Label-efficient model selection for pretrained classifiers

by

Jannes Kasper

Student Name	Student Number
Jannes Kasper	5847281

Supervisor:	Merve Gürel
Responsible Supervisor:	Jan van Gemert
Project Duration:	10.2023 - 07.2024
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science, Delft

Acknowledgement

I would like to thank my supervisor, Merve Gürel, for her guidance and support throughout the process of completing my master thesis. Her insights and encouragement were invaluable to this work. I also wish to thank my family and friends for their continuous support and understanding.

Abstract

In recent advancements within the field of machine learning (ML), the automation of model development and deployment enabled the maintenance of high-quality models in production through continuous retraining, yielding a variety of models for the same problem settings. The fast moving progress of Large-Language-Models and other research areas let to a spiking interest in ML, expanding applications and the market for ML models. Researchers, enthusiasts, programming frameworks and major companies such as Amazon are actively developing and evaluating models for a wide variety of tasks, thus contributing to the growing number of available pretrained models. These models are accessible through various platforms, including Huggingface or the AWS Sagemaker program.

The most common cause for retraining of models in production is a distribution shift in the data. Consequently, continuous retraining on changing production data results in a wide variety of models addressing the same problem, each with different strengths. Each retraining iteration demands substantial amounts of labeled data and significant time investment. Given the increasing availability of pretrained models with distinct strengths and weaknesses, there is strong reason to believe that, instead of retraining, selecting the most suitable pretrained model can yield sufficient performance. The primary challenge in the context of model selection is the need for evidence to assess the quality of the models, and querying labels is the most effective method to address this. Despite the progress in ML research, the acquisition of labeled data remains a significant challenge. Labeling data is mostly achieved through human labor, which is a costly, time consuming, and error-prone process. Estimates based on AWS Mechnical Turk indicate that the expenses of labeling large datasets or complex tasks, such as image segmentation, can easily reach six figures or more. For instance, labeling the entire ImageNet dataset with ten workers costs approximately 190.000€. This necessitates minimizing the number of labels required as evidence for model selection.

The existing literature on model selection primarily focuses on selection of a learning strategy in combination with its optimal hyperparameters to best fit the data. Although the objective of selecting the best model for the data remains the same, the setting in this thesis differs significantly. Specifically, the architecture and hyperparameters of the models are not relevant in this research, as they are predefined by the pretrained candidate models. Furthermore, the data used to assess the models is unlabeled, leading to a discussion on the suitability of alternative methods more closely aligned with this setting, such as active learning (AL), which focuses on intelligently labeling data points for training.

Given the lack of appropriate methods for this problem, this thesis introduces the Model Picker algorithm as a solution for selecting pretrained classifiers. The algorithm aims to minimize labeling cost by employing a probabilistic model to adaptively sample the most valuable instances from the unlabeled data. The informativeness of a data point is estimated using Shannon's mutual information between the latent variable, which represents the decision about the true best model, and the unknown labels of each data point, given the evidence sampled.

The Model Picker algorithm was rigorously evaluated against fundamental baselines such as Query-by-Committee, Active Comparison of Prediction Models, GALAXY, and QDD which were adapted to the setting when necessary. This evaluation utilizes a wide variety of well-established datasets, including up to 114 of different models each for each dataset, such as ImageNet. The results demonstrate that Model Picker consistently outperforms all baselines by a significant margin, with peak performance that allows for accurate model selection with only $1/4$ of the labels needed compared to the next best method. Additionally, this thesis investigates various methods for efficient optimization of Model Picker's single hyperparameter, ϵ, including strategies such as sampling a small subset of data for labeling and generating a noisy oracle. The proposed framework is intended to serve as a fundamental stepping stone for future research in the domain of model selection.

The thesis concludes with a discussion on the implications of the findings for the Model Picker algorithm, as well as for future research in the selection of pretrained classifiers. Furthermore, limitations of this study are discussed and potential future work to further enhance the Model Picker algorithm is proposed.

Contents

Abstract **ii**

Nomenclature **vii**

1 Introduction **1**

2 Background **3**
- 2.1 Model Selection . . . 3
- 2.2 Label-Efficient Learning . . . 4
  - 2.2.1 Active Learning . . . 4
  - 2.2.2 Semi-Supervised Learning . . . 6
  - 2.2.3 Ensemble Learning . . . 6
  - 2.2.4 Weak Supervision . . . 6
- 2.3 Main Takeaway . . . 6

3 Methodology **7**
- 3.1 Model Picker . . . 7
- 3.2 Baselines . . . 10
  - 3.2.1 Query by Committee . . . 10
  - 3.2.2 Active Comparison of Prediction Models . . . 11
  - 3.2.3 GALAXY . . . 12
  - 3.2.4 QDD (Query by Committee with Density and Diversity . . . 12

4 Experiments **14**
- 4.1 Experimental Protocol . . . 14
- 4.2 Experimental Setup . . . 14
  - 4.2.1 Datasets and Model Collection . . . 14
  - 4.2.2 Evaluation Metrics . . . 16
  - 4.2.3 Implementation Details . . . 17
- 4.3 Hyperparameter Optimization . . . 17
  - 4.3.1 Hyperparameter Optimization by True Oracle . . . 17
  - 4.3.2 Hyperparameter Optimization by Subset Sampling . . . 20
  - 4.3.3 Hyperparameter Optimization by Noisy Oracle Estimation . . . 21
- 4.4 Method Comparison . . . 24
- 4.5 Additional Experiments . . . 28
  - 4.5.1 Distribution Shift . . . 28
  - 4.5.2 Behavior Research . . . 30

5 Discussion **31**
- 5.1 Model Picker Behavior . . . 31
- 5.2 Hyperparameter Optimization . . . 32
- 5.3 Results in Context . . . 34

6 Conclusion **36**

References **37**

A Appendix **40**

List of Figures

3.1 Model Picker Probabilistic Model 7

4.1 Hyperparameter Optimization: True Oracle, Part 1 18
4.2 Hyperparameter Optimization: True Oracle, Part 2 19
4.3 Hyperparameter Optimization: Subset Sampling 20
4.4 Hyperparameter Optimization: Noisy Oracle Estimation, Part 1 23
4.5 Hyperparameter Optimization: Noisy Oracle Estimation, Part 2 24
4.6 Model Picker vs. Baselines, Part 2 26
4.7 Model Picker vs. Baselines, Part 3 27
4.8 Model Picker vs. Baselines, Part 4 28
4.9 Model Picker Entropy Behavior Experiment 30

5.1 Reward Ratio 33

A.1 Dataset Model Accuracy Distribution, Part 1 41
A.2 Dataset Model Accuracy Distribution, Part 2 42
A.3 Noisy Oracle Estimation Method Comparison 43
A.4 Hyperparameter Optimization: Wide Grid Search 45

List of Tables

4.1 Dataset Meta Data 15
4.2 Hyperparameter Optimization: Ground Truth Epsilon 18
4.3 Noisy Oracle Similarity Comparison 21
4.4 Noisy Oracle Best Model Similarity Comparison 21
4.5 Noisy Oracle Estimated Hyperparamter Comparison 22
4.6 Label Efficiency of Model Picker vs. Baselines 25
4.7 Method comparison results part 1. The graphs on the left show the success probability of the methods and the graphs on the right present the 90 percentile return accuracy for different datasets and budget sizes. 25
4.8 Distribution Shift Example: PACS 29
4.9 Distribution Shift Example: ImageNet V2 29
4.10 Model Picker Information Gain Experiment 30

A.1 PACS Experiment Training Distributions 42
A.2 Average Posterior Development Example, ϵ: 0.3 44
A.3 Average Posterior Development Example, ϵ: 0.4 44
A.4 Average Posterior Development Example, ϵ: 0.49 44

Nomenclature

Abbreviations

Abbreviation	Definition
AL	Active Learning
AWS	Amazon Web Services
ML	Machine Learning
SSL	Semi-Supervised Learning

1 Introduction

In the realm of Machine Learning (ML), the trend towards automation of model development and deployment offers new ways of autonomously maintaining high quality models in production. The continuous retraining of models over time results in an abundance of slightly different quality models for the same problem setting. With growing interest in ML, particularly due to the recent progress in Large-Language-Models, the number of possible applications and, consequently, the market for ML models has increased drastically (AI Index Steering Committee, 2024). Additionally, countless researchers and hobby programmers train and evaluate their models daily to solve both well-established and novel problems. If these models are published, they often become part of publicly available model repositories such as Huggingface[1]. Major companies, such as Amazon with their Amazon Web Services (AWS) program, maintain their own model registry, containing thousands of pre-trained models for divers applications. Furthermore, popular programming frameworks like PyTorch and Tensorflow offer collections of pre-trained models.

The primary reason for a decline in performance of a model in production, subsequently leading to model retraining, is a distribution shift in the data (Sculley et al., 2015). With the growing number of quality pretrained models available, it becomes increasingly plausible that for a given problem setting, there exists a pre-trained model that performs sufficiently well for the distribution of the problematic data. This realisation forms the foundation for the idea of efficient model selection of pre-trained classifiers. To identify the optimal model given unlabeled data, it is essential to gather evidence. This evidence can be obtained through various methods, such as unsupervised learning. However, the most effective and accurate method is to query for labels, making model selection a supervised learning problem with limited labels. Despite significant progress in ML, labeling data remains a persistent challenge. While collecting raw data through sensors or other processes can be straightforward, labeling is not. The labeling process is mostly carried out by humans in regulated processes, making it a very resource-intensive task. Particularly for more complex tasks like image segmentation or object detection the labeling process becomes extremely costly, time-consuming and prone to errors. Using the price list from AWS Sagemaker[2] and the assumption to employ five workers in AWS Mechanical Turk for more reliable predictions, the cost for 100.000 labeled instances can reach from approximately 12.000€ for image classification tasks up to approximately 426.000€ for image segmentation tasks. To put this into perspective: popular image classification datasets frequently used in ML include CIFAR-10 with 60.000 samples and 10 classes, and ImageNet with 1.200.000 samples and 1000 classes. The high number of classes significantly introduces more labeling noise, requiring more workers to achieve reliable results. Estimating the costs for labeling ImageNet with ten Mechnical Turk workers results in 190.000€. The retraining iterations in production contribute to the already substantial initial labeling costs required to develop a model, resulting in ongoing periodic expenses to maintain high performance. This motivates minimizing the number of labels required as evidence for model selection.

In the literature, ”model selection” is commonly associated with selecting a learning strategy and its

[1] https://huggingface.co/

[2] https://aws.amazon.com/de/sagemaker/groundtruth/pricing/

hyperparameters expected to best fit the data at hand. However, in the scope of this thesis, model selection distinctly refers to selecting a pre-trained classifiers from a pool of candidates based on unlabeled data. Consequently, the hyperparameters and the learning strategies are predefined and don't need to optimized. Although the objective of selecting the best model for the data remains the same, the setting in this thesis differs significantly. The research field of Active Learning (AL) aims to intelligently query unlabeled samples to optimize training performance, providing valuable insights and baselines for this research. Model selection, as described in this setting, is a common problem in practice and can be applied to various applications to determine the best performing model to employ without training.

The model selection process in machine learning involves several key components: a specific problem domain (e.g. classification of spam emails), a set of pre-trained models designed to address this particular problem, and a pool of unlabeled data whose distribution is unknown. The challenge is to identify the model that performs best on the unlabeled data with as few labels as possible. This thesis introduces the first method in the pool-based model selection setting, named Model Picker, that aims to solve this problem in a robust and label-efficient manner. The algorithm is tested extensively and evaluated against fundamental sampling methods in various classification settings, reaching from extreme noise to class imbalance. Model Picker is based on theoretical guarantees and undergoes an extensive empirical evaluation through a series of experiments. Beyond proposing and evaluating a competitive solution, this thesis aims to establish a framework for evaluating and testing future model selection strategies.

The thesis begins with background section, distinguishing this model selection setting from the ones common in the literature and providing insight about related research. This is followed by the methodology chapter, which offers in-depth descriptions of the model selection problem, the Model Picker algorithm, adapted AL baselines, and the metrics used for evaluation. Next, the experiments and their results are presented, followed by a discussion. The thesis concludes with a summary of the results, explaining limitations and future work, and putting the findings into context.

2 Background

2.1. Model Selection

Model selection in machine learning is a pivotal decision-making process that focuses on determining the learning algorithm and parameter setting that minimize an objective function on given data (Zhou, 2021). According to Ding et al., 2018, model selection can serve two primary purposes: inference, which aims to identify the best model for understanding the underlying data generation process, and prediction, which focuses on minimizing the generalization error for future predictions.

When selecting the best learning algorithm and architecture, various approaches exist. Expert knowledge is crucial and often incorporated to limit the search space by defining a selection of algorithms or algorithm components. In the early stages of model selection, methods such as step wise regression were introduced to estimate the best regression model for a given problem. Step wise regression includes techniques like forward selection, backward elimination, and bidirectional elimination, which sequentially add or remove predictors based on statistical criteria, aiming to balance model complexity and performance. Today, alongside manual definition of models, advanced automated model architecture frameworks are used, such as Reinforcement Learning (RL) introduced by Baker et al., 2017 or Evolutionary Algorithms (EA) proposed by Real et al., 2017. Advances in Neural Architecture Search (NAS), that aims to automate the design of neural networks, resulted in solutions like Differential Architecture Search (DARTS) introduced by Liu et al., 2019, which efficiently navigates the search space of neural network components with gradient-based optimization.

To improve the models and prevent fitting, various regularization methods have been proposed. Lasso (Tibshirani, 1996) and Ridge (Hilt and Seegrist, 1977) regression, commonly referred to as L1 and L2 regularization, add bias to the model to decrease variance. A combination of both, proposed by Zou and Hastie, 2005, balances variable selection and regularization, making it effective for handling correlated predictors. Other common regularization techniques include dropout for neural networks (Srivastava et al., 2014), early stopping during training (L. Li et al., 2018), and data augmentation (Simard et al., 2003).

Hyperparameter optimization also plays a major role in a model's performance. Like the selection of learning algorithms, hyperparameter optimization heavily depends on initial expert knowledge to reduce the otherwise potentially infinite search space. Simple brute force solutions like grid search and random search are still widely used depending on the requirements for the solution (Bergstra and Bengio, 2012). Variations of grid search, introduced by and Chih-Jen Lin Chih-Wei Hsu, 2008 and Hesterman et al., 2010, continuously refine their search space according to the results of explored regions. L. Li et al., 2018 introduced Hyperband, which applies dynamic resource allocation for promising configurations combined with random search and successive halving to improve computational cost over traditional random and grid search. Bayesian optimization, discussed by Snoek et al., 2012, utilizes surrogate models (e.g. Gaussian processes) to provide probabilistic estimates for the objective function and its uncertainty. Snoek et al., 2015 improved Bayesian optimization by addressing the challenge of scaling Gaussian processes to larger datasets and higher-dimensional hyperparameter settings. Falkner et al., 2018 introduce BOHB, a robust hyperparameter optimization framework at scale, that combines Bayesian optimization with Hyperband. Additionally many approaches with different surrogate models (e.g. random forest, tree-structured Parzen estimator) and methods aimed to increase performance of Bayesian optimization (Klein et al., 2017) have been proposed. Gradient-based optimization techniques, as shown by Chandra et al., 2022, seek the optimal parameter configuration by utilizing gradient information over the hyperparameter space

Reliable performance estimates are crucial for deciding which learning algorithm to used and how to parameterize it. Information criteria like the Akaike Information Criterion (AIC) introduced by Akaike, 1973, the Bayesian Information Criterion (BIC) proposed by Schwarz, 1978, and the Deviance Information Criterion (DIC) are widely used metrics to estimate the goodness of fit of the model to the data while penalizing high model complexity. Mallows, 1973 created Mallow‘s Cp for linear regression, assessing the goodness of fit by comparing the sum of squared errors to the number of parameters. Assessing the complexity of the models is crucial as overly complex models can lead to overfitting, resulting in poor generalization performance. Cross-validation (CV) is a predictive paradigm that estimates a model’s generalizability by dividing the data into training and validation sets multiple times, ensuring a reliable generalization error estimate given a limited amount of data. The most popular forms are holdout, k-fold cross-validation and leave-one-out cross-validation (LOOCV), with k-fold cross-validation being more common in practice due to LOOCS’s computational cost (Piironen and Vehtari, 2017). Another generalization error estimation technique related to CV is bootstrapping, proposed by Johnson, 2001, which is particularly useful for small datasets or when there is no effective way to split the data. To enhance computational efficiency in model selection, Domhan et al., 2015 extrapolate the learning curve during training to early detect noncompetitive models and then skip their full training. Other methods utilize prior knowledge from previous solved tasks to accelerate learning. Feurer et al., 2018 uses information about the hyperparameters of previous runs to warm-start the new hyperparameter search. The research area of transfer learning aims to utilize weights from already trained models of related tasks to decrease the work in the target task (Pan and Yang, 2010).

The objective of traditional model selection and model selection of pre-trained classifiers align well, but the settings differ significantly. The research setting at hand does not include any information about the models and has no labeled data available, rendering most of the mentioned methods and paradigms inapplicable. Additionally, there is no need to further optimize the candidate models as they have already been fine-tuned. Currently, there is very little research focusing on this scenario.

Kossen et al., 2021 present the ”Active Testing” framework, which aims to collect an unbiased and efficient test set for evaluating trained classifiers. Their focus lies on evaluating a single model as unbiased and efficient as possible by sampling a high quality test set with help of a surrogate model. The work of Sawade et al., 2012 aims to compare a pair of pre-trained classifiers by calculating a new sampling distribution based on relative model risks. Labeling instances according to this distribution should yield an efficient comparison between the models. An early study of ”Active Model Selection” was proposed by Madani et al., 2004, which focuses on a model-centric approach (Hawkins et al., 1987). This means that instead of selecting the best data point to label next, their focus lies on which model to evaluate next.

2.2. Label-Efficient Learning

Since this thesis focuses on the supervised learning setting with limited labels, it is essential to explore research areas addressing this challenge. Numerous learning paradigms aim to enhance label efficiency in supervised learning settings. From here onwards, the term ”model selection” refers to the selection of pre-trained classifiers instead of the traditional model selection process.

2.2.1. Active Learning

Active Learning (AL), a form of experimental design, employs acquisition functions to select a minimal set of unlabeled data for training. The primary objective is to efficiently train a model with as few labels as possible while maintaining or enhancing its performance (Herzberg et al., 1972). This methodology aligns well with model selection despite the differing objectives of training and evaluation. Given that AL is a well-studied field with numerous proposed solutions, the acquisition functions utilized in AL present valuable candidates for model selection baselines.

Uncertainty sampling is a commonly used metric for informativeness of a data point in AL. As the name suggests, the informativeness of a data point is measured by how uncertain a model is regarding its prediction. The more uncertain a model is, the more the data point is expected to benefit the learning process Settles, 2010.

One way to calculate uncertainty in a multi class setting is the **least confidence** measure (Settles, 2010). In the classification setting used in this research, the least confidence metric equals the probability of the most confident prediction on a data point. Therefore, by sampling data points with the lowest least confidence score, the algorithm ensures sampling data points with the highest uncertainty.

Confidence margin defines uncertainty as the difference between the first and second highest output probability Joshi et al., 2009. Minimizing the confidence margin ensures sampling data points where the model is most uncertain about its most confident class compared to the second choice. This measure takes more of the distribution into account compared to the least confidence score.

Entropy is frequently used measure from information theory to calculate uncertainty. It measures the amount of information needed to describe a distribution (Shannon, 1948). The entropy measure is applied to the class prediction of the model to calculate an uncertainty value. Sampling the data points that maximize entropy ensures sampling samples with the highest uncertainty.

Query by Committee (QBC) is a disagreement sampling method (Freund et al., 1997; Seung et al., 1992). As the name suggests, it uses a committee of models trained on the same dataset but differing in some way (e.g. hyperparameter, model architecture, training time). The committee members vote on all data points, with each vote representing the class with the highest prediction probability. The data points where the models disagree the most become the most valuable. Therefore the highest form of disagreement occurs when all classes on a data point receive the same number of votes. Various ways to convert disagreement into a value of informativeness exist, including vote margin and vote entropy, which are very similar to confidence margin and entropy used in uncertainty, but applied to the committee votes. Another method is Kullback-Leibler Divergence (KL-Divergence) (McCallum and Nigam, 1998), used to calculate the difference between two distributions. QBC is particularly suitable for the Model Selection problem because the set of candidate models to evaluate naturally represents the committee.

Incorporating Density with uncertainty and disagreement-based sampling methods yields significant benefits in AL (Ren et al., 2022; Sener and Savarese, 2018). Density sampling methods work in the input space, ensuring selected samples are representative of the distribution. Since uncertainty and disagreement sampling operate only in prediction space, they are prone to sampling uninformative outliers. Density methods counteract this behavior. Solutions that solely depend on density, like Sener and Savarese, 2018 and Chitta et al., 2022, treat AL as a Core-set selection problem, aiming to identify a subset of samples that represent the underlying distribution of the whole dataset. Many other methods combine density and uncertainty measures to benefit from both (Ash et al., 2020; Kee et al., 2018; Margatina et al., 2021). For example, the authors of Kee et al., 2018 proposed to enhance QBC with weighted terms for density and diversity using different similarity measures (e.g. k-NN, cosine similarity), while the authors of Margatina et al., 2021 define valuable samples as being similar in input space (representative) and different in prediction space (uncertain). The informativeness of data points there is based on their k-NN neighborhood (density in input space) and their predictive KL-Divergence (uncertainty in prediction space).

Other AL methods encompass ideas like expected model change, variance reduction, and estimated error reduction (Settles and Craven, 2008). Methods calculating the expected model change aim to maximize the expected effect a data point would have on the model if its label was known. An example is expected gradient length (EGL) proposed by Settles et al., 2007. The algorithm EPIG proposed by Smith et al., 2023 selects data points that maximize the information gain about the model parameters. Established active learning methods like EPIG Smith et al., 2023, BADGE Ash et al., 2020 and BAIT Ash et al., 2021 sample data points based on metrics derived from model intrinsic parameters such as last layer gradients. N. and A., 2001 calculate informativeness by estimating a data points expected error after training.

Beyond AL, semi-supervised learning, programmatic weak supervision and ensemble learning offer different ways to improve training under limited labeled data availability.

2.2.2. Semi-Supervised Learning

Semi-supervised learning can be divided into three types: self-training, co-training and boosting (van Engelen and Hoos, 2020). Self-training uses a model trained on a small labeled dataset to label the unlabeled data. The model iteratively adds its most confident predictions on the unlabeled data to the training set to refine itself over time (Yarowsky, 1995). Co-training employs multiple classifiers trained on different features of the data, using their predictions to label data points independently, allowing the classifiers to be iteratively retrained (van Engelen and Hoos, 2020). Boosting, similarly to co-training, leverages a set of classifiers trained to correct the previous classifiers' mistakes, creating a stronger composite classifier. Although Boosting can be used in SSL, it is generally categorized as ensemble learning method.

2.2.3. Ensemble Learning

Ensemble learning utilizes a set of weaker models to form a better one. A famous boosting method is AdaBoost (Freund et al., 1999), which stands for adaptive boosting. In addition to boosting, there is bagging (e.g. Random-Forest proposed by Ho, 1995), which trains classifiers on different parts of the dataset and aggregates their predictions to create a more reliable one, and stacking, which trains a meta-model that learns to aggregate the weak learners predictions (Pavlyshenko, 2018).

2.2.4. Weak Supervision

Weak supervision is a machine learning technique that generates training data by leveraging multiple sources of noisy, imprecise, or incomplete labels instead of relying solely on manually labeled datasets. These sources can include heuristics, domain-specific rules, crowdsourced annotations, or external databases. The key idea is to use a variety of labeling functions, which are often imperfect, and then aggregate their outputs to produce a more reliable labeled dataset. This method allows for the efficient creation of large-scale training data, reducing the need for extensive manual labeling while still enabling effective model training (Ratner et al., 2020).

2.3. Main Takeaway

The literature review aimed to provide an overview of traditional model selection and existing methods in the realm of model selection and label efficiency. Additionally, the goal was to emphasize that traditional model selection methods do not apply in this setting. Given the limited research in this specific area, the scope was widened to related research fields to investigate fundamental techniques and asses their suitability for model selection. AL emerged as most promising source for baseline candidates due to its query strategies, which align well with the idea of model selection. In particular, the query strategies in the category of QBC present a good fit for this research. However, AL methods that work directly with a model, like the ones mentioned under the "other AL methods" category, are challenging to adapt because the model selection problem is defined in a model-agnostic way.

