represent a paradigm shift in image recognition, leveraging the transformer architecture, originally developed for natural language processing (NLP). Introduced by Dosovitskiy et al. in 2020 [16], ViT treats images as sequences of fixed-size patches, analogous to words in a sentence, and processes these sequences using self-attention mechanisms.

In ViT, an image is first divided into a grid of non-overlapping patches, each of which is flattened and linearly embedded into a fixed-dimensional vector. These patch embeddings are then combined with positional embeddings to retain spatial information. The resulting sequence of embeddings is fed into a standard transformer encoder, which consists of

multiple layers of multi-head self-attention and feed-forward neural networks.

The architecture of a Vision Transformer can be summarized as follows:

1. **Patch Embedding:** Each $N \times N$ patch is flattened into a vector and linearly projected to a fixed dimension D.

2. **Positional Encoding:** Positional embeddings are added to the patch embeddings to incorporate spatial information.

3. **Transformer Encoder:** The sequence of patch embeddings, augmented with positional embeddings, is processed by a stack of transformer encoder layers, each comprising multi-head self-attention and feed-forward networks.

ViT has demonstrated competitive performance compared to traditional convolutional neural networks (CNNs) [38], achieving state-of-the-art results on image classification tasks with significantly fewer inductive biases. The success of ViT has led to further exploration of transformer-based architectures for various computer vision tasks, including object detection and segmentation.

3.1.4 Pretrained Models

Pretrained models [22] are deep learning models that have been previously trained on large datasets and can be used as a starting point for various tasks. Instead of training a model from scratch, which is computationally expensive and time-consuming, pretrained models leverage the knowledge already captured by these models, a process known as transfer

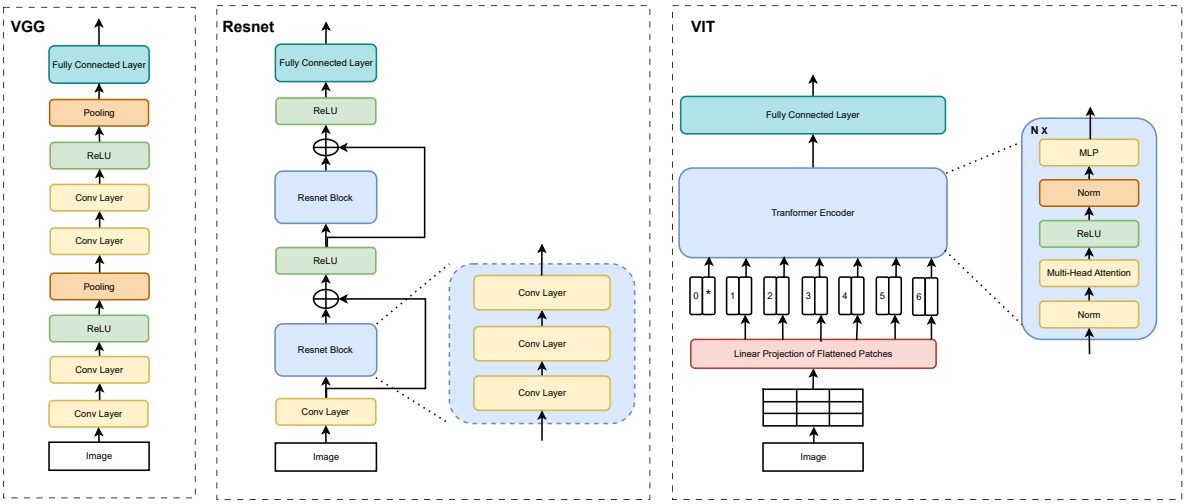

Figure 3.1: Structure of Computer Vision Models

learning. In transfer learning, the pretrained model's learned features are fine-tuned for specific tasks, reducing training time and improving performance, especially when labeled data is limited.

A pretrained model is essentially a neural network trained on a large benchmark dataset for a task similar to the one at hand. For instance, models pretrained on the ImageNet dataset have learned to recognize a wide variety of images, making them suitable for many computer vision tasks. These models contain weights and biases optimized during the initial training process, enabling them to capture general features from the data.

Pretraining involves training a deep neural network on a large dataset to learn general features. This training phase can take significant time and computational resources. During pretraining, the model learns to extract features through multiple layers of neurons, where initial layers capture low-level features like edges and textures, and deeper layers

capture high-level features such as object parts and entire objects.

Fine-tuning a pretrained model involves adapting it to a new, often smaller dataset specific to a particular task. This typically involves selecting a suitable pretrained model and using it as a feature extractor by freezing the pretrained layers and training only the newly added layers on the new data. Optionally, some pretrained layers can be unfrozen and retrained along with the new layers to better fit the new task.

The advantages of using pretrained models are significant. They reduce training time since the model has already learned useful features. They often achieve better performance, particularly with limited labeled data, and lower the resource requirements compared to training a model from scratch. Additionally, pretrained models provide a rich set of features that can be useful for various tasks.

Common pretrained models include VGG networks like VGG-16 and VGG-19, known for their simplicity and depth, ResNet (Residual Networks) models such as ResNet-50, ResNet-101, and ResNet-152, which use residual connections to train very deep networks, and Vision Transformers (ViT), such as ViT-13B, which use large transformer structure originally developed for natural language processing and treat images sequences of patches.

3.2 Optimization Problems

Optimization is a fundamental concept in theoretical and applied mathematics, playing a crucial role in various fields such as science, engineering, and economics. It involves the process of finding the best solution from all feasible solutions, aiming to maximize or

minimize a specific objective function. A general mathematical optimization problem can be written as follows:

$$\min \mathbf{c}^\top \mathbf{x}$$

$$\text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$$

where $\mathbf{x}$ is the vector of decision variables, $\mathbf{c}$ is the vector of coefficients for the objective function, $\mathbf{A}$ is the matrix of coefficients for the constraints, and $\mathbf{b}$ is the vector of constraint bounds. The notation $\mathbb{Z}^p \times \mathbb{R}^{n-p}$ indicates that the first p variables are integers and the remaining $n - p$ variables are continuous.

3.3 Mixed-Integer Linear Programming (MILP)

In the realm of optimization and mathematical modeling, Mixed-Integer Linear Programming (MILP) is a powerful and versatile tool. MILP is a subset of linear programming where some decision variables are restricted to integer values. This technique is essential when optimization problems include binary or integer variables [3].

MILP problems are defined by a linear objective function that needs to be maximized or minimized, subject to a set of linear equality and inequality constraints. The inclusion of integer variables differentiates MILP from standard linear programming. These integer variables can represent inherently discrete decisions, such as the number of units produced

or the routes chosen in a transportation network. In this paper, model confidences are treated as discrete integers ranging from 50 to 100.

MILP has a wide range of applications. In supply chain optimization, it helps determine the optimal production, inventory, and distribution plans to minimize costs while meeting demand. In scheduling and timetabling, MILP models can optimize the allocation of resources (such as personnel or machines) to tasks over time, maximizing efficiency and minimizing conflicts. This paper uses MILP to determine the conditions for filtering samples to be judged by humans, aiming to reduce human labeling effort while meeting desired accuracy levels.

The advantages of MILP include its flexibility in modeling various problems and its ability to provide optimal solutions given sufficient computational resources. However, MILP also faces challenges, particularly in computational complexity. The presence of integer variables significantly increases the problem's difficulty, making large-scale MILP problems computationally intensive. Additionally, the performance of MILP solvers can be highly sensitive to the problem's formulation, requiring careful construction to enhance efficiency.

3.4 Clustering

Clustering is an unsupervised machine learning technique used to group similar data points into clusters based on their inherent characteristics. Unlike classification, which requires labeled data, clustering aims to identify natural groupings within the data. This technique

is widely used in exploratory data analysis, pattern recognition, image segmentation, and information retrieval.

One of the most commonly used clustering algorithms is K-means clustering [9]. K-means aims to partition a dataset into k clusters, where each data point belongs to the cluster with the nearest mean, serving as a prototype of the cluster. The algorithm iteratively assigns data points to clusters and updates the cluster centroids until convergence, minimizing the within-cluster variance. The objective function for K-means can be expressed as:

$$\min \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

where C_i is the set of points in cluster i and μ_i is the mean of points in C_i.

Another popular clustering algorithm is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [32] that can identify clusters of arbitrary shape and handle noise in the data. DBSCAN groups together points that are closely packed and marks points that lie alone in low-density regions as outliers. The two key parameters of DBSCAN are ϵ (the maximum distance between two points to be considered neighbors) and minPts (the minimum number of points required to form a dense region).

Clustering techniques have several applications in deep learning and data labeling. For instance, clustering can be used to identify representative samples from a large dataset, reducing the amount of data that needs to be manually labeled. By selecting a diverse set of samples from different clusters, the labeling effort can be minimized while maintaining the representativeness of the dataset. This approach is particularly useful in active learning,

where the goal is to iteratively select the most informative samples for labeling to improve model performance with fewer labeled instances.

In the context of reducing the cost of test data labeling for deep-learning systems, clustering can play a significant role. By clustering the data and identifying the most representative samples from each cluster, we can prioritize the labeling of these samples, ensuring that the labeled data covers a broad spectrum of the input space. This strategy can significantly reduce the number of samples that need to be manually labeled while still achieving high model accuracy and robustness.

3.5 Active Learning

Active Learning (AL) [46, 55] is a specialized machine learning approach designed to enhance model performance by selectively choosing the most informative data points for labeling. This method is particularly advantageous in situations where labeled data is scarce or expensive to obtain, as it aims to maximize learning efficiency while minimizing the amount of labeled data required. By focusing on the most uncertain or informative samples, Active Learning can significantly boost model accuracy with fewer labeled instances, making it an essential technique in data-driven AI development.

To effectively implement Active Learning, various strategies can be employed to identify the most informative samples. These strategies include uncertainty sampling, Query by Committee (QBC), expected model change, diversity sampling, and adaptive sampling [55]. Each strategy has its strengths and is suitable for different application contexts.

Uncertainty sampling focuses on instances where the model's predictions are least confident, using techniques like entropy-based sampling, least confident sampling, and margin sampling. Query by Committee (QBC) involves multiple models predicting labels for the same data and selecting instances with the highest disagreement among them. Expected model change selects instances that would significantly alter the model's parameters if labeled, maximizing the learning impact of new data. Diversity sampling ensures a wide coverage of the input space by selecting samples that are dissimilar to those already chosen, using methods like clustering and density-weighted sampling. Adaptive sampling combines multiple criteria, such as uncertainty and diversity, to dynamically select the most informative instances based on the model's performance and the nature of the data.

As shown in Fig. 3.2, the workflow of Active Learning typically involves several key steps:

1. **Initial Training:** Begin by training a preliminary model using a small set of labeled data. This initial model serves as the foundation for further learning.

2. **Sample Selection:** Use the trained model to predict labels for a large pool of unlabeled data. Apply the chosen sample selection strategy to identify and select the most informative or uncertain samples.

3. **Human Annotation:** Present the selected samples to human annotators for labeling. The human-provided labels are crucial for ensuring high-quality data.

4. **Model Retraining:** Retrain the model using the newly labeled data combined with the original labeled dataset. This step helps the model improve its performance and accuracy.

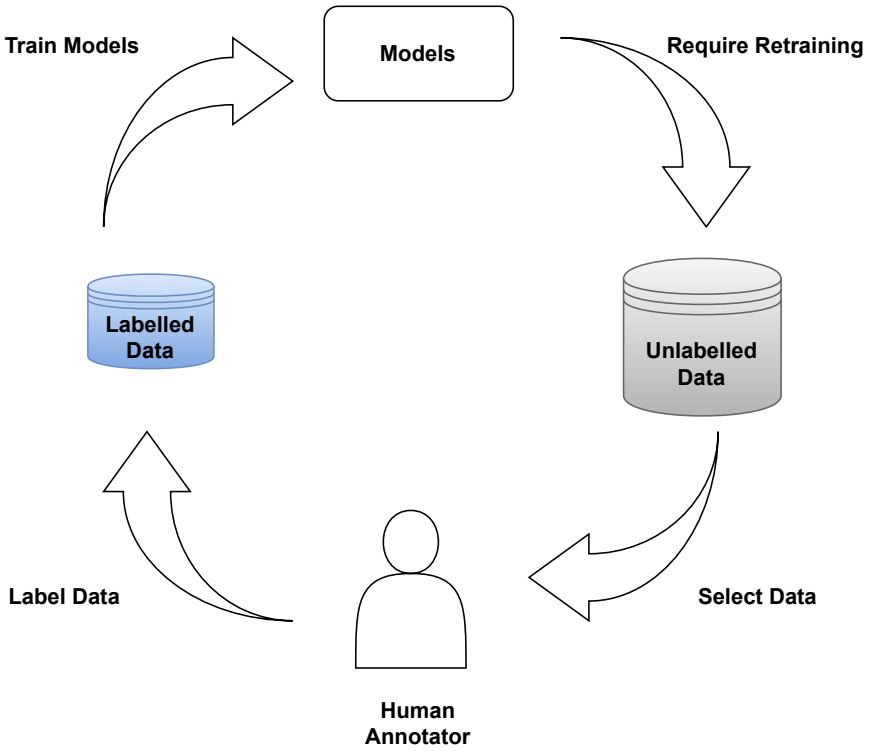

Figure 3.2: Workflow of Active Learning

5. **Iteration:** Repeat the process iteratively, with the model continually selecting new informative samples for human annotation and retraining.

Active Learning offers several key benefits, including efficiency, improved accuracy, and the integration of human expertise. By focusing on the most informative samples, Active Learning reduces the amount of labeled data required, saving time and resources. The iterative process of selecting uncertain samples and retraining the model helps in refining the model's decision boundaries, leading to better performance. Moreover, the involvement of human annotators ensures high-quality labeled data, which is crucial for the success of

the model.

Active Learning has proven effective across various domains, including natural language processing, computer vision, and healthcare. In computer vision, for instance, labeling images can be particularly labor-intensive. By using Active Learning, models can achieve high accuracy with significantly fewer labeled images, which is essential in applications like medical image analysis where expert annotations are costly and time-consuming.

3.6 Human-in-the-Loop Annotation

Human-in-the-loop (HIL) annotation [41] is an interactive data labeling method that combines human expertise with machine learning models. In this approach, humans handle complex samples to ensure high-quality labels, while the model processes simpler samples, thereby reducing human effort. Through active learning, the expertise of human annotators is continuously transferred to the model, further enhancing its performance. This technique leverages the strengths of both human intelligence and machine learning algorithms, improving label accuracy, especially in complex tasks where automated systems alone may fall short.

The HIL annotation process typically begins with training an initial machine learning model on a small set of labeled data. This preliminary model provides baseline predictions that can be reviewed and improved upon. Once the initial training is complete, the model predicts labels for a new set of unlabeled data. Human annotators then review these predictions, identifying and correcting any errors to ensure data accuracy and quality.

This human review adds a crucial layer of expertise and judgment that machines alone may lack.

The corrected labels are fed back into the model, which is then retrained on this enhanced dataset, incorporating both the original labeled data and the newly corrected data. This feedback loop forms a cycle of continuous improvement, with the model becoming increasingly accurate as it learns from human-provided corrections.

Applying active learning strategies to HIL can further enhance efficiency and performance. In active learning, the model identifies the most uncertain or informative samples from the pool of unlabeled data. These samples are prioritized for human annotation, ensuring that human effort is focused on the most challenging and valuable instances. This targeted approach maximizes the efficiency of the annotation process, reducing overall time and cost.

The benefits of HIL annotation are significant. It improves model accuracy by providing high-quality labels for difficult or ambiguous cases, helping the model learn from the most challenging examples and boosting overall performance. Additionally, by focusing human effort on the most informative samples, HIL efficiently balances the demand for high-quality labeled data with the time and cost involved in manual annotation.

Another key advantage of HIL is continuous learning. The iterative nature of the process ensures that the model continually adjusts and improves based on human feedback. This ongoing refinement is crucial for maintaining high performance as the model encounters new and diverse data.

Furthermore, HIL allows the integration of domain-specific expertise into the model.

Human annotators, often experts in their fields, provide valuable insights and corrections, enhancing the model's ability to handle specialized tasks and complex decision-making processes.

Despite its many advantages, HIL annotation also faces challenges. Scalability can be an issue, as the involvement of human annotators may become a bottleneck, particularly with large datasets. Managing human resources and ensuring efficient annotation workflows are vital for maintaining scalability. The quality of human annotations is another critical factor; achieving high-quality results depends on well-trained annotators and robust quality control mechanisms to ensure consistency and accuracy. Cost is also a consideration, as incorporating human annotation increases the expense of the process. Balancing the benefits of improved model accuracy with the associated costs is essential for practical deployment.

HIL annotation is widely applied across various fields, including medical imaging, natural language processing, and autonomous driving. In medical imaging, expert radiologists review and correct machine-generated labels, enhancing diagnostic accuracy. In natural language processing, linguists and language experts annotate and verify text data, improving language model performance. For autonomous vehicles, human reviewers validate and refine the system's predictions to ensure safety and reliability.

3.7 Pseudo Labeling

Pseudo labeling [36] is a semi-supervised learning technique that utilizes both labeled and unlabeled data to enhance the performance of machine learning models. The essence of pseudo labeling lies in using the model's own predictions on unlabeled data as if they were true labels, thereby augmenting the training dataset with additional pseudo-labeled examples.

The process begins with the initial training of a model on a small set of labeled data. Once trained, this model is employed to predict labels for the unlabeled data. The confidence of these predictions is crucial; typically, only the most confident predictions are selected as pseudo labels to minimize the risk of introducing noise into the training data. Confidence can be measured through various criteria, such as prediction probabilities, where a threshold is set to filter out low-confidence predictions.

After selecting the high-confidence pseudo labels, these are combined with the original labeled data to form an augmented training set. This expanded dataset now contains both the genuinely labeled examples and the pseudo-labeled examples, providing a richer and more diverse training resource for the model.

The model is then retrained on this augmented dataset. The retraining process allows the model to learn from the additional data points, which can lead to improved generalization and performance. This iterative process can be repeated multiple times, with the model continually generating new pseudo labels for the unlabeled data and refining its predictions with each iteration.

Pseudo labeling leverages the model's ability to generalize from labeled data to generate

pseudo labels for unlabeled data, thus effectively increasing the size of the labeled dataset without additional manual labeling efforts. This approach can significantly boost model performance, particularly when the initial labeled dataset is small.

However, there are challenges associated with pseudo labeling. One of the primary concerns is error propagation. If the model's predictions are incorrect, these errors can propagate into the pseudo-labeled data, potentially degrading the model's performance. To mitigate this, careful selection of high-confidence predictions is essential. Additionally, the initial biases of the model can be reinforced through pseudo labeling, as the model's predictions on the unlabeled data may reflect its existing biases.

Another critical aspect is setting the confidence threshold appropriately. A too low threshold may include many incorrect labels, while a too high threshold might exclude too many potentially useful pseudo labels. Striking the right balance is key to maximizing the benefits of pseudo labeling.

Despite these challenges, pseudo labeling has been successfully applied in various fields, such as image classification, natural language processing, and speech recognition. It enables models to take advantage of large amounts of unlabeled data, reducing the dependence on extensive labeled datasets and thus lowering the overall cost and effort involved in data labeling.

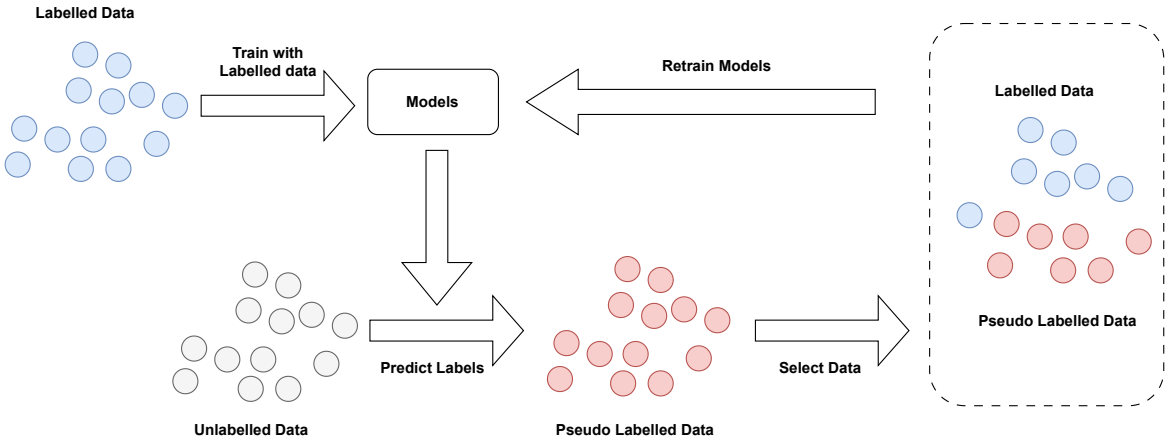

Figure 3.3: Workflow of Pseudo labelling

3.8 Test Input Validity and Use

Input validity refers to whether input data meets the expected requirements and constraints of a system, program, or function [48]. Ensuring input validity is crucial for the stability, security, and correctness of software systems. Traditional software systems typically impose specific preconditions on inputs, such as checking input ranges and data types, to ensure safety. However, for deep learning (DL) systems, the complexity of tasks and the opaque nature of neural networks make it challenging to verify input validity through explicit preconditions as done in traditional software. Therefore, defining input validity for DL systems is more complex.

Riccio et al. define input validity for DL systems as inputs that can be recognized by human domain experts within the input domain. This means that a human can confidently assign a label within the given input domain. For example, in a cat-dog classifier, a clear picture of a cat is a valid input because a human can recognize it and assign the correct

label, "cat." Conversely, a picture of an apple is not a valid input, as a human cannot label it as either "cat" or "dog." Similarly, if a picture of a cat is heavily distorted to the point where it is unrecognizable by a human expert, this input can also be considered invalid for the DL system since a human expert cannot assign a label to it.

Methods for checking test input validity primarily include human validation and automated validation. Human validation involves domain experts manually inspecting the generated test inputs to ensure they are semantically and task-relevant.

When test data is abundant, automated validators can alleviate the burden of manually checking input validity. These validators primarily use techniques such as Variational Autoencoders (VAEs) [33] to detect whether test inputs are out-of-distribution. A VAE is a generative model that reconstructs inputs and detects anomalies based on reconstruction error. A VAE consists of two neural networks: an encoder and a decoder. The encoder maps the original input to a latent space, represented as a multivariate normal probability distribution, while the decoder reconstructs the input from this latent representation. During training, the VAE minimizes the reconstruction error (the difference between the original and reconstructed inputs) and the Kullback-Leibler divergence between the learned latent distribution and the unit Gaussian distribution. When a trained VAE processes out-of-distribution inputs, it produces less accurate reconstructions compared to inputs similar to the training data. This property enables the VAE to act as an effective anomaly detector.

Distribution-Aware Input Validation (DAIV) [13] and SelfOracle [58] are two well-known automated validation techniques. DAIV uses a VAE trained on the same data as the DL model under test and requires an anomaly set for comparison, whereas Self-

Oracle does not require an anomaly set and uses reconstruction error to identify invalid inputs.

Despite the effectiveness of automated validators, they have limitations. Research by Riccio et al. indicates that while these validators can align well with human judgment, they may struggle with large, complex, and feature-rich datasets. In such cases, automated validators may misclassify valid inputs as invalid and vice versa, highlighting the need for continuous improvement and the integration of human validation during the testing process.

Chapter 4

Automatic and semi-automatic Image Labeling Approach

In this chapter, we present your two approaches for assisting humans in labelling images: *Human-in-the-loop* and *Pseudo labelling*. Figure 4.1 shows an overview of our framework for human-assisted automated labelling of images. The input to our framework is an unlabelled dataset D and human annotators and the output is a labelled dataset D_I and the accuracy of the automated labelling. Both *Human-in-the-loop* and *Pseudo labelling* approaches can be captured based on the framework in Figure 4.1. Our framework has three main phases: preprocessing, training and inference. Briefly, the preprocessing phase splits the dataset D into training, validation and inference datasets. Then we ask the human annotators to label the training and validation sets. Next, the training phase uses the labelled training set to train vision models. When the framework is instantiated for the Human-in-the-loop technique, it trains multiple vision models as well as a discriminator. A discriminator is

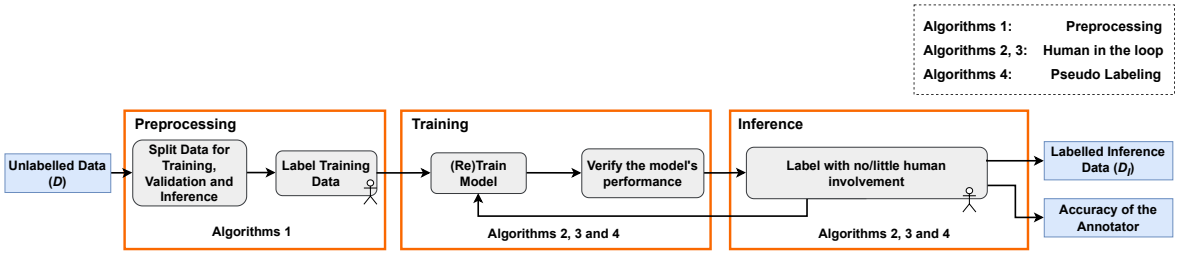

Figure 4.1: Overview of our approach for human-assisted automated labeling of images

a model that assigns a label to an image based on the consensus predictions of multiple vision models, or it refers the image to human annotators for further labeling when the vision models find it too challenging to classify. The vision models are then used in the inference phase to label the images in the inference dataset, and the discriminator is used to determine if the vision models are in agreement with a high-level confidence. When the framework is instantiated for the pseudo labelling technique, it trains one vision model in the training phase. Subsequently, the vision model is used in the inference phase to predict the labels of the images in the inference dataset and is iteratively re-trained on both the training and the inference sets to update the labels. The accuracy of each technique is then determined on the validation set.

It is worth noting that, since our methods learns from the human annotators' answers, it assumes that the answers provided by human annotators are entirely correct. In other words, the human annotators' responses serve as the ground truth for our framework. This assumption is also further discussed in Chapter 6, Section 5.2.1.

In the remainder of this section, we detail each step of the framework shown in Figure 4.1.

4.1 Preprocessing Phase

Algorithm 1 presents an overview of the preprocessing phase. This phase splits the dataset D into three sub-datasets: (1) the training dataset D_T, which is used to train each labeling technique; (2) the validation dataset D_V, which is used to evaluate each labeling technique; and (3) the inference dataset D_I, which is labeled by each labeling technique. To select diverse data for training, we utilize a feature extractor to extract the feature vectors of images (line 1 in Algorithm 1), and subsequently reduce the dimensions of the features. Specifically, we use Principal Component Analysis (PCA) [1] to reduce the dimensionality of the features (line 2 in Algorithm 1). PCA transforms the feature vectors into a lower-dimensional space while maximizing the diversity of the transformed features.

Subsequently, we apply Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to cluster the images based on their transformed feature vectors (line 3 in Algorithm 1). DBSCAN performance is influenced by two parameters: ϵ (the maximum distance for points to be considered neighbors) and the minimum number of points required to form a cluster. To determine optimal parameters, we perform grid search over commonly used parameter ranges and select the combination that maximizes the number of clusters. This approach aims to ensure that the preprocessed training data exhibits the richest possible feature diversity, thereby enhancing the training set's representativeness.

Once the clusters are computed, we randomly select images from each cluster to form the training (D_T) and validation sets (D_V). This way, D_T has a balanced representation of various features, improving the vision models' ability to learn and generalize across different features during the training phase. Meanwhile, D_V serves as a reliable set for

Algorithm 1 Preprocessing Phase

Input D: An unlabelled dataset
Input H: Human Annotator
Input T_N: The size of training dataset
Param N_p: Number of principle components
Output D_T: Training dataset
Output D_V: Validation dataset
Output D_I: Inference dataset

1: $Features \leftarrow$ ExtractFeatures(D); //Use a feature extractor to extract features from D
2: $Features \leftarrow$ PCA($Features, N_p$) //Use PCA to reduce the dimension of the features to N_p
3: $Clusters \leftarrow$ DBSCAN($Features$); //Apply DBSCAN to cluster images based on the extracted features
4: $D_T, D_V \leftarrow$ SplitDataset($D, Clusters, T_N$); //Split the dataset into the training dataset D_T and validation dataset D_V with size T_N from clusters
5: $D_T \leftarrow$ AskHumanToLabel(H, D_T);
6: $D_V \leftarrow$ AskHumanToLabel(H, D_V);
7: $D_I \leftarrow D \setminus (D_T \cup D_V)$
8: **return** D_T, D_V, D_I

evaluating the vision models' ability to handle diverse features effectively.

Furthermore, since D_T and D_V will be used to train and evaluate vision models, respectively, we ask the human annotator to provide labels for the images in D_T and D_V (lines 5–6 in Algorithm 1). The remaining data will be used as the inference set (D_I) in the Inference phase of Figure 4.1 (line 7 in Algorithm 1). The algorithm returns the training (D_T), validation (D_V) and inference (D_I) datasets (line 8 in Algorithm 1).

4.2 Human-in-the-Loop

In this section, we present our Human-in-the-Loop approach, which uses multiple vision models to automate data labeling with human assistance. An overview of our Human-in-the-Loop approach is shown in Figure 4.2. As depicted in the figure, the core steps of our Human-in-the-Loop approach are the training and inference phases. The inputs to this approach include the training dataset D_T, the validation dataset D_V, the inference dataset D_I, human annotators H, and an accepted accuracy level α. The output from this approach is a labeled inference dataset, with the approach's accuracy evaluated against the validation dataset D_V. Further details on the procedure are presented in the pseudo-code in Algorithm 2, which aligns with the process flow shown in Figure 4.2.

Notably, the Human-in-the-Loop approach involves human annotators during both the training and inference phases. These annotators do not need to be the same individuals. In cases where the annotators differ, if there are discrepancies between their labeling styles, the model will gradually adapt during the inference phase to align more closely with the annotator encountered in that phase. However, since this paper assumes that human annotators do not make errors, the two annotators are considered equivalent within the scope of our methodology.

4.2.1 Training Phase.

This phase focuses on training vision models. On line 1 in Algorithm 2, we use half of D_T to train a set of vision models. Then, on line 2 in Algorithm 2, we use the trained

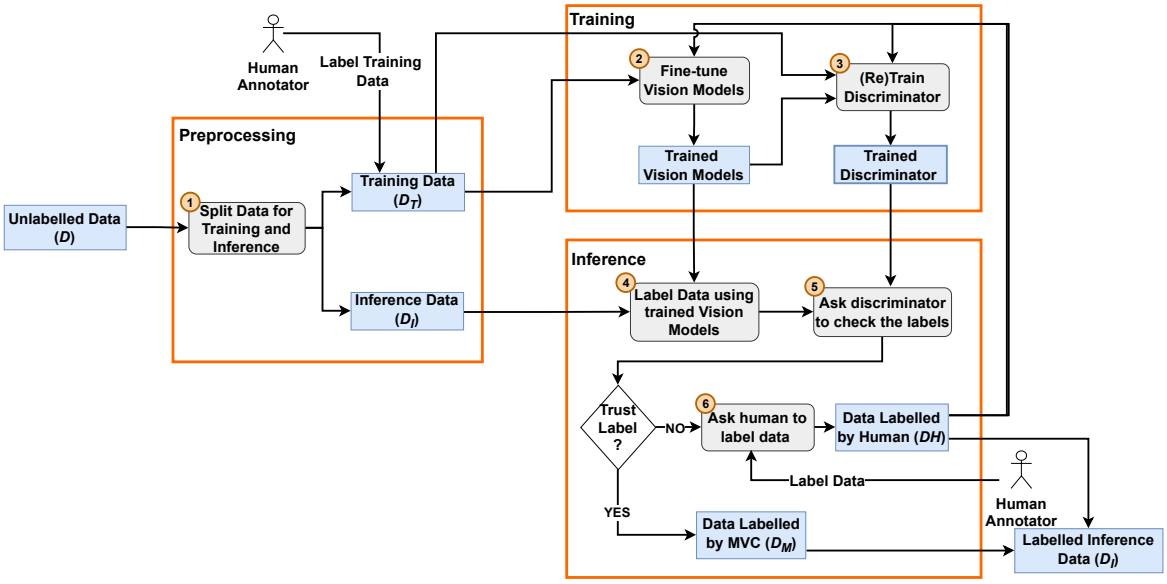

Figure 4.2: Overview of Human-in-the-loop approach

vision models and the remaining half of D_T to tune a discriminator that will be used to assess the quality of the vision models' predictions in inference phase. In particular, our discriminator employs Mixed Integer Linear Programming (MILP) to learn the optimal confidence thresholds for vision models. Specifically, MILP utilizes the predictions and confidences of the vision models on D_T to formulate a linear program where the goal is to attain an accuracy score of at least α while minimizing human effort.