The primary insight gained from the literature review for the following research is the lack of focus on model selection of pretrained classifiers, the major difference to the traditional model selection setting, and the promising applicability of AL methods through acquisition functions.

3 Methodology

Let $\mathcal{M}$ represent a set of k pretrained ML models $\{m_1, \dots, m_k\}$. Based on an unlabeled data pool of size n, denoted $\mathcal{U} = \{x_1, ..., x_n\}$, and a maximum budget of b, model selection aims to identify the optimal model $m^* \in \mathcal{M}$ with the objective of minimizing the number of labels required. The best model is the classifier with the highest utility (e.g. classification accuracy) on the data if all the labels were available. To achieve efficient model selection, methods query the labels of a highly informative subset $\mathcal{L} = \{(x_1, y_1), ..., (x_\mathrm{b}, y_\mathrm{b})\}$, that is expected to yield the most information about identifying the best model.

Let π_b denote a sampling strategy of size b that defines the order of examples to query, then $\pi_{opt[b]}$ represents the optimal policy of size b that determines the optimal order of sampling $\mathcal{L}$. It is known that selecting a set of most informative samples of restricted cardinality is generally NP-Hard (Ko et al., 1995) and therefore calculating the optimal policy is practically infeasible. Consequently, all methods employed in this thesis aim to estimate this optimal policy in various ways.

3.1. Model Picker

In order to solve the model selection problem a probabilistic model inspired by Chen et al., 2015 is defined (see figure 3.1).

Here, M is the latent variable representing the decision about the best model, and all Y_i stand for a number observable random variables. These random variables represent the unknown labels and are used to learn the true distribution of M. An assumption made is that all Y_i are independent from each other given M but dependent on M and ϵ. The hyperparameter ϵ models the prediction noise of the classifiers.

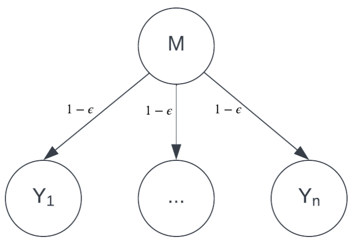


Figure 3.1: Probabilistic model to quantify informativeness of data points. M being the latent variable representing the best model and ϵ being the models prediction noise on the unknown label Y_i.

Let the optimal policy $\pi_{opt[b]}$ be defined as the policy that achieves the maximal expected mutual information about M:

$$\pi_{opt[b]} = \text{argmax}_{\pi \in \Pi_b} \mathbb{I}(\mathrm{M}; \pi) \tag{3.1}$$

where Π_b is the set of policies of size b. The solution proposed in this thesis aims to estimate a policy $\pi_{MP[b]}$ so that:

$$\mathbb{I}(\mathrm{M}; \pi_{MP[b]}) \approx \mathbb{I}(\mathrm{M}; \pi_{opt[b]}) - \delta \tag{3.2}$$

by employing a greedy adaptive sampling algorithm with near optimal guarantees. Here, δ is a small number depending on the noise, bounding the mutual information gained by the greedy policy relative to the optimal one. Chen et al., 2015 show that next to near optimal guarantees for (adaptive-)

submodularity and sequential maximization of mutual information, sequential maximization of mutual information under persistent noise also has near optimal guarantees, given some noise constraint and a sufficient budget.

Each unlabeled data point in $\mathcal{U}$ is represented as a pair $\mathcal{U} = \{(x_1, Y_1), ..., (x_n, Y_n)\}$, where x_i is a data point and Y_i is a random variable associated with the unknown label of that data point. The latent variable $\mathrm{M} \in \{m_1, ..., m_k\}$ incorporates the decision over the best model, given the evidence $\mathcal{L}_t$ at time t. The conditional independence of Y_i implies that querying the label of a specific Y_i does not directly effect the other tests but influences them through providing evidence. This evidence effects M, which in turn affects the other Y_i. This relation is clarified when discussing how the value of each data point is calculated in the following section. The error rate ϵ represents the probability that a model incorrectly predicts the true label of a data point. Conversely, $1 - \epsilon$ represents the probability of predicting the correct true label. A higher ϵ indicates increased noise and reduced trust in the model's predictions

Given $\mathcal{L}_t$, at each sampling step the Model Picker algorithm greedily selects the data point from $\mathcal{U}_t$ that provides maximum information about M. Upon querying the label, the data point label pair (x_i, y_i) is added to the evidence. The process continues until the budget b is exhausted.

Inspired by Chen et al., 2015, the Model Picker algorithm calculates the information value of unlabeled data points by computing Shannon's mutual information between M and each Y_i given $\mathcal{L}_t$. Therefor the optimal data point $x^*(t)$ at time t is given by:

$$x^*(t) = \text{argmax}_{(x_i, Y_i) \in \mathcal{U}_t} \mathbb{I}(\mathrm{M}; Y_i | \, \mathcal{L}_t) \tag{3.3}$$

where x_i and Y_i denote a concrete data point unknown label pair and $t = \{1, \ldots, b\}$. The maximization in equation 3.3 identifies the data point whose label, when observed, is expected to provide the most information about M, given the current evidence $\mathcal{L}_t$. Shannon's mutual information can be rewritten as the difference in entropy:

$$\mathbb{I}(\mathrm{M}; Y_i | \, \mathcal{L}_t) = \mathbb{H}(\mathrm{M} \, | \, \mathcal{L}_t) - \mathbb{H}(\mathrm{M} \, | Y_i, \mathcal{L}_t) \tag{3.4}$$

where $\mathbb{H}(\mathrm{M} \, | \, \mathcal{L}_t)$ is the uncertainty around M given the evidence until t, and $\mathbb{H}(\mathrm{M} \, | Y_i, \mathcal{L}_t)$ is the uncertainty around M when adding an arbitrary Y_i to the evidence.

Since $\mathbb{H}(\mathrm{M} \, | \, \mathcal{L}_t)$ is constant over all x_i and equation 3.4 is a maximization problem, that part of the term can be eliminated. Additionally, by removing the minus sign, the maximization turns into a minimization resulting in:

$$x^*(t) = \text{argmin}_{(x_i, Y_i) \in \mathcal{U}_t} \mathbb{H}(\mathrm{M} \, | Y_i, \mathcal{L}_t) \tag{3.5}$$

Here, instead of maximizing the difference in uncertainty, the data point is selected that, through its estimated information value, yields the lowest uncertainty around M.

Since the informativeness of a random variable can not be calculated directly, it is estimated with the expected entropy of M over all possible classes. Therefore, a hypothetical class is assigned to the unknown labels Y_i and, with that, the rest uncertainty is calculated. Repeating this for all classes and averaging the outcomes results in an estimate of the data point's rest uncertainty. Formally, this leads to:

$$x^*(t) = \text{argmin}_{(x_i, Y_i) \in \mathcal{U}_t} \mathbb{E}_{c \in \mathcal{Y} \sim P_c} \mathbb{H}(\mathrm{M} \, | Y_i = c; \mathcal{L}_t) \tag{3.6}$$

where $\mathcal{Y}$ is the outcome set of all Y_i. Since the input data distribution is unknown, P_c represents the assumption that the class priors are distributed uniform. Formulating this with entropy (Shannon, 1948) results in:

$$x^*(t) = \text{argmin}_{(x_i, Y_i) \in \mathcal{U}_t} \sum_{c \in \mathcal{Y}} \frac{1}{C} \sum_{m \in \mathcal{M}} p(\mathrm{M} = m | Y_i = c, \mathcal{L}_t) \log(p(\mathrm{M} = m | Y_i = c, \mathcal{L}_t)) \tag{3.7}$$

where C is the number of classes in $\mathcal{Y}$ and a uniform prior is assumed for M. The conditional probability $p(\mathrm{M} = m|Y_i = c; \mathcal{L}(t))$ represents the confidence in one model given the evidence and the hypothetically assigned class c. This probability is unknown, but by incorporating the probabilistic model (3.1) this probability can be expressed as:

$$p(\mathrm{M} = m|Y_i = c, \mathcal{L}_t) = (\frac{1-\epsilon}{\epsilon})^{N_{m,c}} = \gamma^{N_{m,c}} \tag{3.8}$$

where γ is a reward constant directly depending on the hyper parameter ϵ. Let $N_m \in \{0, \ldots, t\}$ denote the number of correct predictions made by the model on the evidence up to time t. Then $N_{m,c}$ represents the number of correct predictions up to time t plus the prediction on the hypothetical evidence c. Consequently, $N_{m,c}$ can take any value in $\{0, \ldots, t+1\}$. The reward constant γ is derived from the probabilistic model:

$$p(\mathrm{M} = m|Y_i = c, \mathcal{L}_t) = (1-\epsilon)^{N_{m,c}} \sum_{i=1}^{C-1} (\frac{\epsilon}{C-1})^{t+1-N_{m,c}} \tag{3.9}$$

$$p(\mathrm{M} = m|Y_i = c, \mathcal{L}_t) = (1-\epsilon)^{N_{m,c}} \epsilon^{t+1-N_{m,c}} \tag{3.10}$$

$$p(\mathrm{M} = m|Y_i = c, \mathcal{L}_t) = (\frac{1-\epsilon}{\epsilon})^{N_{m,c}} \epsilon^{t+1} \tag{3.11}$$

The confidence in one classifier m is represented by the probability of $t+1$ independent events, based on the probabilistic model where the unknown labels Y_i are considered independent of each other given M.

An additional event $(t+1)$ is considered, due to the evidence up to time t and the hypothetical evidence $Y_i = c$. These events can be divided into $N_{m,c}$ correct predictions, each with a probability of $1-\epsilon$, and $t+1-N_{m,c}$ false predictions, each with a probability of $\epsilon/(C-1)$ per wrong class. Summing up the uniform probabilities of false predictions for each of the $C-1$ wrong classes in equation 3.9 results in ϵ, leading to equation 3.10. Next, separating $1/\epsilon^{N_{m,c}}$ from $\epsilon^{t+1-N_{m,c}}$ and incorporating it into the brackets results in equation 3.11. Finally, the confidences over all models are combined into a distribution and therefor normalized. Since ϵ^{t+1} is a constant, it can be removed from the term as it has no effect on the result, leading to equation 3.8.

Algorithm 1 Model Picker Algorithm Outline

```
Require: Models: $\mathcal{M} \neq \{\}$, Unlabeled pool: $\mathcal{U} \neq \{\}$,
    Budget: $0 \ll b \leq n$, Classes: $\mathcal{Y} \neq \{\}$, Epsilon: $0 \ll \epsilon < 0.5$
 1: $\mathrm{M} \leftarrow$ uniform distribution of length $|\mathcal{M}|$
 2: $\mathcal{L}_t \leftarrow$ empty evidence $\{\}$
 3: $\mathcal{U}_t \leftarrow \mathcal{U}$
 4: $\gamma \leftarrow \frac{1-\epsilon}{\epsilon}$
 5: for $t \in \{0...b-1\}$ do
 6:     $u_t \leftarrow \{\}$
 7:     for $x \in \mathcal{U}_t$ do
 8:         $u_x \leftarrow 0$
 9:         for $c \in \mathcal{Y}$ do
10:             $\mathrm{M}_{c,x} \leftarrow$ rewardModels$(\mathcal{M}, \mathrm{M}, \gamma, x, c)$
11:             $u_{x,c} \leftarrow$ entropy$(\mathrm{M}_{c,x})$
12:             $u_x = u_x + u_{x,c}$
13:         end for
14:         $u_t \leftarrow u_t \cup \frac{u_x}{|\mathcal{Y}|}$
15:     end for
16:     Select $x^*$ from $u_t$ and query label $y$
17:     $\mathrm{M} \leftarrow$ rewardModels$(\mathcal{M}, \mathrm{M}, \gamma, x^*, y)$
18:     Remove $x^*$ from $\mathcal{U}_t$ and add $(x^*, y)$ to $\mathcal{L}_t$
19: end for
```

The algorithm is designed to gradually build up confidence towards model that performed well based on the accumulated evidence, consequently leading to $\gamma > 1$ and $0 \ll \epsilon < 0.5$. Equation 3.8 illustrates ϵ influences the algorithms behavior. When ϵ is higher (closer to 0.5), γ approaches 1, resulting in minimal reward and little change in M (indicating low trust in prediction and high noise). Conversely, when ϵ is lower, γ increases, meaning that rewarding a model significantly impacts M (indicating high trust in prediction and low noise).

Instead of recalculating 3.8 over a model's full prediction history every time, a more efficient iterative approach is beneficial:

$$p(\mathrm{M}=m|Y_i=c,\mathcal{L}_t) \propto p(\mathrm{M}=m|\,\mathcal{L}_t)\gamma^{P_{m,c}} \tag{3.12}$$

where $P_{m,c} \in \{0,1\}$ solely indicates whether the model m predicted class c on the data point (=1) or not (=0). Consequently, a model is rewarded with γ when predicting c on x_i. How this is applied in Model Picker is shown in algorithm 1. It outlines the process in the most intuitive form, showing the difference between calculating the data point values and calculating the posterior after each sampling step.

The two outer for-loops represent a sampling step and iterating over each data point. As described in equation 3.6, for each data point the algorithm iterates over all possible classes to sum up the conditional rest uncertainties. Within the inner loop the *rewardModels* function is crucial. It utilizes the iterative reward approach to return a hypothetical posterior $\mathrm{M}_{t,c,x}$ by applying equation 3.12 to all models and normalizing the result. Afterward, the entropy of $\mathrm{M}_{t,c,x}$ is calculated, and the result is added to the current value of the data point (u_x). This process is repeated for all classes. Finally, u_x is normalized and stored in u_t, which is the list containing the data point values for this particular sampling step.

After completing the calculation of all data point values, the data point with the least rest uncertainty is selected, and its true label y is queried. The true label is then permanently available and used to calculate the real posterior by again applying the *rewardModel* function. The difference here lies in the usage of the true label y instead of a hypothetical class c. To update the evidence for the next sampling step the new posterior (M_{t+1}) is stored in M_t. Finally, before moving on to the next sampling step, the selected sample is removed from $\mathcal{U}_t$ and added to the $\mathcal{L}_t$.

3.2. Baselines

To benchmark the performance of Model Picker, a variety of baselines was selected from the field of Active Learning were selected. Commonly used uncertainty measures such as entropy, vote margin and least confidence could be easily adapted for this setting. "Active Comparison on prediction Models" is one of the few baselines found that was designed specifically for model comparison of pretrained classifiers. Other sampling methods like QDD and GALAXY focus on different problems in Active Learning, such as sampling higher-representative examples or mitigating class imbalance. In the following sections, each method is explained in detail.

3.2.1. Query by Committee

Query by Committee (QBC) introduced by Seung et al., 1992, is a suitable baseline for the model selection problem. In AL, the committee has to be created from different candidate models, where in this setting, the committee is available through the set of models to compare. QBC operates on the principle of disagreement to determine which data point to sample. The models' class predictions are used to create a vote distribution on each data point. Various uncertainty measures can then be applied to this vote distribution to measure the value of each data point. The more uncertain the committee is, the more value a data point has (Kee et al., 2018). Notably, QBC is a non adaptive sampling algorithm, meaning that the ordering of the samples is defined upfront, and does not change with the collection of evidence. For ease of explanation, QBC is defined as querying up to b data points that have the highest uncertainty according to $u(x)$, where the $u(x)$ is defined by the different uncertainty measures. The vote distribution $p(y|x)$ over all $y \in \mathcal{Y}$ is given by:

$$p(y|x) = \frac{v(y,x)}{k} \tag{3.13}$$

where $v(y,x)$ is the number of votes a class received on data point x and k is the number of models.

Vote margin is one of the two uncertainty measures most commonly used with QBC:

$$u(x) = p(y_2|x) - p(y^*|x) \tag{3.14}$$

where $p(y^*|x)$ represents the probability of the most voted class, and $p(y_2|x)$ represents the probability of the runner up. Since QBC is framed as a maximization problem in this context, the ordering of the two terms needs to be reversed. Consequently, a smaller the gap between the two, indicates a higher uncertainty, with 0 being minimal uncertainty and 1 being maximal uncertainty.

Vote entropy, another frequently used uncertainty measure with QBC, is defined as:

$$u(x) = -\sum_{y \in \mathcal{Y}} p(y|x) \log(p(y|x)) \tag{3.15}$$

This measure calculates the entropy of the vote distribution over all possible outcomes y in the outcome set $\mathcal{Y}$ given x. Entropy ranges from 0 to 1, where 1 indicates maximum uncertainty.

Least confidence is a common uncertainty measure often used independently of QBC. Typically, the least confidence measure is applied on the output distribution of individual models to assess its uncertainty regarding the prediction. Here it is adapted to the context by applying it to the vote distribution:

$$u(x) = 1 - \max_{y \in \mathcal{Y}}(p(y|x)) \tag{3.16}$$

As shown in equation 3.16, least confidence is defined as the complement of the highest class probability according to the votes. Again, the inversion is needed to transform the output to a maximization, resulting in less confidence being the higher uncertainty score.

3.2.2. Active Comparison of Prediction Models

”Active Comparison of Prediction Models” introduced by Sawade et al., 2012 is a non adaptive sampling method designed for model comparison of pretrained classifiers. By estimating empirical risks of each model, the goal is to create an instrumental sampling distribution q^*. Originally created to compare two models, the authors proposed an extension for multi-model comparison by constructing a mixture of pairwise-optimal sampling distributions. The resulting distribution assigns higher probabilities to data points that contain more information about distinguishing the best models based on disagreement and estimated risks.

To create q^*, the empirical risk estimates for each model are calculated. For a model f_j is given by:

$$\hat{R}[f_j] = \frac{1}{C} \sum_{i=1}^{n} \sum_{y \in \mathcal{Y}} p(y, x_i) l(f_j(x_i), y) \tag{3.17}$$

where n is the number of samples in the unlabeled pool, C the number of classes in $\mathcal{Y}$, and $p(y, x) = p(y|x)p(x)$. Assuming a uniform distribution for $p(x)$, results in $p(x) = 1/n$. The authors originally used the softmax output for each class to estimate $p(y|x)$. Here, $p(y|x)$ is estimated with the vote distribution of all models, as discussed in the QBC section. The loss function l is defined as zero-one loss:

$$l(f_j(x), y) = \begin{cases} 1 & \text{if} f_j(x) \neq y \\ 0 & \text{if} f_j(x) = y \end{cases} \tag{3.18}$$

Having calculated $\hat{R}[f_j]$ for all models, the delta matrix $\hat{\Delta}$ can be computed by pairwise comparing all models' risks $\hat{R}[f_j] - \hat{R}[f_k]$. The matrix $\hat{\Delta}$ represents the relative risks, and gives an indication on which model is preferable. Using $\hat{\Delta}$, $q^*_{j,k}$ is constructed, which is the pairwise-optimal sampling distribution for the models f_j and f_k:

$$q^*_{j,k}(x) = \sum_{y \in \mathcal{Y}} \sqrt{(l(f_j(x), y) - l(f_k(x), y) - \hat{\Delta}_{j,k})^2 p(y|x)} \tag{3.19}$$

where $\hat{\Delta}_{j,k}$ is the value in the delta matrix comparing the two models. Sampling distributions are created for every pairwise model comparison in $\hat{\Delta}$, resulting in $m(m-1)$ pairwise-optimal sampling distributions. Finally, the optimal sampling distribution q^* is then calculated by:

$$q^*(x) = \frac{1}{m(m-1)} \sum_{j \neq k} q^*_{j,k}(x) \tag{3.20}$$

Using q^*, up to b samples without replacement can be sampled and used for the model selection process.

3.2.3. GALAXY

GALAXY is an AL algorithm specifically designed to address the issue of class imbalance that often occurs in real-world datasets. A lot of acquisition functions proposed in AL do not consider the class distribution, potentially leading to biased or less informative samples Zhang et al., 2022. The GALAXY algorithm overcomes this challenge by implementing a graph-based method that selects balanced and uncertain examples for labeling.

For each class, GALAXY constructs a one-versus-all graph where nodes represent data points. The data points (nodes) are sorted by their class confidence margin and connected to their direct neighbor, resulting in a linear graph. In this graph, the most uncertain data points are placed at the start, while the least uncertain data points are at the end. The class confidence margin $\delta^y(x)$ for an instance x in the class graph y is given by:

$$\delta^y(x) = p(y|x) - \max_{y' \in \mathcal{Y}}(p(y'|x)) \tag{3.21}$$

It defines the difference in probability between a class y and the most likely class $\max_{y' \in \mathcal{Y}}(p(y'|x_i))$ for a data point. Instead of using the softmax output of a single model, again the vote distribution estimate is used. A significant difference between the AL setting of GALAXY and the implementation used here is that no new graphs are constructed at each time step t. In AL, this is necessary because the model's predictions change after the training step. However, in this setting $p(y|x)$ remains fixed. Additionally, instead of sampling a batch of examples, one example at a time is sampled, to stay consistent with the other methods.

GALAXY requires a small initially labeled set of examples which are chosen randomly from the unlabeled set. All queried examples are marked on each graph, indicating whether they belong to the graph's class or not. Using the Dijkstra algorithm, the shortest path between a labeled node belonging to the class of a graph, and a node belonging to another class is found. This shortest path represents the region of uncertainty. The midpoint of this path is queried. All graphs get updated with the freshly queried sample in the same way as with the initial set. This approach continuously reduces the region of uncertainty, ensuring that samples closer to the true decision boundary are queried.

Different ways exist to decide on which class graph to sample from at each t. The authors propose sampling from the graph that provides the overall shortest shortest path, thus sampling from the region closest to the decision boundary. However, the authors of ”Label Bench”Zhang et al., 2024, a paper comparing different label efficient learning techniques, implemented GALAXY with random selection of graphs for each sampling step. Implementing random selection comes with a speed advantage because it avoids calculating the shortest path for all graphs at each time step. The experiments showed no significant impact on performance, and therefore, random graph selection is used in this implementation.

3.2.4. QDD (Query by Committee with Density and Diversity

The QDD method is an AL algorithm designed to enhance the efficiency and effectiveness of model training Kee et al., 2018. This approach incorporates three critical aspects of AL: uncertainty sampling via QBC, diversity sampling to prevent redundant samples, and density sampling to ensure representativeness of the selected examples. The method was selected to investigate the potential impact of input data on the model selection process. Unlike other methods that purely rely on the prediction space and ignore the input features, QDD takes both into account. Although the uncertainty and density measures for all data points are calculated upfront, QDD can be classified as an adaptive sampling algorithm because the diversity measure changes with the selected instances.

The best data point is determined by maximizing a utility function. According to QDD, the most valuable data point is defined as:

$$x^*(t) = \text{argmax}_{x \in \mathcal{U}_t}(u(x)) \tag{3.22}$$

The utility function $u(x)$ assigns a value to the data points and is defined as:

$$u(x) = (1 - \gamma - \beta) f(x) + \gamma d(x) + \beta h(x) \tag{3.23}$$

Here, $f(x)$ represents the QBC uncertainty measure, $d(x)$ the density measure, and $h(x)$ the diversity measure for a data point. The variables γ and β are weights that assign importance to the different terms and therefore $\gamma + \beta \leq 1$.

The uncertainty measure used in QDD is entropy over models softmax prediction. Again, the softmax output is substituted with the vote distribution resulting in the entropy measure previously explained in QBC (see equation 3.15). For the density and diversity measures, a similarity metric is needed. Cosine similarity is used to compare the features embedded in the last dense layer of a ResNet-18 pretrained on ImageNet. A matrix is created that pairwise compares all data points, resulting in scores from -1 to 1, with 1 indicating equality, 0 indicating no correlation and -1 indicating maximal inequality.

The density measure $d(x)$ is calculated at the start of the algorithm using k-nearest neighbors (k-NN):

$$d(x) = \frac{1}{k} \sum_{x' \in \mathcal{N}_{k,x}} \text{CosSim}(x, x') \tag{3.24}$$

where $\mathcal{N}_{k,x}$ is the set of k examples in the neighborhood of x. The diversity measure $h(x)$ is given as the minimum distance to all already queried examples:

$$h(x) = \min_{x' \in \mathcal{L}_t} (\text{CosSim}(x, x')) \tag{3.25}$$

where $\mathcal{L}_t$ is the set of already labeled instances at time t. After each sampling step the diversity has to be updated because it may have changed due to the new evidence.

4 Experiments

4.1. Experimental Protocol

The objective of this evaluation is to determine the number samples that need to be labeled to reliably identify the best model across a variety of datasets, as well as to obtain an accurate estimate of generalization, strengths, and weaknesses of the Model Picker algorithm. To achieve a reliable estimate, a number of realisations are uniformly sampled from the entire dataset. Unless stated otherwise, 100 realisations, each containing 1000 instances, are sampled across all experiments. This realisation size ensures a good balance between sampling error and representativeness of the underlying data. Each realisation is treated as the complete data pool, with its corresponding best ground truth model, thus running an experiment on a realisation represents running an experiment on the entire dataset. The results of all realisation experiments are aggregated to the full result. The budget during evaluation is always set to the realisation pool size, allowing for investigation of the algorithm's behavior throughout the entire process. The results give an indication of the budget required in the real setting for a reliable model selection process.

4.2. Experimental Setup

The developed software framework can be accessed via the GitHub repository[1]. It is implemented in Python, with necessary libraries listed in dedicated requirements files for easy reproducibility. The model selection experiments primarily utilize Numpy, Scipy and Matplotlib, while tasks related to machine learning models, such as generating predictions for the datasets, are handled using PyTorch. Inspired by Zhang et al., 2024, the framework is designed for usability, making the addition of new methods straight forward. Ideally, this framework will serve as a foundation for evaluation of future model selection methods.

Depending on the experiments and their anticipated runtime, they were executed either locally or on the TU-Delft computation cluster[2]. An Apptainer image was defined for easy deployment on the cluster, including the minimal requirements to run all experiments.

4.2.1. Datasets and Model Collection

The datasets selected for this research have diverse characteristics and aim to provide insights into various aspects of the model selection process. Table 4.1 lists all datasets along with additional information about their characteristics. The goal was to cover a wide variety of scenarios, from different accuracy ranges and class imbalance to extreme noisy problems.

For clarification, some scores presented in the overview are only available in the test setting because here all labels of the test set are available. In production, it is assumed that only little meta data about the problem is known, and potentially unreliable model accuracy evaluated during training. The

[1]https://github.com/janneskasper/efficientModelSelection
[2]https://daic.tudelft.nl/

Datasets	Num. Models	Classes	Acc. Range	Fleiss' Kappa	Disagreement Ratio
Domain Drift	9	6	0.25 - 0.61	0.1168	3600 / 3600 (100.0%)
CIFAR-10 Low	80	10	0.41 - 0.69	0.4231	9926 / 10000 (99.26%)
CIFAR-10 High	80	10	0.55 - 0.92	0.6028	9587 / 10000 (95.87%)
Imagenet	102	1000	0.37 - 0.84	0.6209	42876 / 50000 (85.752%)
Imagenet V2 M-F	114	1000	0.43 - 0.81	0.6712	7780 / 10000 (77.8%)
Imagenet V2 T-0.7	114	1000	0.51 - 0.86	0.7288	7162 / 10000 (71.61%)
Imagenet PyTorch	114	1000	0.55 - 0.87	0.7468	33578 / 50000 (67.15%)
PACS	30	7	0.73 - 0.94	0.7631	5260 / 9991 (52.64%)
Imagenet V2 T-I	114	1000	0.58 - 0.89	0.7678	6588 / 10000 (65.88%)
Emotion Detection	8	4	0.88 - 0.92	0.7807	902 / 5509 (16.37%)

Table 4.1: The table shows meta data and different characteristics for all used datasets ordered by their disagreement (Fleiss' Kappa) score $\kappa \in \{-1, \ldots, 1\}$ with -1 indicating perfect disagreement and 1 indicating perfect agreement. Accuracy range describes the classification accuracy from the worst to the best model. Disagreement ratio indicates the ratio of informative samples where at least one model disagrees with the rest.

accuracy range in table 4.1 represents the worst and best model test accuracy on the test set. The *Fleiss' Kappa* coefficient (κ) (Cohen, 1960) is used to indicate how much the models agree with each other based on their predictions. This score is used throughout the thesis as indication of noisiness in the data and models. The Fleiss' Kappa score has an advantage over a pure percentage agreement measure because it accounts for the agreement that could occur by chance, thus providing a more accurate assessment.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{4.1}$$

Where p_o is the relative observed agreement among models, and p_e is the hypothetical probability of chance agreement. The κ coefficient ranges from -1, indicating perfect disagreement, through 0, representing agreement by chance, to 1, indicating perfect agreement.

The *disagreement ratio* shows how many data points are valuable for the model selection process. A data point is valuable when there is at least one model disagreeing with the rest. A higher kappa score suggests a greater likelihood of data points where where all models agree, resulting in fewer valuable samples. The number of models greatly influences the number of valuable samples as well. The fewer models are used, the more likely samples are produced where all models agree. The model accuracy distribution (appendix figure A.1,A.2) provides insight into how many models fall within defined accuracy ranges. A low number of high quality models can indicate an easier model selection problem due to the lack of valuable candidates.

Creating a dataset in this context refers to using an existing dataset to create a set of class predictions. This involves training different models on the datasets and using them to make predictions on the test sets. Different models can vary in architecture, hyper-parameter, or the portions of training data used. The result is a matrix of $n \times k$ class predictions, where n is the number of instances in the test set and k being the number of models. A corresponding oracle with the correct labels is created as well. The test sets of the used datasets act as the production data with the unknown distribution. Hence, from now on when talking about datasets, it implies the combination of the prediction matrix and the oracle.

The dataset collection includes two versions from the popular CIFAR-10 dataset (Krizhevsky, 2009). One version (CIFAR-10 Low) is noisier, reflected in the lower accuracies of the models. This version includes less accurate models (max. $\approx 70\%$), making it challenging for the algorithm to rely on predictions. The second version (CIFAR-10 High) includes predictions from more accurate models, allowing the algorithm to trust their predictions more.

Next to CIFAR-10, different ImageNet dataset versions were created. ImageNet it is the most competitive image classification benchmark with numerous pre-trained models available. The challenge with ImageNet is that the test set is not publicly available. To address this, the labeled validation set of 50.000 images was used to create the predictions. For the ImageNet-Pytorch version, all pre-trained models available on PyTorch (PyTorch, 2024) were used to predict on the ImageNet validation set.

Additionally, the ImageNet-V2 test sets proposed by Recht et al., 2019 were used. These datasets are standalone test sets for model evaluation purposes, with 10.000 samples each. The goal is to identify whether the progress on the ImageNet challenge was partly due to models generalizing to ImageNet itself. The authors tried to mimic the original dataset creation closely and still detected a significant performance gap in the models on the created test sets, concluding that this gap is mostly caused by a distribution shift in the data and the selection of more difficult samples. Therefore they proposed three different test sets of varying difficulty according to their selection frequency metric: ImageNet-V2 Top-Images (T-I), ImageNet-V2 Threshold-0.7 (T-0.7) and ImageNet-V2 Matched-Frequency (M-F). The difference in model performance over the different test sets mentioned in the paper was successfully reproduced in our dataset creation. The decreasing accuracy range shown in table 4.1 reflects similar behavior, even different models were used. For all three ImageNet V2 versions all available pre-trained models from PyTorch were used.

To research a very noisy setting, the domain drift dataset was created. The underlying data distribution is a combination of different distributions. Each model was trained on one of nine distributions, and the test set is from a different tenth distribution, Resulting in a very noisy dataset with low model performances on the test set. However, this does not necessarily mean that the models performed bad in their own distribution.

The Emotion Detection (Pillai, 2020) dataset was used to create a dataset with extensive class imbalance. Until now, all datasets have had uniform class distribution. This dataset is extremely imbalanced towards one class (≈85%), with the other classes sharing the remaining samples, each hovering around 5%.

Lastly, the PACS dataset (D. Li et al., 2017) was used. It is widely used for testing domain adaptation of models. The PACS dataset includes the a classification problem with seven classes in four different domains: photo, art, cartoon and sketch. This dataset is ideal for demonstrating how distribution shift or, in extreme cases, domain drift influences model accuracy and model ranking. Therefore, the dataset was used to create another test set for model evaluation as well as an example that supports the motivation for model selection.

4.2.2. Evaluation Metrics

To assess the effectiveness of the different methods used, the evaluation focuses on three core metrics: success probability, 90 percentile return model accuracy, and initial labeling cost. The three metrics provide a comprehensive measure for the model selection process, balancing the objectives of accuracy and label efficiency.

To achieve a robust evaluation of the methods and to compute the success probability, a number of realisations is generated from the data. A realisation is a uniformly sampled subset from the entire dataset, large enough to resemble the characteristics of the dataset but small enough to induce some sampling error. By executing the experiments on all realisations, a reliable performance estimate can be calculated.

Success Probability quantifies the consistency with which a method identifies the correct best model across multiple realisations at a certain point in time. It is calculated as the proportion of realisations where the method selected the optimal model out of all trials:

$$\textit{Success Probability at t} = \frac{\textit{Number of correct Best Models returned at t}}{\textit{Number Realisations}} \tag{4.2}$$

A high success probability (greater than 80%) at time t indicates that the method is expected to reliably identify the best model on the underlying dataset.

90 Percentile Return Model Accuracy is a robustness measure and refers to the difference between the ground truth best model's accuracy and the currently selected best model's accuracy at time t in the worst 90% of realisations. Since the true oracle is known in the test setting, the accuracy of each model for each realisation can be calculated beforehand using classification accuracy:

$$\textit{Model Accuracy} = \frac{\textit{Number of correct Predictions}}{\textit{Total number of Predictions}} \tag{4.3}$$

To calculate the return model accuracy difference in a realisation the overall best models accuracy is subtracted from the accuracy of the selected best model at time t. The 90 percentile return model accuracy is the mean of the worst 90% of return model accuracy differences over all realisations. This measure is a good indication of robustness, as it shows how close the selected model at time t is to the optimal model in terms of overall performance in non optimal cases. A return model accuracy close to zero implies that the model chosen at time t has a performance nearly equivalent to the overall model in this realisation. This knowledge is valuable, since the objective is to balance the labeling cost with the quality of the returned model.

Labeling Cost measures the efficiency of the whole model selection process in terms of the number of labels required to achieve a level of performance. This is particularly significant when discussing the initial labeling cost of Model Picker used for estimating the optimal hyperparameter ϵ. Therefore, the total labeling cost consists of two parts:

$$\textit{Labeling Cost} = \textit{Hyperparameter Optimization Labeling Cost} + \textit{Model Selection Labeling Cost} \quad (4.4)$$

For now the hyperparameter labeling cost for all other baselines is set to 0. Since all methods are designed to minimize the process labeling cost, it is crucial that the initial labeling cost for Model Picker is as low as possible, while maintaining a good hyper-parameter configuration. It is important to note these metrics are used to evaluate the model selection methods performance, not the models themselves. The metric to evaluate the model ranking given the evidence at time t can be chosen separately. In this implementation the classification accuracy is used as defined in equation 4.3. Other metrics, like the F1-score, can be applied as well.

4.2.3. Implementation Details

The efficient implementation of Model Picker slightly differs from the algorithm shown in algorithm 1. Instead of looping over all data points, Numpy's matrix operations are employed to accelerate the process. Interestingly, it is more efficient to keep the iteration over the classes compared to adding another dimension to the matrix operations. Consequently, the algorithm's performance scales with number of classes in the dataset. Experiments with datasets containing a lot of classes, such as ImageNet with 1000 classes, tend to take longer. This is particularly important considering the fact that grid search, for finding the optimal epsilon, runs the model selection process 100 times for each epsilon. This results in 1000-2000 model selection processes per experiment (given 10-20 different epsilons). In addition to using Numpy for speed optimization, realisations are executed in parallel because the results are independent and are only get combined after all have finished.

4.3. Hyperparameter Optimization

4.3.1. Hyperparameter Optimization by True Oracle

Before comparing Model Picker to the other baselines, it is crucial to properly parameterize the algorithm by finding the optimal value of ϵ for each dataset. The hyperparameter ϵ models the prediction noise of the classifiers, with higher ϵ values (closer to 0.5) expected for noisy datasets such as Domain Drift and CIFAR-10 Low, and a lower ϵ expected for datasets with less noise, such as PACS and Emotion Detection. The following results present the hyperparameter optimization based on the ground truth oracle. This way of optimization is used to investigate the behavior of ϵ across various settings and serves as a reference for subsequent methods to efficiently learn ϵ without relaying on the true oracle, since this is not available in the real setting.

To identify the optimal ϵ grid search is used. An initial broad range of $\epsilon = \{0.2, \ldots, 0.5\}$ with step size of 0.025 was defined to get an estimate of the region of interest. For the ImageNet datasets the range was reduced to $\epsilon = \{0.3, \ldots, 0.5\}$ with step size of 0.02 due to high sensitivity to low epsilons. The results of this wide search can be found in the appendix A.4. From this, the most effective areas were identified, and a more detailed grid search was conducted. For all datasets the region of interest was within $\epsilon = \{0.35, 0.41, \ldots, 0.49\}$ with a step size of 0.01. For the Domain Drift dataset a wider range of $\epsilon = \{0.2, 0.21, \ldots, 0.49\}$ with step size 0.01 was used because of multiple interesting areas.

Instead of presenting the results of every ϵ configuration, a representative subset was selected for better visualisation. The results are shown in figure 4.2 and 4.1. The selected configurations include low and high ϵ values as well as a number of well-performing configurations in the middle, one of them being the optimal ϵ. The different graphs on the left side in figure 4.2 present the success probability in the y-axis and the sampling time in the x-axis, while the graphs on the right side display the 90 percentile return model accuracy in the y-axis. Despite every experiment being executed with a pool size and budget of 1000 samples, all graphs focus on the most interesting areas of the results for better visualisation. Each data sequence in a graph is a differently parameterized Model Picker, running on the same realisations.

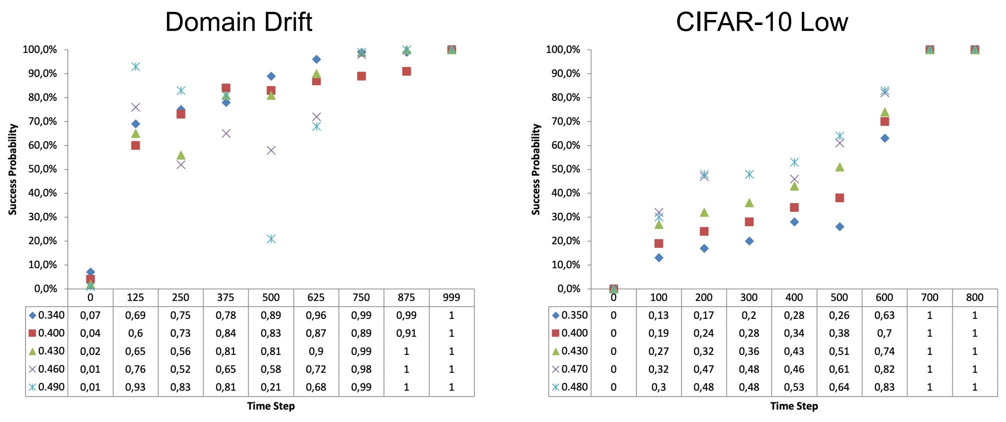
The grid search results reveal different trends. Higher ϵ tend to show a promising start but converge slowly towards the end. In contrast, lower ϵ need more initialization time but then their confidence in the models increases rapidly. Another notable observation is that higher ϵ can get stuck in a ”local optimum” (e.g. Domain Drift, ImageNet V2 T-I, ImageNet), where the success probability drops after an initial peak. For some datasets, little difference is observed for different ϵ configurations in the good range (e.g. ImageNet V2 PyTorch, Emotion Detection), which is also confirmed by the return model accuracy graphs. Generally, the results indicate a close correlation between return model accuracy and the success probability.

The results also provide insight into the consequences of selected an ϵ outside the optimal range. High ϵ, such as 0.48, lead to extreme loss in convergence speed, as indicated by multiple graphs in figure 4.2 e.g. the light blue graph in ImageNet V2 M-F. ImageNet V2 T-0.7 illustrated well the effect of a low ϵ, resulting in early cut-off and poor overall performance (dark blue data with $\epsilon = 0.35$). The data sequence not reaching the 100% success probability for a long time indicates that Model Picker gets stuck with a wrongly identified best model in one or more realisations.

Dataset	Epsilon
Domain Drift	0.34
CIFAR-10 Low	0.47
CIFAR-10 High	0.46
ImageNet	0.4
ImageNet V2 M-F	0.44
ImageNet V2 T-0.7	0.42
ImageNet PyTorch	0.43
PACS	0.42
ImageNet V2 T-I	0.4
Emotion Detection	0.42

Table 4.2: The table shows the optimal ϵ configurations for each dataset based on grid search with ground truth oracle available.

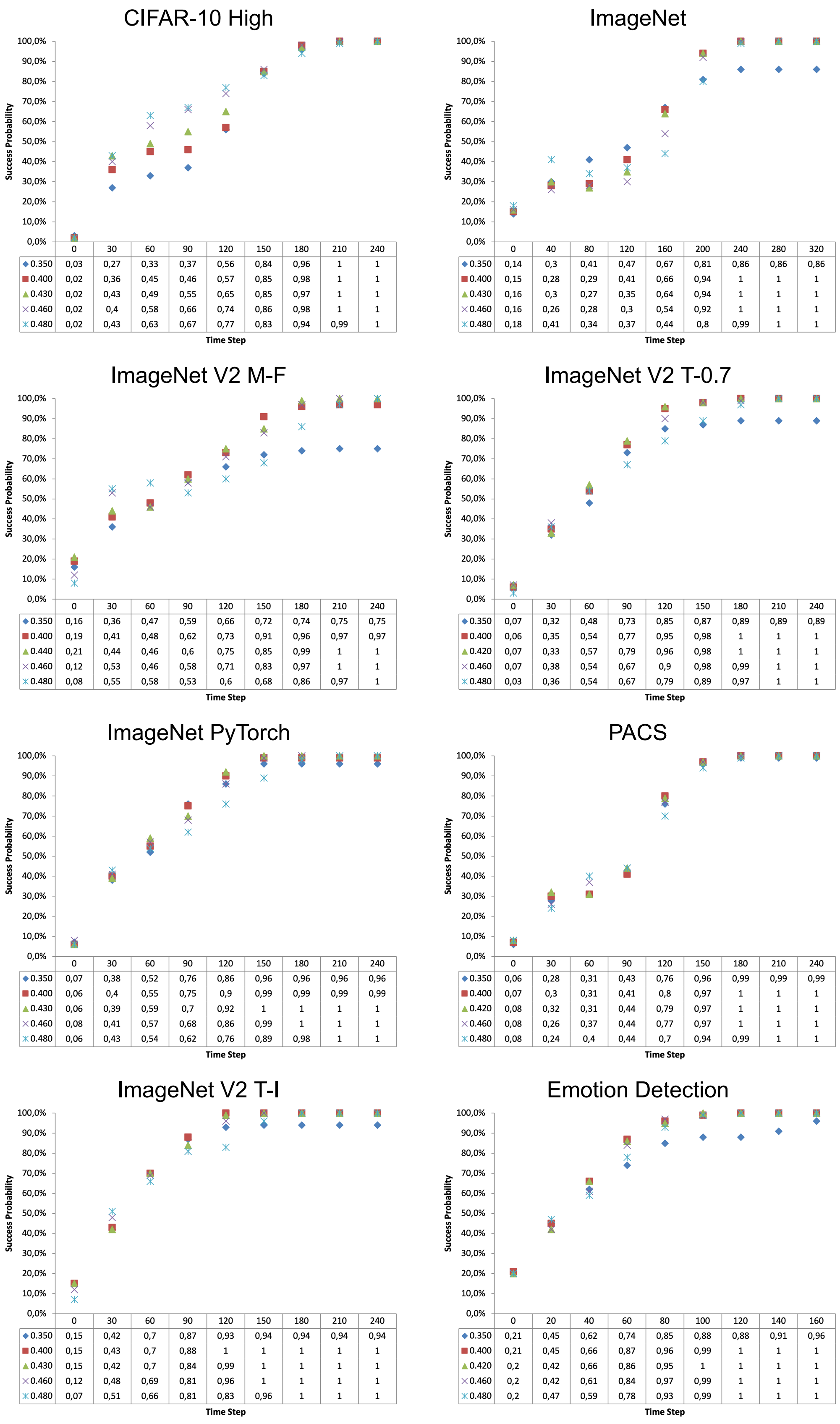
Currently, the best ϵ is selected manually. While the initial phase (<60% success probability) is not very significant, the primary heuristic is the performance between 60-100% success probability. The ϵ that reaches this area the fastest and performs best within it is deemed the optimal one. Table 4.2 lists the best ϵ for each dataset according to this selection procedure. Apart from Domain Drift and ImageNet, a slight correlation between data noisiness (sorted from top to bottom) and ϵ value is observable.



Time Step	0	125	250	375	500	625	750	875	999
◆ 0.340	0,07	0,69	0,75	0,78	0,89	0,96	0,99	0,99	1
■ 0.400	0,04	0,6	0,73	0,84	0,83	0,87	0,89	0,91	1
▲ 0.430	0,02	0,65	0,56	0,81	0,81	0,9	0,99	1	1
× 0.460	0,01	0,76	0,52	0,65	0,58	0,72	0,98	1	1
✳ 0.490	0,01	0,93	0,83	0,81	0,21	0,68	0,99	1	1

Time Step	0	100	200	300	400	500	600	700	800
◆ 0.350	0	0,13	0,17	0,2	0,28	0,26	0,63	1	1
■ 0.400	0	0,19	0,24	0,28	0,34	0,38	0,7	1	1
▲ 0.430	0	0,27	0,32	0,36	0,43	0,51	0,74	1	1
× 0.470	0	0,32	0,47	0,48	0,46	0,61	0,82	1	1
✳ 0.480	0	0,3	0,48	0,48	0,53	0,64	0,83	1	1

Figure 4.1: Grid search result with ground truth oracle available, part 1: The graphs show the success probability of grid search experiments for Model Picker hyperparameter optimization on all datasets with known labels, given different budget sizes.