The MILP formulation for our discriminator model is detailed as follows: Suppose the size of D_T is equal to N. Let $x_1, \ldots, x_N$ be binary variables indicating whether, or not, the i^{th} sample in D_T requires manual labeling. In particular, $x_i = 1$ means that the sample i is labelled by human. Let acc_i such that $1 \leq i \leq N$ be a binary vairable indicating whether, or not, the i^{th} sample in D_T is correct. In particular, $acc_i = 1$ means that the sample i

is labelled correctly. Suppose our discriminator uses three vision models. We denote by l_{i1}, l_{i2} and l_{i3} the labels predicted by our three vision models for the i^{th} sample in D_T, and further, we denote by c_{i1}, c_{i2} and c_{i3} the confidence of our three vision models for predicting the label of the i^{th} sample in D_T.

Our discriminator requires three thresholds, th_1, th_2, and th_3 corresponding to its underlying vision models, to determine whether it accepts the labels provided by these models for a given sample or whether it requires a human to label that sample. Specifically, for a given sample i in D_T, the discriminator accepts the label of this sample if $l_{i1} = l_{i2} = l_{i3}$ and $c_{i1} > th_1$, $c_{i2} > th_2$ and $c_{i3} > th_3$. In other words, the discriminator accepts the labels provided by the three vision models for a sample i, if these models produce the same label with a confidence higher than their corresponding threshold. We use the following MILP formulation to compute optimal thresholds th_1, th_2, and th_3 for the discriminator model that we use in our human-in-the-loop approach:

$$\text{minimize} \quad \sum_{i=1}^{N} x_i$$

$$\text{subject to}$$

$$\frac{\sum_{i=1}^{N} ((1 - x_i) \cdot acc_i + x_i)}{N} \geq \alpha,$$

$$x_i = 1 - \min(y_{i1}, y_{i2}, y_{i3}), \quad \text{for } i = 1, \ldots, N,$$

$$y_{ij} = 1 \quad \text{if } c_{ij} \geq th_j \text{ and } l_{i1} = l_{i2} = l_{i3}, \quad \text{otherwise } y_{ij} = 0,$$

$$0.5 \leq th_j \leq 1.0, \quad \text{for } j = 1, 2, 3.$$

Specifically, the MILP formulation above includes one optimization objective and three constraint conditions:

- *Objective Function*: Our objective function $\min \sum_{i=1}^{N} x_i$ aims to minimize the number of samples requiring manual annotation. This optimization goal helps reduce labor costs and increase processing speed, allowing the system to maintain efficient operation with minimal human intervention.

- *Accuracy Constraint*: To ensure the overall accuracy of the system, we set an accuracy constraint $\frac{\sum_{i=1}^{N}((1-x_i)\cdot acc_i+x_i)}{N} \geq \alpha$ where α is the minimum acceptable accuracy which is an input of the Human-in-the-Loop approach as shown in Algorithm 2. This constraint guarantees that the system's prediction accuracy remains above α in automated mode, thus ensuring the quality of predictions without human intervention.

- *Annotation Decision Constraint*: The annotation decision constraint $x_i = 1-\min(y_{i1}, y_{i2}, y_{i3})$ indicates when manual review is necessary. Specifically, if all y_{i1}, y_{i2}, and y_{i3} are one, then x_i can be labelled automatically using the vision models. Otherwise, if at least one of these variables is zero, it means that the confidence of some model falls below its respective threshold th_j, or there is disagreement among the models' predictions, x_i is set to 1. Hence, there is a need for human labelling. This ensures that in cases of critical uncertainty, human expertise can be leveraged to improve the reliability of the predictions.

- *Confidence Threshold Constraint*: We also set the range of confidence thresholds $0.5 \leq th_j \leq 1.0$ to ensure that each model's confidence in its predictions falls within a

reasonable range. This range allows the model to make decisions automatically when confidence is sufficiently high, while introducing human review when confidence is low, thus balancing automation with accuracy.

It is crucial to clarify that we applied the Big-M method to linearize the conditional logic constraint $c_{ij} \geq th_j$. Let M represent a sufficiently large number. The following linear constraints were used:

$$c_{ij} - th_j + M \times (1 - y_{ij}) \geq 0$$

$$c_{ij} - th_j - M \times y_{ij} \leq 0$$

These constraints ensure that when c_{ij} exceeds th_j, y_{ij} must be 1 to satisfy both conditions. Conversely, if c_{ij} is less than th_j, y_{ij} must be 0.

Through solving this linear program, the discriminator will find the optimal thresholds for vision models' confidence. The confidence thresholds will be used in the inference phase to determine whether the labels predicted by the vision models are accurate enough or not.

It is important to note that using confidence to filter images introduces a potential issue: confidence is also the model's estimation of the label. When the model performs poorly on a task, it may assign high confidence to incorrect predictions, a phenomenon known as the "model hallucination problem." However, our approach can mitigate this issue to some extent. By setting an expected accuracy α as a constraint in the MILP, the confidence thresholds are adjusted to a sufficiently high level to ensure accuracy, even for high-confidence but incorrect predictions.

Our experimental results, presented in Chapter 6 Section 6.1, validate the effectiveness of this mechanism. Additionally, if the thresholds are set excessively high, it can serve as an indicator of poor model performance, suggesting the need to adopt alternative methods or further tune the model.

4.2.2 Inference Phase.

Recall that the unlabelled dataset D is split into three sub datasets (i.e., training, validation, and inference) in the Preprocessing phase (Section 4.1). The goal of this phase is to assist the human annotators in labelling the images in the inference dataset D_I. We start by labelling the images in D_I using the trained vision models in the training phase (line 4 in Algorithm 2). Specifically, we pass the images to each $model_i$ and get the confidence c_i and the predicted label l_i of each image from $model_i$, where $1 \leq i \leq n$, and n is the number of vision models. Therefore, for each image, we have n tuples of confidences and labels $(c_1, l_1), \ldots, (c_n, l_n)$ corresponding to vision models. On lines 5–12 of Algorithm 2, we utilize the trained discriminator from the training phase to assess the confidence of the predicted labels for each image. Specifically, if the predicted labels of the vision models are identical (i.e., $l_1 = \ldots = l_n$) and the confidences of all vision models surpass their respective thresholds calculated by the discriminator (i.e., $c_1 \geq th_1 \wedge \ldots \wedge c_n \geq th_n$), then the predictions of the vision models are deemed accurate enough to label the image (line 6 of Algorithm 2). Otherwise, we ask the human to provide the label of the image (lines 8–9 of Algorithm 2). In our approach, each time that a human provides a label of an image (i.e., the predictions of vision models are not accurate), we re-train the vision models and

tune the discriminator using D_T and the newly-labelled data by humans in D_I (lines 10–12 of Algorithm 2).

We will later show in our evaluation (Chapter 6) that through iterative training of the vision models and the discriminator, their accuracies increases, leading to a lower number of inaccurately labelled images. The process continues until all the images in D_I are labelled. Then, we use D_V to evaluate the accuracy of the human-assisted annotator (line 15 of Algorithm 2). Specifically, we use the vision models to predict the labels of the images in D_V. We then calculate how many images in D_V have correct labels. Finally, the accuracy and D_I are returned as the output (line 16 of Algorithm 2).

4.2.3 Inference Phase with Limited Human Effort.

To control human effort in inference phase, we present a modified version of our original Human-in-the-loop presented in the previous section. This version, which is outlined in Algorithm 3, introduces a threshold for human effort during the inference phase and makes some adjustments to the workflow. Specifically, before passing an image to each vision model, we add a check for the current human effort consumption (see line 5 of Algorithm 3). If the human effort consumption is below the threshold, the process continues as described in subsection 4.2.2 (see lines 6-14 of Algorithm 3). However, when human effort consumption exceeds the threshold, we switch the active learning architecture to an ensemble model and use a majority voting approach to determine the final prediction (see lines 16, 17 Algorithm 3).

4.3 Pseudo Labelling

In this section, we describe the process of employing Pseudo Labeling for the automatic annotation of unlabeled data. An overview of our Pseudo Labeling approach is presented in Figure 4.3, which visualizes the main components of the method including the iterative process of model updates and data utilization.

As shown in Figure 4.3, the inputs to our approach are the training dataset D_T, the validation dataset D_V, the inference dataset D_I, the number of vision model updates in each iteration m, and an accepted accuracy α. The output is a labeled inference dataset, with the approach's accuracy evaluated in a similar manner to Section 4.2. The Pseudo Labeling method iteratively uses both the annotated training data D_T and the unlabeled inference data D_I to update the vision models and enhance their predictive accuracy. It is important to note that D_T contains human annotations, whereas D_I remains unlabeled.

The iterations continue until the vision model's accuracy meets the predefined threshold α or the allotted time budget is exhausted. In each iteration, the vision model checkpoint that demonstrates the highest accuracy is selected to generate labels for the dataset D_I, as depicted in Figure 4.3.

Our Pseudo Labeling approach, which leverages both labeled and unlabeled data in a semi-supervised fashion, is encapsulated in Algorithm 4. The method is structured into three primary steps, each integral to the iterative process illustrated in the figure.

4.3.1 Generating Pseudo Labels

In this step, we compute the pseudo labels. A pseudo label is the vision model's prediction on an unlabeled sample of the dataset. (line 4 of Algorithm 4) A pseudo label might not be correct, but given a well-trained vision model, it has a high probability of being correct. We use the trained vision model to generate pseudo labels for the inference data, D_I, at each iteration.

4.3.2 Forming a Batch

In this step, we form a batch consisting of two parts (lines 6–10 of Algorithm 4). The first part is randomly sampled from the training data (D_T), and the second part is randomly sampled from the inference data with pseudo labels (D_I). A critical aspect of this process is determining the ratio of data from each source. We denote the proportion of D_I included in a batch by the hyperparameter $p \in [0, 1]$ (lines 6–7 of Algorithm 4).

For example, setting $p = 0.1$ means that 10% of a batch is sampled from D_I, while the remaining 90% comes from D_T. By controlling p, we can ensure that the majority of labels in the batch are accurate, which aids the vision model in learning meaningful patterns from the data. Specifically, with $p = 0.1$, more than 90% of the batch labels are guaranteed to be correct, providing the model with a reliable foundation for learning.

The need to carefully control p arises from the cold start problem inherent in pseudo-labeling. In the early stages of training, the vision model is not well-calibrated, leading to a significant number of incorrect pseudo-labels in D_I. To mitigate this, we reduce p during

these early phases, thereby increasing the proportion of accurate training data D_T in each batch. This adjustment accelerates model calibration and reduces noise in the training process.

As training progresses and the vision model improves, it begins to produce more accurate predictions for the pseudo labels in D_I. At this stage, we increase p, incorporating more data from D_I to enhance the model's generalization ability and minimize the risk of overfitting to D_T.

To dynamically adjust p throughout training, we use the model's accuracy on the validation dataset D_V as a guide. Specifically, we set p equal to the model's accuracy on D_V, such that $p =$ accuracy on D_V. This strategy leverages the validation accuracy as an indicator of the model's performance on unlabeled data. When the model's accuracy is low, we use a smaller p, favoring more labeled data from D_T to stabilize training. Conversely, as the model's accuracy improves, we increase p, incorporating more data from D_I to enhance diversity and reduce overfitting.

This dynamic adjustment ensures that the batch composition evolves in tandem with the model's performance. This dynamic adjustment ensures that the batch composition evolves in tandem with the model's performance. When the model performs well, this approach allows the model to access a greater variety of unique data during training, further mitigating the risk of overfitting and improving the overall robustness of the learning process.

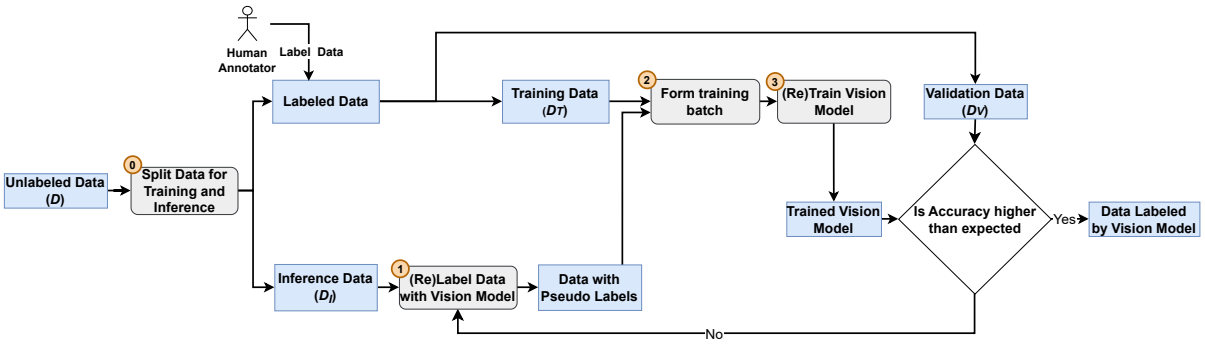

Figure 4.3: Overview of Pseudo Labelling approach

4.3.3 Updating the vision model

In this step, the vision model is updated using the batch formed in previous step (lines 12–19 of Algorithm 4). We introduce a new hyperparameter, n, which specifies the number of vision model updates used with the batch from the previous step. We keep the best vision model checkpoint, based on the validation accuracy, among the n updates of the vision model in this step. For example, by $n = 5$, we update the vision model using the batch from the previous step for 5 times. We then keep the best checkpoint, i.e. the one with the highest validation accuracy, of the vision model during these 5 updates.