CIFAR-10 High

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,03	0,27	0,33	0,37	0,56	0,84	0,96	1	1
■ 0.400	0,02	0,36	0,45	0,46	0,57	0,85	0,98	1	1
▲ 0.430	0,02	0,43	0,49	0,55	0,65	0,85	0,97	1	1
× 0.460	0,02	0,4	0,58	0,66	0,74	0,86	0,98	1	1
✳ 0.480	0,02	0,43	0,63	0,67	0,77	0,83	0,94	0,99	1

ImageNet

	0	40	80	120	160	200	240	280	320
◆ 0.350	0,14	0,3	0,41	0,47	0,67	0,81	0,86	0,86	0,86
■ 0.400	0,15	0,28	0,29	0,41	0,66	0,94	1	1	1
▲ 0.430	0,16	0,3	0,27	0,35	0,64	0,94	1	1	1
× 0.460	0,16	0,26	0,28	0,3	0,54	0,92	1	1	1
✳ 0.480	0,18	0,41	0,34	0,37	0,44	0,8	0,99	1	1

ImageNet V2 M-F

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,16	0,36	0,47	0,59	0,66	0,72	0,74	0,75	0,75
■ 0.400	0,19	0,41	0,48	0,62	0,73	0,91	0,96	0,97	0,97
▲ 0.440	0,21	0,44	0,46	0,6	0,75	0,85	0,99	1	1
× 0.460	0,12	0,53	0,46	0,58	0,71	0,83	0,97	1	1
✳ 0.480	0,08	0,55	0,58	0,53	0,6	0,68	0,86	0,97	1

ImageNet V2 T-0.7

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,07	0,32	0,48	0,73	0,85	0,87	0,89	0,89	0,89
■ 0.400	0,06	0,35	0,54	0,77	0,95	0,98	1	1	1
▲ 0.420	0,07	0,33	0,57	0,79	0,96	0,98	1	1	1
× 0.460	0,07	0,38	0,54	0,67	0,9	0,98	0,99	1	1
✳ 0.480	0,03	0,36	0,54	0,67	0,79	0,89	0,97	1	1

ImageNet PyTorch

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,07	0,38	0,52	0,76	0,86	0,96	0,96	0,96	0,96
■ 0.400	0,06	0,4	0,55	0,75	0,9	0,99	0,99	0,99	0,99
▲ 0.430	0,06	0,39	0,59	0,7	0,92	1	1	1	1
× 0.460	0,08	0,41	0,57	0,68	0,86	0,99	1	1	1
✳ 0.480	0,06	0,43	0,54	0,62	0,76	0,89	0,98	1	1

PACS

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,06	0,28	0,31	0,43	0,76	0,96	0,99	0,99	0,99
■ 0.400	0,07	0,3	0,31	0,41	0,8	0,97	1	1	1
▲ 0.420	0,08	0,32	0,31	0,44	0,79	0,97	1	1	1
× 0.460	0,08	0,26	0,37	0,44	0,77	0,97	1	1	1
✳ 0.480	0,08	0,24	0,4	0,44	0,7	0,94	0,99	1	1

ImageNet V2 T-I

	0	30	60	90	120	150	180	210	240
◆ 0.350	0,15	0,42	0,7	0,87	0,93	0,94	0,94	0,94	0,94
■ 0.400	0,15	0,43	0,7	0,88	1	1	1	1	1
▲ 0.430	0,15	0,42	0,7	0,84	0,99	1	1	1	1
× 0.460	0,12	0,48	0,69	0,81	0,96	1	1	1	1
✳ 0.480	0,07	0,51	0,66	0,81	0,83	0,96	1	1	1

Emotion Detection

	0	20	40	60	80	100	120	140	160
◆ 0.350	0,21	0,45	0,62	0,74	0,85	0,88	0,88	0,91	0,96
■ 0.400	0,21	0,45	0,66	0,87	0,96	0,99	1	1	1
▲ 0.420	0,2	0,42	0,66	0,86	0,95	1	1	1	1
× 0.460	0,2	0,42	0,61	0,84	0,97	0,99	1	1	1
✳ 0.480	0,2	0,47	0,59	0,78	0,93	0,99	1	1	1

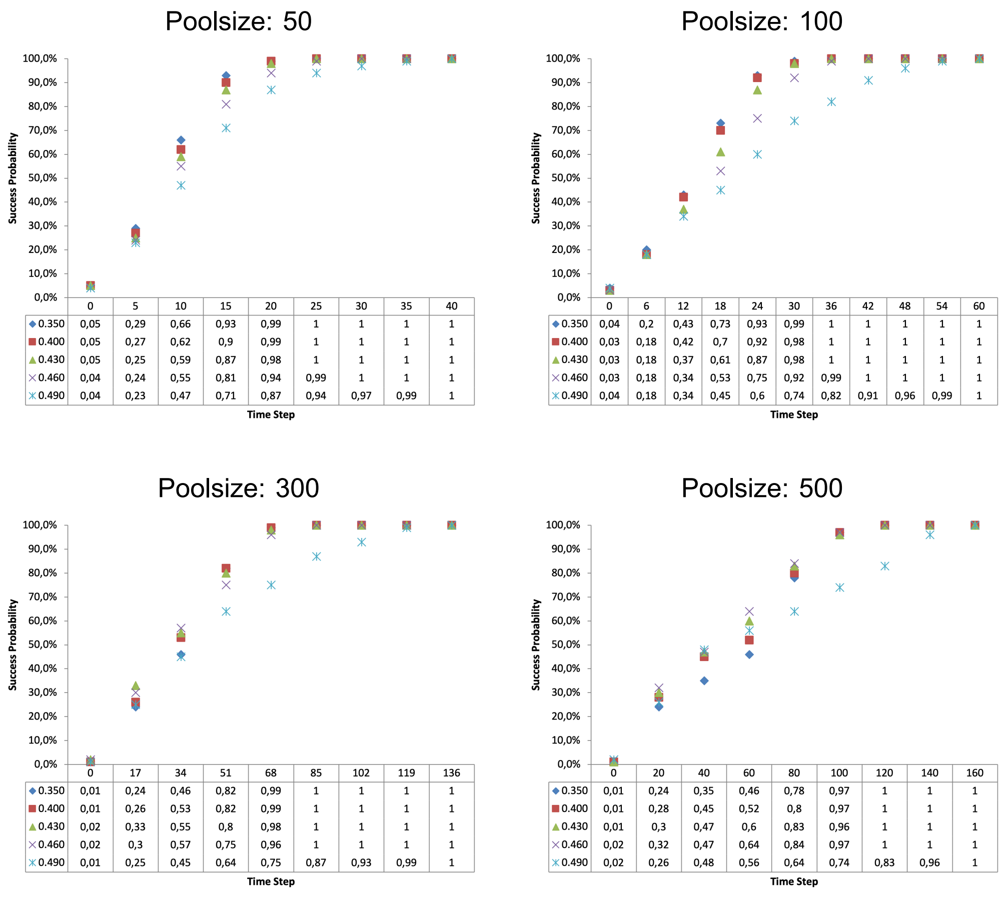
Figure 4.2: Grid search result with ground truth oracle available, part 2: The graphs show the success probability of grid search experiments for Model Picker hyperparameter optimization on all datasets with known labels, given different budget sizes.

4.3.2. Hyperparameter Optimization by Subset Sampling

In the previous chapter, ϵ was tuned by using the ground truth oracle. Since this is not available in the real-world settings, it is essential to find a label-efficient method to learn ϵ without knowing the oracle. The labeling cost arising from hyperparameter optimization is referred to as the initial labeling cost. This section focuses on finding a way to minimize this initial labeling cost. The baseline to beat is selecting a fixed ϵ for all settings. From the results in the previous chapter, an ϵ of 0.43 performs well across all datasets.

The initial idea was to query the labels of a small, uniformly sampled subset of examples and use them to estimate an ϵ to use for the entire dataset. This method was tested empirically by running grid search on different pool sizes, reaching from 50 samples up to 500, to determine how many examples are needed to identify dataset's characteristics. The CIFAR-10 High dataset was chosen for this test due to its significant differences between ϵ configurations, which allows for easier identification of when a good estimate is reached.

Figure 4.3 shows four different experiments with pool sizes of 50, 100, 300 and 500. Over the course of these experiments, the dataset characteristics develop gradually. In the smaller pool sizes, lower ϵ tend to outperform higher ones. Specifically, in the experiments with pool sizes of 50 and 100, the ϵ configurations are ordered by their value, with lower ϵ performing best. At a pool size of 300, differences in the middle part of the algorithm start to become apparent, influencing the ranking of the ϵ configurations to better reflect the expected behavior.



Poolsize: 50

Time Step	0	5	10	15	20	25	30	35	40
◆ 0.350	0,05	0,29	0,66	0,93	0,99	1	1	1	1
■ 0.400	0,05	0,27	0,62	0,9	0,99	1	1	1	1
▲ 0.430	0,05	0,25	0,59	0,87	0,98	1	1	1	1
× 0.460	0,04	0,24	0,55	0,81	0,94	0,99	1	1	1
✳ 0.490	0,04	0,23	0,47	0,71	0,87	0,94	0,97	0,99	1

Poolsize: 100

Time Step	0	6	12	18	24	30	36	42	48	54	60
◆ 0.350	0,04	0,2	0,43	0,73	0,93	0,99	1	1	1	1	1
■ 0.400	0,03	0,18	0,42	0,7	0,92	0,98	1	1	1	1	1
▲ 0.430	0,03	0,18	0,37	0,61	0,87	0,98	1	1	1	1	1
× 0.460	0,03	0,18	0,34	0,53	0,75	0,92	0,99	1	1	1	1
✳ 0.490	0,04	0,18	0,34	0,45	0,6	0,74	0,82	0,91	0,96	0,99	1

Poolsize: 300

Time Step	0	17	34	51	68	85	102	119	136
◆ 0.350	0,01	0,24	0,46	0,82	0,99	1	1	1	1
■ 0.400	0,01	0,26	0,53	0,82	0,99	1	1	1	1
▲ 0.430	0,02	0,33	0,55	0,8	0,98	1	1	1	1
× 0.460	0,02	0,3	0,57	0,75	0,96	1	1	1	1
✳ 0.490	0,01	0,25	0,45	0,64	0,75	0,87	0,93	0,99	1

Poolsize: 500

Time Step	0	20	40	60	80	100	120	140	160
◆ 0.350	0,01	0,24	0,35	0,46	0,78	0,97	1	1	1
■ 0.400	0,01	0,28	0,45	0,52	0,8	0,97	1	1	1
▲ 0.430	0,01	0,3	0,47	0,6	0,83	0,96	1	1	1
× 0.460	0,02	0,32	0,47	0,64	0,84	0,97	1	1	1
✳ 0.490	0,02	0,26	0,48	0,56	0,64	0,74	0,83	0,96	1

Figure 4.3: The figures shows the success probability of different grid search results for Model Picker hyperparameter optimization with subset sampling on the CIFAR-10 High dataset.

4.3.3. Hyperparameter Optimization by Noisy Oracle Estimation

An alternative approach is to use model predictions to generate a noisy oracle. With that, an ϵ over all samples can be estimated. Two different methods to generate the labels were implemented: majority voting (selecting the class with the highest votes) and sampling a label from the vote distribution created by the models. This vote distribution is estimated the same way as for QBC (see equation 3.13).

To further enhance the noisy oracle, 50 samples are labeled to improve the estimation. These samples are selected through three different heuristics: uniformly random sampling, weighted sampling based on uncertainty measured with entropy over the vote distribution, and directly sampling the 50 most uncertain instances (ordering).

Table 4.3 compares the similarity between the noisy oracles and ground truth oracle. Table 4.4 illustrates how often the best model according to the noisy oracle matches the best model according to the ground truth oracle. Interestingly, the methods of estimating the noisy oracle through majority voting and vote distributions have different effects on the best model similarity compared to oracle similarity. The oracle similarity consistently shows lower percentages for vote distribution. However, for model similarity, the vote distribution reduces the probability of identifying the best model in datasets such as CIFAR-10 Low and PACS, whereas for the other datasets, the probability increases.

	N-V	N-VD	E-V-R	E-VD-R	E-V-W	E-VD-W	E-V-O	E-VD-O
Domain Drift	56.9%	*39.9%*	**59.0%**	43.0%	59.2%	43.1%	58.7%	43.9%
CIFAR-10 Low	77.0%	*57.5%*	78.2%	59.7%	78.6%	60.3%	**80.0%**	61.5%
CIFAR-10 High	90.0%	*74.7%*	90.5%	76.0%	90.9%	76.7%	**92.3%**	78.2%
ImageNet	80.3%	*67.4%*	81.4%	69.3%	82.0%	70.1%	**83.3%**	71.6%
ImageNet V2 M-F	74.5%	*67.0%*	76.1%	69.1%	76.9%	70.1%	**78.2%**	71.4%
ImageNet V2 T-0.7	82.1%	*75.1%*	83.3%	76.8%	84.0%	77.7%	**84.9%**	79.0%
ImageNet PyTorch	83.2%	*77.1%*	84.4%	78.7%	85.3%	79.8%	**86.8%**	81.4%
PACS	93.4%	*87.2%*	94.0%	88.4%	94.4%	88.9%	**95.5%**	90.2%
ImageNet V2 T-I	85.7%	*79.8%*	86.7%	81.3%	87.5%	82.2%	**88.4%**	83.5%
Emotion Detection	91.9%	*90.8%*	93.5%	92.7%	93.6%	92.9%	**94.3%**	93.4%

Table 4.3: The table presents the similarity between the ground truth oracle and noisy oracles created with different noisy oracle estimation methods for 1000 realisations with 1000 samples each. Bold indicates maximum similarity and italic minimal. N: Noisy Oracle, E: Enhanced Noisy Oracle, V: Majority Voting, VD: Vote Distribution, R: Random, W: Weighted, O: Ordered

	N-V	N-VD	E-V-R	E-VD-R	E-V-W	E-VD-W	E-V-O	E-VD-O
Domain Drift	0.0%	0.2%	0.0%	0.9%	0.0%	**0.9%**	0.0%	0.0%
CIFAR-10 Low	14.6%	7.9%	16.8%	9.3%	17.7%	10.9%	**20.5%**	14.1%
CIFAR-10 High	2.5%	6.4%	5.2%	8.1%	6.4%	14.7%	17.9%	**26.8%**
ImageNet	5.6%	4.7%	8.1%	9.5%	10.3%	10.9%	15.4%	**19.7%**
ImageNet V2 M-F	0.0%	0.0%	0.0%	0.1%	0.0%	0.1%	0.1%	**2.1%**
ImageNet V2 T-0.7	0.0%	0.0%	0.0%	0.1%	0.0%	0.9%	0.2%	**4.8%**
ImageNet PyTorch	0.0%	0.2%	0.0%	0.4%	0.2%	1.5%	1.2%	**5.4%**
PACS	24.4%	18.8%	31.2%	26.0%	36.3%	29.1%	**47.5%**	43.1%
ImageNet V2 T-I	0.0%	0.0%	0.0%	0.6%	0.2%	1.7%	2.4%	**10.0%**
Emotion Detection	6.7%	16.6%	14.4%	31.2%	17.9%	33.3%	36.1%	**44.6%**

Table 4.4: The table shows the percentage of times the best model on the ground truth oracle matches the best model on the different generated noisy oracles, measured across 1000 realisations with 1000 samples each. Bold indicates maximum similarity. N: Noisy Oracle, E: Enhanced Noisy Oracle, V: Majority Voting, VD: Vote Distribution, R: Random, W: Weighted, O: Ordered

Figure A.3 in the appendix presents the grid search results based on the different noisy labeling methods on the example of CIFAR-10 High. Both label generation methods (majority voting and vote distribution) are used alone and paired with each enhancement method, resulting in a total of eight different results. These are compared to the grid search results with the ground truth oracle. It is evident that using the noisy oracle, which induces labeling noise into the data, leads to a slightly slower convergence time ($\approx$220 samples vs. $\approx$300 samples). This observation aligns with the previous noted trend that noisier datasets tend to converge slower. Another notable observation is that despite the varying results for best model similarity in table 4.4, majority voting estimates the underlying dataset characteristics more accurately. All graphs on the left side of figure A.3 are visually closer to the ground truth figure at the top. Conversely, the vote distribution method estimates the cut off caused by the lowest ϵ configurations (0.35, yellow), which is a crucial indication of too low ϵ configurations. None of the majority vote approaches succeeded in estimating this behavior.

To investigate how well the noisy oracle estimation techniques generalize to other datasets, the most promising methods were selected. These include the pure noisy oracle based on majority voting (top left) and vote distribution (top right), as well as the noisy oracle based on majority voting and vote distribution with ordered enhancement (bottom left and right). This selection ensures a balance between the most accurate oracle estimation and a good distribution estimation. Table 4.5 compares the optimal ϵ estimated via the selected methods to the ground truth optimal ϵ. When multiple ϵ configurations perform equally well, the higher one got selected because it tends to be less likely to cause early cut off. Estimated ϵ values that deviate by less or equal than one from the ground truth ϵ are marked bold. The majority vote and the enhanced vote distribution perform best, closely followed by vote distribution, according the number of times they estimated a valuable epsilon. For the datasets CIFAR-10 Low, all ImageNet V2 versions and ImageNet PyTorch all methods work well. Significant differences occur when comparing the estimations the other datasets. Specifically ImageNet, where the most promising method (N-V) predicts a very low epsilon.

	Optimal	N-V	N-VD	E-V-O	E-VD-O
Domain Drift	0.34	0.43	0.47	0.46	0.49
CIFAR-10 Low	0.47	**0.47**	0.45	**0.46**	**0.47**
CIFAR-10 High	0.46	**0.45**	0.41	**0.45**	0.40
ImageNet	0.4	0.35	**0.39**	0.35	**0.39**
ImageNet V2 M-F	0.44	**0.43**	**0.43**	0.41	**0.45**
ImageNet V2 T-0.7	0.42	**0.43**	**0.43**	**0.43**	**0.43**
ImageNet PyTorch	0.43	**0.43**	**0.43**	0.41	0.39
PACS	0.42	**0.43**	**0.42**	0.46	0.39
ImageNet V2 T-I	0.4	**0.41**	**0.41**	0.43	**0.39**
Emotion Detection	0.42	0.45	0.46	0.44	**0.41**
Best found	-	7	6	3	6

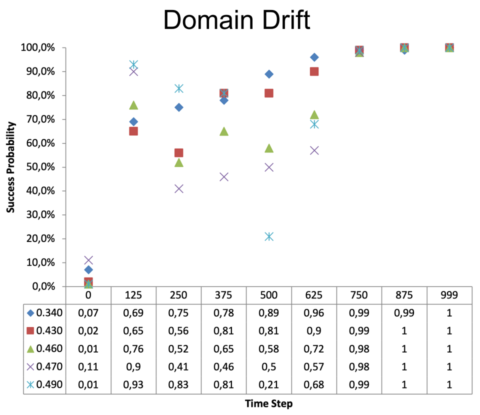
Table 4.5: The table compares the optimal ground truth ϵ values against the ϵ estimated via generating noisy oracles across all datasets. Estimated ϵ values that deviate at max by 1 from the ground truth are marked bold. N: Noisy Oracle, E: Enhanced Noisy Oracle, V: Majority Voting, VD: Vote Distribution, O: Ordered

Using the ϵ values from table 4.5 and the ground truth oracle grid search results from figure 4.2, the effectiveness of the ϵ estimations can be assessed. Figures 4.4 and 4.5 show how the selected ϵ configurations directly compare against the fixed baseline of 0.43, according to their performance on the true oracle grid search.

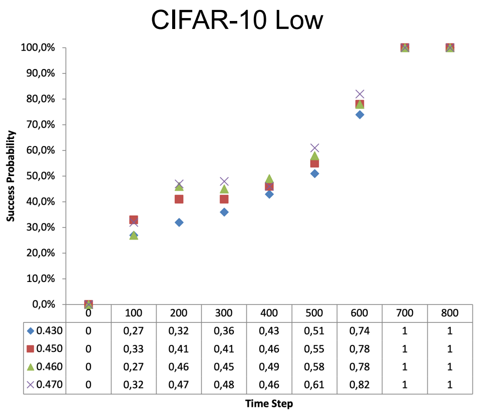
The results in figure 4.4 show mixed performance of the estimation methods. For CIFAR-10 Low all estimation methods perform very well, with N-VD and E-VD-O directly predicting the optimal ϵ, which then also beats the fixed ϵ baseline (blue). For the datasets CIFAR-10 Low, all ImageNet V2 versions and ImageNet PyTorch all methods work well with little deviation. Significant differences occur when comparing the estimations on the other datasets. Specifically ImageNet, where the most promising method (N-V) predicts a very low epsilon, shows a dramatic effect in figure 4.4.

Many optimal ϵ values hover around the 0.43 baseline, making it challenging for the estimators to outperform the baseline. In the scenarios where there is a significant difference between the static baseline and the optimal epsilon figure 4.4 shows some benefit through the estimated ϵ.

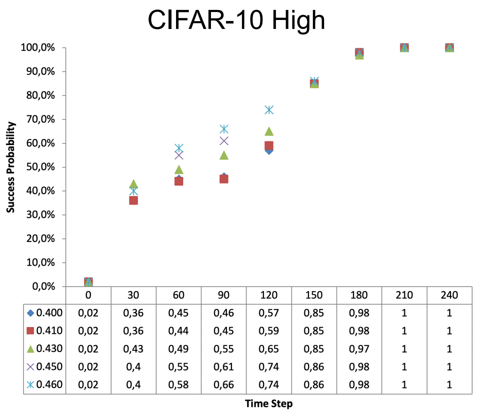
For CIFAR-10 High both majority vote based approaches (N-V and E-V-O) estimate a very good ϵ that outperforms the baseline. The vote distribution based approaches estimate underperforming ϵ values. However, the differences between the top ϵ values in most datasets are minimal, indicating that an estimated ϵ value that is within ± 2 of the optimal one is a reasonably good estimation. This is not the case for datasets such as CIFAR-10, where the performance is greater influenced by little changes in ϵ. Interesting are the ϵ values predicted for ImageNet, where the majority vote methods lead to relatively low ϵ values. These estimates (0.35) perform poorly on the ImageNet dataset according to figure 4.4.



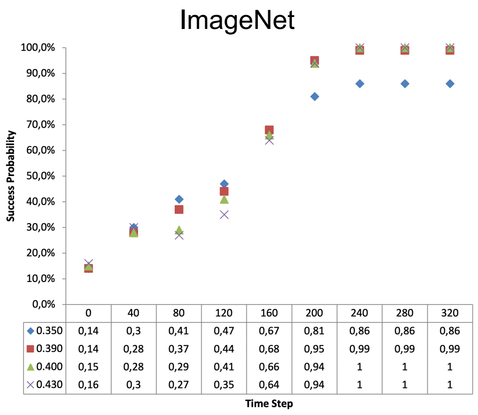
	0	125	250	375	500	625	750	875	999
◆ 0.340	0,07	0,69	0,75	0,78	0,89	0,96	0,99	0,99	1
■ 0.430	0,02	0,65	0,56	0,81	0,81	0,9	0,99	1	1
▲ 0.460	0,01	0,76	0,52	0,65	0,58	0,72	0,98	1	1
× 0.470	0,11	0,9	0,41	0,46	0,5	0,57	0,98	1	1
✳ 0.490	0,01	0,93	0,83	0,81	0,21	0,68	0,99	1	1



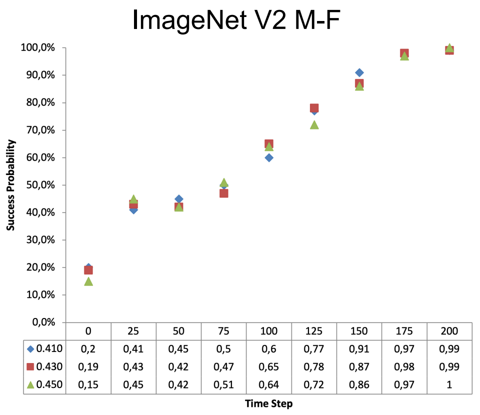
	0	100	200	300	400	500	600	700	800
◆ 0.430	0	0,27	0,32	0,36	0,43	0,51	0,74	1	1
■ 0.450	0	0,33	0,41	0,41	0,46	0,55	0,78	1	1
▲ 0.460	0	0,27	0,46	0,45	0,49	0,58	0,78	1	1
× 0.470	0	0,32	0,47	0,48	0,46	0,61	0,82	1	1



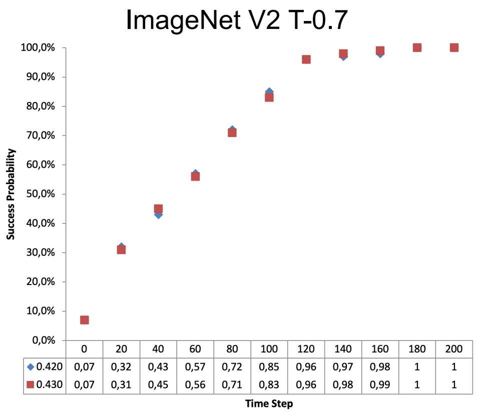
	0	30	60	90	120	150	180	210	240
◆ 0.400	0,02	0,36	0,45	0,46	0,57	0,85	0,98	1	1
■ 0.410	0,02	0,36	0,44	0,45	0,59	0,85	0,98	1	1
▲ 0.430	0,02	0,43	0,49	0,55	0,65	0,85	0,97	1	1
× 0.450	0,02	0,4	0,55	0,61	0,74	0,86	0,98	1	1
✳ 0.460	0,02	0,4	0,58	0,66	0,74	0,86	0,98	1	1



	0	40	80	120	160	200	240	280	320
◆ 0.350	0,14	0,3	0,41	0,47	0,67	0,81	0,86	0,86	0,86
■ 0.390	0,14	0,28	0,37	0,44	0,68	0,95	0,99	0,99	0,99
▲ 0.400	0,15	0,28	0,29	0,41	0,66	0,94	1	1	1
× 0.430	0,16	0,3	0,27	0,35	0,64	0,94	1	1	1

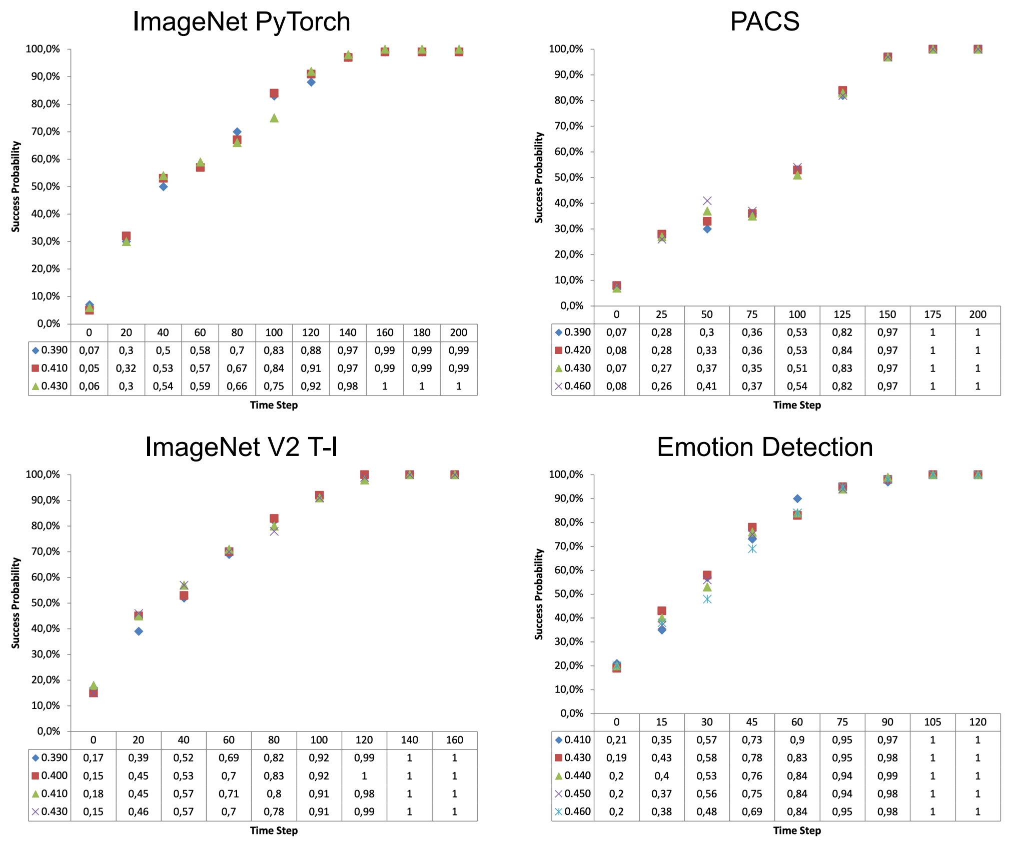


	0	25	50	75	100	125	150	175	200
◆ 0.410	0,2	0,41	0,45	0,5	0,6	0,77	0,91	0,97	0,99
■ 0.430	0,19	0,43	0,42	0,47	0,65	0,78	0,87	0,98	0,99
▲ 0.450	0,15	0,45	0,42	0,51	0,64	0,72	0,86	0,97	1



	0	20	40	60	80	100	120	140	160	180	200
◆ 0.420	0,07	0,32	0,43	0,57	0,72	0,85	0,96	0,97	0,98	1	1
■ 0.430	0,07	0,31	0,45	0,56	0,71	0,83	0,96	0,98	0,99	1	1

Figure 4.4: Noisy oracle estimation evaluation, part 1. The graphs compare the estimated ϵ with the fixed baseline of 0.43 and optimal ϵ for each dataset. It is used to evaluate the performance of the different noisy oracle ϵ estimation methods.



ImageNet PyTorch

	0	20	40	60	80	100	120	140	160	180	200
◆ 0.390	0,07	0,3	0,5	0,58	0,7	0,83	0,88	0,97	0,99	0,99	0,99
■ 0.410	0,05	0,32	0,53	0,57	0,67	0,84	0,91	0,97	0,99	0,99	0,99
▲ 0.430	0,06	0,3	0,54	0,59	0,66	0,75	0,92	0,98	1	1	1

PACS

	0	25	50	75	100	125	150	175	200
◆ 0.390	0,07	0,28	0,3	0,36	0,53	0,82	0,97	1	1
■ 0.420	0,08	0,28	0,33	0,36	0,53	0,84	0,97	1	1
▲ 0.430	0,07	0,27	0,37	0,35	0,51	0,83	0,97	1	1
× 0.460	0,08	0,26	0,41	0,37	0,54	0,82	0,97	1	1

ImageNet V2 T-I

	0	20	40	60	80	100	120	140	160
◆ 0.390	0,17	0,39	0,52	0,69	0,82	0,92	0,99	1	1
■ 0.400	0,15	0,45	0,53	0,7	0,83	0,92	1	1	1
▲ 0.410	0,18	0,45	0,57	0,71	0,8	0,91	0,98	1	1
× 0.430	0,15	0,46	0,57	0,7	0,78	0,91	0,99	1	1

Emotion Detection

	0	15	30	45	60	75	90	105	120
◆ 0.410	0,21	0,35	0,57	0,73	0,9	0,95	0,97	1	1
■ 0.430	0,19	0,43	0,58	0,78	0,83	0,95	0,98	1	1
▲ 0.440	0,2	0,4	0,53	0,76	0,84	0,94	0,99	1	1
× 0.450	0,2	0,37	0,56	0,75	0,84	0,94	0,98	1	1
✳ 0.460	0,2	0,38	0,48	0,69	0,84	0,95	0,98	1	1

Figure 4.5: Noisy oracle estimation evaluation, part 2. The graphs compare the estimated ϵ with the fixed baseline of 0.43 and optimal ϵ for each dataset. It is used to evaluate the performance of the different noisy oracle ϵ estimation methods.

4.4. Method Comparison

The primary result of this thesis is the evaluation of the Model Picker algorithm against selected baselines, presented in this chapter. For each experiment, Model Picker is parameterized according to the optimal epsilon configurations identified with the true oracle and then run alongside the other baselines. The results are shown in figures 4.7-4.8, displaying the success probability and the 90 percentile return accuracy.

The findings show that Model Picker consistently outperforms all other baselines by a significant margin after an initial warm-up phase, with the only exception being the Domain Drift scenario. Table 4.6 highlights the difference in number of samples needed to identify the best model with 90% accuracy. Excluding Domain Drift, the reduction in labels needed when using Model Picker ranges from $2/3$ in CIFAR-10 Low to nearly $1/4$ in ImageNet PyTorch, compared to the runner-up method.

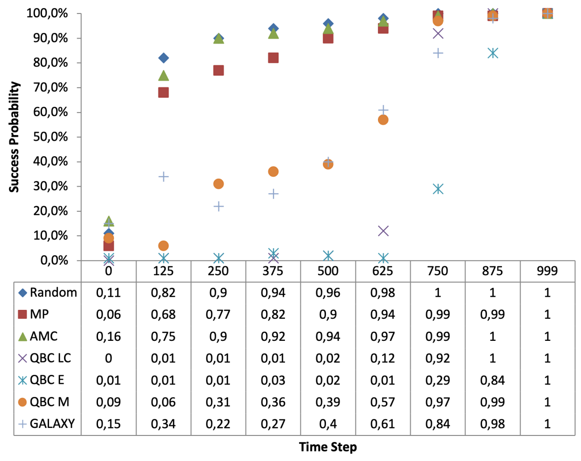
In the Domain Drift setting the Model Picker performs close to Random and Active Model Comparison. This is an important observation as (weighted) disagreement based methods like Model Picker are known to fail under extreme noise conditions. The poor performance of other disagreement based baselines validates this statement.

A notable observation regarding the Model Picker behavior in the success probability graphs is the initialization phase. In most datasets, and particularly in the CIFAR-10 datasets, there exists an initial period where Model Picker performs equally or worse than the baselines. Once this initial phase is over, the performance gap rapidly increases. The length of this initial phase varies and tends to be longer the more noise there is in the dataset. Domain Drift is an exception as it is an extraordinary setting with extreme noise.

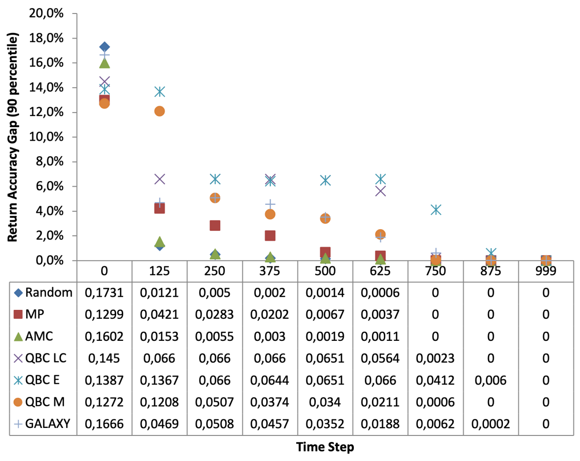
Datasets	Random	Model Picker	ACM	QBC LC	QBC E	QBC M	GALAXY
Domain Drift	**25.6%**	44.7%	65.3%	75.5%	88.6%	68.0%	85.0%
CIFAR-10 Low	91.7%	**63.0%**	84.7%	85.4%	82.5%	86.6%	95.2%
CIFAR-10 High	79.7%	**16.3%**	37.9%	37.5%	46.8%	37.3%	66.3%
ImageNet	82.0%	**19.0%**	59.6%	59.7%	60.0%	60.0%	81.4%
ImageNet V2 M-F	73.2%	**16.2%**	52.2%	53.2%	54.7%	48.4%	73.9%
ImageNet V2 T-0.7	53.3%	**11.6%**	39.7%	42.1%	28.8%	40.6%	65.6%
ImageNet PyTorch	63.6%	**10.9%**	42.5%	42.5%	42.4%	37.4%	63.1%
PACS	48.8%	**12.6%**	41.8%	36.0%	38.2%	38.6%	46.7%
ImageNet V2 T-I	46.2%	**10.4%**	34.4%	34.6%	35.9%	39.8%	48.9%
Emotion Detection	13.1%	**7.3%**	12.3%	14.6%	14.4%	14.6%	14.1%

Table 4.6: The table shows the convergence speed of all tested methods to 90% success probability. The percentage represents the average number of labels needed (max. of 1000) and the winner for each dataset is highlighted bold.

Domain Drift

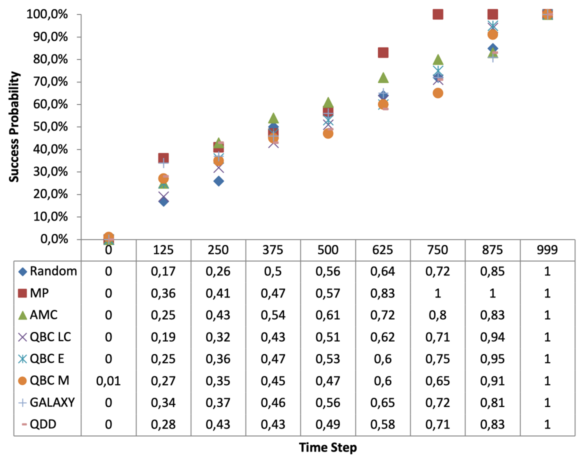


	0	125	250	375	500	625	750	875	999
◆ Random	0,11	0,82	0,9	0,94	0,96	0,98	1	1	1
■ MP	0,06	0,68	0,77	0,82	0,9	0,94	0,99	0,99	1
▲ AMC	0,16	0,75	0,9	0,92	0,94	0,97	0,99	1	1
× QBC LC	0	0,01	0,01	0,01	0,02	0,12	0,92	1	1
✳ QBC E	0,01	0,01	0,01	0,03	0,02	0,01	0,29	0,84	1
● QBC M	0,09	0,06	0,31	0,36	0,39	0,57	0,97	0,99	1
+ GALAXY	0,15	0,34	0,22	0,27	0,4	0,61	0,84	0,98	1

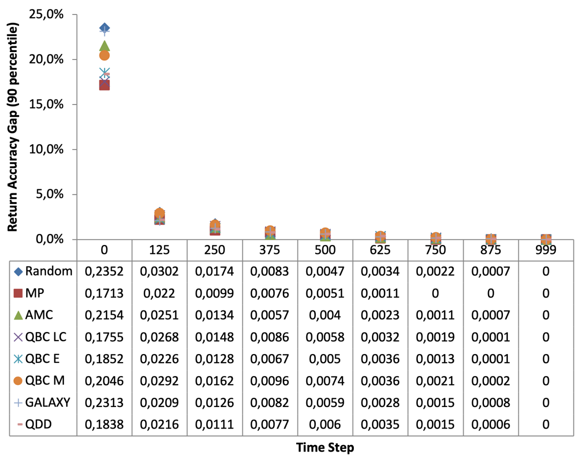


	0	125	250	375	500	625	750	875	999
◆ Random	0,1731	0,0121	0,005	0,002	0,0014	0,0006	0	0	0
■ MP	0,1299	0,0421	0,0283	0,0202	0,0067	0,0037	0	0	0
▲ AMC	0,1602	0,0153	0,0055	0,003	0,0019	0,0011	0	0	0
× QBC LC	0,145	0,066	0,066	0,066	0,0651	0,0564	0,0023	0	0
✳ QBC E	0,1387	0,1367	0,066	0,0644	0,0651	0,066	0,0412	0,006	0
● QBC M	0,1272	0,1208	0,0507	0,0374	0,034	0,0211	0,0006	0	0
+ GALAXY	0,1666	0,0469	0,0508	0,0457	0,0352	0,0188	0,0062	0,0002	0

CIFAR-10 Low



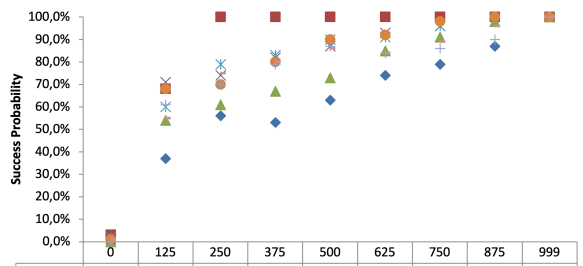
	0	125	250	375	500	625	750	875	999
◆ Random	0	0,17	0,26	0,5	0,56	0,64	0,72	0,85	1
■ MP	0	0,36	0,41	0,47	0,57	0,83	1	1	1
▲ AMC	0	0,25	0,43	0,54	0,61	0,72	0,8	0,83	1
× QBC LC	0	0,19	0,32	0,43	0,51	0,62	0,71	0,94	1
✳ QBC E	0	0,25	0,36	0,47	0,53	0,6	0,75	0,95	1
● QBC M	0,01	0,27	0,35	0,45	0,47	0,6	0,65	0,91	1
+ GALAXY	0	0,34	0,37	0,46	0,56	0,65	0,72	0,81	1
- QDD	0	0,28	0,43	0,43	0,49	0,58	0,71	0,83	1



	0	125	250	375	500	625	750	875	999
◆ Random	0,2352	0,0302	0,0174	0,0083	0,0047	0,0034	0,0022	0,0007	0
■ MP	0,1713	0,022	0,0099	0,0076	0,0051	0,0011	0	0	0
▲ AMC	0,2154	0,0251	0,0134	0,0057	0,004	0,0023	0,0011	0,0007	0
× QBC LC	0,1755	0,0268	0,0148	0,0086	0,0058	0,0032	0,0019	0,0001	0
✳ QBC E	0,1852	0,0226	0,0128	0,0067	0,005	0,0036	0,0013	0,0001	0
● QBC M	0,2046	0,0292	0,0162	0,0096	0,0074	0,0036	0,0021	0,0002	0
+ GALAXY	0,2313	0,0209	0,0126	0,0082	0,0059	0,0028	0,0015	0,0008	0
- QDD	0,1838	0,0216	0,0111	0,0077	0,006	0,0035	0,0015	0,0006	0

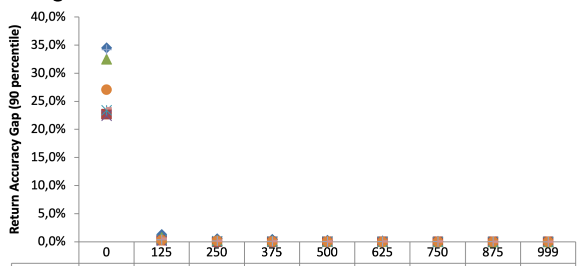
Table 4.7: Method comparison results part 1. The graphs on the left show the success probability of the methods and the graphs on the right present the 90 percentile return accuracy for different datasets and budget sizes.

CIFAR-10 High



	0	125	250	375	500	625	750	875	999
◆ Random	0	0,37	0,56	0,53	0,63	0,74	0,79	0,87	1
■ MP	0,03	0,68	1	1	1	1	1	1	1
▲ AMC	0	0,54	0,61	0,67	0,73	0,85	0,91	0,98	1
× QBC LC	0,01	0,71	0,74	0,82	0,87	0,93	0,96	1	1
✳ QBC E	0,02	0,6	0,79	0,83	0,9	0,91	0,96	1	1
● QBC M	0,01	0,68	0,7	0,8	0,9	0,92	0,98	1	1
+ GALAXY	0	0,61	0,7	0,79	0,87	0,84	0,86	0,9	1
- QDD	0,01	0,55	0,75	0,8	0,86	0,84	0,89	0,96	1

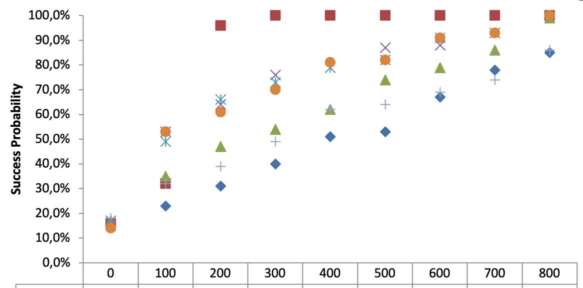
Time Step



	0	125	250	375	500	625	750	875	999
◆ Random	0,3452	0,0126	0,005	0,004	0,0022	0,0013	0,0008	0,0003	0
■ MP	0,2271	0,0027	0	0	0	0	0	0	0
▲ AMC	0,3248	0,0071	0,0027	0,0016	0,001	0,0002	0,0002	0	0
× QBC LC	0,2242	0,0019	0,001	0,0005	0,0002	0	0	0	0
✳ QBC E	0,2338	0,0039	0,0008	0,0004	0,0001	0,0001	0	0	0
● QBC M	0,2705	0,0023	0,0013	0,0005	0,0001	0	0	0	0
+ GALAXY	0,3417	0,0034	0,0013	0,0007	0,0004	0,0003	0,0002	0,0001	0
- QDD	0,2375	0,005	0,0012	0,0006	0,0003	0,0002	0,0001	0	0

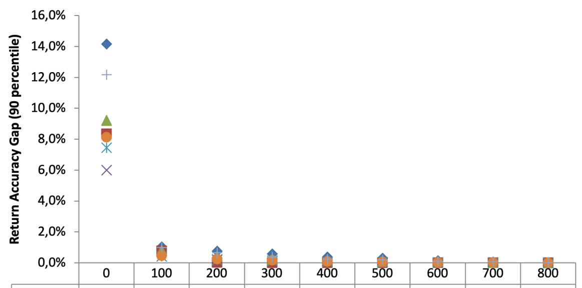
Time Step

ImageNet



	0	100	200	300	400	500	600	700	800
◆ Random	0,15	0,23	0,31	0,4	0,51	0,53	0,67	0,78	0,85
■ MP	0,16	0,32	0,96	1	1	1	1	1	1
▲ AMC	0,17	0,35	0,47	0,54	0,62	0,74	0,79	0,86	0,99
× QBC LC	0,17	0,53	0,64	0,76	0,79	0,87	0,88	0,93	1
✳ QBC E	0,15	0,49	0,66	0,73	0,79	0,82	0,91	0,93	1
● QBC M	0,14	0,53	0,61	0,7	0,81	0,82	0,91	0,93	1
+ GALAXY	0,18	0,32	0,39	0,49	0,62	0,64	0,69	0,74	0,86

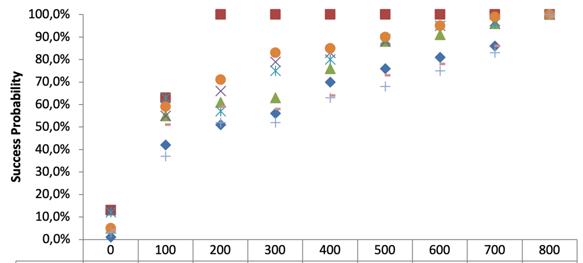
Time Step



	0	100	200	300	400	500	600	700	800
◆ Random	0,1417	0,0103	0,0075	0,0057	0,0037	0,0029	0,0014	0,0006	0,0002
■ MP	0,0835	0,0078	0,0002	0	0	0	0	0	0
▲ AMC	0,0922	0,0079	0,0041	0,0023	0,0018	0,0007	0,0005	0,0002	0
× QBC LC	0,06	0,0042	0,0021	0,0011	0,0008	0,0003	0,0002	0,0001	0
✳ QBC E	0,0744	0,0042	0,0018	0,0014	0,0005	0,0004	0,0002	0,0001	0
● QBC M	0,0812	0,0045	0,0024	0,0016	0,0007	0,0005	0,0001	0,0001	0
+ GALAXY	0,1218	0,0099	0,0065	0,0038	0,0024	0,002	0,001	0,0007	0,0002

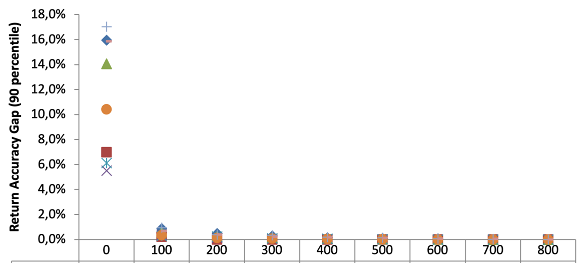
Time Step

ImageNet V2 M-F



	0	100	200	300	400	500	600	700	800
◆ Random	0,01	0,42	0,51	0,56	0,7	0,76	0,81	0,86	1
■ MP	0,13	0,63	1	1	1	1	1	1	1
▲ AMC	0,05	0,55	0,61	0,63	0,76	0,88	0,91	0,96	1
× QBC LC	0,13	0,55	0,66	0,79	0,83	0,88	0,95	0,97	1
✳ QBC E	0,12	0,63	0,57	0,75	0,8	0,89	0,95	0,96	1
● QBC M	0,05	0,59	0,71	0,83	0,85	0,9	0,95	0,99	1
+ GALAXY	0,03	0,37	0,52	0,52	0,63	0,68	0,75	0,83	1
- QDD	0,04	0,51	0,59	0,58	0,64	0,73	0,78	0,86	0,99

Time Step

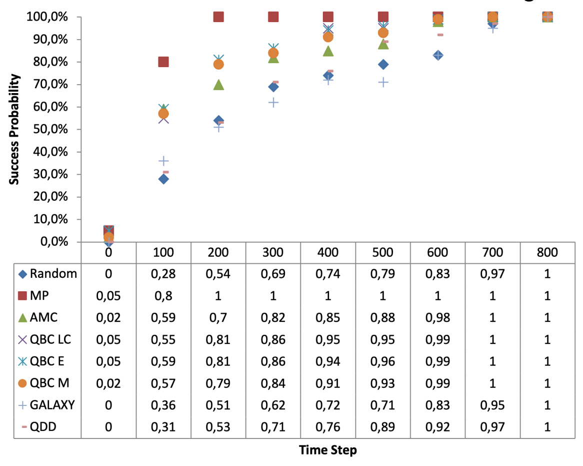


	0	100	200	300	400	500	600	700	800
◆ Random	0,1595	0,0086	0,0048	0,0028	0,0013	0,0009	0,0005	0,0002	0
■ MP	0,0697	0,0022	0	0	0	0	0	0	0
▲ AMC	0,1407	0,0059	0,0025	0,0016	0,0008	0,0003	0,0001	0	0
× QBC LC	0,0551	0,0038	0,0013	0,0007	0,0003	0,0001	0	0	0
✳ QBC E	0,0611	0,0028	0,0023	0,0009	0,0006	0,0001	0	0	0
● QBC M	0,1042	0,0034	0,0012	0,0005	0,0003	0,0001	0	0	0
+ GALAXY	0,1704	0,0094	0,0037	0,0033	0,0018	0,0013	0,0007	0,0002	0
- QDD	0,1584	0,0068	0,0037	0,0021	0,0015	0,001	0,0006	0,0003	0

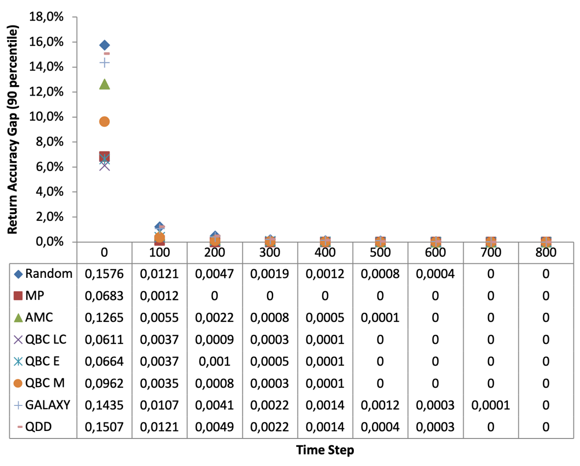
Time Step

Figure 4.6: Method comparison results part 2. The graphs on the left show the success probability of the methods and the graphs on the right present the 90 percentile return accuracy for different datasets and budget sizes.