After each iteration, if the validation accuracy is more than or equal to the expected value, or the processing budget is finished, we terminate the process. At the end, the best accuracy on the validation dataset $best\_acc$ and the annotated inference dataset D_I are returned as the output. (line 16 of Algorithm 4)

Algorithm 2 Human-in-the-loop

Input H: Human Annotator
Input D_T: The training dataset
Input D_V: The validation dataset
Input D_I: The inference dataset
Input α: The accepted accuracy
Output D_I: The labelled inference dataset
Output acc: The accuracy of the annotator on the validation dataset

Training Phase

$\{model_1, \ldots, model_n\} \leftarrow$ TrainVisionModels(D_T);
$(th_1, \ldots, th_n) \leftarrow$ TrainDiscriminator($D_V, \alpha, \{model_1, \ldots, model_n\}$); //Train the discriminator to achieve accuracy α

Inference Phase

while $(\exists \langle image, \texttt{NULL} \rangle \in D_I)$ **do**
    $(c_1, l_1), \ldots, (c_n, l_n) \leftarrow$ PredictLabel(image, $\{model_1, \ldots, model_n\}$); //Predict for the unlabelled image the label l_i and confidence c_i using each $model_i$ such that $1 \leq i \leq n$
    if $(c_1 \geq th_1 \wedge \ldots \wedge c_n \geq th_n) \wedge (l_1 = \ldots = l_n)$ **then**
        $D_I \leftarrow$ UpdateDataset($\langle image, l_1 \rangle$); //Update the label of image in D_I
    else
        $\langle image, l \rangle \leftarrow$ AskHumanToLabel(H, image); //Data is labelled by human
        $D_I \leftarrow$ UpdateDataset($\langle image, l \rangle$);
        $D_T \leftarrow D_T \cup \{\langle image, l \rangle\}$; //Store in the training dataset
        $\{model_1, \ldots, model_n\} \leftarrow$ TrainVisionModels(D_T);
        $(th_1, \ldots, th_n) \leftarrow$ TrainDiscr($D_V, \alpha, \{model_1, \ldots, model_n\}$); //Re-train the discriminator to achieve accuracy α using new training dataset
    end if
end while
$acc \leftarrow$ Evaluate(D_V);
return D_I, acc

Algorithm 3 Human-in-the-loop within Limited Human Effort

Input H: Human Annotator
Input D_T: The training dataset
Input D_V: The validation dataset
Input D_I: The inference dataset
Input α: The accepted accuracy
Input ϵ: The accepted human effort
Output D_I: The labelled inference dataset
Output acc: The accuracy of the annotator on the validation dataset

Training Phase

$\{model_1, \ldots, model_n\} \leftarrow$ TrainVisionModels(D_T);
$(th_1, \ldots, th_n) \leftarrow$ TrainDiscriminator($D_V, \alpha, \{visionmodel_1, \ldots, visionmodel_n\}$); //Train the discriminator to achieve accuracy α

Inference Phase

$HE \leftarrow 0$ //Initialize HE, representing the Human Effort already expended.
while $(\exists \langle image, \texttt{NULL} \rangle \in D_I)$ **do**
    $(c_1, l_1), \ldots, (c_n, l_n) \leftarrow$ PredictLabel(image, $\{model_1, \ldots, model_n\}$); //Predict for the unlabelled image the label l_i and confidence c_i using each $model_i$ such that $1 \leq i \leq n$
    if $HE < \epsilon$ **then**
        if $(c_1 \geq th_1 \wedge \ldots \wedge c_n \geq th_n) \wedge (l_1 = \ldots = l_n)$ **then**
            $D_I \leftarrow$ UpdateDataset($\langle image, l_1 \rangle$); //Update the label of image in D_I
        else
            $\langle image, l \rangle, HE \leftarrow$ AskHumanToLabel(H, image); //Data is labelled by human and the cumulative human effort (HE) is calculated
            $D_I \leftarrow$ UpdateDataset($\langle image, l \rangle$);
            $D_T \leftarrow D_T \cup \{\langle image, l \rangle\}$; //Store in the training dataset
            $\{model_1, \ldots, model_n\} \leftarrow$ TrainVisionModels(D_T);
            $(th_1, \ldots, th_n) \leftarrow$ TrainDiscr($D_V, \alpha, \{model_1, \ldots, model_n\}$); //Re-train the discriminator to achieve accuracy α using new training dataset
        end if
    else
        $l \leftarrow$ SelectMajorityVoting($(l_1 = \ldots = l_n)$); //Take the majority vote
        $D_I \leftarrow$ UpdateDataset($\langle image, l \rangle$);
    end if
end while
$acc \leftarrow$ Evaluate(D_V);
return D_I, acc

Algorithm 4 Pseudo Labeling for Semi-Supervised Learning

Input H: Human Annotator
Input D_T: The training dataset
Input D_V: The validation dataset
Input D_I: The inference dataset
Input α: The accepted accuracy
Param m: The number of vision model's updates used with the same batch
Output D_I: The labelled inference dataset
Output $best\_acc$: The accuracy of the annotator on the validation dataset

$best\_acc \leftarrow 0$
while $(best\_acc < \alpha) \wedge$ Budget Available **do**
    // **Step 1: Generating Pseudo Labels**
    $D_I \leftarrow \textsc{PredictPseudoLabel}(D_I, model)$;
    // **Step 2: Forming a Batch**
    $acc \leftarrow \textsc{Evaluate}(D_V)$;
    $p \leftarrow acc$
    $batch_T \leftarrow \textsc{Sample}(D_T, 1-p)$; //Sample a portion from D_T
    $batch_I \leftarrow \textsc{SampleWithPseudoLabels}(D_I, p)$; //Sample a portion from D_I
    $batch \leftarrow \textsc{Concatenate}(batch_T, batch_I)$;
    // **Step 3: Updating the vision model**
    for $j = 1$ to m **do**
        $model \leftarrow \textsc{UpdateVisionModel}(batch)$;
        $acc \leftarrow \textsc{Evaluate}(D_V)$;
        if $acc > best\_acc$ **then**
            $best\_acc \leftarrow acc$;
            $best\_model \leftarrow model$; //Save the checkpoint of the vision model.
        end if
    end for
    $model \leftarrow best\_model$;
end while
return $D_I, best\_acc$

Chapter 5

Evaluation Strategy

In this chapter, we present a comprehensive evaluation of our proposed approaches. In section 5.1, we present the key research questions addressed by our evaluation. Our aim is to provide a clear understanding of how our techniques measure up in terms of human effort and accuracy, and whether they can achieve near-perfect accuracy for test sets used for testing vision models. Then in section 5.2, we introduce the experiment setup, including the baselines, evaluation metrics, datasets, parameters and training settings.

5.1 Research Questions

Our experiments aim to address the following research questions:

RQ1. How do our approaches compare with respect to human effort and accuracy?

To answer this question, we compare the performance of our two proposed approaches against three baseline models, VGG16, ResNet50, and ViT_13B, under conditions where human effort is applied to less than half of the test sets. As described in Chapter 5 Section 5.2, we conducted comprehensive tests on seven datasets, both public and proprietary. We employed two key methods for this evaluation:

First, since the initialization of a model significantly influences its subsequent performance, we explored the effects of varying the amount of training data used for initialization. Poor initialization not only increases the likelihood of the model producing incorrect but highly confident predictions, but also impairs the ability of the discriminator to select representative data. This affects the determination of confidence thresholds, potentially leading to overly optimistic outcomes and incorrect labeling due to misplaced trust in the model's predictions. Therefore, we examined model performance when initialized with varying proportions of the dataset: 5%, 15%, 25%, 35%, and 45%.

Second, we evaluated the performance of different approaches under constrained human effort. Specifically, we examined how models performed when human effort was limited to 5%, 15%, 25%, 35%, and 45% of the dataset. Given that Human-in-the-loop methods emphasize collaborative data labeling between humans and machines, it is typically difficult to control the extent of human involvement. To assess Human-in-the-loop's performance under limited human effort, we imposed a maximum cap on human effort. Once this limit was reached, human involvement ceased, and the Human-in-the-loop method was converted into an ensemble model. In this phase, the three computer vision models within Human-in-the-loop predicted labels independently, with the majority vote determining the final label. As human intervention was no longer possible, the discriminator was not used

to filter model confidences.

RQ2. Can our techniques produce test sets with very high accuracy, close to 99%?

To address this question, we explored the performance of the baseline models and Human-in-the-loop under conditions of high human effort. For the baseline models, we set human effort at 55%, 65%, 75%, 85%, and 95%. In contrast, for the Human-in-the-loop approach, we adjusted the initial amount of training data, as Human-in-the-loop can automatically assess task difficulty through its discriminator and balance accuracy with human effort.

5.2 Experimental Setup

5.2.1 Baselines

We compare our approaches with two naive baselines: (1) human labeling all the images, and (2) individual computer vision models used in a supervised learning mode. Below we discuss each baseline in detail.

Human Labeling. For the first baseline, we ask the human annotators to label all the images in the dataset D. Given that the purpose of our method is to learn from human labeling patterns, it assumes that human annotations are entirely correct. As a result, the performance of human annotators is always considered to be 100% in terms of accuracy and requires 100% Human Effort. Therefore, we do not explicitly present

the results for this baseline in the experimental results. In real-world scenarios, human annotators may occasionally make mistakes, which would introduce more noise. In such cases, the expected accuracy of our method should theoretically be the product of Human Accuracy and Model Accuracy. However, for simplicity and consistency in presenting our results, we assume Human Accuracy = 100% in all experiments. Under this assumption, the accuracy reported in the results can be interpreted as the degree to which the model approaches human-level annotations rather than as an absolute measure of correctness.

Individual computer vision models. In the second baseline, similar to the pre-processing phase in Section 4.1, we split dataset D into two sub-datasets (i.e., training and inference datasets) and ask human annotators to label the training dataset. For this baseline, unlike our two approaches discussed in chapter 4 where we split the dataset D using DBSCAN, we opt for random splitting of the dataset D in the preprocessing phase. We then train each model on the training dataset and use each model to predict the label of the remaining unlabelled images. We ensure that the size of the training datasets used for training models remains consistent across both our approaches and baselines.

5.2.2 Metrics

We use two metrics, *accuracy* and *human effort*, to evaluate the effectiveness and efficiency of different labelling approaches. Accuracy (Acc) is the number of correctly labelled images in the inference dataset over the inference dataset size ($|D_I|$).

Human effort (HE) is calculated as the number of images labelled by humans divided by the total size of the dataset ($|D|$). An approach with a lower level of human effort

is deemed more efficient than one with a higher level of human effort. This is because labelling images by humans is significantly more costly than utilizing an computer vision model for labelling.

5.2.3 Datasets

We rigorously validate the effectiveness and efficiency of our approaches using a diverse set of datasets sourced from both public and private repositories. From public repositories, we select widely-used datasets from the literature, including CIFAR-10, Imagenet, MNIST, Fashion MNIST, and SVHN. Additionally, we use a dataset from Smart Inside, a private repository provided by our industry partner. Table 5.1 shows the datasets that we have used in our evaluation. Below, we introduce each dataset.

CIFAR-10. The CIFAR-10 dataset contains 32×32 coloured images across 10 categories, with each category comprising 6,000 samples, totaling 60,000 images [35]. The categories include automobile, airplane, bird, etc.

Imagenet-c and CIFAR-10-c. Hu et al. [28] applied transformations to images from the Imagenet dataset [14] based on human perceptual scales to evaluate model's performance under various environmental conditions like frost, and sharp brightness. The average dimensions of these images is 469×387. They considered only two classes in their evaluation: *car* and *not car*. Therefore, this dataset comprises 4812 original images (i.e., not transformed) and 5265 transformed images that are either labelled *car* or *not car*. In our study, we exclusively use transformed images because the models we employ are pretrained on the Imagenet dataset which has overlap with the untransformed portion of this dataset. Addi-

tionally, They also transformed the car and not car images from CIFAR-10. We incorporate both the original and transformed images from this dataset, denoting it as CIFAR-10-c. This dataset comprises 2280 original and 2349 transformed images.

MNIST. The MNIST database [15] comprises 70,000 grayscale images of handwritten digits, including a training set of 60,000 examples and a test set of 10,000 examples. These images are labelled as digital number from 0 to 9. The digits are normalized and centred in a 28×28 image.

Fashion MNIST. Fashion-MNIST [65], featuring 70,000 images of Zalando's fashion articles across 10 categories such as T-shirt, Trouser, Pullover, acts as a benchmark for image classification. Fashion-MNIST maintains the same structure of training and testing splits as in MNIST, yet significantly elevates the difficulty level.

Smart Inside Dataset. Developed in collaboration with an industrial partner, our dataset features images designed to identify faults in power poles and power grid infrastructure. The Smart Inside collection leverages generative AI to simulate adverse weather conditions, such as fog, on high-voltage power poles. This enriched dataset comprises 2538 transformed images, each labeled by two annotators who evaluate the realism of the weather conditions applied. Specifically, labels are designated as *OK* if the power poles remain recognizable despite the adverse conditions. Conversely, if the transformation is deemed unrealistic, the label is marked as *Lost*. We utilize both the original and transformed images for our analysis.

Table 5.1: Datasets used in our study as well as their sizes, dimensions, and categories

Dataset	Size of Dataset	Image Dimensions	Categories
CIFAR-10	70,000	32×32	10 categories: Bird, Cat, Deer, Dog, Frog, etc.
Imagenet-c	10,421	469×387	Car, Not Car
CIFAR-10-c	4,657	32×32	Car, Not Car
MNIST	70,000	28×28	10 categories: zero, one, ..., nine
Fashion MNIST	70,000	28×28	10 categories: T-shirt, Trouser, Pullover, etc.
SVHN	600,000	32×32	10 categories: zero, one, ..., nine
SmartInside	2,538	512×320	OK, Lost

5.2.4 Models Setup

We utilize three state-of-the-art computer vision models: ResNet50, VGG16 and ViT. These models are chosen based on their high performance in image classification tasks, their popularity, and their widespread adoption in deep learning research. These models are pre-trained on the Imagenet-1k dataset [52] and provided by the Pytorch vision library. Below, we provide details on each model.

ResNet50 is a deep convolutional neural network, consisting of 48 convolutional layers, one Max Pooling layer, and one Average Pooling layer [24].

ViT is the short for Vision Transformer, [16]. ViT applies a transformer model which is primarily used for machine translation in natural language processing, to image classification tasks. We use ViT-B-16 which divides images into a sequence of 16×16 patches, to transfer each image to a sequence vector. ViT then utilizes a transformer model to predict

Table 5.2: Annotation and description of the parameters.

Parameter	Description	Value	Used in
T_N	The training dataset size	$5\%, 15\%, \ldots, 45\%$	Algorithm 1
N_P	Number of Principal Components used by PCA	10	Algorithm 1
α	The Accepted Accuracy	99%	Algorithms 2,4
m	The number of MVC's updates used with the same batch	5	Algorithm 4
Time Budget	The max number of times to re-train the vision model	200	Algorithm 4

the label of the image based on the sequence of feature vectors.

VGG16 is a deep convolutional neural network with 16 layers, including 13 convolutional layers, 3 Max Pooling layers and 3 fully connected layers [56].

5.2.5 Parameters Setup

The experiment parameters are shown in Table 5.2. .We apply each human-assisted labelling approach (i.e., Human-in-the-loop and pseudo-labelling) to the datasets mentioned in Section 5.2.3. Specifically, we consider five sizes of training set in the preprocessing phase for each dataset (i.e., T_N). These sizes denote the number of labelled images used for training the models. For Algorithm 2, we use the three vision models mentioned in Section 5.2.4 and for Algorithm 4, we choose ViT due to its superior accuracy compared to other models based on the experiments performed by Dosovitskiy et al. [16]. As for our feature extractor, we use VGG-16 which is commonly used for extracting features from images [56]. The Number of Principal Components (N_p) is set to a relatively small value (e.g. 10) based on our preliminary experiments.

The accepted accuracy used in both approaches is set to 99% since we expect the models to reach very high accuracy. In the pseudo labelling approach (i.e., Algorithm 4), we use five for the number of model's updates in each iteration (m). This is based on our preliminary experiments that show the model can effectively learn new patterns in the training dataset within a maximum of nine epochs. The time budget considered in Algorithm 4 is set to 200 which denotes the number of iterations discussed in chapter 4. In our preliminary experiments, the model reaches a plateau within this time budget. To account for randomness, we run each approach five times. Therefore, for the Human-in-the-loop approach we have $7 \times 5 \times 5 = 175$ experiments, which refer to 7 datasets, 5 D_T sizes, and 5 repetitions. Further, for the pseudo labelling approach we have $7 \times 5 \times 5 = 175$ experiments, which refer to 7 datasets, 5 D_T sizes, and 5 repetitions. Running these experiments takes about 20 days. All the experiments were executed on a high-performance server equipped with two Intel Xeon Gold 6338 processors, DDR 4 and four A40 GPUs.

In addition, for the individual model baselines, we need to run experiments for each of our three models (VIT, ResNET50 and VGG16). That is we perform, $3 \times 7 \times 5 \times 5 = 525$ for baselines. The baselines use the same parameter selection as the Human-in-the-loop approach.