ImageNet V2 T0.7

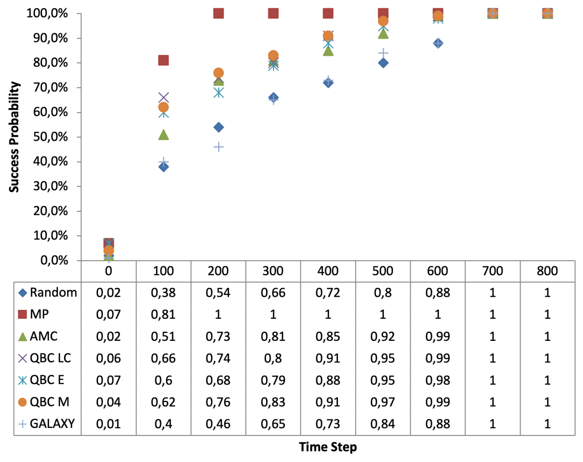


	0	100	200	300	400	500	600	700	800
◆ Random	0	0,28	0,54	0,69	0,74	0,79	0,83	0,97	1
■ MP	0,05	0,8	1	1	1	1	1	1	1
▲ AMC	0,02	0,59	0,7	0,82	0,85	0,88	0,98	1	1
× QBC LC	0,05	0,55	0,81	0,86	0,95	0,95	0,99	1	1
✳ QBC E	0,05	0,59	0,81	0,86	0,94	0,96	0,99	1	1
● QBC M	0,02	0,57	0,79	0,84	0,91	0,93	0,99	1	1
+ GALAXY	0	0,36	0,51	0,62	0,72	0,71	0,83	0,95	1
- QDD	0	0,31	0,53	0,71	0,76	0,89	0,92	0,97	1

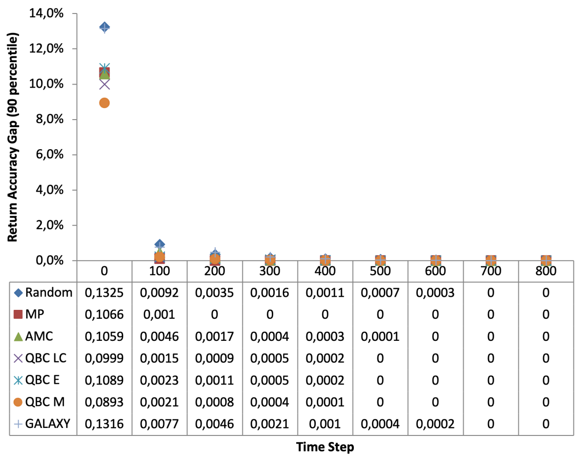


	0	100	200	300	400	500	600	700	800
◆ Random	0,1576	0,0121	0,0047	0,0019	0,0012	0,0008	0,0004	0	0
■ MP	0,0683	0,0012	0	0	0	0	0	0	0
▲ AMC	0,1265	0,0055	0,0022	0,0008	0,0005	0,0001	0	0	0
× QBC LC	0,0611	0,0037	0,0009	0,0003	0,0001	0	0	0	0
✳ QBC E	0,0664	0,0037	0,001	0,0005	0,0001	0	0	0	0
● QBC M	0,0962	0,0035	0,0008	0,0003	0,0001	0	0	0	0
+ GALAXY	0,1435	0,0107	0,0041	0,0022	0,0014	0,0012	0,0003	0,0001	0
- QDD	0,1507	0,0121	0,0049	0,0022	0,0014	0,0004	0,0003	0	0

ImageNet PyTorch

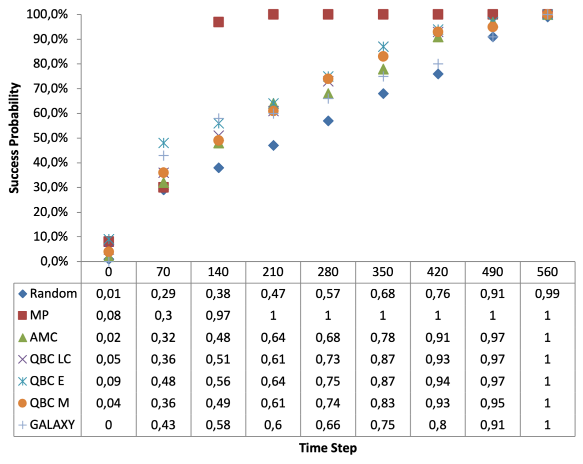


	0	100	200	300	400	500	600	700	800
◆ Random	0,02	0,38	0,54	0,66	0,72	0,8	0,88	1	1
■ MP	0,07	0,81	1	1	1	1	1	1	1
▲ AMC	0,02	0,51	0,73	0,81	0,85	0,92	0,99	1	1
× QBC LC	0,06	0,66	0,74	0,8	0,91	0,95	0,99	1	1
✳ QBC E	0,07	0,6	0,68	0,79	0,88	0,95	0,98	1	1
● QBC M	0,04	0,62	0,76	0,83	0,91	0,97	0,99	1	1
+ GALAXY	0,01	0,4	0,46	0,65	0,73	0,84	0,88	1	1

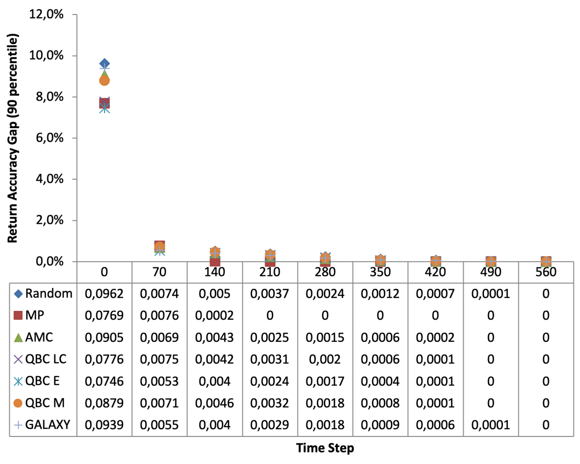


	0	100	200	300	400	500	600	700	800
◆ Random	0,1325	0,0092	0,0035	0,0016	0,0011	0,0007	0,0003	0	0
■ MP	0,1066	0,001	0	0	0	0	0	0	0
▲ AMC	0,1059	0,0046	0,0017	0,0004	0,0003	0,0001	0	0	0
× QBC LC	0,0999	0,0015	0,0009	0,0005	0,0002	0	0	0	0
✳ QBC E	0,1089	0,0023	0,0011	0,0005	0,0002	0	0	0	0
● QBC M	0,0893	0,0021	0,0008	0,0004	0,0001	0	0	0	0
+ GALAXY	0,1316	0,0077	0,0046	0,0021	0,001	0,0004	0,0002	0	0

PACS



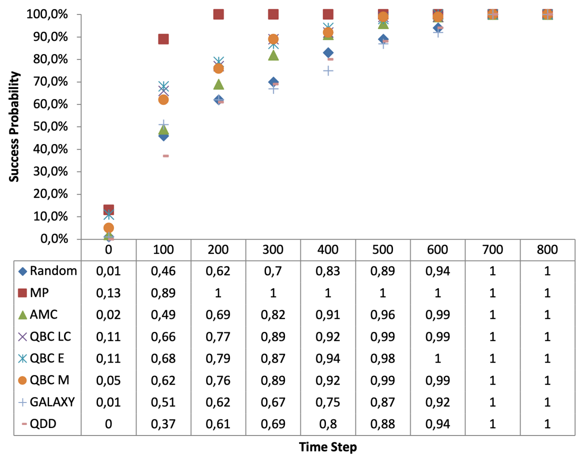
	0	70	140	210	280	350	420	490	560
◆ Random	0,01	0,29	0,38	0,47	0,57	0,68	0,76	0,91	0,99
■ MP	0,08	0,3	0,97	1	1	1	1	1	1
▲ AMC	0,02	0,32	0,48	0,64	0,68	0,78	0,91	0,97	1
× QBC LC	0,05	0,36	0,51	0,61	0,73	0,87	0,93	0,97	1
✳ QBC E	0,09	0,48	0,56	0,64	0,75	0,87	0,94	0,97	1
● QBC M	0,04	0,36	0,49	0,61	0,74	0,83	0,93	0,95	1
+ GALAXY	0	0,43	0,58	0,6	0,66	0,75	0,8	0,91	1



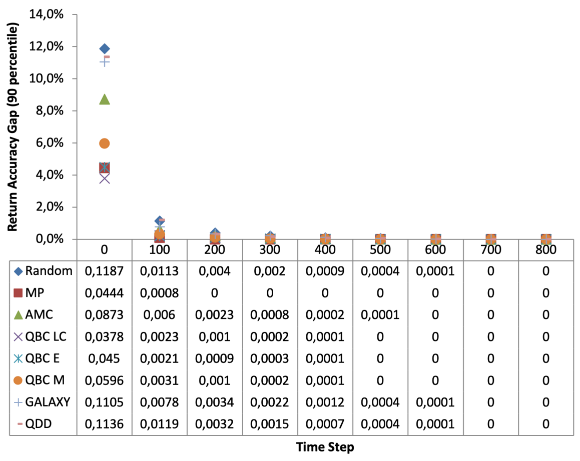
	0	70	140	210	280	350	420	490	560
◆ Random	0,0962	0,0074	0,005	0,0037	0,0024	0,0012	0,0007	0,0001	0
■ MP	0,0769	0,0076	0,0002	0	0	0	0	0	0
▲ AMC	0,0905	0,0069	0,0043	0,0025	0,0015	0,0006	0,0002	0	0
× QBC LC	0,0776	0,0075	0,0042	0,0031	0,002	0,0006	0,0001	0	0
✳ QBC E	0,0746	0,0053	0,004	0,0024	0,0017	0,0004	0,0001	0	0
● QBC M	0,0879	0,0071	0,0046	0,0032	0,0018	0,0008	0,0001	0	0
+ GALAXY	0,0939	0,0055	0,004	0,0029	0,0018	0,0009	0,0006	0,0001	0

Figure 4.7: Method comparison results part 3. The graphs on the left show the success probability of the methods and the graphs on the right present the 90 percentile return accuracy for different datasets and budget sizes.

ImageNet V T-I

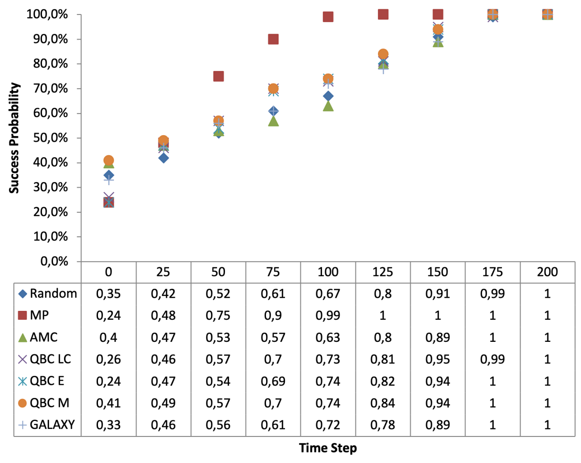


	0	100	200	300	400	500	600	700	800
◆ Random	0,01	0,46	0,62	0,7	0,83	0,89	0,94	1	1
■ MP	0,13	0,89	1	1	1	1	1	1	1
▲ AMC	0,02	0,49	0,69	0,82	0,91	0,96	0,99	1	1
× QBC LC	0,11	0,66	0,77	0,89	0,92	0,99	0,99	1	1
✳ QBC E	0,11	0,68	0,79	0,87	0,94	0,98	1	1	1
● QBC M	0,05	0,62	0,76	0,89	0,92	0,99	0,99	1	1
+ GALAXY	0,01	0,51	0,62	0,67	0,75	0,87	0,92	1	1
- QDD	0	0,37	0,61	0,69	0,8	0,88	0,94	1	1

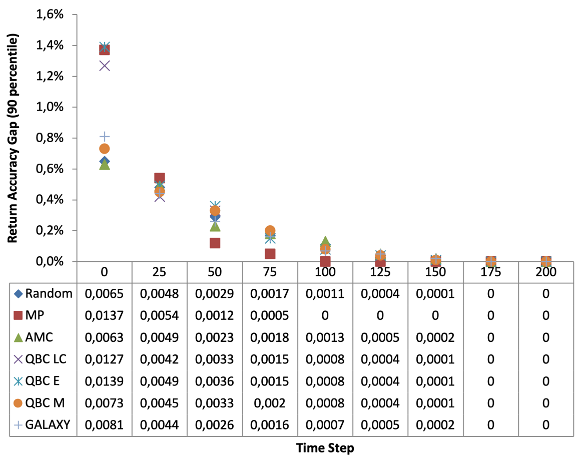


	0	100	200	300	400	500	600	700	800
◆ Random	0,1187	0,0113	0,004	0,002	0,0009	0,0004	0,0001	0	0
■ MP	0,0444	0,0008	0	0	0	0	0	0	0
▲ AMC	0,0873	0,006	0,0023	0,0008	0,0002	0,0001	0	0	0
× QBC LC	0,0378	0,0023	0,001	0,0002	0,0001	0	0	0	0
✳ QBC E	0,045	0,0021	0,0009	0,0003	0,0001	0	0	0	0
● QBC M	0,0596	0,0031	0,001	0,0002	0,0001	0	0	0	0
+ GALAXY	0,1105	0,0078	0,0034	0,0022	0,0012	0,0004	0,0001	0	0
- QDD	0,1136	0,0119	0,0032	0,0015	0,0007	0,0004	0,0001	0	0

Emotion Detection



	0	25	50	75	100	125	150	175	200
◆ Random	0,35	0,42	0,52	0,61	0,67	0,8	0,91	0,99	1
■ MP	0,24	0,48	0,75	0,9	0,99	1	1	1	1
▲ AMC	0,4	0,47	0,53	0,57	0,63	0,8	0,89	1	1
× QBC LC	0,26	0,46	0,57	0,7	0,73	0,81	0,95	0,99	1
✳ QBC E	0,24	0,47	0,54	0,69	0,74	0,82	0,94	1	1
● QBC M	0,41	0,49	0,57	0,7	0,74	0,84	0,94	1	1
+ GALAXY	0,33	0,46	0,56	0,61	0,72	0,78	0,89	1	1



	0	25	50	75	100	125	150	175	200
◆ Random	0,0065	0,0048	0,0029	0,0017	0,0011	0,0004	0,0001	0	0
■ MP	0,0137	0,0054	0,0012	0,0005	0	0	0	0	0
▲ AMC	0,0063	0,0049	0,0023	0,0018	0,0013	0,0005	0,0002	0	0
× QBC LC	0,0127	0,0042	0,0033	0,0015	0,0008	0,0004	0,0001	0	0
✳ QBC E	0,0139	0,0049	0,0036	0,0015	0,0008	0,0004	0,0001	0	0
● QBC M	0,0073	0,0045	0,0033	0,002	0,0008	0,0004	0,0001	0	0
+ GALAXY	0,0081	0,0044	0,0026	0,0016	0,0007	0,0005	0,0002	0	0

Figure 4.8: Method comparison results part 4. The graphs on the left show the success probability of the methods and the graphs on the right present the 90 percentile return accuracy for different datasets and budget sizes.

4.5. Additional Experiments

In this section the results of minor experiments are presented with the goal to explain different observation made in the experiments of the previous chapters. These results become relevant later in the discussion when explaining the behavior of Model Picker.

4.5.1. Distribution Shift

The goal of this chapter is to demonstrate how the model ranking and performance can be influenced by distribution shift.

To investigate this, the PACS dataset was used. It encompasses four distinct domains (art, cartoon, sketch, photo) for the same classification problem. Three different model architectures — ResNet-18, Efficient-Net-B0, and MobileNet-V3 — were trained on ten different dataset distributions. These distributions were created by randomly mixing the training data from the four domains, resulting in training sets of 4000 samples each. Each model was initialized with PyTorch ImageNet weights and then fine tuned for 20 epochs, yielding a total of 30 different models. The test set was created by combining the test data from all four domains. Due to varying numbers of examples available per domain, the test distribution consists of 20% art, 24% cartoon, 39% sketch, and 17% photo. The discrepancy between the training distributions and the test distribution serves as an artificially induced distribution shift. Table 4.8 presents the models' validation (left) and test accuracy (right) scores. The training distributions for the models are detailed in the appendix A.1.

Efficientnet-b0	Mobilenet-v3	Resnet18	Efficientnet-b0	Mobilenet-v3	Resnet18
90.92%	93.25%	98.67%	86.23%	88.65%	92.7%
91.58%	93.17%	98.83%	87.36%	89.43%	93.58%
90.83%	93.67%	98.92%	**87.37%**	**90.29%**	**94.26%**
91.75%	93.83%	98.83%	86.28%	89.26%	92.61%
95.17%	96.08%	**99.75%**	*78.1%*	83.27%	*89.0%*
93.25%	96.5%	99.17%	79.69%	82.85%	87.18%
87.5%	91.25%	98.17%	83.82%	87.65%	92.87%
93.83%	**96.58%**	99.17%	73.03%	*79.75%*	84.8%
90.25%	93.33%	98.83%	85.31%	88.68%	92.86%
86.58%	90.25%	94.33%	86.73%	89.47%	93.82%

Table 4.8: The table highlights the difference in validation accuracy (left) and ranking of three different model architectures trained on ten different data distributions, compared with the test accuracy and ranking on the PACS test set (right). The best performance within a model architecture is marked bold. On the right side, the best model according to the left is marked italic.

Models	V2 T-0.7	V2 M-F	PyTorch	V2 T-I
ViT_H_14	**86.66% (1)**	**81.06% (1)**	**86.98% (1)**	**89.01% (1)**
RegNet_Y_128GF	85.88% (2)	80.39% (2)	86.67% (2)	88.16% (3)
ViT_L_16	85.67% (3)	80.36% (3)	86.52% (3)	88.24% (2)
RegNet_Y_32GF	84.58% (4)	78.24% (4)	85.35% (4)	87.25% (4)
RegNet_Y_16GF	83.75% (5)	77.49% (5)	84.52% (6)	86.83% (5)
RegNet_Y_128GF	83.57% (6)	77.03% (7)	84.54% (5)	86.15% (8)
EfficientNet_V2_L	83.39% (7)	76.77% (8)	84.32% (7)	86.57% (6)
ViT_H_14	83.31% (8)	77.38% (6)	84.17% (8)	86.39% (7)

Table 4.9: The table shows the test accuracy of the PyTorch model collection over all ImageNet V2 test sets and ImageNet validation set. Additionally to the accuracy, the overall ranking of the model is added for comparison. Despite significant differences in accuracy the model ranking only changes marginally.

The training and test scores in table 4.8 indicate that the ResNet-18 model generally outperforms the other architectures. However, a more significant observation is the overall drop in accuracy across all models and change in the best model ranking due to distribution shift. Selecting the model based on the highest training accuracy for testing (ResNet-18 from row five) would result in a loss of approximately 5% accuracy compared to selecting the best model on the test distribution (italic versus bold in the right). This discrepancy is even more extreme for the EfficientNet-B0 and MobileNet-V3 architectures, where the difference there reaches up to approximately 10%.

In addition to the PACS experiment, the model ranking over the different ImageNet datasets was investigated. The creators of the ImageNet V2 test sets noted that, despite their efforts to replicate the ImageNet process, the difference in accuracy between the test sets has to be caused by a distribution shift. The results in table 4.9 show the ranking of the top models across the different ImageNet V2 test sets. While the models' accuracies vary a significantly, which is a sign of distribution shift, the model ranking remains consistent.

4.5.2. Behavior Research

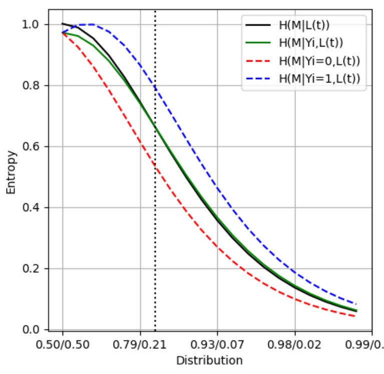


Figure 4.9: This graph illustrates the Model Picker behavior over time assuming one model is always correct (red) and one model is always wrong (blue). The black graph is the entropy of the belief at time t and the green graph is the entropy of the posterior at time t. The dotted vertical line marks the point where rewarding a good model reduces uncertainty less than rewarding a bad model increases it.

	Uniform	Threshold	Confident	Extreme
Agree,3,2,1	0.00071	0.00496	0.00254	-0.00335
Agree,2,1,1	0.0007	-0.0009	-0.00273	-0.00414
Agree,2,2,1	0.0007	0.00423	0.00217	-0.00294
Agree,1,1,1	0.00067	-0.00173	-0.00293	-0.00363
Random,3,2,1	0.01106	0.00312	0.00124	-0.00363
Random,2,1,1	0.0111	-0.00258	-0.00388	-0.00428
Random,2,2,1	0.01094	0.00217	0.00086	-0.00318
Random,1,1,1	0.01098	-0.00343	-0.00414	-0.00381

Table 4.10: The table shows the information gained (difference in uncertainty, before and after) from different data points (rows) given different beliefs (columns). A higher number means a more informative example. The focus hereby lies on the prediction of the top 3 out of 80 models and explaining the impact of different confidence levels in Model Picker. The 3 highest confidences in each column are: Uniform: [...,0.0125,0.0125,0.0125], Threshold: [...,0.07,0.33,0.37], Confident: [...,0.05,0.49,0.28], Extreme: [...,0.02,0.77,0.12]. Predictions are manually chosen to show different disagreement levels. The other models either agree on class 0 or have randomly generated predictions.

Figure 4.9 illustrates a toy example with two models and two classes, showing how the entropy of the prior and posterior develops over time based on the models' rewards. The x-axis represents the evolution of the belief over time, and the y-axis represents the uncertainty. In this examples, all data points have true labels of class 0. The first model always predicts 0, and the second model always predicts 1. The black graph represents the belief $H(\mathrm{M} \mid \mathcal{L}_t)$ at each iteration t. Model Picker estimates the informativeness of a data point over all classes, resulting here in two partial results, one for each class (red=0, blue=1). Aggregating these partial estimations results in the total estimate (green). The green graph represents the expected uncertainty $H(\mathrm{M} \mid Y_i, \mathcal{L}_t)$ calculated as defined in Model Picker. In this handcrafted example, the first model is always correct and thus, after each sampling step, it gets rewarded with a constant factor of 1.5 (epsilon of 0.4). The key observation from this experiment is marked by the vertical dotted line. This line indicates the point where the expected uncertainty of sampling any data point surpasses the uncertainty around the belief at that time.