Chapter 6

Evaluation Results and Analysis

In this section, we present the evaluation results and analysis of the experiments conducted to address the research questions outlined in Section 5.1.

6.1 Results for RQ1

In this section, we present our results for answering our first research question: *How do our approaches compare with respect to human effort and accuracy?*

6.1.1 Pareto Graphs

To address Research Question 1, we compare the performance of our two methods against baseline models across various datasets. As outlined in Chapter 5 Section 5.2, we conduct comprehensive tests on seven public and private datasets, evaluating performance using

two key metrics: accuracy and human effort. We first assess the performance of each method using 5%, 15%, 25%, 35%, and 45% of the data for training.

To visualize and compare these metrics, we employed Pareto graphs, which illustrate the optimal balance between two competing metrics. The Pareto graph, also known as the Pareto frontier, plots the two performance metrics on a scatter plot. Each point on the graph represents the performance of a specific model, with one axis showing accuracy and the other representing human effort. The Pareto frontier consists of a set of points that are not dominated by any other point in both metrics. In other words, a point on the Pareto frontier is considered optimal if no other point has both higher accuracy and lower human effort.

In our study, the Pareto graph effectively enables us to compare our methods with the baseline models. By plotting the results, we can easily identify which models were on the Pareto frontier, indicating that they offered the best trade-off between accuracy and human effort.

6.1.2 Performance of Models with Different Training Data

As shown in Figure. 6.1, when using 1% of the data for training, we observed varying performance across different datasets. For the baseline models and the pseudo-labeling method, since the initial training set is equivalent to the required human effort, models with higher labeling accuracy remain on the Pareto graph. The baseline VGG16 model shows higher labeling accuracy on the SVHN and Fashion MNIST datasets, with accuracies of 76% and 88%, respectively. On the CIFAR10, CIFAR10-c, ImageNet-c, and SmartInside

datasets, the baseline ViT model performs better, achieving accuracies between 84% and 95%. The pseudo-labeling method also demonstrates high accuracy on the MNIST dataset, nearing 95%.

Compared to the baseline models, the Human-in-the-Loop method requires more human effort but achieves higher system accuracy. For our target accuracy of 99The significant accuracy advantage of the Human-in-the-Loop method over other approaches stems from its use of a Discriminator to delegate challenging samples to human annotators. While model confidence scores can sometimes be unreliable due to hallucination issues—where models assign high confidence to incorrect predictions—our experiments show that the Discriminator mechanism remains effective even in such scenarios. For example, on the SVHN dataset, where models tend to perform poorly and exhibit hallucination, the Discriminator successfully delegates a majority of samples to human annotators. This ensures that the overall accuracy stays at the expected level. This robustness can be attributed to the data preprocessing stage, which ensures that the training data for the Discriminator is representative. Consequently, the Discriminator learns to set appropriately high confidence thresholds when the models underperform on validation dataset, thereby mitigating the impact of hallucination and preventing high-confidence errors from propagating into the final labels. This mechanism significantly contributes to the reliability and superior performance of the Human-in-the-Loop approach.

As shown in Figure. 6.2, when using 5% of the data for training, similar results are observed, though the models with the best performance across datasets changed. At 5% human effort, the baseline ViT performs best on the CIFAR10, CIFAR10-c, ImageNet-c, and SmartInside datasets, with accuracy levels ranging from 85% to 95%. On the MNIST

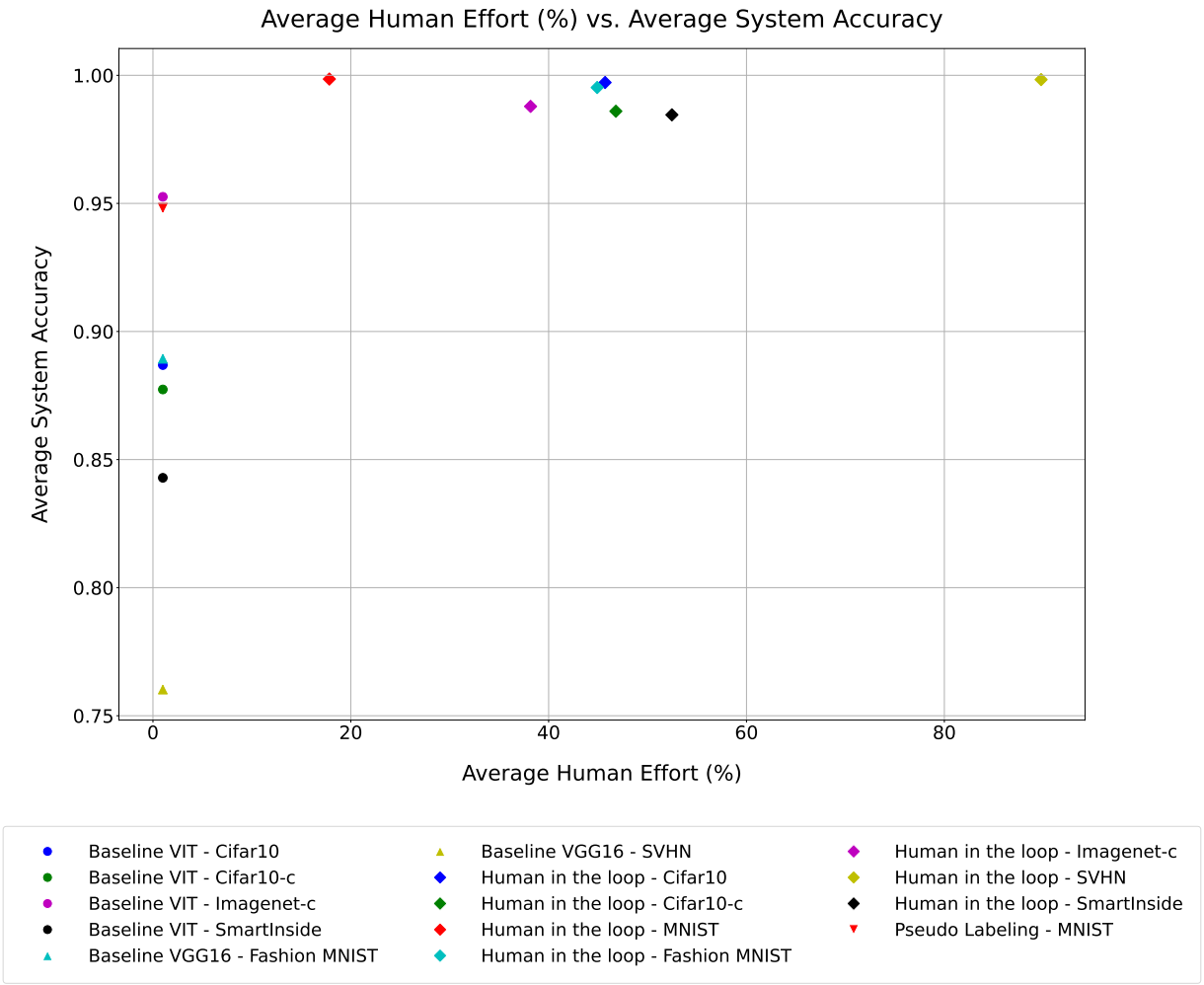

Figure 6.1: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 1% data for training.

and Fashion MNIST datasets, the baseline VGG16 model achieves higher accuracy, with values of 97% and 88%, respectively.

The Human-in-the-Loop method again demonstrates high accuracy across different datasets. Notably, with more initial training data, the visual models within Human-in-the-Loop exhibit stronger performance during fine-tuning, which reduces the human effort required during the inference phase for ImageNet-c and CIFAR10-c. As a result, the overall human effort is lower. However, for CIFAR10 and SmartInside, using more data does not significantly improve the model's initial performance, leading to increased human effort due to the larger amount of labeled data used for training.

As shown in Figure. 6.3, we observe improvements in the performance of both baseline models and the Human-in-the-Loop method with the training dataset expanded to 15%. At 15% human effort, the baseline models show enhanced performance. This improvement is attributed both to the increased accuracy of the models due to more training data and to the greater use of human-labeled data in the final labeled dataset. Consequently, the performance of the baseline models improves. The pseudo-labeling method performs particularly well on the Fashion MNIST dataset, highlighting its efficiency on specific datasets. Meanwhile, the Human-in-the-loop method continues to demonstrate higher system accuracy across multiple datasets, while also reducing the required human effort on ImageNet-c and CIFAR10-c.

As shown in Figure. 6.4, when using 25% of the data for training, overall model performance improves further, although variations among methods across different datasets are still noticeable. The baseline ViT model exhibits high accuracy on the CIFAR10,

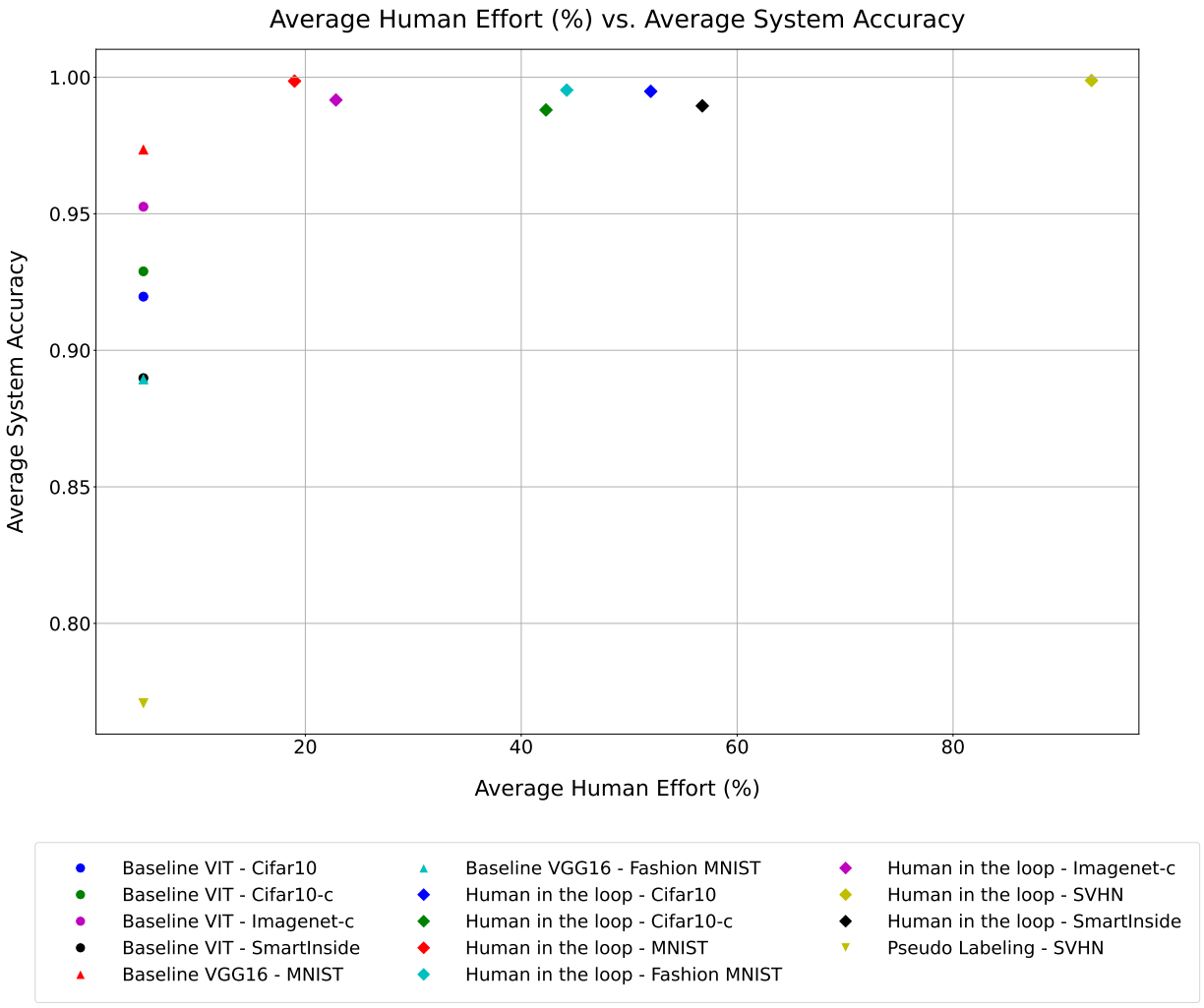

Figure 6.2: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 5% data for training.

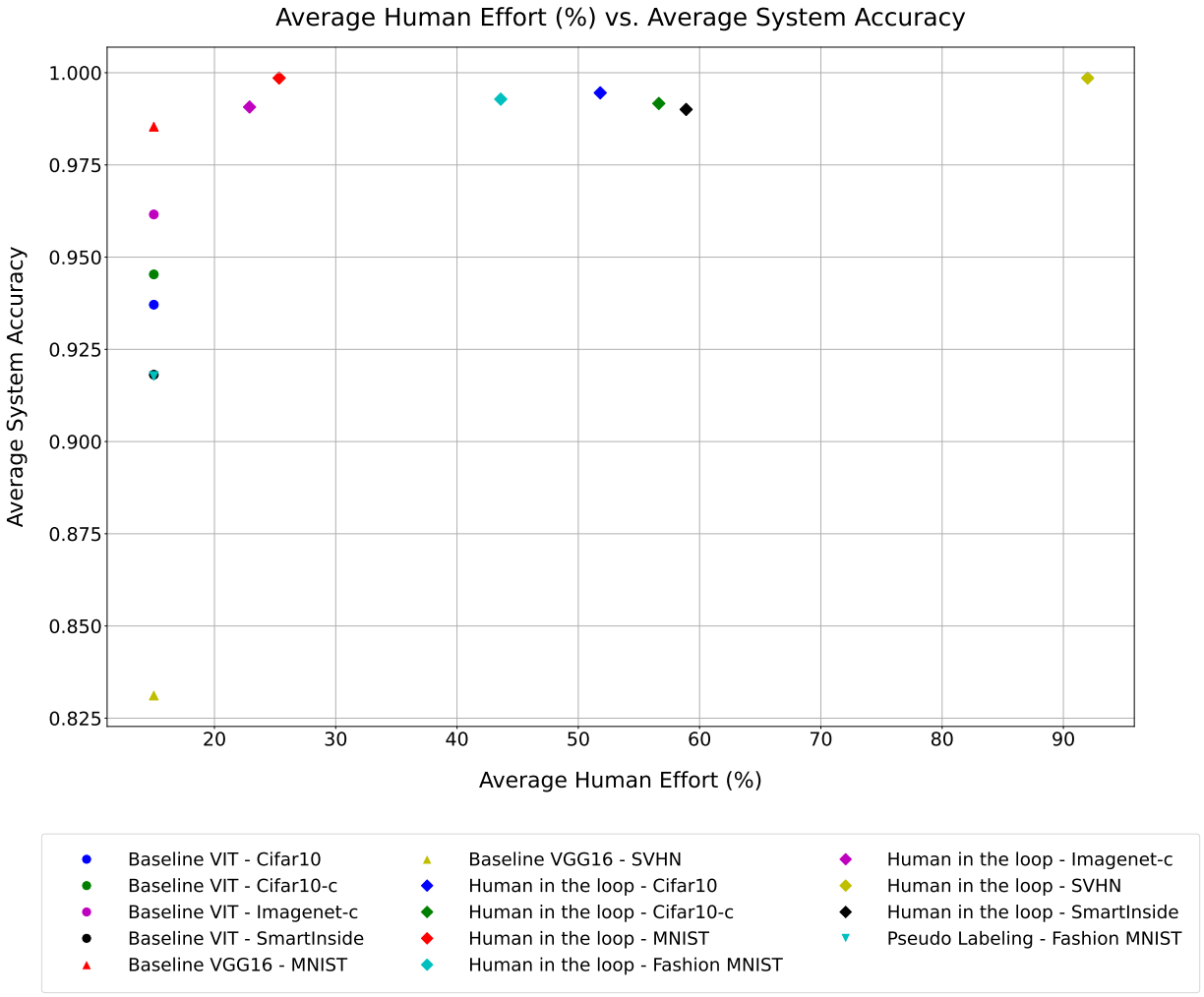

Figure 6.3: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 15% data for training.