The results of table 4.10 show this behavior in the context of a real example. Different scenarios are compared with the goal of showing when and why the algorithm converges. Each cell in the table displays the estimated information gain for a data point (row) given some belief (column). The information gain is hereby calculated with $H(\mathrm{M} \mid \mathcal{L}_t) - H(\mathrm{M} \mid Y_i, \mathcal{L}_t)$, representing the estimated difference in uncertainty before and after observing the data point's label. This way of presenting the results is easier to read than showing the absolute estimated uncertainty values ($H(\mathrm{M} \mid Y_i, \mathcal{L}_t)$). The row titles explain what kind of data point is evaluated: Agree,3,2,1 means that the worst 77 models agree on class 0 and the best three models voted 3,2 and 1. Random,3,2,1 means that the worst 77 models votes are generated randomly between 0 and 9 (CIFAR-10 classes) and the top 3 models voted 3,2 and 1. Comparing the upper and lower part of the first column reveals that given uniform belief, the information gain with more disagreement (bottom 4 rows). The belief used in the next column is averaged over all realisations in a real experiment with CIFAR-10 High, at the point of convergence. This resulted in a distribution of $[\ldots, 0.07, 0.33, 0.37]$, with the other 77 models each having a confidence of less than 0.03. This column indicates that, given this weighting, some data points have negative informativeness. These data points have the characteristic that the top two models agree. Furthermore, despite numerical differences between the "rest agreeing" and "rest random" data points, the same negative informativeness is observed. For the last two columns the belief was altered to create an extreme confidence towards one model, leading to the result that in the last column no data point is considered beneficial anymore.

5 Discussion

The discussion starts with a detailed analysis of the behavior observed in the Model Picker algorithm. This is followed by the interpretation and analysis of the ϵ optimization results. Finally, all results are put into context to give insight about their implications for the initial research problem.

5.1. Model Picker Behavior

The Model Picker algorithm clearly outperforms all baseline methods across all datasets (exceptt for Domain Drift) in terms of labels needed to identify the best model with 90% to 100% certainty. In the case of Domain Drift, although Model Picker does not outperform all baselines, it demonstrates significant improvement over the traditional disagreement-based methods, indicating high level of robustness. Given that Domain Drift is an extremely noisy setting, probabilistic sampling methods such as Random and Active Model Comparison (AMC) tend to be superior. In contrast, disagreement-based sampling methods struggle such settings, as the disagreement often stems from high noise rather than being an indication of informativeness. Consequently, relying solely on disagreement can be misleading and results in suboptimal performance. However, with an appropriate ϵ configuration, Model Picker keeps up with the distribution-based sampling methods (Random, AMC). The performance of Model Picker observed in Domain Drift should not be generalized to all datasets, as it only serves to explore the behavior of the algorithm under extreme noise conditions.

Regarding the general behavior, an observation that stands out is that the performance gap between Model Picker and the baselines does not consistently appear from the beginning, but rather develops rapidly at a certain point during the process. This can be explained by the two phases of the Model Picker algorithm. Initially, the algorithm engages in an exploration phase, during which it seeks to identify models that are likely to perform well. In this phase, the algorithm rewards models for correctly predicting data points selected through disagreement-based sampling. Model Picker samples data points solely based on disagreement as long as there is uniform belief across models. The results presented in the first column ("Uniform") of figure 4.10 illustrate the estimated informativeness assigned to different data points. Its evident that higher disagreement correlates with greater estimated information gain. During the exploration phase, extreme confidence in a single models is rare, maintaining a relatively uniform belief. This gradual build-up of confidence occurs because even the best models occasionally make mistakes on the high-disagreement samples. Over time, as the algorithm continues to sample disagreeing instances, poorly performing models are progressively ruled out (losing confidence), while high-performing candidates prevail (=high confidence).

Once a few models perform well consecutively, the algorithm progressively increases confidence in them, leading to greater weights of their predictions. As this confidence grows, the sampling strategy prioritizes samples that can effectively distinguish among these best model candidates. This phase is the exploitation phase. Consequently, the duration of the initial exploration phase, and thus the period during which Model Picker performs similarly to or worse than the baselines, depends on how quickly a good selection of candidate models is identified. The effect of this confidence weighing of the models' predictions can be observed in the second column ("Threshold") of table 4.10, where only data points on which the best models disagree are considered valuable (positive values).

Model Picker converges when there are no more valuable data points in the unlabeled set that can further distinguish the best model candidates. In other words, once there is sufficient high confidence in a small subset of models, there is no data point on which these models disagree. This again can be observed in the results presented in table 4.10. During the exploitation phase, the algorithm focuses on distinguishing between a few high-performing models, resulting in a very limited number of data points where these models disagree. Consequently, once all these instances are sampled, the algorithm is left with data points that may contain some information, but no the information to further distinguish the best models candidates. In table 4.10, that corresponds to the algorithm being left with data points that all have negative values, indicating agreement among the best models. At this point, the algorithm is converged. Due to the estimation of $p(\mathrm{M} = m|Y_i = c|\,\mathcal{L}_t)$ being based on ϵ, the speed of convergence is directly influenced by ϵ as well. A lower ϵ, which corresponds to a higher reward γ leads to more aggressive commitment once good models candidates are identified.

The results also demonstrate the effects of poor parametrization. If ϵ is set too low, it can lead to early convergence in some realisations, causing the success probability to plateau before reaching 100% (see figure 4.2). This occurs because the algorithm reaches a high level of confidence without sampling sufficient evidence. Early in the process, confidence in some models become disproportionately high, leading the algorithm to prioritize only the data points, where these models disagree. Once these few data points are sampled and the confidence in the same models remains high, the algorithm converges prematurely, failing to obtain a comprehensive estimate of the entire model space. The algorithm assumes that it found the correct model and subsequently samples data points that do not change the current belief, resulting in the flat line until the end, visible in some of the grid search results. In a production setting, the algorithm terminates once it reaches this state and returns the model deemed best to this point in time. Consequently, if this early convergence occurs during production, the returned model might not be the optimal one. Therefore, it is crucial to avoid a too low ϵ configuration.

An ϵ that is set too high leads to slow convergence because data points are sampled that are not selected precise enough. In terms of confidence, this means that even in the later stages, the belief is rather uniformly distributed, allowing disagreement among less interesting models to have too much influence on the selection process. While this approach is beneficial at the beginning to obtain a broader estimate and therefore to avoid early convergence, it is detrimental when it comes to accelerating the convergence time. Once the algorithm has identified a selection of good candidate models, it should prioritize sampling data points that help distinguish these. In table A.2, A.3 and A.4 in the appendix, the evolution of belief over time with different ϵ is presented. Comparing these configurations, it is evident that an ϵ of 0.49 has significantly less confidence in the best models in the last steps than the lower ϵ. As discussed previously, the closer the belief is to a uniform distribution, the more Model Picker behaves like disagreement methods, which perform well initially but converge slowly.

Lastly, the phenomenon of "negative information gain" observed in table 4.10 needs to be explained. According to mutual information as defined by Shannon, 1948, information gain cannot be negative. So why does it appear to be the case here? This discrepancy occurs from the method used to estimate the informativeness of a data point across all classes, and the fact it this is only an estimation. A data points expected informativeness is calculated by rewarding models ($p(M = m|Y_i = c, L(t))$) and by calculating the entropy based on the resulting distribution for each class. The estimation over each class means that each model, regardless of what class it predicted, is rewarded exactly once. The graph in figure 4.9 illustrates that there is a threshold where rewarding a high-confidence model decreases uncertainty less than rewarding a low-confidence model increases uncertainty. Since the estimations for each class are summed up, the estimated rest uncertainty can be higher than the uncertainty about the belief at that time. Table 4.10 confirms this in a more realistic setting, as the last column shows that every data point is considered non valuable (negative) due to extreme confidence in one model.

5.2. Hyperparameter Optimization

The results of the ϵ optimization studies confirm the significance of the hyperparameter. Even when different configurations appear similar at first, as observed in datasets like ImageNet and Emotion

Detection, comparing them along the y-axis (success probability) reveals significant variations in performance. Optimally, an ϵ should be selected that achieves a high success probability as quickly as possible. Therefore, optimizing ϵ is crucial due to the limited budget and unknown distribution in the production setting.

The similarity in convergence time across the different ϵ within a good range suggests the existence of a valuable subset of samples that leads to identifying the best model. The grid search results of CIFAR-10 High show this well, as all good ϵ configurations are able to identify this subset, leading to similar convergence times. The difference lies in in the ordering of the data points, which is influenced by ϵ.

Conceptually, ϵ models the prediction noise of the models, providing an indication about how trustworthy the predictions are. It was expected that the optimal ϵ for noisy datasets (e.g. CIFAR-10 Low and Domain Drift) would be higher than for datasets with less noise (e.g. Emotion Detection and ImageNet V2). The results demonstrate some correlation between dataset noisiness and ϵ (see table 4.2), but this correlation is less significant than expected. Interestingly, the Domain Drift dataset suggests the opposite trend: a relatively low ϵ outperforms higher options in a very noisy setting.

This indicates that the optimal ϵ is influenced by additional factors. Due to normalization, the number of models directly impacts the actual reward a model receives (see figure 5.1). Furthermore, it makes a significant difference how the best models perform on the high disagreement samples. If these models continuously make mistakes on the initial samples, the algorithm accumulates these errors by trusting potentially suboptimal models and subsequently sampling based on this misinformation. Although good models will eventually prevail with a sufficient number of samples, regaining confidence in them takes time. The difference in dataset characteristics are best observed when comparing the CIFAR-10 datasets and the ImageNet versions. The disagreement levels in CIFAR-10 High and ImageNet are similar, as reflected by the length of the exploration phase. However, the impact of ϵ in ImageNet is much less compared to CIFAR-10 High. These observations make it challenging to chose an optimal ϵ based on information available beforehand, such as prediction noise and the number of models.

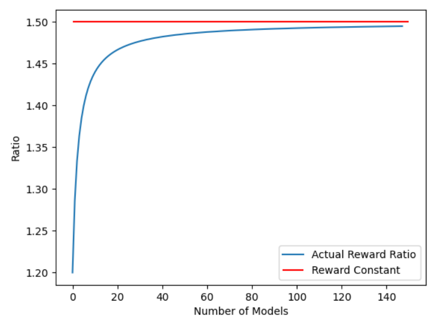


Figure 5.1: The graph shows how much a probability actually increases (y-axis) in a distribution of different sizes (x-axis) when multiplied by a constant factor (red line = 1.5). The actual reward converges towards the constant reward with unlimited distribution size.

Although ϵ significantly impacts the performance of Model Picker, using extra resources to estimate must be justified. To evaluate this, a fixed ϵ of 0.43 was selected as a baseline, as it performs reasonably well across all datasets according to the grid search results in figure 4.2. The experiments for optimizing ϵ were not entirely conclusive. Estimating an optimal ϵ by randomly sampling a small subset of queries proved to be too expensive. The experiments indicate that approximately 30% of labels are required to obtain a reliable estimate for the underlying distribution. This would nearly double the total labeling cost, given that Model Picker typically only needs around 20-30% of labels to identify the best model.

Using noisy oracles for ϵ estimation was expected to be effective, particularly in settings with numerous models and high accuracies, where the resulting noisy oracle should closely approximate the ground truth oracle. Interestingly, the similarity between oracles does not necessarily translate to the quality of the distribution estimate. This can be observed by comparing the results from table 4.3 and 4.4. Noisy oracles created via majority voting appear to be more similar to the ground truth than those created from a vote distribution. Conversely, several cases in table 4.4 show that the ground truth best model was identified more frequently by the less similar noisy oracle. Despite some noisy oracles being able to estimate the true best model directly from the labels with up to 50% (e.g. PACS and Emotion Detection in table 4.4), it is not accurate and reliable enough to be directly used for model selection. The grid search results of different noisy oracles further support this finding as the noisy oracles created with majority voting fail to estimate some of the datasets characteristics. A notable example is shown in figure A.3, where the majority voting oracle fails to identify the lower limit for ϵ, as indicated by early convergence in several of the other graphs.

This can be explained by overfitting. Using majority voting to determine the noisy labels negelects the confidence scores across all classes. This can lead to a loss of information and cause the algorithm to overfit to the noisy consensus instead of the underlying data distribution. This issue becomes more prominent with datasets like ImageNet , which has 1000 classes. More classes introduce more potential disagreement and noise, which is more information that is neglected when relying solely on majority voting. Despite this potential deficiencies of majority voting, the left graphs in figure A.3 show that majority voting results are visually closer to the ground truth results in most variations compared to the graphs on the right, which are generated by sampling from a vote distribution.

Table 4.5 demonstrates that for each tested dataset, at least one estimation method identified the optimal or near optimal ϵ. When selecting the optimal ϵ, there were instances where the differences between the top configurations were marginally small. To address this, the highest ϵ within such a group was chosen to minimize the risk of selecting a too-low ϵ that could cause early convergence. The last row of table 4.5 indicates that majority voting estimates the most settings accurately.

Figure 4.4 shows how well the estimated ϵ values compare to the fixed baseline of 0.43. Some estimated ϵ outperform the static parametrization as they match the optimal ϵ, while others perform worse. Notably, in the CIFAR-10 settings, where the optimal ϵ is significantly higher than 0.43, the estimation methods provide good estimations.

The research on ϵ optimization concludes that learning ϵ can be beneficial in certain cases. Although majority voting performed well in terms of the number of correct estimated ϵ values, its inability to estimate the lower bounds for ϵ is a significant drawback. The results on ImageNet, shown in figure 4.4, illustrate how this limitation can severely impact the performance. Given the variance in the optimal ϵ when repeating experiments on the same dataset, the number of correct ϵ predictions becomes less relevant. More important is whether the estimation method captures the dataset's characteristics as accurately as possible and estimates an ϵ in the right region, even if it is not the most optimal one. Model Picker has shown that it still performs well with ϵ values that are deviate by 0.1 or 0.2 from the optimum as long as it avoids to aggressive rewards. The enhanced vote distribution estimation method best achieves this balance. Although it can be argued that using 50 labels improves the process compared to a fixed ϵ, this is not necessarily justified under the objective of label efficiency. However, using the vote distribution as standalone estimator is not accurate enough. The enhanced vote distribution method demonstrates potential in ϵ estimation. Alternatively, a stepwise fixed ϵ can be used, such as:

$$\epsilon = \begin{cases} 0.45 & \text{if } \kappa \text{ score < 0.6} \\ 0.43 & \text{if } \kappa \text{ score > 0.6} \end{cases} \tag{5.1}$$

where some measure of disagreement, such as Fleiss' Kappa (κ) score, is used to estimate the noisiness of the data.

5.3. Results in Context

The Model Picker algorithm is currently the first method that was explicitly designed for the model selection setting, and it was therefore expected to outperform non-specialized baselines. Adapting more advanced baselines from Active Learning (AL) proved challenging, as most of them rely on information retrieved from a model to train.

Overall, the results show that given the correct parametrization, Model Picker is able to reliably and efficiently identify the best model. This is particularly useful in the Auto-ML setting, as it could provide can potentially reduce the number of retraining cycles with very little labeling cost. Even if the model selection process does not return a model that satisfies the performance requirements, the labeled data points are not wasted. Selected based on disagreement, these data points can still serve as valuable training samples due to their elevated difficulty. Other applications like labeling companies or platforms offering pretrained model collections can also benefit of a reliable and efficient model selection algorithm.

The results indicate that model selection is not always a valuable process. In the case of ImageNet V2, the authors have shown that a distribution shift between training data (ImageNet) and their test data (ImageNet V2 test sets) significantly impacts models performance but has little to no effect on the model ranking. Therefore, it is reasonable to assume that model selection is particularly beneficial when models are trained on different distributions, leading to different strengths and weaknesses. Since all models in the ImageNet comparison case were trained on the same data, this benefit was not observed (see table 4.9). Conversely, the PACS experiment clearly supports the importance of model selection (see figure 4.8), as the models with the highest validation accuracy during training (up to 99%) do not necessarily perform best on different test distributions. In production, the most common cause of retraining is distribution shift in the data. This implies that, despite some uncertainty, the effectiveness of model selection is very likely. Therefore, it is valuable to have an algorithm that can efficiently perform model selection in a model agnostic setting with information about the underlying distribution.

6 Conclusion

This thesis revolved around the development and evaluation of a strategy for model selection of pre-trained classifiers. The primary focus was to introduce a method, named Model Picker, that is able to reliably and efficiently identify the best model given an unknown data distribution and to emphasize the importance of model selection as part of the growing machine learning ecosystem.

The developed framework was designed to be a foundation for assess the performance of future methods across Model Picker and the included baselines. Rigorous experiments demonstrated that Model Picker not only optimizes the use of labeling budget but also maintains robust performance across various settings with different class distributions and noise levels. Especially, the good performance under sever noise is positive outcome as disagreement based methods, such as Model Picker, are known for bad performance in such settings. To establish a reliable benchmark for Model Picker and future methods, the most significant image classification datasets as well as fundamental sampling methods such as Query-by-Committee were employed.

In most scenarios, Model Picker successfully identified the best model with over 90% accuracy while only using 20-30% of possible labels. Compared to the baselines, this represents a fraction of $1/3$ to $1/4$ of the labels needed. However, achieving these performances requires careful hyperparameter tuning according to the test data distribution. This proved be more challenging than anticipated, as the optimal epsilon is dependent on the underlying data distribution as well as other factors. Consequently, obtaining a good epsilon necessitates a good estimate of the data distribution, which is difficult without additional labeling cost. The solution developed to address this generates noisy oracles on which the hyperparameter can be estimated. While the resulting estimate is sufficiently accurate to select a good epsilon, it does not achieve the level of precision necessary for reliable model selection. Although the noisy oracle methods estimated well in some cases, a stepwise fixed epsilon appears to offer more promising results, especially considering label and computation efficiency.

While the results are favorable for the problem settings selected for this thesis, the limitations of this research offer potential for future research. The baselines used for evaluation are well known, but not amongst the most advanced sampling techniques. Given the limited research specifically focused on model selection, incorporating more sophisticated strategies from related fields like Active Learning would result in major modifications to the algorithm. Such modifications would undermine the objective of establishing a fundamental baseline for Model Picker. Although, adapting or recreating such advanced strategies could yield more competitive baselines and a better assessment. Another limitation is the focus on image classification. Expanding the research to large-language-models or regression problems would be a valuable extension, given their widespread use in production.

In conclusion the research shows that model selection can provide valuable insight on relative model performances and the method developed solves the problem in a reliable and efficient manner. With a little extra work, model selection as well as Model Picker can be used in various settings reaching from Auto-ML to pre-trained model providers to efficiently employ the best model.

References

AI Index Steering Committee, S. U. (2024). Ai index report 2024 [Accessed: 2024-07-07]. https://aiindex.stanford.edu/report/2024

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. https://api.semanticscholar.org/CorpusID:64903870

and Chih-Jen Lin Chih-Wei Hsu, C.-C. C. (2008). A practical guide to support vector classification. *BJU international*, *101*.

Ash, J. T., Goel, S., Krishnamurthy, A., & Kakade, S. (2021). Gone fishing: Neural active learning with fisher embeddings. *Advances in Neural Information Processing Systems*, *11*.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., & Agarwal, A. (2020). Deep batch active learning by diverse, uncertain gradient lower bounds. *8th International Conference on Learning Representations, ICLR 2020*.

Baker, B., Gupta, O., Naik, N., & Raskar, R. (2017). Designing neural network architectures using reinforcement learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*.

Chandra, K., Xie, A., Ragan-Kelley, J., & Meijer, E. (2022). Gradient descent: The ultimate optimizer. *Advances in Neural Information Processing Systems*, *35*.

Chen, Y., Hassani, S. H., Karbasi, A., & Krause, A. (2015). Sequential information maximization: When is greedy near-optimal? *Journal of Machine Learning Research*, *40*.

Chitta, K., Alvarez, J. M., Haussmann, E., & Farabet, C. (2022). Training data subset search with ensemble active learning. *IEEE Transactions on Intelligent Transportation Systems*, *23*. https://doi.org/10.1109/TITS.2021.3133268

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, *20*. https://doi.org/10.1177/001316446002000104

Ding, J., Tarokh, V., & Yang, Y. (2018). Model selection techniques: An overview. *IEEE Signal Processing Magazine*, *35*. https://doi.org/10.1109/MSP.2018.2867638

Domhan, T., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. *IJCAI International Joint Conference on Artificial Intelligence*, *2015-January*.

Falkner, S., Klein, A., & Hutter, F. (2018). Bohb: Robust and efficient hyperparameter optimization at scale. *35th International Conference on Machine Learning, ICML 2018*, *4*.

Feurer, M., Letham, B., & Bakshy, E. (2018). Scalable meta-learning for bayesian optimization. *Stat*.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, *14*(771-780), 1612.

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*. https://doi.org/10.1023/a:1007330508534

Hawkins, A. S., Berry, D. A., & Fristedt, B. (1987). Bandit problems-sequential allocation of experiments. *The Statistician*, *36*. https://doi.org/10.2307/2988286

Herzberg, A. M., Wynn, H. P., Fedorov, V. V., Studden, W. J., & Klimko, E. M. (1972). Theory of optimal experiments. *Biometrika*, *59*. https://doi.org/10.2307/2334826

Hesterman, J. Y., Caucci, L., Kupinski, M. A., Barrett, H. H., & Furenlid, L. R. (2010). Maximum-likelihood estimation with a contracting-grid search algorithm. *IEEE Transactions on Nuclear Science*, *57*. https://doi.org/10.1109/TNS.2010.2045898

Hilt, D. E., & Seegrist, D. W. (1977). Ridge: A computer program for calculating ridge regression estimates. https://api.semanticscholar.org/CorpusID:106850190

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282 vol.1. https://doi.org/10.1109/ICDAR.1995.598994

Johnson, R. W. (2001). An introduction to the bootstrap. *Teaching Statistics*, *23*. https://doi.org/10.1111/1467-9639.00050

Joshi, A. J., Porikli, F., & Papanikolopoulos, N. (2009). Multi-class active learning for image classification. *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*. https://doi.org/10.1109/CVPRW.2009.5206627

Kee, S., del Castillo, E., & Runger, G. (2018). Query-by-committee improvement with diversity and density in batch active learning. *Information Sciences*, *454-455*. https://doi.org/10.1016/j.ins.2018.05.014

Klein, A., Falkner, S., Bartels, S., Hennig, P., & Hutter, F. (2017). Fast bayesian optimization of machine learning hyperparameters on large datasets. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*.

Ko, C.-W., Lee, J., & Queyranne, M. (1995). An exact algorithm for maximum entropy sampling. *Operations Research*, *43*. https://doi.org/10.1287/opre.43.4.684

Kossen, J., Farquhar, S., Gal, Y., & Rainforth, T. (2021). Active testing: Sample-efficient model evaluation. *Proceedings of Machine Learning Research*, *139*.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *… Science Department, University of Toronto, Tech. …* https://doi.org/10.1.1.222.9220

Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. M. (2017). Deeper, broader and artier domain generalization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, *18*.

Liu, H., Simonyan, K., & Yang, Y. (2019). Darts: Differentiable architecture search. *7th International Conference on Learning Representations, ICLR 2019*.

Madani, O., Lizotte, D. J., & Greiner, R. (2004). Active model selection. *Strategies*.

Mallows, C. L. (1973). Some comments on cp. *Technometrics*, *15*(4), 661–675. Retrieved July 6, 2024, from http://www.jstor.org/stable/1267380

Margatina, K., Vernikos, G., Barrault, L., & Aletras, N. (2021). Active learning by acquiring contrastive examples. *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*. https://doi.org/10.18653/v1/2021.emnlp-main.51

McCallum, A., & Nigam, K. (1998). Employing em and pool-based active learning for text classification. *Proceedings of the Fifteenth International Conference on Machine Learning*.

N., R., & A., M. (2001). Toward optimal active learning through sampling estimation of error reduction. *Proceedings of 18th International Conference on Machine Learning, ICML*.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pavlyshenko, B. (2018). Using stacking approaches for machine learning models. *2018 IEEE second international conference on data stream mining & processing (DSMP)*, 255–258.

Piironen, J., & Vehtari, A. (2017). Comparison of bayesian predictive methods for model selection. *Statistics and Computing*, *27*. https://doi.org/10.1007/s11222-016-9649-y

Pillai, A. (2020). Emotion detection fer dataset [Accessed: 2024-05-29].

PyTorch. (2024). Pytorch: An imperative style, high-performance deep learning library [Accessed: 2024-05-09]. https://pytorch.org

Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J., Wu, S., & Ré, C. (2020). Snorkel: Rapid training data creation with weak supervision. *VLDB Journal*, *29*. https://doi.org/10.1007/s00778-019-00552-1

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y. L., Tan, J., Le, Q. V., & Kurakin, A. (2017). Large-scale evolution of image classifiers. *34th International Conference on Machine Learning, ICML 2017*, *6*.