CIFAR10-c, ImageNet-c, and SmartInside datasets, with accuracy ranging from 93% to 96%, and human effort matching the training data proportion at around 25%. The baseline VGG16 model excels on the MNIST and SVHN datasets, achieving nearly 99% accuracy on MNIST, 93% on Fashion MNIST, and approximately 86% on SVHN. However, despite the availability of more training data, the pseudo-labeling method does not surpass other models. It is noteworthy that, even when the baseline models on MNIST use the same amount of human effort as the Human-in-the-loop method (as shown in Figure. 6.1), their accuracy remains lower than that of Human-in-the-loop.

The Human-in-the-loop method, with more training data, still outperforms the baseline models, maintaining system accuracy around 99% across all seven datasets. However, on CIFAR10-c, MNIST, and ImageNet-c, the human effort required increases compared to using 15% of the data for training. This is because the additional training data does not significantly enhance model performance but instead consumes more human effort.

As shown in Figure. 6.5, when the training data increases to 35%, we observe similar trends. For baseline methods, human effort remains fixed at the level of the initial training data, but only the VGG16 model on MNIST achieves accuracy comparable to the Human-in-the-loop method.

The Human-in-the-loop method continues to maintain higher system accuracy, though the increase in training data from 25% to 35% results in higher total human effort for datasets like MNIST, ImageNet-c, SmartInside, and CIFAR10-c. This is due to the diminishing returns in model performance improvement from the additional training data, which does not sufficiently offset the increase in human effort. For CIFAR10, Fashion MNIST,

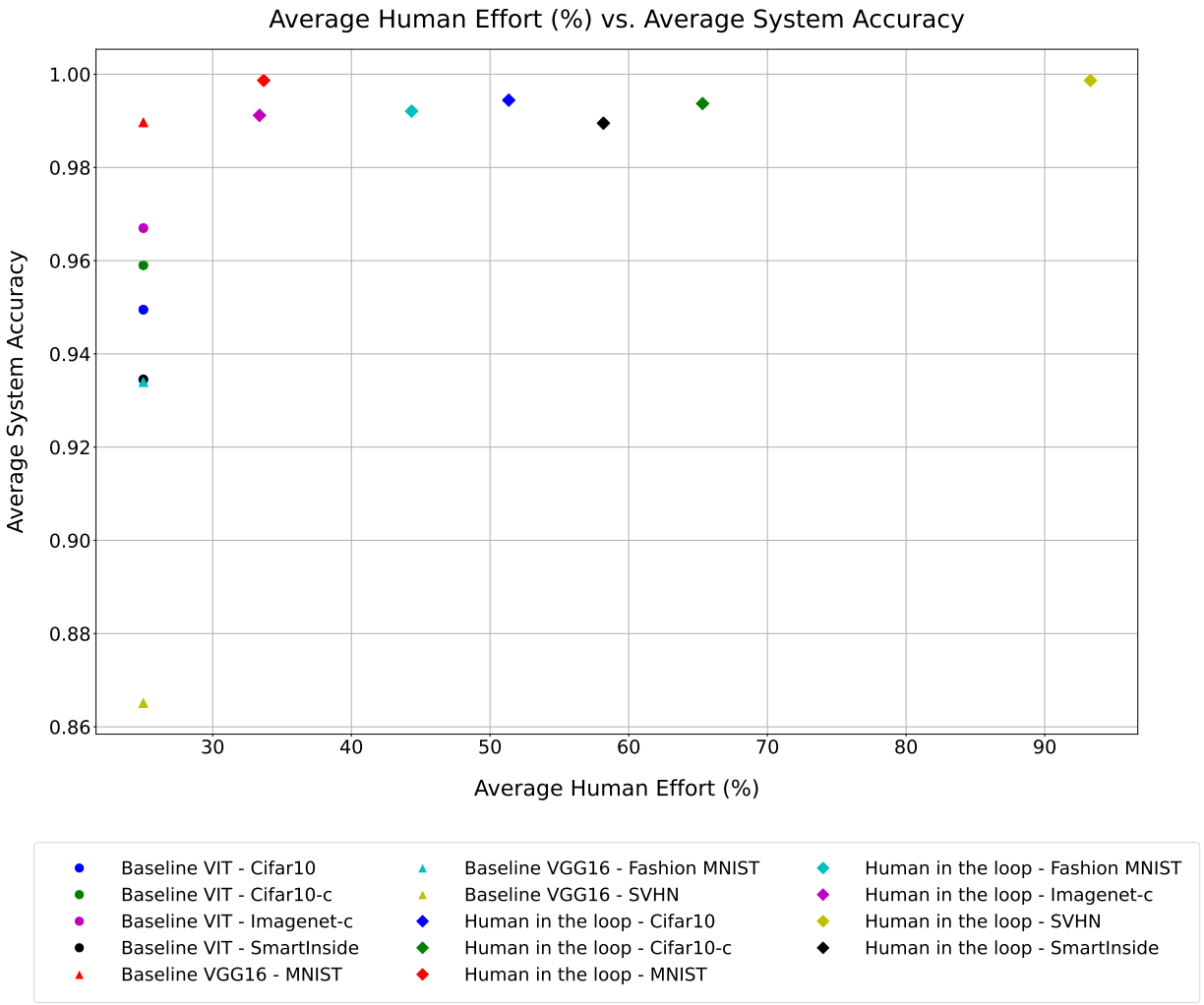

Figure 6.4: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 25% data for training.

and SVHN, the changes are minimal, indicating a balanced trade-off between the benefits and costs.

Figure. 6.6 shows that at 45% training data, our observations remain consistent with the previous findings.

To validate our analysis, we provide an additional Pareto graph of the performance of models with training data proportions of 1%, 5%, 15%, 25%, 35%, and 45% across different datasets as a reference (Figure. 6.7). A comprehensive Pareto optimality analysis is conducted for each model-dataset pair, revealing the trade-offs between average system accuracy and average human effort across various training data sizes. The results presented in the figures align with our previous analysis, further validating the performance differences between models in terms of accuracy and human effort across different datasets and training data conditions.

Our experimental results indicate significant differences in model performance and required human effort across different datasets and training data proportions. Baseline models such as VGG16 and ViT excel on certain datasets, but their performance is limited by fixed human effort. In contrast, while the Human-in-the-loop method requires more human effort, it generally provides higher system accuracy, particularly when aiming for accuracy close to that of human annotators. We also observe that the Human-in-the-loop method outperforms the baseline models in terms of both accuracy and human effort on the MNIST dataset, even with limited training data. Notably, on the MNIST dataset, where the baseline accuracy is highest, it still cannot surpass Human-in-the-loop. Furthermore, Human-in-the-loop achieves higher accuracy than the baseline models with less

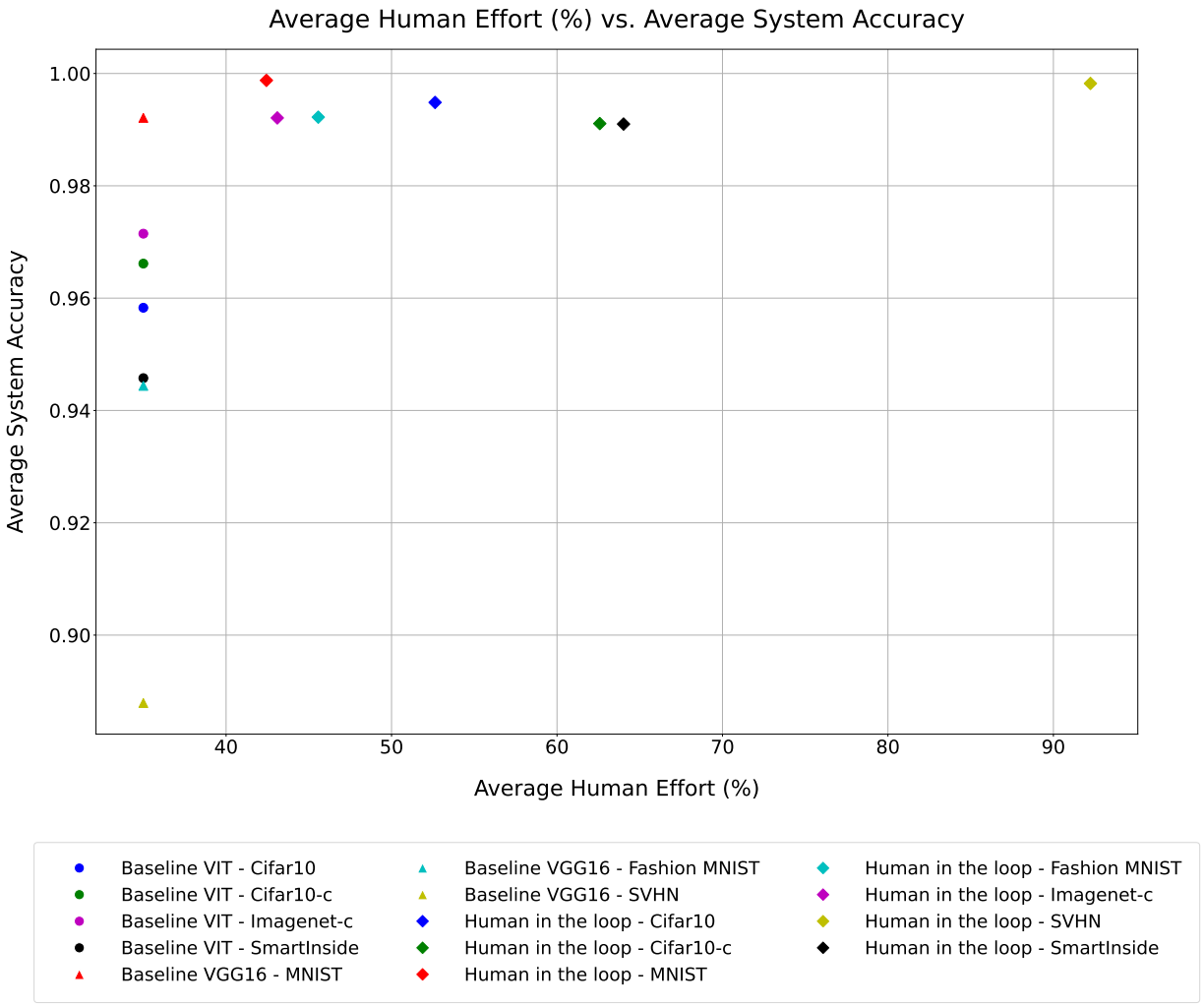

Figure 6.5: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 35% data for training.

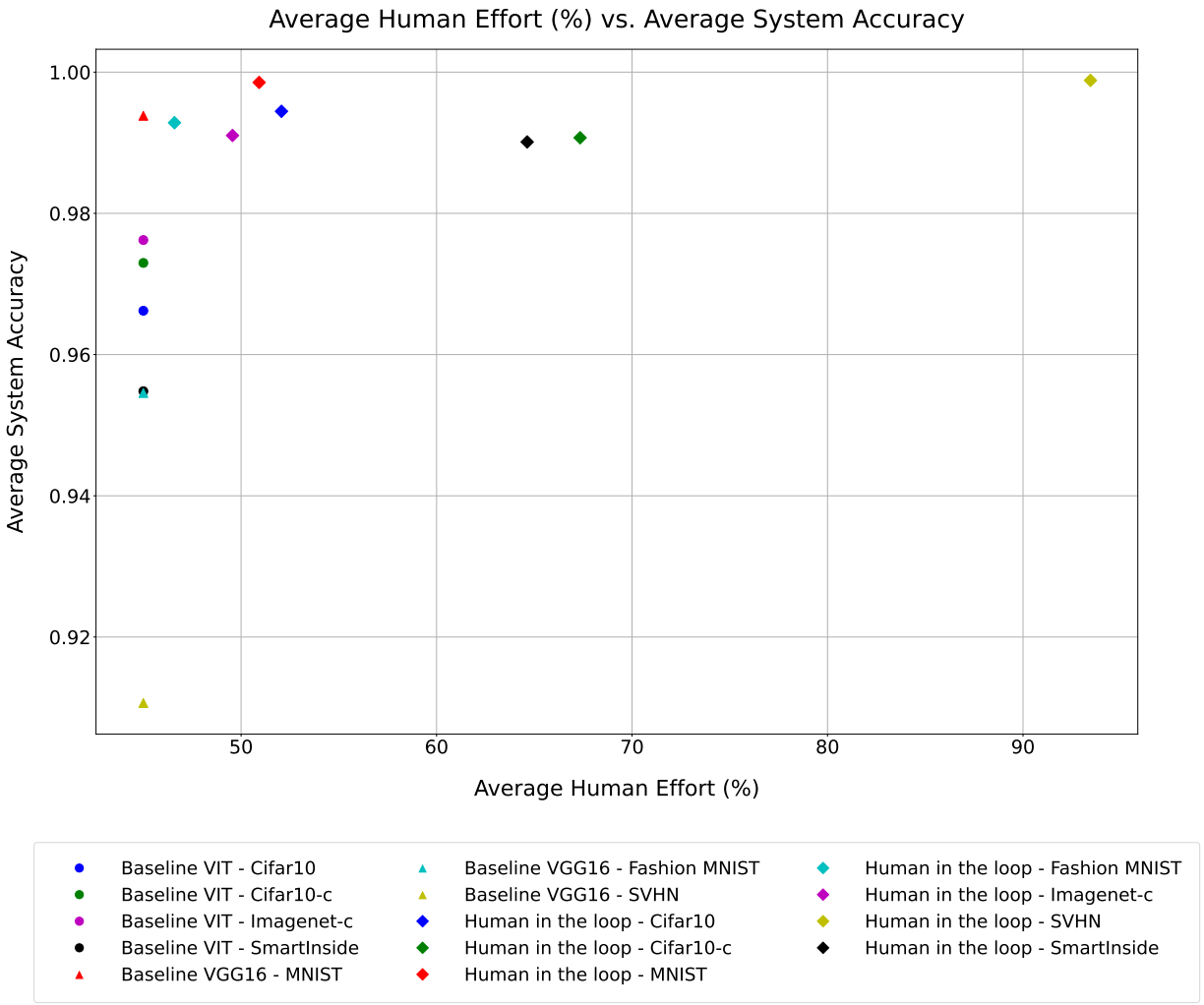

Figure 6.6: Pareto graph illustrating the trade-off between average system accuracy and average human effort for various models using 45% data for training.