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2019). Do imagenet classifiers generalize to imagenet? *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, *2019-June*.

Ren, P., Xiao, Y., Chang, X., Huang, P. Y., Li, Z., Gupta, B. B., Chen, X., & Wang, X. (2022). A survey of deep active learning. https://doi.org/10.1145/3472291

Sawade, C., Landwehr, N., & Scheffer, T. (2012). Active comparison of prediction models. *Advances in Neural Information Processing Systems*, *3*.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*(2), 461–464. Retrieved July 6, 2024, from http://www.jstor.org/stable/2958889

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, *2015-January*.

Sener, O., & Savarese, S. (2018). Active learning for convolutional neural networks: A core-set approach. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Settles, B. (2010). Active learning literature survey. *Machine Learning*, *15*. https://doi.org/10.1.1.167.4245

Settles, B., & Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks. *EMNLP 2008 - 2008 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference: A Meeting of SIGDAT, a Special Interest Group of the ACL*. https://doi.org/10.3115/1613715.1613855

Settles, B., Craven, M., & Ray, S. (2007). Multiple-instance active learning. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems* (Vol. 20). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2007/file/a1519de5b5d44b31a01de013b9b51a80-Paper.pdf

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*. https://doi.org/10.1145/130385.130417

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, *27*. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, *2003-January*. https://doi.org/10.1109/ICDAR.2003.1227801

Smith, F. B., Kirsch, A., Farquhar, S., Gal, Y., Foster, A., & Rainforth, T. (2023). Prediction-oriented bayesian active learning. *Proceedings of Machine Learning Research*, *206*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, *4*.

Snoek, J., Ripped, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M. M. A., Prabhat, & Adams, R. P. (2015). Scalable bayesian optimization using deep neural networks. *32nd International Conference on Machine Learning, ICML 2015*, *3*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*. https://doi.org/10.1007/s10994-019-05855-6

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, *1995-June*. https://doi.org/10.3115/981658.981684

Zhang, J., Chen, Y., Canal, G., Mussmann, S., Das, A. M., Bhatt, G., Zhu, Y., Bilmes, J., Du, S. S., Jamieson, K., & Nowak, R. D. (2024). Labelbench: A comprehensive framework for benchmarking adaptive label-efficient learning.

Zhang, J., Katz-Samuels, J., & Nowak, R. (2022). Galaxy: Graph-based active learning at the extreme. *Proceedings of Machine Learning Research*, *162*.

Zhou, Z.-H. (2021). Model selection and evaluation. In *Machine learning* (pp. 25–55). Springer Singapore. https://doi.org/10.1007/978-981-15-1967-3_2

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, *67*. https://doi.org/10.1111/j.1467-9868.2005.00503.x

A

Appendix

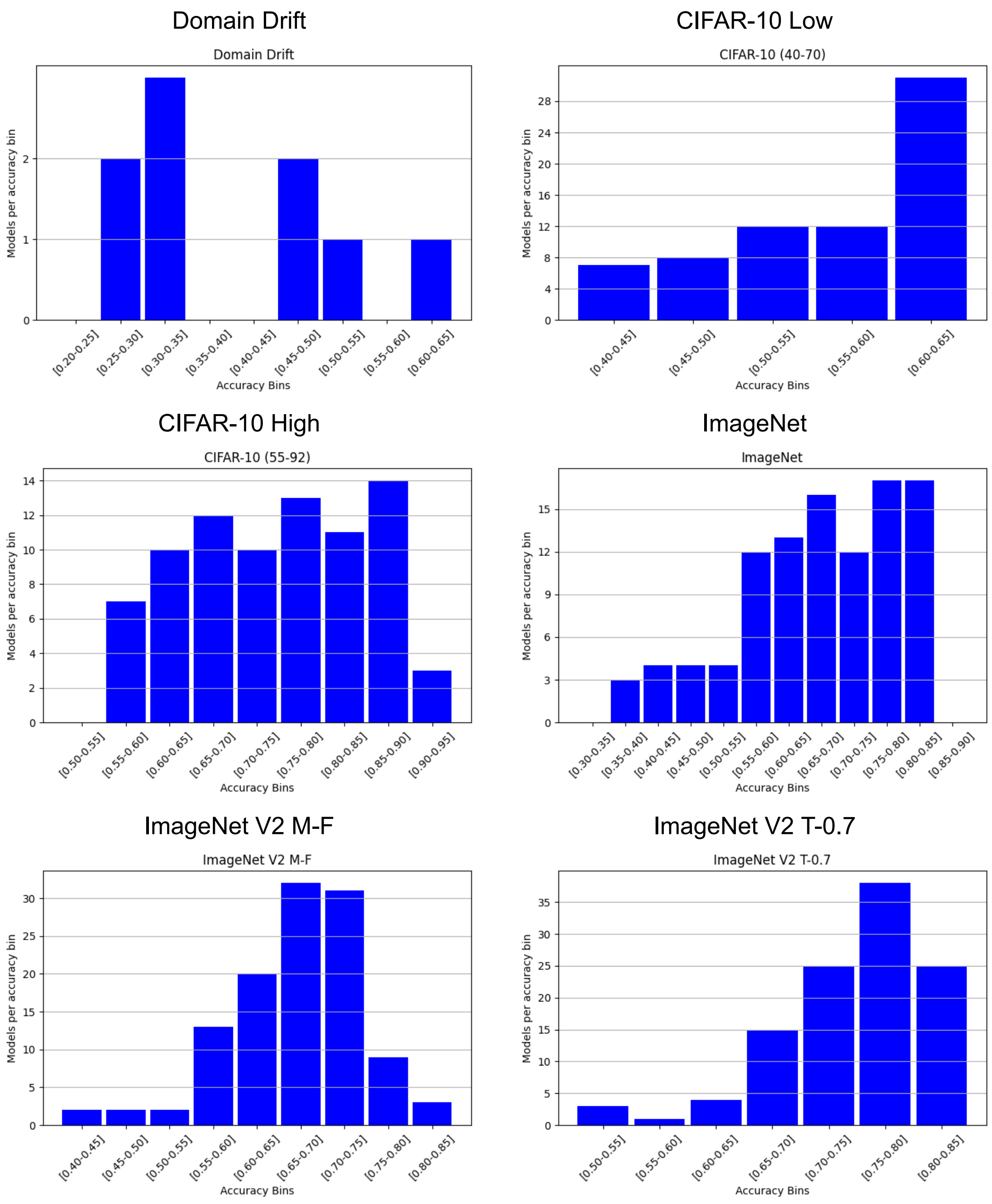


Figure A.1: Dataset model accuracy distribution part 1. The histograms show the distribution of models per accuracy bracket. More models in the highest blocks indicate a more competitive model selection setup.

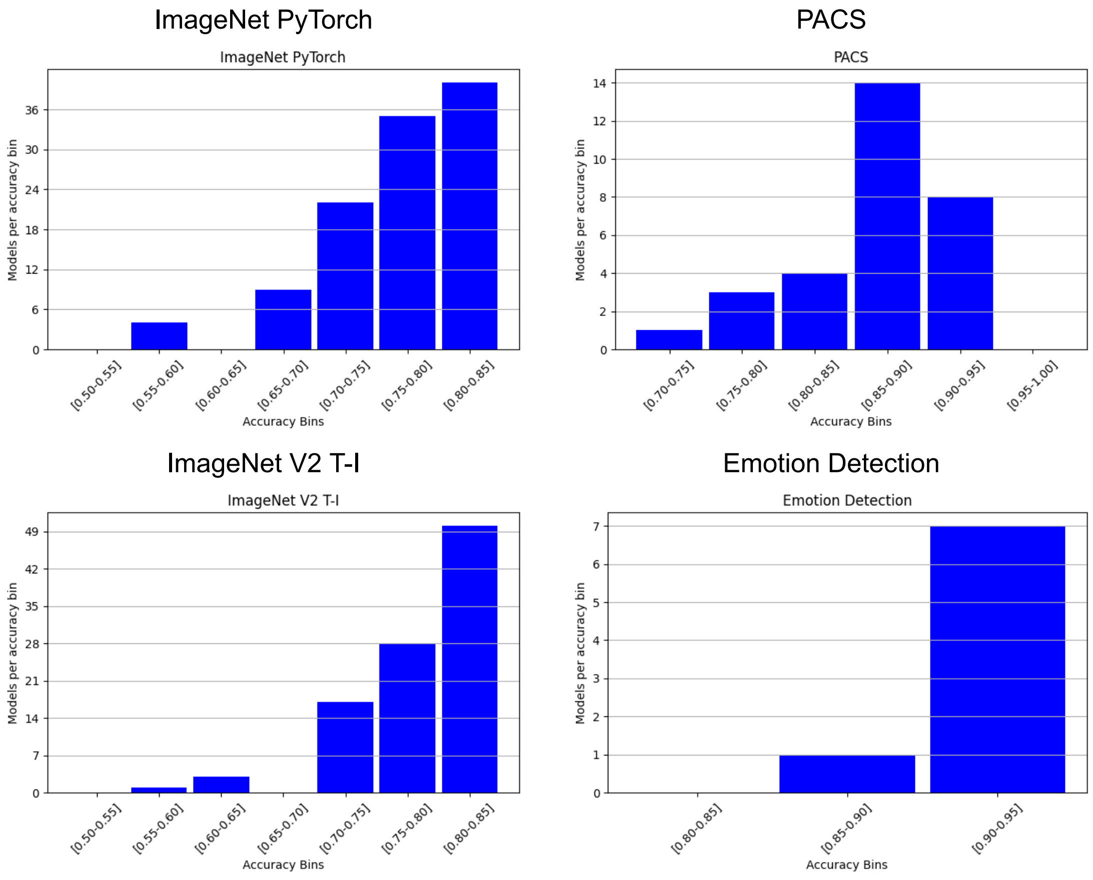


Figure A.2: Dataset model accuracy distribution part 2. The histograms show the distribution of models per accuracy bracket. More models in the highest blocks indicate a more competitive model selection setup.

	Art	Cartoon	Photo	Sketch
Distribution 0	12%	27%	30%	31%
Distribution 1	23%	30%	25%	22%
Distribution 2	28%	34%	31%	8%
Distribution 3	28%	18%	18%	35%
Distribution 4	44%	7%	4%	45%
Distribution 5	68%	19%	2%	10%
Distribution 6	6%	37%	51%	6%
Distribution 7	59%	4%	4%	33%
Distribution 8	28%	5%	45%	21%
Distribution 9	26%	39%	34%	1%

Table A.1: The table shows the randomly generated data distributions used for training the different PACS models. Each of the 3 model architectures was trained on each distribution resulting in the 30 models.

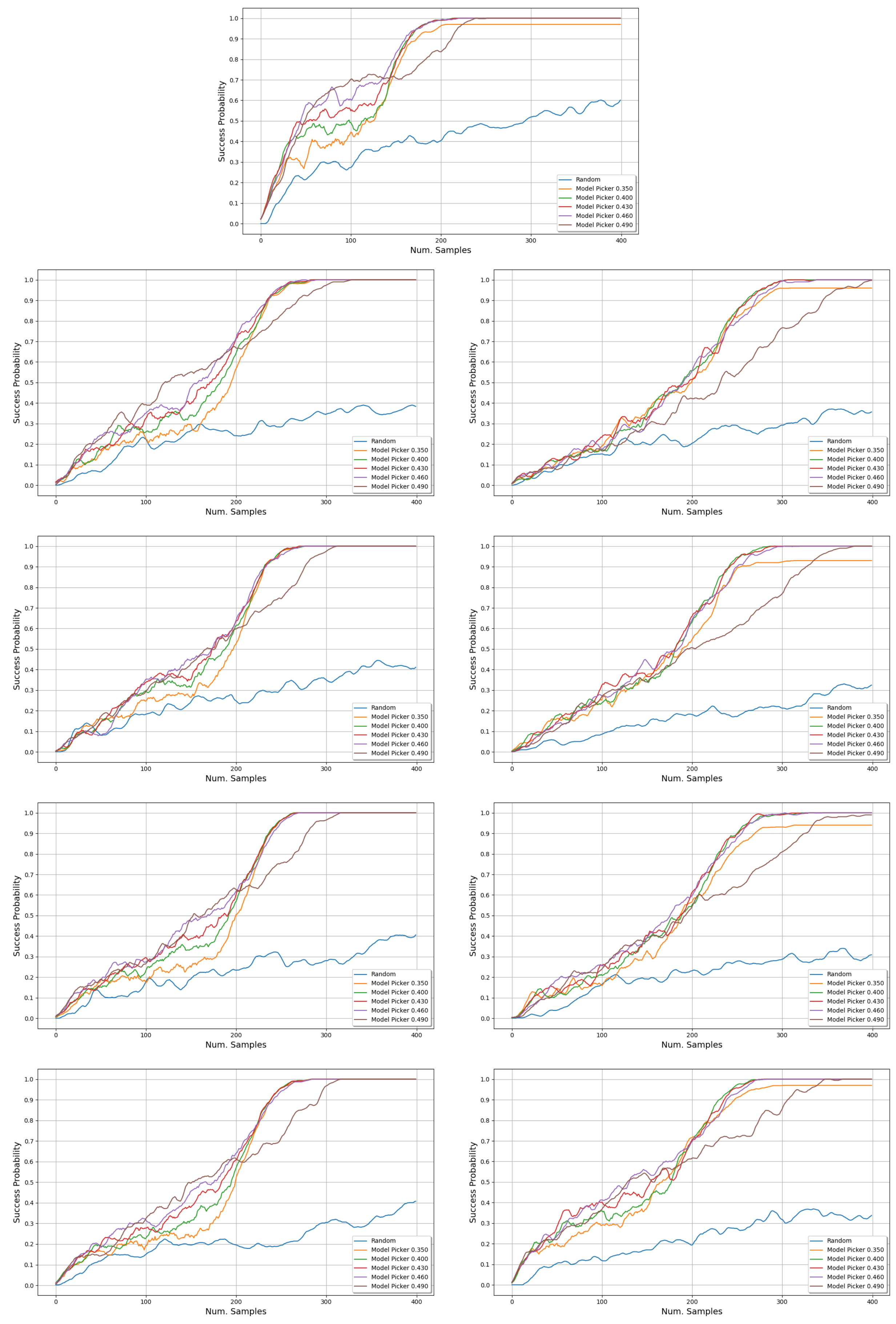


Figure A.3: The graphs show grid search results of different noisy oracle hyperparameter estimations based on CIFAR-10 High. On top is the original result using the ground truth oracle. The left column displays majority voting and the right column uses vote distribution. The vertical order is: pure noisy oracle, randomly enhanced noisy oracle, weighted enhanced noisy oracles and ordered noisy oracle. The focuses lies on visual similarity between the estimates and the ground truth at the top.

t	10	9	8	7	6	5	4	3	2	1
10	0.0182	0.0225	0.0259	0.0277	0.0386	0.0409	0.0306	0.0368	0.0415	0.0643
30	0.0258	0.0306	0.0466	0.0314	0.0377	0.0536	0.0483	0.0521	0.1057	0.1017
50	0.0283	0.0295	0.0469	0.0393	0.0343	0.0531	0.0489	0.0703	0.1603	0.1448
70	0.0272	0.0303	0.0604	0.0437	0.0419	0.0493	0.0453	0.0822	0.1851	0.1663
90	0.0257	0.0328	0.0541	0.0404	0.0389	0.0561	0.057	0.0856	0.2023	0.1922
110	0.0172	0.0291	0.0509	0.0345	0.036	0.0546	0.0479	0.081	0.2311	0.2402
130	0.0194	0.0246	0.0516	0.0259	0.0299	0.0446	0.0422	0.0723	0.2481	0.3034
150	0.0149	0.0234	0.0409	0.0232	0.0241	0.0418	0.0317	0.0765	0.2448	0.3561
170	0.013	0.0176	0.0375	0.0197	0.0242	0.0333	0.0285	0.0755	0.2638	0.378
190	0.0117	0.0159	0.0345	0.0173	0.0216	0.0308	0.0251	0.0693	0.2836	0.3905
210	0.0119	0.0147	0.0299	0.0174	0.0184	0.0291	0.0245	0.0624	0.2946	0.4019
230	0.0119	0.0147	0.0299	0.0174	0.0184	0.0291	0.0245	0.0624	0.2947	0.402
250	0.0119	0.0147	0.0298	0.0174	0.0184	0.0291	0.0245	0.0624	0.2947	0.4022

Table A.2: This table shows the average posterior development over time on the example of CIFAR-10 High experiment with an epsilon of 0.3. Each row represents the confidence values for the top 10 models at the sample time t.

t	10	9	8	7	6	5	4	3	2	1
10	0.0186	0.0194	0.0245	0.022	0.0278	0.0285	0.0234	0.0223	0.0251	0.0358
30	0.025	0.0265	0.0387	0.0257	0.0275	0.0352	0.0358	0.0391	0.0697	0.0809
50	0.0315	0.033	0.0493	0.0319	0.0338	0.0453	0.0449	0.0574	0.1001	0.0987
70	0.0326	0.0352	0.0587	0.0344	0.0388	0.0529	0.0495	0.0738	0.1209	0.1205
90	0.0342	0.0385	0.0608	0.0369	0.0416	0.0569	0.0532	0.0885	0.1328	0.137
110	0.0327	0.0397	0.0637	0.0312	0.0416	0.0588	0.0508	0.0958	0.1416	0.1599
130	0.0341	0.0384	0.0627	0.0301	0.041	0.0609	0.0478	0.0951	0.1461	0.2007
150	0.0321	0.035	0.0607	0.0271	0.0398	0.0548	0.0456	0.092	0.1656	0.2439
170	0.025	0.0297	0.0462	0.0266	0.03	0.0453	0.0379	0.0801	0.202	0.3048
190	0.0221	0.0254	0.0362	0.024	0.0281	0.0387	0.0322	0.0744	0.2188	0.3432
210	0.0215	0.0246	0.0344	0.0224	0.026	0.0339	0.0308	0.0733	0.2274	0.3524
230	0.0216	0.0246	0.0331	0.0218	0.0255	0.0333	0.0304	0.0739	0.2293	0.3535
250	0.0216	0.0246	0.0331	0.0218	0.0255	0.0333	0.0304	0.0739	0.2293	0.3535

Table A.3: This table shows the average posterior development over time on the example of CIFAR-10 High experiment with an epsilon of 0.4. Each row represents the confidence values for the top 10 models at the sample time t.

t	10	9	8	7	6	5	4	3	2	1
10	0.0133	0.0134	0.0136	0.0134	0.0139	0.0142	0.0135	0.0139	0.0135	0.0142
30	0.0148	0.0149	0.0161	0.0157	0.0162	0.017	0.0151	0.0166	0.017	0.0186
50	0.0163	0.0161	0.0184	0.0176	0.0176	0.0195	0.0172	0.0203	0.0208	0.0236
70	0.0177	0.0175	0.0206	0.0197	0.0196	0.0217	0.0192	0.0243	0.0258	0.0293
90	0.0185	0.0184	0.0235	0.0221	0.0215	0.0241	0.021	0.0285	0.0312	0.0348
110	0.0195	0.0194	0.0265	0.0234	0.0235	0.0269	0.0226	0.0326	0.0363	0.0407
130	0.0199	0.0198	0.0297	0.0244	0.0251	0.0299	0.0238	0.0378	0.0415	0.0475
150	0.0208	0.0208	0.0312	0.0249	0.0265	0.0308	0.0252	0.043	0.046	0.0531
170	0.0215	0.0217	0.0332	0.0266	0.0282	0.0327	0.0271	0.0476	0.0517	0.0586
190	0.0225	0.0227	0.0352	0.0285	0.0301	0.0341	0.0298	0.0521	0.0575	0.0651
210	0.0236	0.0235	0.0366	0.03	0.0314	0.0359	0.0323	0.0555	0.0648	0.0732
230	0.0247	0.0247	0.0379	0.0317	0.0324	0.0374	0.0344	0.0582	0.0723	0.0829
250	0.0254	0.0256	0.0379	0.0329	0.033	0.0382	0.0361	0.0607	0.0818	0.0944

Table A.4: This table shows the average posterior development over time on the example of CIFAR-10 High experiment with an epsilon of 0.49. Each row represents the confidence values for the top 10 models at the sample time t.

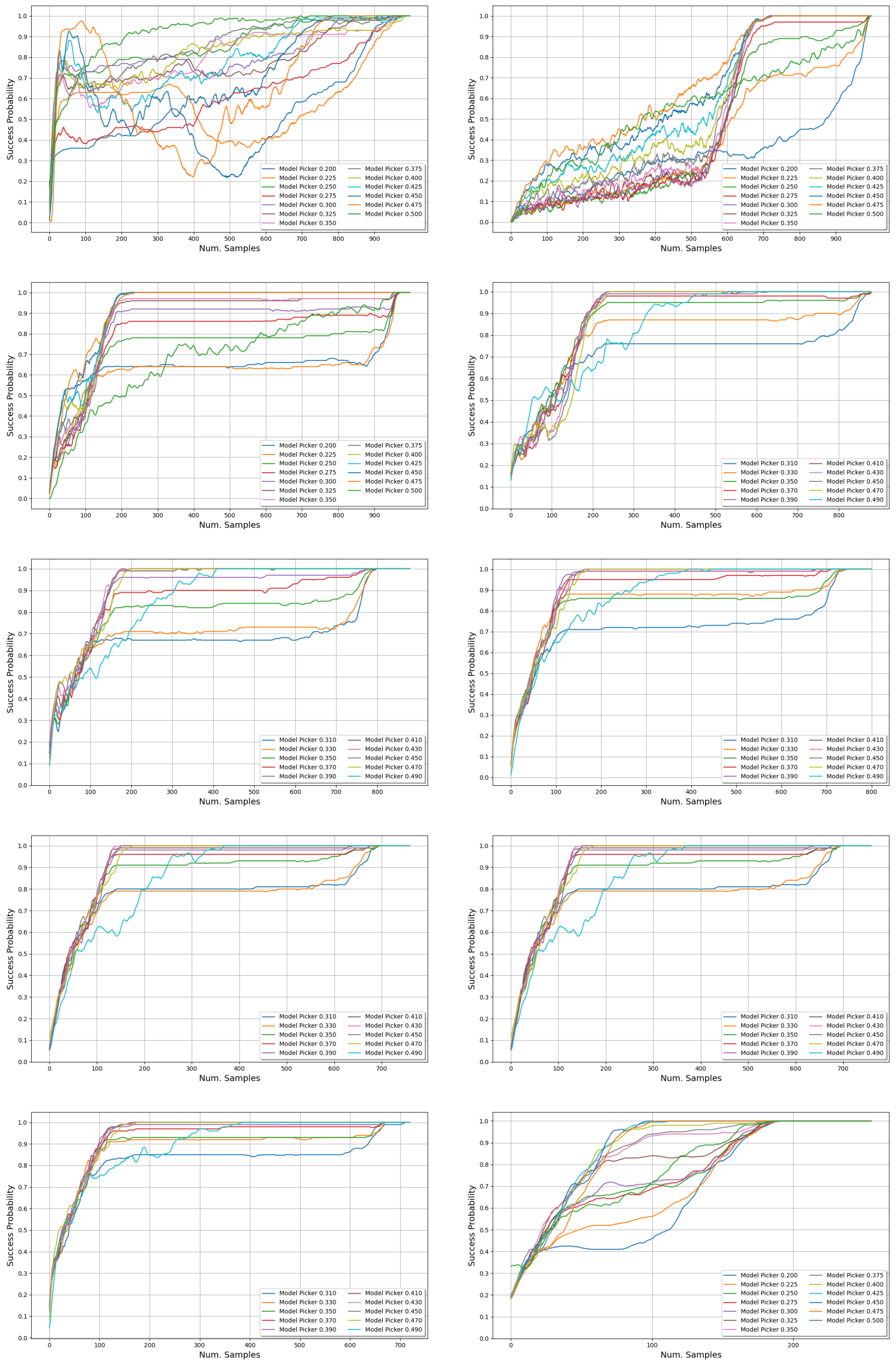


Figure A.4: The graphs show the success probability of the wide grid search results on all datasets including all configurations.