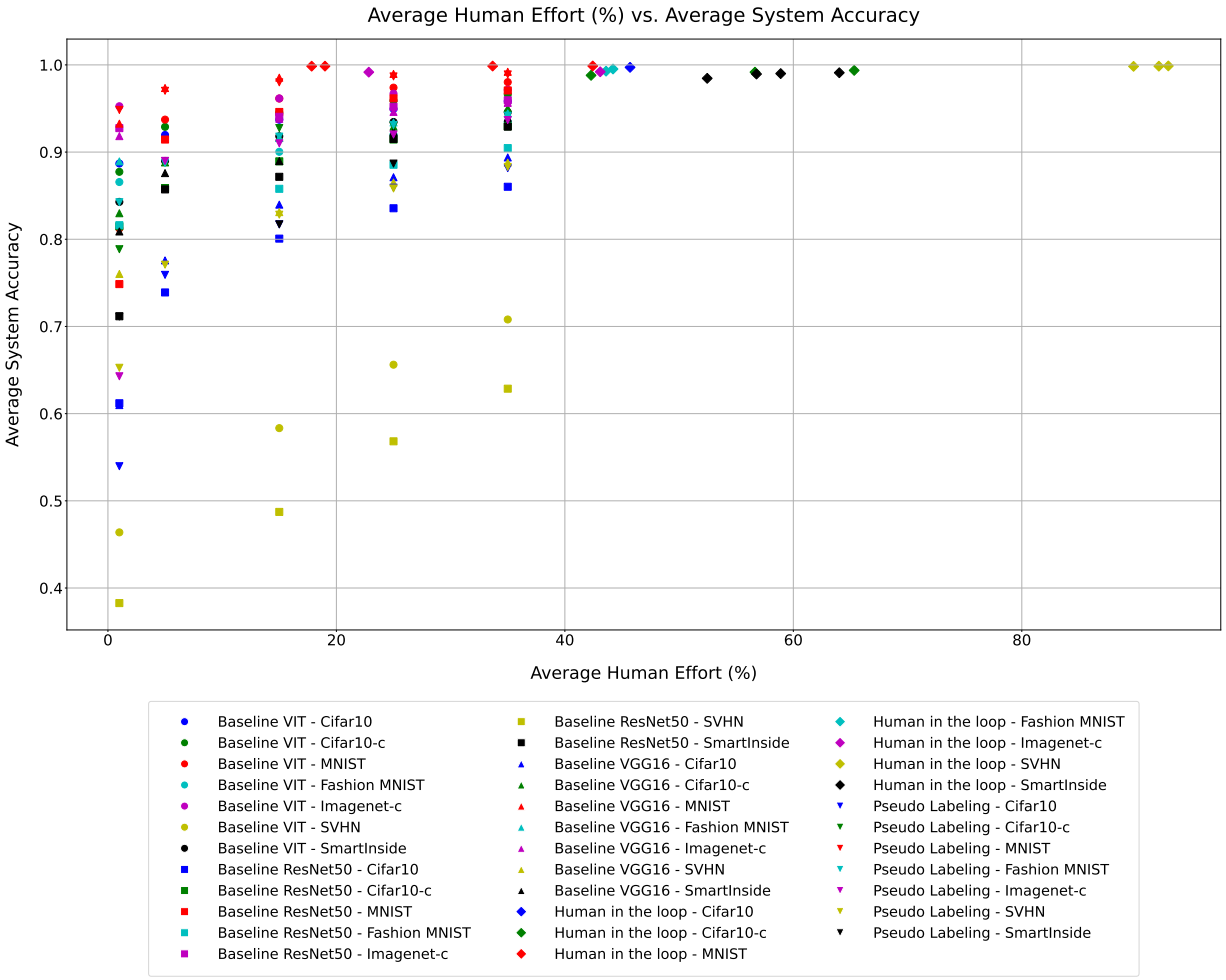

Figure 6.7: The Pareto graph summarizing model-dataset pair performance across varying training data proportions (1%, 5%, 15%, 25%, 35%, 45%).

human effort.

When the proportion of training data is small, the Human-in-the-Loop method exhibits higher labeling accuracy on most datasets, and the system's performance improves with additional training data. However, as the training data proportion increases further, the human effort required by the Human-in-the-Loop method also increases, but the trade-off between system accuracy and human effort remains better than that of the baseline models.

Moreover, while the pseudo-labeling method demonstrates its strengths on specific datasets, it generally does not outperform the Human-in-the-Loop method or baseline models, especially with higher proportions of training data.

Overall, the Human-in-the-Loop method provides the best performance trade-off on the Pareto frontier in most cases, achieving high accuracy while minimizing human effort. The results from the Pareto graph validate the effectiveness of our proposed method in handling training data of different scales and provide valuable insights for future research.

6.1.3 Performance of Models with Limited Human Effort

To further evaluate the real-world applicability of our models and the baseline models, we test their performance when human effort is limited to 5%, 15%, 25%, 35%, and 45%. As discussed in Chapter 5 Section 5.1, we set an upper limit on human effort for the Human-in-the-loop method in this experiment. When the human effort exceeds this limit, Human-in-the-loop is converted into the ensemble model described in Chapter 4 Section 4.2.

As shown in Tables 6.1 and 6.2, the baseline models generally outperform the pseudo-labeling method across most datasets, especially as the proportion of human effort increases. For example, on the CIFAR-10 dataset, ViT achieves an accuracy of 96.62% with 45% human effort, while pseudo-labeling only reaches 90.24% under the same conditions. This trend is consistent across other datasets, such as CIFAR-10c and Fashion MNIST, where the baseline models, particularly ViT and VGG-16, exhibit superior performance compared to pseudo-labeling.

We find that pseudo-labeling can still deliver competitive accuracy at lower levels of human effort, even though its overall performance does not surpass the baseline models. For instance, on the Fashion MNIST dataset, pseudo-labeling achieves 91.78% accuracy with 15% human effort, slightly outperforming VGG16 and ViT, which reach 91.61% and 90.03%, respectively. However, in other datasets, pseudo-labeling's performance falls short of the baseline models, especially VGG and ViT. This indicates that while pseudo-labeling is effective in scenarios with limited human effort, it struggles to match the accuracy of baseline models as human effort increases.

On the other hand, the Human-in-the-loop method demonstrates a clear advantage, particularly at higher levels of human effort. For example, on datasets like CIFAR-10-c, MNIST, ImageNet-c, and SmartInside, Human-in-the-loop consistently delivers the highest accuracy under the same human effort conditions.

However, we also observe that in environments where human effort is severely limited, Human-in-the-loop's performance lags behind that of the baseline models on some datasets. For instance, on the CIFAR-10 dataset, Human-in-the-loop achieves accuracies of 89.02%

and 92.51% with 5% and 15% human effort, respectively, while the ViT model outperforms it with accuracies of 91.97% and 93.71% under the same conditions—a margin of about 2%. A similar trend is observed on the Fashion MNIST dataset, where Human-in-the-loop's accuracy at 5% human effort is 88.81%, slightly behind the baseline model VGG-16, which achieves 88.93%. This is likely due to the fact that, with limited data, the human-in-the-loop mechanism does not filter enough data to significantly improve the visual models.

It is also worth noting that Human-in-the-loop performs poorly on the SVHN dataset, likely because the dataset's complexity makes it difficult for the visual models to make accurate judgments, leading to imbalanced performance among the models used.

It is also worth noting that Human-in-the-loop performs poorly on the SVHN dataset, likely because the dataset's complexity makes it difficult for the visual models to make accurate judgments, leading to imbalanced performance among the models used. Specifically, VGG-16 outperforms ResNet-50 and ViT on this dataset, which results in the ensemble model underperforming compared to the single VGG-16 model. Interestingly, pseudo-labeling performs similarly to the best baseline model, VGG-16, on this dataset, even surpassing VGG-16 when human effort is limited to 5

From Tables 6.1 and 6.2, it can also be observed that Baselines and pseudo-labeling perform poorly on this dataset, especially ResNet-50 and ViT. This directly impacts the accuracy of the Human-in-the-loop approach when human effort is limited, resulting in the ensemble decision being less effective than the decision of the single VGG-16 model.

To understand why models struggle on this dataset, we conducted a detailed inspection of the data. We found that the image labels are determined solely by the digit at the center,

but the images often contain additional digits in the background. This could confuse the models, leading to their poor performance on this dataset.

These findings suggest that while pseudo-labeling may be a cost-effective solution in scenarios with minimal human resources, it does not outperform baseline models (especially ViT and VGG) overall, particularly when sufficient human effort is available. The Human-in-the-loop method proves to be the most effective, balancing human effort and high accuracy, though it may underperform on a few datasets with low human effort. Nevertheless, Human-in-the-loop consistently performs well across most datasets, making it a strong choice for applications where maximizing accuracy is critical, especially when adequate human effort is available.

6.2 Evaluation for RQ2

In this section, we present our results for answering our second research question: *Can our techniques produce test sets with very high accuracy, close to 99%?*

6.2.1 Results for Research Question 2

Given the demand for high accuracy in test datasets, we further investigate whether our proposed methods can achieve a labeling accuracy of 99%. As discussed in Chapter 5 Section 5.1, we again face the challenge of differing human effort levels between the Human-in-the-loop and baseline models, so we use Pareto graphs to present our results.

Figure. 6.8 shows that as human effort increases, the system accuracy of the baseline

Table 6.1: Method Comparison: Classification Accuracy on Different Datasets (ResNet, VIT, VGG)

	Resnet-50					VIT					VGG-16				
	5%	15%	25%	35%	45%	5%	15%	25%	35%	45%	5%	15%	25%	35%	45%
Cifar10	73.90%	80.07%	83.56%	86.02%	88.57%	91.97%	93.71%	94.95%	95.83%	96.62%	77.58%	83.98%	87.12%	89.39%	91.30%
Cifar10-c	85.88%	88.94%	91.44%	92.96%	94.36%	92.89%	94.53%	95.90%	96.62%	97.30%	88.81%	91.65%	92.92%	95.02%	95.68%
MNIST	91.44%	94.60%	96.16%	97.07%	97.57%	93.73%	96.14%	97.39%	98.02%	98.39%	97.35%	98.54%	98.97%	99.21%	99.38%
Fashion MNIST	81.61%	85.79%	88.55%	90.46%	92.29%	86.57%	90.03%	92.00%	93.47%	94.68%	88.93%	91.61%	93.40%	94.44%	95.46%
Imagenet-c	92.75%	94.05%	95.20%	95.94%	96.68%	95.26%	96.16%	96.70%	97.15%	97.62%	91.83%	93.83%	94.61%	95.65%	96.45%
SVHN	38.27%	48.73%	56.82%	62.86%	69.40%	46.38%	58.34%	65.61%	70.80%	76.41%	76.02%	83.11%	86.51%	88.79%	91.07%
SmartInside	85.72%	87.16%	91.51%	92.92%	94.19%	88.98%	91.81%	93.45%	94.58%	95.48%	87.59%	88.96%	92.35%	93.94%	94.98%

Table 6.2: Method Comparison: Classification Accuracy on Different Datasets (Pseudo Labelling, Human in Loop)

	Pseudo-labeling					Human in the Loop				
	5%	15%	25%	35%	45%	5%	15%	25%	35%	45%
Cifar10	75.92%	81.71%	85.92%	88.16%	90.24%	89.02%	92.51%	95.89%	98.08%	99.65%
Cifar10-c	78.63%	92.76%	93.01%	91.65%	92.09%	93.72%	97.82%	99.18%	99.78%	99.78%
MNIST	97.03%	97.98%	98.68%	98.90%	99.15%	97.81%	99.95%	99.95%	99.95%	99.95%
Fashion MNIST	88.76%	91.78%	93.12%	94.17%	95.05%	88.81%	92.77%	95.38%	97.16%	99.54%
Imagenet-c	88.99%	91.00%	92.01%	93.67%	93.81%	96.39%	97.77%	98.94%	98.94%	98.94%
SVHN	77.08%	82.82%	85.83%	88.30%	90.31%	53.28%	67.42%	72.53%	77.47%	82.13%
SmartInside	69.54%	81.71%	88.68%	88.19%	88.01%	90.45%	95.13%	98.23%	99.95%	99.95%

models steadily improves. However, only the ViT model achieves competitive performance on the ImageNet-c and SVHN datasets. The ViT model attains nearly 98% accuracy on SVHN with 55% human effort. On the ImageNet-c dataset, it performs even better, reaching 99.75% accuracy with the same amount of human effort.

On the other hand, the Human-in-the-loop method consistently excels across all models, achieving over 99% accuracy on datasets like CIFAR10-c, MNIST, and Fashion MNIST, while requiring less human effort than the baseline models. Although the Human-in-the-loop method requires relatively high human effort on the SVHN dataset, this is due to the poor performance of ResNet50 and ViT on SVHN, which leads the discriminator to lack confidence in the models' predictions and favor handing the data over to human annotators. However, even on SVHN, the Human-in-the-Loop method is still able to save a portion of the Human Effort.

To verify our results, we provide an additional figure (Figure. 6.9) that presents results without applying Pareto optimality analysis. The outcomes shown in this figure are consistent with our previous analysis, further confirming the superior performance of the Human-in-the-loop model across multiple datasets.

In summary, as human effort increases, the accuracy of the baseline models gradually improves. However, in scenarios with higher human effort, the Human-in-the-loop method consistently outperforms the baseline models. Human-in-the-loop achieves 99% accuracy across multiple datasets while requiring less human effort than the baseline models.

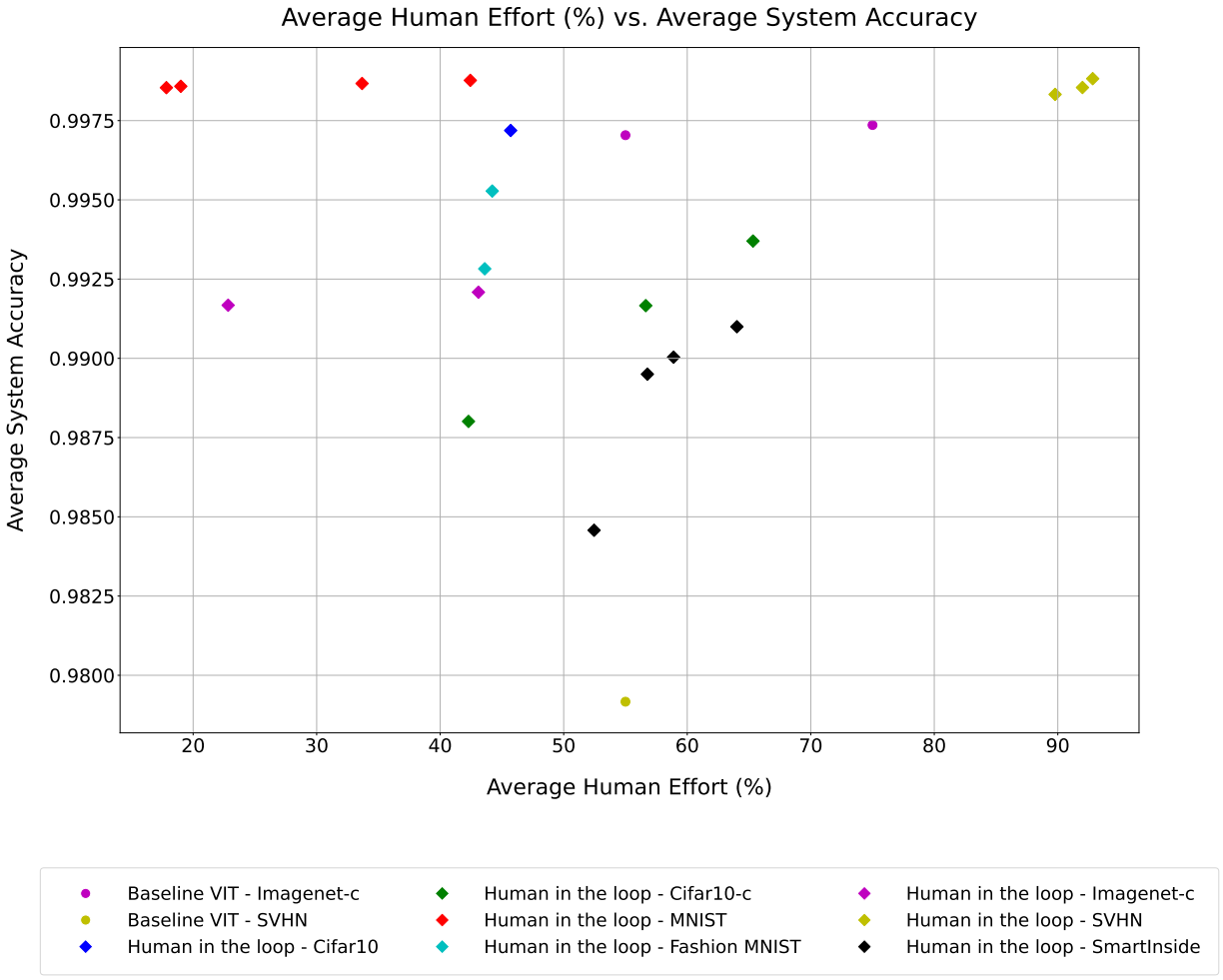

Figure 6.8: The Pareto graph of accuracy within high Human Effort conditions.

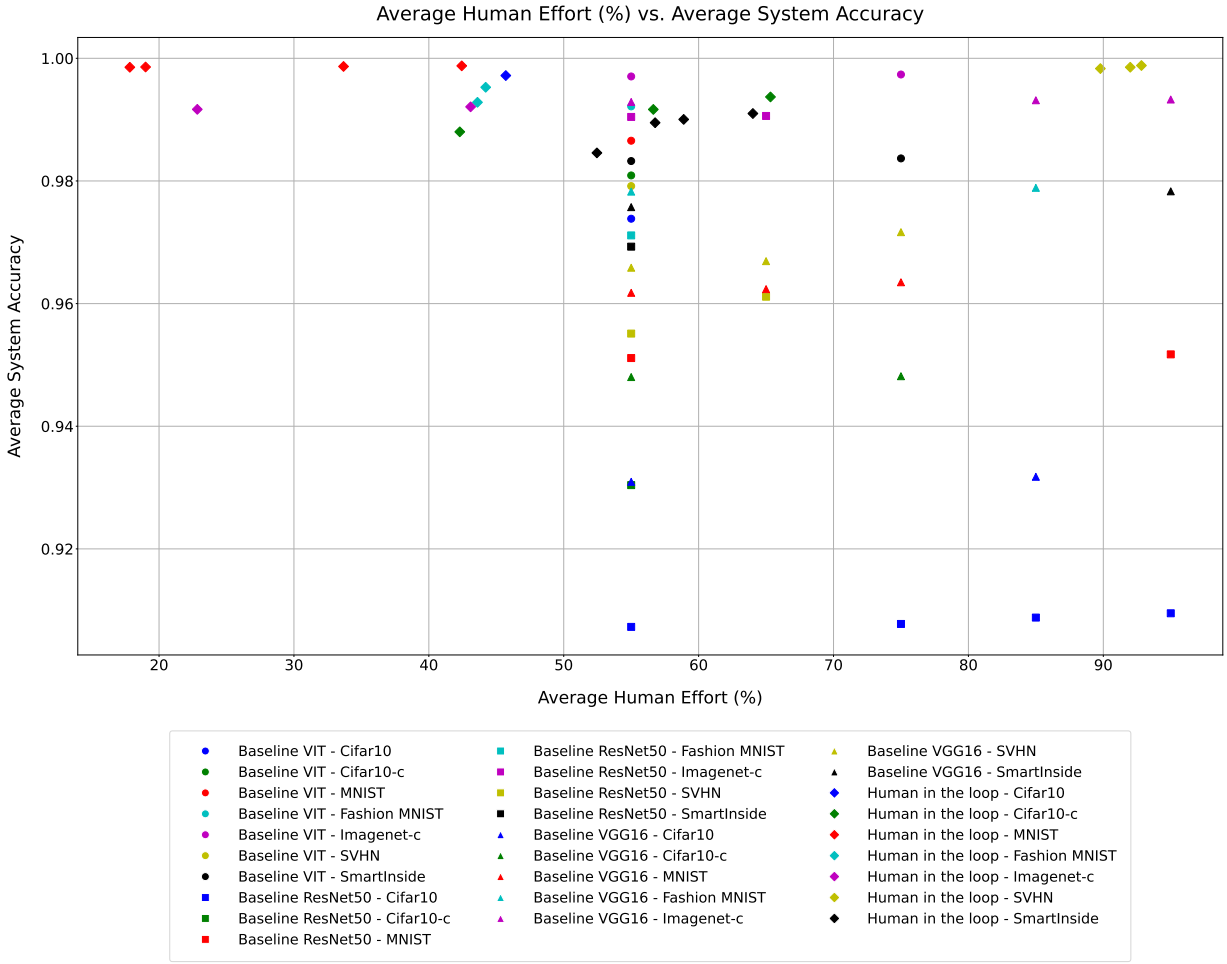

Figure 6.9: The performance of baseline and Human-in-the-loop under high Human Effort conditions.

6.3 Summary of Research Questions

This section summarizes the experimental results and provides a concluding suggestion for how to use our approaches.

RQ1: How do our approaches perform compared to baselines regarding human effort and accuracy? To address this question, we evaluated the performance of our methods across seven datasets, using varying proportions of training data (1%-45%) and visualizing the trade-offs with Pareto graphs (Section 6.1). Based on the Pareto graph results, we conclude that the Human-in-the-Loop method achieves high accuracy when sufficient human effort is available. Baseline models and pseudo-labeling can achieve acceptable accuracy with minimal human effort.

To further investigate whether Human-in-the-Loop remains the best option when human effort is limited, we compared a constrained Human-in-the-Loop method with baselines and pseudo-labeling. The results show that Human-in-the-Loop consistently achieves higher accuracy than baselines with comparable human effort across most datasets, except for SVHN, where pseudo-labeling performs better. Therefore, we conclude that Human-in-the-Loop provides the best trade-off between human effort and accuracy in most cases.

RQ2: Can our techniques produce test sets with very high accuracy, close to 99%? To answer this question, we tested the methods with 55%-95% human effort across multiple datasets and analyzed the results using Pareto graphs to evaluate the trade-offs between accuracy and effort (Section 6.2). The findings reveal that Human-in-the-Loop consistently outperforms baselines and pseudo-labeling when the goal is to achieve near-perfect accuracy with increased human effort.

In summary, Human-in-the-Loop is the optimal choice in most scenarios, particularly when accuracy is a priority. However, when human resources are limited, Human-in-the-Loop should be the first method to try. If the Human-in-the-Loop model performs well, it can be continued or adapted using a transformed version of Human-in-the-Loop. In cases where Human-in-the-Loop struggles, such as on the SVHN dataset, pseudo-labeling becomes a viable alternative.

Chapter 7

Conclusion and Future Work

In this paper, we explored the key challenge of reducing human effort while ensuring the accuracy of labeled test data for deep learning systems. We proposed and evaluated two automated labeling techniques—Human-in-the-Loop and Pseudo-Labeling—to address the trade-off between human effort and labeling accuracy. Our methods were rigorously tested on seven different datasets and comprehensively compared with baseline models such as VGG16, ResNet50, and ViT_13B.

Our results demonstrate that Human-in-the-loop is particularly effective at balancing human effort and accuracy, especially when high labeling precision is required. By combining human expertise with machine learning, Human-in-the-loop consistently outperformed baseline models in most cases, achieving near-human accuracy with significantly reduced human effort. This approach is particularly advantageous when the goal is to achieve close to 99% accuracy, as it optimally leverages human resources while maintaining high system

reliability.

In contrast, while the pseudo-labeling method has certain advantages in scenarios with limited human resources, it generally fails to outperform Human-in-the-loop and the baseline models when larger amounts of labeled data are available. This suggests that while pseudo-labeling may be a cost-effective solution in resource-constrained environments, its application may be limited in scenarios where accuracy is critical.

One key conclusion from our evaluation is that the amount of initial training data is crucial to determining the effectiveness of these labeling methods. We observed that models initialized with a higher proportion of training data generally performed better, although this required more human effort. Notably, the Human-in-the-loop method exhibited a more favorable balance between accuracy and workload across various datasets, particularly as the amount of training data increased.

Our study also highlights the limitations of each method. For example, Human-in-the-loop did not perform as impressively on datasets like SVHN, where the complexity of the data made it difficult for the model to effectively collaborate with human annotators. These findings underscore the need for further refinement of these methods to address the specific challenges posed by different types of data.

In summary, our research provides valuable insights into the practical application of automated and semi-automated labeling techniques for deep learning system testing. For scenarios requiring high precision with minimal human intervention, the Human-in-the-loop method stands out as a reliable solution. However, the choice of labeling technique should be guided by the specific requirements of the application, including the availability

of human resources and the desired level of accuracy.

7.0.1 Future Work

Looking ahead, we plan to explore several directions to further enhance this work:

- **Expanding adaptability to a broader range of datasets:** We intend to extend these methods to more diverse and popular image datasets such as COCO, Panda-70M, and Google Open Images.
- **Developing more sophisticated initial data selection and augmentation methods:** This will help further optimize the balance between human effort and system performance, leading to more cost-effective and accurate labeling solutions. One possible direction is to apply data augmentation techniques to the selected data to build stronger initial models.
- **Improving few-shot learning methods:** We see potential in combining Consistency Regularization techniques with pseudo-labeling to enhance the performance of few-shot learning methods. Recent studies have shown promising results when these techniques are used together, particularly in scenarios with limited labeled data [29, 57]. This suggests that further exploration of few-shot learning methods could be beneficial, especially for tasks that rely on using minimal labeled data for automatic annotation. Investigating the effectiveness of such methods in improving both accuracy and efficiency in these contexts is a compelling direction for future work.

- **Exploring noisy test data for DL system evaluation:** Investigating how to assess DL systems using noisy test data is another promising research avenue. By evaluating the accuracy of the automatic or semi-automatic labeling tool on validation sets, we can estimate the performance of DL systems on noisy datasets. A simple approach could be to multiply the model's performance on the noisy dataset by the estimated dataset accuracy, providing a lower-bound estimate of model performance. Addressing this issue could lead to more effective automatic or semi-automatic labeling methods for test datasets.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Niall Adams. Semi-supervised Learning. *Journal of the Royal Statistical Society Series A*, 172(2):530–530, April 2009.

[3] Christian Artigues, Oumar Koné, Pierre Lopez, and Marcel Mongeau. Mixed-integer linear programming formulations. *Handbook on project management and scheduling vol. 1*, pages 17–41, 2015.

[4] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J Fleet. Synthetic data from diffusion models improves imagenet classification. *arXiv preprint arXiv:2304.08466*, 2023.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*, pages 404–417. Springer, 2006.

[6] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9368–9377, 2018.

[7] Ali Borji. Qualitative failures of image generation models and their application in detecting deepfakes. *Image and Vision Computing*, 137:104771, 2023.

[8] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):1–99, 2018.

[9] John Burkardt. K-means clustering. *Virginia Tech, Advanced Research Computing, Interdisciplinary Center for Applied Mathematics*, 2009.

[10] Jongwon Choi, Kwang Moo Yi, Jihoon Kim, Jinho Choo, Byoungjip Kim, Jinyeop Chang, Youngjune Gwon, and Hyung Jin Chang. Vab-al: Incorporating class imbalance and difficulty with variational bayes for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6749–6758, 2021.

[11] David Cohn. *Active Learning*, pages 10–14. Springer US, Boston, MA, 2010.

[12] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15:201–221, 1994.

[13] Demet Demir, Aysu Betin Can, and Elif Surer. Distribution aware testing framework for deep neural networks. *IEEE Access*, 11:119481–119505, 2023.

[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] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[17] Weijie Fu, Meng Wang, Shijie Hao, and Xindong Wu. Scalable active learning by approximated error reduction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1396–1405, 2018.

[18] Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Query by committee made real. *Advances in neural information processing systems*, 18, 2005.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[20] Benedikt Grimmeisen, Mohammad Chegini, and Andreas Theissler. Visgil: machine learning-based visual guidance for interactive labeling. *The Visual Computer*, 39(10):5097–5119, 2023.

[21] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 739–743, 2018.

[22] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021.

[23] Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74, 2024.

[24] 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. IEEE, 2016.

[25] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and Xiaojuan Qi. Is synthetic data from generative models ready for image recognition? *arXiv preprint arXiv:2210.07574*, 2022.

[26] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

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

[28] Boyue Caroline Hu, Lina Marsso, Krzysztof Czarnecki, Rick Salay, Huakun Shen, and Marsha Chechik. If a human can see it, so should your system: Reliability requirements for machine vision components. In *Proceedings of the 44th International Conference on Software Engineering*, pages 1145–1156. IEEE, 2022.

[29] Reda Abdellah Kamraoui, Vinh-Thong Ta, Nicolas Papadakis, Fanny Compaire, José V Manjon, and Pierrick Coupé. Popcorn: Progressive pseudo-labeling with consistency regularization and neighboring. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part II 24*, pages 373–382. Springer, 2021.

[30] Sungmin Kang, Robert Feldt, and Shin Yoo. Sinvad: Search-based image space navigation for dnn image classifier test input generation. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 521–528. IEEE, 2020.

[31] Julian Katz-Samuels, Jifan Zhang, Lalit Jain, and Kevin Jamieson. Improved algorithms for agnostic pool-based active classification. In *International Conference on Machine Learning*, pages 5334–5344. PMLR, 2021.

[32] Kamran Khan, Saif Ur Rehman, Kamran Aziz, Simon Fong, and Sababady Sarasvady. Dbscan: Past, present and future. In *The fifth international conference on the appli-*

cations of digital information and web technologies (ICADIWT 2014), pages 232–238. IEEE, 2014.

[33] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[34] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

[35] A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009.

[36] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. ICML, 2013.

[37] Daiqing Li, Huan Ling, Seung Wook Kim, Karsten Kreis, Sanja Fidler, and Antonio Torralba. Bigdatasetgan: Synthesizing imagenet with pixel-wise annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21330–21340, 2022.

[38] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

[39] Tony Lindeberg. Scale invariant feature transform. *Scholarpedia*, 7(5):10491, 2012.

[40] Jose Martinez-Carranza, Delia Irazú Hernández-Farías, Victoria Eugenia Vazquez-Meza, Leticia Oyuki Rojas-Perez, and Aldrich Alfredo Cabrera-Ponce. A study on generative models for visual recognition of unknown scenes using a textual description. *Sensors*, 23(21):8757, 2023.

[41] Eduardo Mosqueira-Rey, Elena Hernández-Pereira, David Alonso-Ríos, José Bobes-Bascarán, and Ángel Fernández-Leal. Human-in-the-loop machine learning: a state of the art. *Artificial Intelligence Review*, 56(4):3005–3054, 2023.

[42] Quang Nguyen, Truong Vu, Anh Tran, and Khoi Nguyen. Dataset diffusion: Diffusion-based synthetic data generation for pixel-level semantic segmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

[43] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[44] Tehreem Qamar and Narmeen Zakaria Bawany. Understanding the black-box: towards interpretable and reliable deep learning models. *PeerJ Computer Science*, 9:e1629, 2023.

[45] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

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

[47] Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 876–888, 2020.

[48] Vincenzo Riccio and Paolo Tonella. When and why test generators for deep learning produce invalid inputs: an empirical study. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 1161–1173. IEEE, 2023.

[49] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*, 2021.

[50] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[51] Khushnaseeb Roshan and Aasim Zafar. Utilizing xai technique to improve autoencoder based model for computer network anomaly detection with shapley additive explanation (shap). *arXiv preprint arXiv:2112.08442*, 2021.

[52] 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*, 115:211–252, 2015.

[53] R Schmelzer. Data engineering, preparation, and labeling for ai 2019 cgr-de100, 2019.

[54] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Icml*, volume 2, page 6. Citeseer, 2000.

[55] Burr Settles. Active learning literature survey. *Science*, 10(3):237–304, 1995.

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[57] Tiberiu Sosea and Cornelia Caragea. Marginmatch: Improving semi-supervised learning with pseudo-margins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15773–15782, 2023.

[58] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, pages 359–371, 2020.

[59] M Sutharsan. Smart analysis of automated and semi-automated approaches to data annotation for machine learning. *ICTACT Journal on Data Science and Machine Learning*, 4:467–470, 2023.

[60] Jinhui Tang, Qiang Chen, Meng Wang, Shuicheng Yan, Tat-Seng Chua, and Ramesh Jain. Towards optimizing human labeling for interactive image tagging. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(4):1–18, 2013.

[61] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[62] Deepak Vasisht, Andreas Damianou, Manik Varma, and Ashish Kapoor. Active learning for sparse bayesian multilabel classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 472–481, 2014.

[63] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

[64] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. Want to reduce labeling cost? gpt-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205. EMNLP, 2021.

[65] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[66] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[67] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 93–102, 2019.

[68] Tianning Yuan, Fang Wan, Mengying Fu, Jianzhuang Liu, Songcen Xu, Xiangyang Ji, and Qixiang Ye. Multiple instance active learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5330–5339, 2021.

[69] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, pages 132–142, 2018.

[70] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155. IEEE, 2021.